Red Hat Fuse 7.13

Apache Camel 组件参考

Camel 组件的配置参考

Last Updated: 2024-06-18

Red Hat Fuse 7.13 Apache Camel 组件参考

Camel 组件的配置参考

法律通告

摘要

Apache Camel 有超过 100 个组件，每个组件都是高度可配置的。本指南描述了每个组件的设置。

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。这些更改将在即将发行的几个发行本中逐渐实施。详情请查看我们的 CTO Chris Wright 信息。

第 1 章 组件概述

本章介绍了可用于 Apache Camel 的所有组件。

1.1. 容器类型

Red Hat Fuse 提供各种容器类型，您可在其中部署 Camel 应用程序：

- Spring Boot
- Apache Karaf
- JBoss Enterprise Application Platform (JBoss EAP)

此外，Camel 应用程序可以作为 *无容器运行* ：也就是说，Camel 应用程序直接在 JVM 中运行，而无需任何特殊容器。

在某些情况下，Fuse 可能会在一个容器中支持 Camel 组件，但不支持其他容器。这样做有多种原因，但在有些情况下，一个组件并不适用于所有容器类型。例如，**camel-ejb** 组件是为 Java EE （即 JBoss EAP）而设计的，且在其他容器类型中不支持。

注意

camel-test 组件和扩展组件（如 **camel-test-blueprint**、**camel-test-karaf**、和 **camel-test-spring** ）被支持，并可用于为每个运行时运行 JUnit 测试。但是，这些组件不会在运行时本身执行，而是在 JUnit 中执行。

1.2. 支持的组件

请注意以下键：

符号	描述
✔	支持
▮	不支持或还不支持
已弃用	以后的发行版本中可能会删除

表 1.1 “Apache Camel 组件支持列表” 提供有关哪些容器支持哪些 Camel 组件的综合详情。

表 1.1. Apache Camel 组件支持列表

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP	IBM Power / Spring Boot 2	IBM Z / Spring Boot 2
activemq-camel	✔	✔	✔	✔	✔	✔
activemq-http	✔	▮	▮	✔	▮	▮

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP	IBM Power / Spring Boot 2	IBM Z / Spring Boot 2
camel-ahc	✔	✔	✔	✔	▮	▮
camel-ahc-ws	✔	✔	✔	✔	▮	▮
camel-ahc-wss	✔	✔	✔	✔	▮	▮
camel-amqp	✔	✔	✔	✔	✔	✔
camel-apns	✔	✔	✔	✔	▮	▮
camel-as2	▮	✔	✔	▮	▮	▮
camel-asterisk	✔	✔	✔	✔	▮	▮
camel-atmos	✔	✔	▮	▮	▮	▮
camel-atmosphere-websocket	✔	✔	✔	✔	▮	▮
camel-atom	✔	✔	✔	✔	▮	▮
camel-atomix	✔	✔	✔	✔	▮	▮
camel-avro	✔	✔	✔	✔	✔	✔
camel-aws	✔	✔	✔	✔	▮	▮
camel-azure	✔	✔	✔	✔	▮	▮
camel-bam	*已弃用*	*已弃用*	*已弃用*	▮	▮	▮
camel-bean	✔	✔	✔	✔	▮	▮
camel-bean-validator	✔	✔	✔	✔	▮	▮
camel-beanstalk	✔	✔	✔	▮	▮	▮
camel-binding	*已弃用*	*已弃用*	*已弃用*	✔	▮	▮
camel-blueprint	✔	▮	✔	▮	▮	▮

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP	IBM Power / Spring Boot 2	IBM Z / Spring Boot 2
camel-bonita	✔	▮	▮	▮	▮	▮
camel-box	✔	✔	✔	✔	▮	▮
camel-braintree	✔	✔	✔	✔	▮	▮
camel-browse	✔	✔	✔	✔	▮	▮
camel-cache	*已弃用*	*已弃用*	*已弃用*	▮	▮	▮
camel-caffeine	✔	✔	✔	✔	▮	▮
camel-cdi	✔	▮	*已弃用*	✔	▮	▮
camel-chronicle-engine	✔	✔	✔	✔	▮	▮
camel-chunk	✔	✔	✔	✔	▮	▮
camel-class	✔	✔	✔	✔	▮	▮
camel-cm-sms	✔	✔	✔	✔	▮	▮
camel-cmis	✔	✔	✔	✔	▮	▮
camel-coap	✔	✔	✔	✔	▮	▮
camel-cometd	✔	✔	✔	✔	▮	▮
camel-context	*已弃用*	▮	▮	▮	▮	▮
camel-consul	✔	✔	✔	✔	▮	▮
camel-controlbus	✔	✔	✔	✔	▮	▮
camel-couchbase	✔	✔	✔	✔	▮	▮
camel-couchdb	✔	✔	✔	✔	▮	▮
camel-cql	✔	✔	✔	✔	▮	▮

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP	IBM Power / Spring Boot 2	IBM Z / Spring Boot 2
camel-crypto	✔	✔	✔	✔	▮	▮
camel-crypto-cms	✔	✔	✔	✔	▮	▮
camel-cxf	✔	✔	✔	✔	▮	▮
camel-cxf-transport	✔	✔	✔	✔	▮	▮
camel-dataformat	✔	✔	✔	✔	✔	✔
camel-dataset	✔	✔	✔	✔	▮	▮
camel-digitalocean	✔	✔	✔	✔	▮	▮
camel-direct	✔	✔	✔	✔	▮	▮
camel-direct-vm	✔	✔	✔	✔	▮	▮
camel-disruptor	✔	✔	✔	✔	▮	▮
camel-dns	✔	✔	✔	✔	▮	▮
camel-docker	✔	✔	✔	✔	▮	▮
camel-dozer	✔	✔	✔	✔	▮	▮
camel-drill	✔	✔	✔	▮	▮	▮
camel-dropbox	✔	✔	✔	✔	▮	▮
camel-eclipse	*已弃用*	▮	▮	▮	▮	▮
camel-ehcache	✔	✔	✔	✔	▮	▮
camel-ejb	✔	▮	▮	✔	▮	▮
camel-elasticsearch	✔	✔	✔	✔	▮	▮

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP	IBM Power / Spring Boot 2	IBM Z / Spring Boot 2
camel-elasticsearch5	✔	✔	✔	✔	▮	▮
camel-elasticsearch-rest	✔	✔	✔	✔	▮	▮
camel-elsql	✔	✔	✔	✔	▮	▮
camel-etcd	✔	✔	✔	✔	▮	▮
camel-eventadmin	✔	▮	✔	▮	▮	▮
camel-exec	✔	✔	✔	✔	✔	✔
camel-facebook	✔	✔	✔	✔	▮	▮
camel-fhir	▮	✔	▮	▮	*已弃用*	*已弃用*
camel-file	✔	✔	✔	✔	✔	✔
camel-flatpack	✔	✔	✔	✔	▮	▮
camel-flink	✔	✔	▮	✔	▮	▮
camel-fop	✔	✔	✔	✔	▮	▮
camel-freemarker	✔	✔	✔	✔	▮	▮
camel-ftp	✔	✔	✔	✔	✔	✔
camel-gae	*已弃用*	▮	▮	▮	▮	▮
camel-ganglia	✔	✔	✔	✔	▮	▮
camel-geocoder	✔	✔	✔	✔	▮	▮
camel-git	✔	✔	✔	✔	▮	▮
camel-github	✔	✔	✔	✔	▮	▮

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP	IBM Power / Spring Boot 2	IBM Z / Spring Boot 2
camel-google-bigquery	✔	✔	▮	✔	▮	▮
camel-google-calendar	✔	✔	✔	✔	▮	▮
camel-google-drive	✔	✔	✔	✔	▮	▮
camel-google-mail	✔	✔	✔	✔	▮	▮
camel-google-pubsub	✔	✔	✔	✔	▮	▮
camel-google-sheets	▮	✔	▮	▮	▮	▮
camel-grape	✔	✔	✔	▮	▮	▮
camel-groovy-dsl	*已弃用*	▮	▮	▮	▮	▮
camel-grpc	✔	✔	✔	✔	▮	▮
camel-guava-eventbus	✔	✔	✔	✔	▮	▮
camel-guice	*已弃用*	*已弃用*	▮	▮	▮	▮
camel-hawtdb	*已弃用*	*已弃用*	*已弃用*	▮	▮	▮
camel-hazelcast	✔	✔	✔	✔	▮	▮
camel-hbase	✔	✔	▮	▮	▮	▮
camel-hdfs	*已弃用*	▮	▮	▮	▮	▮
camel-hdfs2	✔	✔	✔	✔	▮	▮
camel-headersmap	✔	✔	✔	✔	▮	▮
camel-hipchat	✔	✔	✔	✔	▮	▮

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP	IBM Power / Spring Boot 2	IBM Z / Spring Boot 2
camel-http	*已弃用*	*已弃用*	✔	▮	▮	▮
camel-http4	✔	✔	✔	✔	✔	✔
camel-hystrix	✔	✔	✔	✔	▮	▮
camel-ibatis	*已弃用*	▮	▮	▮	▮	▮
camel-iec60870	✔	✔	✔	✔	▮	▮
camel-ignite	▮	▮	▮	▮	▮	▮
camel-imap	✔	✔	✔	✔	▮	▮
camel-infinispan	✔	✔	✔	✔	✔	✔
camel-influxdb	✔	✔	✔	✔	▮	▮
camel-ipfs	▮	✔	▮	✔	▮	▮
camel-irc	✔	✔	✔	✔	▮	▮
camel-ironmq	▮	▮	▮	▮	▮	▮
camel-jasypt	✔	✔	✔	✔	▮	▮
camel-javaspace	*已弃用*	▮	▮	▮	▮	▮
camel-jbpm	▮	▮	▮	▮	▮	▮
camel-jcache	✔	✔	✔	✔	▮	▮
camel-jcifs	✔	▮	✔	▮	▮	▮
camel-jclouds	✔	▮	✔	✔	▮	▮
camel-jcr	✔	✔	✔	✔	▮	▮
camel-jdbc	✔	✔	✔	✔	✔	✔
camel-jetty	*已弃用*	*已弃用*	*已弃用*	▮	▮	▮

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP	IBM Power / Spring Boot 2	IBM Z / Spring Boot 2
camel-jetty8	▮	▮	▮	▮	▮	▮
camel-jetty9	✔	✔	✔	▮	✔	✔
camel-jgroups	✔	✔	✔	✔	▮	▮
camel-jing	✔	✔	✔	✔	▮	▮
camel-jira	✔	✔	▮	▮	▮	▮
camel-jms	✔	✔	✔	✔	✔	✔
camel-jmx	✔	✔	✔	✔	▮	▮
camel-jolt	✔	✔	✔	✔	▮	▮
camel-josql	*已弃用*	*已弃用*	*已弃用*	▮	▮	▮
camel-jpa	✔	✔	✔	✔	✔	✔
camel-jsch	✔	✔	✔	✔	▮	▮
camel-json-validator	✔	✔	✔	▮	✔	✔
camel-jt400	✔	✔	✔	✔	▮	▮
camel-juel	*已弃用*	*已弃用*	*已弃用*	▮	▮	▮
camel-kafka	✔	✔	✔	✔	✔	✔
camel-kestrel	*已弃用*	*已弃用*	*已弃用*	▮	▮	▮
camel-krati	*已弃用*	*已弃用*	*已弃用*	▮	▮	▮
camel-kubernetes	✔	✔	✔	✔	▮	▮
camel-kura	✔	▮	✔	▮	▮	▮
camel-ldap	✔	✔	✔	✔	▮	▮
camel-ldif	✔	✔	✔	▮	▮	▮

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP	IBM Power / Spring Boot 2	IBM Z / Spring Boot 2
camel-leveldb	✔	✔	✔	✔	▮	▮
camel-linkedin	▮	▮	▮	▮	▮	▮
camel-log	✔	✔	✔	✔	▮	▮
camel-lpr	✔	✔	✔	✔	▮	▮
camel-lra	▮	▮	▮	▮	▮	▮
camel-lucene	✔	✔	✔	✔	▮	▮
camel-lumberjack	✔	✔	✔	✔	▮	▮
camel-master	✔	✔	✔	▮	▮	▮
camel-mail	✔	✔	✔	✔	▮	▮
camel-metrics	✔	✔	✔	✔	▮	▮
camel-micrometer	▮	✔	▮	▮	▮	▮
camel-milo	✔	✔	✔	✔	▮	▮
camel-mina	*已弃用*	▮	✔	▮	▮	▮
camel-mina2	✔	✔	✔	✔	▮	▮
camel-mllp	✔	✔	✔	✔	▮	▮
camel-mock	✔	✔	✔	✔	▮	▮
camel-mongodb	✔	✔	✔	✔	▮	▮
camel-mongodb-gridfs	✔	✔	✔	✔	▮	▮
camel-mongodb3	✔	✔	✔	✔	▮	▮
camel-mqtt	*已弃用*	*已弃用*	*已弃用*	*已弃用*	▮	▮

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP	IBM Power / Spring Boot 2	IBM Z / Spring Boot 2
camel-msv	✔	✔	✔	✔	▮	▮
camel-mustache	✔	✔	✔	✔	▮	▮
camel-mvel	✔	✔	✔	✔	▮	▮
camel-mybatis	✔	✔	✔	✔	▮	▮
camel-nagios	✔	✔	✔	▮	▮	▮
camel-nats	✔	✔	✔	✔	▮	▮
camel-netty	*已弃用*	*已弃用*	✔	▮	▮	▮
camel-netty-http	*已弃用*	*已弃用*	✔	▮	▮	▮
camel-netty4	✔	✔	✔	✔	▮	▮
camel-netty4-http	✔	✔	✔	▮	▮	▮
camel-nsq	▮	✔	▮	▮	▮	▮
camel-olingo2	✔	✔	✔	✔	▮	▮
camel-olingo4	✔	✔	✔	✔	▮	▮
camel-openapi-java	✔	✔	✔	✔	✔	✔
camel-openshift	*已弃用*	▮	✔	▮	▮	▮
camel-openstack	✔	✔	✔	✔	▮	▮
camel-opentracing	✔	✔	✔	✔	▮	▮
camel-optaplanner	✔	✔	✔	✔	▮	▮
camel-paho	✔	✔	✔	✔	▮	▮

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP	IBM Power / Spring Boot 2	IBM Z / Spring Boot 2
camel-paxlogging	✔	▮	✔	▮	▮	▮
camel-pdf	✔	✔	✔	✔	▮	▮
camel-pgevent	✔	✔	✔	✔	▮	▮
camel-pop3	✔	✔	✔	✔	▮	▮
camel-printer	✔	✔	✔	✔	▮	▮
camel-properties	✔	✔	✔	✔	▮	▮
camel-pubnub	✔	✔	✔	✔	▮	▮
camel-pulsar	✔	✔	✔	▮	▮	▮
camel-quartz	*已弃用*	▮	✔	▮	▮	▮
camel-quartz2	✔	✔	✔	✔	▮	▮
camel-quickfix	✔	✔	✔	✔	✔	✔
camel-rabbitmq	✔	✔	✔	✔	▮	▮
camel-reactive-streams	✔	✔	✔	✔	▮	▮
camel-reactor	✔	✔	✔	✔	▮	▮
camel-ref	✔	✔	✔	✔	▮	▮
camel-rest	✔	✔	✔	✔	✔	✔
camel-rest-api	✔	✔	✔	✔	✔	✔
camel-rest-openapi	✔	✔	✔	✔	✔	✔
camel-rest-swagger	✔	✔	✔	✔	▮	▮
camel-restlet	✔	✔	✔	▮	▮	▮

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP	IBM Power / Spring Boot 2	IBM Z / Spring Boot 2
camel-ribbon	✔	✔	▮	✔	▮	▮
camel-rmi	✔	✔	✔	✔	▮	▮
camel-routebox	*已弃用*	▮	✔	▮	▮	▮
camel-rss	✔	✔	✔	✔	▮	▮
camel-rx	*已弃用*	*已弃用*	*已弃用*	▮	▮	▮
camel-rxjava2	▮	✔	▮	▮	▮	▮
camel-saga	▮	▮	▮	▮	▮	▮
camel-salesforce	✔	✔	✔	✔	▮	▮
camel-sap	✔	✔	✔	✔	✔	✔
camel-sap-netweaver	✔	✔	✔	✔	✔	✔
camel-saxon	✔	✔	✔	✔	▮	▮
camel-scala	*已弃用*	*已弃用*	*已弃用*	▮	▮	▮
camel-scheduler	✔	✔	✔	✔	▮	▮
camel-schematron	✔	✔	✔	✔	▮	▮
camel-scp	✔	✔	✔	✔	▮	▮
camel-scr	*已弃用*	✔	*已弃用*	▮	▮	▮
camel-script	*已弃用*	*已弃用*	*已弃用*	*已弃用*	▮	▮
camel-seda	✔	✔	✔	✔	▮	▮
camel-service	▮	✔	▮	▮	▮	▮
camel-servicenow	✔	✔	✔	✔	▮	▮

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP	IBM Power / Spring Boot 2	IBM Z / Spring Boot 2
camel-servlet	✔	✔	✔	✔	▮	▮
camel-servletlistener	*已弃用*	*已弃用*	*已弃用*	▮	▮	▮
camel-sftp	✔	✔	✔	✔	✔	✔
camel-shiro	✔	✔	✔	✔	▮	▮
camel-sip	✔	✔	✔	✔	▮	▮
camel-sjms	✔	✔	✔	✔	▮	▮
camel-sjms2	✔	✔	✔	✔	▮	▮
camel-slack	✔	✔	✔	✔	▮	▮
camel-smpp	✔	✔	✔	✔	▮	▮
camel-snakeyaml	✔	✔	✔	✔	▮	▮
camel-snmp	✔	✔	✔	✔	▮	▮
camel-solr	✔	✔	✔	✔	▮	▮
camel-spark	✔	✔	▮	▮	▮	▮
camel-spark-rest	✔	✔	▮	▮	▮	▮
camel-splunk	✔	✔	✔	✔	▮	▮
camel-spring	✔	✔	✔	✔	✔	✔
camel-spring-batch	✔	✔	✔	✔	✔	✔
camel-spring-boot	✔	✔	▮	▮	✔	✔
camel-spring-cloud	✔	✔	▮	▮	✔	✔

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP	IBM Power / Spring Boot 2	IBM Z / Spring Boot 2
camel-spring-cloud-consul	▮	✔	▮	▮	▮	▮
camel-spring-cloud-netflix	✔	✔	▮	▮	▮	▮
camel-spring-cloud-zookeeper	▮	✔	▮	▮	▮	▮
camel-spring-event	✔	✔	▮	✔	✔	✔
camel-spring-integration	✔	▮	▮	✔	▮	▮
camel-spring-javaconfig	✔	✔	▮	✔	▮	▮
camel-spring-ldap	✔	✔	✔	✔	▮	▮
camel-spring-redis	✔	✔	▮	✔	▮	▮
camel-spring-security	✔	✔	✔	✔	✔	✔
camel-spring-ws	✔	✔	✔	▮	▮	▮
camel-sql	✔	✔	✔	✔	✔	✔
camel-sql-stored	✔	✔	✔	✔	✔	✔
camel-ssh	✔	✔	✔	✔	✔	✔
camel-stax	✔	✔	✔	✔	▮	▮
camel-stomp	✔	✔	✔	✔	▮	▮
camel-stream	✔	✔	✔	✔	▮	▮

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP	IBM Power / Spring Boot 2	IBM Z / Spring Boot 2
camel-string-template	✔	✔	✔	✔	▮	▮
camel-stub	✔	✔	✔	✔	▮	▮
camel-swagger	*已弃用*	▮	*已弃用*	▮	▮	▮
camel-swagger-java	✔	✔	✔	✔	▮	▮
camel-tagsoup	✔	✔	✔	✔	▮	▮
camel-telegram	✔	✔	✔	✔	▮	▮
camel-test	✔	✔	✔	✔	▮	▮
camel-thrift	✔	✔	✔	✔	▮	▮
camel-tika	✔	✔	✔	✔	▮	▮
camel-timer	✔	✔	✔	✔	▮	▮
camel-twilio	✔	✔	✔	✔	▮	▮
camel-twitter	✔	✔	✔	✔	▮	▮
camel-undertow	✔	✔	✔	✔	▮	▮
camel-urlrewrite	*已弃用*	*已弃用*	*已弃用*	▮	▮	▮
camel-validator	✔	✔	✔	✔	▮	▮
camel-velocity	✔	✔	✔	✔	▮	▮
camel-vertx	✔	✔	✔	✔	▮	▮
camel-vm	✔	✔	✔	✔	▮	▮
camel-weather	✔	✔	✔	✔	▮	▮
camel-web3j	▮	✔	▮	▮	▮	▮

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP	IBM Power / Spring Boot 2	IBM Z / Spring Boot 2
camel-websocket	✔	✔	✔	▮	▮	▮
camel-weka	✔	▮	▮	✔	▮	▮
camel-wordpress	✔	✔	✔	✔	▮	▮
camel-xchange	✔	✔	✔	✔	▮	▮
camel-xmlrpc	▮	▮	▮	▮	▮	▮
camel-xmlsecurity	✔	✔	✔	✔	▮	▮
camel-xmpp	✔	✔	✔	✔	▮	▮
camel-xquery	✔	✔	✔	✔	▮	▮
camel-xslt	✔	✔	✔	✔	▮	▮
camel-yammer	✔	✔	✔	✔	▮	▮
camel-yql	▮	▮	▮	▮	▮	▮
camel-zendesk	✔	✔	▮	✔	▮	▮
camel-zipkin	✔	✔	✔	✔	▮	▮
camel-zookeeper	✔	✔	✔	✔	▮	▮
camel-zookeeper-master	✔	✔	✔	✔	▮	▮

表 1.2. Apache Camel 数据格式支持列表

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP
camel-asn1	✔	✔	✔	✔
camel-avro	✔	✔	✔	✔

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP
camel-barcode	✔	✔	✔	✔
camel-base64	✔	✔	✔	✔
camel-beanio	✔	✔	✔	✔
camel-bindy	✔	✔	✔	✔
camel-boon	✔	✔	✔	✔
camel-castor	*已弃用*	*已弃用*	*已弃用*	✔
camel-crypto	✔	✔	✔	✔
camel-csv	✔	✔	✔	✔
camel-fhir	✔	✔	✔	✔
camel-flatpack	✔	✔	✔	✔
camel-gzip	✔	✔	✔	✔
camel-hessian	*已弃用*	*已弃用*	*已弃用*	*已弃用*
camel-hl7	✔	✔	✔	✔
camel-ical	✔	✔	✔	✔
camel-jacksonxml	✔	✔	✔	✔
camel-jaxb	✔	✔	✔	✔
camel-jibx	✔	✔	✔	✔
camel-json-fastjson	✔	✔	✔	✔
camel-json-gson	✔	✔	✔	✔
camel-json-jackson	✔	✔	✔	✔
camel-json-johnzon	✔	✔	✔	✔
camel-json-xstream	✔	✔	✔	✔

组件	无容器	Spring Boot 2.x	Karaf	JBoss EAP
camel-lzf	✔	✔	✔	✔
camel-mime-multipart	✔	✔	✔	✔
camel-pgp	✔	✔	✔	✔
camel-protobuf	✔	✔	✔	✔
camel-rss	✔	✔	✔	✔
camel-serialization	✔	✔	✔	✔
camel-soapjaxb	✔	✔	✔	✔
camel-string	✔	✔	✔	✔
camel-syslog	✔	✔	✔	✔
camel-tarfile	✔	✔	✔	✔
camel-thrift	✔	✔	✔	✔
camel-univocity-csv	✔	✔	✔	✔
camel-univocity-fixed	✔	✔	✔	✔
camel-univocity-tsv	✔	✔	✔	✔
camel-xmlbeans	*已弃用*	*已弃用*	*已弃用*	✔
camel-xmljson	*已弃用*	*已弃用*	*已弃用*	*已弃用*
camel-xmlrpc	▮	▮	▮	▮
camel-xstream	✔	✔	✔	✔
camel-yaml-snakeyaml	✔	✔	✔	✔
camel-zip	✔	✔	✔	✔
camel-zipfile	✔	✔	✔	✔

表 1.3. Apache Camel 语言支持列表

语言	无容器	Spring Boot 2.x	Karaf	JBoss EAP
Bean 方法	✔	✔	✔	✔
常数	✔	✔	✔	✔
EL	*已弃用*	▮	▮	▮
ExchangeProperty	✔	✔	✔	✔
File	✔	✔	✔	✔
Groovy	✔	✔	✔	✔
标头	✔	✔	✔	✔
JsonPath	✔	✔	✔	✔
jxpath	*已弃用*	▮	▮	▮
MVEL	✔	✔	✔	✔
OGNL	✔	✔	✔	✔
PHP	*已弃用*	*已弃用*	▮	*已弃用*
Python	*已弃用*	*已弃用*	▮	*已弃用*
Ref	✔	✔	✔	✔
Ruby	*已弃用*	*已弃用*	▮	*已弃用*
Simple（简单）	✔	✔	✔	✔
SpEL	✔	✔	▮	✔
tokenize	✔	✔	✔	✔
XML 令牌化	✔	✔	✔	✔
XPath	✔	✔	✔	✔
XQuery	✔	✔	✔	✔

第 2 章 ACTIVEMQ

ACTIVEMQ 组件

利用 ActiveMQ 组件，您可以发送消息到 JMS Queue 或 Topic，或者使用 Apache ActiveMQ 来使用来自 JMS Queue 或 Topic 的消息。

此组件基于 JMS 组件，使用 Spring 的 JMS 支持进行声明性事务，使用 Spring 的 **JmsTemplate** 发送和 **MessageListenerContainer** 以供使用。所有 JMS 组件选项 也适用于 ActiveMQ 组件。

要使用此组件，请确保在 classpath 上具有 **activemq.jar** 或 **activemq-core.jar**，以及任何 Apache Camel 依赖项，如 **camel-core.jar**、**camel-spring.jar** 和 **camel-jms.jar**。

转换和缓存

如果使用 JMS 的事务，请参阅 JMS 页面上的 **Transactions 和 Cache**Levels 部分，因为它可能会影响性能。

URI 格式

```
activemq:[queue:|topic:]destinationName
```

其中 **destinationName** 是 ActiveMQ 队列或主题名称。默认情况下，**destinationName** 解释为队列名称。例如，要连接到队列 **FOO.BAR**，请使用：

```
activemq:FOO.BAR
```

如果需要，您可以包含可选的 **queue:** 前缀：

```
activemq:queue:FOO.BAR
```

要连接到主题，您必须包含 **topic:** 前缀。例如，要连接到主题 **Stocks.Prices**，请使用：

```
activemq:topic:Stocks.Prices
```

选项

所有 JMS 组件选项 也适用于 ActiveMQ 组件。

CAMEL ON EAP 部署

此组件由 EAP 上的 Camel (Wildfly Camel)框架支持，该框架在 Red Hat JBoss Enterprise Application Platform (JBoss EAP)容器上提供了简化的部署模型。

您可以配置 ActiveMQ Camel 组件，以用于嵌入式代理或外部代理。要在 JBoss EAP 容器中嵌入代理，请在 EAP 容器配置文件 configuring ActiveMQ 资源适配器中配置 ActiveMQ 资源适配器，详细信息请参阅 ActiveMQ 资源适配器配置。

配置连接工厂

以下 测试案例 演示了如何在指定用于连接 ActiveMQ 的 brokerURL 时，使用 **activeMQComponent（）** 方法将 ActiveMQComponent 添加到 CamelContext 中。

```
camelContext.addComponent("activemq", activeMQComponent("vm://localhost?
broker.persistent=false"));
```

使用 SPRING XML 配置连接工厂

您可以在 ActiveMQComponent 上配置 ActiveMQ 代理 URL，如下所示

```
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="
      http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
      http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-
spring.xsd">

  <camelContext xmlns="http://camel.apache.org/schema/spring">
  </camelContext>

  <bean id="activemq" class="org.apache.activemq.camel.component.ActiveMQComponent">
    <property name="brokerURL" value="tcp://somehost:61616"/>
  </bean>

</beans>
```

使用连接池

使用 Camel 发送到 ActiveMQ 代理时，最好使用池化连接工厂来处理 JMS 连接、会话和生产者的有效池。如需更多信息，请参阅 ActiveMQ Spring 支持。

1. 使用 Maven 添加 AMQ 池：

```
    <dependency>
      <groupId>org.apache.activemq</groupId>
      <artifactId>activemq-pool</artifactId>
      <version>5.11.0.redhat-630516</version>
    </dependency>
```

2. 设置 **activemq** 组件，如下所示：

```
    <bean id="jmsConnectionFactory" class="org.apache.activemq.ActiveMQConnectionFactory">
        <property name="brokerURL" value="tcp://localhost:61616" />
    </bean>

    <bean id="pooledConnectionFactory"
class="org.apache.activemq.pool.PooledConnectionFactory" init-method="start" destroy-
method="stop">
        <property name="maxConnections" value="8" />
        <property name="connectionFactory" ref="jmsConnectionFactory" />
    </bean>
```

```
<bean id="jmsConfig" class="org.apache.camel.component.jms.JmsConfiguration">
    <property name="connectionFactory" ref="pooledConnectionFactory"/>
    <property name="concurrentConsumers" value="10"/>
</bean>

<bean id="activemq" class="org.apache.activemq.camel.component.ActiveMQComponent">
    <property name="configuration" ref="jmsConfig"/>
</bean>
```

注意

注意池连接工厂的 **init** 和 **destroy** 方法。这些方法是确保正确启动和关闭连接池。

然后，pooled **ConnectionFactory** 将创建一个同时使用最多 8 个连接的连接池。每个连接可由多个会话共享。

有一个名为 **maxActive** 的选项，可用于配置每个连接的最大会话数；默认值为 **500**。

自 **ActiveMQ 5.7** 起，选项已重命名为 **maxActiveSessionPerConnection**，以更好地反映其目的。

请注意，**concurrentConsumers** 被设置为高于 **maxConnections** 的值。允许这样做，因为每个消费者都使用会话，因为会话可以共享相同的连接，所以这可以正常工作。在这个示例中，我们可以同时有 8 * 500 = 4000 活跃的会话。

在路由中调用 MESSAGELISTENER POJO

ActiveMQ 组件还提供了从 JMS MessageListener 到处理器的帮助 类型转换器。https://camel.apache.org/manual/processor.html这意味着 Bean 组件 可以直接在任何路由内调用任何 JMS MessageListener bean。

您可以在 JMS 中创建 MessageListener，如下所示：

Example

```
public class MyListener implements MessageListener {
    public void onMessage(Message jmsMessage) {
        // ...
    }
}
```

然后，在您的路由中使用它，如下所示

Example

```
from("file://foo/bar").
  bean(MyListener.class);
```

也就是说，您可以重复使用任何 Apache Camel 组件，并将它们轻松集成到 JMS **MessageListener** POJO\!

使用 ACTIVEMQ DESTINATION OPTIONS

从 ActiveMQ 5.6 开始提供

您可以使用 "destination." 前缀在端点 uri 中配置 Destination Options。例如，要将消费者标记为 exclusive，并将其预抓取大小设置为 50，您可以执行以下操作： .Example

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="file://src/test/data?noop=true"/>
    <to uri="activemq:queue:foo"/>
  </route>
  <route>
    <!-- use consumer.exclusive ActiveMQ destination option, notice we have to prefix with destination.
-->
    <from uri="activemq:foo?
destination.consumer.exclusive=true&amp;destination.consumer.prefetchSize=50"/>
    <to uri="mock:results"/>
  </route>
</camelContext>
```

消耗公告消息

ActiveMQ 可以生成 公告消息，被放入您可以使用的主题中。当您检测到较慢的用户或构建统计信息时，这些消息可帮助您发送警报（每天生成的消息/数量等）以下 Spring DSL 示例演示了如何从主题读取消息。

Example

```
<route>
 <from uri="activemq:topic:ActiveMQ.Advisory.Connection?mapJmsMessage=false" />
 <convertBodyTo type="java.lang.String"/>
 <transform>
     <simple>${in.body}&#13;</simple>
 </transform>
 <to uri="file://data/activemq/?fileExist=Append&ileName=advisoryConnection-
${date:now:yyyyMMdd}.txt" />
</route>
```

如果您在队列中使用消息，则应在 data/activemq 文件夹下看到以下文件：

```
advisoryConnection-20100312.txt

advisoryProducer-20100312.txt
```

包含字符串：

Example

```
    ActiveMQMessage {commandId = 0, responseRequired = false, messageId = ID:dell-charles-
3258-1268399815140
    -1:0:0:0:221, originalDestination = null, originalTransactionId = null, producerId = ID:dell-charles-
    3258-1268399815140-1:0:0:0, destination = topic://ActiveMQ.Advisory.Connection, transactionId
= null,
    expiration = 0, timestamp = 0, arrival = 0, brokerInTime = 1268403383468, brokerOutTime =
1268403383468,
    correlationId = null, replyTo = null, persistent = false, type = Advisory, priority = 0, groupID = null,
    groupSequence = 0, targetConsumerId = null, compressed = false, userID = null, content = null,
```

```
    marshalledProperties = org.apache.activemq.util.ByteSequence@17e2705, dataStructure =
ConnectionInfo
    {commandId = 1, responseRequired = true, connectionId = ID:dell-charles-3258-1268399815140-
2:50,
    clientId = ID:dell-charles-3258-1268399815140-14:0, userName = , password = *****,
    brokerPath = null, brokerMasterConnector = false, manageable = true, clientMaster = true},
    redeliveryCounter = 0, size = 0, properties = {originBrokerName=master, originBrokerId=ID:dell-
charles-
    3258-1268399815140-0:0, originBrokerURL=vm://master}, readOnlyProperties = true,
readOnlyBody = true,
    droppable = false}
```

获取组件 JAR

您需要这个依赖项：

- **camel-activemq**

ActiveMQ 是与 ActiveMQ 项目 一起发布的 JMS 组件的 扩展。

```
<dependency>
  <groupId>org.fusesource</groupId>
  <artifactId>camel-activemq</artifactId>
  <version>7.11.0.fuse-sb2-7_11_0-00035-redhat-00001</version>
</dependency>
```

第 3 章 AHC 组件

从 Camel 版本 2.8 开始提供

ahc: 组件为消耗外部 HTTP 资源提供基于 HTTP 的端点（作为使用 HTTP 调用外部服务器的客户端）。组件使用 Async Http Client 库。

Maven 用户需要将以下依赖项添加到这个组件的 **pom.xml** 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-ahc</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

3.1. URI 格式

```
ahc:http://hostname[:port][/resourceUri][?options]
ahc:https://hostname[:port][/resourceUri][?options]
```

默认情况下，HTTP 和 443 将端口 80 用于 HTTPS。

您可以在 URI 中附加查询选项，格式为 **?option=value&option=value&...**

3.2. AHCENDPOINT OPTIONS

AHC 端点使用 URI 语法进行配置：

```
ahc:httpUri
```

使用以下路径和查询参数：

3.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
httpUri	**必需** 要使用的 URI，如 http://hostname:port/path		URI

3.2.2. 查询参数(13 参数)：

Name	描述	默认值	类型
bridgeEndpoint (producer)	如果选项为 true，则忽略 Exchange.HTTP_URI 标头，并使用端点的 URI 进行请求。您还可以将 throwExceptionOnFailure 设置为 false，以便 AhcProducer 发送所有错误响应。	false	布尔值

Name	描述	默认值	类型
bufferSize (producer)	在 Camel 和 AHC 客户端之间传输数据时使用的初始内存缓冲区大小。	4096	int
connectionClose (producer)	定义 Connection Close 标头是否已添加到 HTTP Request 中。默认情况下，此参数为 false	false	布尔值
cookieHandler (producer)	配置 Cookie 处理程序，以维护 HTTP 会话		CookieHandler
headerFilterStrategy (producer)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
throwExceptionOnFailure (producer)	如果来自远程服务器的失败响应，用于禁用抛出 AhcOperationFailedException。这样，您可以获取所有响应，而不考虑 HTTP 状态代码。	true	布尔值
transferException (producer)	如果在消费者端启用并交换失败处理，如果原因 Exception 在响应中作为 application/x-java-serialized-object 内容类型发送序列化，例如使用 Jetty 或 Servlet Camel 组件。在生成者端，异常将被反序列化并抛出，而不是 AhcOperationFailedException。原因异常需要按顺序处理。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
绑定 (advanced)	使用自定义 AhcBinding，允许控制如何在 AHC 和 Camel 间绑定。		AhcBinding
clientConfig (advanced)	将 AsyncHttpClient 配置为使用自定义 com.ning.http.client.AsyncHttpClientConfig 实例。		AsyncHttpClientConfig
clientConfigOptions (advanced)	使用映射中的键/值配置 AsyncHttpClientConfig。		Map
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
clientConfigRealmOptions (security)	使用映射中的键/值配置 AsyncHttpClientConfig Realm。		Map
sslContextParameters (security)	对 Registry 中的 org.apache.camel.util.jsse.SSLContextParameters 的引用。此引用覆盖组件级别上配置的 SSLContextParameters。请参阅使用 JSSE 配置实用程序。请注意，配置这个选项将覆盖端点或组件级别的 clientConfig 选项提供的任何 SSL/TLS 配置选项。		SSLContextParameters

3.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
camel.component.ahc.allow-java-serialized-object	当请求使用 context-type=application/x-java-serialized-object This 默认为 off 时允许 java serialization。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
camel.component.ahc.binding	使用自定义 AhcBinding，允许控制如何在 AHC 和 Camel 间绑定。选项是 org.apache.camel.component.ahc.AhcBinding 类型。		字符串
camel.component.ahc.client	使用自定义 AsyncHttpClient。选项是 org.asynchttpclient.AsyncHttpClient 类型。		字符串
camel.component.ahc.client-config	将 AsyncHttpClient 配置为使用自定义 com.ning.http.client.AsyncHttpClientConfig 实例。选项是 org.asynchttpclient.AsyncHttpClientConfig 类型。		字符串
camel.component.ahc.enabled	启用 ahc 组件	true	布尔值
camel.component.ahc.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		字符串
camel.component.ahc.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.ahc.ssl-context-parameters	对 Registry 中的 org.apache.camel.util.jsse.SSLContextParameters 的引用。请注意，配置这个选项将覆盖端点或组件级别的 clientConfig 选项提供的任何 SSL/TLS 配置选项。选项是 org.apache.camel.util.jsse.SSLContextParameters 类型。		字符串
camel.component.ahc.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

3.4. AHCCOMPONENT OPTIONS

AHC 组件支持 8 个选项，如下所列。

Name	描述	默认值	类型
客户端 (advanced)	使用自定义 AsyncHttpClient		AsyncHttpClient
绑定 (advanced)	使用自定义 AhcBinding，允许控制如何在 AHC 和 Camel 间绑定。		AhcBinding
clientConfig (advanced)	将 AsyncHttpClient 配置为使用自定义 com.ning.http.client.AsyncHttpClientConfig 实例。		AsyncHttpClientConfig
sslContextParameters (security)	对 Registry 中的 org.apache.camel.util.jsse.SSLContextParameters 的引用。请注意，配置这个选项将覆盖端点或组件级别的 clientConfig 选项提供的任何 SSL/TLS 配置选项。		SSLContextParameters
允许 JavaSerializedObject (advanced)	当请求使用 context-type=application/x-java-serialized-object This 默认为 off 时允许 java serialization。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

请注意，在 **AhcComponent** 中设置任何选项会将这些选项传播到正在创建的 **AhcEndpoints** 中。但是 **AhcEndpoint** 也可以配置/覆盖自定义选项。在端点上设置的选项始终优先于 **AhcComponent** 中的选项。

3.5. 消息标头

Name	类型	描述
Exchange.HTTP_URI	**字符串**	要调用的 URI。将覆盖直接在端点上设置的现有 URI。

Name	类型	描述
Exchange.HTTP_PATH	**字符串**	请求 URI 的路径，标头将使用 HTTP_URI 构建请求 URI。如果路径以 "/" 开头，http producer 将尝试根据 Exchange.HTTP_BASE_URI 标头或 **exchange.getFromEndpoint （）.getEndpointUri （）；**
Exchange.HTTP_QUERY	**字符串**	**Camel 2.11 onwards:** URI 参数。将覆盖直接在端点上设置的现有 URI 参数。
Exchange.HTTP_RESPONSE_CODE	**int**	来自外部服务器的 HTTP 响应代码。确定为 200。
Exchange.HTTP_CHARACTER_ENCODING	**字符串**	字符编码.
Exchange.CONTENT_TYPE	**字符串**	HTTP 内容类型。在 IN 和 OUT 消息上设置，以提供内容类型，如 **text/html**。
Exchange.CONTENT_ENCODING	**字符串**	HTTP 内容编码。在 IN 和 OUT 消息上设置，以提供内容编码，如 **gzip**。

3.6. 消息正文

Camel 会将来自外部服务器的 HTTP 响应存储在 OUT 正文上。IN 消息中的所有标头都将复制到 OUT 消息，因此在路由过程中会保留标头。另外，Camel 将添加 HTTP 响应标头以及 OUT 消息标头。

3.7. 响应代码

Camel 将根据 HTTP 响应代码进行处理：

- 响应代码位于 100..299 之间，Camel 会将它视为成功的响应。
- 响应代码位于范围 300..399 中，Camel 会将它视为重定向响应，并将抛出带有信息的 **AhcOperationFailedException**。
- 响应代码为 400+，Camel 会将其视为外部服务器故障，并将抛出带有信息的 **AhcOperationFailedException**。
  throwExceptionOnFailure

  可以选项 **throwExceptionOnFailure**，它可以设置为 **false**，以防止为失败的响应代码抛出 **AhcOperationFailedException**。这样，您可以从远程服务器获得任何响应。

3.8. AHCOPERATIONFAILEDEXCEPTION

这个例外包括以下信息：

- HTTP 状态代码
- HTTP status 行（状态代码的文本）
- 重定向位置，如果服务器返回重定向
- 如果服务器提供了正文作为响应，则响应正文作为 **java.lang.String**

3.9. 使用 GET 或 POST 调用

以下算法用于确定是否应使用 **GET** 或 **POST** HTTP 方法：
1。标头中提供的使用方法。
2.如果标头中提供了查询字符串，**GET**。
3.**GET** 如果端点配置了查询字符串。
4.**POST** 如果有要发送的数据（任何人不是 null）。
5.**GET** 否则。

3.10. 将 URI 配置为调用

您可以直接设置 HTTP 制作者的 URI，形成端点 URI。在以下路由中，Camel 将使用 HTTP 调用外部服务器 **oldhost**。

```
from("direct:start")
        .to("ahc:http://oldhost");
```

以及等同的 Spring 示例：

```
<camelContext xmlns="http://activemq.apache.org/camel/schema/spring">
  <route>
    <from uri="direct:start"/>
    <to uri="ahc:http://oldhost"/>
  </route>
</camelContext>
```

您可以通过在消息中添加带有密钥 **Exchange.HTTP_URI** 的标头来覆盖 HTTP 端点 URI。

```
from("direct:start")
    .setHeader(Exchange.HTTP_URI, constant("http://newhost"))
    .to("ahc:http://oldhost");
```

3.11. 配置 URI 参数

ahc producer 支持将 URI 参数发送到 HTTP 服务器。URI 参数可以直接在端点 URI 上设置，也可以作为消息上带有密钥 **Exchange.HTTP_QUERY** 的标头。

```
from("direct:start")
        .to("ahc:http://oldhost?order=123&detail=short");
```

或在标头中提供的选项：

```
from("direct:start")
            .setHeader(Exchange.HTTP_QUERY, constant("order=123&detail=short"))
        .to("ahc:http://oldhost");
```

3.12. 如何将 HTTP 方法设置为 HTTP 生成者

HTTP 组件通过设置消息标头来提供设置 HTTP 请求方法的方法。下面是一个示例；

```
from("direct:start")
            .setHeader(Exchange.HTTP_METHOD, constant("POST"))
        .to("ahc:http://www.google.com")
            .to("mock:results");
```

以及等同的 Spring 示例：

```
<camelContext xmlns="http://activemq.apache.org/camel/schema/spring">
  <route>
    <from uri="direct:start"/>
    <setHeader headerName="CamelHttpMethod">
        <constant>POST</constant>
    </setHeader>
    <to uri="ahc:http://www.google.com"/>
    <to uri="mock:results"/>
  </route>
</camelContext>
```

3.13. 配置 CHARSET

如果您使用 **POST** 发送数据，您可以使用 **Exchange** 属性配置 **charset** ：

```
exchange.setProperty(Exchange.CHARSET_NAME, "iso-8859-1");
```

3.13.1. 端点 URI 中的 URI 参数

在本例中，我们有一个完整的 URI 端点，即您在 Web 浏览器中键入的内容。可以使用 **&** 作为分隔符设置多个 URI 参数，就像您在 Web 浏览器中一样。Camel 这里没有复杂的操作。

```
// we query for Camel at the Google page
template.sendBody("ahc:http://www.google.com/search?q=Camel", null);
```

3.13.2. 消息中的 URI 参数

```
Map headers = new HashMap();
headers.put(Exchange.HTTP_QUERY, "q=Camel&lr=lang_en");
// we query for Camel and English language at Google
template.sendBody("ahc:http://www.google.com/search", null, headers);
```

在上面的标头值中，**它不应该** 带有前缀 **?**，您可以使用 **&** amp; char 分隔参数。

3.13.3. 获取响应代码

您可以通过使用 **Exchange.HTTP_RESPONSE_CODE** 从 Out message 标头获取值，从 AHC 组件获取 HTTP 响应代码。

```
Exchange exchange = template.send("ahc:http://www.google.com/search", new Processor() {
        public void process(Exchange exchange) throws Exception {
            exchange.getIn().setHeader(Exchange.HTTP_QUERY, constant("hl=en&q=activemq"));
        }
});
Message out = exchange.getOut();
int responseCode = out.getHeader(Exchange.HTTP_RESPONSE_CODE, Integer.class);
```

3.14. 配置 ASYNCHTTPCLIENT

AsyncHttpClient 客户端使用 **AsyncHttpClientConfig** 来配置客户端。如需了解更多详细信息，请参阅 Async Http Client 文档。

在 Camel 2.8 中，配置仅限于使用 **AsyncHttpClientConfig.Builder** 提供的构建器模式。在 Camel 2.8 中，**Async** HttpClientConfig 不支持 getters/setters，因此无法使用 Spring bean 风格（如 XML 文件中的 <bean> 标签）轻松创建/配置。

以下示例演示了如何使用构建器创建在 **AhcComponent** 上配置的 **AsyncHttpClientConfig**。

在 Camel 2.9 中，AHC 组件使用 Async HTTP 库 1.6.4。此更新的版本添加了对 plain bean 风格配置的支持。**AsyncHttpClientConfigBean** 类为 **AsyncHttpClientConfig** 中提供的配置选项提供 getters 和 setters。**AsyncHttpClientConfigBean** 实例可以直接传递到 AHC 组件，或使用 **clientConfig** URI 参数在端点 URI 中引用。

Camel 2.9 还可用于直接在 URI 中设置配置选项。以"clientConfig."开头的 URI 参数可用于设置 **AsyncHttpClientConfig** 的各种可配置属性。端点 URI 中指定的属性与 "clientConfig" URI 参数引用的配置中指定的属性合并，使用 "clientConfig." 参数设置时具有优先权。引用的 **AsyncHttpClientConfig** 实例始终为每个端点复制，使得一个端点上的设置将独立于任何之前创建的端点的设置。以下示例演示了如何使用 "clientConfig." 类型 URI 参数配置 AHC 组件。

```
from("direct:start")
    .to("ahc:http://localhost:8080/foo?
clientConfig.maxRequestRetry=3&clientConfig.followRedirects=true")
```

3.15. SSL 支持(HTTPS)

使用 JSSE 配置实用程序

自 Camel 2.9 起，AHC 组件通过 Camel JSSE 配置实用程序支持 SSL/TLS 配置 。这个实用程序可大大减少您需要写入的组件特定代码量，并在端点和组件级别进行配置。以下示例演示了如何将 实用程序与 AHC 组件一起使用。

组件的编程配置

```
KeyStoreParameters ksp = new KeyStoreParameters();
ksp.setResource("/users/home/server/keystore.jks");
ksp.setPassword("keystorePassword");

KeyManagersParameters kmp = new KeyManagersParameters();
kmp.setKeyStore(ksp);
kmp.setKeyPassword("keyPassword");

SSLContextParameters scp = new SSLContextParameters();
scp.setKeyManagers(kmp);

AhcComponent component = context.getComponent("ahc", AhcComponent.class);
component.setSslContextParameters(scp));
```

基于 Spring DSL 的端点配置

```
...
  <camel:sslContextParameters
     id="sslContextParameters">
   <camel:keyManagers
      keyPassword="keyPassword">
     <camel:keyStore
        resource="/users/home/server/keystore.jks"
        password="keystorePassword"/>
   </camel:keyManagers>
  </camel:sslContextParameters>...
...
  <to uri="ahc:https://localhost/foo?sslContextParameters=#sslContextParameters"/>
...
```

3.16. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- Jetty
- HTTP
- HTTP4

第 4 章 AHC WEBSOCKET COMPONENT

从 Camel 版本 2.14 开始提供

ahc-ws 组件为通过 Websocket 进行通信的客户端提供基于 Websocket 的端点（作为客户端打开到外部服务器的 websocket 连接）。
组件使用 AHC 组件，后者使用 Async Http Client 库。

Maven 用户需要将以下依赖项添加到这个组件的 **pom.xml** 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-ahc-ws</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

4.1. URI 格式

```
ahc-ws://hostname[:port][/resourceUri][?options]
ahc-wss://hostname[:port][/resourceUri][?options]
```

默认情况下，ahc-ws 和 443 对 ahc-wss 使用端口 80。

4.2. AHC-WS 选项

因为 AHC-WS 组件基于 AHC 组件，因此您可以使用 AHC 组件的各种配置选项。

AHC Websocket 组件支持 8 个选项，如下所列。

Name	描述	默认值	类型
客户端 (advanced)	使用自定义 AsyncHttpClient		AsyncHttpClient
绑定 (advanced)	使用自定义 AhcBinding，允许控制如何在 AHC 和 Camel 间绑定。		AhcBinding
clientConfig (advanced)	将 AsyncHttpClient 配置为使用自定义 com.ning.http.client.AsyncHttpClientConfig 实例。		AsyncHttpClientConfig
sslContextParameters (security)	对 Registry 中的 org.apache.camel.util.jsse.SSLContextParameters 的引用。请注意，配置这个选项将覆盖端点或组件级别的 clientConfig 选项提供的任何 SSL/TLS 配置选项。		SSLContextParameters
允许 JavaSerialized Object (advanced)	当请求使用 context-type=application/x-java-serialized-object This 默认为 off 时允许 java serialization。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值

Name	描述	默认值	类型
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AHC Websocket 端点使用 URI 语法进行配置：

```
ahc-ws:httpUri
```

使用以下路径和查询参数：

4.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
httpUri	**必需** 要使用的 URI，如 http://hostname:port/path		URI

4.2.2. 查询参数(18 参数)：

Name	描述	默认值	类型
bridgeEndpoint (common)	如果选项为 true，则忽略 Exchange.HTTP_URI 标头，并使用端点的 URI 进行请求。您还可以将 throwExceptionOnFailure 设置为 false，以便 AhcProducer 发送所有错误响应。	false	布尔值
bufferSize (common)	在 Camel 和 AHC 客户端之间传输数据时使用的初始内存缓冲区大小。	4096	int
headerFilterStrategy (common)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
throwExceptionOnFailure (common)	如果来自远程服务器的失败响应，用于禁用抛出 AhcOperationFailedException。这样，您可以获取所有响应，而不考虑 HTTP 状态代码。	true	布尔值

Name	描述	默认值	类型
transferException (common)	如果在消费者端启用并交换失败处理，如果原因 Exception 在响应中作为 application/x-java-serialized-object 内容类型发送序列化，例如使用 Jetty 或 Servlet Camel 组件。在生成者端，异常将被反序列化并抛出，而不是 AhcOperationFailedException。原因异常需要按顺序处理。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
sendMessageOnError (consumer)	如果 web-socket 侦听器收到错误，是否发送一条消息。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
connectionClose (producer)	定义 Connection Close 标头是否已添加到 HTTP Request 中。默认情况下，此参数为 false	false	布尔值
cookieHandler (producer)	配置 Cookie 处理程序，以维护 HTTP 会话		CookieHandler
useStreaming (producer)	启用流，以作为多个文本片段发送数据。	false	布尔值
绑定 (advanced)	使用自定义 AhcBinding，允许控制如何在 AHC 和 Camel 间绑定。		AhcBinding
clientConfig (advanced)	将 AsyncHttpClient 配置为使用自定义 com.ning.http.client.AsyncHttpClientConfig 实例。		AsyncHttpClientConfig
clientConfigOptions (advanced)	使用映射中的键/值配置 AsyncHttpClientConfig。		Map

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
clientConfigRealmOptions (security)	使用映射中的键/值配置 AsyncHttpClientConfig Realm。		Map
sslContextParameters (security)	对 Registry 中的 org.apache.camel.util.jsse.SSLContextParameters 的引用。此引用覆盖组件级别上配置的 SSLContextParameters。请参阅使用 JSSE 配置实用程序。请注意，配置这个选项将覆盖端点或组件级别的 clientConfig 选项提供的任何 SSL/TLS 配置选项。		SSLContextParameters

4.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
camel.component.ahc-ws.allow-java-serialized-object	当请求使用 context-type=application/x-java-serialized-object This 默认为 off 时允许 java serialization。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
camel.component.ahc-ws.binding	使用自定义 AhcBinding，允许控制如何在 AHC 和 Camel 间绑定。选项是 org.apache.camel.component.ahc.AhcBinding 类型。		字符串
camel.component.ahc-ws.client	使用自定义 AsyncHttpClient。选项是 org.asynchttpclient.AsyncHttpClient 类型。		字符串
camel.component.ahc-ws.client-config	将 AsyncHttpClient 配置为使用自定义 com.ning.http.client.AsyncHttpClientConfig 实例。选项是 org.asynchttpclient.AsyncHttpClientConfig 类型。		字符串
camel.component.ahc-ws.enabled	启用 ahc-ws 组件	true	布尔值
camel.component.ahc-ws.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		字符串

Name	描述	默认值	类型
camel.component.ahc-ws.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.ahc-ws.ssl-context-parameters	对 Registry 中的 org.apache.camel.util.jsse.SSLContextParameters 的引用。请注意，配置这个选项将覆盖端点或组件级别的 clientConfig 选项提供的任何 SSL/TLS 配置选项。选项是 org.apache.camel.util.jsse.SSLContextParameters 类型。		字符串
camel.component.ahc-ws.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

4.4. 通过 WEBSOCKET 编写和读取数据

ahc-ws 端点可以将数据写入套接字，或者从套接字读取，具体取决于端点是否分别配置为生成者或消费者。

4.5. 将 URI 配置为 WRITE 或 READ DATA

在以下路由中，Camel 将写入指定的 websocket 连接。

```
from("direct:start")
        .to("ahc-ws://targethost");
```

以及等同的 Spring 示例：

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="direct:start"/>
    <to uri="ahc-ws://targethost"/>
  </route>
</camelContext>
```

在以下路由中，Camel 将从指定的 websocket 连接中读取。

```
from("ahc-ws://targethost")
        .to("direct:next");
```

以及等同的 Spring 示例：

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
```

```
  <route>
    <from uri="ahc-ws://targethost"/>
    <to uri="direct:next"/>
  </route>
</camelContext>
```

4.6. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- AHC
- Atmosphere-Websocket

第 5 章 AMQP 组件

从 Camel 版本 1.2 开始提供

amqp: 组件支持使用 Qpid 项目的 JMS Client API 的 AMQP 1.0 协议。如果要使用 AMQP 0.9 （特别是 RabbitMQ），您可能还对 Camel RabbitMQ 组件感兴趣。请注意，在 Camel 2.17.0 AMQP 组件前，支持 AMQP 0.9 及更高版本，但自 Camel 2.17.0 起，它只支持 AMQP 1.0。

Maven 用户需要将以下依赖项添加到这个组件的 **pom.xml** 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-amqp</artifactId>
  <version>${camel.version}</version> <!-- use the same version as your Camel core version -->
</dependency>
```

5.1. URI 格式

```
amqp:[queue:|topic:]destinationName[?options]
```

5.2. AMQP 选项

您可以在目标名称后指定 JMS 组件的所有不同配置选项。

AMQP 组件支持 81 选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	使用共享的 JMS 配置		JmsConfiguration
acceptMessagesWhileStopping (consumer)	指定消费者在停止时是否接受消息。如果您在运行时启动和停止 JMS 路由，同时队列中仍然有消息排队，您可以考虑启用此选项。如果此选项为 false，并且您停止 JMS 路由，则消息可能会被拒绝，并且 JMS 代理必须尝试恢复，并且再次可能被拒绝，最后该消息可能在 JMS 代理的死信队列中移动。为避免这种情况，建议启用这个选项。	false	布尔值
allowReplyManagerQuickStop (consumer)	在 request-reply 消息传递的回复管理器中使用的 DefaultMessageListenerContainer 是否允许 DefaultMessageListenerContainer.runningAllowed 标志快速停止，以防 JmsConfigurationBuildisAcceptMessagesWhileStopping 已启用，并且 org.apache.camel.CamelContext 当前已停止。默认情况下，在常规 JMS 用户中启用此快速停止功能，但要启用回复管理器，您必须启用此标志。	false	布尔值
acknowledgementMode (consumer)	JMS 确认模式定义为整数。允许您为确认模式设置特定于供应商的扩展。对于常规模式，最好改为使用 confirmmentModeName。		int

Name	描述	默认值	类型
eagerLoadingOfProperties (consumer)	加载消息时，启用强制加载 JMS 属性，这通常效率低效，因为 JMS 属性可能并不需要，但有时可以提前捕获与底层 JMS 提供程序的任何问题，以及使用 JMS 属性	false	布尔值
acknowledgementModeName (consumer)	JMS 确认名称，即 SESSION_TRANSACTED、CLIENT_ACKNOWLEDGE、AUTO_ACKNOWLEDGE, DUPS_OK_ACKNOWLEDGE	AUTO_确认	字符串
autoStartup (consumer)	指定消费者容器是否应自动启动。	true	布尔值
cacheLevel (consumer)	根据底层 JMS 资源的 ID 设置缓存级别。如需了解更多详细信息，请参阅 cacheLevelName 选项。		int
cacheLevelName (consumer)	按名称为底层 JMS 资源设置缓存级别。可能的值有：CACHE_AUTO、CACHE_CONNECTION、CACHE_CONSUMER、CACHE_NONE 和 CACHE_SESSION。默认设置为 CACHE_AUTO。如需更多信息，请参阅 Spring 文档和事务缓存级别。	CACHE_AUTO	字符串
replyToCacheLevelName (producer)	在通过 JMS 进行请求/回复时，按名称设置缓存级别。这个选项只适用于使用固定回复队列（而不是临时）。Camel 默认将使用：CACHE_CONSUMER 用于 exclusive 或 shared w/ replyToSelectorName。和 CACHE_SESSION 用于在没有 replyToSelectorName 的情况下共享。有些 JMS 代理（如 IBM WebSphere）可能需要设置 replyToCacheLevelName=CACHE_NONE 才能工作。注意：如果不使用临时队列，则不允许 CACHE_NONE，且您必须使用更高的值，如 CACHE_CONSUMER 或 CACHE_SESSION。		字符串
clientId (common)	设置要使用的 JMS 客户端 ID。请注意，如果指定，这个值必须是唯一的，且只能由单个 JMS 连接实例使用。它通常只需要 durable 主题订阅。如果使用 Apache ActiveMQ，您可能更喜欢使用虚拟主题。		字符串
concurrentConsumers (consumer)	指定从 JMS 消耗时的默认并发消费者数（而不是通过 JMS 进行请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。当通过 JMS 进行请求/回复时，会使用选项 replyToConcurrentConsumers 来控制回复消息监听器上的并发用户数量。	1	int
replyToConcurrent Consumers (producer)	指定通过 JMS 进行请求/回复时的默认并发消费者数量。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。	1	int

Name	描述	默认值	类型
ConnectionFactory (common)	要使用的连接工厂。必须在组件或端点上配置连接工厂。		ConnectionFactory
用户名 (security)	与 ConnectionFactory 搭配使用的用户名。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
密码 (security)	与 ConnectionFactory 搭配使用的密码。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
deliveryPersistent (producer)	指定是否默认使用持久性交付。	true	布尔值
deliveryMode (producer)	指定要使用的交付模式。Possibles 值是由 javax.jms.DeliveryMode 定义的值。NON_PERSISTENT = 1 和 PERSISTENT = 2.		整数
durableSubscriptionName (common)	用于指定持久主题订阅的持久化订阅者名称。还必须配置 clientId 选项。		字符串
exceptionListener (advanced)	指定正在获得任何底层 JMS 异常通知的 JMS Exception Listener。		ExceptionListener
errorHandler (advanced)	指定在处理消息时引发任何未捕获的异常时调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置 errorHandler，这些例外将在 WARN 级别记录。您可以配置日志记录级别，并使用 errorHandlerLoggingLevel 和 errorHandlerLogStackTrace 选项记录堆栈 trace。这样可以更轻松地配置，而不是对自定义 errorHandler 进行编码。		ErrorHandler
errorHandlerLoggingLevel (logging)	允许为日志记录未捕获的异常配置默认的 errorHandler 日志记录级别。	WARN	LoggingLevel
errorHandlerLogStackTrace (logging)	允许由默认的 errorHandler 控制是否应记录堆栈追踪。	true	布尔值
explicitQosEnabled (producer)	设置在发送消息时应使用 deliveryMode、priority 或 timeToLive qualities。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用到当前端点。这与 preserveMessageQos 选项相反，它以消息粒度运行，从 Camel In 消息标头读取 QoS 属性。	false	布尔值

Name	描述	默认值	类型
exposeListenerSession (consumer)	指定侦听器会话是否应该在消耗消息时公开。	false	布尔值
idleTaskExecutionLimit (advanced)	指定闲置执行接收任务的限值，没有在其执行中收到任何消息。如果达到这个限制，任务将关闭并保持接收其他执行任务（如果是动态调度，请参阅 maxConcurrentConsumers 设置）。Spring 中提供了额外的文档。	1	int
idleConsumerLimit (advanced)	指定在任意给定时间允许闲置的用户数量的限制。	1	int
maxConcurrentConsumers (consumer)	指定从 JMS 消耗时的最大并发消费者数（而不是通过 JMS 进行请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。当通过 JMS 进行请求/回复时，选项 replyToMaxConcurrentConsumers 用于控制回复消息监听器上的并发用户数量。		int
replyToMaxConcurrent Consumers (producer)	指定通过 JMS 使用请求/回复时的最大并发消费者数。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。		int
replyOnTimeoutToMax ConcurrentConsumers (producer)	指定在通过 JMS 使用请求/回复时超时时继续路由的最大并发消费者数。	1	int
maxMessagesPerTask (advanced)	每个任务的消息数量。-1 代表没有限制。如果您将范围用于并发消费者（如 min max），则可以使用此选项将值设置为 eg 100，以控制消费者在需要较少的工作时缩小的速度。	-1	int
messageConverter (advanced)	要使用自定义 Spring org.springframework.jms.support.converter.MessageConverter，以便您可以控制如何映射到 javax.jms.Message。		MessageConverter
mapJmsMessage (advanced)	指定 Camel 是否应该自动将收到的 JMS 消息映射到合适的有效负载类型，如 javax.jms.TextMessage 到 String 等。	true	布尔值
messageIdEnabled (advanced)	发送时，指定是否应添加消息 ID。这只是 JMS 代理的提示。如果 JMS 提供程序接受此提示，这些消息必须将消息 ID 设为 null；如果供应商忽略提示，则必须将消息 ID 设置为其普通唯一值	true	布尔值

Name	描述	默认值	类型
messageTimestampEnabled (advanced)	指定在发送消息时是否默认启用时间戳。这只是 JMS 代理的提示。如果 JMS 提供程序接受此提示，这些消息必须设为零；如果供应商忽略提示，则必须将时间戳设置为正常值	true	布尔值
alwaysCopyMessage (producer)	如果为 true，Camel 会在传递给制作者以进行发送时始终生成消息的 JMS 消息副本。在某些情况下需要复制消息，例如当设置了 replyToDestinationSelectorName 时（如果设置了 replyToDestinationSelectorName，Camel 会将 alwaysCopyMessage 选项设置为 true，如果设置了 replyToDestinationSelectorName ）	false	布尔值
useMessageIDAsCorrelationID (advanced)	指定 JMSMessageID 是否应该始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值
priority (producer)	大于 1 的值在发送时指定消息优先级（其中 0 是最低优先级，9 是最高）。必须启用 explicitQosEnabled 选项，才能使此选项有任何效果。	4	int
pubSubNoLocal (advanced)	指定是否禁止发送由其自身连接发布的消息。	false	布尔值
receiveTimeout (advanced)	接收消息的超时时间（以毫秒为单位）。	1000	long
recoveryInterval (advanced)	指定恢复尝试之间的间隔，例如当连接被刷新时，以毫秒为单位。默认值为 5000 ms，即 5 秒。	5000	long
taskExecutor (consumer)	允许您指定用于消耗消息的自定义任务 executor。		TaskExecutor
deliveryDelay (producer)	设置用于 JMS 发送调用的交付延迟。此选项要求 JMS 2.0 兼容代理。	-1	long
timeToLive (producer)	发送消息时，指定消息的生存时间（以毫秒为单位）。	-1	long
转换 （事务）	指定是否使用转换模式	false	布尔值
lazyCreateTransaction Manager (transaction)	如果为 true，如果选项 transacted=true 时没有注入的 transactionManager，则 Camel 将创建一个 JmsTransactionManager。	true	布尔值

Name	描述	默认值	类型
transactionManager (transaction)	要使用的 Spring 事务管理器。		平台交易管理器
transactionName (transaction)	要使用的事务的名称。		字符串
transactionTimeout (transaction)	如果使用转换模式，事务的超时值（以秒为单位）。	-1	int
testConnectionOn Startup (common)	指定是否在启动时测试连接。这样可确保当 Camel 启动所有 JMS 用户与 JMS 代理的有效连接时。如果无法授予连接，则 Camel 会在启动时抛出异常。这样可确保 Camel 不使用失败的连接启动。JMS producers 也经过测试。	false	布尔值
asyncStartListener (advanced)	在启动路由时，是否异步启动 JmsConsumer 消息监听程序。例如，如果 JmsConsumer 无法连接到远程 JMS 代理，则在重试和/或故障转移时可能会阻止。这将导致 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 以异步模式使用专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果无法建立连接，那么会记录异常在 WARN 级别，消费者将无法接收消息；然后您可以重启路由来重试。	false	布尔值
asyncStopListener (advanced)	在停止路由时，是否异步停止 JmsConsumer 消息监听程序。	false	布尔值
forceSendOriginal Message (producer)	使用 mapJmsMessage=false Camel 时，如果您涉及路由期间的标头(get 或 set)，则会创建一个新的 JMS 消息来发送到新的 JMS 目的地。将此选项设置为 true，以强制 Camel 发送收到的原始 JMS 消息。	false	布尔值
requestTimeout (producer)	使用 InOut Exchange Pattern 时等待回复的超时（以毫秒为单位）。默认值为 20 秒。您可以包含标头 CamelJmsRequestTimeout 来覆盖此端点配置的超时值，因此每个消息都有单独的超时值。另请参阅 requestTimeoutCheckerInterval 选项。	20000	long
requestTimeoutChecker Interval (advanced)	配置 Camel 在通过 JMS 进行请求/回复时应检查超时的频率。默认情况下，Camel 检查每秒一次。但是，如果发生超时时必须更快地响应，则可以降低这个间隔，以便更频繁地检查。超时由 option requestTimeout 决定。	1000	long

Name	描述	默认值	类型
transferExchange (advanced)	您可以通过线线传输交换，而不只是正文和标头。以下字段会被传输： 在 body, Out body, Fault body, In headers, Out headers, Fault 标头, Exchange properties, Exchange exception。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。您必须在生成者和消费者端启用这个选项，因此 Camel 知道有效负载是交换而不是常规有效负载。	false	布尔值
transferException (advanced)	如果启用并且您使用 Request Reply messaging (InOut)和在消费者端的 Exchange 失败，则原因 Exception 将发回为 javax.jms.ObjectMessage 的响应。如果客户端是 Camel，则返回的 Exception 为 rerown。这允许您使用 Camel JMS 作为路由中的网桥，例如，使用持久性队列来启用强大的路由。请注意，如果您也启用了 transferExchange，这个选项会优先使用。需要被捕获的例外是串行的。在返回到生成者时，消费者侧的原始 Exception 可以嵌套在外部异常中，如 org.apache.camel.RuntimeCamelException。	false	布尔值
transferFault (advanced)	如果启用且您使用 Request Reply messaging (InOut)，且交换失败，在消费者端使用 SOAP 错误（非例外），则 Message SerialisFault （）上的错误标志将重新作为带有键 org.apache.camel.component.jms.JmsConstants SerialJMS_TRANSFER_FAULT:JMS_TRANSFER_FAULT 的 JMS 标头发回。如果客户端是 Camel，则返回的 fault 标志将在 org.apache.camel.MessageAllsetFault (boolean)上设置。在使用支持错误（如 SOAP）（如 cxf 或 spring-ws）的 Camel 组件时，您可能需要启用此功能。	false	布尔值
jmsOperations (advanced)	允许您使用您自己的 org.springframework.jms.core.JmsOperations 接口实施。Camel 使用 JmsTemplate 作为默认值。可用于测试目的，但不按 spring API 文档中所述使用。		JmsOperations
destinationResolver (advanced)	可插拔 org.springframework.jms.support.destination.DestinationResolver，允许您使用您自己的解析器（例如，在 JNDI 注册表中查找实际目的地）。		DestinationResolver

Name	描述	默认值	类型
replyToType (producer)	允许明确指定在通过 JMS 进行请求/回复时，用于 replyTo 队列的策略。可能的值有： Temporary、Shared 或 Exclusive。默认情况下，Camel 将使用临时队列。但是，如果配置了 replyTo，则默认使用 Shared。此选项允许您使用专用队列而不是共享队列。如需了解更多详细信息，请参阅 Camel JMS 文档，特别是有关在集群环境中运行时所影响的备注，而共享回复队列的性能比其 alternatives Temporary 和 Exclusive 性能较低。		ReplyToType
preserveMessageQos (producer)	如果要使用消息上指定的 QoS 设置发送消息，而不是 JMS 端点上的 QoS 设置，则设置为 true。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供所有或只提供其中一些。如果没有提供，Camel 将回退到使用端点中的值。因此，在使用此选项时，标头会覆盖来自端点的值。相反，explicitQosEnabled 选项只会使用端点上设置的选项，而不使用消息标头中设置的值。	false	布尔值
asyncConsumer (consumer)	JmsConsumer 是否异步处理 Exchange。如果启用，JmsConsumer 可能会从 JMS 队列中选择下一个消息，而前面的消息异步处理（通过 Asynchronous Routing Engine）。这意味着消息可能没有完全严格按照顺序进行处理。如果禁用（作为默认），则在 JmsConsumer 将从 JMS 队列获取下一个消息之前完全处理 Exchange。请注意，如果启用了 transacted，则 asyncConsumer=true 不会异步运行，因为事务必须同步执行(Camel 3.0 可能支持 async 事务)。	false	布尔值
allowNullBody (producer)	是否允许发送没有正文的消息。如果此选项为 false，并且消息正文为 null，则抛出 JMSException。	true	布尔值
includeSentJMSMessageID (producer)	仅在使用 InOnly （如触发和忘记）发送到 JMS 目的地时才适用。启用此选项将使用实际的 JMSMessageID 增强 Camel Exchange，当消息发送到 JMS 目的地时，由 JMS 客户端使用。	false	布尔值
includeAllJMSXProperties (advanced)	从 JMS 映射到 Camel 消息时，是否要包含所有 JMSXxxx 属性。把它设置为 true 将包括 JMSXAppID 和 JMSXUserID 等属性。注： 如果您使用自定义 headerFilterStrategy，则不会应用这个选项。	false	布尔值

Name	描述	默认值	类型
defaultTaskExecutor Type (consumer)	指定在 DefaultMessageListenerContainer 中使用的默认 TaskExecutor 类型，用于消费者端点和生成者端点的 ReplyTo consumer。可能的值： SimpleAsync（使用 Spring 的 SimpleAsyncTaskExecutor）或 ThreadPool （使用 Spring 的 ThreadPoolTaskExecutor 带有最佳值 - 缓存线程池）。如果没有设置，则默认为之前的行为，它使用缓存的线程池作为消费者端点，而 SimpleAsync 用于回复消费者。建议使用 ThreadPool，以动态增加和减少并发消费者在弹性配置中减少线程回收。		DefaultTaskExecutor Type
jmsKeyFormatStrategy (advanced)	可插拔编码和解码 JMS 密钥策略，以便它们能够与 JMS 规范兼容。Camel 提供了两个开箱即用的实现：default 和 passthrough。默认策略可以安全地放入点和连字符(. 和 -)。passthrough 策略将密钥保留原样。可用于 JMS 代理，其不小心是 JMS 标头键是否包含非法字符。您可以提供自己的 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 # 表示法引用它。		JmsKeyFormatStrategy
allowAdditionalHeaders (producer)	此选项用于允许其他标头，这些标头可能具有根据 JMS 规格无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 包含带有字节数组或其他无效类型的值的标头名称。您可以指定用逗号分开的多个标头名称，并使用 作为后缀进行通配符匹配。		字符串
queueBrowseStrategy (advanced)	在浏览队列时使用自定义 QueueBrowseStrategy		QueueBrowseStrategy
messageCreatedStrategy (advanced)	要使用给定的 MessageCreatedStrategy，后者在 Camel 在 Camel 发送 JMS 消息时创建 javax.jms.Message 对象的新实例。		MessageCreatedStrategy
waitForProvisionCorrelationToBeUpdated Counter (advanced)	在通过 JMS 进行请求/回复时，以及启用选项 useMessageIDAsCorrelationID 时，等待 provisional correlation id 的次数。	50	int
waitForProvisionCorrelationToBeUpdated ThreadSleepingTime (advanced)	millis 中的间隔，在每次等待配置关联 ID 时处于睡眠状态。	100	long

Name	描述	默认值	类型
correlationProperty (producer)	使用此 JMS 属性来关联 InOut Exchange pattern (request-reply)中的消息，而不是 JMSCorrelationID 属性。这样，您可以将消息与不使用 JMSCorrelationID JMS 属性关联消息的系统交换。如果使用的 JMSCorrelationID 不会被 Camel 使用或设置。如果没有在相同名称下的消息标头中提供，则会生成此处 named 属性的值。		字符串
subscriptionDurable (consumer)	设置是否使订阅持久。要使用的持久订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册持久化订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。仅在侦听主题(pub-sub domain)时具有意义，因此此方法也会切换 pubSubDomain 标志。	false	布尔值
subscriptionShared (consumer)	设置是否使订阅共享。要使用的共享订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。请注意，共享订阅也可能是持久的，因此此标志也可以与 subscriptionDurable 结合使用。仅在侦听主题(pub-sub domain)时具有意义，因此此方法也会切换 pubSubDomain 标志。需要兼容 JMS 2.0 的消息代理。	false	布尔值
subscriptionName (consumer)	设置要创建的订阅的名称。当具有共享或持久化订阅的主题(pub-sub domain)时应用。订阅名称需要在此客户端的 JMS 客户端 id 中唯一。default 是指定消息监听程序的类名称。注意：除了共享订阅（需要 JMS 2.0）外，每个订阅只允许 1 个并发消费者（此消息监听程序容器的默认值）。		字符串
streamMessageType Enabled (producer)	设置是否启用 StreamMessage 类型。文件、InputStream 等消息有效负载（如文件、InputStream 等）将通过 BytesMessage 或 StreamMessage 发送。此选项控制将使用哪种类型。默认情况下，使用 BytesMessage 来强制将整个消息有效负载读取到内存中。通过启用此选项，消息有效负载以块的形式读取内存，然后每个块写入 StreamMessage，直到没有更多数据。	false	布尔值
formatDateHeadersTo Iso8601 (producer)	设置日期标头是否应根据 ISO 8601 标准进行格式化。	false	布尔值
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AMQP 端点使用 URI 语法进行配置：

```
amqp:destinationType:destinationName
```

使用以下路径和查询参数：

5.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
destinationType	要使用的目的地类型	queue	字符串
destinationName	用作目的地的队列或主题的 **必需** 名称		字符串

5.2.2. 查询参数(92 参数)：

Name	描述	默认值	类型
clientId (common)	设置要使用的 JMS 客户端 ID。请注意，如果指定，这个值必须是唯一的，且只能由单个 JMS 连接实例使用。它通常只需要 durable 主题订阅。如果使用 Apache ActiveMQ，您可能更喜欢使用虚拟主题。		字符串
ConnectionFactory (common)	要使用的连接工厂。必须在组件或端点上配置连接工厂。		ConnectionFactory
disableReplyTo (common)	指定 Camel 是否忽略消息中的 JMSReplyTo 标头。如果为 true，Camel 不会发送回复回 JMSReplyTo 标头中指定的目的地。如果您希望 Camel 从路由消耗且您不希望 Camel 自动发送回复消息，因为代码中的另一个组件处理回复消息，则可以使用这个选项。如果要使用 Camel 作为不同消息代理之间的代理，并希望将消息从一个系统路由到另一个系统，也可以使用这个选项。	false	布尔值
durableSubscriptionName (common)	用于指定持久主题订阅的持久化订阅者名称。还必须配置 clientId 选项。		字符串

Name	描述	默认值	类型
jmsMessageType (common)	允许您强制使用特定的 javax.jms.Message 实施来发送 JMS 消息。可能的值有：Bytes, Map, Object, Stream, Text。默认情况下，Camel 将决定要从 In 正文类型使用哪个 JMS 消息类型。这个选项允许您指定它。		JmsMessageType
testConnectionOnStartup (common)	指定是否在启动时测试连接。这样可确保当 Camel 启动所有 JMS 用户与 JMS 代理的有效连接时。如果无法授予连接，则 Camel 会在启动时抛出异常。这样可确保 Camel 不使用失败的连接启动。JMS producers 也经过测试。	false	布尔值
acknowledgementModeName (consumer)	JMS 确认名称，即 SESSION_TRANSACTED、CLIENT_ACKNOWLEDGE、AUTO_ACKNOWLEDGE, DUPS_OK_ACKNOWLEDGE	AUTO_确认	字符串
asyncConsumer (consumer)	JmsConsumer 是否异步处理 Exchange。如果启用，JmsConsumer 可能会从 JMS 队列中选择下一个消息，而前面的消息异步处理（通过 Asynchronous Routing Engine）。这意味着消息可能没有完全严格按照顺序进行处理。如果禁用（作为默认），则在 JmsConsumer 将从 JMS 队列获取下一个消息之前完全处理 Exchange。请注意，如果启用了 transacted，则 asyncConsumer=true 不会异步运行，因为事务必须同步执行(Camel 3.0 可能支持 async 事务)。	false	布尔值
autoStartup (consumer)	指定消费者容器是否应自动启动。	true	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
cacheLevel (consumer)	根据底层 JMS 资源的 ID 设置缓存级别。如需了解更多详细信息，请参阅 cacheLevelName 选项。		int
cacheLevelName (consumer)	按名称为底层 JMS 资源设置缓存级别。可能的值有：CACHE_AUTO、CACHE_CONNECTION、CACHE_CONSUMER、CACHE_NONE 和 CACHE_SESSION。默认设置为 CACHE_AUTO。如需更多信息，请参阅 Spring 文档和事务缓存级别。	CACHE_AUTO	字符串

Name	描述	默认值	类型
concurrentConsumers (consumer)	指定从 JMS 消耗时的默认并发消费者数（而不是通过 JMS 进行请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。当通过 JMS 进行请求/回复时，会使用选项 replyToConcurrentConsumers 来控制回复消息监听器上的并发用户数量。	1	int
maxConcurrentConsumers (consumer)	指定从 JMS 消耗时的最大并发消费者数（而不是通过 JMS 进行请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。当通过 JMS 进行请求/回复时，选项 replyToMaxConcurrentConsumers 用于控制回复消息监听器上的并发用户数量。		int
replyTo (consumer)	提供一个显式 ReplyTo 目的地，它会覆盖任何 Message.getJMSReplyTo （）的任何传入值。		字符串
replyToDeliveryPersistent (consumer)	指定默认情况下是否使用持久发送回复。	true	布尔值
selector (consumer)	设置要使用的 JMS 选择器		字符串
subscriptionDurable (consumer)	设置是否使订阅持久。要使用的持久订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册持久化订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。仅在侦听主题(pub-sub domain)时具有意义，因此此方法也会切换 pubSubDomain 标志。	false	布尔值
subscriptionName (consumer)	设置要创建的订阅的名称。当具有共享或持久化订阅的主题(pub-sub domain)时应用。订阅名称需要在此客户端的 JMS 客户端 id 中唯一。default 是指定消息监听程序的类名称。注意：除了共享订阅（需要 JMS 2.0）外，每个订阅只允许 1 个并发消费者（此消息监听程序容器的默认值）。		字符串

Name	描述	默认值	类型
subscriptionShared (consumer)	设置是否使订阅共享。要使用的共享订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。请注意，共享订阅也可能是持久的，因此此标志也可以与 subscriptionDurable 结合使用。仅在侦听主题(pub-sub domain)时具有意义，因此此方法也会切换 pubSubDomain 标志。需要兼容 JMS 2.0 的消息代理。	false	布尔值
acceptMessagesWhileStopping (consumer)	指定消费者在停止时是否接受消息。如果您在运行时启动和停止 JMS 路由，同时队列中仍然有消息排队，您可以考虑启用此选项。如果此选项为 false，并且您停止 JMS 路由，则消息可能会被拒绝，并且 JMS 代理必须尝试恢复，并且再次可能被拒绝，最后该消息可能在 JMS 代理的死信队列中移动。为避免这种情况，建议启用这个选项。	false	布尔值
allowReplyManagerQuickStop (consumer)	在 request-reply 消息传递的回复管理器中使用的 DefaultMessageListenerContainer 是否允许 DefaultMessageListenerContainer.runningAllowed 标志快速停止，以防 JmsConfigurationBuildisAcceptMessagesWhileStopping 已启用，并且 org.apache.camel.CamelContext 当前已停止。默认情况下，在常规 JMS 用户中启用此快速停止功能，但要启用回复管理器，您必须启用此标志。	false	布尔值
consumerType (consumer)	要使用的消费者类型，可以是： Simple、Default 或 Custom 之一。consumer 类型决定要使用的 Spring JMS 侦听器。default 将使用 org.springframework.jms.listener.DefaultMessageListenerContainer，Simple 将使用 org.springframework.jms.listener.SimpleMessageListenerContainer。当指定 Custom 时，messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 将决定要使用的 org.springframework.jms.listener.AbstractMessageListenerContainer。	default	ConsumerType

Name	描述	默认值	类型
defaultTaskExecutorType (consumer)	指定在 DefaultMessageListenerContainer 中使用的默认 TaskExecutor 类型，用于消费者端点和生成者端点的 ReplyTo consumer。可能的值： SimpleAsync（使用 Spring 的 SimpleAsyncTaskExecutor）或 ThreadPool （使用 Spring 的 ThreadPoolTaskExecutor 带有最佳值 - 缓存线程池）。如果没有设置，则默认为之前的行为，它使用缓存的线程池作为消费者端点，而 SimpleAsync 用于回复消费者。建议使用 ThreadPool，以动态增加和减少并发消费者在弹性配置中减少线程回收。		DefaultTaskExecutor Type
eagerLoadingOfProperties (consumer)	在加载消息时，启用 JMS 属性和有效负载的强制加载，这通常效率低效，因为 JMS 属性可能并不是必需的，但有时可以在底层 JMS 提供程序和 JMS 属性中使用任何问题时捕获早期出现的问题	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
exposeListenerSession (consumer)	指定侦听器会话是否应该在消耗消息时公开。	false	布尔值
replyToSameDestination Allowed (consumer)	是否允许 JMS 使用者向消费者使用的同一目的地发送回复消息。这可防止通过消耗并向其自身发送相同的消息，从而防止无限循环。	false	布尔值
taskExecutor (consumer)	允许您指定用于消耗消息的自定义任务 executor。		TaskExecutor
deliveryDelay (producer)	设置用于 JMS 发送调用的交付延迟。此选项要求 JMS 2.0 兼容代理。	-1	long
deliveryMode (producer)	指定要使用的交付模式。Possibles 值是由 javax.jms.DeliveryMode 定义的值。NON_PERSISTENT = 1 和 PERSISTENT = 2.		整数
deliveryPersistent (producer)	指定是否默认使用持久性交付。	true	布尔值

Name	描述	默认值	类型
explicitQosEnabled (producer)	设置在发送消息时应使用 deliveryMode、priority 或 timeToLive qualities。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用到当前端点。这与 preserveMessageQos 选项相反，它以消息粒度运行，从 Camel In 消息标头读取 QoS 属性。	false	布尔值
formatDateHeadersToIso8601 (producer)	设置 JMS date 属性是否应根据 ISO 8601 标准进行格式化。	false	布尔值
preserveMessageQos (producer)	如果要使用消息上指定的 QoS 设置发送消息，而不是 JMS 端点上的 QoS 设置，则设置为 true。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供所有或只提供其中一些。如果没有提供，Camel 将回退到使用端点中的值。因此，在使用此选项时，标头会覆盖来自端点的值。相反，explicitQosEnabled 选项只会使用端点上设置的选项，而不使用消息标头中设置的值。	false	布尔值
priority (producer)	大于 1 的值在发送时指定消息优先级（其中 0 是最低优先级，9 是最高）。必须启用 explicitQosEnabled 选项，才能使此选项有任何效果。	4	int
replyToConcurrentConsumers (producer)	指定通过 JMS 进行请求/回复时的默认并发消费者数量。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。	1	int
replyToMaxConcurrent Consumers (producer)	指定通过 JMS 使用请求/回复时的最大并发消费者数。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。		int
replyToOnTimeoutMax ConcurrentConsumers (producer)	指定在通过 JMS 使用请求/回复时超时时继续路由的最大并发消费者数。	1	int
replyToOverride (producer)	在 JMS 消息中提供一个显式 ReplyTo 目的地，它将覆盖 replyTo 的设置。如果您要将消息转发到远程 Queue，并从 ReplyTo 目的地接收回复消息，这非常有用。		字符串

Name	描述	默认值	类型
replyToType (producer)	允许明确指定在通过 JMS 进行请求/回复时，用于 replyTo 队列的策略。可能的值有： Temporary、Shared 或 Exclusive。默认情况下，Camel 将使用临时队列。但是，如果配置了 replyTo，则默认使用 Shared。此选项允许您使用专用队列而不是共享队列。如需了解更多详细信息，请参阅 Camel JMS 文档，特别是有关在集群环境中运行时所影响的备注，而共享回复队列的性能比其 alternatives Temporary 和 Exclusive 性能较低。		ReplyToType
requestTimeout (producer)	使用 InOut Exchange Pattern 时等待回复的超时（以毫秒为单位）。默认值为 20 秒。您可以包含标头 CamelJmsRequestTimeout 来覆盖此端点配置的超时值，因此每个消息都有单独的超时值。另请参阅 requestTimeoutCheckerInterval 选项。	20000	long
timeToLive (producer)	发送消息时，指定消息的生存时间（以毫秒为单位）。	-1	long
allowAdditionalHeaders (producer)	此选项用于允许其他标头，这些标头可能具有根据 JMS 规格无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 包含带有字节数组或其他无效类型的值的标头名称。您可以指定用逗号分开的多个标头名称，并使用 作为后缀进行通配符匹配。		字符串
allowNullBody (producer)	是否允许发送没有正文的消息。如果此选项为 false，并且消息正文为 null，则抛出 JMSException。	true	布尔值
alwaysCopyMessage (producer)	如果为 true，Camel 会在传递给制作者以进行发送时始终生成消息的 JMS 消息副本。在某些情况下需要复制消息，例如当设置了 replyToDestinationSelectorName 时（如果设置了 replyToDestinationSelectorName，Camel 会将 alwaysCopyMessage 选项设置为 true，如果设置了 replyToDestinationSelectorName ）	false	布尔值
correlationProperty (producer)	使用 InOut 交换模式时，使用此 JMS 属性而不是 JMSCorrelationID JMS 属性来关联消息。如果设置的消息将只与此属性 JMSCorrelationID 属性的值关联，则不会被 Camel 忽略，且不会由 Camel 设置。		字符串

Name	描述	默认值	类型
disableTimeToLive (producer)	使用这个选项强制禁用生存时间。例如，当您通过 JMS 进行请求/回复时，Camel 默认将使用 requestTimeout 值作为所发送消息的时间。问题是发送者和接收器系统必须同步其时钟，因此它们处于同步状态。这并非总是容易进行归档。因此，您可以使用 disableTimeToLive=true 设置在发送的消息上生存时间值。然后，消息不会在接收方系统中过期。如需了解更多详细信息，请参见关于生存时间 一节中的。	false	布尔值
forceSendOriginalMessage (producer)	使用 mapJmsMessage=false Camel 时，如果您涉及路由期间的标头(get 或 set)，则会创建一个新的 JMS 消息来发送到新的 JMS 目的地。将此选项设置为 true，以强制 Camel 发送收到的原始 JMS 消息。	false	布尔值
includeSentJMSMessageID (producer)	仅在使用 InOnly （如触发和忘记）发送到 JMS 目的地时才适用。启用此选项将使用实际的 JMSMessageID 增强 Camel Exchange，当消息发送到 JMS 目的地时，由 JMS 客户端使用。	false	布尔值
replyToCacheLevelName (producer)	在通过 JMS 进行请求/回复时，按名称设置缓存级别。这个选项只适用于使用固定回复队列（而不是临时）。Camel 默认将使用：CACHE_CONSUMER 用于 exclusive 或 shared w/ replyToSelectorName。和 CACHE_SESSION 用于在没有 replyToSelectorName 的情况下共享。有些 JMS 代理（如 IBM WebSphere）可能需要设置 replyToCacheLevelName=CACHE_NONE 才能工作。注意：如果不使用临时队列，则不允许 CACHE_NONE，且您必须使用更高的值，如 CACHE_CONSUMER 或 CACHE_SESSION。		字符串
replyToDestinationSelector Name (producer)	使用固定名称设置 JMS Selector，以便在使用共享队列时过滤您自己的回复信息（也就是说，如果您不使用临时回复队列）。		字符串
streamMessageTypeEnabled (producer)	设置是否启用 StreamMessage 类型。文件、InputStream 等消息有效负载（如文件、InputStream 等）将通过 BytesMessage 或 StreamMessage 发送。此选项控制将使用哪种类型。默认情况下，使用 BytesMessage 来强制将整个消息有效负载读取到内存中。通过启用此选项，消息有效负载以块的形式读取内存，然后每个块写入 StreamMessage，直到没有更多数据。	false	布尔值
allowSerializedHeaders (advanced)	控制是否包含序列化标头。仅在 transferExchange 为 true 时才适用。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。	false	布尔值

Name	描述	默认值	类型
asyncStartListener (advanced)	在启动路由时，是否异步启动 JmsConsumer 消息监听程序。例如，如果 JmsConsumer 无法连接到远程 JMS 代理，则在重试和/或故障转移时可能会阻止。这将导致 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 以异步模式使用专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果无法建立连接，那么会记录异常在 WARN 级别，消费者将无法接收消息；然后您可以重启路由来重试。	false	布尔值
asyncStopListener (advanced)	在停止路由时，是否异步停止 JmsConsumer 消息监听程序。	false	布尔值
destinationResolver (advanced)	可插拔 org.springframework.jms.support.destination.DestinationResolver，允许您使用您自己的解析器（例如，在 JNDI 注册表中查找实际目的地）。		DestinationResolver
errorHandler (advanced)	指定在处理消息时引发任何未捕获的异常时调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置 errorHandler，这些例外将在 WARN 级别记录。您可以配置日志记录级别，并使用 errorHandlerLoggingLevel 和 errorHandlerLogStackTrace 选项记录堆栈 trace。这样可以更轻松地配置，而不是对自定义 errorHandler 进行编码。		ErrorHandler
exceptionListener (advanced)	指定正在获得任何底层 JMS 异常通知的 JMS Exception Listener。		ExceptionListener
headerFilterStrategy (advanced)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
idleConsumerLimit (advanced)	指定在任意给定时间允许闲置的用户数量的限制。	1	int
idleTaskExecutionLimit (advanced)	指定闲置执行接收任务的限值，没有在其执行中收到任何消息。如果达到这个限制，任务将关闭并保持接收其他执行任务（如果是动态调度，请参阅 maxConcurrentConsumers 设置）。Spring 中提供了额外的文档。	1	int
includeAllJMSXProperties (advanced)	从 JMS 映射到 Camel 消息时，是否要包含所有 JMSXxxx 属性。把它设置为 true 将包括 JMSXAppID 和 JMSXUserID 等属性。注： 如果您使用自定义 headerFilterStrategy，则不会应用这个选项。	false	布尔值

Name	描述	默认值	类型
jmsKeyFormatStrategy (advanced)	可插拔编码和解码 JMS 密钥策略，以便它们能够与 JMS 规范兼容。Camel 提供了两个开箱即用的实现：default 和 passthrough。默认策略可以安全地放入点和连字符(. 和 -)。passthrough 策略将密钥保留原样。可用于 JMS 代理，其不小心是 JMS 标头键是否包含非法字符。您可以提供自己的 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 # 表示法引用它。		字符串
mapJmsMessage (advanced)	指定 Camel 是否应该自动将收到的 JMS 消息映射到合适的有效负载类型，如 javax.jms.TextMessage 到 String 等。	true	布尔值
maxMessagesPerTask (advanced)	每个任务的消息数量。-1 代表没有限制。如果您将范围用于并发消费者（如 min max），则可以使用此选项将值设置为 eg 100，以控制消费者在需要较少的工作时缩小的速度。	-1	int
messageConverter (advanced)	要使用自定义 Spring org.springframework.jms.support.converter.MessageConverter，以便您可以控制如何映射到 javax.jms.Message。		MessageConverter
messageCreatedStrategy (advanced)	要使用给定的 MessageCreatedStrategy，后者在 Camel 在 Camel 发送 JMS 消息时创建 javax.jms.Message 对象的新实例。		MessageCreatedStrategy
messageIdEnabled (advanced)	发送时，指定是否应添加消息 ID。这只是 JMS 代理的提示。如果 JMS 提供程序接受此提示，这些消息必须将消息 ID 设为 null；如果供应商忽略提示，则必须将消息 ID 设置为其普通唯一值	true	布尔值
messageListenerContainerFactory (advanced)	用于确定用于使用消息的 org.springframework.jms.listener.AbstractMessageListenerContainer 的 MessageListenerContainer 的 registry ID。设置此选项将自动将 consumerType 设置为 Custom。		MessageListenerContainerFactory
messageTimestampEnabled (advanced)	指定在发送消息时是否默认启用时间戳。这只是 JMS 代理的提示。如果 JMS 提供程序接受此提示，这些消息必须设为零；如果供应商忽略提示，则必须将时间戳设置为正常值	true	布尔值
pubSubNoLocal (advanced)	指定是否禁止发送由其自身连接发布的消息。	false	布尔值

Name	描述	默认值	类型
receiveTimeout (advanced)	接收消息的超时时间（以毫秒为单位）。	1000	long
recoveryInterval (advanced)	指定恢复尝试之间的间隔，例如当连接被刷新时，以毫秒为单位。默认值为 5000 ms，即 5 秒。	5000	long
requestTimeoutChecker Interval (advanced)	配置 Camel 在通过 JMS 进行请求/回复时应检查超时的频率。默认情况下，Camel 检查每秒一次。但是，如果发生超时时必须更快地响应，则可以降低这个间隔，以便更频繁地检查。超时由 option requestTimeout 决定。	1000	long
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
transferException (advanced)	如果启用并且您使用 Request Reply messaging (InOut)和在消费者端的 Exchange 失败，则原因 Exception 将发回为 javax.jms.ObjectMessage 的响应。如果客户端是 Camel，则返回的 Exception 为 rerown。这允许您使用 Camel JMS 作为路由中的网桥，例如，使用持久性队列来启用强大的路由。请注意，如果您也启用了 transferExchange，这个选项会优先使用。需要被捕获的例外是串行的。在返回到生成者时，消费者侧的原始 Exception 可以嵌套在外部异常中，如 org.apache.camel.RuntimeCamelException。	false	布尔值
transferExchange (advanced)	您可以通过线线传输交换，而不只是正文和标头。以下字段会被传输： 在 body, Out body, Fault body, In headers, Out headers, Fault 标头, Exchange properties, Exchange exception。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。您必须在生成者和消费者端启用这个选项，因此 Camel 知道有效负载是交换而不是常规有效负载。	false	布尔值
transferFault (advanced)	如果启用且您使用 Request Reply messaging (InOut)，且交换失败，在消费者端使用 SOAP 错误（非例外），则 Message SerialisFault （）上的错误标志将重新作为带有键 org.apache.camel.component.jms.JmsConstants SerialJMS_TRANSFER_FAULT:JMS_TRANSFER_FAULT 的 JMS 标头发回。如果客户端是 Camel，则返回的 fault 标志将在 org.apache.camel.MessageAllsetFault (boolean)上设置。在使用支持错误（如 SOAP）（如 cxf 或 spring-ws）的 Camel 组件时，您可能需要启用此功能。	false	布尔值

Name	描述	默认值	类型
useMessageIDAsCorrelation ID (advanced)	指定 JMSMessageID 是否应该始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值
waitForProvisionCorrelationToBeUpdatedCounter (advanced)	在通过 JMS 进行请求/回复时，以及启用选项 useMessageIDAsCorrelationID 时，等待 provisional correlation id 的次数。	50	int
waitForProvisionCorrelationToBeUpdatedThreadSleeping Time (advanced)	millis 中的间隔，在每次等待配置关联 ID 时处于睡眠状态。	100	long
errorHandlerLoggingLevel (logging)	允许为日志记录未捕获的异常配置默认的 errorHandler 日志记录级别。	WARN	LoggingLevel
errorHandlerLogStackTrace (logging)	允许由默认的 errorHandler 控制是否应记录堆栈追踪。	true	布尔值
密码 (security)	与 ConnectionFactory 搭配使用的密码。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
用户名 (security)	与 ConnectionFactory 搭配使用的用户名。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
转换 （事务）	指定是否使用转换模式	false	布尔值
lazyCreateTransaction Manager (transaction)	如果为 true，如果选项 transacted=true 时没有注入的 transactionManager，则 Camel 将创建一个 JmsTransactionManager。	true	布尔值
transactionManager (transaction)	要使用的 Spring 事务管理器。		平台交易管理器
transactionName (transaction)	要使用的事务的名称。		字符串
transactionTimeout (transaction)	如果使用转换模式，事务的超时值（以秒为单位）。	-1	int

5.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 81 选项，如下所列。

Name	描述	默认值	类型
camel.component.amqp.accept-messages-while-stopping	指定消费者在停止时是否接受消息。如果您在运行时启动和停止 JMS 路由，同时队列中仍然有消息排队，您可以考虑启用此选项。如果此选项为 false，并且您停止 JMS 路由，则消息可能会被拒绝，并且 JMS 代理必须尝试恢复，并且再次可能被拒绝，最后该消息可能在 JMS 代理的死信队列中移动。为避免这种情况，建议启用这个选项。	false	布尔值
camel.component.amqp.acknowledgement-mode	JMS 确认模式定义为整数。允许您为确认模式设置特定于供应商的扩展。对于常规模式，最好改为使用 confirmmentModeName。		整数
camel.component.amqp.acknowledgement-mode-name	JMS 确认名称，即 SESSION_TRANSACTED、CLIENT_ACKNOWLEDGE、AUTO_ACKNOWLEDGE, DUPS_OK_ACKNOWLEDGE	AUTO_确认	字符串
camel.component.amqp.allow-additional-headers	此选项用于允许其他标头，这些标头可能具有根据 JMS 规格无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 包含带有字节数组或其他无效类型的值的标头名称。您可以指定用逗号分开的多个标头名称，并使用 作为后缀进行通配符匹配。		字符串
camel.component.amqp.allow-null-body	是否允许发送没有正文的消息。如果此选项为 false，并且消息正文为 null，则抛出 JMSException。	true	布尔值
camel.component.amqp.allow-reply-manager-quick-stop	在 request-reply 消息传递的回复管理器中使用的 DefaultMessageListenerContainer 是否允许 DefaultMessageListenerContainer.runningAllowed 标志快速停止，以防 JmsConfigurationBuildisAcceptMessagesWhileStopping 已启用，并且 org.apache.camel.CamelContext 当前已停止。默认情况下，在常规 JMS 用户中启用此快速停止功能，但要启用回复管理器，您必须启用此标志。	false	布尔值
camel.component.amqp.always-copy-message	如果为 true，Camel 会在传递给制作者以进行发送时始终生成消息的 JMS 消息副本。在某些情况下需要复制消息，例如当设置了 replyToDestinationSelectorName 时（如果设置了 replyToDestinationSelectorName，Camel 会将 alwaysCopyMessage 选项设置为 true，如果设置了 replyToDestinationSelectorName ）	false	布尔值

Name	描述	默认值	类型
camel.component.amqp.async-consumer	JmsConsumer 是否异步处理 Exchange。如果启用，JmsConsumer 可能会从 JMS 队列中选择下一个消息，而前面的消息异步处理（通过 Asynchronous Routing Engine）。这意味着消息可能没有完全严格按照顺序进行处理。如果禁用（作为默认），则在 JmsConsumer 将从 JMS 队列获取下一个消息之前完全处理 Exchange。请注意，如果启用了 transacted，则 asyncConsumer=true 不会异步运行，因为事务必须同步执行(Camel 3.0 可能支持 async 事务)。	false	布尔值
camel.component.amqp.async-start-listener	在启动路由时，是否异步启动 JmsConsumer 消息监听程序。例如，如果 JmsConsumer 无法连接到远程 JMS 代理，则在重试和/或故障转移时可能会阻止。这将导致 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 以异步模式使用专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果无法建立连接，那么会记录异常在 WARN 级别，消费者将无法接收消息；然后您可以重启路由来重试。	false	布尔值
camel.component.amqp.async-stop-listener	在停止路由时，是否异步停止 JmsConsumer 消息监听程序。	false	布尔值
camel.component.amqp.auto-startup	指定消费者容器是否应自动启动。	true	布尔值
camel.component.amqp.cache-level	根据底层 JMS 资源的 ID 设置缓存级别。如需了解更多详细信息，请参阅 cacheLevelName 选项。		整数
camel.component.amqp.cache-level-name	按名称为底层 JMS 资源设置缓存级别。可能的值有：CACHE_AUTO、CACHE_CONNECTION、CACHE_CONSUMER、CACHE_NONE 和 CACHE_SESSION。默认设置为 CACHE_AUTO。如需更多信息，请参阅 Spring 文档和事务缓存级别。	CACHE_AUTO	字符串
camel.component.amqp.client-id	设置要使用的 JMS 客户端 ID。请注意，如果指定，这个值必须是唯一的，且只能由单个 JMS 连接实例使用。它通常只需要 durable 主题订阅。如果使用 Apache ActiveMQ，您可能更喜欢使用虚拟主题。		字符串
camel.component.amqp.concurrent-consumers	指定从 JMS 消耗时的默认并发消费者数（而不是通过 JMS 进行请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。当通过 JMS 进行请求/回复时，会使用选项 replyToConcurrentConsumers 来控制回复消息监听器上的并发用户数量。	1	整数

Name	描述	默认值	类型
camel.component.amqp.configuration	使用共享的 JMS 配置。选项是 org.apache.camel.component.jms.JmsConfiguration 类型。		字符串
camel.component.amqp.connection-factory	要使用的连接工厂。必须在组件或端点上配置连接工厂。选项是 javax.jms.ConnectionFactory 类型。		字符串
camel.component.amqp.correlation-property	使用此 JMS 属性来关联 InOut Exchange pattern (request-reply)中的消息，而不是 JMSCorrelationID 属性。这样，您可以将消息与不使用 JMSCorrelationID JMS 属性关联消息的系统交换。如果使用的 JMSCorrelationID 不会被 Camel 使用或设置。如果没有在相同名称下的消息标头中提供，则会生成此处 named 属性的值。		字符串
camel.component.amqp.default-task-executor-type	指定在 DefaultMessageListenerContainer 中使用的默认 TaskExecutor 类型，用于消费者端点和生成者端点的 ReplyTo consumer。可能的值： SimpleAsync （使用 Spring 的 SimpleAsyncTaskExecutor）或 ThreadPool （使用 Spring 的 ThreadPoolTaskExecutor 带有最佳值 - 缓存线程池）。如果没有设置，则默认为之前的行为，它使用缓存的线程池作为消费者端点，而 SimpleAsync 用于回复消费者。建议使用 ThreadPool，以动态增加和减少并发消费者在弹性配置中减少线程回收。		DefaultTaskExecutor Type
camel.component.amqp.delivery-mode	指定要使用的交付模式。Possibles 值是由 javax.jms.DeliveryMode 定义的值。 NON_PERSISTENT = 1 和 PERSISTENT = 2.		整数
camel.component.amqp.delivery-persistent	指定是否默认使用持久性交付。	true	布尔值
camel.component.amqp.destination-resolver	可插拔 org.springframework.jms.support.destination.DestinationResolver，允许您使用您自己的解析器（例如，在 JNDI 注册表中查找实际目的地）。选项是一个 org.springframework.jms.support.destination.DestinationResolver 类型。		字符串
camel.component.amqp.durable-subscription-name	用于指定持久主题订阅的持久化订阅者名称。还必须配置 clientId 选项。		字符串

Name	描述	默认值	类型
camel.component.amqp.eager-loading-of-properties	加载消息时，启用强制加载 JMS 属性，这通常效率低效，因为 JMS 属性可能并不需要，但有时可以提前捕获与底层 JMS 提供程序的任何问题，以及使用 JMS 属性	false	布尔值
camel.component.amqp.enabled	启用 amqp 组件	true	布尔值
camel.component.amqp.error-handler	指定在处理消息时引发任何未捕获的异常时调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置 errorHandler，这些例外将在 WARN 级别记录。您可以配置日志记录级别，并使用 errorHandlerLoggingLevel 和 errorHandlerLogStackTrace 选项记录堆栈 trace。这样可以更轻松地配置，而不是对自定义 errorHandler 进行编码。选项是一个 org.springframework.util.ErrorHandler 类型。		字符串
camel.component.amqp.error-handler-log-stack-trace	允许由默认的 errorHandler 控制是否应记录堆栈追踪。	true	布尔值
camel.component.amqp.error-handler-logging-level	允许为日志记录未捕获的异常配置默认的 errorHandler 日志记录级别。		LoggingLevel
camel.component.amqp.exception-listener	指定正在获得任何底层 JMS 异常通知的 JMS Exception Listener。选项是 javax.jms.ExceptionListener 类型。		字符串
camel.component.amqp.explicit-qos-enabled	设置在发送消息时应使用 deliveryMode、priority 或 timeToLive qualities。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用到当前端点。这与 preserveMessageQos 选项相反，它以消息粒度运行，从 Camel In 消息标头读取 QoS 属性。	false	布尔值
camel.component.amqp.expose-listener-session	指定侦听器会话是否应该在消耗消息时公开。	false	布尔值
camel.component.amqp.force-send-original-message	使用 mapJmsMessage=false Camel 时，如果您涉及路由期间的标头(get 或 set)，则会创建一个新的 JMS 消息来发送到新的 JMS 目的地。将此选项设置为 true，以强制 Camel 发送收到的原始 JMS 消息。	false	布尔值

Name	描述	默认值	类型
camel.component.amqp.format-date-headers-to-iso8601	设置日期标头是否应根据 ISO 8601 标准进行格式化。	false	布尔值
camel.component.amqp.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		字符串
camel.component.amqp.idle-consumer-limit	指定在任意给定时间允许闲置的用户数量的限制。	1	整数
camel.component.amqp.idle-task-execution-limit	指定闲置执行接收任务的限值，没有在其执行中收到任何消息。如果达到这个限制，任务将关闭并保持接收其他执行任务（如果是动态调度，请参阅 maxConcurrentConsumers 设置）。Spring 中提供了额外的文档。	1	整数
camel.component.amqp.include-all-j-m-s-x-properties	从 JMS 映射到 Camel 消息时，是否要包含所有 JMSXxxx 属性。把它设置为 true 将包括 JMSXAppID 和 JMSXUserID 等属性。注： 如果您使用自定义 headerFilterStrategy，则不会应用这个选项。	false	布尔值
camel.component.amqp.include-sent-j-m-s-message-i-d	仅在使用 InOnly （如触发和忘记）发送到 JMS 目的地时才适用。启用此选项将使用实际的 JMSMessageID 增强 Camel Exchange，当消息发送到 JMS 目的地时，由 JMS 客户端使用。	false	布尔值
camel.component.amqp.jms-key-format-strategy	可插拔编码和解码 JMS 密钥策略，以便它们能够与 JMS 规范兼容。Camel 提供了两个开箱即用的实现：default 和 passthrough。默认策略可以安全地放入点和连字符(. 和 -)。passthrough 策略将密钥保留原样。可用于 JMS 代理，其不小心是 JMS 标头键是否包含非法字符。您可以提供自己的 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 # 表示法引用它。选项是 org.apache.camel.component.jms.JmsKeyFormatStrategy 类型。		字符串
camel.component.amqp.jms-operations	允许您使用您自己的 org.springframework.jms.core.JmsOperations 接口实施。Camel 使用 JmsTemplate 作为默认值。可用于测试目的，但不按 spring API 文档中所述使用。选项是一个 org.springframework.jms.core.JmsOperations 类型。		字符串

Name	描述	默认值	类型
camel.component.amqp.lazy-create-transaction-manager	如果为 true，如果选项 transacted=true 时没有注入的 transactionManager，则 Camel 将创建一个 JmsTransactionManager。	true	布尔值
camel.component.amqp.map-jms-message	指定 Camel 是否应该自动将收到的 JMS 消息映射到合适的有效负载类型，如 javax.jms.TextMessage 到 String 等。	true	布尔值
camel.component.amqp.max-concurrent-consumers	指定从 JMS 消耗时的最大并发消费者数（而不是通过 JMS 进行请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。当通过 JMS 进行请求/回复时，选项 replyToMaxConcurrentConsumers 用于控制回复消息监听器上的并发用户数量。		整数
camel.component.amqp.max-messages-per-task	每个任务的消息数量。-1 代表没有限制。如果您将范围用于并发消费者（如 min max），则可以使用此选项将值设置为 eg 100，以控制消费者在需要较少的工作时缩小的速度。	-1	整数
camel.component.amqp.message-converter	要使用自定义 Spring org.springframework.jms.support.converter.MessageConverter，以便您可以控制如何映射到 javax.jms.Message。选项是 org.springframework.jms.support.converter.MessageConverter 类型。		字符串
camel.component.amqp.message-created-strategy	要使用给定的 MessageCreatedStrategy，后者在 Camel 在 Camel 发送 JMS 消息时创建 javax.jms.Message 对象的新实例。选项是 org.apache.camel.component.jms.MessageCreatedStrategy 类型。		字符串
camel.component.amqp.message-id-enabled	发送时，指定是否应添加消息 ID。这只是 JMS 代理的提示。如果 JMS 提供程序接受此提示，这些消息必须将消息 ID 设为 null；如果供应商忽略提示，则必须将消息 ID 设置为其普通唯一值	true	布尔值
camel.component.amqp.message-timestamp-enabled	指定在发送消息时是否默认启用时间戳。这只是 JMS 代理的提示。如果 JMS 提供程序接受此提示，这些消息必须设为零；如果供应商忽略提示，则必须将时间戳设置为正常值	true	布尔值
camel.component.amqp.password	与 ConnectionFactory 搭配使用的密码。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串

Name	描述	默认值	类型
camel.component.amqp.preserve-message-qos	如果要使用消息上指定的 QoS 设置发送消息，而不是 JMS 端点上的 QoS 设置，则设置为 true。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供所有或只提供其中一些。如果没有提供，Camel 将回退到使用端点中的值。因此，在使用此选项时，标头会覆盖来自端点的值。相反，explicitQosEnabled 选项只会使用端点上设置的选项，而不使用消息标头中设置的值。	false	布尔值
camel.component.amqp.priority	大于 1 的值在发送时指定消息优先级（其中 0 是最低优先级，9 是最高）。必须启用 explicitQosEnabled 选项，才能使此选项有任何效果。	4	整数
camel.component.amqp.pub-sub-no-local	指定是否禁止发送由其自身连接发布的消息。	false	布尔值
camel.component.amqp.queue-browse-strategy	在浏览队列时使用自定义 QueueBrowseStrategy。选项是 org.apache.camel.component.jms.QueueBrowseStrategy 类型。		字符串
camel.component.amqp.receive-timeout	接收消息的超时时间（以毫秒为单位）。	1000	Long
camel.component.amqp.recovery-interval	指定恢复尝试之间的间隔，例如当连接被刷新时，以毫秒为单位。默认值为 5000 ms，即 5 秒。	5000	Long
camel.component.amqp.reply-on-timeout-to-max-concurrent-consumers	指定在通过 JMS 使用请求/回复时超时时继续路由的最大并发消费者数。	1	整数
camel.component.amqp.reply-to-cache-level-name	在通过 JMS 进行请求/回复时，按名称设置缓存级别。这个选项只适用于使用固定回复队列（而不是临时）。Camel 默认将使用：CACHE_CONSUMER 用于 exclusive 或 shared w/ replyToSelectorName。和 CACHE_SESSION 用于在没有 replyToSelectorName 的情况下共享。有些 JMS 代理（如 IBM WebSphere）可能需要设置 replyToCacheLevelName=CACHE_NONE 才能工作。注意：如果不使用临时队列，则不允许 CACHE_NONE，且您必须使用更高的值，如 CACHE_CONSUMER 或 CACHE_SESSION。		字符串

Name	描述	默认值	类型
camel.component.amqp.reply-to-concurrent-consumers	指定通过 JMS 进行请求/回复时的默认并发消费者数量。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。	1	整数
camel.component.amqp.reply-to-max-concurrent-consumers	指定通过 JMS 使用请求/回复时的最大并发消费者数。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。		整数
camel.component.amqp.reply-to-type	允许明确指定在通过 JMS 进行请求/回复时，用于 replyTo 队列的策略。可能的值有： Temporary、Shared 或 Exclusive。默认情况下，Camel 将使用临时队列。但是，如果配置了 replyTo，则默认使用 Shared。此选项允许您使用专用队列而不是共享队列。如需了解更多详细信息，请参阅 Camel JMS 文档，特别是有关在集群环境中运行时所影响的备注，而共享回复队列的性能比其 alternatives Temporary 和 Exclusive 性能较低。		ReplyToType
camel.component.amqp.request-timeout	使用 InOut Exchange Pattern 时等待回复的超时（以毫秒为单位）。默认值为 20 秒。您可以包含标头 CamelJmsRequestTimeout 来覆盖此端点配置的超时值，因此每个消息都有单独的超时值。另请参阅 requestTimeoutCheckerInterval 选项。	20000	Long
camel.component.amqp.request-timeout-checker-interval	配置 Camel 在通过 JMS 进行请求/回复时应检查超时的频率。默认情况下，Camel 检查每秒一次。但是，如果发生超时时必须更快地响应，则可以降低这个间隔，以便更频繁地检查。超时由 option requestTimeout 决定。	1000	Long
camel.component.amqp.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.amqp.stream-message-type-enabled	设置是否启用 StreamMessage 类型。文件、InputStream 等消息有效负载（如文件、InputStream 等）将通过 BytesMessage 或 StreamMessage 发送。此选项控制将使用哪种类型。默认情况下，使用 BytesMessage 来强制将整个消息有效负载读取到内存中。通过启用此选项，消息有效负载以块的形式读取内存，然后每个块写入 StreamMessage，直到没有更多数据。	false	布尔值

Name	描述	默认值	类型
camel.component.amqp.subscription-durable	设置是否使订阅持久。要使用的持久订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册持久化订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。仅在侦听主题(pub-sub domain)时具有意义，因此此方法也会切换 pubSubDomain 标志。	false	布尔值
camel.component.amqp.subscription-name	设置要创建的订阅的名称。当具有共享或持久化订阅的主题(pub-sub domain)时应用。订阅名称需要在此客户端的 JMS 客户端 id 中唯一。default 是指定消息监听程序的类名称。注意：除了共享订阅（需要 JMS 2.0）外，每个订阅只允许 1 个并发消费者（此消息监听程序容器的默认值）。		字符串
camel.component.amqp.subscription-shared	设置是否使订阅共享。要使用的共享订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。请注意，共享订阅也可能是持久的，因此此标志也可以与 subscriptionDurable 结合使用。仅在侦听主题(pub-sub domain)时具有意义，因此此方法也会切换 pubSubDomain 标志。需要兼容 JMS 2.0 的消息代理。	false	布尔值
camel.component.amqp.task-executor	允许您指定用于消耗消息的自定义任务 executor。选项是一个 org.springframework.core.task.TaskExecutor 类型。		字符串
camel.component.amqp.test-connection-on-startup	指定是否在启动时测试连接。这样可确保当 Camel 启动所有 JMS 用户与 JMS 代理的有效连接时。如果无法授予连接，则 Camel 会在启动时抛出异常。这样可确保 Camel 不使用失败的连接启动。JMS producers 也经过测试。	false	布尔值
camel.component.amqp.time-to-live	发送消息时，指定消息的生存时间（以毫秒为单位）。	-1	Long
camel.component.amqp.transacted	指定是否使用转换模式	false	布尔值
camel.component.amqp.transaction-manager	要使用的 Spring 事务管理器。选项是一个 org.springframework.transaction.PlatformTransactionManager 类型。		字符串
camel.component.amqp.transaction-name	要使用的事务的名称。		字符串

Name	描述	默认值	类型
camel.component.amqp.transaction-timeout	如果使用转换模式，事务的超时值（以秒为单位）。	-1	整数
camel.component.amqp.transfer-exception	如果启用并且您使用 Request Reply messaging (InOut)和在消费者端的 Exchange 失败，则原因 Exception 将发回为 javax.jms.ObjectMessage 的响应。如果客户端是 Camel，则返回的 Exception 为 rerown。这允许您使用 Camel JMS 作为路由中的网桥，例如，使用持久性队列来启用强大的路由。请注意，如果您也启用了 transferExchange，这个选项会优先使用。需要被捕获的例外是串行的。在返回到生成者时，消费者侧的原始 Exception 可以嵌套在外部异常中，如 org.apache.camel.RuntimeCamelException。	false	布尔值
camel.component.amqp.transfer-exchange	您可以通过线线传输交换，而不只是正文和标头。以下字段会被传输： 在 body, Out body, Fault body, In headers, Out headers, Fault 标头, Exchange properties, Exchange exception。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。您必须在生成者和消费者端启用这个选项，因此 Camel 知道有效负载是交换而不是常规有效负载。	false	布尔值
camel.component.amqp.transfer-fault	如果启用且您使用 Request Reply messaging (InOut)，且交换失败，在消费者端使用 SOAP 错误（非例外），则 Message SerialisFault （）上的错误标志将重新作为带有键 org.apache.camel.component.jms.JmsConstants #JMS_TRANSFER_FAULT kexecJMS_TRANSFER_FAULT 的 JMS 标头发回，作为带有密钥 org.apache.camel.component.jms.JmsConstants #JMS_TRANSFER_FAULT 的响应中。如果客户端是 Camel，则返回的 fault 标志将在 org.apache.camel.MessageAllsetFault (boolean)上设置。在使用支持错误（如 SOAP）（如 cxf 或 spring-ws）的 Camel 组件时，您可能需要启用此功能。	false	布尔值
camel.component.amqp.use-message-i-d-as-correlation-i-d	指定 JMSMessageID 是否应该始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值
camel.component.amqp.username	与 ConnectionFactory 搭配使用的用户名。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串

Name	描述	默认值	类型
camel.component.amqp.wait-for-provision-correlation-to-be-updated-counter	在通过 JMS 进行请求/回复时，以及启用选项 useMessageIDAsCorrelationID 时，等待 provisional correlation id 的次数。	50	整数
camel.component.amqp.wait-for-provision-correlation-to-be-updated-thread-sleeping-time	millis 中的间隔，在每次等待配置关联 ID 时处于睡眠状态。	100	Long

5.4. 使用方法

由于 AMQP 组件从 JMS 组件继承，因此前者的使用几乎与后者一致：

使用 AMQP 组件

```
// Consuming from AMQP queue
from("amqp:queue:incoming").
  to(...);

// Sending message to the AMQP topic
from(...).
  to("amqp:topic:notify");
```

5.5. 配置 AMQP 组件

从 Camel 2.16.1 开始，您还可以使用 **AMQPComponent"amqp10Component (String connectionURI)** factory 方法返回带有预先配置的主题前缀的 AMQP 1.0 组件：

创建 AMQP 1.0 组件

```
 AMQPComponent amqp =
AMQPComponent.amqp10Component("amqp://guest:guest@localhost:5672");
```

请记住，从 Camel 2.17 开始，**AMQPComponent"amqp10Component (String connectionURI)** factory 方法已被代表 **AMQPComponent"amqpComponent (String connectionURI)**:

创建 AMQP 1.0 组件

```
AMQPComponent amqp = AMQPComponent.amqpComponent("amqp://localhost:5672");

AMQPComponent authorizedAmqp = AMQPComponent.amqpComponent("amqp://localhost:5672",
"user", "password");
```

从 Camel 2.17 开始，为了自动配置 AMQP 组件，您还可以将 **org.apache.camel.component.amqp.AMQPConnectionDetails** 实例添加到 registry。例如，对于 Spring Boot，您只需要定义 bean：

AMQP 连接详情自动配置

```
@Bean
AMQPConnectionDetails amqpConnection() {
  return new AMQPConnectionDetails("amqp://localhost:5672");
}

@Bean
AMQPConnectionDetails securedAmqpConnection() {
  return new AMQPConnectionDetails("amqp://localhost:5672", "username", "password");
}
```

同样，在使用 Camel-CDI 时也可以使用 CDI producer 方法

CDI 的 AMQP 连接详情自动配置

```
@Produces
AMQPConnectionDetails amqpConnection() {
  return new AMQPConnectionDetails("amqp://localhost:5672");
}
```

您还可以依赖 Camel 属性 来读取 AMQP 连接详情。factory 方法 **AMQPConnectionDetails.discoverAMQP （）** 尝试读取类似 Kubernetes 的约定中的 Camel 属性，就像以下代码片段所示：

AMQP 连接详情自动配置

```
export AMQP_SERVICE_HOST = "mybroker.com"
export AMQP_SERVICE_PORT = "6666"
export AMQP_SERVICE_USERNAME = "username"
export AMQP_SERVICE_PASSWORD = "password"

...

@Bean
AMQPConnectionDetails amqpConnection() {
  return AMQPConnectionDetails.discoverAMQP();
}
```

启用 AMQP 特定选项

例如，如果需要启用 **amqp.traceFrames**，您可以通过将选项附加到 URI 来完成此操作，如下例所示：

```
AMQPComponent amqp = AMQPComponent.amqpComponent("amqp://localhost:5672?
amqp.traceFrames=true");
```

要进行参考，请查看 QPID JMS 客户端配置

5.6. 使用主题

要使用与 **camel-amqp** 一起工作的主题，您需要将组件配置为使用 **topic://** 作为主题前缀，如下所示：

```
<bean id="amqp" class="org.apache.camel.component.amqp.AmqpComponent">
  <property name="connectionFactory">
    <bean class="org.apache.qpid.jms.JmsConnectionFactory" factory-method="createFromURL">
      <property name="remoteURI" value="amqp://localhost:5672" />
      <property name="topicPrefix" value="topic://" />  <!-- only necessary when connecting to
ActiveMQ over AMQP 1.0 -->
    </bean>
  </property>
 </bean>
```

请记住，**AMQPComponent#amqpComponent （）** 方法和 **AMQPConnectionDetails** 均使用主题前缀预配置组件，因此您不必显式配置它。

5.7. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 6 章 APNS 组件

从 Camel 版本 2.8 开始提供

apns 组件用于向 iOS 设备发送通知。apns 组件使用 javapns 库。
组件支持向 Apple Push Notification Servers (APNS)发送通知，并从服务器消耗反馈。

默认情况下，消费者被配置为 3600 秒进行轮询，因为它最好只消耗来自 Apple Push Notification Servers 的反馈流。例如：每 1 小时避免大量服务器。

反馈流提供有关不活跃设备的信息。如果您的移动应用程序不是大量使用，您只需要每小时获取此信息。

Maven 用户需要将以下依赖项添加到这个组件的 **pom.xml** 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-apns</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

6.1. URI 格式

发送通知：

```
apns:notify[?options]
```

使用反馈：

```
apns:consumer[?options]
```

6.2. 选项

APNS 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
apnsService (common)	**必需** 要使用的 ApnsService。org.apache.camel.component.apns.factory.ApnsServiceFactory 可用于构建 ApnsService		ApnsService
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

APNS 端点使用 URI 语法进行配置：

```
apns:name
```

使用以下路径和查询参数：

6.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	端点的名称		字符串

6.2.2. 查询参数(20 参数)：

Name	描述	默认值	类型
令牌 (common)	如果要静态声明与您要通知的设备相关的令牌，请配置此属性。令牌用逗号分开。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int

Name	描述	默认值	类型
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

6.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.apns.apns-service	要使用的 ApnsService。org.apache.camel.component.apns.factory.ApnsServiceFactory 可用于构建 ApnsService。选项是一个 com.notnoop.apns.ApnsService 类型。		字符串
camel.component.apns.enabled	启用 apns 组件	true	布尔值
camel.component.apns.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

您可以在 URI 中附加查询选项，格式为 **?option=value&option=value&...**

6.3.1. 组件

ApnsComponent 必须使用 **com.notnoop.apns.ApnsService** 配置。可以使用 **org.apache.camel.component.apns.factory.ApnsServiceFactory** 来创建和配置该服务。有关示例，请参阅以下。在测试源代码中，以及测试源代码中。

6.3.1.1. SSL 设置

要使用安全连接，**org.apache.camel.util.jsse.SSLContextParameters** 实例应该注入到 **org.apache.camel.component.apns.factory.ApnsServiceFactory**，用于配置组件。有关示例，请参阅测试资源。SSL 示例

6.4. 交换数据格式

当 Camel 将获取与不活跃设备对应的反馈数据时，它将检索一个 InactiveDevice 对象列表。检索列表的每个 InactiveDevice 对象将设置为 In 正文，然后由消费者端点处理。

6.5. 消息标头

Camel Apns 使用这些标头。

属性	默认	描述
CamelApnsTokens		默认为空。

属性	默认	描述
CamelApnsMessageType	**STRING, PAYLOAD, APNS_NOTIFICATION**	如果您选择 PAYLOAD 作为消息类型，则消息将被视为 APNS 有效负载，并按原样发送。如果您选择 STRING，消息将转换为 APNS 有效负载。从 **Camel 2.16** onwards APNS_NOTIFICATION 用于发送消息正文，如 com.notnoop.apns.ApnsNotification 类型。

6.6. APNSSERVICEFACTORY 构建器回调

ApnsServiceFactory 附带空的回调方法，可用于配置（甚至替换）默认的 **ApnsServiceBuilder** 实例。该方法的签名可能如下所示：

```
protected ApnsServiceBuilder configureServiceBuilder(ApnsServiceBuilder serviceBuilder);
```

可以像以下所示使用：

```
ApnsServiceFactory proxiedApnsServiceFactory = new ApnsServiceFactory(){

  @Override
  protected ApnsServiceBuilder configureServiceBuilder(ApnsServiceBuilder serviceBuilder) {
    return serviceBuilder.withSocksProxy("my.proxy.com", 6666);
  }

};
```

6.7. SAMPLES

6.7.1. Camel Xml 路由

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:camel="http://camel.apache.org/schema/spring"
       xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
        http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-
spring.xsd">

    <!-- Replace by desired values -->
    <bean id="apnsServiceFactory"
class="org.apache.camel.component.apns.factory.ApnsServiceFactory">

        <!-- Optional configuration of feedback host and port -->
        <!-- <property name="feedbackHost" value="localhost" /> -->
        <!-- <property name="feedbackPort" value="7843" /> -->
```

```
        <!-- Optional configuration of gateway host and port -->
        <!-- <property name="gatewayHost" value="localhost" /> -->
        <!-- <property name="gatewayPort" value="7654" /> -->

        <!-- Declaration of certificate used -->
            <!-- from Camel 2.11 onwards you can use prefix: classpath:, file: to refer to load the
certificate from classpath or file. Default it classpath -->
        <property name="certificatePath" value="certificate.p12" />
        <property name="certificatePassword" value="MyCertPassword" />

        <!-- Optional connection strategy - By Default: No need to configure -->
        <!-- Possible options: NON_BLOCKING, QUEUE, POOL or Nothing -->
        <!-- <property name="connectionStrategy" value="POOL" /> -->
        <!-- Optional pool size -->
        <!-- <property name="poolSize" value="15" /> -->

        <!-- Optional connection strategy - By Default: No need to configure -->
        <!-- Possible options: EVERY_HALF_HOUR, EVERY_NOTIFICATION or Nothing (Corresponds
to NEVER javapns option) -->
        <!-- <property name="reconnectionPolicy" value="EVERY_HALF_HOUR" /> -->
    </bean>

    <bean id="apnsService" factory-bean="apnsServiceFactory" factory-method="getApnsService" />

    <!-- Replace this declaration by wanted configuration -->
    <bean id="apns" class="org.apache.camel.component.apns.ApnsComponent">
        <property name="apnsService" ref="apnsService" />
    </bean>

    <camelContext id="camel-apns-test" xmlns="http://camel.apache.org/schema/spring">
            <route id="apns-test">
                <from uri="apns:consumer?initialDelay=10&amp;delay=3600&amp;timeUnit=SECONDS"
/>
                <to uri="log:org.apache.camel.component.apns?showAll=true&amp;multiline=true" />
                <to uri="mock:result" />
            </route>
    </camelContext>

</beans>
```

6.7.2. Camel Java 路由

创建 camel 上下文，并以编程方式声明 apns 组件

```
    protected CamelContext createCamelContext() throws Exception {
        CamelContext camelContext = super.createCamelContext();

        ApnsServiceFactory apnsServiceFactory = new ApnsServiceFactory();
        apnsServiceFactory.setCertificatePath("classpath:/certificate.p12");
        apnsServiceFactory.setCertificatePassword("MyCertPassword");

        ApnsService apnsService = apnsServiceFactory.getApnsService(camelContext);

        ApnsComponent apnsComponent = new ApnsComponent(apnsService);
        camelContext.addComponent("apns", apnsComponent);
```

```
        return camelContext;
    }
```

[[APNS-ApnsProducer-iOStargetdevicedynamicallyconfiguredviaheader:"CamelApnsTokens"]]
ApnsProducer - iOS 目标设备通过 header: **"CamelApnsTokens"**动态配置

```
    protected RouteBuilder createRouteBuilder() throws Exception {
        return new RouteBuilder() {
            public void configure() throws Exception {
                from("direct:test")
                    .setHeader(ApnsConstants.HEADER_TOKENS, constant(IOS_DEVICE_TOKEN))
                    .to("apns:notify");
                }
        }
    }
```

ApnsProducer - iOS 目标设备通过 uri 静态配置

```
    protected RouteBuilder createRouteBuilder() throws Exception {
        return new RouteBuilder() {
            public void configure() throws Exception {
                from("direct:test").
                to("apns:notify?tokens=" + IOS_DEVICE_TOKEN);
            }
        };
    }
```

ApnsConsumer

```
from("apns:consumer?initialDelay=10&delay=3600&timeUnit=SECONDS")
    .to("log:com.apache.camel.component.apns?showAll=true&multiline=true")
    .to("mock:result");
```

6.8. 另请参阅

- 组件
- 端点 * 关于使用 APNS 的博客（在 french 中）

第 7 章 ASN.1 文件数据格式

从 Camel 版本 2.20 开始提供

ASN.1 数据格式 [Introduction to ASN.1](https://www.itu.int/en/ITU-T/asn1/Pages/introduction.aspx)是一个 Camel Framework 的数据格式实施，它基于 Bouncy Castle 的 bcprov-jdk18on 库和 jASN.1 的 java 编译器用于描述由电信协议传输的数据，无论这些数据的语言实施和物理表示，无论是复杂的应用程序，还是非常简单。消息可以是 unmarshalled （对 Java POJO 的简单 Java POJO）到普通 Java 对象。通过 Camel 的路由引擎和数据转换的帮助，您可以使用 POJO 进行播放，并应用自定义格式并调用其他 Camel 组件来转换和向上游系统发送信息。

7.1. ASN.1 数据格式选项

ASN.1 文件 dataformat 支持 3 个选项，如下所列。

Name	默认值	Java 类型	描述
usingIterator	**false**	**布尔值**	如果 asn1 文件有多个条目，则将此选项设置为 true，允许使用 splitter EIP （在流传输模式中使用迭代器来分割数据）。
clazzName		**字符串**	unmarshalling 时使用的类名称
contentTypeHeader	**false**	**布尔值**	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

7.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.asn1.clazz-name	unmarshalling 时使用的类名称		字符串
camel.dataformat.asn1.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.asn1.enabled	是否启用 asn1 数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.asn1.using-iterator	如果 asn1 文件有多个条目，则将此选项设置为 true，允许使用 splitter EIP （在流传输模式中使用迭代器来分割数据）。	false	布尔值

ND

7.3. UNMARSHAL

unmarshal ASN.1 结构化消息的三种不同方式：（通常为二进制文件）

在本示例中，我们 unmarshal BER 文件有效负载到 OutputStream 并将其发送到 mock 端点。

```
from("direct:unmarshal").unmarshal(asn1).to("mock:unmarshal");
```

在第二个示例中，我们使用 Split EIP 将 unmarshal BER 文件有效负载字节阵列。应用 Split EIP 的原因是，每个 BER 文件或(ASN.1 结构化文件)通常包含多个记录来处理和 Split EIP 有助于我们获得文件中的每个记录作为字节阵列，实际是 ASN1Primitive 的实例（通过对 Bouncy Castle 的 ASN.1 支持）。bcprov-jdk18on 库）由 中的公共静态方法(ASN1Primitive.fromByteArray)在本例中转换为 ASN1Primitive.fromByteArray）请注意，**您需要使用Iterator**=true 设置

```
from("direct:unmarshal").unmarshal(asn1).split(body(Iterator.class)).streaming().to("mock:unmarshal"
);
```

在最后一个示例中，我们使用 Split EIP 将 unmarshal BER 文件有效负载为纯旧的 Java 对象。上例中已经提到了应用 Split EIP 的原因。请注意，请注意原因。在本例中，我们还需要设置类的完全限定名称，或 <YourObject>.class 通过数据格式引用。这里需要注意的重要事项是，您的对象应由 jasn1 编译器生成，它是一个可生成 ASN.1 结构的 java 对象表示的工具。有关使用 jasn1 编译器的参考信息，请参阅 [JASN.1 Project Page](https://www.openmuc.org/asn1/)，并查看如何使用 maven 的 exec 插件调用编译器。例如，在这个数据格式的单元中测试了一个示例 ASN.1 结构(TestSMSBerCdr.asn1)，在 **src/test/resources/asn1_structure**. jasn1 编译器被调用，并在 **${basedir}/target/generated/src/test/java** The nice things 中生成 java 对象的表示。

```
from("direct:unmarshaldsl")
        .unmarshal()
        .asn1("org.apache.camel.dataformat.asn1.model.testsmscbercdr.SmsCdr")
        .split(body(Iterator.class)).streaming()
.to("mock:unmarshaldsl");
```

7.4. 依赖项

要在 camel 路由中使用 ASN.1 数据格式，您需要添加针对实施此数据格式的 **camel-asn1** 的依赖。

如果使用 Maven，您只需在 **pom.xml** 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-asn1</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

第 8 章 AS2 组件

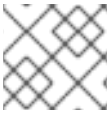
重要

Karaf 的 **camel-as2** 组件只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。

这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。有关红帽技术预览功能支持范围的更多信息，请参阅 https://access.redhat.com/support/offerings/techpreview。

从 Camel 版本 2.22 开始提供

AS2 组件使用 RFC4130 中指定的 HTTP 传输协议提供 EDI 消息传输。



注意

此组件目前正在进行中。预期此组件的将来版本更改的 URI 选项和路径和查询参数。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-as2</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

8.1. URI 格式

```
as2://apiName/methodName
```

apiName 可以是以下之一：

- client
- server

8.2. AS2 选项

AS2 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	使用共享配置		AS2Configuration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AS2 端点使用 URI 语法进行配置：

```
as2:apiName
```

使用以下路径和查询参数：

8.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
apiName	**需要** 执行什么操作		AS2ApiName

8.2.2. 查询参数(30 参数)：

Name	描述	默认值	类型
as2From (common)	AS2From 标头的值。		字符串
as2MessageStructure (common)	AS2 消息的结构.其中之一： PLAIN - No encryption, no signature, SIGNED - No encryption, signature, ENCRYPTED - Encryption, no signature, ENCRYPTED_SIGNED - 加密, 签名		AS2MessageStructure
as2To (common)	AS2 Message 的 AS2To 标头的值。		字符串
as2Version (common)	AS2 协议的版本。	1.1	字符串
clientFqdn (common)	客户端完全限定域名(FQDN)。用于端点发送的消息 ID。	camel.apache.org	字符串
dispositionNotificationTo (common)	Disposition-Notification-To 标头的值。为此参数分配一个值，请求 AS2 消息的消息分布通知(MDN)。		字符串
ediMessageTransferEncoding (common)	EDI 消息传输编码。		字符串
ediMessageType (common)	内容类型 EDI 消息。应用程序/事实之一, application/edi-x12, application/edi-consent		ContentType
encryptAlgorithm (common)	用于加密 EDI 消息的算法。		AS2EncryptionAlgorithm
encryptCertificateChain (common)	用于加密 EDI 消息的证书链。		Certificate[]

Name	描述	默认值	类型
encryptPrivateKey (common)	用于加密 EDI 消息的密钥。		PrivateKey
from (common)	AS2 消息的 From 标头的值。		字符串
inBody (common)	设置要在交换 In Body 中传递的参数名称		字符串
methodName (common)	**必需的** 所选操作使用哪些子操作		字符串
requestUri (common)	EDI 消息的请求 URI。	/	字符串
server (common)	Server message 标头中包含的值标识 AS2 服务器。	Camel AS2 服务器端点	字符串
serverFqdn (common)	服务器完全限定域名(FQDN)。用于端点发送的消息 ID。	camel.apache.org	字符串
serverPortNumber (common)	服务器的端口号。		整数
signedReceiptMicAlgorithms (common)	要求以首选方式生成消息完整性检查(MIC)列表，在消息发送通知(MDN)中返回的消息完整性检查(MIC)		string[]
signingAlgorithm (common)	用于签署 EDI 消息的算法。		AS2SignatureAlgorithm
signingCertificateChain (common)	用于签署 EDI 消息的证书链。		Certificate[]
signingPrivateKey (common)	用于签署 EDI 消息的密钥。		PrivateKey
subject (common)	AS2 消息的 Subject 标头值。		字符串
targetHostname (common)	目标主机的主机名(IP 或 DNS 名称)。		字符串
targetPortNumber (common)	目标主机的端口号。-1 表示方案的默认端口。		整数

Name	描述	默认值	类型
userAgent (common)	User-Agent 消息标头中包含的值标识 AS2 用户代理。	Camel AS2 客户端端点	字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

8.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 28 个选项，如下所列。

Name	描述	默认值	类型
camel.component.as2.configuration.api-name	要执行的操作类型		AS2ApiName
camel.component.as2.configuration.as2-from	AS2From 标头的值。		字符串
camel.component.as2.configuration.as2-message-structure	AS2 消息的结构.其中之一： PLAIN - No encryption, no signature, SIGNED - No encryption, signature, ENCRYPTED - Encryption, no signature, ENCRYPTED_SIGNED - 加密, 签名		AS2MessageStructure
camel.component.as2.configuration.as2-to	AS2 Message 的 AS2To 标头的值。		字符串

Name	描述	默认值	类型
camel.component.as2.configuration.as2-version	AS2 协议的版本。	1.1	字符串
camel.component.as2.configuration.client-fqdn	客户端完全限定域名(FQDN)。用于端点发送的消息 ID。	camel.apache.org	字符串
camel.component.as2.configuration.disposition-notification-to	Disposition-Notification-To 标头的值。为此参数分配一个值，请求 AS2 消息的消息分布通知(MDN)。		字符串
camel.component.as2.configuration.edi-message-transfer-encoding	EDI 消息传输编码。		字符串
camel.component.as2.configuration.edi-message-type	内容类型 EDI 消息。应用程序/事实之一, application/edi-x12, application/edi-consent		ContentType
camel.component.as2.configuration.encrypting-algorithm	用于加密 EDI 消息的算法。		AS2EncryptionAlgorithm
camel.component.as2.configuration.encrypting-certificate-chain	用于加密 EDI 消息的证书链。		Certificate[]
camel.component.as2.configuration.encrypting-private-key	用于加密 EDI 消息的密钥。		PrivateKey
camel.component.as2.configuration.from	AS2 消息的 From 标头的值。		字符串
camel.component.as2.configuration.method-name	用于所选操作的子操作		字符串

Name	描述	默认值	类型
camel.component.as2.configuration.request-uri	EDI 消息的请求 URI。	/	字符串
camel.component.as2.configuration.server	Server message 标头中包含的值标识 AS2 服务器。	Camel AS2 服务器端点	字符串
camel.component.as2.configuration.server-fqdn	服务器完全限定域名(FQDN)。用于端点发送的消息 ID。	camel.apache.org	字符串
camel.component.as2.configuration.server-port-number	服务器的端口号。		整数
camel.component.as2.configuration.signed-receipt-mic-algorithms	要求以首选方式生成消息完整性检查(MIC)列表，在消息发送通知(MDN)中返回的消息完整性检查(MIC)		string[]
camel.component.as2.configuration.signing-algorithm	用于签署 EDI 消息的算法。		AS2SignatureAlgorithm
camel.component.as2.configuration.signing-certificate-chain	用于签署 EDI 消息的证书链。		Certificate[]
camel.component.as2.configuration.signing-private-key	用于签署 EDI 消息的密钥。		PrivateKey
camel.component.as2.configuration.subject	AS2 消息的 Subject 标头值。		字符串
camel.component.as2.configuration.target-hostname	目标主机的主机名(IP 或 DNS 名称)。		字符串

Name	描述	默认值	类型
camel.component.as2.configuration.target-port-number	目标主机的端口号。-1 表示方案的默认端口。		整数
camel.component.as2.configuration.user-agent	User-Agent 消息标头中包含的值标识 AS2 用户代理。	Camel AS2 客户端端点	字符串
camel.component.as2.enabled	是否启用 as2 组件的自动配置。这默认是启用的。		布尔值
camel.component.as2.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

8.4. 客户端端点：

客户端端点使用端点前缀 **客户端**，后跟方法名称和相关选项的名称。端点 URI MUST 包含前缀 **客户端**。

```
as2://client/method?[options]
```

不是强制的端点选项由 **[]** 表示。当端点没有强制选项时，必须提供其中一组 **[]** 选项。生成者端点也可以使用特殊选项 **inBody**，它应包含端点选项的名称，其值将包含在 Camel Exchange In 消息中。

任何端点选项都可以在端点 URI 中提供，或者在消息标头中动态提供。消息标头名称的格式必须是 **CamelAS2.<option>**。请注意，**inBody** 选项会覆盖消息标头，即 **Body=option 中的端点** 选项会覆盖 **CamelAS2.option** 标头。

如果没有为端点 URI 或消息标头中为 option **defaultRequest** 提供值，则会假定为 **null**。请注意，只有其他选项无法满足匹配端点时，才会使用 **null** 值。

如果是 AS2 API 错误，端点将抛出一个 RuntimeCamelException，并带有 **org.apache.http.HttpException** 派生异常原因。

方法	选项	结果正文类型

方法	选项	结果正文类型
send	ediMessage, requestUri, subject, from, as2From, as2To, as2MessageStructure, ediMessageContentType, ediMessageTransferEncoding, dispositionNotificationTo, signedReceiptMicAlgorithms	org.apache.http.protocol.HttpCoreContext

*客户端的*URI 选项

Name	类型
ediMessage	字符串
requestUri	字符串
subject	字符串
from	字符串
as2From	字符串
as2To	字符串

Name	类型
as2MessageStructure	org.apache.camel.component.as2.api.AS2MessageStructure
ediMessageContentType	字符串
ediMessageTransferEncoding	字符串
dispositionNotificationTo	字符串
signedReceiptMicAlgorithms	string[]

8.5. 服务器端点：

服务器端点使用端点前缀 **服务器**，后跟一个方法的名称和相关选项。端点 URI MUST 包含前缀 **服务器**。

```
as2://server/method?[options]
```

不是强制的端点选项由 **[]** 表示。当端点没有强制选项时，必须提供其中一组 **[]** 选项。生成者端点也可以使用特殊选项 **inBody**，它应包含端点选项的名称，其值将包含在 Camel Exchange In 消息中。

任何端点选项都可以在端点 URI 中提供，或者在消息标头中动态提供。消息标头名称的格式必须是 **CamelAS2.<option>**。请注意，**inBody** 选项会覆盖消息标头，即 **Body=option 中的端点** 选项会覆盖 **CamelAS2.option** 标头。

如果没有为端点 URI 或消息标头中为 option **defaultRequest** 提供值，则会假定为 **null**。请注意，只有其他选项无法满足匹配端点时，才会使用 **null** 值。

如果是 AS2 API 错误，端点将抛出一个 RuntimeCamelException，并带有 **org.apache.http.HttpException** 派生异常原因。

方法	选项	结果正文类型

方法	选项	结果正文类型
listen	requestUriPattern	org.apache.http.protocol.HttpCoreContext

*服务器的*URI 选项

Name	类型
requestUriPattern	字符串

第 9 章 星号组件

从 Camel 版本 2.18 开始提供

通过 **星号：** 组件，您可以使用 星号-java轻松处理 Asterisk PBX Server http://www.asterisk.org/

此组件有助于与 Asterisk Manager接口接口

Maven 用户需要将以下依赖项添加到这个组件的 **pom.xml** 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-asterisk</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

9.1. URI 格式

```
asterisk:name[?options]
```

9.2. 选项

Asterisk 组件没有选项。

Asterisk 端点使用 URI 语法进行配置：

```
asterisk:name
```

使用以下路径和查询参数：

9.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	**所需的** 逻辑名称		字符串

9.2.2. 查询参数(8 参数)：

Name	描述	默认值	类型
hostname (common)	**必需** 星号服务器的主机名		字符串
password (common)	**所需的** 登录密码		字符串

Name	描述	默认值	类型
username (common)	**所需的** 登录用户名		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
action (producer)	执行什么操作，如获取队列状态、sip peers 或 extension 状态。		AsteriskAction
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

9.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.asterisk.enabled	启用星号组件	true	布尔值
camel.component.asterisk.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

9.4. 操作

支持的操作有：

- QUEUE_STATUS, Queue Status
- SIP_PEERS，列出 SIP Peers
- EXTENSION_STATE, Check Extension Status

第 10 章 ATMOS 组件

从 Camel 版本 2.15 开始提供

Camel-Atmos 是一个 Apache Camel 组件，允许您使用 Atmos Client 使用 ViPR 对象数据服务。

```
from("atmos:foo/get?remotePath=/path").to("mock:test");
```

10.1. 选项

Atmos 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
fullTokenId (security)	传递给 Atmos 客户端的令牌 ID		字符串
secretKey (security)	传递给 Atmos 客户端的 secret 密钥		字符串
URI (advanced)	要连接的 Atmos 客户端的服务器的 URI		字符串
sslValidation (security)	Atmos 客户端是否应该执行 SSL 验证	false	布尔值
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Atmos 端点使用 URI 语法进行配置：

```
atmos:name/operation
```

使用以下路径和查询参数：

10.1.1. 路径参数(2 参数)：

Name	描述	默认值	类型
name	Atmos 名称		字符串
operation	**要执行** 所需的操作		AtmosOperation

10.1.2. 查询参数(12 参数)：

Name	描述	默认值	类型
enableSslValidation (common)	Atmos SSL 验证	false	布尔值
fullTokenId (common)	Atmos client fullTokenId		字符串
localPath (common)	放置文件的本地路径		字符串
newRemotePath (common)	在移动文件时在 Atmos 上新路径		字符串
query (common)	在 Atmos 上搜索查询		字符串
remotePath (common)	在 Atmos 上放置文件的位置		字符串
secretKey (common)	Atmos 共享 secret		字符串
URI (common)	Atomos server uri		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

10.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
camel.component.atmos.enabled	启用tmos 组件	true	布尔值
camel.component.atmos.full-token-id	传递给 Atmos 客户端的令牌 ID		字符串
camel.component.atmos.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.atmos.secret-key	传递给 Atmos 客户端的 secret 密钥		字符串
camel.component.atmos.ssl-validation	Atmos 客户端是否应该执行 SSL 验证	false	布尔值
camel.component.atmos.uri	要连接的 Atmos 客户端的服务器的 URI		字符串

10.3. 依赖项

要在 camel 路由中使用 Atmos，您需要添加对实现此数据格式的 **camel-atmos** 的依赖关系。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-atmos</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

10.4. 集成

当您查看tmos 集成时，有一种类型的 consumer GetConsumer，这是 ScheduledPollConsumer 的类型。

- **Get**

而有 4 种类型的生成者，它们有 4 个

- **Get**
- **del**
- **Move**

- **put**

10.5. 例子

这些示例取自测试：

```
from("atmos:foo/get?remotePath=/path").to("mock:test");
```

这里是一个消费者示例。**remotePath** 代表数据读取并将 camel 交换传递给 regarding producer Underneath 的路径，这个组件会为这个和所有其他操作使用tmos 客户端 API。

```
from("direct:start")
.to("atmos://get?remotePath=/dummy/dummy.txt")
.to("mock:result");
```

在这里，这是生成者示例。**remotePath** 代表 ViPR 对象数据服务上操作的路径。在生产者中，操作 (**Get**,**Del**,**Move**,**Put**)在 ViPR 对象数据服务上运行，结果在 camel 交换标头上设置。

关于操作，以下标头在 camel exhange 中设置

```
DOWNLOADED_FILE, DOWNLOADED_FILES, UPLOADED_FILE, UPLOADED_FILES,
FOUND_FILES, DELETED_PATH, MOVED_PATH;
```

10.6. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 11 章 ATLASMAP 组件

注意

仅支持生成者。

您可以使用 AtlasMap 组件通过 AtlasMap 数据映射定义来处理数据映射。从 AtlasMap Data Mapper UI 导出 AtlasMap 映射时，它将被打包为 ADM 归档文件。

注意： 虽然可以加载未打包到 ADM 归档文件中的映射定义 JSON 文件，但某些功能将无法正常工作。我们建议您将 ADM 归档文件用于生产目的。

要将组件与 Maven 搭配使用，请将以下依赖项添加到 **pom.xml** 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-atlasmap</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

另外，您还可以包含 Apache Daffodil 模块 DFDL 模块：

```
<dependency>
  <groupId>io.atlasmap</groupId>
  <artifactId>atlas-dfdl-module</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as atlasmap-core in camel-atlasmap -->
</dependency>
```

11.1. URI 格式

```
atlas:mappingName[?options]
```

mappingName 是要处理的 AtlasMap 映射定义的 classpath-local URI，可以是 ADM 归档文件（最好）或映射定义 JSON 文件。

11.2. 配置选项

Camel 组件在两个独立级别上配置：

- 组件级别
- 端点级别

11.2.1. 配置组件级别选项

组件级别是最高配置级别。它包含所有端点的通用配置。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或者直接使用 Java 代码配置组件。

某些组件只有几个选项，其他组件可能会有许多选项。组件可能具有安全设置、用于身份验证的凭证、网络连接的 URL 等。

组件通常已经为最常见的情况预配置默认值，因此您可能不需要配置任何选项，或者仅配置几个。

11.2.2. 组件选项

AtlasMap 组件支持 4 个选项：

Name	描述	注释	default	类型
lazyStartProducer (producer)	生成者的 lazy 启动。 生产者从第一个消息开始。	在生成者无法启动并导致路由失败时，允许 CamelContext 和路由启动。 启用 lazy start 时，您可以通过 Camel 的路由错误处理程序在路由消息期间处理故障。 当处理第一个消息时，创建并启动制作者可能会延长总处理时间。	false	布尔值
atlasContextFactory (advanced)	要使用 AtlasContextFactory，否则会创建新的引擎。		AtlasContextFactory	**autowiredEnabled** (advanced)
是否启用自动关闭。这用于自动关闭选项（选项必须标记为 autowired），方法是在 registry 中查找查找是否有单个匹配类型实例，然后在组件上配置。这可用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值	**propertiesFile** (advanced)	用于 AtlasContextFactory 初始化的属性文件的 URI。

11.2.3. 配置端点级别选项

在端点级别上，包含端点本身的配置。

您可以将端点 URI 直接配置为路径和查询参数。您还可以使用 Endpoint DSL 和 Data Format DSL 作为 Java 中配置端点的 *安全方法*。

端点通常有许多选项来配置您需要的端点。

端点选项按照其用途进行分类，可以是作为消费者(**从**)或生成者(**到**)或两者的分类。

在配置选项时，最好为 urls、端口号和敏感信息使用 Property Placeholders 而不是硬编码设置。

使用占位符从代码外部化配置，使其更灵活且可重复使用。

11.2.4. 端点选项

Apache Camel 组件参考端点使用 URI 语法进行配置，其路径和查询参数：

```
atlas:resourceUri
```

11.2.4.1. 路径参数(1 参数)

Name	描述	default	类型
resourceUri (producer)	资源 **所需的** 路径。您可以使用前缀前缀：**classpath**、**file**、**http**、**ref** 或 **bean**。 前缀 **类路径**、**文件** 和 **http** 使用这些协议加载资源。**ref** 前缀在 registry 中查找资源。前缀 **bean** 调用 bean 方法，按名称用作资源，在 dot: **bean:myBean.myMethod** 后提供。	**classpath**	字符串

11.2.4.2. 查询参数(7 参数)

Name	描述	注释	default	类型
allowContextMapAll (producer)	允许访问所有上下文映射详细信息。 默认情况下，只允许访问消息正文和标头。	启用后，**allowContextMapAll** 允许完全访问当前 Exchange 和 CamelContext，**这会带来潜在的安全风险**，因为这会打开 CamelContext API 的全部功能。	false	布尔值
contentCache (producer)	使用资源内容缓存。		false	布尔值

forceReload (producer)	使用强制重新加载模式。 这会从每个交换上的文件加载 ADM。	默认情况下，ADM 文件仅从第一个 Exchange 上的文件加载，而 AtlasContext 将会被重复使用，直到端点被重新创建为止。	false	布尔值
lazyStartProducer (producer) (advanced)	生成者的 lazy 启动。 生产者从第一个消息开始。	在生成者无法启动并导致路由失败时，允许 CamelContext 和路由启动。 启用 lazy start 时，您可以通过 Camel 的路由错误处理程序在路由消息期间处理故障。 处理第一个消息时，创建和启动制作者可能会延长总处理时间。	false	布尔值
sourceMapName (producer)	源消息映射的 Exchange 属性名称，其中包含 **java.util.Map<String, Message>**，其中键是 AtlasMap Document ID。	AtlasMap 将 Message bodies 用作源文档，以及消息标头作为源属性，其中范围等于 Document ID。		字符串
targetMapMode (producer)	**TargetMapMode** enum 值，以指定在存在多个目标文档时如何发送它们。 Enum 值： * MAP * MESSAGE_HEADER * EXCHANGE_PROPERTY	**MAP**: 将文档存储在 **java.util.Map** 中。如果指定了 **targetMapName**，则 **java.util.Map** 设置为 exchange 属性，否则设置为消息正文。**MESSAGE_HEADER**：将它们存储在消息标头中。**EXCHANGE_PROPERTY**：将它们存储在交换属性中。	**MAP**	TargetMapMode

11.3. 例子

11.3.1. 生成者示例

以下示例显示了来自 AtlasMap Data Mapper UI 中的 和 ADM 归档文件导出：

```
from("activemq:My.Queue").
  to("atlas:atlasmap-mapping.adm");
```

Apache Camel 组件参考端点没有路径参数。

第 12 章 一个TMOSPHERE WEBSOCKET 组件

从 Camel 版本 2.14 开始提供

atmo consist-websocket: 组件为通过 Websocket 的外部客户端通信提供基于 Websocket 的端点（作为接受外部客户端的 websocket 连接的 servlet）。
组件使用 SERVLET 组件，并使用 Atmosphere 库来支持各种 Servlet 容器中的 Websocket 传输（如 Jetty、Tomcat、...）。

与启动嵌入式 Jetty 服务器的 Websocket 组件不同，此组件使用容器的 servlet 提供程序。

Maven 用户需要将以下依赖项添加到这个组件的 **pom.xml** 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-atmosphere-websocket</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

12.1. 一个TMOSPHERE-WEBSOCKET 选项

Atmosphere Websocket 组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
servletName (consumer)	要使用的默认 servlet 名称。默认名称为 CamelServlet。	Camel Servlet	字符串
httpRegistry (consumer)	使用自定义 org.apache.camel.component.servlet.HttpRegistry。		HttpRegistry
attachmentMultipart Binding (consumer)	是否自动将 multipart/form-data 作为 Camel Exchange 上的附件进行自动绑定。选项 attachmentMultipartBinding=true 和 disableStreamCache=false 无法一起工作。删除 disableStreamCache 以使用 AttachmentMultipartBinding。默认情况下关闭此设置，因为这可能需要 servlet 特定的配置，以便在使用 Servlet 时启用此功能。	false	布尔值
fileNameExtWhitelist (consumer)	接受上传的文件的已接受文件扩展名白名单。可以使用逗号分隔多个扩展，如 txt、xml。		字符串
httpBinding (advanced)	使用自定义 HttpBinding 来控制 Camel 消息和 HttpClient 之间的映射。		HttpBinding
httpConfiguration (advanced)	将共享 HttpConfiguration 用作基础配置。		HttpConfiguration

Name	描述	默认值	类型
允许 JavaSerialized Object (advanced)	当请求使用 context-type=application/x-java-serialized-object 时，是否允许 java serialization。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Atmosphere Websocket 端点使用 URI 语法进行配置：

```
atmosphere-websocket:servicePath
```

使用以下路径和查询参数：

12.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
servicePath	websocket 端点的名称		字符串

12.1.2. 查询参数(38 参数)：

Name	描述	默认值	类型
chunked (common)	如果此选项为 false，则 Servlet 将禁用 HTTP 流，并在响应上设置 content-length 标头	true	布尔值

Name	描述	默认值	类型
disableStreamCache (common)	确定 Servlet 中的原始输入流是否被缓存(Camel 将会在 memory/overflow 中读取流到文件，流缓存)缓存。默认情况下，Camel 将缓存 Servlet 输入流，以支持多次读取，以确保其 Camel 可以从流检索所有数据。但是，当您需要访问原始流时，您可以将此选项设置为 true，例如将其直接流传输到文件或其他持久性存储。DefaultHttpBinding 将请求输入流复制到流缓存中，如果此选项为 false，则将它放入消息正文，以支持多次读取流。如果您使用 Servlet 到 bridge/proxy 端点，则请考虑启用此选项来提高性能，如果您不需要多次读取消息有效负载。http/http4 producer 默认缓存响应正文流。如果将此选项设置为 true，则生成者不会缓存响应正文流，而是使用响应流作为消息正文。	false	布尔值
headerFilterStrategy (common)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
sendToAll (common)	是否发送到所有（广播）或发送到单个接收器。	false	布尔值
transferException (common)	如果在消费者端启用并交换失败处理，如果原因 Exception 在响应中作为 application/x-java-serialized-object 内容类型发送序列化，则结果为 application/x-java-serialized-object 内容类型。在生产者侧，异常将反序列化并丢弃为原样，而不是 HttpOperationFailedException。原因异常需要按顺序处理。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
useStreaming (common)	启用流，以作为多个文本片段发送数据。	false	布尔值
httpBinding (common)	使用自定义 HttpBinding 来控制 Camel 消息和 HttpClient 之间的映射。		HttpBinding
async (consumer)	将消费者配置为在 async 模式下工作	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值

Name	描述	默认值	类型
httpMethodRestrict (consumer)	仅在 HttpMethod 匹配时才允许使用，如 GET/POST/PUT 等。可以使用逗号分隔多个方法。		字符串
matchOnUriPrefix (consumer)	如果找不到完全匹配，消费者是否应该尝试通过匹配 URI 前缀来查找目标消费者。	false	布尔值
responseBufferSize (consumer)	在 javax.servlet.ServletResponse 上使用自定义缓冲区大小。		整数
servletName (consumer)	要使用的 servlet 的名称	Camel Servlet	字符串
attachmentMultipartBinding (consumer)	是否自动将 multipart/form-data 作为 Camel Exchange 上的附件进行自动绑定。选项 attachmentMultipartBinding=true 和 disableStreamCache=false 无法一起工作。删除 disableStreamCache 以使用 AttachmentMultipartBinding。默认情况下关闭此设置，因为这可能需要 servlet 特定的配置，以便在使用 Servlet 时启用此功能。	false	布尔值
eagerCheckContentAvailable (consumer)	如果 content-length 标头为 0，还是 eager 检查 HTTP 请求是否有内容。如果 HTTP 客户端没有发送流数据，则可以打开此项。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
fileNameExtWhitelist (consumer)	接受上传的文件的已接受文件扩展名白名单。可以使用逗号分隔多个扩展，如 txt、xml。		字符串
optionsEnabled (consumer)	指定是否为这个 Servlet 消费者启用 HTTP OPTIONS。默认情况下关闭 OPTIONS。	false	布尔值
traceEnabled (consumer)	指定是否为这个 Servlet 使用者启用 HTTP TRACE。默认情况下关闭 TRACE。	false	布尔值
bridgeEndpoint (producer)	如果选项为 true，则 HttpProducer 将忽略 Exchange.HTTP_URI 标头，并使用端点的 URI 进行请求。您也可以将 选项 throwExceptionOnFailure 设置为 false，让 HttpProducer 发送所有错误响应。	false	布尔值

Name	描述	默认值	类型
connectionClose (producer)	指定是否应将 Connection Close 标头添加到 HTTP Request 中。默认情况下，connectionClose 为 false。	false	布尔值
copyHeaders (producer)	如果此选项为 true，则 IN Exchange 标头将根据复制策略复制到 OUT 交换标头。把它设置为 false，仅允许包含来自 HTTP 响应的标头（而不是传播 IN 标头）。	true	布尔值
httpMethod (producer)	配置要使用的 HTTP 方法。如果设置，HttpMethod 标头无法覆盖此选项。		HttpMethods
ignoreResponseBody (producer)	如果此选项为 true，http producer 不会读取响应正文，并缓存输入流	false	布尔值
preserveHostHeader (producer)	如果选项为 true，HttpProducer 会将 Host 标头设置为当前交换主机标头中包含的值，这适用于您希望下游服务器收到的 Host 标头来反映上游客户端调用的 URL 的 URL，这将允许使用 Host 标头为代理服务生成准确的 URL 的应用	false	布尔值
throwExceptionOnFailure (producer)	如果来自远程服务器的失败响应，用于禁用抛出 HttpOperationFailedException。这样，您可以获取所有响应，而不考虑 HTTP 状态代码。	true	布尔值
cookieHandler (producer)	配置 Cookie 处理程序，以维护 HTTP 会话		CookieHandler
okStatusCodeRange (producer)	被视为成功响应的状态代码。值包含。可以定义多个范围，用逗号分开，例如 200-204,209,301-304。每个范围必须是单个数字，或从 到，其中包含短划线。	200-299	字符串
urlRewrite (producer)	**弃用** 引用自定义 org.apache.camel.component.http.UrlRewrite，它允许您在 bridge/proxy 端点时重写 url。更多信息，请访问 http://camel.apache.org/urlrewrite.html		UrlRewrite
mapHttpMessageBody (advanced)	如果此选项为 true，则交换的 IN Exchange Body 将映射到 HTTP 正文。把它设置为 false 可以避免 HTTP 映射。	true	布尔值
mapHttpMessageFormUrlEncodedBody (advanced)	如果此选项为 true，则交换的 IN Exchange Form Encoded body 将映射到 HTTP。把它设置为 false 可以避免 HTTP Form Encoded body 映射。	true	布尔值

Name	描述	默认值	类型
mapHttpMessageHeaders (advanced)	如果此选项为 true，则交换的 IN Exchange Headers 将映射到 HTTP 标头。把它设置为 false 将避免 HTTP 标头映射。	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
proxyAuthScheme (proxy)	要使用的代理身份验证方案		字符串
proxyHost (proxy)	要使用的代理主机名		字符串
proxyPort (proxy)	要使用的代理端口		int
authHost (security)	与 NTML 搭配使用的身份验证主机		字符串

12.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
camel.component.atmosphere-websocket.allow-java-serialized-object	当请求使用 context-type=application/x-java-serialized-object 时，是否允许 java serialization。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
camel.component.atmosphere-websocket.attachment-multipart-binding	是否自动将 multipart/form-data 作为 Camel Exchange 上的附件进行自动绑定。选项 attachmentMultipartBinding=true 和 disableStreamCache=false 无法一起工作。删除 disableStreamCache 以使用 AttachmentMultipartBinding。默认情况下关闭此设置，因为这可能需要 servlet 特定的配置，以便在使用 Servlet 时启用此功能。	false	布尔值
camel.component.atmosphere-websocket.enabled	启用tmosphere-websocket 组件	true	布尔值

Name	描述	默认值	类型
camel.component.atmosphere-websocket.file-name-ext-whitelist	接受上传的文件的已接受文件扩展名白名单。可以使用逗号分隔多个扩展，如 txt、xml。		字符串
camel.component.atmosphere-websocket.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		字符串
camel.component.atmosphere-websocket.http-binding	使用自定义 HttpBinding 来控制 Camel 消息和 HttpClient 之间的映射。选项是 org.apache.camel.http.common.HttpBinding 类型。		字符串
camel.component.atmosphere-websocket.http-configuration	将共享 HttpConfiguration 用作基础配置。选项是 org.apache.camel.http.common.HttpConfiguration 类型。		字符串
camel.component.atmosphere-websocket.http-registry	使用自定义 org.apache.camel.component.servlet.HttpRegistry。选项是 org.apache.camel.component.servlet.HttpRegistry 类型。		字符串
camel.component.atmosphere-websocket.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.atmosphere-websocket.servlet-name	要使用的默认 servlet 名称。默认名称为 CamelServlet。	Camel Servlet	字符串

12.3. URI 格式

```
atmosphere-websocket:///relative path[?options]
```

12.4. 通过 WEBSOCKET 读取和写入数据

atmopshere-websocket 端点可以将数据写入套接字，或者从套接字读取，具体取决于端点是否配置为生成者或消费者。

12.5. 将 URI 配置为读或写数据

在以下路由中，Camel 将从指定的 websocket 连接中读取。

```
from("atmosphere-websocket:///servicepath")
        .to("direct:next");
```

以及等同的 Spring 示例：

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="atmosphere-websocket:///servicepath"/>
    <to uri="direct:next"/>
  </route>
</camelContext>
```

在以下路由中，Camel 将从指定的 websocket 连接中读取。

```
from("direct:next")
        .to("atmosphere-websocket:///servicepath");
```

以及等同的 Spring 示例：

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="direct:next"/>
    <to uri="atmosphere-websocket:///servicepath"/>
  </route>
</camelContext>
```

12.6. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- SERVLET
- AHC-WS * Websocket

第 13 章 ATOM 组件

从 Camel 版本 1.2 开始提供

atom: 组件用于轮询 Atom 源。

默认情况下，Camel 将每 60 秒轮询一次源。
注意： 组件目前仅支持轮询（耗时）源。

Maven 用户需要将以下依赖项添加到这个组件的 **pom.xml** 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-atom</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

13.1. URI 格式

```
atom://atomUri[?options]
```

其中 **atomUri** 是要轮询的 Atom 源的 URI。

13.2. 选项

Atom 组件没有选项。

Atom 端点使用 URI 语法进行配置：

```
atom:feedUri
```

使用以下路径和查询参数：

13.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
feedUri	**需要** 要轮询的源的 URI。		字符串

13.2.2. 查询参数(27 参数)：

Name	描述	默认值	类型

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
feedHeader (consumer)	设置是否将 source 对象添加为标头	true	布尔值
filter (consumer)	设置是否使用过滤或条目。	true	布尔值
lastUpdate (consumer)	设置用于从 atom 源过滤条目的时间戳。这个选项只与 splitEntries 结合使用。		Date
password (consumer)	设置在从 HTTP 源轮询时用于基本身份验证的密码		字符串
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
sortEntries (consumer)	设置是否按公布的日期对条目进行排序。仅在 splitEntries = true 时才有效。	false	布尔值
splitEntries (consumer)	设置条目是否应单独发送，还是将整个源作为单个消息发送	true	布尔值
throttleEntries (consumer)	设置在单个源轮询中识别的所有条目是否应立即发送。如果为 true，则每个 consumer.delay 只处理一个条目。仅在 splitEntries = true 时才适用。	true	布尔值
用户名 (consumer)	设置在从 HTTP 源轮询时用于基本身份验证的用户名		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

13.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.atom.enabled	启用 atom 组件	true	布尔值
camel.component.atom.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

您可以在 URI 中附加查询选项，格式为 **?option=value&option=value&...**

13.4. 交换数据格式

Camel 将在返回的 **Exchange** 上使用条目设置 In 正文。根据 **splitEntries** 标志 Camel 将返回一个条目或 **List< Entry >**。

选项	value	行为
splitEntries	**true**	仅设置当前被处理源中的单个条目： **exchange.in.body (Entry)**
splitEntries	**false**	源的整个条目列表被设置： **exchange.in.body (List<Entry>)**

Camel 可以在 In 标头上设置 **Feed** 对象（请参阅 **feedHeader** 选项来禁用此功能）：

13.5. 消息标头

Camel atom 使用这些标头。

标头	描述
Camel Atom Feed	在消耗 **org.apache.abdera.model.Feed** 对象时，会将这个标头设置为这个标头。

13.6. SAMPLES

在本示例中，我们轮询 regarding Strachan 的博客。

```
from("atom://http://macstrac.blogspot.com/feeds/posts/default").to("seda:feeds");
```

在本例中，我们只想过滤我们喜欢的 SEDA 队列的好博客。示例还演示了如何设置 Camel 独立，而不是在任何容器或使用 Spring 中运行。

13.7. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- RSS

第 14 章 ATOMIX MAP 组件

从 Camel 版本 2.20 开始提供

通过 camel atomix-map 组件，您可以使用 Atomix 的 Distributed Map 集合。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-atomix</artifactId>
  <version>${camel-version}</version>
</dependency>
```

14.1. URI 格式

```
atomix-map:mapName
```

14.2. 选项

Atomix Map 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	共享组件配置		AtomixMapConfiguration
atomix (common)	shared AtomixClient 实例		AtomixClient
nodes (common)	AtomixClient 应该连接到的节点		list
configurationUri (common)	AtomixClient 配置的路径		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Atomix Map 端点使用 URI 语法进行配置：

```
atomix-map:resourceName
```

使用以下路径和查询参数：

14.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceName	**所需的** 分布式资源名称		字符串

14.2.2. 查询参数(18 参数)：

Name	描述	默认值	类型
atomix (common)	要使用的 Atomix 实例		Atomix
configurationUri (common)	Atomix 配置 uri.		字符串
defaultAction (common)	默认操作。	PUT	操作
key (common)	如果标头中设置了 none 或监听特定密钥的事件，则使用密钥。		对象
nodes (common)	组成集群的节点地址。		字符串
resultHeader (common)	唤醒结果的标头。		字符串
transport (common)	设置 Atomix 传输。	io.atomix.catalyst.transport.netty.NettyTransport	传输
TTL (common)	资源 ttl。		long
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern

Name	描述	默认值	类型
defaultResourceConfig (advanced)	集群范围内的默认资源配置。		Properties
defaultResourceOptions (advanced)	本地默认资源选项。		Properties
Ephemeral (advanced)	设置本地成员是否应该将组加入为 PersistentMember 或 not。如果设置为 ephemeral，本地成员将收到自动生成的 ID，因此本地成员将被忽略。	false	布尔值
readConsistency (advanced)	读取一致性级别。		ReadConsistency
resourceConfigs (advanced)	集群范围内的资源配置。		Map
resourceOptions (advanced)	本地资源配置		Map
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

14.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
camel.component.atomix-map.atomix	shared AtomixClient 实例。选项是一个 io.atomix.AtomixClient 类型。		字符串
camel.component.atomix-map.configuration-uri	AtomixClient 配置的路径		字符串
camel.component.atomix-map.configuration.default-action	默认操作。		AtomixMap$Action
camel.component.atomix-map.configuration.key	如果标头中设置了 none 或监听特定密钥的事件，则使用密钥。		对象

Name	描述	默认值	类型
camel.component.atomix-map.configuration.result-header	唤醒结果的标头。		字符串
camel.component.atomix-map.configuration.ttl	资源 ttl。		Long
camel.component.atomix-map.enabled	是否启用 atomix-map 组件的自动配置。这默认是启用的。		布尔值
camel.component.atomix-map.nodes	AtomixClient 应该连接到的节点		list
camel.component.atomix-map.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

14.4. HEADERS

Name	类型	值	描述

Name	类型	值	描述
CamelAtomixResourceAction	AtomixMap.Action	• PUT • PUT_IF_ABSENT • GET • CLEAR • SIZE • CONTAINS_KEY • CONTAINS_VALUE • IS_EMPTY • ENTRY_SET • 删除 • REPLACE • 值	要执行的操作
CamelAtomixResourceKey	对象	-	操作的密钥
CamelAtomixResourceValue	对象	-	如果使用了在 Body 中缺少的值
CamelAtomixResourceOldValue	对象	-	旧值
CamelAtomixResourceTTL	字符串 / long	-	条目 TTL

Name	类型	值	描述
Camel Atomix ResourceReadConsistency	ReadConsistency	• ATOMIC • ATOMIC_LEASE • 顺序 • LOCAL	读取一致性级别

14.5. 配置组件以连接到 ATOMIX 集群

要加入的 Atomix 集群的节点可以在 Endpoint 或组件级别（推荐）中，在一些示例下：

- **端点：**

```
<beans xmlns="...">
    <camelContext xmlns="http://camel.apache.org/schema/spring">
        <from uri="direct:start"/>
            <to uri="atomix-map:myMap?nodes=node-1.atomix.cluster:8700,node-
2.atomix.cluster:8700"/>
        </route>
    </camelContext>
</beans>
```

- **component:**

```
<beans xmlns="...">
    <bean id="atomix-map"
class="org.apache.camel.component.atomix.client.map.AtomixMapComponent">
        <property name="nodes" value="nodes=node-1.atomix.cluster:8700,node-
2.atomix.cluster:8700"/>
    </bean>

    <camelContext xmlns="http://camel.apache.org/schema/spring">
        <from uri="direct:start"/>
            <to uri="atomix-map:myMap"/>
        </route>
    </camelContext>
</beans>
```

14.6. 使用示例：

- **使用 TTL 为 1 秒的 PUT 元素：**

```
FluentProducerTemplate.on(context)
    .withHeader(AtomixClientConstants.RESOURCE_ACTION, AtomixMap.Action.PUT)
    .withHeader(AtomixClientConstants.RESOURCE_KEY, key)
    .withHeader(AtomixClientConstants.RESOURCE_TTL, "1s")
```

```
        .withBody(val)
        .to("direct:start")
        .send();
```

第 15 章 ATOMIX 消息传递组件

从 Camel 版本 2.20 开始提供

通过 camel atomix-messaging 组件，您可以使用 Atomix 的 Group Messaging。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-atomix</artifactId>
  <version>${camel-version}</version>
</dependency>
```

15.1. URI 格式

```
atomix-messaging:group
```

Atomix Messaging 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	共享组件配置		AtomixMessaging 配置
atomix (common)	shared AtomixClient 实例		AtomixClient
nodes (common)	AtomixClient 应该连接到的节点		list
configurationUri (common)	AtomixClient 配置的路径		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Atomix Messaging 端点使用 URI 语法进行配置：

```
atomix-messaging:resourceName
```

使用以下路径和查询参数：

15.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceName	**所需的** 分布式资源名称		字符串

15.1.2. 查询参数(19 参数)：

Name	描述	默认值	类型
atomix (common)	要使用的 Atomix 实例		Atomix
broadcastType (common)	广播类型。	ALL	BroadcastType
channelName (common)	消息传递频道名称		字符串
configurationUri (common)	Atomix 配置 uri.		字符串
defaultAction (common)	默认操作。	DIRECT	操作
memberName (common)	Atomix 组成员名称		字符串
nodes (common)	组成集群的节点地址。		字符串
resultHeader (common)	唤醒结果的标头。		字符串
transport (common)	设置 Atomix 传输。	io.atomix.catalyst.transport.netty.NettyTransport	传输
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern

Name	描述	默认值	类型
defaultResourceConfig (advanced)	集群范围内的默认资源配置。		Properties
defaultResourceOptions (advanced)	本地默认资源选项。		Properties
Ephemeral (advanced)	设置本地成员是否应该将组加入为 PersistentMember 或 not。如果设置为 ephemeral，本地成员将收到自动生成的 ID，因此本地成员将被忽略。	false	布尔值
readConsistency (advanced)	读取一致性级别。		ReadConsistency
resourceConfigs (advanced)	集群范围内的资源配置。		Map
resourceOptions (advanced)	本地资源配置		Map
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

15.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
camel.component.atomix-messaging.atomix	shared AtomixClient 实例。选项是一个 io.atomix.AtomixClient 类型。		字符串
camel.component.atomix-messaging.configuration-uri	AtomixClient 配置的路径		字符串
camel.component.atomix-messaging.configuration.broadcast-type	广播类型。		AtomixMessaging $ BroadcastType

Name	描述	默认值	类型
camel.component.atomix-messaging.configuration.channel-name	消息传递频道名称		字符串
camel.component.atomix-messaging.configuration.default-action	默认操作。		AtomixMessaging$Action
camel.component.atomix-messaging.configuration.member-name	Atomix 组成员名称		字符串
camel.component.atomix-messaging.configuration.result-header	唤醒结果的标头。		字符串
camel.component.atomix-messaging.enabled	是否启用 atomix-messaging 组件的自动配置。这默认是启用的。		布尔值
camel.component.atomix-messaging.nodes	AtomixClient 应该连接到的节点		list
camel.component.atomix-messaging.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 16 章 ATOMIX MULTIMAP 组件

从 Camel 版本 2.20 开始提供

camel atomix-multimap 组件允许您使用 Atomix 的 Distributed MultiMap 集合。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-atomix</artifactId>
  <version>${camel-version}</version>
</dependency>
```

16.1. URI 格式

```
atomix-multimap:multiMapName
```

Atomix MultiMap 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
configuration (consumer)	共享组件配置		AtomixMultiMap 配置
atomix (consumer)	shared AtomixClient 实例		AtomixClient
nodes (consumer)	AtomixClient 应该连接到的节点		list
configurationUri (consumer)	AtomixClient 配置的路径		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Atomix MultiMap 端点使用 URI 语法进行配置：

```
atomix-multimap:resourceName
```

使用以下路径和查询参数：

16.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceName	**所需的** 分布式资源名称		字符串

16.1.2. 查询参数(18 参数)：

Name	描述	默认值	类型
atomix (consumer)	要使用的 Atomix 实例		Atomix
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
configurationUri (consumer)	Atomix 配置 uri.		字符串
defaultAction (consumer)	默认操作。	PUT	操作
key (consumer)	如果标头中设置了 none 或监听特定密钥的事件，则使用密钥。		对象
nodes (consumer)	组成集群的节点地址。		字符串
resultHeader (consumer)	唤醒结果的标头。		字符串
Transport (consumer)	设置 Atomix 传输。	io.atomix.catalyst.transport.netty.NettyTransport	传输
TTL (consumer)	资源 ttl。		long
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
defaultResourceConfig (advanced)	集群范围内的默认资源配置。		Properties

Name	描述	默认值	类型
defaultResourceOptions (advanced)	本地默认资源选项。		Properties
Ephemeral (advanced)	设置本地成员是否应该将组加入为 PersistentMember 或 not。如果设置为 ephemeral，本地成员将收到自动生成的 ID，因此本地成员将被忽略。	false	布尔值
readConsistency (advanced)	读取一致性级别。		ReadConsistency
resourceConfigs (advanced)	集群范围内的资源配置。		Map
resourceOptions (advanced)	本地资源配置		Map
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

16.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
camel.component.atomix-multimap.atomix	shared AtomixClient 实例。选项是一个 io.atomix.AtomixClient 类型。		字符串
camel.component.atomix-multimap.configuration-uri	AtomixClient 配置的路径		字符串
camel.component.atomix-multimap.configuration.default-action	默认操作。		AtomixMultiMap$Action
camel.component.atomix-multimap.configuration.key	如果标头中设置了 none 或监听特定密钥的事件，则使用密钥。		对象

Name	描述	默认值	类型
camel.component.atomix-multimap.configuration.result-header	唤醒结果的标头。		字符串
camel.component.atomix-multimap.configuration.ttl	资源 ttl。		Long
camel.component.atomix-multimap.enabled	是否启用 atomix-multimap 组件的自动配置。这默认是启用的。		布尔值
camel.component.atomix-multimap.nodes	AtomixClient 应该连接到的节点		list
camel.component.atomix-multimap.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 17 章 ATOMIX QUEUE 组件

从 Camel 版本 2.20 开始提供

camel atomix-queue 组件允许您处理 Atomix 的分布式 队列集合。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-atomix</artifactId>
  <version>${camel-version}</version>
</dependency>
```

17.1. URI 格式

```
atomix-queue:queueName
```

Atomix Queue 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	共享组件配置		AtomixQueue 配置
atomix (common)	shared AtomixClient 实例		AtomixClient
nodes (common)	AtomixClient 应该连接到的节点		list
configurationUri (common)	AtomixClient 配置的路径		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Atomix Queue 端点使用 URI 语法进行配置：

```
atomix-queue:resourceName
```

使用以下路径和查询参数：

17.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceName	**所需的** 分布式资源名称		字符串

17.1.2. 查询参数(16 参数)：

Name	描述	默认值	类型
atomix (common)	要使用的 Atomix 实例		Atomix
configurationUri (common)	Atomix 配置 uri.		字符串
defaultAction (common)	默认操作。	添加	操作
nodes (common)	组成集群的节点地址。		字符串
resultHeader (common)	唤醒结果的标头。		字符串
transport (common)	设置 Atomix 传输。	io.atomix.catalyst.transport.netty.NettyTransport	传输
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
defaultResourceConfig (advanced)	集群范围内的默认资源配置。		Properties
defaultResourceOptions (advanced)	本地默认资源选项。		Properties
Ephemeral (advanced)	设置本地成员是否应该将组加入为 PersistentMember 或 not。如果设置为 ephemeral，本地成员将收到自动生成的 ID，因此本地成员将被忽略。	false	布尔值

Name	描述	默认值	类型
readConsistency (advanced)	读取一致性级别。		ReadConsistency
resourceConfigs (advanced)	集群范围内的资源配置。		Map
resourceOptions (advanced)	本地资源配置		Map
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

17.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 7 个选项，如下所列。

Name	描述	默认值	类型
camel.component.atomix-queue.atomix	shared AtomixClient 实例。选项是一个 io.atomix.AtomixClient 类型。		字符串
camel.component.atomix-queue.configuration-uri	AtomixClient 配置的路径		字符串
camel.component.atomix-queue.configuration.default-action	默认操作。		AtomixQueue$Action
camel.component.atomix-queue.configuration.result-header	唤醒结果的标头。		字符串
camel.component.atomix-queue.enabled	是否启用 atomix-queue 组件的自动配置。这默认是启用的。		布尔值
camel.component.atomix-queue.nodes	AtomixClient 应该连接到的节点		list

Name	描述	默认值	类型
camel.component.atomix-queue.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 18 章 ATOMIX 设置组件

从 Camel 版本 2.20 开始提供

camel atomix-set 组件允许您使用 Atomix 的分布式集合集合。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-atomix</artifactId>
  <version>${camel-version}</version>
</dependency>
```

18.1. URI 格式

```
atomix-set:setName
```

Atomix Set 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	共享组件配置		AtomixSetConfiguration
atomix (common)	shared AtomixClient 实例		AtomixClient
nodes (common)	AtomixClient 应该连接到的节点		list
configurationUri (common)	AtomixClient 配置的路径		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Atomix Set 端点使用 URI 语法进行配置：

```
atomix-set:resourceName
```

使用以下路径和查询参数：

18.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceName	**所需的** 分布式资源名称		字符串

18.1.2. 查询参数(17 参数)：

Name	描述	默认值	类型
atomix (common)	要使用的 Atomix 实例		Atomix
configurationUri (common)	Atomix 配置 uri.		字符串
defaultAction (common)	默认操作。	添加	操作
nodes (common)	组成集群的节点地址。		字符串
resultHeader (common)	唤醒结果的标头。		字符串
transport (common)	设置 Atomix 传输。	io.atomix.catalyst.transport.netty.NettyTransport	传输
TTL (common)	资源 ttl。		long
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
defaultResourceConfig (advanced)	集群范围内的默认资源配置。		Properties
defaultResourceOptions (advanced)	本地默认资源选项。		Properties

Name	描述	默认值	类型
Ephemeral (advanced)	设置本地成员是否应该将组加入为 PersistentMember 或 not。如果设置为 ephemeral，本地成员将收到自动生成的 ID，因此本地成员将被忽略。	false	布尔值
readConsistency (advanced)	读取一致性级别。		ReadConsistency
resourceConfigs (advanced)	集群范围内的资源配置。		Map
resourceOptions (advanced)	本地资源配置		Map
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

18.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 8 个选项，如下所列。

Name	描述	默认值	类型
camel.component.atomix-set.atomix	shared AtomixClient 实例。选项是一个 io.atomix.AtomixClient 类型。		字符串
camel.component.atomix-set.configuration-uri	AtomixClient 配置的路径		字符串
camel.component.atomix-set.configuration.default-action	默认操作。		AtomixSet$Action
camel.component.atomix-set.configuration.result-header	唤醒结果的标头。		字符串
camel.component.atomix-set.configuration.ttl	资源 ttl。		Long

Name	描述	默认值	类型
camel.component.atomix-set.enabled	是否启用 atomix-set 组件的自动配置。这默认是启用的。		布尔值
camel.component.atomix-set.nodes	AtomixClient 应该连接到的节点		list
camel.component.atomix-set.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 19 章 ATOMIX 值组件

从 Camel 版本 2.20 开始提供

camel atomix-value 组件允许您使用 Atomix 的分布式值。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-atomix</artifactId>
  <version>${camel-version}</version>
</dependency>
```

19.1. URI 格式

```
atomix-value:valueName
```

Atomix Value 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	共享组件配置		AtomixValue 配置
atomix (common)	shared AtomixClient 实例		AtomixClient
nodes (common)	AtomixClient 应该连接到的节点		list
configurationUri (common)	AtomixClient 配置的路径		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Atomix Value 端点使用 URI 语法进行配置：

```
atomix-value:resourceName
```

使用以下路径和查询参数：

19.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceName	**所需的** 分布式资源名称		字符串

19.1.2. 查询参数(17 参数)：

Name	描述	默认值	类型
atomix (common)	要使用的 Atomix 实例		Atomix
configurationUri (common)	Atomix 配置 uri.		字符串
defaultAction (common)	默认操作。	SET	操作
nodes (common)	组成集群的节点地址。		字符串
resultHeader (common)	唤醒结果的标头。		字符串
transport (common)	设置 Atomix 传输。	io.atomix.catalyst.transport.netty.NettyTransport	传输
TTL (common)	资源 ttl。		long
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
defaultResourceConfig (advanced)	集群范围内的默认资源配置。		Properties
defaultResourceOptions (advanced)	本地默认资源选项。		Properties

Name	描述	默认值	类型
Ephemeral (advanced)	设置本地成员是否应该将组加入为 PersistentMember 或 not。如果设置为 ephemeral，本地成员将收到自动生成的 ID，因此本地成员将被忽略。	false	布尔值
readConsistency (advanced)	读取一致性级别。		ReadConsistency
resourceConfigs (advanced)	集群范围内的资源配置。		Map
resourceOptions (advanced)	本地资源配置		Map
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

19.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 8 个选项，如下所列。

Name	描述	默认值	类型
camel.component.atomix-value.atomix	shared AtomixClient 实例。选项是一个 io.atomix.AtomixClient 类型。		字符串
camel.component.atomix-value.configuration-uri	AtomixClient 配置的路径		字符串
camel.component.atomix-value.configuration.default-action	默认操作。		AtomixValue$Action
camel.component.atomix-value.configuration.result-header	唤醒结果的标头。		字符串
camel.component.atomix-value.configuration.ttl	资源 ttl。		Long

Name	描述	默认值	类型
camel.component.atomix-value.enabled	是否启用 atomix-value 组件的自动配置。这默认是启用的。		布尔值
camel.component.atomix-value.nodes	AtomixClient 应该连接到的节点		list
camel.component.atomix-value.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 20 章 AVRO 组件

从 Camel 版本 2.10 开始提供

此组件为 avro 提供了一个 dataformat，允许使用 Apache Avro 的二进制 dataformat 序列化和反序列化消息。此外，它还通过为使用 avro over netty 或 http 提供生产者和消费者端点，从而提供对 Apache Avro 的 rpc 的支持。

Maven 用户需要将以下依赖项添加到这个组件的 **pom.xml** 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-avro</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

20.1. APACHE AVRO 概述

avro 允许您使用 json 等 json 定义消息类型和协议，然后为指定的类型和消息生成 java 代码。如下为 schema 的示例。

```
{"namespace": "org.apache.camel.avro.generated",
 "protocol": "KeyValueProtocol",

 "types": [
    {"name": "Key", "type": "record",
     "fields": [
         {"name": "key",   "type": "string"}
     ]
    },
    {"name": "Value", "type": "record",
     "fields": [
         {"name": "value",   "type": "string"}
     ]
    }
 ],

 "messages": {
    "put": {
       "request": [{"name": "key", "type": "Key"}, {"name": "value", "type": "Value"} ],
       "response": "null"
    },
    "get": {
       "request": [{"name": "key", "type": "Key"}],
       "response": "Value"
    }
 }
}
```

您可以使用 maven 和 ant 等从架构轻松生成类。更多详情请参阅 Apache Avro 文档。

但是，它不强制执行一个模式，第一个方法，您可以为现有类创建模式。由于 2.12 您可以使用现有协议接口进行 RCP 调用。您应该将接口用于协议本身，POJO Bean 或原语/字符串类用于参数和结果类型。以下是与上述 schema 对应的类示例：

```
package org.apache.camel.avro.reflection;

public interface KeyValueProtocol {
    void put(String key, Value value);
    Value get(String key);
}

class Value {
    private String value;
    public String getValue() { return value; }
    public void setValue(String value) { this.value = value; }
}
```

注意：现有类只能用于 RPC （请参阅以下），而不是数据格式。

20.2. 使用 AVRO 数据格式

使用 avro 数据格式就像指定您要在路由中 marshal 或 unmarshal 的类一样容易。

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
    <route>
        <from uri="direct:in"/>
        <marshal>
            <avro instanceClass="org.apache.camel.dataformat.avro.Message"/>
        </marshal>
        <to uri="log:out"/>
    </route>
</camelContext>
```

另一种方法是在上下文中指定 dataformat，并从您的路由引用它。

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
     <dataFormats>
        <avro id="avro" instanceClass="org.apache.camel.dataformat.avro.Message"/>
    </dataFormats>
    <route>
        <from uri="direct:in"/>
        <marshal ref="avro"/>
        <to uri="log:out"/>
    </route>
</camelContext>
```

同样，您可以使用 avro 数据格式进行 umarshal。

20.3. 在 CAMEL 中使用 AVRO RPC

如前文所述，Avro 还提供对多个传输（如 http 和 netty）的 RPC 支持。Camel 为这两个传输提供使用者和制作者。

```
avro:[transport]:[host]:[port][?options]
```

支持的传输值目前为 http 或 netty。

因为 2.12，您可以在 URI 中指定消息名称：

```
avro:[transport]:[host]:[port][/messageName][?options]
```

对于消费者，您可以将多个路由附加到同一套接字。分配给正确的路由将自动由 avro 组件完成。没有指定 messageName （若有）的路由将用作默认值。

当将 camel producer 用于 avro ipc 时，"in"消息正文需要包含 avro 协议中指定的操作的参数。响应将在"out"消息的正文中添加。

在使用 avro ipc 的 camel avro consumers 时，请求参数将放置在创建的交换的"in"消息正文中，并在处理"out"消息的正文后，将作为响应发送。

注： 默认情况下，消费者参数被嵌套成数组。如果您只有一个参数，因为 2.12 可以使用 **singleParameter** URI 选项在 "in" 消息正文中接收它，而无需数组换行。

20.4. AVRO RPC URI 选项

Avro 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	使用共享 AvroConfiguration 一次配置选项		AvroConfiguration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Avro 端点使用 URI 语法进行配置：

```
avro:transport:host:port/messageName
```

使用以下路径和查询参数：

20.4.1. 路径参数(4 参数)：

Name	描述	默认值	类型
传输	**需要** 传输要使用的传输，可以是 http 或 netty		AvroTransport
port	要使用 **所需的** 端口号		int
主机	要使用的主机名		字符串
messageName	要发送的消息的名称。		字符串

20.4.2. 查询参数(10 parameters):

Name	描述	默认值	类型
protocol (common)	要使用的avro 协议		协议
protocolClassName (common)	使用 FQN 类名称定义的avro 协议		字符串
protocolLocation (common)	avro 协议位置		字符串
reflectionProtocol (common)	如果提供的协议对象反映协议。应该只与 protocol 参数一起使用，因为会自动检测到 protocolClassName 协议类型	false	布尔值
singleParameter (common)	如果为 true，则 consumer 参数不会嵌套成数组。如果协议为消息指定了更多 1 参数，将失败	false	布尔值
uriAuthority (common)	要使用的授权（用户名和密码）		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

20.5. SPRING BOOT AUTO-CONFIGURATION

组件支持 15 个选项，如下所列。

Name	描述	默认值	类型
camel.component.avro.configuration.host	要使用的主机名		字符串
camel.component.avro.configuration.message-name	要发送的消息的名称。		字符串
camel.component.avro.configuration.port	要使用的端口号		整数
camel.component.avro.configuration.protocol	要使用的avro 协议		协议
camel.component.avro.configuration.protocol-class-name	使用 FQN 类名称定义的avro 协议		字符串
camel.component.avro.configuration.protocol-location	avro 协议位置		字符串
camel.component.avro.configuration.reflection-protocol	如果提供的协议对象反映协议。应该只与 protocol 参数一起使用，因为会自动检测到 protocolClassName 协议类型	false	布尔值
camel.component.avro.configuration.single-parameter	如果为 true，则 consumer 参数不会嵌套成数组。如果协议为消息指定了更多 1 参数，将失败	false	布尔值
camel.component.avro.configuration.transport	要使用的传输，可以是 http 或 netty		AvroTransport
camel.component.avro.configuration.uri-authority	要使用的授权（用户名和密码）		字符串
camel.component.avro.enabled	启用 avro 组件	true	布尔值

Name	描述	默认值	类型
camel.component.avro.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.dataformat.avro.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.avro.enabled	启用 avro dataformat	true	布尔值
camel.dataformat.avro.instance-class-name	用于 marshal 和 unmarshalling 的类名称		字符串

20.6. AVRO RPC HEADERS

Name	描述
CamelAvroMessageName	要发送的消息的名称。在消费者中，可覆盖 URI 中的消息名称（如果有）

20.7. 例子

通过 http 使用 camel avro producers 的示例：

```
        <route>
            <from uri="direct:start"/>
            <to uri="avro:http:localhost:{{avroport}}?
protocolClassName=org.apache.camel.avro.generated.KeyValueProtocol"/>
            <to uri="log:avro"/>
        </route>
```

在上例中，您需要填写 **CamelAvroMessageName** 标头。**因为 2.12**，您可以使用以下语法调用恒定的信息：

```
        <route>
            <from uri="direct:start"/>
            <to uri="avro:http:localhost:{{avroport}}/put?
protocolClassName=org.apache.camel.avro.generated.KeyValueProtocol"/>
            <to uri="log:avro"/>
        </route>
```

通过 netty 使用 camel avro consumers 消耗消息的示例：

```
        <route>
            <from uri="avro:netty:localhost:{{avroport}}?
protocolClassName=org.apache.camel.avro.generated.KeyValueProtocol"/>
            <choice>
                <when>
                    <el>${in.headers.CamelAvroMessageName == 'put'}</el>
                    <process ref="putProcessor"/>
                </when>
                <when>
                    <el>${in.headers.CamelAvroMessageName == 'get'}</el>
                    <process ref="getProcessor"/>
                </when>
            </choice>
        </route>
```

因为 2.12 您可以设置两个不同的路由来执行相同的任务：

```
        <route>
            <from uri="avro:netty:localhost:{{avroport}}/put?
protocolClassName=org.apache.camel.avro.generated.KeyValueProtocol">
            <process ref="putProcessor"/>
        </route>
        <route>
            <from uri="avro:netty:localhost:{{avroport}}/get?
protocolClassName=org.apache.camel.avro.generated.KeyValueProtocol&singleParameter=true"/>
            <process ref="getProcessor"/>
        </route>
```

在上例中，get 只使用一个参数，因此使用 **singleParameter**，而 **getProcessor** 将直接在正文中接收 Value 类，而 **putProcessor** 将收到带有 String key 和 Value 值的数组，并填充为数组内容。

第 21 章 AVRO DATAFORMAT

从 Camel 版本 2.14 开始提供

此组件为 avro 提供了一个 dataformat，允许使用 Apache Avro 的二进制 dataformat 序列化和反序列化消息。此外，它还通过为使用 avro over netty 或 http 提供生产者和消费者端点，从而提供对 Apache Avro 的 rpc 的支持。

Maven 用户需要将以下依赖项添加到这个组件的 **pom.xml** 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-avro</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

21.1. APACHE AVRO 概述

avro 允许您使用 json 等 json 定义消息类型和协议，然后为指定的类型和消息生成 java 代码。如下为 schema 的示例。

```
{"namespace": "org.apache.camel.avro.generated",
 "protocol": "KeyValueProtocol",

 "types": [
    {"name": "Key", "type": "record",
     "fields": [
         {"name": "key",   "type": "string"}
     ]
    },
    {"name": "Value", "type": "record",
     "fields": [
         {"name": "value",   "type": "string"}
     ]
    }
 ],

 "messages": {
    "put": {
       "request": [{"name": "key", "type": "Key"}, {"name": "value", "type": "Value"} ],
       "response": "null"
    },
    "get": {
       "request": [{"name": "key", "type": "Key"}],
       "response": "Value"
    }
 }
}
```

您可以使用 maven 和 ant 等从架构轻松生成类。更多详情请参阅 Apache Avro 文档。

但是，它不强制执行一个模式，第一个方法，您可以为现有类创建模式。由于 2.12 您可以使用现有协议接口进行 RCP 调用。您应该将接口用于协议本身，POJO Bean 或原语/字符串类用于参数和结果类型。以下是与上述 schema 对应的类示例：

```
package org.apache.camel.avro.reflection;

public interface KeyValueProtocol {
    void put(String key, Value value);
    Value get(String key);
}

class Value {
    private String value;
    public String getValue() { return value; }
    public void setValue(String value) { this.value = value; }
}
```

注意：现有类只能用于 RPC （请参阅以下），而不是数据格式。

21.2. 使用 AVRO 数据格式

使用 avro 数据格式就像指定您要在路由中 marshal 或 unmarshal 的类一样容易。

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
    <route>
        <from uri="direct:in"/>
        <marshal>
            <avro instanceClass="org.apache.camel.dataformat.avro.Message"/>
        </marshal>
        <to uri="log:out"/>
    </route>
</camelContext>
```

另一种方法是在上下文中指定 dataformat，并从您的路由引用它。

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
     <dataFormats>
        <avro id="avro" instanceClass="org.apache.camel.dataformat.avro.Message"/>
    </dataFormats>
    <route>
        <from uri="direct:in"/>
        <marshal ref="avro"/>
        <to uri="log:out"/>
    </route>
</camelContext>
```

同样，您可以使用 avro 数据格式进行 umarshal。

21.3. AVRO DATAFORMAT 选项

Avro dataformat 支持 2 个选项，如下所列。

Name	默认值	Java 类型	描述
instanceClassName		字符串	用于 marshal 和 unmarshalling 的类名称
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

21.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 15 个选项，如下所列。

Name	描述	默认值	类型
camel.component.avro.configuration.host	要使用的主机名		字符串
camel.component.avro.configuration.message-name	要发送的消息的名称。		字符串
camel.component.avro.configuration.port	要使用的端口号		整数
camel.component.avro.configuration.protocol	要使用的avro 协议		协议
camel.component.avro.configuration.protocol-class-name	使用 FQN 类名称定义的avro 协议		字符串
camel.component.avro.configuration.protocol-location	avro 协议位置		字符串
camel.component.avro.configuration.reflection-protocol	如果提供的协议对象反映协议。应该只与 protocol 参数一起使用，因为会自动检测到 protocolClassName 协议类型	false	布尔值

Name	描述	默认值	类型
camel.component.avro.configuration.single-parameter	如果为 true，则 consumer 参数不会嵌套成数组。如果协议为消息指定了更多 1 参数，将失败	false	布尔值
camel.component.avro.configuration.transport	要使用的传输，可以是 http 或 netty		AvroTransport
camel.component.avro.configuration.uri-authority	要使用的授权（用户名和密码）		字符串
camel.component.avro.enabled	启用 avro 组件	true	布尔值
camel.component.avro.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.dataformat.avro.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.avro.enabled	启用 avro dataformat	true	布尔值
camel.dataformat.avro.instance-class-name	用于 marshal 和 unmarshalling 的类名称		字符串

ND

第 22 章 AWS CLOUDWATCH 组件

从 Camel 版本 2.11 开始提供

CW 组件允许消息发送到 Amazon CloudWatch 指标。Amazon API 的实现由 AWS SDK 提供。

先决条件

您必须有一个有效的 Amazon Web Services 开发人员帐户，并有权使用 Amazon CloudWatch。如需更多信息，请参阅 Amazon CloudWatch。

22.1. URI 格式

```
aws-cw://namespace[?options]
```

如果指标不存在，则会创建它们。
您可以在 URI 中附加查询选项，格式为 **?options=value&option2=value&...**

22.2. URI 选项

AWS CloudWatch 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	AWS CW 默认配置		CwConfiguration
AccessKey (producer)	Amazon AWS 访问密钥		字符串
secretKey (producer)	Amazon AWS Secret 密钥		字符串
region (producer)	CW 客户端需要工作的区域		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AWS CloudWatch 端点使用 URI 语法进行配置：

```
aws-cw:namespace
```

使用以下路径和查询参数：

22.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
namespace	**必需** 的指标命名空间		字符串

Name	描述	默认值	类型

22.2.2. 查询参数(11 参数)：

Name	描述	默认值	类型
amazonCwClient（生成者）	将 AmazonCloudWatch 用作客户端		AmazonCloudWatch
name (producer)	指标名称		字符串
proxyHost (producer)	在实例化 CW 客户端时定义代理主机		字符串
proxyPort (producer)	在实例化 CW 客户端时定义代理端口		整数
region (producer)	CW 客户端需要工作的区域		字符串
timestamp (producer)	指标时间戳		Date
unit (producer)	指标单元		字符串
value (producer)	指标值		å�Œ
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
accessKey (security)	Amazon AWS 访问密钥		字符串
secretKey (security)	Amazon AWS Secret 密钥		字符串

22.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 16 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws-cw.access-key	Amazon AWS 访问密钥		字符串

Name	描述	默认值	类型
camel.component.aws-cw.configuration.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-cw.configuration.amazon-cw-client	将 AmazonCloudWatch 用作客户端		AmazonCloudWatch
camel.component.aws-cw.configuration.name	指标名称		字符串
camel.component.aws-cw.configuration.namespace	指标命名空间		字符串
camel.component.aws-cw.configuration.proxy-host	在实例化 CW 客户端时定义代理主机		字符串
camel.component.aws-cw.configuration.proxy-port	在实例化 CW 客户端时定义代理端口		整数
camel.component.aws-cw.configuration.region	CW 客户端需要工作的区域		字符串
camel.component.aws-cw.configuration.secret-key	Amazon AWS Secret 密钥		字符串
camel.component.aws-cw.configuration.timestamp	指标时间戳		Date
camel.component.aws-cw.configuration.unit	指标单元		字符串

Name	描述	默认值	类型
camel.component.aws-cw.configuration.value	指标值		å€Œ
camel.component.aws-cw.enabled	启用 aws-cw 组件	true	布尔值
camel.component.aws-cw.region	CW 客户端需要工作的区域		字符串
camel.component.aws-cw.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.aws-cw.secret-key	Amazon AWS Secret 密钥		字符串

所需的 CW 组件选项

您必须在 Registry 或 accessKey 和 secretKey 中提供 amazonCwClient，才能访问 Amazon 的 CloudWatch。

22.4. 使用方法

22.4.1. 由 CW producer 评估的消息标头

标头	类型	描述
CamelAwsCwMetricName	**字符串**	Amazon CW 指标名称。
CamelAwsCwMetricValue	**å€Œ**	Amazon CW 指标值。
CamelAwsCwMetricUnit	**字符串**	Amazon CW 指标单元。

标头	类型	描述
CamelAwsCwMetricNamespace	**字符串**	Amazon CW 指标命名空间。
CamelAwsCwMetricTimestamp	**Date**	Amazon CW 指标时间戳。
CamelAwsCwMetricDimensionName	**字符串**	**Camel 2.12 ：** Amazon CW 指标维度名称。
CamelAwsCwMetricDimensionValue	**字符串**	**Camel 2.12 ：** Amazon CW 指标维度值。
CamelAwsCwMetricDimensions	**Map<String, String>**	**Camel 2.12 ：** 维度名称和维度值的映射。

22.4.2. 高级 AmazonCloudWatch 配置

如果您需要对 **AmazonCloudWatch** 实例配置进行更多控制，您可以创建自己的实例并从 URI 引用它：

```
from("direct:start")
.to("aws-cw://namepsace?amazonCwClient=#client");
```

#client 指的是 Registry 中的 **AmazonCloudWatch**。

例如，如果您的 Camel 应用程序在防火墙后面运行：

```
AWSCredentials awsCredentials = new BasicAWSCredentials("myAccessKey", "mySecretKey");
```

```
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setProxyHost("http://myProxyHost");
clientConfiguration.setProxyPort(8080);

AmazonCloudWatch client = new AmazonCloudWatchClient(awsCredentials, clientConfiguration);

registry.bind("client", client);
```

22.5. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-aws</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.10 或更高版本)。

22.6. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- AWS 组件

第 23 章 AWS DYNAMODB COMPONENT

从 Camel 版本 2.10 开始提供

DynamoDB 组件支持从/到 Amazon 的 DynamoDB 服务存储和检索数据。

先决条件

您必须有一个有效的 Amazon Web Services 开发人员帐户，并签名以使用 Amazon DynamoDB。如需更多信息，请参阅 Amazon DynamoDB。

23.1. URI 格式

```
aws-ddb://domainName[?options]
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

23.2. URI 选项

AWS DynamoDB 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	AWS DDB 默认配置		DdbConfiguration
AccessKey (producer)	Amazon AWS 访问密钥		字符串
secretKey (producer)	Amazon AWS Secret 密钥		字符串
region (producer)	DDB 客户端需要工作的区域		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AWS DynamoDB 端点使用 URI 语法进行配置：

```
aws-ddb:tableName
```

使用以下路径和查询参数：

23.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
tableName	**需要** 当前有效的表的名称。		字符串

23.2.2. 查询参数(13 参数)：

Name	描述	默认值	类型
amazonDDBClient (producer)	将 AmazonDynamoDB 用作客户端		AmazonDynamoDB
consistentRead (producer)	决定在读取数据时是否应强制执行强一致性。	false	布尔值
keyAttributeName (producer)	创建表时的属性名称		字符串
keyAttributeType (producer)	创建表时的属性类型		字符串
operation (producer)	要执行的操作	PutItem	DdbOperations
proxyHost (producer)	在实例化 DDB 客户端时定义代理主机		字符串
proxyPort (producer)	在实例化 DDB 客户端时定义代理端口		整数
readCapacity (producer)	要从您的表中读取资源的置备吞吐量		Long
region (producer)	DDB 客户端需要工作的区域		字符串
writeCapacity (producer)	为向表写入资源而保留置备的吞吐量		Long
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
accessKey (security)	Amazon AWS 访问密钥		字符串
secretKey (security)	Amazon AWS Secret 密钥		字符串

23.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 18 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws-ddb.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-ddb.configuration.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-ddb.configuration.amazon-d-d-b-client	将 AmazonDynamoDB 用作客户端		AmazonDynamoDB
camel.component.aws-ddb.configuration.consistent-read	决定在读取数据时是否应强制执行强一致性。	false	布尔值
camel.component.aws-ddb.configuration.key-attribute-name	创建表时的属性名称		字符串
camel.component.aws-ddb.configuration.key-attribute-type	创建表时的属性类型		字符串
camel.component.aws-ddb.configuration.operation	要执行的操作		DdbOperations
camel.component.aws-ddb.configuration.proxy-host	在实例化 DDB 客户端时定义代理主机		字符串
camel.component.aws-ddb.configuration.proxy-port	在实例化 DDB 客户端时定义代理端口		整数

Name	描述	默认值	类型
camel.component.aws-ddb.configuration.read-capacity	要从您的表中读取资源的置备吞吐量		Long
camel.component.aws-ddb.configuration.region	DDB 客户端需要工作的区域		字符串
camel.component.aws-ddb.configuration.secret-key	Amazon AWS Secret 密钥		字符串
camel.component.aws-ddb.configuration.table-name	当前有效的表的名称。		字符串
camel.component.aws-ddb.configuration.write-capacity	为向表写入资源而保留置备的吞吐量		Long
camel.component.aws-ddb.enabled	启用 aws-ddb 组件	true	布尔值
camel.component.aws-ddb.region	DDB 客户端需要工作的区域		字符串
camel.component.aws-ddb.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.aws-ddb.secret-key	Amazon AWS Secret 密钥		字符串

所需的 DDB 组件选项

您必须在 Registry 或 accessKey 和 secretKey 中提供 amazonDDBClient，才能访问 Amazon 的 DynamoDB。

23.4. 使用方法

23.4.1. 由 DDB producer 评估的消息标头

标头	类型	描述
CamelAwsDdbBatchItems	**Map<String, KeysAndAttributes>**	表名称和对应项目的映射，由主密钥获取。
CamelAwsDdbTableName	**字符串**	此操作的表名称。
CamelAwsDdbKey	**键**	唯一标识表中的每个项目的主要键。**在 Camel 2.16.0 中，此标头的类型是 Map<String, AttributeValue> 而不是 Key**
CamelAwsDdbReturnValues	**字符串**	如果您要在修改前或之后获取属性 name-value 对(NONE, ALL_OLD, UPDATED_OLD, ALL_NEW, UPDATED_NEW)，则使用此参数。
CamelAwsDdbUpdateCondition	**Map<String, ExpectedAttributeValue>**	为条件修改指定属性。
CamelAwsDdbAttributeNames	**collection<String>**	如果没有指定属性名称，则返回所有属性。
CamelAwsDdbConsistentRead	**布尔值**	如果设置为 true，则会发出一致的读取，否则最终会使用一致性。

标头	类型	描述
CamelAwsDdbIndexName	**字符串**	如果设置将用作查询操作的 Secondary Index。
CamelAwsDdbItem	**Map<String, AttributeValue>**	项目的属性映射，必须包含定义项目的主要键值。
CamelAwsDdbExactCount	**布尔值**	如果设置为 true，Amazon DynamoDB 会返回与查询参数匹配的项目总数，而不是匹配的项目及其属性的列表。**从 Camel 2.16.0 中，此标头不再存在。**
CamelAwsDdbKeyConditions	**Map<String, Condition>**	**从 Camel 2.16.0。此标头指定查询的选择条件，并合并两个旧的标头 CamelAwsDdbHashKeyValue 和 CamelAwsDdbScanRangeKeyCondition**
CamelAwsDdbStartKey	**键**	项目的主密钥，以便从中继续之前的查询。
CamelAwsDdbHashKeyValue	**AttributeValue**	复合主密钥的 hash 组件的值。**从 Camel 2.16.0 中，此标头不再存在。**
CamelAwsDdbLimit	**整数**	要返回的项目的最大数量。
CamelAwsDdbScanRangeKeyCondition	**状况**	用于查询的属性值和比较运算符的容器。**从 Camel 2.16.0 中，此标头不再存在。**

标头	类型	描述
CamelAwsDdbScanIndexForward	**布尔值**	指定索引的正向和反向遍历。
CamelAwsDdbScanFilter	**Map<String, Condition>**	评估扫描结果，仅返回所需的值。
CamelAwsDdbUpdateValues	**Map<String, AttributeValueUpdate>**	将属性名称映射到更新的新值和操作。

23.4.2. 在 BatchGetItems 操作过程中设置的消息标头

标头	类型	描述
CamelAwsDdbBatchResponse	**Map<String,BatchResponse>**	表名称以及对应的项目属性。
CamelAwsDdbUnprocessedKeys	**Map<String,KeysAndAttributes>**	包含表映射及其对应的键，这些键没有使用当前响应进行处理。

23.4.3. 在 DeleteItem 操作过程中设置消息标头

标头	类型	描述

标头	类型	描述
CamelAwsDdbAttributes	**Map<String, AttributeValue>**	操作返回的属性列表。

23.4.4. 在 DeleteTable 操作过程中设置消息标头

标头	类型	描述
CamelAwsDdbProvisionedThroughput		
ProvisionedThroughputDescription		此表的 ProvisionedThroughput 属性的值
CamelAwsDdbCreationDate	**Date**	此表的创建 DateTime。
CamelAwsDdbTableItemCount	**Long**	此表的项目数。
CamelAwsDdbKeySchema	**KeySchema**	标识此表的主键的 KeySchema。**在 Camel 2.16.0 中，此标头的类型是 List<KeySchemaElement> 而不是 KeySchema**

标头	类型	描述
CamelAwsDdbTableName	**字符串**	表名称。
CamelAwsDdbTableSize	**Long**	表大小（以字节为单位）。
CamelAwsDdbTableStatus	**字符串**	表的状态：CREATING, UPDATING, DELETING, ACTIVE

23.4.5. 在 DescribeTable 操作过程中设置的消息标头

标头	类型	描述
CamelAwsDdbProvisionedThroughput	{{ProvisionedThroughputDescription}}	此表的 ProvisionedThroughput 属性的值
CamelAwsDdbCreationDate	**Date**	此表的创建 DateTime。
CamelAwsDdbTableItemCount	**Long**	此表的项目数。
CamelAwsDdbKeySchema	{{KeySchema}}	标识此表的主键的 KeySchema。**在 Camel 2.16.0 中，此标头的类型是 List<KeySchemaElement> 而不是 KeySchema**

标头	类型	描述
CamelAwsDdbTableName	**字符串**	表名称。
CamelAwsDdbTableSize	**Long**	表大小（以字节为单位）。
CamelAwsDdbTableStatus	**字符串**	表的状态：CREATING, UPDATING, DELETING, ACTIVE
CamelAwsDdbReadCapacity	**Long**	此表的 ReadCapacityUnits 属性。
CamelAwsDdbWriteCapacity	**Long**	此表的 WriteCapacityUnits 属性。

23.4.6. 在 GetItem 操作过程中设置的消息标头

标头	类型	描述
CamelAwsDdbAttributes	**Map<String, AttributeValue>**	操作返回的属性列表。

23.4.7. 在 PutItem 操作过程中设置的消息标头

标头	类型	描述
CamelAwsDdbAttributes	**Map<String, AttributeValue>**	操作返回的属性列表。

23.4.8. 在 Query 操作过程中设置消息标头

标头	类型	描述
CamelAwsDdbItems	**List<java.util.Map<String,AttributeValue>>**	操作返回的属性列表。
CamelAwsDdbLastEvaluatedKey	**键**	查询操作停止的项目的主要键，其中包含上一个结果集。
CamelAwsDdbConsumedCapacity	**å€Œ**	操作期间消耗的表的置备吞吐量数。
CamelAwsDdbCount	**整数**	响应中的项目数。

23.4.9. 扫描操作期间设置的消息标头

标头	类型	描述

标头	类型	描述
CamelAwsDdbItems	**List<java.util.Map<String,AttributeValue>>**	操作返回的属性列表。
CamelAwsDdbLastEvaluatedKey	**键**	查询操作停止的项目的主要键，其中包含上一个结果集。
CamelAwsDdbConsumedCapacity	**å€Œ**	操作期间消耗的表的置备吞吐量数。
CamelAwsDdbCount	**整数**	响应中的项目数。
CamelAwsDdbScannedCount	**整数**	应用任何过滤器前，完成扫描中的项目数。

23.4.10. 在 UpdateItem 操作过程中设置的消息标头

标头	类型	描述
CamelAwsDdbAttributes	**Map<String, AttributeValue>**	操作返回的属性列表。

23.4.11. 高级 AmazonDynamoDB 配置

如果您需要对 **AmazonDynamoDB** 实例配置进行更多控制，您可以创建自己的实例并从 URI 引用它：

-

```
from("direct:start")
.to("aws-ddb://domainName?amazonDDBClient=#client");
```

#client 指的是 Registry 中的 **AmazonDynamoDB**。

例如，如果您的 Camel 应用程序在防火墙后面运行：

```
AWSCredentials awsCredentials = new BasicAWSCredentials("myAccessKey", "mySecretKey");
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setProxyHost("http://myProxyHost");
clientConfiguration.setProxyPort(8080);

AmazonDynamoDB client = new AmazonDynamoDBClient(awsCredentials, clientConfiguration);

registry.bind("client", client);
```

23.5. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-aws</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.10 或更高版本)。

23.6. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- AWS 组件

第 24 章 AWS DYNAMODB STREAMS COMPONENT

从 Camel 版本 2.17 开始提供

DynamoDB Stream 组件支持从 Amazon DynamoDB Stream 服务接收消息。

先决条件

您必须有一个有效的 Amazon Web Services 开发人员帐户，并签名以使用 Amazon DynamoDB Streams。如需更多信息，请参阅 AWS DynamoDB

24.1. URI 格式

```
aws-ddbstream://table-name[?options]
```

在使用流前需要创建流。
您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

24.2. URI 选项

AWS DynamoDB Streams 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	AWS DDB 流默认配置		DdbStreamConfiguration
accessKey (consumer)	Amazon AWS 访问密钥		字符串
secretKey (consumer)	Amazon AWS Secret 密钥		字符串
region (consumer)	Amazon AWS 区域		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AWS DynamoDB Streams 端点使用 URI 语法进行配置：

```
aws-ddbstream:tableName
```

使用以下路径和查询参数：

24.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
tableName	dynamodb 表的 **必需** 名称		字符串

24.2.2. 查询参数(28 参数)：

Name	描述	默认值	类型
amazonDynamoDbStreams Client (consumer)	用于此端点的所有请求的 Amazon DynamoDB 客户端		AmazonDynamoDBStreams
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
iteratorType (consumer)	定义 DynaboDB 流中开始获取记录的位置。请注意，使用 TRIM_HORIZON 可能会导致流及时出现大量延迟。如果使用 AT,AFTER_SEQUENCE_NUMBER，则会提供一个 sequenceNumberProvider MUST。	LATEST	ShardIteratorType
maxResultsPerRequest (consumer)	每次轮询中将获取的最大记录数		int
proxyHost (consumer)	在实例化 DDBStreams 客户端时定义代理主机		字符串
proxyPort (consumer)	在实例化 DDBStreams 客户端时定义代理端口		整数
region (consumer)	DDBStreams 客户端需要工作的区域		字符串
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
sequenceNumberProvider (consumer)	当使用两个 ShardIteratorType.AT,AFTER_SEQUENCE_NUMBER iterator 类型时，序列号的供应商。可以是 registry 引用，也可以是一个字面序列号。		SequenceNumberProvider
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler

Name	描述	默认值	类型
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map

Name	描述	默认值	类型
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
accessKey (security)	Amazon AWS 访问密钥		字符串
secretKey (security)	Amazon AWS Secret 密钥		字符串

24.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 15 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws-ddbstream.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-ddbstream.configuration.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-ddbstream.configuration.amazon-dynamo-db-streams-client	用于此端点的所有请求的 Amazon DynamoDB 客户端		AmazonDynamoDBStreams
camel.component.aws-ddbstream.configuration.iterator-type	定义 DynaboDB 流中开始获取记录的位置。请注意，使用 TRIM_HORIZON 可能会导致流及时出现大量延迟。如果使用 AT,AFTER_SEQUENCE_NUMBER，则会提供一个 sequenceNumberProvider MUST。		ShardIteratorType

Name	描述	默认值	类型
camel.component.aws-ddbstream.configuration.max-results-per-request	每次轮询中将获取的最大记录数		整数
camel.component.aws-ddbstream.configuration.proxy-host	在实例化 DDBStreams 客户端时定义代理主机		字符串
camel.component.aws-ddbstream.configuration.proxy-port	在实例化 DDBStreams 客户端时定义代理端口		整数
camel.component.aws-ddbstream.configuration.region	DDBStreams 客户端需要工作的区域		字符串
camel.component.aws-ddbstream.configuration.secret-key	Amazon AWS Secret 密钥		字符串
camel.component.aws-ddbstream.configuration.sequence-number-provider	当使用两个 ShardIteratorType.AT,AFTER_SEQUENCE_NUMBER iterator 类型时，序列号的供应商。可以是 registry 引用，也可以是一个字面序列号。		SequenceNumberProvider
camel.component.aws-ddbstream.configuration.table-name	dynamodb 表的名称		字符串
camel.component.aws-ddbstream.enabled	启用 aws-ddbstream 组件	true	布尔值

Name	描述	默认值	类型
camel.component.aws-ddbstream.region	Amazon AWS 区域		字符串
camel.component.aws-ddbstream.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.aws-ddbstream.secret-key	Amazon AWS Secret 密钥		字符串

所需的 DynampDBStream 组件选项

您必须在 Registry 中提供 amazonDynamoDbStreamsClient，并配置了代理和相关凭证。

24.4. 序列号

您可以提供一个字面字符串作为序列号，或者在 registry 中提供 bean。使用 bean 的示例是将当前位置保存到更改源中，并在 Camel 启动时恢复它。

这是一个错误，它提供大于 describe-streams 结果中的最大序列号，因为这会导致 AWS 调用返回 HTTP 400。

24.5. BATCH CONSUMER

这个组件实现了 Batch Consumer。

这样，您可以让实例知道此批处理中存在多少个消息，而实例则让聚合器聚合此消息数量。

24.6. 使用方法

24.6.1. AmazonDynamoDBStreamsClient configuration

您需要创建一个 AmazonDynamoDBStreamsClient 实例，并将其绑定到 registry

```
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setProxyHost("http://myProxyHost");
clientConfiguration.setProxyPort(8080);

Region region = Region.getRegion(Regions.fromName(region));
region.createClient(AmazonDynamoDBStreamsClient.class, null, clientConfiguration);
// the 'null' here is the AWSCredentialsProvider which defaults to an instance of
```

```
DefaultAWSCredentialsProviderChain

registry.bind("kinesisClient", client);
```

24.6.2. 提供 AWS 凭证

建议使用 DefaultAWSCredentialsProviderChain 获取凭证，这是创建新 ClientConfiguration 实例时的默认设置，但在调用 createClient (...)时可以指定不同的 AWSCredentialsProvider。

24.7. 使用 DOWNTIME 共存

24.7.1. AWS DynamoDB Streams 在 24 小时内中断

消费者将从最后看到的序列号中恢复（由 CAMEL-9515实施），因此当中断没有包括 DynamoDB 本身的情况下，您应当快速收到大量事件。

24.7.2. AWS DynamoDB Streams 中断时间超过 24 小时

假设 AWS 只保留 24 小时的更改，无论相关的缓解方案是什么，您都会错过了更改事件。

24.8. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-aws</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.7 或更高版本)。

24.9. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- AWS Component
  +

第 25 章 AWS EC2 组件

从 Camel 版本 2.16 开始提供

EC2 组件支持 create、run、start、stop 和 terminate AWS EC2 实例。

先决条件

您必须有一个有效的 Amazon Web Services 开发人员帐户，并签名以使用 Amazon EC2。如需更多信息，请参阅 Amazon EC2。

25.1. URI 格式

```
aws-ec2://label[?options]
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

25.2. URI 选项

AWS EC2 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	AWS EC2 默认配置		EC2Configuration
region (producer)	EC2 客户端需要工作的区域		字符串
AccessKey (producer)	Amazon AWS 访问密钥		字符串
secretKey (producer)	Amazon AWS Secret 密钥		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AWS EC2 端点使用 URI 语法进行配置：

```
aws-ec2:label
```

使用以下路径和查询参数：

25.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
label	**所需的** 逻辑名称		字符串

25.2.2. 查询参数(8 参数)：

Name	描述	默认值	类型
AccessKey (producer)	Amazon AWS 访问密钥		字符串
amazonEc2Client (producer)	使用现有配置的 AmazonEC2Client 作为客户端		AmazonEC2Client
operation (producer)	需要执行的操作。**它可以是 createAndRunInstances, startInstances, stopInstances, terminateInstances, describeInstances, describeInstances, rebootInstances, monitorInstances, unmonitorInstances, unmonitorInstances, createTags 或 deleteTags**		EC2Operations
proxyHost (producer)	在实例化 EC2 客户端时定义代理主机		字符串
proxyPort (producer)	在实例化 EC2 客户端时定义代理端口		整数
region (producer)	EC2 客户端需要工作的区域		字符串
secretKey (producer)	Amazon AWS Secret 密钥		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

25.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 12 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws-ec2.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-ec2.configuration.access-key	Amazon AWS 访问密钥		字符串

Name	描述	默认值	类型
camel.component.aws-ec2.configuration.amazon-ec2-client	使用现有配置的 AmazonEC2Client 作为客户端		AmazonEC2Client
camel.component.aws-ec2.configuration.operation	要执行的操作。它可以是 createAndRunInstances, startInstances, stopInstances, terminateInstances, describeInstances, describeInstances, rebootInstances, monitorInstances, unmonitorInstances, unmonitorInstances, createTags 或 deleteTags		EC2Operations
camel.component.aws-ec2.configuration.proxy-host	在实例化 EC2 客户端时定义代理主机		字符串
camel.component.aws-ec2.configuration.proxy-port	在实例化 EC2 客户端时定义代理端口		整数
camel.component.aws-ec2.configuration.region	EC2 客户端需要工作的区域		字符串
camel.component.aws-ec2.configuration.secret-key	Amazon AWS Secret 密钥		字符串
camel.component.aws-ec2.enabled	启用 aws-ec2 组件	true	布尔值
camel.component.aws-ec2.region	EC2 客户端需要工作的区域		字符串
camel.component.aws-ec2.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.aws-ec2.secret-key	Amazon AWS Secret 密钥		字符串

所需的 EC2 组件选项

您必须在 Registry 或 accessKey 和 secretKey 中提供 amazonEc2Client，才能访问 Amazon EC2 服务。

25.4. 使用方法

25.4.1. 由 EC2 producer 评估的消息标头

标头	类型	描述
CamelAwsEC2ImageId	**字符串**	AWS 市场的镜像 ID
CamelAwsEC2InstanceType	com.amazonaws.services.ec2.model.InstanceType	我们要创建和运行的实例类型
CamelAwsEC2Operation	**字符串**	我们要执行的操作
CamelAwsEC2InstanceMinCount	**int**	要运行的实例数。
CamelAwsEC2InstanceMaxCount	**int**	您要运行的最大实例数。
CamelAwsEC2InstanceMonitoring	布尔值	定义是否希望监控正在运行的实例

标头	类型	描述
CamelAwsEC2InstanceEbsOptimized	**布尔值**	定义是否已创建实例是否为 EBS I/O 进行了优化。
CamelAwsEC2InstanceSecurityGroups	集合	与实例关联的安全组
CamelAwsEC2Instancesids	**集合**	用于执行 start、stop、describe 和 terminate 操作的实例 ID 集合。
CamelAwsEC2InstancesTags	**集合**	从 EC2 资源添加或删除的标签集合

依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-aws</artifactId>
    <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.16 或更高版本)。

25.5. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

- AWS 组件

第 26 章 AWS IAM 组件

从 Camel 版本 2.23 开始提供

KMS 组件支持 create、run、start、stop 和 terminate AWS IAM 实例。

先决条件

您必须有一个有效的 Amazon Web Services 开发人员帐户，并签名以使用 Amazon IAM。如需更多信息，请参阅 Amazon IAM。

26.1. URI 格式

```
aws-kms://label[?options]
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

26.2. URI 选项

AWS IAM 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	AWS IAM 默认配置		IAMConfiguration
AccessKey (producer)	Amazon AWS 访问密钥		字符串
secretKey (producer)	Amazon AWS Secret 密钥		字符串
region (producer)	IAM 客户端需要工作的区域		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AWS IAM 端点使用 URI 语法进行配置：

```
aws-iam:label
```

使用以下路径和查询参数：

26.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
label	**所需的** 逻辑名称		字符串

26.2.2. 查询参数(8 参数)：

Name	描述	默认值	类型
AccessKey (producer)	Amazon AWS 访问密钥		字符串
iamClient (producer)	使用现有配置的 AWS IAM 作为客户端		AmazonIdentityManagementClient
operation (producer)	所需的操作		IAMOperations
proxyHost (producer)	在实例化 KMS 客户端时定义代理主机		字符串
proxyPort (producer)	在实例化 KMS 客户端时定义代理端口		整数
region (producer)	KMS 客户端需要工作的区域		字符串
secretKey (producer)	Amazon AWS Secret 密钥		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

26.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 12 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws-iam.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-iam.configuration.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-iam.configuration.iam-client	使用现有配置的 AWS IAM 作为客户端		AmazonIdentityManagementClient

Name	描述	默认值	类型
camel.component.aws-iam.configuration.operation	要执行的操作		IAMOperations
camel.component.aws-iam.configuration.proxy-host	在实例化 KMS 客户端时定义代理主机		字符串
camel.component.aws-iam.configuration.proxy-port	在实例化 KMS 客户端时定义代理端口		整数
camel.component.aws-iam.configuration.region	KMS 客户端需要工作的区域		字符串
camel.component.aws-iam.configuration.secret-key	Amazon AWS Secret 密钥		字符串
camel.component.aws-iam.enabled	是否启用 aws-iam 组件的自动配置。这默认是启用的。		布尔值
camel.component.aws-iam.region	IAM 客户端需要工作的区域		字符串
camel.component.aws-iam.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.aws-iam.secret-key	Amazon AWS Secret 密钥		字符串

所需的 IAM 组件选项

您必须在 Registry 或 accessKey 和 secretKey 中提供 amazonKmsClient，才能访问 Amazon IAM 服务。

26.4. 使用方法

26.4.1. IAM producer 评估的消息标头

标头	类型	描述
CamelAwsIAMOperation	**字符串**	我们要执行的操作
CamelAwsIAMUsername	**字符串**	要管理的用户的用户名
CamelAwsIAMAccessKeyID	**字符串**	要管理的 accessKey
CamelAwsIAMAccessKeyStatus	**字符串**	您要设置的 AccessKey 状态，可能的值是 active 和 inactive

26.4.2. IAM Producer 操作

Camel-AWS IAM 组件在生成者端提供以下操作：

- listAccessKeys
- createUser
- deleteUser
- listUsers
- getUser
- createAccessKey
- deleteAccessKey
- updateAccessKey

依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
```

```
    <artifactId>camel-aws</artifactId>
    <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.16 或更高版本)。

26.5. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- AWS 组件

第 27 章 AWS KINESIS 组件

从 Camel 版本 2.17 开始提供

Kinesis 组件支持从 Amazon Kinesis 服务接收信息并将信息发送到 Amazon Kinesis 服务。

先决条件

您必须有一个有效的 Amazon Web Services 开发人员帐户，并使用 Amazon Kinesis 注册。如需更多信息，请参阅 AWS Kinesis

27.1. URI 格式

```
aws-kinesis://stream-name[?options]
```

在使用流前需要创建流。
您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

27.2. URI 选项

AWS Kinesis 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	AWS S3 默认配置		KinesisConfiguration
accessKey (common)	Amazon AWS 访问密钥		字符串
secretKey (common)	Amazon AWS Secret 密钥		字符串
region (common)	Amazon AWS 区域		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AWS Kinesis 端点使用 URI 语法进行配置：

```
aws-kinesis:streamName
```

使用以下路径和查询参数：

27.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
streamName	流 **必需** 的名称		字符串

27.2.2. 查询参数(30 参数)：

Name	描述	默认值	类型
amazonKinesisClient (common)	Amazon Kinesis 客户端用于此端点的所有请求		AmazonKinesis
proxyHost (common)	在实例化 DDBStreams 客户端时定义代理主机		字符串
proxyPort (common)	在实例化 DDBStreams 客户端时定义代理端口		整数
region (common)	Kinesis 客户端需要工作的区域		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
iteratorType (consumer)	定义在 Kinesis 流中开始获取记录的位置	TRIM_HORIZON	ShardIteratorType
maxResultsPerRequest (consumer)	每次轮询中将获取的最大记录数	1	int
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
sequenceNumber (consumer)	开始轮询的序列号。如果 iteratorType 设置为 AFTER_SEQUENCE_NUMBER 或 AT_SEQUENCE_NUMBER，则需要此项		字符串
shardClosed (consumer)	定义在分片关闭时的行为是什么。可能的值有 ignore, silent 和 fail。如果忽略消息，则会从头开始重启，如果静默，则不会记录日志，消费者将从头开始，在这种情况下，会引发 ReachedClosedStateException。	ignore	KinesisShardClosed StrategyEnum
shardId (consumer)	定义 Kinesis 流中要从哪些分片 ID 获取记录		字符串

Name	描述	默认值	类型
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler

Name	描述	默认值	类型
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
accessKey (security)	Amazon AWS 访问密钥		字符串
secretKey (security)	Amazon AWS Secret 密钥		字符串

27.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 17 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws-kinesis.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-kinesis.configuration.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-kinesis.configuration.amazon-kinesis-client	Amazon Kinesis 客户端用于此端点的所有请求		AmazonKinesis
camel.component.aws-kinesis.configuration.iterator-type	定义在 Kinesis 流中开始获取记录的位置		ShardIteratorType

Name	描述	默认值	类型
camel.component.aws-kinesis.configuration.max-results-per-request	每次轮询中将获取的最大记录数	1	整数
camel.component.aws-kinesis.configuration.proxy-host	在实例化 DDBStreams 客户端时定义代理主机		字符串
camel.component.aws-kinesis.configuration.proxy-port	在实例化 DDBStreams 客户端时定义代理端口		整数
camel.component.aws-kinesis.configuration.region	Kinesis 客户端需要工作的区域		字符串
camel.component.aws-kinesis.configuration.secret-key	Amazon AWS Secret 密钥		字符串
camel.component.aws-kinesis.configuration.sequence-number	开始轮询的序列号。如果 iteratorType 设置为 AFTER_SEQUENCE_NUMBER 或 AT_SEQUENCE_NUMBER，则需要此项		字符串
camel.component.aws-kinesis.configuration.shard-closed	定义在分片关闭时的行为是什么。可能的值有 ignore, silent 和 fail。如果忽略消息，则会从头开始重启，如果静默，则不会记录日志，消费者将从头开始，在这种情况下，会引发 ReachedClosedStateException。		KinesisShardClosed StrategyEnum
camel.component.aws-kinesis.configuration.shard-id	定义 Kinesis 流中要从哪些分片 ID 获取记录		字符串
camel.component.aws-kinesis.configuration.stream-name	流的名称		字符串

Name	描述	默认值	类型
camel.component.aws-kinesis.enabled	启用 aws-kinesis 组件	true	布尔值
camel.component.aws-kinesis.region	Amazon AWS 区域		字符串
camel.component.aws-kinesis.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.aws-kinesis.secret-key	Amazon AWS Secret 密钥		字符串

所需的 Kinesis 组件选项

您必须在 Registry 中提供 amazonKinesisClient，并配置了代理和相关凭证。

27.4. BATCH CONSUMER

这个组件实现了 Batch Consumer。

这样，您可以让实例知道此批处理中存在多少个消息，而实例则让聚合器聚合此消息数量。

27.5. 使用方法

27.5.1. 由 Kinesis consumer 设置的消息标头

标头	类型	描述
CamelAwsKinesisSequenceNumber	**字符串**	记录的序列号。这表示为一个字符串，因为它的大小不是由 API 定义。如果要将它用作数字类型，则使用

标头	类型	描述
CamelAwsKinesisApproximateArrivalTimestamp	**字符串**	为记录的 arrival 时间分配的时间 AWS。
CamelAwsKinesisPartitionKey	**字符串**	标识数据记录的流中分配给的分片。

27.5.2. AmazonKinesis 配置

您需要创建一个 AmazonKinesisClient 实例，并将其绑定到 registry

```
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setProxyHost("http://myProxyHost");
clientConfiguration.setProxyPort(8080);

Region region = Region.getRegion(Regions.fromName(region));
region.createClient(AmazonKinesisClient.class, null, clientConfiguration);
// the 'null' here is the AWSCredentialsProvider which defaults to an instance of
DefaultAWSCredentialsProviderChain

registry.bind("kinesisClient", client);
```

然后，您必须在 **amazonKinesisClient** URI 选项中引用 AmazonKinesisClient。

```
from("aws-kinesis://mykinesisstream?amazonKinesisClient=#kinesisClient")
  .to("log:out?showAll=true");
```

27.5.3. 提供 AWS 凭证

建议使用 DefaultAWSCredentialsProviderChain 获取凭证，这是创建新 ClientConfiguration 实例时的默认设置，但在调用 createClient (...)时可以指定不同的 AWSCredentialsProvider。

27.5.4. Kinesis producer 用来写入 Kinesis 的消息标头。生产者希望消息正文是 byte[]。

标头	类型	描述
CamelAwsKinesisPartitionKey	**字符串**	要传递给 Kinesis 来存储此记录的 PartitionKey。
CamelAwsKinesisSequenceNumber	**字符串**	可选参数以指示此记录的序列号。

27.5.5. 在记录成功存储时由 Kinesis producer 设置的消息标头

标头	类型	描述
CamelAwsKinesisSequenceNumber	**字符串**	记录的序列号，如 Response Syntax中定义的
CamelAwsKinesisShardId	**字符串**	存储记录的分片 ID

27.6. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-aws</artifactId>
    <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.17 或更高版本)。

27.7. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- AWS 组件

第 28 章 AWS KINESIS FIREHOSE 组件

从 Camel 版本 2.19 开始提供

Kinesis Firehose 组件支持向 Amazon Kinesis Firehose 服务发送消息。

先决条件

您必须有一个有效的 Amazon Web Services 开发人员帐户，并签名以使用 Amazon Kinesis Firehose。如需更多信息，请参阅 AWS Kinesis Firehose

28.1. URI 格式

```
aws-kinesis-firehose://delivery-stream-name[?options]
```

在使用流前需要创建流。
您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

28.2. URI 选项

AWS Kinesis Firehose 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	AWS Kinesis Firehose 默认配置		KinesisFirehose 配置
AccessKey (producer)	Amazon AWS 访问密钥		字符串
secretKey (producer)	Amazon AWS Secret 密钥		字符串
region (producer)	Amazon AWS 区域		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AWS Kinesis Firehose 端点使用 URI 语法进行配置：

```
aws-kinesis-firehose:streamName
```

使用以下路径和查询参数：

28.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
streamName	流 **必需** 的名称		字符串

28.2.2. 查询参数(7 参数)：

Name	描述	默认值	类型
amazonKinesisFirehoseClient (producer)	Amazon Kinesis Firehose 客户端用于此端点的所有请求		AmazonKinesisFirehose
proxyHost (producer)	在实例化 DDBStreams 客户端时定义代理主机		字符串
proxyPort (producer)	在实例化 DDBStreams 客户端时定义代理端口		整数
region (producer)	Kinesis 客户端需要工作的区域		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
accessKey (security)	Amazon AWS 访问密钥		字符串
secretKey (security)	Amazon AWS Secret 密钥		字符串

28.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 12 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws-kinesis-firehose.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-kinesis-firehose.configuration.access-key	Amazon AWS 访问密钥		字符串

Name	描述	默认值	类型
camel.component.aws-kinesis-firehose.configuration.amazon-kinesis-firehose-client	Amazon Kinesis Firehose 客户端用于此端点的所有请求		AmazonKinesisFirehose
camel.component.aws-kinesis-firehose.configuration.proxy-host	在实例化 DDBStreams 客户端时定义代理主机		字符串
camel.component.aws-kinesis-firehose.configuration.proxy-port	在实例化 DDBStreams 客户端时定义代理端口		整数
camel.component.aws-kinesis-firehose.configuration.region	Kinesis 客户端需要工作的区域		字符串
camel.component.aws-kinesis-firehose.configuration.secret-key	Amazon AWS Secret 密钥		字符串
camel.component.aws-kinesis-firehose.configuration.stream-name	流的名称		字符串
camel.component.aws-kinesis-firehose.enabled	启用 aws-kinesis-firehose 组件	true	布尔值
camel.component.aws-kinesis-firehose.region	Amazon AWS 区域		字符串
camel.component.aws-kinesis-firehose.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Name	描述	默认值	类型
camel.component.aws-kinesis-firehose.secret-key	Amazon AWS Secret 密钥		字符串

所需的 Kinesis Firehose 组件选项

您必须在 Registry 中提供 amazonKinesisClient，并配置了代理和相关凭证。

28.4. 使用方法

28.4.1. Amazon Kinesis Firehose 配置

您需要创建一个 AmazonKinesisClient 实例，并将其绑定到 registry

```
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setProxyHost("http://myProxyHost");
clientConfiguration.setProxyPort(8080);

Region region = Region.getRegion(Regions.fromName(region));
region.createClient(AmazonKinesisClient.class, null, clientConfiguration);
// the 'null' here is the AWSCredentialsProvider which defaults to an instance of
DefaultAWSCredentialsProviderChain

registry.bind("kinesisFirehoseClient", client);
```

然后，您必须在 **amazonKinesisFirehoseClient** URI 选项中引用 AmazonKinesisFirehoseClient URI 选项。

```
from("aws-kinesis-firehose://mykinesisdeliverystream?amazonKinesisFirehoseClient=#kinesisClient")
  .to("log:out?showAll=true");
```

28.4.2. 提供 AWS 凭证

建议使用 DefaultAWSCredentialsProviderChain 获取凭证，这是创建新 ClientConfiguration 实例时的默认设置，但在调用 createClient (...)时可以指定不同的 AWSCredentialsProvider。

28.4.3. 在记录成功存储时由 Kinesis producer 设置的消息标头

标头	类型	描述
CamelAwsKinesisFirehoseRecordId	**字符串**	记录 ID，如 Response 语法中定义的

28.5. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-aws</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.19 或更高版本)。

28.6. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- AWS 组件

第 29 章 AWS KMS 组件

从 Camel 版本 2.21 开始提供

KMS 组件支持创建、运行、启动、停止和终止 AWS KMS 实例。

先决条件

您必须有一个有效的 Amazon Web Services 开发人员帐户，并签名以使用 Amazon KMS。如需更多信息，请参阅 Amazon KMS。

29.1. URI 格式

```
aws-kms://label[?options]
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

29.2. URI 选项

AWS KMS 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	AWS KMS 默认配置		KMSConfiguration
AccessKey (producer)	Amazon AWS 访问密钥		字符串
secretKey (producer)	Amazon AWS Secret 密钥		字符串
region (producer)	KMS 客户端需要工作的区域		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AWS KMS 端点使用 URI 语法进行配置：

```
aws-kms:label
```

使用以下路径和查询参数：

29.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
label	**所需的** 逻辑名称		字符串

29.2.2. 查询参数(8 参数)：

Name	描述	默认值	类型
AccessKey (producer)	Amazon AWS 访问密钥		字符串
kmsClient (producer)	使用现有配置的 AWS KMS 作为客户端		AWSKMS
operation (producer)	所需的操作		KMSOperations
proxyHost (producer)	在实例化 KMS 客户端时定义代理主机		字符串
proxyPort (producer)	在实例化 KMS 客户端时定义代理端口		整数
region (producer)	KMS 客户端需要工作的区域		字符串
secretKey (producer)	Amazon AWS Secret 密钥		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

29.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 12 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws-kms.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-kms.configuration.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-kms.configuration.kms-client	使用现有配置的 AWS KMS 作为客户端		AWSKMS

Name	描述	默认值	类型
camel.component.aws-kms.configuration.operation	要执行的操作		KMSOperations
camel.component.aws-kms.configuration.proxy-host	在实例化 KMS 客户端时定义代理主机		字符串
camel.component.aws-kms.configuration.proxy-port	在实例化 KMS 客户端时定义代理端口		整数
camel.component.aws-kms.configuration.region	KMS 客户端需要工作的区域		字符串
camel.component.aws-kms.configuration.secret-key	Amazon AWS Secret 密钥		字符串
camel.component.aws-kms.enabled	是否启用 aws-kms 组件的自动配置。这默认是启用的。		布尔值
camel.component.aws-kms.region	KMS 客户端需要工作的区域		字符串
camel.component.aws-kms.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.aws-kms.secret-key	Amazon AWS Secret 密钥		字符串

所需的 KMS 组件选项

您必须在 Registry 或 accessKey 和 secretKey 中提供 amazonKmsClient，才能访问 Amazon KMS 服务。

29.4. 使用方法

29.4.1. 由 MQ producer 评估的消息标头

标头	类型	描述
CamelAwsKMSLimit	**整数**	执行 listKeys 操作时要返回的键数量
CamelAwsKMSOperation	**字符串**	我们要执行的操作
CamelAwsKMSDescription	**字符串**	执行 createKey 操作时使用的密钥描述
CamelAwsKMSKeyId	**字符串**	密钥 Id

29.4.2. KMS Producer 操作

Camel-AWS KMS 组件在生成者端提供以下操作：

- listKeys
- createKey
- disableKey
- scheduleKeyDeletion
- describeKey
- enableKey

依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-aws</artifactId>
    <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.16 或更高版本)。

29.5. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- AWS 组件

第 30 章 AWS LAMBDA 组件

从 Camel 版本 2.20 开始提供

Lambda 组件支持 create, get, list, delete 和 invoke AWS Lambda 功能。

先决条件

您必须有一个有效的 Amazon Web Services 开发人员帐户，并使用 Amazon Lambda 注册。如需更多信息，请参阅 Amazon Lambda。

在创建 Lambda 功能时，您需要指定一个 IAM 角色，该角色至少附加了 AWSLambdaBasicExecuteRole 策略。

Warning

Lambda 是区域服务。与 S3 存储桶不同，在给定区域中创建的 Lambda 功能在其他区域不可用。

30.1. URI 格式

```
aws-lambda://functionName[?options]
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

30.2. URI 选项

AWS Lambda 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	AWS Lambda 默认配置		LambdaConfiguration
AccessKey (producer)	Amazon AWS 访问密钥		字符串
secretKey (producer)	Amazon AWS Secret 密钥		字符串
region (producer)	Amazon AWS 区域		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AWS Lambda 端点使用 URI 语法进行配置：

```
aws-lambda:function
```

使用以下路径和查询参数：

30.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
function	Lambda 函数**必需的**名称。		字符串

30.2.2. 查询参数(8 参数)：

Name	描述	默认值	类型
operation (producer)	需要执行的操作。**它可以是 listFunctions、getFunction、createFunction、deleteFunction 或 invokeFunction**		LambdaOperations
region (producer)	Amazon AWS 区域		字符串
awsLambdaClient (advanced)	使用现有配置的 AwsLambdaClient 作为客户端		AWSLambda
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
proxyHost (proxy)	在实例化 Lambda 客户端时定义代理主机		字符串
proxyPort (proxy)	在实例化 Lambda 客户端时定义代理端口		整数
accessKey (security)	Amazon AWS 访问密钥		字符串
secretKey (security)	Amazon AWS Secret 密钥		字符串

30.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 13 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws-lambda.access-key	Amazon AWS 访问密钥		字符串

Name	描述	默认值	类型
camel.component.aws-lambda.configuration.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-lambda.configuration.aws-lambda-client	使用现有配置的 AwsLambdaClient 作为客户端		AWSLambda
camel.component.aws-lambda.configuration.function	Lambda 函数的名称。		字符串
camel.component.aws-lambda.configuration.operation	要执行的操作。它可以是 listFunctions、getFunction、createFunction、deleteFunction 或 invokeFunction		LambdaOperations
camel.component.aws-lambda.configuration.proxy-host	在实例化 Lambda 客户端时定义代理主机		字符串
camel.component.aws-lambda.configuration.proxy-port	在实例化 Lambda 客户端时定义代理端口		整数
camel.component.aws-lambda.configuration.region	Amazon AWS 区域		字符串
camel.component.aws-lambda.configuration.secret-key	Amazon AWS Secret 密钥		字符串
camel.component.aws-lambda.enabled	是否启用 aws-lambda 组件的自动配置。这默认是启用的。		布尔值
camel.component.aws-lambda.region	Amazon AWS 区域		字符串

Name	描述	默认值	类型
camel.component.aws-lambda.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.aws-lambda.secret-key	Amazon AWS Secret 密钥		字符串

所需的 Lambda 组件选项

您必须在 Registry 或 accessKey 和 secretKey 中提供 awsLambdaClient，才能访问 Amazon Lambda 服务。

30.4. 使用方法

30.4.1. 由 Lambda producer 评估的消息标头

操作	标头	类型	描述	必填
All	**CamelAwsLambdaOperation**	**字符串**	我们要执行的操作。覆盖作为查询参数传递的操作	是

操作	标头	类型	描述	必填
createFunction	**CamelAwsLambdaS3Bucket**	**字符串**	存储包含部署软件包的 .zip 文件的 Amazon S3 bucket 名称。此存储桶必须位于您要创建 Lambda 功能的同一 AWS 区域。	否
createFunction	**CamelAwsLambdaS3Key**	**字符串**	要上传的 Amazon S3 对象（部署软件包）密钥名称。	否

操作	标头	类型	描述	必填
createFunction	**CamelAwsLambdaS3ObjectVersion**	字符串	要上传的 Amazon S3 对象（部署软件包）版本。	否
createFunction	**CamelAwsLambdaZipFile**	**字符串**	zip 文件的本地路径（部署软件包）。zip 文件的内容也可以放在消息正文中。	否

操作	标头	类型	描述	必填
createFunction	**CamelAwsLambdaRole**	**字符串**	当执行您的功能来访问任何其他 Amazon Web Services (AWS)资源时，Larmbda 假定 IAM 角色的 Amazon Resource Name (ARN)。	是
createFunction	**CamelAwsLambdaRuntime**	字符串	您上传的 Lambda 功能的运行时环境。(nodejs, nodejs4.3, nodejs6.10, java8, python2.7, python3.6, dotnetcore1.0, odejs4.3-edge)	是

操作	标头	类型	描述	必填
createFunction	**CamelAwsLambdaHandler**	**字符串**	Lambda 调用的代码中的功能，开始执行。对于 Node.js，它是您的函数中的 module-name.export 值。对于 Java，它可以是 package.class-name::handler 或 package.class-name。	是
createFunction	**CamelAwsLambdaDescription**	**字符串**	用户提供的描述。	否

操作	标头	类型	描述	必填
createFunction	**CamelAwsLambdaTargetArn**	**字符串**	包含 Amazon SQS 队列或 Amazon SNS 主题的目标 ARN (Amazon Resource Name) 的父对象。	否
createFunction	**CamelAwsLambdaMemorySize**	**整数**	为该功能配置的内存大小（以 MB 为单位）。必须是 64 MB 的倍数。	否

操作	标头	类型	描述	必填
createFunction	**CamelAwsLambdaKMSKeyArn**	**字符串**	用于加密功能环境变量的 KMS 密钥的 Amazon 资源名称(ARN)。如果没有提供，AWS Lambda 将使用默认服务密钥。	否
createFunction	**CamelAwsLambdaPublish**	**布尔值**	此布尔值参数可用于请求 AWS Lambda 来创建 Lambda 功能，并将版本作为原子操作发布。	否

操作	标头	类型	描述	必填
createFunction	**CamelAwsLambdaTimeout**	**整数**	Lambda 应该终止函数的功能执行时间。默认值为 3 秒。	否
createFunction	**CamelAwsLambdaTracingConfig**	**字符串**	您功能的追踪设置（活跃或传递）。	否

操作	标头	类型	描述	必填
createFunction	**CamelAwsLambdaEnvironmentVariables**	**Map<String, String>**	代表您的环境配置设置的键值对。	否

操作	标头	类型	描述	必填
createFunction	**CamelAwsLambdaEnvironmentTags**	**Map<String, String>**	分配给新功能的标签（键值对）列表。	否

操作	标头	类型	描述	必填
createFunction	**CamelAwsLambdaSecurityGroupIds**	**List<String>**	如果您的 Lambda 功能访问 VPC 中的资源，则 VPC 中的一个或多个安全组 ID 列表。	否
createFunction	**CamelAwsLambdaSubnetIds**	**List<String>**	如果您的 Lambda 功能访问 VPC 中的资源，则 VPC 中的一个或多个子网 ID 列表。	否

30.5. 可运行的操作列表

- listFunctions
- getFunction,
- createFunction
- deleteFunction
- invokeFunction
- updateFunction
- createEventSourceMapping
- deleteEventSourceMapping
- listEventSourceMapping

30.6. EXAMPLE

要完全了解组件的工作方式，您可以参阅此 集成测试

30.7. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-aws</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.16 或更高版本)。

30.8. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- AWS 组件

第 31 章 AWS MQ 组件

从 Camel 版本 2.21 开始提供

EC2 组件支持 create、run、start、stop 和 terminate AWS MQ 实例。

先决条件

您必须有一个有效的 Amazon Web Services 开发人员帐户，并签名以使用 Amazon MQ。如需更多信息，请参阅 Amazon MQ。

31.1. URI 格式

```
aws-mq://label[?options]
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

31.2. URI 选项

AWS MQ 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	AWS MQ 默认配置		MQConfiguration
AccessKey (producer)	Amazon AWS 访问密钥		字符串
secretKey (producer)	Amazon AWS Secret 密钥		字符串
region (producer)	MQ 客户端需要工作的区域		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AWS MQ 端点使用 URI 语法进行配置：

```
aws-mq:label
```

使用以下路径和查询参数：

31.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
label	**所需的** 逻辑名称		字符串

31.2.2. 查询参数(8 参数)：

Name	描述	默认值	类型
AccessKey (producer)	Amazon AWS 访问密钥		字符串
amazonMqClient (producer)	使用现有配置的 AmazonMQClient 作为客户端		AmazonMQ
operation (producer)	需要执行的操作。**它可以是 listBrokers,createBroker,deleteBroker**		MQOperations
proxyHost (producer)	在实例化 MQ 客户端时定义代理主机		字符串
proxyPort (producer)	在实例化 MQ 客户端时定义代理端口		整数
region (producer)	MQ 客户端需要工作的区域		字符串
secretKey (producer)	Amazon AWS Secret 密钥		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

31.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 12 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws-mq.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-mq.configuration.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-mq.configuration.amazon-mq-client	使用现有配置的 AmazonMQClient 作为客户端		AmazonMQ

Name	描述	默认值	类型
camel.component.aws-mq.configuration.operation	要执行的操作。它可以是 listBrokers,createBroker,deleteBroker		MQOperations
camel.component.aws-mq.configuration.proxy-host	在实例化 MQ 客户端时定义代理主机		字符串
camel.component.aws-mq.configuration.proxy-port	在实例化 MQ 客户端时定义代理端口		整数
camel.component.aws-mq.configuration.region	MQ 客户端需要工作的区域		字符串
camel.component.aws-mq.configuration.secret-key	Amazon AWS Secret 密钥		字符串
camel.component.aws-mq.enabled	是否启用 aws-mq 组件的自动配置。这默认是启用的。		布尔值
camel.component.aws-mq.region	MQ 客户端需要工作的区域		字符串
camel.component.aws-mq.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.aws-mq.secret-key	Amazon AWS Secret 密钥		字符串

所需的 EC2 组件选项

您必须在 Registry 或 accessKey 和 secretKey 中提供 amazonEc2Client，才能访问 Amazon EC2 服务。

31.4. 使用方法

31.4.1. 由 MQ producer 评估的消息标头

标头	类型	描述
CamelAwsMQMaxResults	**字符串**	必须从 listBrokers 操作检索的结果数
CamelAwsMQBrokerName	**字符串**	代理名称
CamelAwsMQOperation	**字符串**	我们要执行的操作
CamelAwsMQBrokerId	**字符串**	代理 ID
CamelAwsMQBrokerDeploymentMode	**字符串**	createBroker 操作中的代理部署模式
CamelAwsMQBrokerInstanceType	**字符串**	createBroker 操作中的 EC2 机器的实例类型
CamelAwsMQBrokerEngine	**字符串**	MQ 的代理引擎。默认为 ACTIVEMQ
CamelAwsMQBrokerEngineVersion	**字符串**	MQ 的 Broker Engine 版本。目前，您可以选择 5.15.6 和 5.15.0 of ACTIVEMQ

标头	类型	描述
CamelAwsMQBrokerUsers	**List<User>**	MQ 用户列表
CamelAwsMQBrokerPubliclyAccessible	**布尔值**	如果 MQ 实例必须公开可用或不可用。默认值为 false。

依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-aws</artifactId>
    <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.16 或更高版本)。

31.5. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- AWS 组件

第 32 章 AWS S3 STORAGE SERVICE 组件

从 Camel 版本 2.8 开始提供

S3 组件支持从/到 Amazon 的 S3 服务存储和检索 objetcs。

先决条件

您必须拥有有效的 Amazon Web Services 开发人员帐户，并有权限使用 Amazon S3。如需更多信息，请参阅 Amazon S3。

32.1. URI 格式

```
aws-s3://bucketNameOrArn[?options]
```

如果存储桶不存在，则会创建存储桶。
您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

例如，要从存储桶 **helloBucket** 读取文件 **hello.txt**，请使用以下片断：

```
from("aws-s3:helloBucket?accessKey=yourAccessKey&secretKey=yourSecretKey&prefix=hello.txt")
  .to("file:/var/downloaded");
```

32.2. URI 选项

AWS S3 Storage Service 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	AWS S3 默认配置		S3Configuration
accessKey (common)	Amazon AWS 访问密钥		字符串
secretKey (common)	Amazon AWS Secret 密钥		字符串
region (common)	bucket 所在的区域。这个选项在 com.amazonaws.services.s3.model.CreateBucketRequest 中使用。		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AWS S3 Storage Service 端点使用 URI 语法进行配置：

```
aws-s3:bucketNameOrArn
```

使用以下路径和查询参数：

32.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
bucketNameOrArn	**所需的** Bucket 名称或 ARN		字符串

32.2.2. 查询参数(50 参数)：

Name	描述	默认值	类型
amazonS3Client (common)	对链接 https://camel.apache.org/registry.htmlRegistry 中的 com.amazonaws.services.sqs.AmazonS3 的引用。		AmazonS3
pathStyleAccess (common)	S3 客户端是否应该使用路径风格访问	false	布尔值
policy (common)	此队列的策略在 com.amazonaws.services.s3.AmazonS3#setBucketPolicy() 方法中设置。		字符串
proxyHost (common)	在实例化 SQS 客户端时定义代理主机		字符串
proxyPort (common)	指定要在客户端定义中使用的代理端口。		整数
region (common)	S3 客户端需要工作的区域		字符串
useIAMCredentials (common)	设置 S3 客户端是否应该预期在 EC2 实例上加载凭据，或希望传递静态凭据。	false	布尔值
encryptionMaterials (common)	在 Symmetric/Asymmetric 客户端用法时要使用的加密资料		EncryptionMaterials
useEncryption (common)	定义是否使用加密	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值

Name	描述	默认值	类型
deleteAfterRead (consumer)	在检索后，从 S3 删除对象。只有在提交 Exchange 时，才会执行删除。如果进行回滚，则对象不会被删除。如果此选项为 false，则同一对象将被通过并再次在轮询上检索。因此，您需要使用路由中的 Idempotent Consumer EIP 来过滤重复项。您可以使用 S3Constants#BUCKET_NAME 和 S3Constants SerialKEY 标头进行过滤，或者只过滤 S3Constants SerialKEY 标头。	true	布尔值
fileName (consumer)	要从具有给定文件名的存储桶获取对象		字符串
includeBody (consumer)	如果为 true，则交换正文将设置为文件内容的流。如果为 false，标头将使用 S3 对象元数据设置，但正文为 null。这个选项与 autocloseBody 选项密切相关。如果将 includeBody 设为 true，并且 autocloseBody 设为 false，它将是关闭 S3Object 流的调用者。将 autocloseBody 设置为 true，将自动关闭 S3Object 流。	true	布尔值
maxConnections (consumer)	在 S3 客户端配置中设置 maxConnections 参数	60	int
maxMessagesPerPoll (consumer)	获取最大消息数，作为每次轮询轮询的限制。默认为无限限制，但使用 0 或负数来禁用它。	10	int
prefix (consumer)	com.amazonaws.services.s3.model.ListObjectsRequest 中使用的前缀，仅用于消费我们感兴趣的对象。		字符串
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
autocloseBody (consumer)	如果此选项为 true，并且 includeBody 为 true，则在交换完成时调用 S3Object.close （）方法。此选项与 includeBody 选项密切相关。如果将 includeBody 设为 true，并且 autocloseBody 设为 false，它将是关闭 S3Object 流的调用者。将 autocloseBody 设置为 true，将自动关闭 S3Object 流。	true	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern

Name	描述	默认值	类型
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
deleteAfterWrite (producer)	在 S3 文件上传后删除文件对象	false	布尔值
multiPartUpload (producer)	如果为 true，camel 将上传带有多部分格式的文件，由 partSize 选项决定部分大小	false	布尔值
operation (producer)	当用户不希望只进行上传时，要执行的操作		S3Operations
partSize (producer)	设置多部分上传中使用的 partSize，默认大小为 25M。	26214400	long
ServerSideEncryption (producer)	在使用 AWS 管理的密钥加密对象时设置服务器端加密算法。例如，使用 AES256。		字符串
storageClass (producer)	在 com.amazonaws.services.s3.model.PutObjectRequest 请求中设置的存储类。		字符串
awsKMSKeyId (producer)	定义在启用 KMS 时要使用的 KMS 密钥 ID		字符串
useAwsKMS (producer)	定义是否必须使用 KMS	false	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
accelerateModeEnabled (advanced)	定义是否启用 Accelerate Mode 为 true 或 false	false	布尔值
chunkedEncodingDisabled (advanced)	定义是否禁用了 Chunked Encoding 为 true 或 false	false	布尔值
dualstackEnabled （高级）	定义 Dualstack enabled 为 true 或 false	false	布尔值
forceGlobalBucketAccess Enabled (advanced)	定义是否强制全局 Bucket Access 启用为 true 或 false	false	布尔值

Name	描述	默认值	类型
payloadSigningEnabled (advanced)	定义是否启用了 Payload Signing 为 true 或 false	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit

Name	描述	默认值	类型
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
accessKey (security)	Amazon AWS 访问密钥		字符串
secretKey (security)	Amazon AWS Secret 密钥		字符串

32.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 34 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws-s3.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-s3.configuration.accelerate-mode-enabled	定义是否启用 Accelerate Mode 为 true 或 false	false	布尔值
camel.component.aws-s3.configuration.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-s3.configuration.amazon-s3-client	对链接 https://camel.apache.org/registry.htmlRegistry 中的 com.amazonaws.services.sqs.AmazonS3 的引用。		AmazonS3
camel.component.aws-s3.configuration.autoclose-body	如果此选项为 true，并且 includeBody 为 true，则在交换完成时调用 S3Object.close （）方法。此选项与 includeBody 选项密切相关。如果将 includeBody 设为 true，并且 autocloseBody 设为 false，它将是关闭 S3Object 流的调用者。将 autocloseBody 设置为 true，将自动关闭 S3Object 流。	true	布尔值
camel.component.aws-s3.configuration.aws-k-m-s-key-id	定义在启用 KMS 时要使用的 KMS 密钥 ID		字符串

Name	描述	默认值	类型
camel.component.aws-s3.configuration.bucket-name	bucket 的名称。如果存储桶不存在，则会创建存储桶。		字符串
camel.component.aws-s3.configuration.chunked-encoding-disabled	定义是否禁用了 Chunked Encoding 为 true 或 false	false	布尔值
camel.component.aws-s3.configuration.delete-after-read	在检索后，从 S3 删除对象。只有在提交 Exchange 时，才会执行删除。如果进行回滚，则对象不会被删除。如果此选项为 false，则同一对象将被通过并再次在轮询上检索。因此，您需要使用路由中的 Idempotent Consumer EIP 来过滤重复项。您可以使用 S3Constants#BUCKET_NAME 和 S3Constants SerialKEY 标头进行过滤，或者只过滤 S3Constants SerialKEY 标头。	true	布尔值
camel.component.aws-s3.configuration.delete-after-write	在 S3 文件上传后删除文件对象	false	布尔值
camel.component.aws-s3.configuration.dualstack-enabled	定义 Dualstack enabled 为 true 或 false	false	布尔值
camel.component.aws-s3.configuration.encryption-materials	在 Symmetric/Asymmetric 客户端用法时要使用的加密资料		EncryptionMaterials
camel.component.aws-s3.configuration.file-name	要从具有给定文件名的存储桶获取对象		字符串
camel.component.aws-s3.configuration.force-global-bucket-access-enabled	定义是否强制全局 Bucket Access 启用为 true 或 false	false	布尔值

Name	描述	默认值	类型
camel.component.aws-s3.configuration.include-body	如果为 true，则交换正文将设置为文件内容的流。如果为 false，标头将使用 S3 对象元数据设置，但正文为 null。这个选项与 autocloseBody 选项密切相关。如果将 includeBody 设为 true，并且 autocloseBody 设为 false，它将是关闭 S3Object 流的调用者。将 autocloseBody 设置为 true，将自动关闭 S3Object 流。	true	布尔值
camel.component.aws-s3.configuration.multi-part-upload	如果为 true，camel 将上传带有多部分格式的文件，由 partSize 选项决定部分大小	false	布尔值
camel.component.aws-s3.configuration.operation	当用户不希望只进行上传时，要执行的操作		S3Operations
camel.component.aws-s3.configuration.part-size	设置多部分上传中使用的 partSize，默认大小为 25M。	26214400	Long
camel.component.aws-s3.configuration.path-style-access	S3 客户端是否应该使用路径风格访问	false	布尔值
camel.component.aws-s3.configuration.payload-signing-enabled	定义是否启用了 Payload Signing 为 true 或 false	false	布尔值
camel.component.aws-s3.configuration.policy	此队列的策略在 com.amazonaws.services.s3.AmazonS3#setBucketPolicy() 方法中设置。		字符串
camel.component.aws-s3.configuration.prefix	com.amazonaws.services.s3.model.ListObjectsRequest 中使用的前缀，仅用于消费我们感兴趣的对象。		字符串

Name	描述	默认值	类型
camel.component.aws-s3.configuration.proxy-host	在实例化 SQS 客户端时定义代理主机		字符串
camel.component.aws-s3.configuration.proxy-port	指定要在客户端定义中使用的代理端口。		整数
camel.component.aws-s3.configuration.region	S3 客户端需要工作的区域		字符串
camel.component.aws-s3.configuration.secret-key	Amazon AWS Secret 密钥		字符串
camel.component.aws-s3.configuration.server-side-encryption	在使用 AWS 管理的密钥加密对象时设置服务器端加密算法。例如，使用 AES256。		字符串
camel.component.aws-s3.configuration.storage-class	在 com.amazonaws.services.s3.model.PutObjectRequest 请求中设置的存储类。		字符串
camel.component.aws-s3.configuration.use-aws-k-m-s	定义是否必须使用 KMS	false	布尔值
camel.component.aws-s3.configuration.use-encryption	定义是否使用加密	false	布尔值
camel.component.aws-s3.enabled	启用 aws-s3 组件	true	布尔值
camel.component.aws-s3.region	bucket 所在的区域。这个选项在 com.amazonaws.services.s3.model.CreateBucketRequest 中使用。		字符串

Name	描述	默认值	类型
camel.component.aws-s3.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.aws-s3.secret-key	Amazon AWS Secret 密钥		字符串

所需的 S3 组件选项

您必须在 Registry 或 accessKey 和 secretKey 中提供 amazonS3Client，才能访问 Amazon 的 S3。

32.4. BATCH CONSUMER

这个组件实现了 Batch Consumer。

这样，您可以让实例知道此批处理中存在多少个消息，而实例则让聚合器聚合此消息数量。

32.5. 使用方法

32.5.1. S3 producer 评估的消息标头

标头	类型	描述
CamelAwsS3BucketName	**字符串**	此对象的 bucket 名称将存储或用于当前操作
CamelAwsS3BucketDestinationName	**字符串**	Camel 2.18: 用于当前操作的存储桶目标名称
CamelAwsS3ContentLength	**Long**	此对象的内容长度。

标头	类型	描述
CamelAwsS3ContentType	**字符串**	此对象的内容类型。
CamelAwsS3ContentControl	**字符串**	Camel 2.8.2： 此对象的内容控制。
CamelAwsS3ContentDisposition	**字符串**	Camel 2.8.2： 此对象的内容分布。
CamelAwsS3ContentEncoding	**字符串**	Camel 2.8.2： 此对象的内容编码。
CamelAwsS3ContentMD5	**字符串**	Camel 2.8.2： 此对象的 md5 checksum。
CamelAwsS3DestinationKey	**字符串**	Camel 2.18: 用于当前操作的 Destination 键
CamelAwsS3Key	**字符串**	此对象将存储或用于当前操作的密钥
CamelAwsS3LastModified	**java.util.Date**	Camel 2.8.2： 此对象的最后修改的时间戳。

标头	类型	描述
CamelAwsS3Operation	**字符串**	Camel 2.18 ：要执行的操作。允许的值有 copyObject, deleteObject, listBuckets, deleteBucket, downloadLink, listObjects
CamelAwsS3StorageClass	**字符串**	Camel 2.8.4 ： 此对象的存储类。
CamelAwsS3CannedAcl	**字符串**	Camel 2.11.0： 应用于对象的 canned acl。有关允许的值，请参阅 **com.amazonaws.services.s3.model.CannedAccessControlList**。
CamelAwsS3Acl	**com.amazonaws.services.s3.model.AccessControlList**	Camel 2.11.0： 一个精心构建的 Amazon S3 Access Control List 对象。请参阅 **com.amazonaws.services.s3.model.AccessControlList** 以了解更多详细信息
CamelAwsS3Headers	**Map<String,String>**	Camel 2.15.0: 支持获取或设置自定义 objectMetadata 标头。
CamelAwsS3ServerSideEncryption	字符串	Camel 2.16 ： 在使用 AWS 管理的密钥加密对象时设置服务器端加密算法。例如，使用 AES256。
CamelAwsS3VersionId	**字符串**	要存储或从当前操作返回的对象的版本 Id

32.5.2. S3 producer 设置的消息标头

标头	类型	描述
CamelAwsS3ETag	**字符串**	新上传对象的 ETag 值。
CamelAwsS3VersionId	**字符串**	新上传对象的**可选**版本 ID。
CamelAwsS3DownloadLinkExpiration	**字符串**	URL 下载链接的过期(millis)。链接将存储在 **CamelAwsS3DownloadLink** 响应标头中。

32.5.3. S3 使用者设置的消息标头

标头	类型	描述
CamelAwsS3Key	**字符串**	存储此对象的密钥。
CamelAwsS3BucketName	**字符串**	包含此对象的存储桶的名称。
CamelAwsS3ETag	**字符串**	根据 RFC 1864，对相关对象的十六进制编码的 128 位 MD5 摘要。此数据用作完整性检查，以验证调用者收到的数据是否与 Amazon S3 发送的数据相同。
CamelAwsS3LastModified	**Date**	Last-Modified 标头的值，指示 Amazon S3 最后记录对关联对象的修改的日期和时间。
CamelAwsS3VersionId	**字符串**	关联的 Amazon S3 对象的版本 ID （如果可用）。只有当对象上传到启用了对象版本控制的 Amazon S3 存储桶时，才会将版本 ID 分配给对象。

标头	类型	描述
CamelAwsS3ContentType	**字符串**	Content-Type HTTP 标头，它表示存储在关联对象中的内容类型。此标头的值是标准 MIME 类型。
CamelAwsS3ContentMD5	**字符串**	根据 RFC 1864，使用 base64 编码的相关对象(content - 不包括标头)的 base64 编码的 128 位 MD5 摘要。此数据用作消息完整性检查，以验证 Amazon S3 收到的数据是否与调用者发送的数据相同。
CamelAwsS3ContentLength	**Long**	Content-Length HTTP 标头表示关联对象的大小（以字节为单位）。
CamelAwsS3ContentEncoding	**字符串**	**可选的** Content-Encoding HTTP 标头指定将什么内容编码应用到对象，必须应用哪些解码机制来获取 Content-Type 字段引用的 media-type。
CamelAwsS3ContentDisposition	**字符串**	**可选的** Content-Disposition HTTP 标头，它指定要保存的对象的建议文件名等。
CamelAwsS3ContentControl	**字符串**	**可选的** Cache-Control HTTP 标头，允许用户在 HTTP 请求/恢复链中指定缓存行为。
CamelAwsS3ServerSideEncryption	字符串	**Camel 2.16：** 使用 AWS 管理的密钥加密对象时的服务器端加密算法。

32.5.4. S3 Producer 操作

Camel-AWS s3 组件在生成者端提供以下操作：

- copyObject
- deleteObject
- listBuckets
- deleteBucket
- downloadLink
- listObjects

32.5.5. 高级 AmazonS3 配置

如果您的 Camel 应用程序在防火墙后面运行，或者您需要对 **AmazonS3** 实例配置拥有更多控制，您可以创建自己的实例：

```
AWSCredentials awsCredentials = new BasicAWSCredentials("myAccessKey", "mySecretKey");

ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setProxyHost("http://myProxyHost");
clientConfiguration.setProxyPort(8080);

AmazonS3 client = new AmazonS3Client(awsCredentials, clientConfiguration);

registry.bind("client", client);
```

并在 Camel aws-s3 组件配置中引用它：

```
from("aws-s3://MyBucket?amazonS3Client=#client&delay=5000&maxMessagesPerPoll=5")
.to("mock:result");
```

32.5.6. 将 KMS 与 S3 组件一起使用

要使用 AWS KMS 加密/解密数据，您可以使用 2.21.x 中引入的选项，如下例所示

```
from("file:tmp/test?fileName=test.txt")
    .setHeader(S3Constants.KEY, constant("testFile"))
    .to("aws-s3://mybucket?amazonS3Client=#client&useAwsKMS=true&awsKMSKeyId=3f0637ad-296a-3dfe-a796-e60654fb128c");
```

这样，您将要求 S3 使用 KMS 密钥 3f0637ad-296a-3dfe-a796-e60654fb128c 来加密文件 test.txt。当您要求下载该文件时，将在下载前直接进行解密。

32.5.7. 使用带有 s3 组件的 "useIAMCredentials"

要使用 AWS IAM 凭证，您必须首先验证启动 Camel 应用程序的 EC2 是否关联有与它关联的 IAM 角色，其中包含用于有效运行的适当策略。请记住，这个功能应只在远程实例上设置为 "true"。为了进一步说明，您仍然必须在本地使用静态凭证，因为 IAM 是一个 AWS 特定组件，但 AWS 环境现在应该更易于管理。在实现并理解后，您可以将 AWS 环境的查询参数 "useIAMCredentials" 设置为 "true"!要根据本地和远程环境有效切换它，您可以考虑使用系统环境变量启用此查询参数。例如，当名为 "isRemote" 的系统环境变量被设置为 true 时，您的代码可以将 "useIAMCredentials" 查询参数设置为 "true" （这是执行此操

作的许多其他方法，这应该作为简单示例）。虽然它不完全不需要静态凭证，但在 AWS 环境中使用 IAM 凭证可以无需刷新远程环境，并添加主要的安全提升(IAM 凭证会每 6 小时刷新一次，并在更新策略时更新)。这是 AWS 推荐的管理凭证的方法，因此应尽可能使用。

32.6. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-aws</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.8 或更高版本)。

32.7. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- AWS 组件

第 33 章 AWS SIMPLEDB 组件

从 Camel 版本 2.9 开始提供

sdb 组件支持从/到 Amazon 的 SDB 服务存储和检索数据。

先决条件

您必须有一个有效的 Amazon Web Services 开发人员帐户，并签名以使用 Amazon SDB。如需更多信息，请参阅 Amazon SDB。

33.1. URI 格式

```
aws-sdb://domainName[?options]
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

33.2. URI 选项

AWS SimpleDB 组件没有选项。

AWS SimpleDB 端点使用 URI 语法进行配置：

```
aws-sdb:domainName
```

使用以下路径和查询参数：

33.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
domainName	**必需** 当前有效的域的名称。		字符串

33.2.2. 查询参数(10 parameters):

Name	描述	默认值	类型
AccessKey (producer)	Amazon AWS 访问密钥		字符串
amazonSDBClient (producer)	将 AmazonSimpleDB 用作客户端		AmazonSimpleDB
consistentRead (producer)	决定在读取数据时是否应强制执行强一致性。	false	布尔值
maxNumberOfDomains (producer)	您要返回的最大域名数。范围为 1 到 100。		整数

Name	描述	默认值	类型
operation (producer)	要执行的操作	PutAttributes	SdbOperations
proxyHost (producer)	在实例化 SDB 客户端时定义代理主机		字符串
proxyPort (producer)	在实例化 SDB 客户端时定义代理端口		整数
region (producer)	SDB 客户端需要工作的区域		字符串
secretKey (producer)	Amazon AWS Secret 密钥		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

33.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws-sdb.enabled	启用 aws-sdb 组件	true	布尔值
camel.component.aws-sdb.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

所需的 SDB 组件选项

您必须在 Registry 或 accessKey 和 secretKey 中提供 amazonSDBClient，才能访问 Amazon 的 SDB。

33.4. 使用方法

33.4.1. 由 SDB producer 评估的消息标头

标头	类型	描述
CamelAwsSdbAttributes	**collection<Attribute>**	要执行操作的属性列表。
CamelAwsSdbAttributeNames	**collection<String>**	要检索的属性的名称。
CamelAwsSdbConsistentRead	**布尔值**	决定在读取数据时是否应强制执行强一致性。
CamelAwsSdbDeletableItems	**collection<DeletableItem>**	要在批处理中执行 delete 操作的项目列表。
CamelAwsSdbDomainName	**字符串**	当前有效的域的名称。
CamelAwsSdbItemName	**字符串**	此项目的唯一键
CamelAwsSdbMaxNumberOfDomains	**整数**	您要返回的最大域名数。范围为 1 * 到 100。
CamelAwsSdbNextToken	**字符串**	指定在何处启动域/item 名称列表的字符串。

标头	类型	描述
CamelAwsSdbOperation	**字符串**	覆盖 URI 选项中的操作。
CamelAwsSdbReplaceableAttributes	**collection<ReplaceableAttribute>**	要放入 Item 的属性列表。
CamelAwsSdbReplaceableItems	**collection<ReplaceableItem>**	要放入域中的项目列表。
CamelAwsSdbSelectExpression	**字符串**	用于查询域的表达式。
CamelAwsSdbUpdateCondition	**UpdateCondition**	如果指定，则更新条件决定是否更新/删除指定的属性。

33.4.2. 在 DomainMetadata 操作过程中设置的消息标头

标头	类型	描述
CamelAwsSdbTimestamp	**整数**	在 Epoch (UNIX)秒计算元数据时的数据和时间。

标头	类型	描述
CamelAwsSdbItemCount	**整数**	域中所有项目的数量。
CamelAwsSdbAttributeNameCount	**整数**	域中唯一属性名称的数量。
CamelAwsSdbAttributeValueCount	**整数**	域中所有属性名称/值对的数量。
CamelAwsSdbAttributeNameSize	**Long**	域中所有唯一属性名称的总大小，以字节为单位。
CamelAwsSdbAttributeValueSize	**Long**	域中所有属性值的总大小，以字节为单位。
CamelAwsSdbItemNameSize	**Long**	域中所有项目名称的总大小，以字节为单位。

33.4.3. 在 GetAttributes 操作过程中设置的消息标头

标头	类型	描述
CamelAwsSdbAttributes	**List<Attribute>**	操作返回的属性列表。

33.4.4. 在 ListDomains 操作过程中设置的消息标头

标头	类型	描述
CamelAwsSdbDomainNames	**List<String>**	与表达式匹配的域名列表。
CamelAwsSdbNextToken	**字符串**	不透明令牌表示存在超过指定的 MaxNumberOfDomains 的域。

33.4.5. 在 Select operation 过程中设置的消息标头

标头	类型	描述
CamelAwsSdbItems	**List<Item>**	与所选表达式匹配的项目列表。
CamelAwsSdbNextToken	**字符串**	不透明令牌表示匹配的项目数量超过 MaxNumberOfItems，响应大小超过 1MB，或执行时间超过 5 秒。

33.4.6. 高级 AmazonSimpleDB 配置

如果您需要对 **AmazonSimpleDB** 实例配置进行更多控制，您可以创建自己的实例并从 URI 引用它：

```
from("direct:start")
.to("aws-sdb://domainName?amazonSDBClient=#client");
```

#client 指的是 Registry 中的 **AmazonSimpleDB**。

例如，如果您的 Camel 应用程序在防火墙后面运行：

```
AWSCredentials awsCredentials = new BasicAWSCredentials("myAccessKey", "mySecretKey");
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setProxyHost("http://myProxyHost");
clientConfiguration.setProxyPort(8080);

AmazonSimpleDB client = new AmazonSimpleDBClient(awsCredentials, clientConfiguration);

registry.bind("client", client);
```

33.5. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-aws</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.8.4 或更高版本)。

33.6. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- AWS 组件

第 34 章 AWS SIMPLE EMAIL SERVICE 组件

从 Camel 版本 2.9 开始提供

ses 组件支持通过 Amazon 的 SES 服务发送电子邮件。

先决条件

您必须有一个有效的 Amazon Web Services 开发人员帐户，并签名以使用 Amazon SES。如需更多信息，请参阅 Amazon SES。

34.1. URI 格式

```
aws-ses://from[?options]
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

34.2. URI 选项

AWS Simple Email Service 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	AWS SES 默认配置		SesConfiguration
AccessKey (producer)	Amazon AWS 访问密钥		字符串
secretKey (producer)	Amazon AWS Secret 密钥		字符串
region (producer)	SES 客户端需要工作的区域		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AWS Simple Email Service 端点使用 URI 语法进行配置：

```
aws-ses:from
```

使用以下路径和查询参数：

34.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
from	**必需** 发送者的电子邮件地址。		字符串

34.2.2. 查询参数(11 参数)：

Name	描述	默认值	类型
amazonSESClient (producer)	使用 AmazonSimpleEmailService 作为客户端		AmazonSimpleEmail Service
proxyHost (producer)	在实例化 SES 客户端时定义代理主机		字符串
proxyPort (producer)	在实例化 SES 客户端时定义代理端口		整数
region (producer)	SES 客户端需要工作的区域		字符串
replyToAddresses (producer)	消息回复电子邮件地址列表，使用 'CamelAwsSesReplyToAddresses' 标头覆盖它。		list
returnPath (producer)	要转发退回通知的电子邮件地址，使用 'CamelAwsSesReturnPath' 标头覆盖它。		字符串
subject (producer)	如果邮件标头 'CamelAwsSesSubject' 不存在，则使用主题。		字符串
to (producer)	目标电子邮件地址列表。可以使用 'CamelAwsSesTo' 标头覆盖。		list
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
accessKey (security)	Amazon AWS 访问密钥		字符串
secretKey (security)	Amazon AWS Secret 密钥		字符串

34.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 16 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws-ses.access-key	Amazon AWS 访问密钥		字符串

Name	描述	默认值	类型
camel.component.aws-ses.configuration.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-ses.configuration.amazon-s-e-s-client	使用 AmazonSimpleEmailService 作为客户端		AmazonSimpleEmail Service
camel.component.aws-ses.configuration.from	发件人电子邮件地址。		字符串
camel.component.aws-ses.configuration.proxy-host	在实例化 SES 客户端时定义代理主机		字符串
camel.component.aws-ses.configuration.proxy-port	在实例化 SES 客户端时定义代理端口		整数
camel.component.aws-ses.configuration.region	SES 客户端需要工作的区域		字符串
camel.component.aws-ses.configuration.reply-to-addresses	消息回复电子邮件地址列表，使用 'CamelAwsSesReplyToAddresses' 标头覆盖它。		list
camel.component.aws-ses.configuration.return-path	要转发退回通知的电子邮件地址，使用 'CamelAwsSesReturnPath' 标头覆盖它。		字符串
camel.component.aws-ses.configuration.secret-key	Amazon AWS Secret 密钥		字符串

Name	描述	默认值	类型
camel.component.aws-ses.configuration.subject	如果邮件标头 'CamelAwsSesSubject' 不存在，则使用主题。		字符串
camel.component.aws-ses.configuration.to	目标电子邮件地址列表。可以使用 'CamelAwsSesTo' 标头覆盖。		list
camel.component.aws-ses.enabled	启用 aws-ses 组件	true	布尔值
camel.component.aws-ses.region	SES 客户端需要工作的区域		字符串
camel.component.aws-ses.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.aws-ses.secret-key	Amazon AWS Secret 密钥		字符串

所需的 SES 组件选项

您必须在 Registry 或 accessKey 和 secretKey 中提供 amazonSESClient，才能访问 Amazon 的 SES。

34.4. 使用方法

34.4.1. SES producer 评估的消息标头

标头	类型	描述
CamelAwsSesFrom	字符串	发件人电子邮件地址。
CamelAwsSesTo	List<String>	此电子邮件的目的地。

标头	类型	描述
CamelAwsSesSubject	**字符串**	消息主题。
CamelAwsSesReplyToAddresses	**List<String>**	邮件的回复电子邮件地址。
CamelAwsSesReturnPath	**字符串**	要转发退回通知的电子邮件地址。
CamelAwsSesHtmlEmail	**布尔值**	从 Camel 2.12.3 开始，显示电子邮件内容是否为 HTML 的标志。

34.4.2. SES producer 设置的消息标头

标头	类型	描述
CamelAwsSesMessageId	**字符串**	Amazon SES 消息 ID。

34.4.3. 高级 AmazonSimpleEmailService 配置

如果您需要对 **AmazonSimpleEmailService** 实例配置进行更多控制，您可以创建自己的实例并从 URI 引用它：

```
from("direct:start")
.to("aws-ses://example@example.com?amazonSESClient=#client");
```

#client 指的是 Registry 中的 **AmazonSimpleEmailService**。

例如，如果您的 Camel 应用程序在防火墙后面运行：

```
AWSCredentials awsCredentials = new BasicAWSCredentials("myAccessKey", "mySecretKey");
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setProxyHost("http://myProxyHost");
```

```
clientConfiguration.setProxyPort(8080);
AmazonSimpleEmailService client = new AmazonSimpleEmailServiceClient(awsCredentials,
clientConfiguration);

registry.bind("client", client);
```

34.5. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-aws</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.8.4 或更高版本)。

34.6. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- AWS 组件

第 35 章 AWS SIMPLE NOTIFICATION SYSTEM 组件

从 Camel 版本 2.8 开始提供

SNS 组件允许消息发送到 Amazon Simple Notification 主题。Amazon API 的实现由 AWS SDK 提供。

先决条件

您必须有一个有效的 Amazon Web Services 开发人员帐户，并签名以使用 Amazon SNS。如需更多信息，请参阅 Amazon SNS。

35.1. URI 格式

```
aws-sns://topicNameOrArn[?options]
```

如果主题不存在，则会创建它们。
您可以在 URI 中附加查询选项，格式为 **?options=value&option2=value&...**

35.2. URI 选项

AWS Simple Notification System 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	AWS SNS 默认配置		SnsConfiguration
AccessKey (producer)	Amazon AWS 访问密钥		字符串
secretKey (producer)	Amazon AWS Secret 密钥		字符串
region (producer)	SNS 客户端需要工作的区域		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AWS Simple Notification System 端点使用 URI 语法进行配置：

```
aws-sns:topicNameOrArn
```

使用以下路径和查询参数：

35.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
topicNameOrArn	**所需的** 主题名称或 ARN		字符串

Name	描述	默认值	类型

35.2.2. 查询参数(11 参数)：

Name	描述	默认值	类型
amazonSNSClient (producer)	将 AmazonSNS 用作客户端		AmazonSNS
headerFilterStrategy (producer)	使用自定义 HeaderFilterStrategy 将标头映射到/来自 Camel。		HeaderFilterStrategy
messageStructure (producer)	使用 json 的消息结构		字符串
policy (producer)	此队列的策略		字符串
proxyHost (producer)	在实例化 SNS 客户端时定义代理主机		字符串
proxyPort (producer)	在实例化 SNS 客户端时定义代理端口		整数
region (producer)	SNS 客户端需要工作的区域		字符串
subject (producer)	如果邮件标头 'CamelAwsSnsSubject' 不存在，则使用主题。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
accessKey (security)	Amazon AWS 访问密钥		字符串
secretKey (security)	Amazon AWS Secret 密钥		字符串

35.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 16 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws-sns.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-sns.configuration.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-sns.configuration.amazon-s-n-s-client	将 AmazonSNS 用作客户端		AmazonSNS
camel.component.aws-sns.configuration.message-structure	使用 json 的消息结构		字符串
camel.component.aws-sns.configuration.policy	此队列的策略		字符串
camel.component.aws-sns.configuration.proxy-host	在实例化 SNS 客户端时定义代理主机		字符串
camel.component.aws-sns.configuration.proxy-port	在实例化 SNS 客户端时定义代理端口		整数
camel.component.aws-sns.configuration.region	SNS 客户端需要工作的区域		字符串
camel.component.aws-sns.configuration.secret-key	Amazon AWS Secret 密钥		字符串

Name	描述	默认值	类型
camel.component.aws-sns.configuration.subject	如果邮件标头 'CamelAwsSnsSubject' 不存在，则使用主题。		字符串
camel.component.aws-sns.configuration.topic-arn	分配给创建主题的 Amazon 资源名称(ARN)。		字符串
camel.component.aws-sns.configuration.topic-name	主题的名称		字符串
camel.component.aws-sns.enabled	启用 aws-sns 组件	true	布尔值
camel.component.aws-sns.region	SNS 客户端需要工作的区域		字符串
camel.component.aws-sns.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.aws-sns.secret-key	Amazon AWS Secret 密钥		字符串

所需的 SNS 组件选项

您必须在 Registry 或 accessKey 和 secretKey 中提供 amazonSNSClient，才能访问 Amazon 的 SNS。

35.4. 使用方法

35.4.1. 由 SNS producer 评估的消息标头

标头	类型	描述
CamelAwsSnsSubject	**字符串**	Amazon SNS 消息主题。如果没有设置，则使用 **SnsConfiguration** 中的主题。

35.4.2. SNS producer 设置的消息标头

标头	类型	描述
CamelAwsSnsMessageId	**字符串**	Amazon SNS 消息 ID。

35.4.3. 高级 AmazonSNS 配置

如果您需要对 **AmazonSNS** 实例配置进行更多控制，您可以创建自己的实例并从 URI 引用它：

```
from("direct:start")
.to("aws-sns://MyTopic?amazonSNSClient=#client");
```

#client 指的是 Registry 中的 **AmazonSNS**。

例如，如果您的 Camel 应用程序在防火墙后面运行：

```
AWSCredentials awsCredentials = new BasicAWSCredentials("myAccessKey", "mySecretKey");
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setProxyHost("http://myProxyHost");
clientConfiguration.setProxyPort(8080);
AmazonSNS client = new AmazonSNSClient(awsCredentials, clientConfiguration);

registry.bind("client", client);
```

35.5. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-aws</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.8 或更高版本)。

35.6. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- AWS 组件

第 36 章 AWS SIMPLE QUEUE SERVICE 组件

从 Camel 版本 2.6 开始提供

sqs 组件支持向 Amazon 的 SQS 服务发送和接收消息。

先决条件

您必须有一个有效的 Amazon Web Services 开发人员帐户，并签名以使用 Amazon SQS。如需更多信息，请参阅 Amazon SQS。

36.1. URI 格式

```
aws-sqs://queueNameOrArn[?options]
```

如果队列不存在，则会创建队列。
您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

36.2. URI 选项

AWS Simple Queue Service 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	AWS SQS 默认配置		SqsConfiguration
accessKey (common)	Amazon AWS 访问密钥		字符串
secretKey (common)	Amazon AWS Secret 密钥		字符串
region (common)	指定可用于 queueOwnerAWSAccountId 的队列区域，以构建服务 URL。		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AWS Simple Queue Service 端点使用 URI 语法进行配置：

```
aws-sqs:queueNameOrArn
```

使用以下路径和查询参数：

36.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
queueNameOrArn	**所需的** 队列名称或 ARN		字符串

36.2.2. 查询参数(49 参数)：

Name	描述	默认值	类型
amazonAWSHost (common)	Amazon AWS 云的主机名。	amazonaws.com	字符串
amazonSQSClient (common)	将 AmazonSQS 用作客户端		AmazonSQS
headerFilterStrategy (common)	使用自定义 HeaderFilterStrategy 将标头映射到/来自 Camel。		HeaderFilterStrategy
queueOwnerAWSAccountId (common)	当您需要将队列与不同的帐户所有者连接时，指定队列所有者 aws 帐户 ID。		字符串
region (common)	指定可用于 queueOwnerAWSAccountId 的队列区域，以构建服务 URL。		字符串
useIAMCredentials (common)	设置 SQS 客户端是否应该预期在 EC2 实例上加载凭据，或者希望传递静态凭据。	false	布尔值
attributeNames (consumer)	在消费时要接收的属性名称列表。可以使用逗号分隔多个名称。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
concurrentConsumers (consumer)	允许您使用多个线程轮询 sqs 队列来提高吞吐量	1	int
defaultVisibilityTimeout (consumer)	默认可见性超时（以秒为单位）		整数
deleteAfterRead (consumer)	读取后，从 SQS 删除消息	true	布尔值

Name	描述	默认值	类型
deleteIfFiltered (consumer)	如果交换无法通过过滤器，是否将 DeleteMessage 发送到 SQS 队列。如果 'false' 和 Exchange 没有通过路由中的 Camel 过滤器进行，则不要发送 DeleteMessage。	true	布尔值
extendMessageVisibility (consumer)	如果启用，则调度的后台任务将在 SQS 上保持消息可见性。如果处理消息需要很长时间。如果设置为 true defaultVisibilityTimeout，则必须设置。请参阅 Amazon 文档。	false	布尔值
maxMessagesPerPoll (consumer)	获取最大消息数，作为每次轮询轮询的限制。默认为无限限制，但使用 0 或负数来禁用它。		int
messageAttributeNames (consumer)	在消费时要接收的消息属性名称列表。可以使用逗号分隔多个名称。		字符串
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
visibilityTimeout (consumer)	在由 ReceiveMessage 请求检索后，收到的消息会被隐藏在 com.amazonaws.services.sqs.model.SetQueueAttributesRequest 中检索的持续时间（以秒为单位）。这只有在与 defaultVisibilityTimeout 不同时才有意义。它永久更改队列可见性超时属性。		整数
waitTimeSeconds (consumer)	ReceiveMessage 操作调用的持续时间(0 到 20)将等待直到队列中消息包含在响应中。		整数
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
delaySeconds (producer)	延迟发送消息的秒数。		整数

Name	描述	默认值	类型
messageDeduplicationId Strategy (producer)	仅适用于 FIFO 队列。在消息上设置 messageDeduplicationId 的策略。可以是以下选项之一：useExchangeId, useContentBasedDeduplication. 对于 useContentBasedDeduplication 选项，消息中不会设置 messageDeduplicationId。	useExchangeId	MessageDeduplicationId Strategy
messageGroupId Strategy (producer)	仅适用于 FIFO 队列。在消息上设置 messageGroupId 的策略。可以是以下选项之一：useConstant, useExchangeId, usePropertyValue.对于 usePropertyValue 选项，将使用属性 CamelAwsMessageGroupId 的值。		MessageGroupIdStrategy
delayQueue (advanced)	定义您是否要将 delaySeconds 选项应用到队列或单个消息	false	布尔值
queueUrl (advanced)	明确定义 queueUrl：所有其他参数（会影响 queueUrl）将被忽略。这个参数被用来连接到 SQS 的模拟实施，用于测试。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel

Name	描述	默认值	类型
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
proxyHost (proxy)	在实例化 SQS 客户端时定义代理主机		字符串
proxyPort (proxy)	在实例化 SQS 客户端时定义代理端口		整数
maximumMessageSize (queue)	maximumMessageSize （以字节为单位）SQS 消息可以包含此队列。		整数
messageRetentionPeriod (queue)	SQS 为此队列保留一个消息的 messageRetentionPeriod （以秒为单位）。		整数
policy (queue)	此队列的策略		字符串
receiveMessageWaitTime Seconds (queue)	如果您没有在请求中指定 WaitTimeSeconds，则使用 queue 属性 ReceiveMessageWaitTimeSeconds 来确定要等待的时间。		整数
redrivePolicy (queue)	指定发送消息到 DeadLetter 队列的策略。请参阅 Amazon 文档。		字符串
accessKey (security)	Amazon AWS 访问密钥		字符串
secretKey (security)	Amazon AWS Secret 密钥		字符串

36.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 31 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws-sqs.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-sqs.configuration.access-key	Amazon AWS 访问密钥		字符串
camel.component.aws-sqs.configuration.amazon-a-w-s-host	Amazon AWS 云的主机名。	amazonaws.com	字符串
camel.component.aws-sqs.configuration.amazon-s-q-s-client	将 AmazonSQS 用作客户端		AmazonSQS
camel.component.aws-sqs.configuration.attribute-names	在消费时要接收的属性名称列表。可以使用逗号分隔多个名称。		字符串
camel.component.aws-sqs.configuration.concurrent-consumers	允许您使用多个线程轮询 sqs 队列来提高吞吐量	1	整数
camel.component.aws-sqs.configuration.default-visibility-timeout	默认可见性超时（以秒为单位）		整数
camel.component.aws-sqs.configuration.delay-queue	定义您是否要将 delaySeconds 选项应用到队列或单个消息	false	布尔值
camel.component.aws-sqs.configuration.delay-seconds	延迟发送消息的秒数。		整数

Name	描述	默认值	类型
camel.component.aws-sqs.configuration.delete-after-read	读取后，从 SQS 删除消息	true	布尔值
camel.component.aws-sqs.configuration.delete-if-filtered	如果交换无法通过过滤器，是否将 DeleteMessage 发送到 SQS 队列。如果 'false' 和 Exchange 没有通过路由中的 Camel 过滤器进行，则不要发送 DeleteMessage。	true	布尔值
camel.component.aws-sqs.configuration.extend-message-visibility	如果启用，则调度的后台任务将在 SQS 上保持消息可见性。如果处理消息需要很长时间。如果设置为 true defaultVisibilityTimeout，则必须设置。请参阅 Amazon 文档。	false	布尔值
camel.component.aws-sqs.configuration.maximum-message-size	maximumMessageSize （以字节为单位）SQS 消息可以包含此队列。		整数
camel.component.aws-sqs.configuration.message-attribute-names	在消费时要接收的消息属性名称列表。可以使用逗号分隔多个名称。		字符串
camel.component.aws-sqs.configuration.message-retention-period	SQS 为此队列保留一个消息的 messageRetentionPeriod （以秒为单位）。		整数
camel.component.aws-sqs.configuration.policy	此队列的策略		字符串
camel.component.aws-sqs.configuration.proxy-host	在实例化 SQS 客户端时定义代理主机		字符串
camel.component.aws-sqs.configuration.proxy-port	在实例化 SQS 客户端时定义代理端口		整数

Name	描述	默认值	类型
camel.component.aws-sqs.configuration.queue-name	队列的名称。如果队列不存在，将创建队列。		字符串
camel.component.aws-sqs.configuration.queue-owner-aws-account-id	当您需要将队列与不同的帐户所有者连接时，指定队列所有者 aws 帐户 ID。		字符串
camel.component.aws-sqs.configuration.queue-url	明确定义 queueUrl：所有其他参数（会影响 queueUrl）将被忽略。这个参数被用来连接到 SQS 的模拟实施，用于测试。		字符串
camel.component.aws-sqs.configuration.receive-message-wait-time-seconds	如果您没有在请求中指定 WaitTimeSeconds，则使用 queue 属性 ReceiveMessageWaitTimeSeconds 来确定要等待的时间。		整数
camel.component.aws-sqs.configuration.redrive-policy	指定发送消息到 DeadLetter 队列的策略。请参阅 Amazon 文档。		字符串
camel.component.aws-sqs.configuration.region	指定可用于 queueOwnerAWSAccountId 的队列区域，以构建服务 URL。		字符串
camel.component.aws-sqs.configuration.secret-key	Amazon AWS Secret 密钥		字符串
camel.component.aws-sqs.configuration.visibility-timeout	在由 ReceiveMessage 请求检索后，收到的消息会被隐藏在 com.amazonaws.services.sqs.model.SetQueueAttributesRequest 中检索的持续时间（以秒为单位）。这只有在与 defaultVisibilityTimeout 不同时才有意义。它永久更改队列可见性超时属性。		整数
camel.component.aws-sqs.configuration.wait-time-seconds	ReceiveMessage 操作调用的持续时间(0 到 20)将等待直到队列中消息包含在响应中。		整数

Name	描述	默认值	类型
camel.component.aws-sqs.enabled	启用 aws-sqs 组件	true	布尔值
camel.component.aws-sqs.region	指定可用于 queueOwnerAWSAccountId 的队列区域，以构建服务 URL。		字符串
camel.component.aws-sqs.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.aws-sqs.secret-key	Amazon AWS Secret 密钥		字符串

所需的 SQS 组件选项

您必须在 Registry 或 accessKey 和 secretKey 中提供 amazonSQSClient，才能访问 Amazon 的 SQS。

36.4. BATCH CONSUMER

这个组件实现了 Batch Consumer。

这样，您可以让实例知道此批处理中存在多少个消息，而实例则让聚合器聚合此消息数量。

36.5. 使用方法

36.5.1. SQS producer 设置的消息标头

标头	类型	描述
CamelAwsSqsMD5OfBody	**字符串**	Amazon SQS 消息的 MD5 checksum。
CamelAwsSqsMessageId	**字符串**	Amazon SQS 消息 ID。

标头	类型	描述
CamelAwsSqsDelaySeconds	**整数**	自 **Camel 2.11** 起，Amazon SQS 消息可以看到的延迟秒数。

36.5.2. SQS consumer 设置的消息标头

标头	类型	描述
CamelAwsSqsMD5OfBody	**字符串**	Amazon SQS 消息的 MD5 checksum。
CamelAwsSqsMessageId	**字符串**	Amazon SQS 消息 ID。
CamelAwsSqsReceiptHandle	**字符串**	Amazon SQS 消息接收句柄。
CamelAwsSqsMessageAttributes	**Map<String, String>**	Amazon SQS 消息属性。

36.5.3. 高级 AmazonSQS 配置

如果您的 Camel 应用程序在防火墙后面运行，或者您需要对 AmazonSQS 实例配置拥有更多控制，您可以创建自己的实例：

```
AWSCredentials awsCredentials = new BasicAWSCredentials("myAccessKey", "mySecretKey");

ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setProxyHost("http://myProxyHost");
clientConfiguration.setProxyPort(8080);

AmazonSQS client = new AmazonSQSClient(awsCredentials, clientConfiguration);

registry.bind("client", client);
```

-

并在 Camel aws-sqs 组件配置中引用它：

```
from("aws-sqs://MyQueue?amazonSQSClient=#client&delay=5000&maxMessagesPerPoll=5")
.to("mock:result");
```

36.5.4. 创建或更新 SQS 队列

在 SQS 组件中，当端点启动时，会执行检查来获取与队列存在的信息。您可以使用 SQSConfiguration 选项通过 QueueAttributeName 映射自定义创建。

```
from("aws-sqs://MyQueue?amazonSQSClient=#client&delay=5000&maxMessagesPerPoll=5")
.to("mock:result");
```

在本例中，如果 AWS 上尚未创建 MyQueue 队列，它将使用 SQS 配置中的默认参数创建。如果已在 AWS 上启动，则使用 SQS 配置选项来覆盖现有的 AWS 配置。

36.5.5. DelayQueue VS Delay for Single 消息

从 2.23.0，组件具有一个新的选项：delayQueue。当选项设为 true 时，SQS Queue 将是一个 DelayQueue，带有 DelaySeconds 选项作为延迟。有关 DelayQueue 的更多信息，您可以阅读 AWS SQS 文档。要考虑的一个重要信息是：

- 对于标准队列，每个队列延迟设置不会重新修改设置不会影响队列中已存在的消息的延迟。
- 对于 FIFO 队列，每个队列延迟设置正在重新修改设置，会影响队列中已存在的消息延迟。

如官方文档中所述。如果要在单个消息上指定延迟，您可以忽略 delayQueue 选项，但如果您需要向所有消息添加固定延迟，则可以将此选项设置为 true。

36.5.6. 在 SQS 组件中使用 AWS IAM 凭证

要使用 AWS IAM 凭证，您必须首先确认启动 Camel 应用程序的 EC2 实例具有与其关联的 IAM 角色，并附加适当的策略。

此功能应只在远程实例上设置为 **true**。另外，还必须在本地使用静态凭证，因为 IAM 是 AWS 特定组件。

要实现此功能，**请将IAMCredentials** 设置为 **true**。

注意

要根据本地和远程环境打开和关闭此功能，您可以考虑使用系统环境变量启用此查询参数。例如，如果名为 **isRemote** 的系统环境变量设为 **true**，您的代码可能会将 **useIAMCredentials** query 参数设置为 **true**。

虽然此功能不会完全消除对静态凭证的需求，但在 AWS 环境中使用 IAM 凭证不再需要刷新远程环境，并更安全，因为 IAM 凭证每 6 小时自动刷新，并在更新 EC2 安全策略时进行更新。

这是 AWS 推荐的管理凭证的方法，因此应尽可能使用。

36.6. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-aws</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.6 或更高版本)。

36.7. JMS-STYLE SELECTORS

SQS 不允许选择器，但您可以使用 Camel Filter EIP 和设置适当的 **visibilityTimeout** 来有效地达到此目的。当 SQS 分配消息时，它将在尝试将消息分配给不同的消费者之前等待可见性超时，除非收到 DeleteMessage。默认情况下，Camel 始终在路由末尾发送 DeleteMessage，除非路由失败。要实现适当的过滤，且不会在成功完成路由时发送 DeleteMessage，请使用 Filter：

```
from("aws-sqs://MyQueue?
amazonSQSClient=#client&defaultVisibilityTimeout=5000&deleteIfFiltered=false")
.filter("${header.login} == true")
.to("mock:result");
```

在上面的代码中，如果交换没有适当的标头，它不会通过过滤器 AND 将其从 SQS 队列中删除。5000 miliseconds 后，消息将对其他消费者可见。

36.8. 删除单一消息

使用 deleteMessage 操作删除单个消息。您需要为您要删除的消息设置接收句柄标头。

```
from("direct:start")
  .setHeader(SqsConstants.SQS_OPERATION, constant("deleteMessage"))
  .setHeader(SqsConstants.RECEIPT_HANDLE, constant("123456"))
  .to("aws-sqs://camel-1?accessKey=RAW(xxx)&secretKey=RAW(xxx)&region=EU_WEST_1");
```

因此，您将获取一个包含 **DeleteMessageResult** 实例的交换，您可以使用它来检查消息是否已被删除。

36.9. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- AWS 组件

第 37 章 AWS SIMPLE WORKFLOW 组件

从 Camel 版本 2.13 开始提供

Simple Workflow 组件支持从 Amazon 简单工作流服务 管理工作流。

先决条件

您必须有一个有效的 Amazon Web Services 开发人员帐户，并签名以使用 Amazon Simple 工作流。如需更多信息，请参阅 Amazon Simple Workflow。

37.1. URI 格式

```
aws-swf://<workflow|activity>[?options]
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

37.2. URI 选项

AWS Simple Workflow 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	AWS SWF 默认配置		SWFConfiguration
accessKey (common)	Amazon AWS 访问密钥.		字符串
secretKey (common)	Amazon AWS Secret 密钥。		字符串
region (common)	Amazon AWS 区域。		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

AWS Simple Workflow 端点使用 URI 语法进行配置：

```
aws-swf:type
```

使用以下路径和查询参数：

37.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
type	**所需的** 活动或工作流		字符串

37.2.2. 查询参数(30 参数)：

Name	描述	默认值	类型
amazonSWClient (common)	使用给定的 AmazonSimpleWorkflowClient 作为客户端		AmazonSimpleWorkflow Client
dataConverter (common)	一个 com.amazonaws.services.simpleworkflow.flow.DataConverter 实例，用于序列化/禁止数据。		DataConverter
domainName (common)	要使用的工作流域。		字符串
eventName (common)	要使用的工作流或活动事件名称。		字符串
region (common)	Amazon AWS 区域。		字符串
version (common)	要使用的工作流或活动事件版本。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
ClientConfiguration Parameters (advanced)	使用映射中的键/值配置 ClientConfiguration。		Map
startWorkflowOptions 参数 (advanced)	使用映射中的键/值配置 StartWorkflowOptions。		Map
sWClientParameters (advanced)	使用映射中的键/值配置 AmazonSimpleWorkflowClient。		Map
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

Name	描述	默认值	类型
activityList （活动）	要使用活动的列表名称。		字符串
activitySchedulingOptions (activity)	活动调度选项		ActivityScheduling 选项
activityThreadPoolSize (activity)	工作池中的最大线程数用于活动。	100	int
activityTypeExecution Options (activity)	活动执行选项		ActivityTypeExecution 选项
activityTypeRegistration Options (activity)	活动注册选项		ActivityType RegistrationOptions
childPolicy (workflow)	终止工作流时，要在子工作流上使用的策略。		字符串
executionStartToClose Timeout (workflow)	将执行 start 设置为关闭超时。	3600	字符串
操作 （工作流）	工作流操作	开始	字符串
signalName (workflow)	发送到工作流的信号的名称。		字符串
stateResultType (workflow)	查询工作流状态时的结果类型。		字符串
taskStartToClose Timeout (workflow)	将任务 start 设置为关闭超时。	600	字符串
terminationDetails (workflow)	终止工作流的详情。		字符串
terminationReason (workflow)	终止工作流的原因。		字符串
workflowList (workflow)	使用工作流的列表名称。		字符串

Name	描述	默认值	类型
workflowTypeRegistration Options (workflow)	工作流注册选项		WorkflowTypeRegistrationOptions
accessKey (security)	Amazon AWS 访问密钥.		字符串
secretKey (security)	Amazon AWS Secret 密钥。		字符串

37.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 32 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws-swf.access-key	Amazon AWS 访问密钥.		字符串
camel.component.aws-swf.configuration.access-key	Amazon AWS 访问密钥.		字符串
camel.component.aws-swf.configuration.activity-list	要使用活动的列表名称。		字符串
camel.component.aws-swf.configuration.activity-scheduling-options	活动调度选项		ActivityScheduling 选项
camel.component.aws-swf.configuration.activity-thread-pool-size	工作池中的最大线程数用于活动。	100	整数

Name	描述	默认值	类型
camel.component.aws-swf.configuration.activity-type-execution-options	活动执行选项		ActivityTypeExecution 选项
camel.component.aws-swf.configuration.activity-type-registration-options	活动注册选项		ActivityTypeRegistrationOptions
camel.component.aws-swf.configuration.amazon-s-w-client	使用给定的 AmazonSimpleWorkflowClient 作为客户端		AmazonSimpleWorkflow Client
camel.component.aws-swf.configuration.child-policy	终止工作流时，要在子工作流上使用的策略。		字符串
camel.component.aws-swf.configuration.client-configuration-parameters	使用映射中的键/值配置 ClientConfiguration。		Map
camel.component.aws-swf.configuration.data-converter	一个 com.amazonaws.services.simpleworkflow.flow.DataConverter 实例，用于序列化/禁止数据。		DataConverter
camel.component.aws-swf.configuration.domain-name	要使用的工作流域。		字符串
camel.component.aws-swf.configuration.event-name	要使用的工作流或活动事件名称。		字符串

Name	描述	默认值	类型
camel.component.aws-swf.configuration.execution-start-to-close-timeout	将执行 start 设置为关闭超时。	3600	字符串
camel.component.aws-swf.configuration.operation	工作流操作	开始	字符串
camel.component.aws-swf.configuration.region	Amazon AWS 区域。		字符串
camel.component.aws-swf.configuration.s-w-client-parameters	使用映射中的键/值配置 AmazonSimpleWorkflowClient。		Map
camel.component.aws-swf.configuration.secret-key	Amazon AWS Secret 密钥。		字符串
camel.component.aws-swf.configuration.signal-name	发送到工作流的信号的名称。		字符串
camel.component.aws-swf.configuration.start-workflow-options-parameters	使用映射中的键/值配置 StartWorkflowOptions。		Map
camel.component.aws-swf.configuration.state-result-type	查询工作流状态时的结果类型。		字符串
camel.component.aws-swf.configuration.task-start-to-close-timeout	将任务 start 设置为关闭超时。	600	字符串

Name	描述	默认值	类型
camel.component.aws-swf.configuration.termination-details	终止工作流的详情。		字符串
camel.component.aws-swf.configuration.termination-reason	终止工作流的原因。		字符串
camel.component.aws-swf.configuration.type	活动或工作流		字符串
camel.component.aws-swf.configuration.version	要使用的工作流或活动事件版本。		字符串
camel.component.aws-swf.configuration.workflow-list	使用工作流的列表名称。		字符串
camel.component.aws-swf.configuration.workflow-type-registration-options	工作流注册选项		WorkflowTypeRegistrationOptions
camel.component.aws-swf.enabled	启用 aws-swf 组件	true	布尔值
camel.component.aws-swf.region	Amazon AWS 区域。		字符串
camel.component.aws-swf.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Name	描述	默认值	类型
camel.component.aws-swf.secret-key	Amazon AWS Secret 密钥。		字符串

所需的 SWF 组件选项

您必须在 Registry 或 accessKey 和 secretKey 中提供 amazonSWClient，才能访问 Amazon 的 Simple Workflow Service。

37.4. 使用方法

37.4.1. SWF Workflow Producer 评估的消息标头

工作流制作者允许与工作流交互。它可以启动新的工作流执行，查询其状态，向正在运行的工作流发送信号，或者终止和取消它。

标头	类型	描述
CamelSWFOperation	**字符串**	要在工作流上执行的操作。支持的操作包括： SIGNAL, CANCEL, TERMINATE, GET_STATE, START, DESCRIBE, GET_HISTORY。
CamelSWFWorkflowId	**字符串**	要使用的工作流 ID。
CamelAwsDdbKeyCamelSWFRunId	**字符串**	要使用的更糟糕的运行 ID。
CamelSWFStateResultType	**字符串**	查询工作流状态时的结果类型。
CamelSWFEventName	**字符串**	要使用的工作流或活动事件名称。

标头	类型	描述
CamelSWFVersion	**字符串**	要使用的工作流或活动事件版本。
CamelSWFReason	**字符串**	终止工作流的原因。
CamelSWFDetails	**字符串**	终止工作流的详情。
CamelSWFChildPolicy	**字符串**	终止工作流时，要在子工作流上使用的策略。

37.4.2. SWF Workflow Producer 设置的消息标头

标头	类型	描述
CamelSWFWorkflowId	**字符串**	使用的 worfklow ID 或新生成的。
CamelAwsDdbKey CamelSWFRunId	**字符串**	worfklow 运行 ID 使用或生成的。

37.4.3. SWF Workflow Consumer 设置的消息标头

工作流使用者代表工作流逻辑。启动之后，它将开始轮询工作流决策任务并处理它们。除了处理决策任务外，工作流消费者路由还接收信号（从工作流制作者发送）或状态查询。工作流消费者的主要目的是调度使用活动制作者执行的活动任务。实际上的活动任务只能从工作流消费者启动的线程中调度。

标头	类型	描述
CamelSWFAction	**字符串**	指明 什么是当前事件：CamelSWFActionExecute、CamelSWFSignalReceivedAction 或 CamelSWFGetStateAction。

标头	类型	描述
CamelSWFWorkflowReplaying	**布尔值**	指明当前决策任务是否为重播。
CamelSWFWorkflowStartTime	**long**	此决策任务的开始事件时间。

37.4.4. SWF Activity Producer 设置的消息标头

活动制作者允许调度活动任务。活动制作者只能从由工作流消费者发起的线程使用，它可以处理由工作流消费者启动的同步交换。

标头	类型	描述
CamelSWFEventName	**字符串**	要调度的活动名称。
CamelSWFVersion	**字符串**	要调度的活动版本。

37.4.5. SWF Activity Consumer 设置的消息标头

标头	类型	描述
CamelSWFTaskToken	**字符串**	报告手动完成任务完成的任务令牌。

37.4.6. 高级 amazonSWClient 配置

如果您需要对 AmazonSimpleWorkflowClient 实例配置进行更多控制，您可以创建自己的实例并从 URI 引用它：

#client 指的是 Registry 中的 AmazonSimpleWorkflowClient。

例如，如果您的 Camel 应用程序在防火墙后面运行：

```
AWSCredentials awsCredentials = new BasicAWSCredentials("myAccessKey", "mySecretKey");
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setProxyHost("http://myProxyHost");
clientConfiguration.setProxyPort(8080);

AmazonSimpleWorkflowClient client = new AmazonSimpleWorkflowClient(awsCredentials,
clientConfiguration);

registry.bind("client", client);
```

37.5. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-aws</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.13 或更高版本)。

37.6. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

AWS 组件

第 38 章 AWS XRAY 组件

从 Camel 2.21 开始提供

camel-aws-xray 组件用于追踪和使用 AWS XRay 的传入和传出 Camel 消息进行跟踪和计时。

为发送到/来自 Camel 的传入和传出消息捕获事件（子网段）。

38.1. 依赖项

要将 AWS XRay 支持包含在 Camel 中，需要添加包含 Camel 相关 AWS XRay 相关类的存档。此外，还需要提供 AWS XRay 库。

要同时包含 AWS XRay 和 Camel，依赖项使用以下 Maven 导入：

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-xray-recorder-sdk-bom</artifactId>
      <version>1.3.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
    <dependency>
      <groupId>org.apache.camel</groupId>
      <artifactId>camel-aws-xray</artifactId>
    </dependency>

    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-xray-recorder-sdk-core</artifactId>
    </dependency>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-xray-recorder-sdk-aws-sdk</artifactId>
    </dependency>
<dependencies>
```

38.2. 配置

AWS XRay tracer 的配置属性有：

选项	默认	描述
addExcludePatterns		设置将针对与模式匹配的 Camel 消息禁用追踪的排除模式。内容是一个 Set<String>，其中键是与 routeId 的模式匹配。该模式使用来自 Intercept 的规则。

选项	默认	描述
setTracingStrategy	NoopTracingStrategy	允许提供自定义 Camel **InterceptStrategy** 来跟踪调用的处理器定义，如 **BeanDefinition** 或 **ProcessDefinition**。**TraceAnnotatedTracingStrategy** 将跟踪通过 **.bean (...)** 或 **.process (...)** 调用的任何类，其中包含类级别的 **@XRayTrace** 注释。

目前，只有 AWS XRay tracer 可以配置为为 Camel 应用程序提供分布式追踪：

38.2.1. explicit

在您的 POM 中包含 **camel-aws-xray** 组件，以及与 AWS XRay Tracer 关联的任何特定依赖项。

要明确配置 AWS XRay 支持，请实例化 **XRayTracer** 并初始化 camel 上下文。您可以选择指定 **Tracer**，也可以使用 **Registry** 或 **ServiceLoader** 隐式发现。

```
XRayTracer xrayTracer = new XRayTracer();
// By default it uses a NoopTracingStrategy, but you can override it with a specific InterceptStrategy implementation.
xrayTracer.setTracingStrategy(...);
// And then initialize the context
xrayTracer.init(camelContext);
```

要在 XML 中使用 XRayTracer，您需要做的都是定义 AWS XRay tracer bean。Camel 将自动发现并使用它。

```
<bean id="tracingStrategy" class="..."/>
<bean id="aws-xray-tracer" class="org.apache.camel.component.aws.xray.XRayTracer" />
  <property name="tracer" ref="tracingStrategy"/>
</bean>
```

如果默认 **NoopTracingStrategy** 只跟踪创建和删除交换，但不会跟踪某些 Bean 或 EIP 模式的调用。

38.2.2. 跟踪全面的路由执行

为了跟踪多个路由之间交换的执行，在交换创建时，会在标头中生成并存储在标头中（如果没有对应的值）。此追踪 ID 被复制到新的交换，以便保持已处理交换的一致性视图。

由于 AWS XRay trace 会根据当前子/网段的本地线程工作，因此应复制到新线程，并根据 AWS XRay 文档中所述 进行设置。因此，Camel AWS XRay 组件提供了一个额外的标头字段，组件将传递的 AWS XRay **Entity** 设置为新线程，从而使跟踪的数据保留在路由中，而不是公开一个新的片段，它看似与任何执行的路由无关。

组件将使用交换标头中找到的以下常量：

标头	描述
Camel-AWS-XRay-Trace-ID	包含对 AWS XRay **TraceID** 对象的引用，以提供调用路由的综合视图

标头	描述
Camel-AWS-XRay-Trace-Entity	包含对复制到新线程的实际 AWS XRay **Segment** 或 **Subsegment** 的引用。当生成新线程时，应设置此标头，并且执行的任务应作为执行的路由的一部分公开，而不是创建新的不相关的片段。

请注意，AWS XRay **Entity** （例如，**Segment** 和 **Subsegment**）不可序列化，因此不应传递给其他 JVM 进程。

38.3. EXAMPLE

您可以找到一个示例演示了在此项目附带的测试中配置 AWS XRay tracing 的方法。

第 39 章 WINDOWS AZURE SERVICES 的 CAMEL 组件

Windows Azure Services 的 Camel 组件提供从 Camel 到 Azure 服务的连接。

Azure Service	Camel 组件	Camel 版本	组件描述
Storage Blob Service	Azure-Blob	2.9.0	支持存储和检索 Blob
存储队列服务	Azure-Queue	2.9.0	支持在队列中存储和检索消息

第 40 章 AZURE STORAGE BLOB SERVICE 组件

从 Camel 版本 2.19 开始提供

Azure Blob 组件支持存储和检索 Blob 到 Azure Storage Blob 服务。

先决条件

您必须有一个有效的 Windows Azure Storage 帐户。如需更多信息，请参阅 Azure 文档门户。

40.1. URI 格式

```
azure-blob://accountName/containerName[/blobName][?options]
```

在大多数情况下，需要一个 blobName，如果不存在 blob，则会创建 blob。
您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

例如，若要从 **camelazure** 存储帐户中的 **container1** 上的 public 块 **blob** 块**Blob** 下载 blob 内容，请使用以下片断：

```
from("azure-blob:/camelazure/container1/blockBlob").
to("file://blobdirectory");
```

40.2. URI 选项

Azure Storage Blob Service 组件没有选项。

Azure Storage Blob Service 端点使用 URI 语法进行配置：

```
azure-blob:containerOrBlobUri
```

使用以下路径和查询参数：

40.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
containerOrBlobUri	**所需的** Container 或 Blob 紧凑 Uri		字符串

40.2.2. 查询参数(19 参数)：

Name	描述	默认值	类型
azureBlobClient (common)	blob 服务客户端		CloudBlob
blobOffset (common)	为上传或下载操作设置 blob 偏移，默认为 0	0	Long

Name	描述	默认值	类型
blobType (common)	设置 blob 类型，'blockblob' 是默认的	blockblob	BlobType
closeStreamAfterRead (common)	读取或保持打开后，关闭流，默认为 true	true	布尔值
credentials (common)	设置存储凭证，大多数情况下需要		StorageCredentials
dataLength (common)	为下载或页面 blob 上传操作设置数据长度		Long
fileDir (common)	设置下载的 blob 将保存到的文件目录		字符串
publicForRead (common)	如果启用此属性，则存储资源可以被公开读取其内容，则不必设置凭证	false	布尔值
streamReadSize (common)	在读取 blob 内容时设置最小读取大小（以字节为单位）		int
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
blobMetadata (producer)	设置 blob meta-data		Map
blobPrefix (producer)	设置可用于列出 Blob 的前缀		字符串
closeStreamAfterWrite (producer)	在写入或保持打开后关闭流，默认为 true	true	布尔值
operation (producer)	到制作者的 Blob 服务操作提示	listBlobs	BlobServiceOperations

Name	描述	默认值	类型
streamWriteSize (producer)	设置写入块和页面块的缓冲区的大小		int
useFlatListing (producer)	指定是否应使用 flat 或ier blob 列表	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

40.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.azure-blob.enabled	启用 azure-blob 组件	true	布尔值
camel.component.azure-blob.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

所需的 Azure Storage Blob Service 组件选项

如果需要访问私有 blob，您必须提供 containerOrBlob 名称和凭证。

40.4. 使用方法

40.4.1. Azure Storage Blob Service producer 评估的消息标头

标头	类型	描述

40.4.2. Azure Storage Blob Service producer 设置的消息标头

标头	类型	描述
CamelFileName	**字符串**	下载 blob 内容的文件名。

40.4.3. Azure Storage Blob Service producer 使用者设置的消息标头

标头	类型	描述
CamelFileName	**字符串**	下载 blob 内容的文件名。

40.4.4. Azure Blob Service 操作

适用于所有块类型的操作

操作	描述
getBlob	获取 blob 的内容。您可以将此操作的输出限制为 blob 范围。
deleteBlob	删除 blob。
listBlobs	列出 Blob。

块 blob 操作

操作	描述
updateBlockBlob	放置块 blob 内容，用于创建新块 blob 或覆盖现有的块 blob 内容。
uploadBlobBlocks	上传块 blob 内容，首先生成 blob 块序列，然后将它们提交到 blob。如果启用消息 CommitBlockListLater 属性，您可以稍后使用 **commitBlobBlockList** 操作来执行提交。您可以稍后更新单个块 Blob。
commitBlobBlockList	将 blob 块序列提交到您之前上传到 blob 服务的块列表（使用启用了消息 CommitBlockListLater 的 **updateBlockBlob** 操作）。
getBlobBlockList	获取块 blob 列表。

附加 blob 操作

操作	描述
createAppendBlob	创建附加块。默认情况下，如果块已存在，则不重置。请注意，您可以通过启用消息 AppendBlobCreated 属性并使用 **updateAppendBlob** 操作来替代创建一个 append blob。
updateAppendBlob	将新内容附加到 blob。如果不存在，此操作也会创建 blob，如果您启用了消息 AppendBlobCreated 属性。

页面块操作

操作	描述
createPageBlob	创建 page 块。默认情况下，如果块已存在，则不重置。请注意，您还可以通过启用消息 PageBlobCreated 属性并使用 **updatePageBlob** 操作来创建页面 blob （并设置其内容）。
updatePageBlob	创建一个页面块（除非启用消息 PageBlobCreated 属性以及相同的命名块已存在），并设置此 blob 的内容。
resizePageBlob	调整页面 blob 的大小。
clearPageBlob	清除 page blob。
getPageBlobRanges	获取页面 blob 页面范围。

40.4.5. Azure Blob Client 配置

如果您的 Camel 应用程序在防火墙后面运行，或者需要对 Azure Blob 客户端配置进行更多控制，您可以创建自己的实例：

```
StorageCredentials credentials = new StorageCredentialsAccountAndKey("camelazure", "thekey");

CloudBlob client = new CloudBlob("camelazure", credentials);

registry.bind("azureBlobClient", client);
```

并在 Camel azure-blob 组件配置中引用它：

```
from("azure-blob:/camelazure/container1/blockBlob?azureBlobClient=#client")
.to("mock:result");
```

40.5. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

■

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-azure</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.19.0 或更高版本)。

40.6. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- Azure Component

第 41 章 AZURE STORAGE QUEUE SERVICE 组件

从 Camel 版本 2.19 开始提供

Azure Queue 组件支持向 Azure Storage Queue 服务存储和检索信息。

先决条件

您必须有一个有效的 Microsoft Azure 帐户。如需更多信息，请参阅 Azure Portal。

41.1. URI 格式

```
azure-queue://accountName/queueName[?options]
```

如果队列尚不存在，则会创建队列。
您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

例如，若要从 **camelazure** 存储帐户中的队列 **messageQueue** 获取消息内容，并使用以下代码片段：

```
from("azure-queue:/camelazure/messageQueue").
to("file://queuedirectory");
```

41.2. URI 选项

Azure Storage Queue Service 组件没有选项。

Azure Storage Queue Service 端点使用 URI 语法进行配置：

```
azure-queue:containerAndQueueUri
```

使用以下路径和查询参数：

41.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
containerAndQueueUri	**所需的** 容器队列紧凑 Uri		字符串

41.2.2. 查询参数(10 parameters):

Name	描述	默认值	类型
azureQueueClient (common)	队列服务客户端		CloudQueue
credentials (common)	设置存储凭证，大多数情况下需要		StorageCredentials

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
messageTimeToLive (producer)	消息生存时间（以秒为单位）		int
messageVisibilityDelay (producer)	消息 Visibility Delay （以秒为单位）		int
operation (producer)	当用户不想发送消息时，要执行的操作。有三个 enums 选项，值可以是以下之一： sendBatchMessage, deleteMessage, listQueues	listQueues	QueueServiceOperations
queuePrefix (producer)	设置可用于列出队列的前缀		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

41.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.azure-queue.enabled	启用 azure-queue 组件	true	布尔值

Name	描述	默认值	类型
camel.component.azure-queue.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

所需的 Azure Storage Queue Service 组件选项

您必须提供 containerAndQueue URI 和凭证。

41.4. 使用方法

41.4.1. 由 Azure Storage Queue Service producer 评估的消息标头

标头	类型	描述

41.4.2. Azure Storage Queue Service producer 设置的消息标头

标头	类型	描述

41.4.3. Azure Storage Queue Service producer 使用者设置的消息标头

标头	类型	描述

41.4.4. Azure Queue Service 操作

操作	描述
listQueues	列出队列。
createQueue	创建队列。
deleteQueue	删除队列。
addMessage	向队列添加消息。

操作	描述
retrieveMessage	从队列检索消息。
peekMessage	查看队列中的消息，例如，确定消息是否到达正确的队列。
updateMessage	更新队列中的消息。
deleteMessage	删除队列中的消息。

41.4.5. Azure Queue Client 配置

如果您的 Camel 应用程序在防火墙后面运行，或者需要对 Azure Queue Client 配置进行更多控制，您可以创建自己的实例：

```
StorageCredentials credentials = new StorageCredentialsAccountAndKey("camelazure", "thekey");

CloudQueue client = new CloudQueue("camelazure", credentials);

registry.bind("azureQueueClient", client);
```

并在 Camel azure-queue 组件配置中引用它：

```
from("azure-queue:/camelazure/messageQueue?azureQueueClient=#client")
.to("mock:result");
```

41.5. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-azure</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.19.0 或更高版本)。

41.6. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- Azure Component

第 42 章 BARCODE DATAFORMAT

从 Camel 版本 2.14 开始提供

barcode 数据格式基于 zxing 库。此组件的目标是从 String (marshal)和来自 barcode 镜像(unmarshal)的 String 创建 barcode 镜像。您可以自由地使用 zxing 所提供的所有功能。

42.1. 依赖项

要在 camel 路由中使用 barcode 数据格式，您需要添加对实现此数据格式的 **camel-barcode** 的依赖。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-barcode</artifactId>
  <version>x.x.x</version>
</dependency>
```

42.2. BARCODE 选项

Barcode dataformat 支持 5 个选项，如下所列。

Name	默认值	Java 类型	描述
width		**整数**	barcode 的宽度
height		**整数**	barcode 的高度
imageType		**字符串**	barcode 的镜像类型，如 png
barcodeFormat		**字符串**	Barcode 格式，如 QR-Code
contentTypeHeader	**false**	**布尔值**	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

42.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.barcode.barcode-format	Barcode 格式，如 QR-Code		字符串

Name	描述	默认值	类型
camel.dataformat.barcode.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.barcode.enabled	启用 barcode dataformat	true	布尔值
camel.dataformat.barcode.height	barcode 的高度		整数
camel.dataformat.barcode.image-type	barcode 的镜像类型，如 png		字符串
camel.dataformat.barcode.width	barcode 的宽度		整数

ND

42.4. 使用 JAVA DSL

首先，您必须初始化 barcode 数据 fomat 类。您可以使用默认构造器或参数化之一（请参阅 JavaDoc）。默认值为：

参数	默认值
镜像类型 (BarcodeImageType)	PNG
width	100 px
height	100 px
编码	UTF-8
Barcode 格式 (BarcodeFormat)	QR-Code

```
// QR-Code default
DataFormat code = new BarcodeDataFormat();
```

如果要使用 zxing hints，您可以使用 BarcodeDataFormat 实例的 'addToHintMap' 方法：

```
code.addToHintMap(DecodeHintType.TRY_HARDER, Boolean.true);
```

如需可能提示，请参阅 xzing 文档。

42.4.1. Marshalling

```
from("direct://code")
  .marshal(code)
  .to("file://barcode_out");
```

您可以使用以下方法从测试类调用路由：

```
template.sendBody("direct://code", "This is a testmessage!");
```

您应该在这个镜像的 'barcode_out' 文件夹中找到：

42.4.2. unmarshalling

unmarshaller 是通用的。对于 unmarshalling，您可以使用任何 BarcodeDataFormat 实例。如果您有两个实例，一个用于（生成）QR-Code，一个用于 PDF417，这无关紧要使用哪个实例。

```
from("file://barcode_in?noop=true")
  .unmarshal(code) // for unmarshalling, the instance doesn't matter
  .to("mock:out");
```

如果您将上面的 QR-Code 镜像粘贴到 'barcode_in' 文件夹中，您应该在模拟中找到 'This is a testmessage!'。您可以将 barcode 数据格式作为标头变量找到：

Name	类型	描述
BarcodeFormat	字符串	com.google.zxing.BarcodeFormat 的值。

第 43 章 BASE64 DATAFORMAT

从 Camel 版本 2.11 开始提供

Base64 数据格式用于 base64 编码和解码。

43.1. 选项

Base64 dataformat 支持 4 个选项，如下所列。

Name	默认值	Java 类型	描述
lineLength	76	整数	特定编码数据的最大行长度。默认使用 76。
lineSeparator		字符串	要使用的行分隔符。默认使用新行字符(CRLF)。
urlSafe	false	布尔值	我们单独发出 '-' 和 '_'，而不是发出 '-' 和 '_'。urlSafe 只应用于编码操作。解码无缝处理这两种模式。默认为 false。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

43.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.base64.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.base64.enabled	启用 base64 dataformat	true	布尔值
camel.dataformat.base64.line-length	特定编码数据的最大行长度。默认使用 76。	76	整数
camel.dataformat.base64.line-separator	要使用的行分隔符。默认使用新行字符(CRLF)。		字符串

Name	描述	默认值	类型
camel.dataformat .base64.url-safe	我们单独发出 '-' 和 '_'，而不是发出 '-' 和 '_'。urlSafe 只应用于编码操作。解码无缝处理这两种模式。默认为 false。	false	布尔值

ND

在 Spring DSL 中，您可以使用此标签配置数据格式：

```
<camelContext>
  <dataFormats>
    <!-- for a newline character (\n), use the HTML entity notation coupled with the ASCII code. -->
    <base64 lineSeparator="&#10;" id="base64withNewLine" />
    <base64 lineLength="64" id="base64withLineLength64" />
  </dataFormats>
  ...
</camelContext>
```

然后您可以稍后参考来使用它：

```
<route>
    <from uri="direct:startEncode" />
    <marshal ref="base64withLineLength64" />
    <to uri="mock:result" />
</route>
```

大多数时候，如果使用默认选项，则不需要声明数据格式。在这种情况下，您可以声明内联数据格式，如下所示。

43.3. MARSHAL

在本例中，我们将文件内容绑定到 base64 对象。

```
from("file://data.bin")
   .marshal().base64()
   .to("jms://myqueue");
```

在 Spring DSL 中：

```
<from uri="file://data.bin">
<marshal>
   <base64/>
</marshal>
<to uri="jms://myqueue"/>
```

43.4. UNMARSHAL

在本例中，我们在由新的Order 处理器处理前，从 JMS 队列到 byte[] 对象的载荷，再变为 byte[]。

```
from("jms://queue/order")
    .unmarshal().base64()
    .process("newOrder");
```

在 Spring DSL 中：

```
<from uri="jms://queue/order">
<unmarshal>
    <base64/>
</unmarshal>
<to uri="bean:newOrder"/>
```

43.5. 依赖项

要在 Camel 路由中使用 Base64，您需要添加对 **camel-base64** 的依赖，它实现了这个数据格式。

如果使用 Maven，您只需在 pom.xml 中添加以下内容：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-base64</artifactId>
  <version>x.x.x</version>  <!-- use the same version as your Camel core version -->
</dependency>
```

第 44 章 BEAN 组件

从 Camel 版本 1.0 开始提供

bean: 组件将 Bean 绑定到 Camel 消息交换。

44.1. URI 格式

```
bean:beanName[?options]
```

其中 **beanID** 可以是用于在 Registry 中查找 bean 的任何字符串

44.2. 选项

Bean 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
缓存 （高级）	如果启用，Camel 将缓存第一个 Registry 查找的结果。如果 Registry 中的 bean 定义为单例范围，则可以启用缓存。		布尔值
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Bean 端点使用 URI 语法进行配置：

```
bean:beanName
```

使用以下路径和查询参数：

44.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
beanName	**必需** 设置要调用的 bean 的名称		字符串

44.2.2. 查询参数(5 参数)：

Name	描述	默认值	类型
method (producer)	设置要在 bean 上调用的方法名称		字符串

Name	描述	默认值	类型
缓存 （高级）	如果启用，Camel 将缓存第一个 Registry 查找的结果。如果 Registry 中的 bean 定义为单例范围，则可以启用缓存。		布尔值
multiParameterArray (advanced)	**弃用** How to treat 处理从消息正文传递的参数；如果为 true，消息正文应该是参数数组。注意：此选项由 Camel 内部使用，不面向最终用户使用。弃用备注：此选项由 Camel 内部使用，不适用于最终用户使用。	false	布尔值
参数 (advanced)	用于在 bean 中配置附加属性		Map
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

您可以在 URI 中附加查询选项，格式为 **?option=value&option=value&...**

44.3. 使用

用于消耗消息的对象实例必须明确注册到 Registry。例如，如果您使用 Spring，则必须在 Spring 配置 **spring.xml**; 中定义 bean；或者如果您不使用 Spring，则在 JNDI 中注册 bean。

错误格式化宏： snippet: java.lang.IndexOutOfBoundsException: Index: 20, Size: 20

注册端点后，您可以构建 Camel 路由来用它来处理交换。

Bean: 端点不能定义为路由的输入；即您无法从它使用，您只能从某些入站消息 Endpoint 路由到 bean 端点作为输出。因此，请考虑使用 **direct:** 或 **queue:** 端点作为输入。

您可以使用 ProxyHelper 上的 **createProxy** （） 方法创建将生成 BeanExchanges 的代理并将其发送到任何端点：

和使用 Spring DSL 的同一路由：

```
<route>
  <from uri="direct:hello">
  <to uri="bean:bye"/>
</route>
```

44.4. BEAN 作为端点

Camel 还支持将 Bean 作为端点调用。在以下路由中：

发生了什么情况，当交换路由到 **myBean** Camel 时，将使用 Bean Binding 来调用 bean。
Bean 的源只是一个普通 POJO：

Camel 将使用 Bean Binding 来调用 **sayHello** 方法，方法是将 Exchange 的 In 正文转换为 **String** 类型，并将方法的输出存储在 Exchange Out 正文上。

44.5. JAVA DSL BEAN 语法

Java DSL 附带 Bean 组件的 syntactic sugar。您可以使用以下语法，而不是明确指定 bean 作为端点（例如 **to ("bean:beanName")**）：

```
// Send message to the bean endpoint
// and invoke method resolved using Bean Binding.
from("direct:start").beanRef("beanName");

// Send message to the bean endpoint
// and invoke given method.
from("direct:start").beanRef("beanName", "methodName");
```

您可以指定 bean 本身，而不是向 bean 传递名称（以便 Camel 将在 registry 中查找它）：

```
// Send message to the given bean instance.
from("direct:start").bean(new ExampleBean());

// Explicit selection of bean method to be invoked.
from("direct:start").bean(new ExampleBean(), "methodName");

// Camel will create the instance of bean and cache it for you.
from("direct:start").bean(ExampleBean.class);
```

44.6. BEAN BINDING

要调用的 bean 方法是如何被选择（如果没有通过 **method** 参数明确指定），以及从 Message 构建的参数值是如何由 Camel 中所有不同 Bean 集成机制使用的 Bean Binding 机制定义的。

44.7. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- 类 组件
- Bean Binding
- Bean 集成

第 45 章 BEANIO DATAFORMAT

从 Camel 版本 2.10 开始提供

BeanIO 数据格式使用 BeanIO 来处理扁平有效负载（如 XML、CSV、分隔或固定长度格式）。

BeanIO 使用 映射 XML 文件进行配置，该文件从扁平格式定义到对象(POJO)的映射。此映射文件是必需的。

45.1. 选项

BeanIO dataformat 支持 9 个选项，如下所列。

Name	默认值	Java 类型	描述
映射		**字符串**	BeanIO 映射文件。默认情况下，从 classpath 加载。您可以使用 file:、http: 或 classpath: 前缀，以表示从加载映射文件的位置。
streamName		**字符串**	要使用的流的名称。
ignoreUnidentifiedRecords	**false**	**布尔值**	是否忽略未识别的记录。
ignoreUnexpectedRecords	**false**	**布尔值**	是否忽略意外的记录。
ignoreInvalidRecords	**false**	**布尔值**	是否忽略无效的记录。
编码		**字符串**	要使用的 charset。默认情况下，JVM 平台的默认 charset。
beanReaderErrorHandlerType		**字符串**	在解析时，使用自定义 org.apache.camel.dataformat.beanio.BeanIOErrorHandler 作为错误处理程序。配置错误处理程序的完全限定类名称。请注意，当您使用自定义错误处理程序时，选项 ignoreUnidentifiedRecords, ignoreUnexpectedRecords, 和 ignoreInvalidRecords 可能没有被使用。
unmarshalSingleObject	**false**	**布尔值**	此选项控制是否将 unmarshal 作为对象列表或仅作为单个对象控制。前者是默认模式，后者仅用于特殊用例，其中 beanio 将 Camel 消息映射到单个 POJO Bean。
contentTypeHeader	**false**	**布尔值**	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

45.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.beanio.bean-reader-error-handler-type	在解析时，使用自定义 org.apache.camel.dataformat.beanio.BeanIOErrorHandler 作为错误处理程序。配置错误处理程序的完全限定类名称。请注意，当您使用自定义错误处理程序时，选项 ignoreUnidentifiedRecords, ignoreUnexpectedRecords, 和 ignoreInvalidRecords 可能没有被使用。		字符串
camel.dataformat.beanio.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.beanio.enabled	启用 beanio dataformat	true	布尔值
camel.dataformat.beanio.encoding	要使用的 charset。默认情况下，JVM 平台的默认 charset。		字符串
camel.dataformat.beanio.ignore-invalid-records	是否忽略无效的记录。	false	布尔值
camel.dataformat.beanio.ignore-unexpected-records	是否忽略意外的记录。	false	布尔值
camel.dataformat.beanio.ignore-unidentified-records	是否忽略未识别的记录。	false	布尔值
camel.dataformat.beanio.mapping	BeanIO 映射文件。默认情况下，从 classpath 加载。您可以使用 file:、http: 或 classpath: 前缀，以表示从加载映射文件的位置。		字符串
camel.dataformat.beanio.stream-name	要使用的流的名称。		字符串
camel.dataformat.beanio.unmarshal-single-object	此选项控制是否将 unmarshal 作为对象列表或仅作为单个对象控制。前者是默认模式，后者仅用于特殊用例，其中 beanio 将 Camel 消息映射到单个 POJO Bean。	false	布尔值

ND

45.3. 使用方法

映射文件的示例如下：

45.3.1. 使用 Java DSL

要使用 **BeanIODataFormat**，您需要使用映射文件以及流的名称配置数据格式。
在 Java DSL 中，这可按照如下所示完成。streamName 是 "employeeFile"。

然后我们有两个路由。第一个路由用于将 CSV 数据转换为 List<Employee> Java 对象。然后，我们分割，因此 mock 端点
接收每行的消息。

第二种路由用于反向操作，可将 List<Employee> 转换为 CSV 数据流。

CSV 数据可能如下所示：

45.3.2. 使用 XML DSL

要在 XML 中使用 BeanIO 数据格式，您需要使用 <beanio> XML 标签进行配置，如下所示。路由与上例类似。

45.4. 依赖项

要在 Camel 路由中使用 BeanIO，您需要添加对实现此数据格式的 **camel-beanio** 的依赖。

如果使用 Maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-beanio</artifactId>
  <version>2.10.0</version>
</dependency>
```

第 46 章 BEANSTALK 组件

从 Camel 版本 2.15 开始提供

camel-beanstalk 项目提供了一个 Camel 组件，用于对 Beanstalk 作业的作业检索和后处理。

您可以在 Beanstalk 协议 中找到 Beanstalk 作业生命周期的详细说明。

46.1. 依赖项

Maven 用户需要将以下依赖项添加到其 **pom.xml**中

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-beanstalk</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 **${camel-version}** 必须替换为 Camel 的实际版本(2.15.0 或更高版本)。

46.2. URI 格式

```
beanstalk://[host[:port]][/tube][?options]
```

您可以为要使用的 Beanstalk 默认值省略 **port** 或 **host** 和 port：("localhost" 和 11300)。如果省略 **tube**，则 Beanstalk 组件将使用 tube，名称为 "default"。

收听时，您可能想监视来自多个问题的作业。只需用加号来分隔它们，例如：

```
beanstalk://localhost:11300/tube1+tube2
```

Tube 名称将进行 URL 解码。因此，如果您的标题名称包含诸如 + 或？等特殊字符，您需要相应地对 URL 进行编码，或使用 RAW 语法，请参阅 此处的更多详细信息。

通过这种方式，当您将作业写入到 Beanstalk 中时，您无法指定多个 tubes。

46.3. BEANSTALK 选项

Beanstalk 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
connectionSettings Factory (common)	自定义 ConnectionSettingsFactory。指定用于连接 Beantalkd 的 ConnectionSettingsFactory。对于没有 Beantalkd 守护进程的单元测试非常有用（您可以在没有 Beantalkd 守护进程中进行模拟连接Settings）		ConnectionSettings Factory
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Beanstalk 端点使用 URI 语法进行配置：

```
beanstalk:connectionSettings
```

使用以下路径和查询参数：

46.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
connectionSettings	连接设置 host:port/tube		字符串

46.3.2. 查询参数(26 参数)：

Name	描述	默认值	类型
command (common)	将作业放入 Beanstalk 中。作业正文在 Camel 消息正文中指定。作业 ID 将在 beanstalk.jobId 消息标头中返回。delete, release, touch 或 bury 预期在消息标头 Beantalk.jobId 中的作业 ID。在 beanstalk.result 消息标头中返回操作的结果是 start 预期在消息正文中启动的作业数量，并返回实际在消息标头 Beantalk.result 中启动的作业数量。		BeanstalkCommand
jobDelay (common)	作业延迟（以秒为单位）。	0	int
jobPriority (common)	作业优先级.(0 是最高，请参阅 Beanstalk 协议)	1000	long
jobTimeToRun (common)	以秒为单位运行的作业时间。（当 0 时，beanstalkd 守护进程会将其提高至 1，请参阅 Beanstalk 协议。）	60	int
awaitJob (consumer)	是否在从 beanstalk 中处理作业前等待作业完成	true	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
OnFailure (consumer)	在处理失败时使用的命令。		BeanstalkCommand

Name	描述	默认值	类型
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
useBlockIO (consumer)	是否使用 blockIO。	true	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long

Name	描述	默认值	类型
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

46.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.beanstalk.connection-settings-factory	自定义 ConnectionSettingsFactory。指定用于连接 Beantalkd 的 ConnectionSettingsFactory。对于在没有 beanstalkd 守护进程的情况下进行单元测试（您可以模拟连接Settings）特别有用。选项是一个 org.apache.camel.component.beanstalk.ConnectionSettingsFactory 类型。		字符串
camel.component.beanstalk.enabled	启用 Beantalk 组件	true	布尔值
camel.component.beanstalk.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

生产者行为受 **command** 参数的影响，该参数告知如何使用作业，它可以是

消费者可以在保留作业或等待 Camel 路由处理进程后立即删除该作业。虽然第一种情况与"消息队列"类似，第二个则类似于"作业队列"。此行为由 **consumer.awaitJob** 参数控制，该参数默认等于 **true** （遵循 Beanstalkd nature）。

同步时，消费者调用在成功完成时 **删除**，并在失败时调用 **bury**。您可以通过在 URI 中指定 **consumer.onFailure** 参数来选择在失败时要执行的命令。它可以取 bur **y**、**delete** 或 **release** 的值。

有一个布尔值参数 **consumer.useBlockIO**，它对应于 netobservtalkClient 库中的同一参数。默认为 **true**。

在指定 **发行版本** 时请小心，因为失败的作业将立即可用，并且您的消费者将尝试再次获取它。您可以 **释放** 并指定 *jobDelay*。

beanstalk consumer 是一个 Scheduled Polling Consumer，这意味着您可以配置更多选项，如消费者应轮询的频率。如需了解更多详细信息，请参阅 Polling Consumer。

46.5. 消费者标头

消费者在 Exchange 消息中存储多个作业标头：

属性	类型	描述
beanstalk.jobId	long	作业 ID
beanstalk.tube	string	包含此作业的 tube 的名称
beanstalk.state	string	"ready" 或 "delayed" 或 "reserved" 或 "buried" （必须为 "reserved"）
beanstalk.priority	long	设置的优先级值
beanstalk.age	int	创建此作业的 put 命令的时间（以秒为单位）
beanstalk.time-left	int	服务器将该作业放入就绪队列中前的秒数
beanstalk.timeouts	int	此作业在保留过程中超时的次数
beanstalk.releases	int	客户端从保留中释放此作业的次数

属性	类型	描述
beanstalk.buries	int	此任务已禁止的次数
beanstalk.kicks	int	启动此作业的次数

46.6. 例子

此 Camel 组件可让您请求作业进行处理，并将其提供给 Beanstalkd 守护进程。我们的简单演示路由可能如下所示

```
from("beanstalk:testTube").
  log("Processing job #${property.beanstalk.jobId} with body ${in.body}").
  process(new Processor() {
    @Override
    public void process(Exchange exchange) {
      // try to make integer value out of body
      exchange.getIn().setBody( Integer.valueOf(exchange.getIn().getBody(classOf[String])) );
    }
  }).
  log("Parsed job #${property.beanstalk.jobId} to body ${in.body}");
```

```
from("timer:dig?period=30seconds").
  setBody(constant(10)).log("Kick ${in.body} buried/delayed tasks").
  to("beanstalk:testTube?command=kick");
```

在第一个路由中，我们在 tube "testTube"中侦听新作业。当它们到达时，我们尝试从消息正文解析整数值。如果成功，我们会记录日志并成功交换完成，使 Camel 组件从 Beanstalk 自动删除该作业。*取而代之，当无法解析作业数据时，交换失败，以及默认接收它的 Camel 组件，以便稍后进行处理，或者我们可能会手动检查失败的作业。*

因此，第二个路由定期请求 Beanstalk，从而向正常队列启动 10 个作业和/或延迟状态。

46.7. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 47 章 BEAN VALIDATOR 组件

从 Camel 版本 2.3 开始提供

Validator 组件使用 Java Bean Validation API (JSR 303)执行消息正文的 bean 验证。Camel 使用引用实施，即 Hibernate Validator。

Maven 用户需要将以下依赖项添加到这个组件的 **pom.xml** 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-bean-validator</artifactId>
  <version>x.y.z</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

47.1. URI 格式

```
bean-validator:label[?options]
```

or

```
bean-validator://label[?options]
```

其中 **label** 是描述端点的任意文本值。
您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

47.2. URI 选项

Bean Validator 组件没有选项。

Bean Validator 端点使用 URI 语法进行配置：

```
bean-validator:label
```

使用以下路径和查询参数：

47.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
label	**必需的** where 标签是描述端点的任意文本值		字符串

47.2.2. 查询参数(6 参数)：

Name	描述	默认值	类型
constraintValidatorFactory (producer)	使用自定义 ConstraintValidatorFactory		ConstraintValidator Factory
group (producer)	使用自定义验证组	javax.validation.groups.Default	字符串
messageInterpolator (producer)	使用自定义 MessageInterpolator		MessageInterpolator
traversableResolver (producer)	使用自定义 TraversableResolver		TraversableResolver
validationProviderResolver (producer)	使用自定义 ValidationProviderResolver		ValidationProviderResolver
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

47.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.bean-validator.enabled	启用 bean-validator 组件	true	布尔值
camel.component.bean-validator.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

47.4. OSGI 部署

要在 OSGi 环境中使用 Hibernate Validator，请使用专用的 ***ValidationProviderResolver*** *实施，就像* ***org.apache.camel.component.bean.validator.HibernateValidationProviderResolver*** *一样。以下片段演示了此方法。请记住，您可以使用从 Camel 2.13.0 开始的* ***HibernateValidationProviderResolver****。*

使用 HibernateValidationProviderResolver

```
from("direct:test").
  to("bean-validator://ValidationProviderResolverTest?
validationProviderResolver=#myValidationProviderResolver");

...

<bean id="myValidationProviderResolver"
class="org.apache.camel.component.bean.validator.HibernateValidationProviderResolver"/>
```

如果没有定义自定义 **ValidationProviderResolver**，且验证器组件已部署到 OSGi 环境中，则会自动使用 **HibernateValidationProviderResolver**。

47.5. 示例

假设有一个带有以下注解的 java bean

Car.java

```
public class Car {

    @NotNull
    private String manufacturer;

    @NotNull
    @Size(min = 5, max = 14, groups = OptionalChecks.class)
    private String licensePlate;

    // getter and setter
}
```

以及自定义验证组的接口定义

OptionalChecks.java

```
public interface OptionalChecks {
}
```

使用以下 Camel 路由，只有属性 manufacturer 和 licensePlate 的 **@NotNull** 限制进行验证(Camel 使用默认组 **javax.validation.groups.Default**)。

```
from("direct:start")
.to("bean-validator://x")
.to("mock:end")
```

如果要从 **OptionalChecks** 组检查约束，您必须定义如下路由

```
from("direct:start")
.to("bean-validator://x?group=OptionalChecks")
.to("mock:end")
```

如果要检查这两个组中的限制，您必须首先定义新接口

AllChecks.java

-

```
@GroupSequence({Default.class, OptionalChecks.class})
public interface AllChecks {
}
```

然后您的路由定义应如下所示

```
from("direct:start")
.to("bean-validator://x?group=AllChecks")
.to("mock:end")
```

您必须提供自己的消息插入器，即遍历解析器和约束验证器工厂，您必须编写类似此的路由。

```
<bean id="myMessageInterpolator" class="my.ConstraintValidatorFactory" />
<bean id="myTraversableResolver" class="my.TraversableResolver" />
<bean id="myConstraintValidatorFactory" class="my.ConstraintValidatorFactory" />

from("direct:start")
.to("bean-validator://x?group=AllChecks&messageInterpolator=#myMessageInterpolator
&traversableResolver=#myTraversableResolver&constraintValidatorFactory=#myConstraintValidatorFac
tory")
.to("mock:end")
```

您还可以将限制描述为 XML，而不是 Java 注解。在这种情况下，您必须提供文件 **META-INF/validation.xml**，该文件可能类似如下

validation.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<validation-config
    xmlns="http://jboss.org/xml/ns/javax/validation/configuration"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://jboss.org/xml/ns/javax/validation/configuration">
    <default-provider>org.hibernate.validator.HibernateValidator</default-provider>
    <message-
interpolator>org.hibernate.validator.engine.ResourceBundleMessageInterpolator</message-
interpolator>
    <traversable-
resolver>org.hibernate.validator.engine.resolver.DefaultTraversableResolver</traversable-resolver>
    <constraint-validator-
factory>org.hibernate.validator.engine.ConstraintValidatorFactoryImpl</constraint-validator-factory>

    <constraint-mapping>/constraints-car.xml</constraint-mapping>
</validation-config>
```

和 **constraints-car.xml** 文件

constraints-car.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<constraint-mappings xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://jboss.org/xml/ns/javax/validation/mapping validation-mapping-1.0.xsd"
    xmlns="http://jboss.org/xml/ns/javax/validation/mapping">
    <default-package>org.apache.camel.component.bean.validator</default-package>
```

```
    <bean class="CarWithoutAnnotations" ignore-annotations="true">
      <field name="manufacturer">
        <constraint annotation="javax.validation.constraints.NotNull" />
      </field>

      <field name="licensePlate">
        <constraint annotation="javax.validation.constraints.NotNull" />

        <constraint annotation="javax.validation.constraints.Size">
          <groups>
            <value>org.apache.camel.component.bean.validator.OptionalChecks</value>
          </groups>
          <element name="min">5</element>
          <element name="max">14</element>
        </constraint>
      </field>
    </bean>
</constraint-mappings>
```

*以下是 **OrderedChecks** 可以是 https://github.com/apache/camel/blob/master/components/camel-bean-validator/src/test/java/org/apache/camel/component/bean/validator/OrderedChecks.java的示例路由定义的XML 语法。*

请注意，正文应包含要验证的类实例。

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
    http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
    http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd">

    <camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
        <route>
            <from uri="direct:start"/>
            <to uri="bean-validator://x?
group=org.apache.camel.component.bean.validator.OrderedChecks"/>
        </route>
    </camelContext>
</beans>
```

47.6. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 48 章 BINDING COMPONENT （已弃用）

从 Camel 版本 2.11 开始提供

在 Camel 术语中，绑定 是一种将端点嵌套在合同中的一种方式，如数据格式、内容增强 或验证步骤。绑定是完全可选的，您可以选择在任何 camel 端点中使用它们。

绑定通过为 Camel 等各种技术 添加服务合同的工作而激发。Camel Bindings 提供了通过 Camel 框架本身的合同来嵌套 Camel 端点，而不是在 SCA 中包装 Camel 端点；因此您可以在任何 Camel 路由内轻松使用。

48.1. 选项

Binding 组件没有选项。

Binding 端点使用 URI 语法进行配置：

```
binding:bindingName:delegateUri
```

使用以下路径和查询参数：

48.1.1. 路径参数(2 参数)：

Name	描述	默认值	类型
bindingName	Camel registry 中要查找的绑定 **必需** 名称。		字符串
delegateUri	**需要** Uri of the delegate endpoint。		字符串

48.1.2. 查询参数(4 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外，该处理程序记录在 WARN/ERROR 级别并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，该例外记录在 WARN/ERROR 级别并忽略。		ExceptionHandler
exchangePattern (consumer)	在创建交换时设置默认交换模式。		ExchangePattern

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

48.2. 使用绑定

绑定目前是一个定义合同的 bean （虽然我们希望向 Camel DSL 中添加绑定）。

定义绑定端点（即与绑定的端点）有几种方法。

48.3. 使用绑定 URI

*您可以使用 **binding:nameOfBinding:** 前缀任何端点 URI，其中 nameOfBinding 是 registry 中的 Binding bean 的名称。*

```
from("binding:jaxb:activemq:myQueue").to("binding:jaxb:activemq:anotherQueue")
```

在这里，我们使用"jaxb"绑定，例如，使用 JAXB 数据格式将 JAXB 数据格式用于 marshal 和 unmarshal 消息。

48.4. 使用 BINDINGCOMPONENT

有一个名为 BindingComponent 的组件，它可以通过依赖项注入在 Registry 中配置，允许创建已绑定到某些绑定的端点。

例如，如果您使用如下代码在 registry 中注册了一个名为 "jsonmq" 的新组件

```
JacksonDataFormat format = new JacksonDataFormat(MyBean.class);
context.bind("jsonmq", new BindingComponent(new DataFormatBinding(format), "activemq:foo."));
```

然后，您可以像任何其他端点一样使用端点。

```
from("jsonmq:myQueue").to("jsonmq:anotherQueue")
```

使用 queueus "foo.myQueue" 和 "foo.anotherQueue"，并使用给定的 Jackson Data Format to marshal on 和 off 队列。

48.5. 使用绑定的时间

如果您在一个路由中只使用端点一次；绑定实际上可能比直接使用 'raw' 端点并且正常使用显式 marshalling 和 validation 来更复杂。

但是，当将多个路由组合在一起时，绑定可以帮助将多个路由用作配置输入和输出端点的"模板"；绑定随后提供了将合同和端点嵌套在一起的 nice 方法。

绑定的另一个好用例是，如果您使用多个使用相同绑定的端点，而不是始终提到特定的数据格式或验证规则，您只需使用 BindingComponent 来将端点嵌套在您选择的绑定中。

因此，绑定实际上是一个组成工具；只有在有意义时，只能使用它们，除非有大量路由或端点，否则可能并不值得使用。

第 49 章 BINDY DATAFORMAT

从 Camel 版本 2.0 开始提供

此组件的目标是允许解析/绑定非结构化数据（或者更精确的非 XML 数据）到使用注解定义的绑定映射的 Java Beans。使用 Bindy，您可以从源（如）绑定数据：

- CSV 记录，
- 固定记录，
- FIX 消息，
- 或者几乎任何其他非结构化数据

到一个或多个 Plain Old Java 对象(POJO)。bindy 根据 java 属性的类型转换数据。在某些情况下，POJO 可以与一对多关系连接在一起。此外，对于 Date, Double, Float, Integer, Short, Long 和 BigDecimal 等数据类型，您可以在属性格式化期间提供要应用的模式。

对于 BigDecimal 号，您还可以定义精度，以及十进制或分组分隔符。

类型	格式类型	特征示例	Link
Date	DateFormat	**dd-MM-yyyy**	http://java.sun.com/j2se/1.5.0/docs/api/java/text/SimpleDateFormat.html
十进制*	DecimalFormat	**..##**	http://java.sun.com/j2se/1.5.0/docs/api/java/text/DecimalFormat.html

decimal* = Double, Integer, Float, Short, Long

> *Format supported*

第一个版本只支持逗号分隔的值字段和键值对字段（例如：FIX 消息）。

要使用 camel-bindy，您必须首先在软件包（如 com.acme.model）和每个模型类（如 Order, Client, Instrument, ...）中定义您的模型，将所需的注解（这里介绍）添加到 Class 或 字段。

> *Multiple models*

如果您使用多个模型，则必须将每个模型放在它自己的软件包中，以防止无法预计的结果。

现在，从 **Camel 2.16** 开始，这已不再是这种情况，因为您可以在同一个软件包中安全地有多个模型，因为您使用类名称而不是软件包名称配置绑定。

49.1. 选项

Bindy dataformat 支持 5 个选项，如下所列。

Name	默认值	Java 类型	描述
type		**Bindy Type**	是否使用 csv、fixd 或 key value 对模式。根据所选的 dataformat，默认值为 Csv 或 KeyValue。
classType		**字符串**	要使用的模型类名称。
locale		**字符串**	要配置要使用的默认区域设置，如 us 用于单元状态。使用 JVM 平台默认区域设置，然后使用名称 default
unwrapSingleInstance	**true**	**布尔值**	当 unmarshalling 应该解封并返回时，而不是嵌套在 java.util.List 中。
contentTypeHeader	**false**	**布尔值**	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

49.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 18 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.bindy-csv.class-type	要使用的模型类名称。		字符串
camel.dataformat.bindy-csv.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.bindy-csv.enabled	启用 bindy-csv dataformat	true	布尔值
camel.dataformat.bindy-csv.locale	要配置要使用的默认区域设置，如 us 用于单元状态。使用 JVM 平台默认区域设置，然后使用名称 default		字符串
camel.dataformat.bindy-csv.type	是否使用 csv、fixd 或 key value 对模式。		BindyType
camel.dataformat.bindy-csv.unwrap-single-instance	当 unmarshalling 应该解封并返回时，而不是嵌套在 java.util.List 中。	true	布尔值

Name	描述	默认值	类型
camel.dataformat.bindy-fixed.class-type	要使用的模型类名称。		字符串
camel.dataformat.bindy-fixed.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.bindy-fixed.enabled	启用 bindy-fixed dataformat	true	布尔值
camel.dataformat.bindy-fixed.locale	要配置要使用的默认区域设置，如 us 用于单元状态。使用 JVM 平台默认区域设置，然后使用名称 default		字符串
camel.dataformat.bindy-fixed.type	是否使用 csv、fixd 或 key value 对模式。		BindyType
camel.dataformat.bindy-fixed.unwrap-single-instance	当 unmarshalling 应该解封并返回时，而不是嵌套在 java.util.List 中。	true	布尔值
camel.dataformat.bindy-kvp.class-type	要使用的模型类名称。		字符串
camel.dataformat.bindy-kvp.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.bindy-kvp.enabled	启用 bindy-kvp dataformat	true	布尔值
camel.dataformat.bindy-kvp.locale	要配置要使用的默认区域设置，如 us 用于单元状态。使用 JVM 平台默认区域设置，然后使用名称 default		字符串
camel.dataformat.bindy-kvp.type	是否使用 csv、fixd 或 key value 对模式。		BindyType
camel.dataformat.bindy-kvp.unwrap-single-instance	当 unmarshalling 应该解封并返回时，而不是嵌套在 java.util.List 中。	true	布尔值

ND

49.3. 注解

创建的注解允许将不同的模型概念映射到 POJO，例如：

- *记录类型(csv，键值对（如 FIX 消息)、固定长度 ...），*
- *链接（到另一个对象中的链接对象），*
- *DataField 及其属性(int, type, ...),*
- *KeyValuePairField （用于 key = 值格式，如我们在 FIX 财务消息中一样），*
- *部分（识别标题、正文和页脚部分）、*
- *OneToMany,*
- *BindyConverter （自 2.18.0 开始），*
- *FormatFactories （自 2.18.0 开始）*

本节将描述它们：

49.4. 1.CSVRECORD

CsvRecord 注解用于标识模型的根类。它表示一个记录 = CSV 文件一行，并可链接到多个子级类。

注解名称	记录类型	级别
CsvRecord	csv	类

参数名称	type	info
分隔符	string	必需 - 可以是 ',' 或 ';' 或 'anything'。唯一支持空格字符是 tab (\t)。不支持其他空格字符（空格）。这个值被解释为正则表达式。如果要使用在正则表达式中有一个特殊含义的符号，例如 '|' 符号，而不是屏蔽它，如 '|'
skipFirstLine	布尔值	可选 - 默认值 = false - 允许跳过 CSV 文件的第一行
crlf	string	可选 - 可能的值 = WINDOWS,UNIX,MAC, 或 custom; 默认值。WINDOWS - 允许定义要使用的回车符。如果您指定了前面列出的三个值，则您输入的值(custom)将用作 CRLF 字符。

参数名称	type	info
generateHeaderColumns	布尔值	可选 - default value = false - 用来生成 CSV 生成的标头列
autospanLine	布尔值	**Camel 2.13/2.12.2:** 可选 - 默认值 = false - 如果启用，则最后一列将自动生成到行尾，例如，如果为注释，这将允许行包含所有字符，也是分隔符字符。
isOrdered	布尔值	可选 - default value = false - 允许在生成 CSV 时更改字段的顺序
quote	字符串	**Camel 2.8.3/2.9:** 选项 - 允许在生成 CSV 时指定字段的引号字符。此注解与模型的根类关联，必须被声明一次。
quoting	布尔值	*Camel 2.11:*optional - default value = false - 表示在生成 CSV 时，值（和标头）是否必须加引号。
endWithLineBreak	布尔值	**Camel 2.21 ：** 可选 - 默认值 = true - 表示 CSV 生成的文件是否应以行中断结尾。

case 1 : separator = ','

用于隔离 CSV 记录中的字段的分隔符为 ',' ：

> *10, J, Pauline, M, XD12345678, Fortis Dynamic 15/15, 2500, USD,08-01-2009*

```
@CsvRecord( separator = "," )
public Class Order {

}
```

case 2 : separator = ';'

与前面的问题单进行比较，这里的分隔符为 ';' 而不是 ',' ：

10; j; Pauline; M; XD12345678; Fortis Dynamic 15/15; 2500; USD; 08-01-2009

```
@CsvRecord( separator = ";" )
public Class Order {

}
```

case 3 : separator = '|'

与前面的问题单进行比较，这里的分隔符为 '|' 而不是 ';' ：

```
10| J| Pauline| M| XD12345678| Fortis Dynamic 15/15| 2500| USD|
08-01-2009
```

```
@CsvRecord( separator = "\\|" )
public Class Order {

}
```

case 4 : separator = '\",\"'

适用于 Camel 2.8.2 或更早的版本

当要解析 CSV 记录的字段包含 ',' 或 ';' （也用作分隔符）时，我们会找到另一个策略
告知 camel 绑定如何处理这个问题单。要使用逗号定义包含数据的字段，您可以使用 simple 或双引号
作为分隔符（例如：'10', 'Street 10, NY', 'USA' 或 "10", "Street 10, NYA"）。
Remark : 在这种情况下，行中的第一个和最后一个字符是简单或双引号将按绑定删除

```
"10","J","Pauline"," M","XD12345678","Fortis Dynamic 15,15"
2500","USD","08-01-2009"
```

```
@CsvRecord( separator = "\",\"" )
public Class Order {

}
```

*从 **Camel 2.8.3/2.9 或从** bindy 将自动检测记录是否用单引号或双引号括起，并在从 CSV 到对象时自动删除这些引号。因此，**不要在** 分隔符中包含引号，但非常简单，如下所示：*

```
"10","J","Pauline"," M","XD12345678","Fortis Dynamic 15,15"
2500","USD","08-01-2009"
```

```
@CsvRecord( separator = "," )
public Class Order {

}
```

*请注意，如果您想要从对象到 CSV 并使用引号，则需要使用 @CsvRecord 上的 **quote** 属性来指定要使用的引号字符，如下所示：*

```
@CsvRecord( separator = ",", quote = "\"" )
public Class Order {

}
```

case 5 : separator & skipfirstline

当客户端希望在文件的第一行中包括该功能时，其数据字段的名称很有意义：

order id, client id, first name, last name, isin code, instrument name, number, currency, date

要告知绑定在解析过程中必须跳过此第一行，我们使用属性 ：

```
@CsvRecord(separator = ",", skipFirstLine = true)
```

```
public Class Order {

}
```

case 6 : generateHeaderColumns

要在生成的 CSV 的第一行中添加，在注解中必须将属性 generateHeaderColumns 设置为 true，如下所示：

```
@CsvRecord( generateHeaderColumns = true )
public Class Order {

}
```

因此，在 unmarshaling 过程中绑定将生成 CSV，如下所示：

order id, client id, first name, last name, isin code, instrument name, number, currency, date

```
10, J, Pauline, M, XD12345678, Fortis Dynamic 15/15, 2500, USD,08-01-2009
```

case 7 : carriage return

如果运行 camel-bindy 的平台不是 Windows 但 Macintosh 或 Unix 的平台，而您可以更改 crlf 属性，如下所示。有三个值可用：WINDOWS、UNIX 或 MAC

```
@CsvRecord(separator = ",", crlf="MAC")
public Class Order {

}
```

另外，如果出于某种原因需要添加不同的行尾字符，您可以选择使用 crlf 参数来指定它。在以下示例中，我们可以使用逗号结尾行，后跟换行符：

```
@CsvRecord(separator = ",", crlf=",\n")
public Class Order {

}
```

case 8 : isOrdered

有时，从模型创建 CSV 记录时应遵循的顺序与解析过程中使用的顺序不同。然后，我们可以使用属性 isOrdered = true 来指明此属性与 DataField 注解的属性 'position' 结合使用。

```
@CsvRecord(isOrdered = true)
public Class Order {

  @DataField(pos = 1, position = 11)
  private int orderNr;

  @DataField(pos = 2, position = 10)
  private String clientNr;

}
```

Remark : pos 用于解析文件，而位置用于生成 CSV 时定位

49.5. 2.LINK

link 注释允许将对象链接到一起。

注解名称	记录类型	级别
Link	all	类和属性

参数名称	type	info
linkType	LinkType	可选 - 默认情况下，值为 LinkType.oneToOne - 因此您没有义务提及它

只允许一对一关系。

例如：如果模型类客户端链接到 Order 类，请在 Order 类中使用注解链接，如下所示：

属性链接

```
@CsvRecord(separator = ",")
public class Order {

    @DataField(pos = 1)
    private int orderNr;

    @Link
    private Client client;
}
```

对于类客户端，以及：

类链接

```
@Link
public class Client {

}
```

49.6. 3.DATAFIELD

DataField 注解定义字段的属性。每个 datafield 都由其在记录中的位置、类型（字符串、int、date、...）以及可选的模式来标识

注解名称	记录类型	级别
DataField	all	属性

参数名称	type	info
pos	int	Mandatory - **字段的输入** 位置。从 1 到 ... 开始的数字，请参阅 position 参数。
pattern	string	可选 - 默认值 = "" - 将用于格式化 Decimal, Date,
length	int	可选 - 代表字段的固定长度格式长度
精度	int	可选 - 代表在 Decimal 数字被格式化/解析时使用的精度
pattern	string	可选 - 默认值 = "" - 由 Java 格式器使用（例如，SimpleDateFormat）来格式化/验证数据。如果使用模式，则建议在绑定数据格式上设置区域设置。设置为一个已知的区域设置，如 "us"，或使用 "default" 来使用平台默认区域设置。请注意，"default"需要 Camel 2.14/2.13.3/2.12.5。
position	int	可选 - 当 CSV 中字段的位置（输出消息）必须与输入位置不同时，必须使用。请参阅 pos 参数。
required	布尔值	可选 - 默认值 = "false"
trim	布尔值	可选 - 默认值 = "false"
defaultValue	string	**Camel 2.10:** 可选 - 默认值 = "" - 当对应的 CSV 字段为空/未可用时定义字段的默认值
impliedDecimalSeparator	布尔值	**Camel 2.11:** 可选 - 默认值 = "false" - 代表在指定位置有一个十进制点表示
lengthPos	int	**Camel 2.11**: 可选 - 可用于识别固定长度记录中的 data 字段，该字段定义此字段的固定长度
对齐	string	可选 - 默认值 = "R" - 将文本设置为右或左在固定长度字段中。使用值 'R' 或 'L'
delimiter	string	**Camel 2.11:** 可选 - 可用于在固定长度记录中分离变量长度字段的末尾

问题单 1：pos

这个参数/属性代表 csv 记录中字段的位置

position

```
@CsvRecord(separator = ",")
public class Order {

    @DataField(pos = 1)
    private int orderNr;

    @DataField(pos = 5)
    private String isinCode;

}
```

如本例中所示，位置从 '1' 开始，但将继续在类 Order 中的 '5' 上。类客户端中定义了从 '2' 到 '4' 的数字（在此后参阅此）。

position 继续在另一个模型类中

```
public class Client {

    @DataField(pos = 2)
    private String clientNr;

    @DataField(pos = 3)
    private String firstName;

    @DataField(pos = 4)
    private String lastName;
}
```

case 2 : pattern

该模式可以增强或验证您的数据格式

pattern

```
@CsvRecord(separator = ",")
public class Order {

    @DataField(pos = 1)
    private int orderNr;

    @DataField(pos = 5)
    private String isinCode;

    @DataField(name = "Name", pos = 6)
    private String instrumentName;

    @DataField(pos = 7, precision = 2)
    private BigDecimal amount;

    @DataField(pos = 8)
    private String currency;
```

```
    // pattern used during parsing or when the date is created
    @DataField(pos = 9, pattern = "dd-MM-yyyy")
    private Date orderDate;
}
```

case 3 : 精度

当您要定义数字的十进制部分时，精度非常有用

精度

```
@CsvRecord(separator = ",")
public class Order {

    @DataField(pos = 1)
    private int orderNr;

    @Link
    private Client client;

    @DataField(pos = 5)
    private String isinCode;

    @DataField(name = "Name", pos = 6)
    private String instrumentName;

    @DataField(pos = 7, precision = 2)
    private BigDecimal amount;

    @DataField(pos = 8)
    private String currency;

    @DataField(pos = 9, pattern = "dd-MM-yyyy")
    private Date orderDate;
}
```

问题单 4：位置在输出中有所不同

position 属性将告知如何将字段放在生成的 CSV 记录中。默认情况下，使用的位置与属性 'pos' 定义的位置对应。如果位置不同（这意味着我们有一个 asymetric 进程与来自 unmarshaling 分离）的比较，但我们可以使用"位置"来指示这一点。

下面是一个示例

位置在输出中有所不同

```
@CsvRecord(separator = ",", isOrdered = true)
public class Order {

    // Positions of the fields start from 1 and not from 0

    @DataField(pos = 1, position = 11)
    private int orderNr;

    @DataField(pos = 2, position = 10)
```

```
    private String clientNr;

    @DataField(pos = 3, position = 9)
    private String firstName;

    @DataField(pos = 4, position = 8)
    private String lastName;

    @DataField(pos = 5, position = 7)
    private String instrumentCode;

    @DataField(pos = 6, position = 6)
    private String instrumentNumber;
}
```

注释 @DataField 的此属性必须与注释 @CsvRecord 的 attribute isOrdered = true 结合使用

case 5 : required

如果一个字段是必须的，只需使用 'required' setted to true

必填

```
@CsvRecord(separator = ",")
public class Order {

    @DataField(pos = 1)
    private int orderNr;

    @DataField(pos = 2, required = true)
    private String clientNr;

    @DataField(pos = 3, required = true)
    private String firstName;

    @DataField(pos = 4, required = true)
    private String lastName;
}
```

如果记录中没有此字段，解析器会引发错误，其中包含以下信息：

缺少一些字段（可选或强制），行：

case 6 : trim

如果某个字段在处理前带有前应删除的前导和/或尾随空格，只需使用属性 'trim' setted 为 true

Trim

```
@CsvRecord(separator = ",")
public class Order {

    @DataField(pos = 1, trim = true)
    private int orderNr;

    @DataField(pos = 2, trim = true)
```

```
    private Integer clientNr;

    @DataField(pos = 3, required = true)
    private String firstName;

    @DataField(pos = 4)
    private String lastName;
}
```

case 7 : defaultValue

如果没有定义字段，则使用 defaultValue 属性所示的值

默认值

```
@CsvRecord(separator = ",")
public class Order {

    @DataField(pos = 1)
    private int orderNr;

    @DataField(pos = 2)
    private Integer clientNr;

    @DataField(pos = 3, required = true)
    private String firstName;

    @DataField(pos = 4, defaultValue = "Barin")
    private String lastName;
}
```

此属性仅适用于可选字段。

49.7. 4.FIXEDLENGTHRECORD

FixedLengthRecord 注解用于识别模型的根类。它代表了一个记录，即 "一个文件/message 行，其中包含数据固定长度格式"，并可链接到多个子模型类。这个格式非常重要，因为字段的数据可以与右侧或左侧保持一致。

当数据的大小没有完全填写字段的长度时，我们可以添加"平板"字符。

注解名称	记录类型	级别
FixedLengthRecord	已修复	类

参数名称	type	必填	默认值	info
countGrapheme	布尔值		false	表示如何计算收费。

参数名称	type	必填	默认值	info
CRLF	string		WINDOWS	用于在每个记录后添加回车的字符（可选）。可能的值有：WINDOWS、UNIX、MAC 或自定义。这个选项仅在 marshalling 期间使用，unmarshalling 使用系统默认 JDK 提供的行分隔符，除非自定义了 eol。
EOL	string			在 unmarshalling 时，要在每个记录后处理行尾的字符（可选 - default = ""），这有助于使用默认 JDK 提供的行分隔符，除非提供任何其他行分隔符）。这个选项只在 unmarshalling 期间使用，其中 marshalling 使用系统默认提供的行分隔符为"WINDOWS"，除非提供了任何其他值。
footer	类		void	表示此类型的记录可能后跟一个文件末尾的一个页脚记录。
header				表示此类型的记录前面可能在文件的开头有一个标题记录。
ignoreMissingChars				指明是否忽略了太短行
ignoreTrailingChars				表示当 unmarshalling / 解析时，可以忽略除最后一个映射的文件以外的字符。此注解与模型的根类关联，必须被声明一次。
length	int		0	记录的固定长度（字符数）。这意味着记录始终是默认使用 {""paddingChar （）} 的 padded。
name	字符串			描述记录的名称（可选）。
paddingChar	char			使用 pad 进行字符.
skipFooter	布尔值		false	配置数据格式，以跳过页脚记录的 marshalling / unmarshalling，在主记录上配置此参数（例如，而不是标头或页脚）。
skipHeader	布尔值		false	配置数据格式，以跳过标题记录的 / unmarshalling。在主记录上配置此参数（例如，而不是标头或页脚）。

记录可能不是标头/页脚和主固定长度记录。

问题单 1：简单固定长度记录

这个简单示例演示了如何设计模型来解析/格式化固定消息。

> *10A9PaulineMISINXD12345678BUYShare2500.45USD01-08-2009*

fixed-simple

```
@FixedLengthRecord(length=54, paddingChar=' ')
public static class Order {

    @DataField(pos = 1, length=2)
    private int orderNr;

    @DataField(pos = 3, length=2)
    private String clientNr;

    @DataField(pos = 5, length=7)
    private String firstName;

    @DataField(pos = 12, length=1, align="L")
    private String lastName;

    @DataField(pos = 13, length=4)
    private String instrumentCode;

    @DataField(pos = 17, length=10)
    private String instrumentNumber;

    @DataField(pos = 27, length=3)
    private String orderType;

    @DataField(pos = 30, length=5)
    private String instrumentType;

    @DataField(pos = 35, precision = 2, length=7)
    private BigDecimal amount;

    @DataField(pos = 42, length=3)
    private String currency;

    @DataField(pos = 45, length=10, pattern = "dd-MM-yyyy")
    private Date orderDate;
}
```

问题单 2：使用对齐和填充修复长度记录

这更详细的示例演示了如何为字段定义对齐以及如何分配 padding 字符（本例中为 ':

```
10A9 PaulineM ISINXD12345678BUYShare2500.45USD01-08-2009
```

fixed-padding-align

```
@FixedLengthRecord(length=60, paddingChar=' ')
public static class Order {

    @DataField(pos = 1, length=2)
    private int orderNr;

    @DataField(pos = 3, length=2)
    private String clientNr;
```

```
    @DataField(pos = 5, length=9)
    private String firstName;

    @DataField(pos = 14, length=5, align="L")   // align text to the LEFT zone of the block
    private String lastName;

    @DataField(pos = 19, length=4)
    private String instrumentCode;

    @DataField(pos = 23, length=10)
    private String instrumentNumber;

    @DataField(pos = 33, length=3)
    private String orderType;

    @DataField(pos = 36, length=5)
    private String instrumentType;

    @DataField(pos = 41, precision = 2, length=7)
    private BigDecimal amount;

    @DataField(pos = 48, length=3)
    private String currency;

    @DataField(pos = 51, length=10, pattern = "dd-MM-yyyy")
    private Date orderDate;
}
```

case 3：字段填充

*有时，为记录定义的默认 padding 不能应用于字段，因为我们有一个数字格式，我们将希望使用 '0' 而不是 ''。在这种情况下，在模型中，您可以使用 **@DataField** 上的属性 **paddingChar** 来设置这个值。*

```
10A9 PaulineM ISINXD12345678BUYShare000002500.45USD01-08-2009
```

fixed-padding-field

```
@FixedLengthRecord(length = 65, paddingChar = ' ')
public static class Order {

    @DataField(pos = 1, length = 2)
    private int orderNr;

    @DataField(pos = 3, length = 2)
    private String clientNr;

    @DataField(pos = 5, length = 9)
    private String firstName;

    @DataField(pos = 14, length = 5, align = "L")
    private String lastName;

    @DataField(pos = 19, length = 4)
    private String instrumentCode;
```

```
    @DataField(pos = 23, length = 10)
    private String instrumentNumber;

    @DataField(pos = 33, length = 3)
    private String orderType;

    @DataField(pos = 36, length = 5)
    private String instrumentType;

    @DataField(pos = 41, precision = 2, length = 12, paddingChar = '0')
    private BigDecimal amount;

    @DataField(pos = 53, length = 3)
    private String currency;

    @DataField(pos = 56, length = 10, pattern = "dd-MM-yyyy")
    private Date orderDate;
}
```

问题单 4：使用分隔符修复长度记录

固定长度记录有时会在记录中有分隔的内容。在以下示例中，firstName 和 lastName 字段用以下示例中的 ^ 字符分隔：

```
10A9Pauline^M^ISINXD12345678BUYShare000002500.45USD01-08-2009
```

fixed-delimited

```
@FixedLengthRecord
public static class Order {

    @DataField(pos = 1, length = 2)
    private int orderNr;

    @DataField(pos = 2, length = 2)
    private String clientNr;

    @DataField(pos = 3, delimiter = "^")
    private String firstName;

    @DataField(pos = 4, delimiter = "^")
    private String lastName;

    @DataField(pos = 5, length = 4)
    private String instrumentCode;

    @DataField(pos = 6, length = 10)
    private String instrumentNumber;

    @DataField(pos = 7, length = 3)
    private String orderType;

    @DataField(pos = 8, length = 5)
    private String instrumentType;
```

```
    @DataField(pos = 9, precision = 2, length = 12, paddingChar = '0')
    private BigDecimal amount;

    @DataField(pos = 10, length = 3)
    private String currency;

    @DataField(pos = 11, length = 10, pattern = "dd-MM-yyyy")
    private Date orderDate;
}
```

可以使用 ordinal 或 sequential 值而不是精确列号来定义固定长度记录中的 ***pos*** *值。*

问题单 5 ：使用记录定义字段长度修复长度记录

有时，固定长度记录可能包含一个字段，用于定义同一记录中另一个字段的预期长度。在以下示例中，inst ***Number 字段值*** *的长度由记录中的* ***instrumentNumberLen*** *字段的值定义。*

```
10A9Pauline^M^ISIN10XD12345678BUYShare000002500.45USD01-08-2009
```

fixed-delimited

```
@FixedLengthRecord
public static class Order {

    @DataField(pos = 1, length = 2)
    private int orderNr;

    @DataField(pos = 2, length = 2)
    private String clientNr;

    @DataField(pos = 3, delimiter = "^")
    private String firstName;

    @DataField(pos = 4, delimiter = "^")
    private String lastName;

    @DataField(pos = 5, length = 4)
    private String instrumentCode;

    @DataField(pos = 6, length = 2, align = "R", paddingChar = '0')
    private int instrumentNumberLen;

    @DataField(pos = 7, lengthPos=6)
    private String instrumentNumber;

    @DataField(pos = 8, length = 3)
    private String orderType;

    @DataField(pos = 9, length = 5)
    private String instrumentType;

    @DataField(pos = 10, precision = 2, length = 12, paddingChar = '0')
    private BigDecimal amount;
```

```
    @DataField(pos = 11, length = 3)
    private String currency;

    @DataField(pos = 12, length = 10, pattern = "dd-MM-yyyy")
    private Date orderDate;
}
```

case 6 ：使用标头和页脚修复长度记录

bindy 将发现配置为模型的一部分的固定长度标头和页脚记录，这些模型与主 **@FixedLengthRecord** 类位于同一个软件包中，或者在其中一个配置的扫描软件包中存在。以下文本演示了两个已由头记录和页脚记录的固定长度记录。

```
101-08-2009
10A9 PaulineM ISINXD12345678BUYShare000002500.45USD01-08-2009
10A9 RichN ISINXD12345678BUYShare000002700.45USD01-08-2009
9000000002
```

fixed-header-and-footer-main-class

```
@FixedLengthRecord(header = OrderHeader.class, footer = OrderFooter.class)
public class Order {

    @DataField(pos = 1, length = 2)
    private int orderNr;

    @DataField(pos = 2, length = 2)
    private String clientNr;

    @DataField(pos = 3, length = 9)
    private String firstName;

    @DataField(pos = 4, length = 5, align = "L")
    private String lastName;

    @DataField(pos = 5, length = 4)
    private String instrumentCode;

    @DataField(pos = 6, length = 10)
    private String instrumentNumber;

    @DataField(pos = 7, length = 3)
    private String orderType;

    @DataField(pos = 8, length = 5)
    private String instrumentType;

    @DataField(pos = 9, precision = 2, length = 12, paddingChar = '0')
    private BigDecimal amount;

    @DataField(pos = 10, length = 3)
    private String currency;

    @DataField(pos = 11, length = 10, pattern = "dd-MM-yyyy")
    private Date orderDate;
```

```
}

@FixedLengthRecord
public  class OrderHeader {
    @DataField(pos = 1, length = 1)
    private int recordType = 1;

    @DataField(pos = 2, length = 10, pattern = "dd-MM-yyyy")
    private Date recordDate;
}

@FixedLengthRecord
public class OrderFooter {

    @DataField(pos = 1, length = 1)
    private int recordType = 9;

    @DataField(pos = 2, length = 9, align = "R", paddingChar = '0')
    private int numberOfRecordsInTheFile;
}
```

case 7 : 在解析固定长度记录时跳过内容。

通常，与提供固定长度记录的系统集成，这些记录包含比目标用例所需的信息多。在这种情况下，跳过我们不需要的那些字段的声明和解析非常有用。为了说明这一点，Bindy 将跳过转发到记录中下一个映射字段（如果下一个声明的字段的 ***pos*** *值超出了最后一个解析字段的光标位置）的下一个映射字段。对感兴趣的字段使用绝对* ***pos*** *位置（而不是 ordinal 值）会导致 Bindy 在两个字段之间跳过内容。*

同样，某些字段以外的任何内容都不可能值得关注。在这种情况下，您可以通过设置 ***@FixedLengthRecord*** *声明上的* ***ignoreTrailingChars*** *属性，告知 Bindy 跳过上次映射字段之外的所有内容的解析。*

```
@FixedLengthRecord(ignoreTrailingChars = true)
public static class Order {

    @DataField(pos = 1, length = 2)
    private int orderNr;

    @DataField(pos = 3, length = 2)
    private String clientNr;

    // any characters that appear beyond the last mapped field will be ignored

}
```

49.8. 5.消息

Message 注解用于标识将包含键值对字段的模型类。这种格式主要在 Exchange Protocol 协议消息(FIX)中使用。不过，此注解可用于通过键标识数据的任何其他格式。密钥对值彼此分隔为分隔符，可以是诸如 tab delimitor (unicode representation : \u0009）或标题的开头(unicode 表示 : \u0001)的特殊字符。

```
*"FIX information"*
```

有关 FIX 的更多信息，请参阅此网站：http://www.fixprotocol.org/。要使用 FIX 消息，模型必须包含链接到根消息类的 Header 和 Trailer 类，该类可以是 Order 类。这并不是强制性的，但当您将 camel-bindy 与 camel-fix 结合使用时，它是一个基于快速Fix 项目修复网关 http://www.quickfixj.org/ 时很有用。

注解名称	记录类型	级别
消息	键值对	类

参数名称	type	info
pairSeparator	string	Mandatory - 可以是 '=' 或 ';' 或 'anything'
keyValuePairSeparair	string	mandatory - 可以是 '\u0001', '\u0009', '#' 或 'anything'
crlf	string	可选 - 可能的值 = WINDOWS,UNIX,MAC, 或 custom; 默认值 = WINDOWS - 允许定义要使用的回车返回字符。如果您指定了前面列出的三个值，则您输入的值(custom)将用作 CRLF 字符。
type	string	可选 - 定义消息类型（如 FIX、EMX、...）
version	string	可选 - 消息版本（如 4.1）
isOrdered	布尔值	可选 - default value = false - 允许在生成 FIX 消息时更改字段的顺序。此注解与模型的消息类关联，必须被声明一次。

case 1: separator = 'u0001'

用于隔离 FIX 消息中的键值对字段的分隔符为 ASCII '01' 字符或 unicode 格式 '\u0001'。这个字符必须进行第二次转义，以避免 java 运行时错误。下面是一个示例：

```
8=FIX.4.1 9=20 34=1 35=0 49=INVMGR 56=BRKR 1=BE.CHM.001 11=CHM0001-01
22=4 ...
```

以及如何使用注解

FIX - 信息

```
@Message(keyValuePairSeparator = "=", pairSeparator = "\u0001", type="FIX", version="4.1")
public class Order {

}
```

```
*Look at test cases*
```

ASCII 字符，如 tab, ... 不能显示在 WIKI 页面中。因此，有一个名为 camel-bindy 的测试案例，以了解 FIX 消息如何像(src\test\data\fix\fix\fix.txt)和 Order, Trailer, Header classes (src\test\java\org\apache\camel\dataformat\bindy\model\simple\Order.java)

49.9. 6.KEYVALUEPAIRFIELD

KeyValuePairField 注释定义键值对字段的属性。每个 KeyValuePairField 都由一个标签(= 键)及其值关联，一个类型（字符串，int、date、...）、可选y 特征，以及是否需要该字段

注解名称	记录类型	级别
KeyValuePairField	Key Value Pair - FIX	属性

参数名称	type	info
tag	int	Mandatory - 标识消息中的字段的数字数 - 必须是唯一的
pattern	string	可选 - 默认值 = "" - 将用于格式化 Decimal, Date, ...
精度	int	可选 - 数字数字 - 代表在 Decimal 数字将被格式化/解析时使用的精度
position	int	可选 - 当 FIX 消息中的 key/tag 的位置必须不同时，必须使用
required	布尔值	可选 - 默认值 = "false"
impliedDecimalSeparator	布尔值	**Camel 2.11:** 可选 - 默认值 = "false" - 代表在指定位置有一个十进制点表示

case 1: tag

此参数表示消息中字段的键

FIX message - Tag

```
@Message(keyValuePairSeparator = "=", pairSeparator = "\u0001", type="FIX", version="4.1")
public class Order {

    @Link Header header;

    @Link Trailer trailer;
```

```
    @KeyValuePairField(tag = 1) // Client reference
    private String Account;

    @KeyValuePairField(tag = 11) // Order reference
    private String ClOrdId;

    @KeyValuePairField(tag = 22) // Fund ID type (Sedol, ISIN, ...)
    private String IDSource;

    @KeyValuePairField(tag = 48) // Fund code
    private String SecurityId;

    @KeyValuePairField(tag = 54) // Movement type ( 1 = Buy, 2 = sell)
    private String Side;

    @KeyValuePairField(tag = 58) // Free text
    private String Text;
}
```

问题单 2：输出中的不同位置

如果我们将放入 FIX 消息中的 tags/keys 必须根据预定义的顺序进行排序，则使用注释 @KeyValuePairField 的属性"position"

FIX message - Tag - sort

```
@Message(keyValuePairSeparator = "=", pairSeparator = "\\u0001", type = "FIX", version = "4.1",
isOrdered = true)
public class Order {

    @Link Header header;

    @Link Trailer trailer;

    @KeyValuePairField(tag = 1, position = 1) // Client reference
    private String account;

    @KeyValuePairField(tag = 11, position = 3) // Order reference
    private String clOrdId;
}
```

49.10. 7.部分

在固定长度记录的 FIX 消息中，在信息 : header, body 和 section 的表示中通常会有不同的部分。注释 @Section 的目的是告知绑定哪个类模型代表标头(= 第 1 节)、正文(= 第 2 节)和页脚(= 第 3 节)

此注释仅存在一个属性/参数。

注解名称	记录类型	级别
部分	FIX	类

参数名称	type	info
number	int	标识部分位置的数字

问题单 1：章节

标头部分的定义

FIX message - Section - Header

```
@Section(number = 1)
public class Header {

    @KeyValuePairField(tag = 8, position = 1) // Message Header
    private String beginString;

    @KeyValuePairField(tag = 9, position = 2) // Checksum
    private int bodyLength;
}
```

body 部分的定义

FIX message - Section - Body

```
@Section(number = 2)
@Message(keyValuePairSeparator = "=", pairSeparator = "\\u0001", type = "FIX", version = "4.1",
isOrdered = true)
public class Order {

    @Link Header header;

    @Link Trailer trailer;

    @KeyValuePairField(tag = 1, position = 1) // Client reference
    private String account;

    @KeyValuePairField(tag = 11, position = 3) // Order reference
    private String clOrdId;
```

footer 部分的定义

FIX message - Section - Footer

```
@Section(number = 3)
public class Trailer {

    @KeyValuePairField(tag = 10, position = 1)
    // CheckSum
    private int checkSum;
```

```
    public int getCheckSum() {
        return checkSum;
    }
```

49.11. 8.ONETOMANY

注释 @OneToMany 的目的是允许处理 **List<?>** 字段定义了 POJO 类，或者从包含重复组的记录中使用。

> *Restrictions OneToMany*

请注意，很多绑定不允许处理在层次结构的多个级别上定义的重复性

在以下情况下，关系 OneToMany ONLY WORKS：

- 阅读包含重复组的 FIX 消息（标签/密钥组）
- 生成带有重复数据的 CSV

注解名称	记录类型	级别
OneToMany	all	属性

参数名称	type	info
mappedTo	string	可选 - string - 与 Class> 的 List<Type 类型关联的类名称

问题单 1：使用重复数据生成 CSV

以下是我们想要的 CSV 输出：

```
Claus,Ibsen,Camel in Action 1,2010,35
Claus,Ibsen,Camel in Action 2,2012,35
Claus,Ibsen,Camel in Action 3,2013,35
Claus,Ibsen,Camel in Action 4,2014,35
```

Remark：重复数据涉及书的标题及其发布日期，而姓氏和年龄是通用的

以及用于对其进行建模的类。Author 类包含 Book 列表。

使用重复数据生成 CSV

```
@CsvRecord(separator=",")
public class Author {

    @DataField(pos = 1)
    private String firstName;
```

```
    @DataField(pos = 2)
    private String lastName;

    @OneToMany
    private List<Book> books;

    @DataField(pos = 5)
    private String Age;
}

public class Book {

    @DataField(pos = 3)
    private String title;

    @DataField(pos = 4)
    private String year;
}
```

非常简单！！!!

case 2：阅读包含标签/密钥组的 FIX 消息

以下是我们在我们的模型中要处理的信息：

```
8=FIX 4.19=2034=135=049=INVMGR56=BRKR
1=BE.CHM.00111=CHM0001-0158=this is a camel - bindy test
22=448=BE000124567854=1
22=548=BE000987654354=2
22=648=BE000999999954=3
10=220
```

标签 22、48 和 54 重复

和代码

读取包含标签/密钥组的 FIX 消息

```
public class Order {

    @Link Header header;

    @Link Trailer trailer;

    @KeyValuePairField(tag = 1) // Client reference
    private String account;

    @KeyValuePairField(tag = 11) // Order reference
    private String clOrdId;

    @KeyValuePairField(tag = 58) // Free text
    private String text;

    @OneToMany(mappedTo =
"org.apache.camel.dataformat.bindy.model.fix.complex.onetomany.Security")
    List<Security> securities;
```

```
}

public class Security {

    @KeyValuePairField(tag = 22) // Fund ID type (Sedol, ISIN, ...)
    private String idSource;

    @KeyValuePairField(tag = 48) // Fund code
    private String securityCode;

    @KeyValuePairField(tag = 54) // Movement type ( 1 = Buy, 2 = sell)
    private String side;
}
```

49.12. 9.BINDYCONVERTER

注释 @BindyConverter 的目的是定义要在字段级别上使用的转换器。提供的类必须实施 Format 接口。

```
@FixedLengthRecord(length = 10, paddingChar = ' ')
public static class DataModel {
    @DataField(pos =  1, length = 10, trim = true)
    @BindyConverter(CustomConverter.class)
    public String field1;
}

public static class CustomConverter implements Format<String> {
    @Override
    public String format(String object) throws Exception {
        return (new StringBuilder(object)).reverse().toString();
    }

    @Override
    public String parse(String string) throws Exception {
        return (new StringBuilder(string)).reverse().toString();
    }
}
```

49.13. 10.FORMATFACTORIES

注释 @FormatFactories 的目的是在记录级别上定义一组转换器。提供的类必须实施 FormatFactoryInterface 接口。

```
@CsvRecord(separator = ",")
@FormatFactories({OrderNumberFormatFactory.class})
public static class Order {

    @DataField(pos = 1)
    private OrderNumber orderNr;

    @DataField(pos = 2)
    private String firstName;
}

public static class OrderNumber {
```

```
    private int orderNr;

    public static OrderNumber ofString(String orderNumber) {
        OrderNumber result = new OrderNumber();
        result.orderNr = Integer.valueOf(orderNumber);
        return result;
    }
}

public static class OrderNumberFormatFactory extends AbstractFormatFactory {

    {
        supportedClasses.add(OrderNumber.class);
    }

    @Override
    public Format<?> build(FormattingOptions formattingOptions) {
        return new Format<OrderNumber>() {
            @Override
            public String format(OrderNumber object) throws Exception {
                return String.valueOf(object.orderNr);
            }

            @Override
            public OrderNumber parse(String string) throws Exception {
                return OrderNumber.ofString(string);
            }
        };
    }
}
```

49.14. 支持的 DATATYPES

DefaultFormatFactory 的格式通过根据提供的 FormattingOptions 返回接口 FormatFactoryInterface 实例来提供以下 datatype 格式化：

- BigDecimal
- BigInteger
- 布尔值
- byte
- 字符
- Date
- å�Œ
- Enums
- æµ®ç‚¹å€¼
- 整数

- LocalDate (java 8 since 2.18.0)
- LocalDateTime (java 8 since 2.18.0)
- localtime (java 8 since 2.18.0)
- Long
- short
- 字符串

通过提供使用的 registry 中的 FactoryRegistry 实例（如 spring 或 JNDI），可以覆盖 DefaultFormatFactory。

49.15. 使用 JAVA DSL

下一步包含实例化与此记录类型关联的 DataFormat bindy 类，并提供 Java 软件包名称作为参数。

例如，以下命令使用类 **BindyCsvDataFormat**（与与 CSV 记录类型关联的类对应），该类使用 **com.acme.model** 软件包名称初始化在此软件包中配置的模型对象。

```
// Camel 2.15 or older (configure by package name)
DataFormat bindy = new BindyCsvDataFormat("com.acme.model");


// Camel 2.16 onwards (configure by class name)
DataFormat bindy = new BindyCsvDataFormat(com.acme.model.MyModel.class);
```

49.15.1. 设置区域设置

bindy 支持在 dataformat 上配置区域设置，例如

```
// Camel 2.15 or older (configure by package name)
BindyCsvDataFormat bindy = new BindyCsvDataFormat("com.acme.model");
// Camel 2.16 onwards (configure by class name)
BindyCsvDataFormat bindy = new BindyCsvDataFormat(com.acme.model.MyModel.class);

bindy.setLocale("us");
```

或者要使用平台默认区域设置，然后使用 "default" 作为区域设置名称。请注意，这需要 Camel 2.14/2.13.3/2.12.5。

```
// Camel 2.15 or older (configure by package name)
BindyCsvDataFormat bindy = new BindyCsvDataFormat("com.acme.model");
// Camel 2.16 onwards (configure by class name)
BindyCsvDataFormat bindy = new BindyCsvDataFormat(com.acme.model.MyModel.class);

bindy.setLocale("default");
```

对于旧版本，您可以使用 Java 代码设置它，如下所示

```
// Camel 2.15 or older (configure by package name)
BindyCsvDataFormat bindy = new BindyCsvDataFormat("com.acme.model");
```

```
// Camel 2.16 onwards (configure by class name)
BindyCsvDataFormat bindy = new BindyCsvDataFormat(com.acme.model.MyModel.class);

bindy.setLocale(Locale.getDefault().getISO3Country());
```

49.15.2. Unmarshaling

```
from("file://inbox")
  .unmarshal(bindy)
  .to("direct:handleOrders");
```

另外，您可以使用对数据格式的命名引用，然后在 Registry 中定义该格式，例如您的 Spring XML 文件：

```
from("file://inbox")
  .unmarshal("myBindyDataFormat")
  .to("direct:handleOrders");
```

Camel 路由将选择 inbox 目录中的文件，unmarshall CSV 记录到模型对象集合中，并将集合发送到 'handleOrders' 所引用的路由。

返回的集合是一个 **Map 对象列表**。列表中的每个 map 包含 CSV 每行的模型对象。其后面的原因是 每行可以对应于多个对象。当您只是预期每行返回一个对象时，这可能会造成混淆。

每个对象都可以使用其类名称来检索。

```
List<Map<String, Object>> unmarshaledModels = (List<Map<String, Object>>)
exchange.getIn().getBody();

int modelCount = 0;
for (Map<String, Object> model : unmarshaledModels) {
  for (String className : model.keySet()) {
     Object obj = model.get(className);
     LOG.info("Count : " + modelCount + ", " + obj.toString());
  }
 modelCount++;
}

LOG.info("Total CSV records received by the csv bean : " + modelCount);
```

假设您要从此映射中提取单个 Order 对象来处理路由中，您可以按照以下所示使用 Splitter 和 Processor 的组合：

```
from("file://inbox")
    .unmarshal(bindy)
    .split(body())
        .process(new Processor() {
            public void process(Exchange exchange) throws Exception {
                Message in = exchange.getIn();
                Map<String, Object> modelMap = (Map<String, Object>) in.getBody();
                in.setBody(modelMap.get(Order.class.getCanonicalName()));
            }
        })
        .to("direct:handleSingleOrder")
    .end();
```

请注意，Bindy 使用 CHARSET_NAME 属性或 CHARSET_NAME 标头，如 Exchange 接口中定义的字符集转换，以执行为 unmarshalling 收到的输入流的字符集转换。在一些生成者（如 file-endpoint）中，您可以定义一个 characterset。字符集转换可能已经由此制作者完成。有时，在将此属性或标头发送到 unmarshal 之前，您需要从交换中删除此属性或标头。如果您没有删除转换，则可能会进行两次，这可能会导致不需要的结果。

```
from("file://inbox?charset=Cp922")
  .removeProperty(Exchange.CHARSET_NAME)
  .unmarshal("myBindyDataFormat")
  .to("direct:handleOrders");
```

49.15.3. marshaling

要从模型对象集合生成 CSV 记录，您需要创建以下路由：

```
from("direct:handleOrders")
   .marshal(bindy)
   .to("file://outbox")
```

49.16. 使用 SPRING XML

这真正容易使用 Spring 作为您首选的 DSL 语言，以声明要用于 camel-bindy 的路由。以下示例显示了第一个将从文件提取记录的两个路由，unmarshal 内容并将其绑定到其模型。然后，结果会发送到 pojo （什么特殊操作），并将它们放入队列中。

第二个路由将从队列中提取 pojos，并将内容 marshal 生成包含 csv 记录的文件。上面的例子是使用 Camel 2.16 以后。

Spring dsl

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://camel.apache.org/schema/spring
       http://camel.apache.org/schema/spring/camel-spring.xsd">

    <!-- Queuing engine - ActiveMq - work locally in mode virtual memory -->
    <bean id="activemq" class="org.apache.activemq.camel.component.ActiveMQComponent">
        <property name="brokerURL" value="vm://localhost:61616"/>
    </bean>

    <camelContext xmlns="http://camel.apache.org/schema/spring">
        <dataFormats>
          <bindy id="bindyDataformat" type="Csv" classType="org.apache.camel.bindy.model.Order"/>
        </dataFormats>

        <route>
            <from uri="file://src/data/csv/?noop=true" />
            <unmarshal ref="bindyDataformat" />
            <to uri="bean:csv" />
```

```
        <to uri="activemq:queue:in" />
    </route>

    <route>
        <from uri="activemq:queue:in" />
        <marshal ref="bindyDataformat" />
        <to uri="file://src/data/csv/out/" />
    </route>
  </camelContext>
</beans>
```

注意

验证您的模型类是否实现序列化，否则队列管理器将引发错误

49.17. 依赖项

要在 camel 路由中使用 Bindy，您需要添加对实现此数据格式的 **camel-bindy** 的依赖关系。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-bindy</artifactId>
  <version>x.x.x</version>
</dependency>
```

第 50 章 使用带有 CAMEL 的 OSGI 蓝图

已创建蓝图的自定义 XML 命名空间，以便您利用 nice XML dialect。给定的 Blueprint 自定义命名空间还没有标准化，此命名空间只能用于 Apache Aries Blueprint 实现，这是 Apache Karaf 使用的。

50.1. 概述

XML 模式与 Spring 基本相同，因此引用 Spring XML 的文档中的所有 xml 片断也适用于 Blueprint 路由。

以下是使用蓝图的简单路由定义：

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">

  <camelContext xmlns="http://camel.apache.org/schema/blueprint">
    <route>
      <from uri="timer:test" />
      <to uri="log:test" />
    </route>
  </camelContext>

</blueprint>
```

此时，支持的 xml 元素有一些限制（与 Spring xml 语法不同）：

- beanPostProcessor 特定于 Spring，不被允许

但是，当您在 OSGi 中部署应用程序时使用蓝图具有以下优点：

- 当升级到新的 camel 版本时，您不必更改命名空间，因为会根据捆绑包导入的 camel 软件包选择正确的版本
- 没有与自定义命名空间和捆绑包相关的启动排序问题
- 您可以使用 Blueprint 属性占位符

50.2. 使用 CAMEL-BLUEPRINT

要在 OSGi 中利用 camel-blueprint，除了 camel-core 及其依赖项外，您还需要 Aries Blueprint 捆绑包和 camel-blueprint 捆绑包。

如果使用 Karaf，您可以使用名为 camel-blueprint 的功能，该功能将安装所有所需的捆绑包。

第 51 章 BONITA 组件

从 Camel 版本 2.19 开始提供

用于与远程 Bonita BPM 流程引擎通信。

51.1. URI 格式

```
bonita:///[operation]?[options]
```

其中 **operation** 是对 Bonita 执行的特定操作。

51.2. 常规选项

Bonita 组件没有选项。

Bonita 端点使用 URI 语法进行配置：

```
bonita:operation
```

使用以下路径和查询参数：

51.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
operation	要使用的必要操作		BonitaOperation

51.2.2. 查询参数(9 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
hostname (consumer)	Bonita 引擎运行的主机名	localhost	字符串
port (consumer)	托管 Bonita 引擎的服务器端口	8080	字符串
processName (consumer)	操作中涉及的进程的名称		字符串

Name	描述	默认值	类型
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
密码 (security)	向 Bonita 引擎进行身份验证的密码。		字符串
用户名 (security)	向 Bonita 引擎进行身份验证的用户名。		字符串

51.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.bonita.enabled	启用 bonita 组件	true	布尔值
camel.component.bonita.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

51.4. 正文内容

对于 startCase 操作，输入变量从正文消息中检索。这需要包含 Map<String,Serializable>。

51.5. 例子

以下示例在 Bonita 中启动新的问题单：

```
from("direct:start").to("bonita:startCase?
hostname=localhost&amp;port=8080&amp;processName=TestProcess&amp;username=install&amp;pa
ssword=install")
```

51.6. 依赖项

*要在 Camel 路由中使用 Bonita，您需要对实现该组件的 **camel-bonita** 添加依赖项。*

如果使用 Maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-bonita</artifactId>
  <version>x.x.x</version>
</dependency>
```

第 52 章 BOON DATAFORMAT

从 Camel 版本 2.16 开始提供

boon 是一个 Data Format，它使用 Boon JSON marshalling 库将 JSON 有效负载 unmarshal 到一个 Java 对象，或将 Java 对象 marshal Java 对象放入 JSON 有效负载中。boon 旨在比当前使用的 其他常见解析程序简单而快速的 解析程序。

52.1. 选项

Boon dataformat 支持 3 个选项，如下所列。

Name	默认值	Java 类型	描述
unmarshalTypeName		字符串	取消警报时要使用的 java 类型的类名称
useList	false	布尔值	要取消警报到映射列表或 Pojo 列表。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

52.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.boon.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.boon.enabled	启用 boon dataformat	true	布尔值
camel.dataformat.boon.unmarshal-type-name	取消警报时要使用的 java 类型的类名称		字符串
camel.dataformat.boon.use-list	要取消警报到映射列表或 Pojo 列表。	false	布尔值

ND

52.3. 使用 JAVA DSL

```
DataFormat boonDataFormat = new BoonDataFormat("com.acme.model.Person");

from("activemq:My.Queue")
  .unmarshal(boonDataFormat)
  .to("mqseries:Another.Queue");
```

52.4. 使用蓝图 XML

```
<bean id="boonDataFormat" class="org.apache.camel.component.boon.BoonDataFormat">
  <argument value="com.acme.model.Person"/>
</bean>

<camelContext id="camel" xmlns="http://camel.apache.org/schema/blueprint">
  <route>
    <from uri="activemq:My.Queue"/>
    <unmarshal ref="boonDataFormat"/>
    <to uri="mqseries:Another.Queue"/>
  </route>
</camelContext>
```

52.5. 依赖项

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-boon</artifactId>
  <version>x.x.x</version>
</dependency>
```

第 53 章 BOX COMPONENT

从 Camel 版本 2.14 开始提供

Box 组件提供对使用 https://github.com/box/box-java-sdk 访问的所有 Box.com API 的访问权限。它允许生成消息，以上传和下载文件、创建、编辑和管理文件夹等。它还支持 API，允许轮询用户帐户更新，甚至更改企业帐户等。

box.com 要求将 OAuth2.0 用于所有客户端应用身份验证。要将 camel-box 与您的帐户搭配使用，您需要在 Box.com 中创建一个新应用程序，网址为 https://developer.box.com/Box 应用的客户端 id 和 secret 将允许访问需要当前用户的 Box API。用户访问令牌由 API 为最终用户生成和管理。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-box</artifactId>
  <version>${camel-version}</version>
</dependency>
```

53.1. 连接身份验证类型

Box 组件支持三种不同类型的身份验证连接：

53.1.1. 标准身份验证

标准身份验证 使用 **OAuth 2.0 三委派的身份验证流程** 通过 Box.com 验证其连接。这种类型的身份验证支持 Box **管理用户和 外部用户** 通过 Box 组件访问、编辑和保存其 Box 内容。

53.1.2. 应用程序企业身份验证

应用 **企业身份验证 使用带 JSON Web 令牌(JWT)的 OAuth 2.0** 验证其连接作为 **Box 应用的服务帐户。这种类型的身份验证可让服务帐户通过 Box 组件访问、编辑并保存其 Box 应用 的内容。**

53.1.3. 应用程序用户身份验证

应用程序 用户身份验证 使用带 JSON Web 令牌(JWT)的 OAuth 2.0 验证其连接，作为 Box 应用的 App User。这种类型的身份验证可让 应用程序用户访问、编辑 Box 内容，并通过 Box 组件将其 Box 内容保存到其 Box 应用中。

53.2. 选框选项

Box 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	使用共享配置		BoxConfiguration

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Box 端点使用 URI 语法进行配置：

```
box:apiName/methodName
```

使用以下路径和查询参数：

53.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
apiName	**需要** 执行什么操作		BoxApiName
methodName	**必需的** 所选操作使用哪些子操作		字符串

53.2.2. 查询参数(20 参数)：

Name	描述	默认值	类型
clientId (common)	box 应用程序客户端 ID		字符串
enterpriseId (common)	用于应用程序企业的企业 ID。		字符串
inBody (common)	设置要在交换 In Body 中传递的参数名称		字符串
userId (common)	用于应用程序用户的用户 ID。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler

Name	描述	默认值	类型
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
httpParams (advanced)	用于代理主机等设置的自定义 HTTP 参数		Map
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
accessTokenCache (security)	用于存储和检索访问令牌的自定义访问令牌缓存。		IAccessTokenCache
clientSecret (security)	box 应用程序客户端 secret		字符串
encryptionAlgorithm (security)	JWT 的加密算法类型。支持的算法：RSA_SHA_256 RSA_SHA_384 RSA_SHA_512	RSA_SHA_256	EncryptionAlgorithm
maxCacheEntries (security)	缓存中访问令牌的最大数量。	100	int
authenticationType (authentication)	用于连接的验证类型。身份验证类型：STANDARD_AUTHENTICATION - OAuth 2.0 (3-legged) SERVER_AUTHENTICATION - OAuth 2.0 with JSON Web Tokens	APP_USER_AUTHENTICATION	字符串
privateKeyFile (security)	用于生成 JWT 签名的私钥。		字符串
privateKeyPassword (security)	私钥的密码。		字符串
publicKeyId (security)	用于验证 JWT 签名的公钥 ID。		字符串
sslContextParameters (security)	使用 SSLContextParameters 配置安全性。		SSLContextParameters
用户名 (security)	选框用户名，必须提供		字符串
userPassword (security)	如果没有设置 authSecureStorage，或者在第一次调用时返回 null 用户密码，则必须提供 MUST		字符串

53.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 17 个选项，如下所列。

Name	描述	默认值	类型
camel.component.box.configuration.access-token-cache	用于存储和检索访问令牌的自定义访问令牌缓存。		IAccessTokenCache
camel.component.box.configuration.api-name	要执行的操作类型		BoxApiName
camel.component.box.configuration.authentication-type	用于连接的验证类型。身份验证类型：STANDARD_AUTHENTICATION - OAuth 2.0 (3-legged) SERVER_AUTHENTICATION - OAuth 2.0 with JSON Web Tokens	APP_USER_AUTHENTICATION	字符串
camel.component.box.configuration.client-id	box 应用程序客户端 ID		字符串
camel.component.box.configuration.client-secret	box 应用程序客户端 secret		字符串
camel.component.box.configuration.enterprise-id	用于应用程序企业的企业 ID。		字符串
camel.component.box.configuration.http-params	用于代理主机等设置的自定义 HTTP 参数		Map
camel.component.box.configuration.method-name	用于所选操作的子操作		字符串
camel.component.box.configuration.private-key-file	用于生成 JWT 签名的私钥。		字符串
camel.component.box.configuration.private-key-password	私钥的密码。		字符串
camel.component.box.configuration.public-key-id	用于验证 JWT 签名的公钥 ID。		字符串

Name	描述	默认值	类型
camel.component.box.configuration.ssl-context-parameters	使用 SSLContextParameters 配置安全性。		SSLContextParameters
camel.component.box.configuration.user-id	用于应用程序用户的用户 ID。		字符串
camel.component.box.configuration.user-name	选框用户名，必须提供		字符串
camel.component.box.configuration.user-password	如果没有设置 authSecureStorage，或者在第一次调用时返回 null 用户密码，则必须提供 MUST		字符串
camel.component.box.enabled	启用框组件	true	布尔值
camel.component.box.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

53.4. URI 格式

```
box:apiName/methodName
```

apiName 可以是以下之一：

- *协作*
- *注释*
- *event-logs*
- *files*
- *文件夹*
- *groups*
- *events*
- *search*
- *tasks*
- *users*

53.5. 生产者端点：

生产者端点可以使用端点前缀，后跟端点名称和关联的选项。可以将简写别名用于某些端点。端点 URI MUST 包含前缀。

不是强制的端点选项由 [] 表示。当端点没有强制选项时，必须提供其中一组 [] 选项。生成者端点也可以使用特殊选项 inBody，它应包含端点选项的名称，其值将包含在 Camel Exchange In 消息中。

任何端点选项都可以在端点 URI 中提供，或者在消息标头中动态提供。消息标头名称必须是 CamelBox.<option> 格式。请注意，inBody 选项会覆盖消息标头，即 Body=option 中的端点 选项会覆盖 CamelBox.option 标头。

如果没有为端点 URI 或消息标头中为 option defaultRequest 提供值，则会假定为 null。请注意，只有其他选项无法满足匹配端点时，才会使用 null 值。

如果是 Box API 错误，端点将引发一个 RuntimeCamelException，并带有 com.box.sdk.BoxAPIException 派生异常原因。

53.5.1. 端点 前缀协作

有关 Box 协作的更多信息，请参阅 https://developer.box.com/reference#collaboration-object。以下端点可以通过前缀 协作调用， 如下所示：

```
box:collaborations/endpoint?[options]
```

端点	简写别名	选项	结果正文类型
addFolderCollaboration	add	folderId, collaborator, role	com.box.sdk.BoxCollaboration
addFolderCollaborationByEmail	addByEmail	folderId, email, role	com.box.sdk.BoxCollaboration
deleteCollaboration	delete	collaborationId	
getFolderCollaborations	协作	folderId	java.util.Collection

端点	简写别名	选项	结果正文类型
getPendingCollaborations	pendingCollaborations		java.util.Collection
getCollaborationInfo	info	collaborationId	com.box.sdk.BoxCollaboration.Info
updateCollaborationInfo	updateInfo	collaborationId, info	com.box.sdk.BoxCollaboration

协作的 URI 选项

Name	类型
collaborationId	字符串
collaborator	com.box.sdk.BoxCollaborator
role	com.box.sdk.BoxCollaboration.Role
folderId	字符串
email	字符串
info	com.box.sdk.BoxCollaboration.Info

53.5.2. 端点前缀 注释

有关 Box 注释的更多信息，请参阅 https://developer.box.com/reference#comment-object。以下端点可以通过前缀 注释 调用，如下所示：

```
box:comments/endpoint?[options]
```

端点	简写别名	选项	结果正文类型
addFileComment	add	fileId, message	com.box.sdk.BoxFile
changeCommentMessage	updateMessage	commentId, message	com.box.sdk.BoxComment
deleteComment	delete	commentId	
getCommentInfo	info	commentId	com.box.sdk.BoxComment.Info
getFileComments	注释	fileId	java.util.List
replyToComment	回复	commentId, message	com.box.sdk.BoxComment

协作的 URI 选项

Name	类型
commentId	字符串
fileId	字符串
message	字符串

53.5.3. 端点前缀 events-logs

有关 Box 事件日志的更多信息，请参阅 https://developer.box.com/reference#events。以下端点可以通过前缀 event-logs 调用，如下所示：

```
box:event-logs/endpoint?[options]
```

端点	简写别名	选项	结果正文类型
getEnterpriseEvents	events	position, after, before, [types]	java.util.List

event-logs的URI 选项

Name	类型
position	字符串
after	Date
before	Date
类型	com.box.sdk.BoxEvent.Types[]

53.5.4. 端点前缀 文件

有关 Box 文件的更多信息，请参阅 https://developer.box.com/reference#file-object。以下端点可以通过前缀 文件 调用，如下所示：

```
box:files/endpoint?[options]
```

端点	简写别名	选项	结果正文类型
uploadFile	上传	parentFolderId, content, fileName, [created], [modified], [size], [listener]	com.box.sdk.BoxFile

端点	简写别名	选项	结果正文类型
downloadFile	下载	fileId, output, [rangeStart], [rangeEnd], [listener]	java.io.OutputStream
copyFile	复制	fileId, destinationFolderId, [newName]	com.box.sdk.BoxFile
moveFile	Move	fileId, destinationFolderId, [newName]	com.box.sdk.BoxFile
renameFile	rename	fileId, newFileName	com.box.sdk.BoxFile
createFileSharedLink	link	fileId, access, [unshareDate], [permissions]	com.box.sdk.BoxSharedLink
deleteFile	delete	fileId	
uploadNewFileVersion	uploadVersion	fileId, fileContent, [modified], [fileSize], [listener]	com.box.boxsdk.BoxFile

端点	简写别名	选项	结果正文类型
promoteFileVersion	promoteVersion	fileId, version	com.box.sdk.BoxFileVersion
getFileVersions	版本	fileId	java.util.Collection
downloadPreviousFileVersions	downloadVersion	fileId, version, output, [listener]	java.io.OutputStream
deleteFileVersion	deleteVersion	fileId, version	
getFileInfo	info	fileId, fields	com.box.sdk.BoxFile.Info
updateFileInfo	updateInfo	fileId, info	com.box.sdk.BoxFile
createFileMetadata	createMetadata	fileId, metadata, [typeName]	com.box.sdk.Metadata
getFileMetadata	metadata	fileId, [typeName]	com.box.sdk.Metadata
updateFileMetadata	updateMetadata	fileId, metadata	com.box.sdk.Metadata
deleteFileMetadata	deleteMetadata	fileId	
getDownloadUrl	url	fileId	java.net.URL

端点	简写别名	选项	结果正文类型
getPreviewLink	预览	fileId	java.net.URL
getFileThumbnail	thumbnail	fileId, fileType, minWidth, minHeight, maxWidth, maxHeight	byte[]

文件的URI 选项

Name	类型
parentFolderId	字符串
content	java.io.InputStream
fileName	字符串
created	Date
modified	Date
size	Long
listener	com.box.sdk.ProgressListener
output	java.io.OutputStream
rangeStart	Long

Name	类型
rangeEnd	Long
outputStreams	java.io.OutputStream[]
destinationFolderId	字符串
newName	字符串
fields	string[]
info	com.box.sdk.BoxFile.Info
fileSize	Long
version	整数
access	com.box.sdk.BoxSharedLink.Access
unshareDate	Date
权限	com.box.sdk.BoxSharedLink.Permissions
fileType	com.box.sdk.BoxFile.ThumbnailFileType
minWidth	整数
minHeight	整数
maxWidth	整数
maxHeight	整数
metadata	com.box.sdk.Metadata

Name	类型
typeName	字符串

53.5.5. 端点前缀 文件夹

有关 Box 文件夹的更多信息，请参阅 https://developer.box.com/reference#folder-object。以下端点可以通过前缀 文件夹 调用，如下所示：

> *box:folders/endpoint?[options]*

端点	简写别名	选项	结果正文类型
getRootFolder	root		com.box.sdk.BoxFolder
createFolder	create	parentFolderId, folderName	com.box.sdk.BoxFolder
createFolder	create	parentFolderId, path	com.box.sdk.BoxFolder
copyFolder	复制	folderId, destinationfolderId, [newName]	com.box.sdk.BoxFolder
moveFolder	Move	folderId, destinationFolderId, newName	com.box.sdk.BoxFolder
renameFolder	rename	folderId, newFolderName	com.box.sdk.BoxFolder

端点	简写别名	选项	结果正文类型
createFolderSharedLink	link	folderId, access, [unsharedDate], [permissions]	java.util.List
deleteFolder	delete	folderId	
getFolder	folder	path	com.box.sdk.BoxFolder
getFolderInfo	info	folderId, fields	com.box.sdk.BoxFolder.Info
getFolderItems	items	folderId, offset, limit, fields	java.util.List
updateFolderInfo	updateInfo	folderId, info	com.box.sdk.BoxFolder

文件夹的 URI 选项

Name	类型
path	string[]
folderId	字符串
offset	Long
limit	Long
fields	string[]

Name	类型
parentFolderId	字符串
folderName	字符串
destinationFolderId	字符串
newName	字符串
newFolderName	字符串
info	字符串
access	com.box.sdk.BoxSharedLink.Access
unshareDate	Date
权限	com.box.sdk.BoxSharedLink.Permissions

53.5.6. 端点前缀 组

有关 Box 组的更多信息，请参阅 https://developer.box.com/reference#group-object。以下端点可以通过前缀 组 调用，如下所示：

> *box:groups/endpoint?[options]*

端点	简写别名	选项	结果正文类型

端点	简写别名	选项	结果正文类型
createGroup	create	Name, [provenance, externalSyncIdentifier, description, invitabilityLevel, memberViewabilityLevel]	com.box.sdk.BoxGroup
addGroupMembership	createMembership	GroupId, userId, role	com.box.sdk.BoxGroupMembership
deleteGroup	delete	groupId	
getAllGroups	groups		java.util.Collection
getGroupInfo	info	groupId	com.box.sdk.BoxGroup.Info
updateGroupInfo	updateInfo	groupId, groupInfo	com.box.sdk.BoxGroup
addGroupMembership	addMembership	GroupId, userId, role	com.box.sdk.BoxGroupMembership
deleteGroupMembership	deleteMembership	groupMembershipId	

端点	简写别名	选项	结果正文类型
getGroupMemberships	成员资格	groupId	java.uti.Collection
getGroupMembershipInfo	membershipInfo	groupMemebershipId	com.box.sdk.BoxGroup.Info
updateGroupMembershipInfo	updateMembershipInfo	groupMemebershipId, info	com.box.sdk.BoxGroupMembership

组的URI 选项

Name	类型
name	字符串
groupId	字符串
userId	字符串
role	com.box.sdk.BoxGroupMembership.Role
groupMembershipId	字符串
info	com.box.sdk.BoxGroupMembership.Info

53.5.7. 端点前缀 搜索

有关 Box 搜索 API 的更多信息，请参阅 https://developer.box.com/reference#searching-for-content。以下端点可以通过前缀 搜索 来调用，如下所示：

```
box:search/endpoint?[options]
```

端点	简写别名	选项	结果正文类型
searchFolder	search	folderId, query	java.util.Collection

搜索的 URI 选项

Name	类型
folderId	字符串
query	字符串

53.5.8. 端点前缀 任务

有关 Box 任务的详情，请参考 https://developer.box.com/reference#task-object-1。以下端点可以通过前缀 任务 调用：

```
box:tasks/endpoint?[options]
```

端点	简写别名	选项	结果正文类型
addFileTask	add	fileId, action, dueAt, [message]	com.box.sdk.BoxUser
deleteTask	delete	taskId	
getFileTasks	tasks	fileId	java.util.List
getTaskInfo	info	taskId	com.box.sdk.BoxTask.Info
updateTaskInfo	updateInfo	TaskID, info	com.box.sdk.BoxTask

端点	简写别名	选项	结果正文类型
addAssignmentToTask	addAssignment	taskId, assignTo	com.box.sdk.BoxTask
deleteTaskAssignment	deleteAssignment	taskAssignmentId	
getTaskAssignments	分配	taskId	java.util.List
getTaskAssignmentInfo	assignmentInfo	taskAssignmentId	com.box.sdk.BoxTaskAssignment.Info

任务的URI 选项

Name	类型
fileId	字符串
action	com.box.sdk.BoxTask.Action
dueAt	Date
message	字符串
taskId	字符串
info	com.box.sdk.BoxTask.Info
assignTo	com.box.sdk.BoxUser
taskAssignmentId	字符串

53.5.9. 端点前缀 用户

有关 Box 用户的详情，请参考 https://developer.box.com/reference#user-object。以下端点可以通过前缀 用户 调用，如下所示：

```
box:users/endpoint?[options]
```

端点	简写别名	选项	结果正文类型
getCurrentUser	currentUser		com.box.sdk.BoxUser
getAllEnterpriseOrExternalUsers	users	filterTerm, [fields]	com.box.sdk.BoxUser
createAppUser	create	Name, [params]	com.box.sdk.BoxUser
createEnterpriseUser	create	login, name, [params]	com.box.sdk.BoxUser
deleteUser	delete	userId, notifyUser, force	
getUserEmailAlias	emailAlias	userId	com.box.sdk.BoxUser
deleteUserEmailAlias	deleteEmailAlias	userId, emailAliasId	java.util.List
getUserInfo	info	userId	com.box.sdk.BoxUser.Info
updateUserInfo	updateInfo	userId, info	com.box.sdk.BoxUser
moveFolderToUser	-	userId, sourceUserId	com.box.sdk.BoxFolder.Info

用户的URI 选项

Name	类型
defaultRequest	com.box.restclientv2.requestsbase.BoxDefaultRequestObject
emailAliasRequest	com.box.boxjavalibv2.requests.requestobjects.BoxEmailAliasRequestObject
emailId	字符串
filterTerm	字符串
folderId	字符串
simpleUserRequest	com.box.boxjavalibv2.requests.requestobjects.BoxSimpleUserRequestObject
userDeleteRequest	com.box.boxjavalibv2.requests.requestobjects.BoxUserDeleteRequestObject
userId	字符串
userRequest	com.box.boxjavalibv2.requests.requestobjects.BoxUserRequestObject
userUpdateLoginRequest	com.box.boxjavalibv2.requests.requestobjects.BoxUserUpdateLoginRequestObject

53.6. 消费者端点：

有关 Box 事件的更多信息，请参阅 https://developer.box.com/reference#events。消费者端点只能使用端点前缀 事件，如下一个示例所示。

```
box:events/endpoint?[options]
```

端点	简写别名	选项	结果正文类型

端点	简写别名	选项	结果正文类型
events		[startingPosition]	com.box.sdk.BoxEvent

事件的URI 选项

Name	类型
startingPosition	Long

53.7. 消息标头

任何选项都可以在带有 CamelBox. 前缀的制作者端点的消息标头中提供。

53.8. 消息正文

所有结果消息正文都使用 Box Java SDK 提供的对象。生产者端点可以在 inBody 端点参数中指定传入消息正文的选项名称。

53.9. 配置组件并启用身份验证

对于使用 Box 进行身份验证，请在组件上设置身份验证属性，如下例所示：

```
BoxConfiguration configuration = new BoxConfiguration();
configuration.setClientId("clientId");
configuration.setClientSecret("clientSecret");
configuration.setUserName("userName");
configuration.setUserPassword("userPassword");

// add BoxComponent to Camel context
BoxComponent component = new BoxComponent(context);
component.setConfiguration(configuration);
context.addComponent("box", component);
```

53.10. SAMPLES

以下路由将新文件上传到用户的根目录：

```
from("file:...")
    .to("box://files/upload/inBody=fileUploadRequest");
```

以下路由轮询用户帐户是否有更新：

```
from("box://events/listen?startingPosition=-1")
    .to("bean:blah");
```

以下路由使用带有动态标头选项的制作者。fileId 属性具有 Box file id，output 属性包含文件内容的输出流，因此它们分别被分配给 CamelBox.fileId 标头和 CamelBox.output 标头，如下所示：

```
from("direct:foo")
    .setHeader("CamelBox.fileId", header("fileId"))
    .setHeader("CamelBox.output", header("output"))
    .to("box://files/download")
    .to("file://...");
```

第 54 章 BRAINTREE 组件

从 Camel 版本 2.17 开始提供

Braintree 组件通过 Java SDK 提供对 Braintree 支付 的访问。

所有客户端应用都需要 API 凭据来处理付款。要将 camel-braintree 与您的帐户搭配使用，您需要创建一个新的 Sandbox 或 Production 帐户。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-braintree</artifactId>
  <version>${camel-version}</version>
</dependency>
```

54.1. BRAINTREE 选项

Braintree 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	使用共享配置		BraintreeConfiguration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Braintree 端点使用 URI 语法进行配置：

```
braintree:apiName/methodName
```

使用以下路径和查询参数：

54.1.1. 路径参数(2 参数)：

Name	描述	默认值	类型
apiName	**需要** 执行什么操作		BraintreeApiName
methodName	用于所选操作的子操作		字符串

54.1.2. 查询参数(14 参数)：

Name	描述	默认值	类型
environment (common)	环境 Either SANDBOX 或 PRODUCTION		字符串
inBody (common)	设置要在交换 In Body 中传递的参数名称		字符串
merchantId (common)	Braintree 提供的 merchant id。		字符串
privatekey (common)	Braintree 提供的私钥。		字符串
publickey (common)	Braintree 提供的公钥。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
accessToken (advanced)	为代表处理事务而被管理员授予的访问令牌。用于替换环境、erchant id、公钥和私钥字段。		字符串
httpReadTimeout (advanced)	为 http 调用设置读取超时。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
httpLogLevel (logging)	为 http 调用设置日志记录级别，请参阅 java.util.logging.Level		字符串
proxyHost (proxy)	代理主机		字符串
proxyPort (proxy)	代理端口		整数

54.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 14 个选项，如下所列。

Name	描述	默认值	类型
camel.component.braintree.configuration.access-token	为代表处理事务而被管理员授予的访问令牌。用于替换环境、erchant id、公钥和私钥字段。		字符串
camel.component.braintree.configuration.api-name	要执行的操作类型		BraintreeApiName
camel.component.braintree.configuration.environment	环境 Either SANDBOX 或 PRODUCTION		字符串
camel.component.braintree.configuration.http-log-level	为 http 调用设置日志记录级别，请参阅 java.util.logging.Level		级别
camel.component.braintree.configuration.http-log-name	设置日志类别，以用于记录 http 调用，默认"Braintree"		字符串
camel.component.braintree.configuration.http-read-timeout	为 http 调用设置读取超时。		整数
camel.component.braintree.configuration.merchant-id	Braintree 提供的 merchant id。		字符串
camel.component.braintree.configuration.method-name	用于所选操作的子操作		字符串
camel.component.braintree.configuration.private-key	Braintree 提供的私钥。		字符串

Name	描述	默认值	类型
camel.component.braintree.configuration.proxy-host	代理主机		字符串
camel.component.braintree.configuration.proxy-port	代理端口		整数
camel.component.braintree.configuration.public-key	Braintree 提供的公钥。		字符串
camel.component.braintree.enabled	启用 braintree 组件	true	布尔值
camel.component.braintree.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

54.3. URI 格式

> ***braintree://endpoint-prefix/endpoint?[options]***

端点前缀可以是以下之一：

- *addOn*
- *address*
- *clientToken*
- *creditCardverification*
- *客户*
- *折现*
- *dispute*
- *documentUpload*
- *merchantAccount*
- *paymentmethod*
- *paymentmethodNonce*
- *plan*

- *报告*
- *settlementBatchSummary*
- *subscription*
- *transaction*
- *webhookNotification*

54.4. BRAINTREECOMPONENT

Braintree 组件可以使用以下选项进行配置。这些选项可以使用组件的 bean 属性配置 提供，类型为 org.apache.camel.component.braintree.BraintreeConfiguration。

选项	类型	描述
环境	字符串	指定请求应定向到的位置 - 沙盒或生产环境的值
merchantId	字符串	网关帐户的唯一标识符，它与您的 merchant 帐户 ID 不同
publicKey	字符串	特定于用户的公共标识符
privateKey	字符串	用户特定的安全标识符不应共享 - 即使我们一样！
accessToken	字符串	使用 Braintree Auth 授予的令牌，允许它们代表另一个处理事务。用于替换环境 merchantId、publicKey 和 privateKey 选项。

以上所有选项都由 Braintree Payments 提供

54.5. 生产者端点：

生产者端点可以使用端点前缀，后跟端点名称和关联的选项。可以将简写别名用于某些端点。端点 URI MUST 包含前缀。

未强制的端点选项由 [] 表示。当端点没有强制选项时，必须提供其中一组 [] 选项。生成者端点也可以使用特殊选项 inBody，它应包含端点选项的名称，其值将包含在 Camel Exchange In 消息中。

任何端点选项都可以在端点 URI 中提供，或者在消息标头中动态提供。消息标头名称必须是 CamelBraintree.<option> 格式。请注意，inBody 选项会覆盖消息标头，即 Body=option 中的端点选项会覆盖 CamelBraintree.option 标头。

有关端点和选项的更多信息，请参阅 Braintree 参考
https://developers.braintreepayments.com/reference/overview

54.5.1. 端点前缀 addOn

以下端点可以通过 prefix addOn 调用，如下所示：

> ***braintree://addOn/endpoint***

端点	简写别名	选项	结果正文类型
all			List<com.braintreegateway.Addon>

54.5.2. 端点前缀 地址

以下端点可以通过前缀 地址 调用，如下所示：

> ***braintree://address/endpoint?[options]***

端点	简写别名	选项	结果正文类型
create		customerId, request	com.braintreegateway.Result<com.braintreegateway.Address>
delete		customerId, id	com.braintreegateway.Result<com.braintreegateway.Address>
查找		customerId, id	com.braintreegateway.Address
update		customerId, id, request	com.braintreegateway.Result<com.braintreegateway.Address>

地址的URI 选项

Name	类型
customerId	字符串
request	com.braintreegateway.AddressRequest
id	字符串

54.5.3. 端点前缀 clientToken

以下端点可以通过前缀 clientToken 调用，如下所示：

> ***braintree://clientToken/endpoint?[options]***

端点	简写别名	选项	结果正文类型
generate		request	字符串

clientToken的 URI 选项

Name	类型
request	com.braintreegateway.ClientTokenrequest

54.5.4. 端点前缀 creditCardVerification

以下端点可以通过前缀 creditCardverification 调用，如下所示：

> ***braintree://creditCardVerification/endpoint?[options]***

端点	简写别名	选项	结果正文类型
查找		id	com.braintreegateway.CreditCardVerification
search		query	com.braintreegateway.ResourceCollection<com.braintreegateway.CreditCardVerification>

creditCardVerification的 URI 选项

Name	类型
id	字符串
query	com.braintreegateway.CreditCardVerificationSearchRequest

54.5.5. 端点前缀 客户

以下端点可以通过前缀 客户 调用：

> ***braintree://customer/endpoint?[options]***

端点	简写别名	选项	结果正文类型
all			

端点	简写别名	选项	结果正文类型
create		request	com.braintreegateway.Result<com.braintreegateway.Customer>
delete		id	com.braintreegateway.Result<com.braintreegateway.Customer>
查找		id	com.braintreegateway.Customer
search		query	com.braintreegateway.ResourceCollection<com.braintreegateway.Customer>
update		id, request	com.braintreegateway.Result<com.braintreegateway.Customer>

客户的URI 选项

Name	类型
id	字符串
request	com.braintreegateway.CustomerRequest
query	com.braintreegateway.CustomerSearchRequest

54.5.6. 端点前缀 折扣

以下端点可以通过前缀 discount 调用， 如下所示：

> ***braintree://discount/endpoint***

端点	简写别名	选项	结果正文类型
all			List<com.braintreegateway.Discount>

> ***+***

> ***+***

54.5.7. 端点前缀 冲突

以下端点可以使用前缀冲突调用， 如下所示：

> **braintree://dispute/endpoint?[options]**

端点	简写别名	选项	结果正文类型
accept		id	com.braintreegateway.Result
addFileEvidence		disputeId, documentId	com.braintreegateway.Result<DisputeEvidence>
addFileEvidence		disputeId, fileEvidenceRequest	com.braintreegateway.Result<DisputeEvidence>
addTextEvidence		disputeId, content	com.braintreegateway.Result<DisputeEvidence>
addTextEvidence		disputeId, textEvidenceRequest	com.braintreegateway.Result<DisputeEvidence>
完成		id	com.braintreegateway.Result
查找		id	com.braintreegateway.Dispute
removeEvidence		id	com.braintreegateway.Result
search		disputeSearchRequest	com.braintreegateway.PaginatedCollection<com.braintreegateway.Dispute>

dispute的 URI 选项

Name	类型
id	字符串
disputeId	字符串
documentId	字符串
fileEvidenceRequest	com.braintreegateway.FileEvidenceRequest

Name	类型
content	字符串
	textEvidenceRequest
com.braintreegateway.TextEvidenceRequest	disputeSearchRequest

54.5.8. 端点前缀 documentUpload

以下端点可以通过前缀 documentUpload 调用，如下所示：

> ***braintree://documentUpload/endpoint?[options]***

端点	简写别名	选项	结果正文类型
create		request	com.braintreegateway.Result<com.braintreegateway.DocumentUpload>

documentUpload的 URI 选项

Name	类型
request	com.braintreegateway.DocumentUploadRequest

54.5.9. 端点前缀 merchantAccount

以下端点可以通过前缀 merchantAccount 调用，如下所示：

> ***braintree://merchantAccount/endpoint?[options]***

端点	简写别名	选项	结果正文类型
create		request	com.braintreegateway.Result<com.braintreegateway.MerchantAccount>
createForCurrency		currencyRequest	com.braintreegateway.Result<com.braintreegateway.MerchantAccount>
查找		id	com.braintreegateway.MerchantAccount

端点	简写别名	选项	结果正文类型
update		id, request	com.braintreegateway.Result<com.braintreegateway.MerchantAccount>

merchantAccount的 URI 选项

Name	类型
id	字符串
request	com.braintreegateway.MerchantAccountRequest
currencyRequest	com.braintreegateway.MerchantAccountCreateForCurrencyRequest

54.5.10. 端点前缀 paymentMethod

以下端点可以使用前缀paymentMethod 调用，如下所示：

> ***braintree://paymentMethod/endpoint?[options]***

端点	简写别名	选项	结果正文类型
create		request	com.braintreegateway.Result<com.braintreegateway.PaymentMethod>
delete		token, deleteRequest	com.braintreegateway.Result<com.braintreegateway.PaymentMethod>
查找		token	com.braintreegateway.PaymentMethod
update		令牌、请求	com.braintreegateway.Result<com.braintreegateway.PaymentMethod>

paymentMethod的 URI 选项

Name	类型
token	字符串
request	com.braintreegateway.PaymentMethodRequest
deleteRequest	com.braintreegateway.PaymentMethodDeleteRequest

54.5.11. 端点前缀paymentMethodNonce

以下端点可以使用前缀paymentMethodNonce 调用，如下所示：

> ***braintree://paymentMethodNonce/endpoint?[options]***

端点	简写别名	选项	结果正文类型
create		paymentMethodToken	com.braintreegateway.Result<com.braintreegateway.PaymentMethodNonce>
查找		paymentMethodNonce	com.braintreegateway.PaymentMethodNonce

paymentMethodNonce的URI 选项

Name	类型
paymentMethodToken	字符串
paymentMethodNonce	字符串

54.5.12. 端点前缀 计划

以下端点可以通过前缀 计划 调用：

> ***braintree://plan/endpoint***

端点	简写别名	选项	结果正文类型

端点	简写别名	选项	结果正文类型
all			List<com.braintreegateway.Plan>

54.5.13. 端点前缀 报告

以下端点可以通过前缀 报告 调用：

```
braintree://plan/report?[options]
```

端点	简写别名	选项	结果正文类型
transactionLevelFees		request	com.braintreegateway.Result<com.braintreegateway.TransactionLevelFeeReport>

报告的URI 选项

Name	类型
request	com.braintreegateway.TransactionLevelFeeReportRequest

54.5.14. 端点前缀settlementBatchSummary

以下端点可以通过前缀settlementBatchSummary 调用，如下所示：

```
braintree://settlementBatchSummary/endpoint?[options]
```

端点	简写别名	选项	结果正文类型
generate		request	com.braintreegateway.Result<com.braintreegateway.SettlementBatchSummary>

settlement BatchSummary 的URI 选项

Name	类型
settlementDate	日历

Name	类型
groupByCustomField	字符串

54.5.15. 端点前缀 订阅

以下端点可以通过前缀 订阅 调用：

> ***braintree://subscription/endpoint?[options]***

端点	简写别名	选项	结果正文类型
取消		id	com.braintreegateway.Result<com.braintreegateway.Subscription>
create		request	com.braintreegateway.Result<com.braintreegateway.Subscription>
delete		customerId, id	com.braintreegateway.Result<com.braintreegateway.Subscription>
查找		id	com.braintreegateway.Subscription
retryCharge		subscriptionId, amount	com.braintreegateway.Result<com.braintreegateway.Transaction>
search		searchRequest	com.braintreegateway.ResourceCollection<com.braintreegateway.Subscription>
update		id, request	com.braintreegateway.Result<com.braintreegateway.Subscription>

订阅的URI 选项

Name	类型
id	字符串

Name	类型
request	com.braintreegateway.SubscriptionRequest
customerId	字符串
subscriptionId	字符串
amount	BigDecimal
searchRequest	com.braintreegateway.SubscriptionSearchRequest.

54.5.16. 端点前缀 事务

以下端点可以通过前缀 事务 调用：

> ***braintree://transaction/endpoint?[options]***

端点	简写别名	选项	结果正文类型
cancelRelease		id	com.braintreegateway.Result<com.braintreegateway.Transaction>
cloneTransaction		ID, cloneRequest	com.braintreegateway.Result<com.braintreegateway.Transaction>
credit		request	com.braintreegateway.Result<com.braintreegateway.Transaction>
查找		id	com.braintreegateway.Transaction
holdInEscrow		id	com.braintreegateway.Result<com.braintreegateway.Transaction>
releaseFromEscrow		id	com.braintreegateway.Result<com.braintreegateway.Transaction>

端点	简写别名	选项	结果正文类型
退款		id, amount, refundRequest	com.braintreegateway.Result<com.braintreegateway.Transaction>
销售		request	com.braintreegateway.Result<com.braintreegateway.Transaction>
search		query	com.braintreegateway.ResourceCollection<com.braintreegateway.Transaction>
submitForPartialSettlement		id, amount	com.braintreegateway.Result<com.braintreegateway.Transaction>
submitForSettlement		id, amount, request	com.braintreegateway.Result<com.braintreegateway.Transaction>
voidTransaction		id	com.braintreegateway.Result<com.braintreegateway.Transaction>

事务的 URI 选项

Name	类型
id	字符串
request	com.braintreegateway.TransactionCloneRequest
cloneRequest	com.braintreegateway.TransactionCloneRequest
refundRequest	com.braintreegateway.TransactionRefundRequest
amount	BigDecimal
query	com.braintreegateway.TransactionSearchRequest

54.5.17. 端点前缀 webhookNotification

以下端点可以使用前缀 webhookNotification 调用，如下所示：

```
braintree://webhookNotification/endpoint?[options]
```

端点	简写别名	选项	结果正文类型
parse		签名、有效负载	com.braintreegateway.WebhookNotification
验证		挑战	字符串

WebhookNotification的URI选项

Name	类型
签名	字符串
payload	字符串
挑战	字符串

54.6. 消费者端点

任何制作者端点都可以用作消费者端点。消费者端点可以使用Scheduled Poll Consumer Options和consumer.前缀来调度端点调用。默认情况下，返回数组或集合的Consumer端点将为每个元素生成一个交换，它们的路由会为每个交换执行一次。要更改此行为，请使用属性consumer.splitResults=true返回整个列表或数组的单一交换。

54.7. 消息标头

任何URI选项可以在生成者端点的消息标头中提供，并带有CamelBraintree.前缀。

54.8. 消息正文

所有结果消息正文都使用Braintree Java SDK提供的对象。生产者端点可以在inBody端点参数中指定传入消息正文的选项名称。

54.9. 例子

Blueprint

```
<?xml version="1.0"?>
```

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:cm="http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.0.0"
          xsi:schemaLocation="
            http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.0.0
http://aries.apache.org/schemas/blueprint-cm/blueprint-cm-1.0.0.xsd
            http://www.osgi.org/xmlns/blueprint/v1.0.0
https://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd
            http://camel.apache.org/schema/blueprint http://camel.apache.org/schema/blueprint/camel-
blueprint.xsd">


    <cm:property-placeholder id="placeholder" persistent-id="camel.braintree">
    </cm:property-placeholder>

    <bean id="braintree" class="org.apache.camel.component.braintree.BraintreeComponent">
        <property name="configuration">
            <bean class="org.apache.camel.component.braintree.BraintreeConfiguration">
                <property name="environment" value="${environment}"/>
                <property name="merchantId" value="${merchantId}"/>
                <property name="publicKey" value="${publicKey}"/>
                <property name="privateKey" value="${privateKey}"/>
            </bean>
        </property>
    </bean>

    <camelContext trace="true" xmlns="http://camel.apache.org/schema/blueprint" id="braintree-
example-context">
        <route id="braintree-example-route">
            <from uri="direct:generateClientToken"/>
            <to uri="braintree://clientToken/generate"/>
            <to uri="stream:out"/>
        </route>
    </camelContext>

</blueprint>
```

54.10. 另请参阅

* 配置 Camel * Component * Endpoint * Getting Started

第 55 章 浏览组件

从 Camel 版本 1.3 开始提供

Browse 组件提供一个简单的 BrowsableEndpoint，可用于测试、可视化工具或调试。发送到端点的交换都可用于浏览。

55.1. URI 格式

```
browse:someName[?options]
```

其中 someName 可以是任意字符串来唯一标识端点。

55.2. 选项

Browse 组件没有选项。

Browse 端点使用 URI 语法进行配置：

```
browse:name
```

使用以下路径和查询参数：

55.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	**必需的** A 名称，可以是任意字符串来唯一标识端点		字符串

55.2.2. 查询参数(4 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外，该处理程序记录在 WARN/ERROR 级别并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，该例外记录在 WARN/ERROR 级别并忽略。		ExceptionHandler
exchangePattern (consumer)	在创建交换时设置默认交换模式。		ExchangePattern

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

55.3. 示例

在以下路由中，我们插入一个 浏览： 组件以浏览要通过的交换：

```
from("activemq:order.in").to("browse:orderReceived").to("bean:processOrder");
```

现在，我们可以从 Java 代码中检查接收的交换：

```
private CamelContext context;

public void inspectRecievedOrders() {
    BrowsableEndpoint browse = context.getEndpoint("browse:orderReceived",
BrowsableEndpoint.class);
    List<Exchange> exchanges = browse.getExchanges();

    // then we can inspect the list of received exchanges from Java
    for (Exchange exchange : exchanges) {
        String payload = exchange.getIn().getBody();
        // do something with payload
    }
}
```

55.4. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 56 章 EHCACHE 组件（已弃用）

从 Camel 版本 2.1 开始提供

缓存 组件可让您使用 EHCache 作为缓存实现来执行缓存操作。缓存本身是按需创建的，或者当该名称的缓存已存在时，只需将其原始设置搭配使用。

此组件支持基于制作者和事件的消费者端点。

缓存消费者是基于事件的消费者，可用于侦听和响应特定的缓存活动。如果您需要从预先存在的缓存中执行选择，请使用为缓存组件定义的处理器。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-cache</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

56.1. URI 格式

```
cache://cacheName[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option =value&option=""beanRef&...

56.2. 选项

EHCache 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
cacheManagerFactory (advanced)	使用给定的 CacheManagerFactory 创建 CacheManager。默认情况下，使用 DefaultCacheManagerFactory。		CacheManagerFactory
configuration (common)	设置缓存配置		CacheConfiguration
configurationFile (common)	设置 ehcache.xml 文件的位置，以便从 classpath 或文件系统加载。默认情况下，该文件从 classpath:ehcache.xml 加载	classpath:ehcache.xml	字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

EHCache 端点使用 URI 语法进行配置：

-

cache:cacheName

使用以下路径和查询参数：

56.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	缓存 **所需的** 名称		字符串

56.2.2. 查询参数(19 参数)：

Name	描述	默认值	类型
diskExpiryThreadInterval Seconds (common)	磁盘到期线程运行间隔的秒数。		long
diskPersistent (common)	磁盘存储是否在应用程序重启之间保留。	false	布尔值
diskStorePath (common)	**弃用** 此参数将被忽略。CacheManager 使用 setter 注入来设置它。		字符串
eternal (common)	设置元素是否结束。如果 eternal，则忽略超时，并且元素永远不会过期。	false	布尔值
key (common)	要使用的默认密钥。如果在消息标头中提供了密钥，则标题中的键具有优先权。		字符串
maxElementsInMemory (common)	可以在内存中定义的缓存中存储的元素数。	1000	int
memoryStoreEvictionPolicy (common)	达到内存中元素数时要使用的驱除策略。策略定义要删除的元素。LRU - 最近使用 LFU - 最低频率使用 FIFO - 优先选择第一出	LFU	MemoryStoreEviction Policy
objectCache (common)	是否打开允许将非序列化对象存储在缓存中。如果启用了这个选项，则无法启用磁盘溢出。	false	布尔值
operation (common)	要使用的默认缓存操作。如果消息标头中的操作，则标题中的操作具有优先权。		字符串
overflowToDisk (common)	指定缓存是否可能会在磁盘中溢出	true	布尔值
timeToIdleSeconds (common)	元素在某一元素到期前访问的最大时间	300	long

Name	描述	默认值	类型
timeToLiveSeconds (common)	创建时间和某一元素过期之间的最长时间。仅在元素不是 eternal 时使用	300	long
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
cacheLoaderRegistry (advanced)	使用 CacheLoaderRegistry 配置缓存加载程序		CacheLoaderRegistry
cacheManagerFactory (advanced)	使用自定义 CacheManagerFactory 来创建由此端点使用的 CacheManager。默认情况下，使用组件上配置的 CacheManagerFactory。		CacheManagerFactory
eventListenerRegistry (advanced)	使用 CacheEventListenerRegistry 配置事件监听程序		CacheEventListener Registry
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

56.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 17 个选项，如下所列。

Name	描述	默认值	类型
camel.component.cache.cache-manager-factory	使用给定的 CacheManagerFactory 创建 CacheManager。默认情况下，使用 DefaultCacheManagerFactory。选项是 org.apache.camel.component.cache.CacheManagerFactory 类型。		字符串

Name	描述	默认值	类型
camel.component.cache.configuration-file	设置 ehcache.xml 文件的位置，以便从 classpath 或文件系统加载。默认情况下，该文件从 classpath:ehcache.xml 加载	classpath:ehcache.xml	字符串
camel.component.cache.configuration.cache-loader-registry	使用 CacheLoaderRegistry 配置缓存加载程序		CacheLoaderRegistry
camel.component.cache.configuration.cache-name	缓存的名称		字符串
camel.component.cache.configuration.disk-expiry-thread-interval-seconds	磁盘到期线程运行间隔的秒数。		Long
camel.component.cache.configuration.disk-persistent	磁盘存储是否在应用程序重启之间保留。	false	布尔值
camel.component.cache.configuration.eternal	设置元素是否结束。如果 eternal，则忽略超时，并且元素永远不会过期。	false	布尔值
camel.component.cache.configuration.event-listener-registry	使用 CacheEventListenerRegistry 配置事件监听程序		CacheEventListener Registry
camel.component.cache.configuration.max-elements-in-memory	可以在内存中定义的缓存中存储的元素数。	1000	整数
camel.component.cache.configuration.memory-store-eviction-policy	达到内存中元素数时要使用的驱除策略。策略定义要删除的元素。LRU - 最近使用 LFU - 最低频率使用 FIFO - 优先选择第一出		MemoryStoreEviction Policy
camel.component.cache.configuration.object-cache	是否打开允许将非序列化对象存储在缓存中。如果启用了这个选项，则无法启用磁盘溢出。	false	布尔值

Name	描述	默认值	类型
camel.component.cache.configuration.overflow-to-disk	指定缓存是否可能会在磁盘中溢出	true	布尔值
camel.component.cache.configuration.time-to-idle-seconds	元素在某一元素到期前访问的最大时间	300	Long
camel.component.cache.configuration.time-to-live-seconds	创建时间和某一元素过期之间的最长时间。仅在元素不是 eternal 时使用	300	Long
camel.component.cache.enabled	启用缓存组件	true	布尔值
camel.component.cache.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.cache.configuration.disk-store-path	这个参数会被忽略。CacheManager 使用 setter 注入来设置它。		字符串

56.4. 将消息发送到/从缓存中发送/接收消息

56.4.1. 消息标头最多 Camel 2.7

标头	描述
CACHE_OPERATION	在缓存上执行的操作。有效选项包括 * GET * CHECK * ADD * UPDATE * DELETEALL **GET** 和 **CHECK** 需要 Camel 2.3 以后。
CACHE_KEY	用于将消息存储在缓存中的缓存密钥。如果 CACHE_OPERATION 为 DELETEALL，则缓存键是可选的

56.4.2. Message Headers Camel 2.8+

Camel 2.8 中的标头更改

标头名称和支持的值已更改为以 'CamelCache' 前缀，并使用混合问题单。这使得它们更易于识别并与其他标头分开。CacheConstants 变量名称保持不变，只有其值已被更改。另外，这些标头现在在执行缓存操作后从交换中删除。

标头	描述
CamelCacheOperation	在缓存上执行的操作。有效选项包括 * CamelCacheGet * CamelCacheCheck * CamelCacheAdd * CamelCacheUpdate * CamelCacheDelete * CamelCacheDeleteAll
CamelCacheKey	用于将消息存储在缓存中的缓存密钥。如果 CamelCacheOperation 为 CamelCacheDeleteAll，则缓存密钥是可选的

CamelCacheAdd 和 CamelCacheUpdate 操作支持额外的标头：

标头	类型	描述
CamelCacheTimeToLive	**整数**	**Camel 2.11：** 到实时的时间（以秒为单位）。
CamelCacheTimeToIdle	**整数**	**Camel 2.11 ：** 以秒数为单位闲置的时间。
CamelCacheEternal	**布尔值**	**Camel 2.11：** 内容是否是eternal。

56.4.3. 缓存生成者

向缓存发送数据涉及将有效负载直接存储在预先存在的或创建时所需的缓存中。执行此操作的机制涉及

- ***设置上述显示的消息交换标头。***
- ***确保 Message Exchange Body 包含定向到缓存的消息***

56.4.4. cache Consumer

从缓存接收数据涉及 CacheConsumer 在任何缓存活动发生时（即 CamelCacheGet/CamelCacheUpdate/CamelCacheDelete/CamelCacheDelete/CamelCacheDelete/Ca 侦听预存在的缓存。发生此类活动后

- ***将放置包含消息交换标头和包含刚刚添加/更新有效负载的消息交换正文的交换。***
- ***如果是 CamelCacheDeleteAll 操作，则不会填充 Message Exchange Header CamelCacheKey 和 Message Exchange Body。***

56.4.5. 缓存处理器

有一组可执行缓存查找和有选择地替换有效负载内容的 nice 处理器

- 正文（body）
- token
- XPath 级别

56.5. 缓存使用示例

56.5.1. 示例 1：配置缓存

```
from("cache://MyApplicationCache" +
        "?maxElementsInMemory=1000" +
        "&memoryStoreEvictionPolicy=" +
           "MemoryStoreEvictionPolicy.LFU" +
        "&overflowToDisk=true" +
        "&eternal=true" +
        "&timeToLiveSeconds=300" +
        "&timeToIdleSeconds=true" +
        "&diskPersistent=true" +
        "&diskExpiryThreadIntervalSeconds=300")
```

56.5.2. 示例 2：在缓存中添加密钥

```
RouteBuilder builder = new RouteBuilder() {
   public void configure() {
    from("direct:start")
    .setHeader(CacheConstants.CACHE_OPERATION,
constant(CacheConstants.CACHE_OPERATION_ADD))
    .setHeader(CacheConstants.CACHE_KEY, constant("Ralph_Waldo_Emerson"))
    .to("cache://TestCache1")
  }
};
```

56.5.3. 示例 2：更新缓存中的现有密钥

```
RouteBuilder builder = new RouteBuilder() {
   public void configure() {
    from("direct:start")
    .setHeader(CacheConstants.CACHE_OPERATION,
constant(CacheConstants.CACHE_OPERATION_UPDATE))
    .setHeader(CacheConstants.CACHE_KEY, constant("Ralph_Waldo_Emerson"))
    .to("cache://TestCache1")
  }
};
```

56.5.4. 示例 3：删除缓存中的现有密钥

```
RouteBuilder builder = new RouteBuilder() {
```

```
    public void configure() {
     from("direct:start")
     .setHeader(CacheConstants.CACHE_OPERATION,
constant(CacheConstants.CACHE_DELETE))
     .setHeader(CacheConstants.CACHE_KEY", constant("Ralph_Waldo_Emerson"))
     .to("cache://TestCache1")
    }
};
```

56.5.5. 示例 4：删除缓存中的所有现有密钥

```
RouteBuilder builder = new RouteBuilder() {
    public void configure() {
     from("direct:start")
     .setHeader(CacheConstants.CACHE_OPERATION,
constant(CacheConstants.CACHE_DELETEALL))
     .to("cache://TestCache1");
    }
};
```

56.5.6. 示例 5：通知缓存中注册到处理器和其他 Producers 的任何更改

```
RouteBuilder builder = new RouteBuilder() {
    public void configure() {
     from("cache://TestCache1")
     .process(new Processor() {
        public void process(Exchange exchange)
              throws Exception {
           String operation = (String)
exchange.getIn().getHeader(CacheConstants.CACHE_OPERATION);
           String key = (String) exchange.getIn().getHeader(CacheConstants.CACHE_KEY);
           Object body = exchange.getIn().getBody();
           // Do something
        }
     })
    }
};
```

56.5.7. 示例 6：使用处理器有选择地将 payload 替换为缓存值

```
RouteBuilder builder = new RouteBuilder() {
  public void configure() {
    //Message Body Replacer
    from("cache://TestCache1")
    .filter(header(CacheConstants.CACHE_KEY).isEqualTo("greeting"))
    .process(new CacheBasedMessageBodyReplacer("cache://TestCache1","farewell"))
    .to("direct:next");

   //Message Token replacer
   from("cache://TestCache1")
   .filter(header(CacheConstants.CACHE_KEY).isEqualTo("quote"))
   .process(new CacheBasedTokenReplacer("cache://TestCache1","novel","#novel#"))
   .process(new CacheBasedTokenReplacer("cache://TestCache1","author","#author#"))
```

```
    .process(new CacheBasedTokenReplacer("cache://TestCache1","number","#number#"))
    .to("direct:next");

    //Message XPath replacer
    from("cache://TestCache1").
    .filter(header(CacheConstants.CACHE_KEY).isEqualTo("XML_FRAGMENT"))
    .process(new CacheBasedXPathReplacer("cache://TestCache1","book1","/books/book1"))
    .process (new CacheBasedXPathReplacer("cache://TestCache1","book2","/books/book2"))
    .to("direct:next");
   }
};
```

56.5.8. 示例 7：从缓存获取条目

```
from("direct:start")
    // Prepare headers
    .setHeader(CacheConstants.CACHE_OPERATION,
constant(CacheConstants.CACHE_OPERATION_GET))
    .setHeader(CacheConstants.CACHE_KEY, constant("Ralph_Waldo_Emerson")).
    .to("cache://TestCache1").
    // Check if entry was not found
    .choice().when(header(CacheConstants.CACHE_ELEMENT_WAS_FOUND).isNull()).
        // If not found, get the payload and put it to cache
        .to("cxf:bean:someHeavyweightOperation").
        .setHeader(CacheConstants.CACHE_OPERATION,
constant(CacheConstants.CACHE_OPERATION_ADD))
        .setHeader(CacheConstants.CACHE_KEY, constant("Ralph_Waldo_Emerson"))
        .to("cache://TestCache1")
    .end()
    .to("direct:nextPhase");
```

56.5.9. 示例 8：检查缓存中的条目

注意：CHECK 命令测试在缓存中存在一个条目，但不会在正文中放置一条消息。

```
from("direct:start")
    // Prepare headers
    .setHeader(CacheConstants.CACHE_OPERATION,
constant(CacheConstants.CACHE_OPERATION_CHECK))
    .setHeader(CacheConstants.CACHE_KEY, constant("Ralph_Waldo_Emerson")).
    .to("cache://TestCache1").
    // Check if entry was not found
    .choice().when(header(CacheConstants.CACHE_ELEMENT_WAS_FOUND).isNull()).
        // If not found, get the payload and put it to cache
        .to("cxf:bean:someHeavyweightOperation").
        .setHeader(CacheConstants.CACHE_OPERATION,
constant(CacheConstants.CACHE_OPERATION_ADD))
        .setHeader(CacheConstants.CACHE_KEY, constant("Ralph_Waldo_Emerson"))
        .to("cache://TestCache1")
    .end();
```

56.6. EHCACHE 的管理

Eh cache 具有自己的统计信息，并从 JMX 管理。

以下是如何在 Spring 应用程序上下文中通过 JMX 公开它们的代码片段：

```
<bean id="ehCacheManagementService" class="net.sf.ehcache.management.ManagementService"
init-method="init" lazy-init="false">
  <constructor-arg>
    <bean class="net.sf.ehcache.CacheManager" factory-method="getInstance"/>
  </constructor-arg>
  <constructor-arg>
    <bean class="org.springframework.jmx.support.JmxUtils" factory-method="locateMBeanServer"/>
  </constructor-arg>
  <constructor-arg value="true"/>
  <constructor-arg value="true"/>
  <constructor-arg value="true"/>
  <constructor-arg value="true"/>
</bean>
```

当然，您可以在直接 Java 中执行同样的操作：

```
ManagementService.registerMBeans(CacheManager.getInstance(), mbeanServer, true, true,
true, true);
```

您可以以这种方式获得缓存命中、未命中、内存命中、磁盘命中和大小统计。您还可以实时更改 CacheConfiguration 参数。

56.7. 缓存复制 CAMEL 2.8

Camel 缓存组件可以使用几种不同的复制机制在服务器节点之间分发缓存，包括：RMI、JGroups、JMS 和 Cache Server。

可以通过两种方式使其工作：

1.您可以手动配置 **ehcache.xml**

或者

2.您可以配置这三个选项：

- cacheManagerFactory
- eventListenerRegistry
- cacheLoaderRegistry

使用第一个选项配置 Camel 缓存复制是一些硬工作，因为您必须单独配置所有缓存。因此，当所有缓存名称都不知道时，使用 **ehcache.xml** 并不是一个好主意。

当您想要使用多个不同的缓存时，第二个选项更好，因为您不需要定义每个缓存的选项。这是因为复制选项是针对每个 **CacheManager** 和每个 **CacheEndpoint** 设置的。另外，这是在开发阶段不知道缓存名称的唯一方法。

注意 ：读取 EHCache 手册 以更好地了解 Camel 缓存复制机制时可能很有用。

56.7.1. 示例：JMS 缓存复制

JMS 复制是最强大且安全的复制方法。与 Camel 缓存复制一同使用使得它也很简单。示例位于 一个单独的页面 中。

第 57 章 CAFFEINE CACHE COMPONENT

从 Camel 版本 2.20 开始提供

caffeine-cache 组件允许您使用 Caffeine 的简单缓存执行缓存操作。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-caffeine</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

57.1. URI 格式

```
caffeine-cache://cacheName[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option =value&option=""beanRef&...

57.2. 选项

Caffeine 缓存组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	设置全局组件配置		CaffeineConfiguration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Caffeine Cache 端点使用 URI 语法进行配置：

```
caffeine-cache:cacheName
```

使用以下路径和查询参数：

57.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	**所需的** 缓存名称		字符串

57.2.2. 查询参数(19 参数)：

Name	描述	默认值	类型
createCacheIfNotExist (common)	如果缓存存在或无法预先配置，则配置是否需要创建缓存。	true	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
action (producer)	配置默认缓存操作。如果在消息标头中设置了操作，则标题中的操作将具有优先权。		字符串
cache (producer)	配置要使用的已实例化缓存		Cache
cacheLoader (producer)	在 LoadCache 使用时配置 CacheLoader		CacheLoader
evictionType (producer)	为此缓存设置驱除类型	SIZE_BASED	EvictionType
expireAfterAccessTime (producer)	在基于时间的驱除（以秒为单位）时设置过期时间	300	int
expireAfterWriteTime (producer)	在基于时间的 Eviction 时设置过期后访问写（以秒为单位）	300	int
initialCapacity (producer)	为缓存设置初始容量	10000	int
key (producer)	配置默认操作密钥。如果在消息标头中设置了键，则标题中的键具有优先权。		对象
maximumSize (producer)	为缓存设置最大大小	10000	int
removalListener (producer)	为缓存设置特定的删除 Listener		RemovalListener

Name	描述	默认值	类型
statsCounter (producer)	为缓存统计设置特定的 Stats Counter		StatsCounter
statsEnabled (producer)	在缓存上启用统计信息	false	布尔值
keyType (advanced)	缓存密钥类型，默认 java.lang.Object	java.lang.Object	字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
valueType (advanced)	缓存值类型，默认为 java.lang.Object	java.lang.Object	字符串

57.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 17 个选项，如下所列。

Name	描述	默认值	类型
camel.component.caffeine-cache.configuration.action	配置默认缓存操作。如果在消息标头中设置了操作，则标题中的操作将具有优先权。		字符串
camel.component.caffeine-cache.configuration.cache	配置要使用的已实例化缓存		Cache
camel.component.caffeine-cache.configuration.cache-loader	在 LoadCache 使用时配置 CacheLoader		CacheLoader
camel.component.caffeine-cache.configuration.create-cache-if-not-exist	如果缓存存在或无法预先配置，则配置是否需要创建缓存。	true	布尔值

Name	描述	默认值	类型
camel.component.caffeine-cache.configuration.eviction-type	为此缓存设置驱除类型		EvictionType
camel.component.caffeine-cache.configuration.expire-after-access-time	在基于时间的驱除（以秒为单位）时设置过期时间	300	整数
camel.component.caffeine-cache.configuration.expire-after-write-time	在基于时间的 Eviction 时设置过期后访问写（以秒为单位）	300	整数
camel.component.caffeine-cache.configuration.initial-capacity	为缓存设置初始容量	10000	整数
camel.component.caffeine-cache.configuration.key	配置默认操作密钥。如果在消息标头中设置了键，则标题中的键具有优先权。		对象
camel.component.caffeine-cache.configuration.key-type	缓存密钥类型，默认 java.lang.Object	java.lang.Object	字符串
camel.component.caffeine-cache.configuration.maximum-size	为缓存设置最大大小	10000	整数
camel.component.caffeine-cache.configuration.removal-listener	为缓存设置特定的删除 Listener		RemovalListener
camel.component.caffeine-cache.configuration.stats-counter	为缓存统计设置特定的 Stats Counter		StatsCounter

Name	描述	默认值	类型
camel.component.caffeine-cache.configuration.stats-enabled	在缓存上启用统计信息	false	布尔值
camel.component.caffeine-cache.configuration.value-type	缓存值类型，默认为 java.lang.Object	java.lang.Object	字符串
camel.component.caffeine-cache.enabled	是否启用 caffeine-cache 组件的自动配置。这默认是启用的。		布尔值
camel.component.caffeine-cache.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

57.4. 例子

您可以将缓存与以下代码一起使用：

```
@Override
protected RouteBuilder createRouteBuilder() throws Exception {
    return new RouteBuilder() {
        public void configure() {
            from("direct://start")
                .toF("caffeine-cache://%s?cache=#cache&action=PUT&key=1", "test")
                .toF("caffeine-cache://%s?cache=#cache&key=1&action=GET", "test")
                .log("Test! ${body}")
                .to("mock:result");
        }
    };
}
```

这样，您将始终在 registry 中的同一缓存中工作。

57.5. 检查操作结果

每次您将在缓存中使用操作时，您将有两个不同的标头来检查状态：

```
CaffeineConstants.ACTION_HAS_RESULT
CaffeineConstants.ACTION_SUCCEEDED
```

第 58 章 CAFFEINE LOADCACHE 组件

从 Camel 版本 2.20 开始提供

caffeine-loadcache 组件允许您使用 Caffeine 中的 The Load cache perform cache operations。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-caffeine</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

58.1. URI 格式

```
caffeine-loadcache://cacheName[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option =value&option=""beanRef&...

58.2. 选项

Caffeine LoadCache 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
配置（高级）	设置全局组件配置		CaffeineConfiguration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Caffeine LoadCache 端点使用 URI 语法进行配置：

```
caffeine-loadcache:cacheName
```

使用以下路径和查询参数：

58.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	**所需的** 缓存名称		字符串

58.2.2. 查询参数(19 参数)：

Name	描述	默认值	类型
createCacheIfNotExist (common)	如果缓存存在或无法预先配置，则配置是否需要创建缓存。	true	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
action (producer)	配置默认缓存操作。如果在消息标头中设置了操作，则标题中的操作将具有优先权。		字符串
cache (producer)	配置要使用的已实例化缓存		Cache
cacheLoader (producer)	在 LoadCache 使用时配置 CacheLoader		CacheLoader
evictionType (producer)	为此缓存设置驱除类型	SIZE_BASED	EvictionType
expireAfterAccessTime (producer)	在基于时间的驱除（以秒为单位）时设置过期时间	300	int
expireAfterWriteTime (producer)	在基于时间的 Eviction 时设置过期后访问写（以秒为单位）	300	int
initialCapacity (producer)	为缓存设置初始容量	10000	int
key (producer)	配置默认操作密钥。如果在消息标头中设置了键，则标题中的键具有优先权。		对象
maximumSize (producer)	为缓存设置最大大小	10000	int
removalListener (producer)	为缓存设置特定的删除 Listener		RemovalListener

Name	描述	默认值	类型
statsCounter (producer)	为缓存统计设置特定的 Stats Counter		StatsCounter
statsEnabled (producer)	在缓存上启用统计信息	false	布尔值
keyType (advanced)	缓存密钥类型，默认 java.lang.Object	java.lang.Object	字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
valueType (advanced)	缓存值类型，默认为 java.lang.Object	java.lang.Object	字符串

58.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 17 个选项，如下所列。

Name	描述	默认值	类型
camel.component.caffeine-loadcache.configuration.action	配置默认缓存操作。如果在消息标头中设置了操作，则标题中的操作将具有优先权。		字符串
camel.component.caffeine-loadcache.configuration.cache	配置要使用的已实例化缓存		Cache
camel.component.caffeine-loadcache.configuration.cache-loader	在 LoadCache 使用时配置 CacheLoader		CacheLoader
camel.component.caffeine-loadcache.configuration.create-cache-if-not-exist	如果缓存存在或无法预先配置，则配置是否需要创建缓存。	true	布尔值

Name	描述	默认值	类型
camel.component.caffeine-loadcache.configuration.eviction-type	为此缓存设置驱除类型		EvictionType
camel.component.caffeine-loadcache.configuration.expire-after-access-time	在基于时间的驱除（以秒为单位）时设置过期时间	300	整数
camel.component.caffeine-loadcache.configuration.expire-after-write-time	在基于时间的 Eviction 时设置过期后访问写（以秒为单位）	300	整数
camel.component.caffeine-loadcache.configuration.initial-capacity	为缓存设置初始容量	10000	整数
camel.component.caffeine-loadcache.configuration.key	配置默认操作密钥。如果在消息标头中设置了键，则标题中的键具有优先权。		对象
camel.component.caffeine-loadcache.configuration.key-type	缓存密钥类型，默认 java.lang.Object	java.lang.Object	字符串
camel.component.caffeine-loadcache.configuration.maximum-size	为缓存设置最大大小	10000	整数
camel.component.caffeine-loadcache.configuration.removal-listener	为缓存设置特定的删除 Listener		RemovalListener

Name	描述	默认值	类型
camel.component.caffeine-loadcache.configuration.stats-counter	为缓存统计设置特定的 Stats Counter		StatsCounter
camel.component.caffeine-loadcache.configuration.stats-enabled	在缓存上启用统计信息	false	布尔值
camel.component.caffeine-loadcache.configuration.value-type	缓存值类型，默认为 java.lang.Object	java.lang.Object	字符串
camel.component.caffeine-loadcache.enabled	是否启用 caffeine-loadcache 组件的自动配置。这默认是启用的。		布尔值
camel.component.caffeine-loadcache.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 59 章 CASTOR DATAFORMAT （已弃用）

从 Camel 版本 2.1 开始提供

Castor 是一个 Data Format，它使用 Castor XML 库将 XML 有效负载 unmarshal 到 Java 对象，或将 Java 对象嵌套到 XML 有效负载中。

通常，您可以使用 Java DSL 或 Spring XML 来使用 Castor Data Format。

59.1. 使用 JAVA DSL

```
from("direct:order").
  marshal().castor().
  to("activemq:queue:order");
```

例如，以下命令使用名为 Castor 的名为 DataFormat，它使用默认的 Castor 数据绑定功能。

```
CastorDataFormat castor = new CastorDataFormat ();

from("activemq:My.Queue").
  unmarshal(castor).
  to("mqseries:Another.Queue");
```

如果您希望使用命名引用的数据格式，然后在 Registry 中定义，如通过您的 Spring XML 文件。

```
from("activemq:My.Queue").
  unmarshal("mycastorType").
  to("mqseries:Another.Queue");
```

如果要通过提供映射文件来覆盖默认的映射模式，您可以进行如下设置：

```
CastorDataFormat castor = new CastorDataFormat ();
castor.setMappingFile("mapping.xml");
```

另外，如果您要对 Castor Marshaller 和 Unmarshaller 具有更多控制，您可以通过以下方式访问它们。

```
castor.getMarshaller();
castor.getUnmarshaller();
```

59.2. 使用 SPRING XML

以下示例演示了如何使用 Castor 来使用 Spring 配置 castor 数据类型

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="direct:start"/>
    <unmarshal>
      <castor validation="true" />
    </unmarshal>
    <to uri="mock:result"/>
  </route>
</camelContext>
```

本例演示了如何只配置数据类型一次，并在多个路由中重复使用它。您必须直接在 <camelContext> 中设置 <castor> 元素。

```
<camelContext>
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <dataFormats>
    <castor id="myCastor"/>
  </dataFormats>

  <route>
    <from uri="direct:start"/>
    <marshal ref="myCastor"/>
    <to uri="direct:marshalled"/>
  </route>
  <route>
    <from uri="direct:marshalled"/>
    <unmarshal ref="myCastor"/>
    <to uri="mock:result"/>
  </route>

</camelContext>
```

59.3. 选项

Castor dataformat 支持 9 个选项，如下所列。

Name	默认值	Java 类型	描述
mappingFile		字符串	到 Castor 映射文件的路径，以便从 classpath 加载。
whitelistEnabled	true	布尔值	定义是否启用了 Whitelist 功能
allowedUnmarshallObjects		字符串	定义允许的对象为 unmarshalled。您可以指定允许对象的 FQN 类名称，您可以使用逗号分隔多个条目。也可以使用通配符和正则表达式，它们基于链接 org.apache.camel.util.EndpointHelperBuildmatchPattern (String, String)定义的模式。拒绝的对象优先于允许的对象。
deniedUnmarshallObjects		字符串	将被拒绝的对象定义为 unmarshalled。您可以指定 FQN 类名称 deined 对象，您可以使用逗号分隔多个条目。也可以使用通配符和正则表达式，它们基于链接 org.apache.camel.util.EndpointHelperBuildmatchPattern (String, String)定义的模式。拒绝的对象优先于允许的对象。
验证	true	布尔值	验证是打开还是关闭。默认为 true。
编码	UTF-8	字符串	将对象聚合到 XML 时使用的编码。默认是 UTF-8
软件包		string[]	将其他软件包添加到 Castor XmlContext

Name	默认值	Java 类型	描述
类		**string[]**	向 Castor XmlContext 添加额外的类名称
contentTypeHeader	**false**	**布尔值**	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

59.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.castor.allowed-unmarshall-objects	定义允许的对象为 unmarshalled。您可以指定允许对象的 FQN 类名称，您可以使用逗号分隔多个条目。也可以使用通配符和正则表达式，它们基于链接 org.apache.camel.util.EndpointHelperBuildmatchPattern (String, String)定义的模式。拒绝的对象优先于允许的对象。		字符串
camel.dataformat.castor.classes	向 Castor XmlContext 添加额外的类名称		string[]
camel.dataformat.castor.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.castor.denied-unmarshall-objects	将被拒绝的对象定义为 unmarshalled。您可以指定 FQN 类名称 deined 对象，您可以使用逗号分隔多个条目。也可以使用通配符和正则表达式，它们基于链接 org.apache.camel.util.EndpointHelperBuildmatchPattern (String, String)定义的模式。拒绝的对象优先于允许的对象。		字符串
camel.dataformat.castor.enabled	启用 castor dataformat	true	布尔值
camel.dataformat.castor.encoding	将对象聚合到 XML 时使用的编码。默认是 UTF-8	UTF-8	字符串
camel.dataformat.castor.mapping-file	到 Castor 映射文件的路径，以便从 classpath 加载。		字符串

Name	描述	默认值	类型
camel.dataformat.castor.packages	将其他软件包添加到 Castor XmlContext		string[]
camel.dataformat.castor.validation	验证是打开还是关闭。默认为 true。	true	布尔值
camel.dataformat.castor.whitelist-enabled	定义是否启用了 Whitelist 功能	true	布尔值

ND

59.5. 依赖项

要在 camel 路由中使用 Castor，您需要添加对实现此数据格式的 camel-castor 的依赖。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-castor</artifactId>
  <version>x.x.x</version>
</dependency>
```

第 60 章 CAMEL CDI

Camel CDI 组件使用 CDI 作为依赖项注入框架，基于惯例配置，为 Apache Camel 提供自动配置。它自动检测应用程序中提供的 Camel 路由，并为常见的 Camel 原语提供 Bean，如 **Endpoint**、**FluentProducerTemplate**、**ProducerTemplate** 或 **TypeConverter**。它实施标准的 Camel Bean 集成，以便 Camel 注解（如 **@Consume**、**@Produce** 和 **@PropertyInject** ）可以在 CDI Bean 中无缝使用。此外，它还将 Camel 事件桥接（如 **RouteAddedEvent**, **CamelContextStartedEvent**, **ExchangeCompletedEvent**, ...）作为 CDI 事件，并提供 CDI 事件端点，可用于消耗 / 从 / 到 Camel 路由的 CDI 事件。

尽管 Camel CDI 组件作为 Camel 2.10 提供，但在 Camel 2.17 中已被重写，以更好地适应 CDI 编程模型。因此，一些功能（如 Camel 事件到 CDI 事件网桥）和 CDI 事件端点仅应用启动 Camel 2.17。

有关如何测试 Camel CDI 应用程序的更多详细信息，请参阅 Camel CDI 测试。

小心

camel-cdi 在 OSGi 中已弃用，且不受支持。如果使用带有 OSGi 的 Camel，请使用 OSGi 蓝图。

60.1. 自动配置的 CAMEL 上下文

Camel CDI 会自动部署和配置 **CamelContext** Bean。当 CDI 容器初始化(resp. 关闭)时，CamelContext bean 会自动实例化、配置和启动（停止）。它可以在应用程序中注入，例如：

```
@Inject
CamelContext context;
```

此默认 **CamelContext Bean** 与内置 **@Default qualifier** 限定，范围为 **@ApplicationScoped**，类型为 **DefaultCamelContext**。

请注意，这个 **Bean** 可以以编程方式自定义，其他 **Camel** 上下文 **Bean** 也可以部署到应用程序中。

60.2. 自动探测 CAMEL 路由

Camel CDI 会在应用程序中自动收集所有 **RoutesBuilder Bean**，并在 **CDI** 容器初始化时将它们添加到 **CamelContext bean** 实例中。例如，添加 **Camel** 路由就像声明类一样简单，例如：

```
class MyRouteBean extends RouteBuilder {

    @Override
    public void configure() {
        from("jms:invoices").to("file:/invoices");
    }
}
```

请注意，您可以根据需要声明任意数量的 RoutesBuilder Bean。此外，在容器初始化时，RouteContainer Bean 也会自动收集、实例化并添加到由 Camel CDI 管理的 CamelContext bean 实例中。

从 Camel 2.19 开始提供

在某些情况下，可能需要禁用 RouteBuilder 和 RouteContainer Bean 的自动配置。这可以通过观察 CdiCamelConfiguration 事件来实现，例如：

```
static void configuration(@Observes CdiCamelConfiguration configuration) {
    configuration.autoConfigureRoutes(false);
}
```

同样，也可以取消激活配置的 CamelContext Bean 的自动启动，例如：

```
static void configuration(@Observes CdiCamelConfiguration configuration) {
    configuration.autoStartContexts(false);
}
```

60.3. 自动配置的 CAMEL 原语

Camel CDI 为通用 Camel 原语提供 Bean，可在任何 CDI Bean 中注入，例如：

```
@Inject
@Uri("direct:inbound")
ProducerTemplate producerTemplate;

@Inject
@Uri("direct:inbound")
FluentProducerTemplate fluentProducerTemplate;

@Inject
MockEndpoint outbound; // URI defaults to the member name, i.e. mock:outbound

@Inject
@Uri("direct:inbound")
Endpoint endpoint;

@Inject
TypeConverter converter;
```

60.4. CAMEL 上下文配置

如果您只想更改默认 CamelContext bean 的名称，您可以使用 Camel CDI 提供的 @ContextName qualifier，例如：

```
@ContextName("camel-context")
class MyRouteBean extends RouteBuilder {

    @Override
    public void configure() {
        from("jms:invoices").to("file:/invoices");
    }
}
```

否则，如果需要更多自定义，都可以使用任何 CamelContext 类来声明自定义 Camel 上下文 Bean。然后，可以执行 @PostConstruct 和 @PreDestroy 生命周期回调来进行自定义，例如：

```
@ApplicationScoped
class CustomCamelContext extends DefaultCamelContext {

    @PostConstruct
    void customize() {
        // Set the Camel context name
        setName("custom");
        // Disable JMX
        disableJMX();
    }

    @PreDestroy
    void cleanUp() {
        // ...
    }
}
```

生产者和发布者方法也可用于自定义 Camel 上下文 Bean，例如：
http://docs.jboss.org/cdi/spec/1.2/cdi-spec.html#disposer_method

```
class CamelContextFactory {

    @Produces
    @ApplicationScoped
    CamelContext customize() {
        DefaultCamelContext context = new DefaultCamelContext();
        context.setName("custom");
        return context;
    }

    void cleanUp(@Disposes CamelContext context) {
        // ...
    }
}
```

同样，可以使用 producer 字段，例如：

```
@Produces
@ApplicationScoped
CamelContext context = new CustomCamelContext();

class CustomCamelContext extends DefaultCamelContext {

    CustomCamelContext() {
        setName("custom");
    }
}
```

例如，可以通过调用 setAutoStartup 方法来避免在容器初始化时自动启动 Camel 上下文路由，例如：

```
@ApplicationScoped
class ManualStartupCamelContext extends DefaultCamelContext {

    @PostConstruct
    void manual() {
        setAutoStartup(false);
    }
}
```

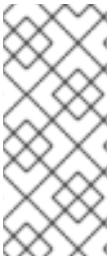
60.5. 多个 CAMEL 上下文

实际可在应用程序中声明任意数量的 CamelContext Bean，如上文所述。在这种情况下，使用这些 CamelContext Bean 上声明的 CDI 限定符用于将 Camel 路由和其他 Camel 原语绑定到对应的 Camel 上下文。在示例中，如果声明了以下 Bean：

```
@ApplicationScoped
@ContextName("foo")
class FooCamelContext extends DefaultCamelContext {
}

@ApplicationScoped
@BarContextQualifier
class BarCamelContext extends DefaultCamelContext {
}

@ContextName("foo")
class RouteAddedToFooCamelContext extends RouteBuilder {

    @Override
    public void configure() {
        // ...
    }
```

```
}

@BarContextQualifier
class RouteAddedToBarCamelContext extends RouteBuilder {

    @Override
    public void configure() {
        // ...
    }
}

@ContextName("baz")
class RouteAddedToBazCamelContext extends RouteBuilder {

    @Override
    public void configure() {
        // ...
    }
}

@MyOtherQualifier
class RouteNotAddedToAnyCamelContext extends RouteBuilder {

    @Override
    public void configure() {
        // ...
    }
}
```

带有 @ContextName 限定的 RoutesBuilder Bean 由 Camel CDI 自动添加到对应的 CamelContext Bean 中。如果没有这样的 CamelContext bean，它会被自动创建，如 RouteAddedToBazCamelContext bean。请注意，这只适用于 Camel CDI 提供的 @ContextName 限定符。因此，RouteNotAddedToAnyCamelContext bean 与用户定义的 @MyOtherQualifier 限定符不会添加到任何 Camel 上下文。例如，这非常有用，例如，在应用程序执行过程中可能需要添加的 Camel 路由。



注意

自 Camel 版本 2.17.0 起，Camel CDI 能够管理任何类型的 CamelContext Bean （如 DefaultCamelContext）。在以前的版本中，它只能管理类型为 CdiCamelContext 的 Bean，因此需要扩展它。

CamelContext Bean 上声明的 CDI 限定符也用于绑定对应的 Camel 原语，例如：

```
@Inject
@ContextName("foo")
@Uri("direct:inbound")
ProducerTemplate producerTemplate;
```

```
@Inject
@ContextName("foo")
@Uri("direct:inbound")
FluentProducerTemplate fluentProducerTemplate;

@Inject
@BarContextQualifier
MockEndpoint outbound; // URI defaults to the member name, i.e. mock:outbound

@Inject
@ContextName("baz")
@Uri("direct:inbound")
Endpoint endpoint;
```

60.6. 配置属性

要配置 Camel 用来解析属性占位符的配置属性的源，您可以使用 @Named ("properties")声明符合 @Named ("properties") 的 PropertiesComponent bean，例如：

```
@Produces
@ApplicationScoped
@Named("properties")
PropertiesComponent propertiesComponent() {
    Properties properties = new Properties();
    properties.put("property", "value");
    PropertiesComponent component = new PropertiesComponent();
    component.setInitialProperties(properties);
    component.setLocation("classpath:placeholder.properties");
    return component;
}
```

如果要使用 DeltaSpike 配置机制，您可以声明以下 PropertiesComponent bean ：

```
@Produces
@ApplicationScoped
@Named("properties")
PropertiesComponent properties(PropertiesParser parser) {
    PropertiesComponent component = new PropertiesComponent();
    component.setPropertiesParser(parser);
    return component;
}

// PropertiesParser bean that uses DeltaSpike to resolve properties
static class DeltaSpikeParser extends DefaultPropertiesParser {
    @Override
    public String parseProperty(String key, String value, Properties properties) {
        return ConfigResolver.getPropertyValue(key);
    }
}
```

您可以使用 DeltaSpike 配置机制查看 camel-example-cdi-properties 示例。

60.7. 自动配置的类型转换器

使用 @Converter 注释标注的 CDI Bean 会自动注册到部署的 Camel 上下文中，例如：

```
@Converter
public class MyTypeConverter {

    @Converter
    public Output convert(Input input) {
        //...
    }
}
```

请注意，在类型转换器中支持 CDI 注入。

60.8. CAMEL BEAN 集成

60.8.1. Camel 注解

作为 Camel Bean 集成 的一部分，Camel 附带了一组由 Camel CDI 无缝支持 的注解。因此，您可以在 CDI Bean 中使用任何这些注解，例如：

Camel 注解		CDI 等效

配置属性	<pre>@PropertyInject("key")
	String value;</pre>
		如果使用 DeltaSpike 配置机制 ：
		<pre>@Inject
		@ConfigProperty(name = "key")
		String value;</pre>
		如需了解更多详细信息，请参阅配置属性。

生成者模板注入（默认 Camel 上下文）	<pre>@Produce(uri = "mock:outbound")
	ProducerTemplate producer;
	
	@Produce(uri = "mock:outbound")
	FluentProducerTemplate producer;</pre>
		<pre>@Inject
		@Uri("direct:outbound")
		ProducerTemplate producer;
		
		@Produce(uri = "direct:outbound")
		FluentProducerTemplate producer;</pre>

	Camel 注解	CDI 等效

端点注入（默认 Camel 上下文）	`@EndpointInject(uri = "direct:inbound")`
	`Endpoint endpoint;`
		`@Inject`
		`@Uri("direct:inbound")`
		`Endpoint endpoint;`

端点注入（按名称为Camel 上下文）	`@EndpointInject(uri = "direct:inbound",`
	`context = "foo")`
	`Endpoint contextEndpoint;`
		`@Inject`
		`@ContextName("foo")`
		`@Uri("direct:inbound")`
		`Endpoint contextEndpoint;`

Bean 注入（按类型）	`@BeanInject` `MyBean bean;`	`@Inject` `MyBean bean;`

Bean 注入（按名称）	`@BeanInject("foo")`
	`MyBean bean;`
		`@Inject`
		`@Named("foo")`
		`MyBean bean;`

POJO 消耗	`@Consume(uri = "seda:inbound")`
	`void consume(@Body String body) {`
	`//...`
	`}`	

60.8.2. Bean 组件

您可以使用 Java Camel DSL 根据类型或名称来引用 CDI Bean，例如使用 Java Camel DSL：

```
class MyBean {
    //...
}

from("direct:inbound").bean(MyBean.class);
```

或根据名称从 Java DSL 查找 CDI Bean：

```
@Named("foo")
class MyNamedBean {
    //...
```

```
}

from("direct:inbound").bean("foo");
```

60.8.3. 从 Endpoint URI 引用 Bean

当使用 URI 语法配置端点时，您可以使用 # 表示法引用 Registry 中的 Bean。如果 URI 参数值以 # 符号开头，则 Camel CDI 将按名称查找给定类型的 bean，例如：

```
from("jms:queue:{{destination}}?
transacted=true&transactionManager=#jtaTransactionManager").to("...");
```

将以下 CDI Bean 符合 @Named ("jtaTransactionManager")：

```
@Produces
@Named("jtaTransactionManager")
PlatformTransactionManager createTransactionManager(TransactionManager
transactionManager, UserTransaction userTransaction) {
    JtaTransactionManager jtaTransactionManager = new JtaTransactionManager();
    jtaTransactionManager.setUserTransaction(userTransaction);
    jtaTransactionManager.setTransactionManager(transactionManager);
    jtaTransactionManager.afterPropertiesSet();
    return jtaTransactionManager;
}
```

60.9. CAMEL 事件到 CDI 事件

从 Camel 2.17 开始提供

Camel 提供了一组 管理事件，可以订阅以侦听 Camel 上下文、服务、路由和交换事件。Camel CDI 无缝将这些 Camel 事件转换为 CDI 事件，这些事件可以使用 CDI 观察器方法，例如：

```
void onContextStarting(@Observes CamelContextStartingEvent event) {
    // Called before the default Camel context is about to start
}
```

从 Camel 2.18 开始，可以观察特定路由的事件 (RouteAddedEvent、RouteStartedEvent、RouteStoppedEvent 和 RouteRemovedEvent)应该有一个明确定义的，例如：

```
from("...").routeId("foo").to("...");

void onRouteStarted(@Observes @Named("foo") RouteStartedEvent event) {
```

```
    // Called after the route "foo" has started
}
```

当 CDI 容器中存在多个 Camel 上下文时，Camel 上下文 Bean qualifiers （如 @ContextName ）可用于将观察器方法解析优化为观察器方法解析，如 观察器解析，例如：

```
void onRouteStarted(@Observes @ContextName("foo") RouteStartedEvent event) {
    // Called after the route 'event.getRoute()' for the Camel context 'foo' has started
}

void onContextStarted(@Observes @Manual CamelContextStartedEvent event) {
    // Called after the Camel context qualified with '@Manual' has started
}
```

同样，如果存在多个上下文，则可以使用 @Default qualifier 观察 默认 Camel 上下文的 Camel 事件，例如：

```
void onExchangeCompleted(@Observes @Default ExchangeCompletedEvent event) {
    // Called after the exchange 'event.getExchange()' processing has completed
}
```

在该示例中，如果没有指定限定符，则隐式假定 @Any qualifier，以便接收所有 Camel 上下文的对应事件。

请注意，只有在部署中检测到侦听 Camel 事件的方法以及每个 Camel 上下文时，才会激活对 Camel 事件转换为 CDI 事件的支持。

60.10. CDI 事件端点

从 Camel 2.17 开始提供

CDI 事件端点通过 Camel 路由将 CDI 事件 桥接到 CDI 事件，以便无缝观察 CDI 事件（由 Camel 使用者生成/触发）（由 Camel producers）。

Camel CDI 提供的 CdiEventEndpoint<T > bean 可用于观察/消耗 事件类型 为 T 的 CDI 事件，例如：

```
@Inject
CdiEventEndpoint<String> cdiEventEndpoint;
```

```
from(cdiEventEndpoint).log("CDI event received: ${body}");
```

这等同于编写：

```
@Inject
@Uri("direct:event")
ProducerTemplate producer;

void observeCdiEvents(@Observes String event) {
    producer.sendBody(event);
}

from("direct:event").log("CDI event received: ${body}");
```

相反，CdiEventEndpoint<T > bean 可以用来生成 / fire CDI 事件，其 事件类型 为 T，例如：

```
@Inject
CdiEventEndpoint<String> cdiEventEndpoint;

from("direct:event").to(cdiEventEndpoint).log("CDI event sent: ${body}");
```

这等同于编写：

```
@Inject
Event<String> event;

from("direct:event").process(new Processor() {
    @Override
    public void process(Exchange exchange) {
        event.fire(exchange.getBody(String.class));
    }
}).log("CDI event sent: ${body}");
```

或使用 Java 8 lambda 表达式：

```
@Inject
Event<String> event;

from("direct:event")
    .process(exchange -> event.fire(exchange.getIn().getBody(String.class)))
    .log("CDI event sent: ${body}");
```

特定 CdiEventEndpoint< T > 注入点的 type 变量 T （resp.

```
@Inject
@FooQualifier
CdiEventEndpoint<List<String>> cdiEventEndpoint;

from("direct:event").to(cdiEventEndpoint);

void observeCdiEvents(@Observes @FooQualifier List<String> event) {
    logger.info("CDI event: {}", event);
}
```

当 CDI 容器中存在多个 Camel 上下文时，Camel 上下文 bean qualifiers （如 @ContextName ）可用于验证 CdiEventEndpoint<T& gt; 注入点，例如：

```
@Inject
@ContextName("foo")
CdiEventEndpoint<List<String>> cdiEventEndpoint;
// Only observes / consumes events having the @ContextName("foo") qualifier
from(cdiEventEndpoint).log("Camel context (foo) > CDI event received: ${body}");
// Produces / fires events with the @ContextName("foo") qualifier
from("...").to(cdiEventEndpoint);

void observeCdiEvents(@Observes @ContextName("foo") List<String> event) {
    logger.info("Camel context (foo) > CDI event: {}", event);
}
```

请注意，CDI 事件 Camel 端点会动态地为 事件类型 和事件 限定符 组合添加 观察器方法 ，并且仅依赖于容器类型safe 观察器解析，这会导致实施效率更高。

此外，作为 CDI 的 typesafe nature 和 Camel 组件 模型 的动态性质 之间的障碍非常高，因此无法通过 URI 创建 CDI 事件 Camel 端点的实例。实际上，CDI 事件组件的 URI 格式是：

```
cdi-event://PayloadType<T1,...,Tn>[?qualifiers=QualifierType1[,...[,QualifierTypeN]...]]
```

由于 authority PayloadType (resp. QualifierType)是 URI 转义的完全限定名称(resp. qualifier) raw 类型，后跟由 payload 参数化类型的类型参数部分分隔的 type 参数部分。这会导致 不友好的 URI，例如：

```
cdi-event://org.apache.camel.cdi.example.EventPayload%3Cjava.lang.Integer%3E?
qualifiers=org.apache.camel.cdi.example.FooQualifier%2Corg.apache.camel.cdi.example.Bar
Qualifier
```

但是，这可以防止在端点实例和观察器方法之间有效绑定，因为 CDI 容器在部署阶段没有发现 Camel 上下文模型的任何方法。

60.11. CAMEL XML 配置导入

从 Camel 2.18 开始提供

虽然 CDI 优先使用类型安全依赖关系注入机制，但将现有 Camel XML 配置文件重复利用到 Camel CDI 应用程序可能很有用。 在其他用例中，可能很方便地依赖 Camel XML DSL 来配置其 Camel 上下文。

您可以使用任何 CDI Bean 和 Camel CDI 上 Camel CDI 提供的 @ImportResource 注解将自动在指定位置载入 Camel XML 配置，例如：

```
@ImportResource("camel-context.xml")
class MyBean {
}
```

Camel CDI 将在类路径中指定的位置加载资源（以后可能会添加其他协议）。

来自导入的资源的每个 CamelContext 元素和其他 Camel 原语 会在容器 bootstrap 中自动部署为 CDI Bean，以便它们可从 Camel CDI 提供的自动配置中受益，并在运行时被注入。如果这个元素设置了显式 id 属性，则对应的 CDI bean 与 @Named qualifier （如以下 Camel XML 配置）授权：

```
<camelContext id="foo">
    <endpoint id="bar" uri="seda:inbound">
        <property key="queue" value="#queue"/>
        <property key="concurrentConsumers" value="10"/>
    </endpoint>
<camelContext/>
```

对应的 CDI Bean 被自动部署，并可注入，例如：

```
@Inject
@ContextName("foo")
CamelContext context;

@Inject
@Named("bar")
Endpoint endpoint;
```

请注意，CamelContext Bean 会自动与 @Named 和 @ContextName 限定符一起自动授权。 如果导入的 CamelContext 元素没有 id 属性，则对应的 bean 将使用内置 @Default qualifier 部署。

相反，应用程序中部署的 CDI Bean 可以从 Camel XML 配置引用，通常使用 ref 属性，例如，在声明了以下 bean 时：

```
@Produces
@Named("baz")
Processor processor = exchange -> exchange.getIn().setHeader("qux", "quux");
```

在导入的 Camel XML 配置中可以声明对该 bean 的引用，例如：

```
<camelContext id="foo">
    <route>
        <from uri="..."/>
        <process ref="baz"/>
    </route>
<camelContext/>
```

60.12. 事务支持

从 Camel 2.19 开始提供

Camel CDI 提供对使用 JTA 的 Camel 事务客户端的支持。

这个支持是可选的，因此您需要在应用程序类路径（如使用 Maven 时明确添加 JTA 作为依赖项）将 JTA 作为依赖项：

```
<dependency>
    <groupId>javax.transaction</groupId>
    <artifactId>javax.transaction-api</artifactId>
    <scope>runtime</scope>
</dependency>
```

您必须将您的应用程序部署到 JTA 功能容器中，或者提供独立的 JTA 实施。

小心

请注意，对于时间，事务管理器使用 java:/TransactionManager 键查找为 JNDI 资源。

将在以后添加更灵活的策略，以支持更广泛的部署方案。

60.12.1. 事务策略

Camel CDI 为通常支持的 Camel TransactedPolicy 作为 CDI Bean 提供实施。这些策略可以使用转换的 EIP （例如：

```
class MyRouteBean extends RouteBuilder {

    @Override
    public void configure() {
        from("activemq:queue:foo")
            .transacted("PROPAGATION_REQUIRED")
            .bean("transformer")
            .to("jpa:my.application.entity.Bar")
            .log("${body.id} inserted");
    }
}
```

这等同于：

```
class MyRouteBean extends RouteBuilder {

    @Inject
    @Named("PROPAGATION_REQUIRED")
    Policy required;

    @Override
    public void configure() {
        from("activemq:queue:foo")
            .policy(required)
            .bean("transformer")
            .to("jpa:my.application.entity.Bar")
            .log("${body.id} inserted");
    }
}
```

支持的事务策略名称是：

- PROPAGATION_NEVER,
- PROPAGATION_NOT_SUPPORTED,
- PROPAGATION_SUPPORTS,

- PROPAGATION_REQUIRED,
- PROPAGATION_REQUIRES_NEW,
- PROPAGATION_NESTED,
- PROPAGATION_MANDATORY.

60.12.2. 事务错误处理程序

Camel CDI 提供了扩展重新传送错误处理程序的事务错误处理程序，每当发生异常时强制进行回滚，并为每个重新发送创建新事务。

Camel CDI 提供 CdiRouteBuilder 类，它公开 transactionErrorHandler 帮助程序方法，以启用对配置的快速访问，例如：

```
class MyRouteBean extends CdiRouteBuilder {

    @Override
    public void configure() {
        errorHandler(transactionErrorHandler()
            .setTransactionPolicy("PROPAGATION_SUPPORTS")
            .maximumRedeliveries(5)
            .maximumRedeliveryDelay(5000)
            .collisionAvoidancePercent(10)
            .backOffMultiplier(1.5));
    }
}
```

60.13. 自动配置的 OSGI 集成

从 Camel 2.17 开始提供

Camel 上下文 Bean 由 Camel CDI 自动调整，以便它们作为 OSGi 服务注册，以及各种解析器（如 ComponentResolver 和 DataFormatResolver）与 OSGi 注册表集成。这意味着 Karaf Camel 命令可用于运行由 Camel CDI 自动配置的 Camel 上下文，例如：

```
karaf@root()> camel:context-list
```

```
Context       Status              Total #       Failed #     Inflight #   Uptime
-------       ------              -------       --------     ----------   ------
camel-cdi     Started                 1             0              0   1 minute
```

有关 Camel CDI OSGi 集成的工作示例，请参阅 camel-example-cdi-osgi 示例。

60.14. LAZY INJECTION / PROGRAMMATIC LOOKUP

虽然 CDI 编程模型优先选择应用程序初始化时发生的 类型安全解析 机制，但稍后可以使用 编程查找 机制在应用程序执行期间执行动态/延迟注入。

Camel CDI 提供方便注解，与 CDI 限定符对应，可用于标准 Camel 原语注入。这些注解文字可以和 javax.enterprise.inject.Instance 接口结合使用，即 CDI 入口点执行 lazy 注入 / programmatic lookup。

例如，您可以将 @Uri qualifier 的提供的注解文字用于Lazily lookup for Camel primitives，如 ProducerTemplate Bean：

```
@Any
@Inject
Instance<ProducerTemplate> producers;

ProducerTemplate inbound = producers
    .select(Uri.Literal.of("direct:inbound"))
    .get();
```

或对于 Endpoint Bean，例如：

```
@Any
@Inject
Instance<Endpoint> endpoints;

MockEndpoint outbound = endpoints
    .select(MockEndpoint.class, Uri.Literal.of("mock:outbound"))
    .get();
```

同样，您可以使用 @ContextName qualifier 提供的注解文字为 CamelContext Bean 的 lazily 查找，例如：

```
@Any
@Inject
Instance<CamelContext> contexts;
```

```
CamelContext context = contexts
    .select(ContextName.Literal.of("foo"))
    .get();
```

您还可以根据 Camel 上下文类型重新定义选择，例如：

```
@Any
@Inject
Instance<CamelContext> contexts;

// Refine the selection by type
Instance<DefaultCamelContext> context = contexts.select(DefaultCamelContext.class);

// Check if such a bean exists then retrieve a reference
if (!context.isUnsatisfied())
    context.get();
```

甚至迭代 Camel 上下文，例如：

```
@Any
@Inject
Instance<CamelContext> contexts;

for (CamelContext context : contexts)
    context.setUseBreadcrumb(true);
```

60.15. MAVEN ARCHETYPE

在可用的 Camel Maven archetypes 中，您可以使用提供的 camel-archetype-cdi 来生成 Camel CDI Maven 项目，例如：

```
mvn archetype:generate -DarchetypeGroupId=org.apache.camel.archetypes -
DarchetypeArtifactId=camel-archetype-cdi
```

60.16. 支持的容器

Camel CDI 组件与任何 CDI 1.0、CDI 1.1 和 CDI 1.2 兼容运行时兼容。它根据以下运行时成功测试：

Container	Version	Runtime
Weld SE	**1.1.28.final**	CDI 1.0 / Java SE 7

Container	Version	Runtime
OpenWebBeans	**1.2.7**	CDI 1.0 / Java SE 7
Weld SE	**2.4.2.final**	CDI 1.2 / Java SE 7
OpenWebBeans	**1.7.2**	CDI 1.2 / Java SE 7
WildFly	**8.2.1.final**	CDI 1.2 / Java EE 7
WildFly	**9.0.1.final**	CDI 1.2 / Java EE 7
WildFly	**10.1.0.Final**	CDI 1.2 / Java EE 7

60.17. 例子

以下示例包括在 Camel 项目 的示例 目录中：

Example	描述
camel-example-cdi	演示了如何使用 CDI 使用 Camel 配置组件、端点和 Bean
camel-example-cdi-kubernetes	演示了 Camel、CDI 和 Kubernetes 之间的集成
camel-example-cdi-metrics	演示 Camel、Dropwizard Metrics 和 CDI 之间的集成
camel-example-cdi-properties	演示 Camel, DeltaSpike 和 CDI 之间的集成，用于配置属性
camel-example-cdi-osgi	使用 SJMS 组件的 CDI 应用程序，可以使用 PAX CDI 在 OSGi 容器内执行
camel-example-cdi-rest-servlet	演示了在 Web 应用程序中使用 CDI 作为依赖项注入框架的 Camel REST DSL
camel-example-cdi-test	演示作为 Camel 和 CDI 间集成的一部分提供的测试功能
camel-example-cdi-xml	演示在 Camel CDI 应用程序中使用 Camel XML 配置文件
camel-example-openapi-cdi	在 CDI 中使用 REST DSL 和 OpenAPI Java 的示例

Example	描述
camel-example-swagger-cdi	在 CDI 中使用 REST DSL 和 Swagger Java 的示例
camel-example-widget-gadget-cdi	使用 CDI 依赖项注入在 Java 中实施的 EIP 书中的 Widget 和 Gadget 用例

60.18. 另请参阅

- ***Camel CDI 测试***
- ***CDI 规格网站***
- ***CDI 生态系统***
- ***Weld 主页***
- ***OpenWebBeans 主页***
- ***进一步使用 CDI 和 Camel （请参阅 Camel CDI 部分）***

60.19. CAMEL CDI 用于 WILDFLY-CAMEL 上的 EAR 部署

与标准的 WAR 或 JAR 部署相比，WildFly-Camel 上的 Camel CDI EAR 部署在类和资源加载方面有一些区别。

WildFly bootstraps Weld 使用 EAR 部署类Loader。WildFly 还要求所有 EAR 子部署仅创建并共享一个 CDI 扩展。

这会导致"Auto-configured" CDI Camel 上下文使用 EAR 部署类加载器来动态加载类和资源。默认情况下，这个 ClassLoader 无法访问 EAR 子部署中的资源。

对于 EAR 部署，建议使用 'Auto-configured' CDI Camel Context，并且 RouteBuilder 类使用

@ContextName 标注，或者通过 @ImportResource 注释或通过 CDI producer 方法和字段创建 CamelContext。这有助于 WildFly-Camel 确定用于 Camel 的正确 ClassLoader。

第 61 章 CHRONICLE ENGINE 组件

从 Camel 版本 2.18 开始提供

camel chronicle-engine 组件可让您利用 OpenHFT 的 Chronicle-Engine 的强大功能

61.1. URI 格式

```
chronicle-engine:addresses/path[?options]
```

61.2. URI 选项

Chronicle Engine 组件没有选项。

Chronicle Engine 端点使用 URI 语法进行配置：

```
chronicle-engine:addresses/path
```

使用以下路径和查询参数：

61.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
addresses	**所需的** 引擎地址。可以使用逗号分隔多个地址。		字符串
path	**所需的** 引擎路径		字符串

61.2.2. 查询参数(12 参数)：

Name	描述	默认值	类型
action (common)	执行的默认操作和有效值是： - PUBLISH - PPUBLISH_AND_INDEX - PPUT - PGET_AND_PUT - PPUT_ALL - PPUT_IF_ABSENT - PGET - PGET_AND_REMOVE - PREMOVE - PIS_EMPTY - PSIZE		字符串

Name	描述	默认值	类型
clusterName (common)	队列的集群名称		字符串
filteredMapEvents (common)	以逗号分隔的 map 事件类型列表到 filer，有效值有：INSERT、UPDATE、REMOVE。		字符串
persistent (common)	启用/禁用数据持久性	true	布尔值
subscribeMapEvents (common)	设置消费者是否应该订阅映射事件，默认为 true。	true	布尔值
subscribeTopicEvents (common)	设置消费者是否应该订阅 TopicEvents，d efault false。	false	布尔值
subscribeTopologicalEvents (common)	设置消费者是否应该订阅 TopologicalEvents，d efault false。	false	布尔值
wireType (common)	要使用的 Wire 类型，默认为二进制连接。	二进制	字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

61.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.chronicle-engine.enabled	启用 chronicle-engine 组件	true	布尔值
camel.component.chronicle-engine.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 62 章 块组件

从 Camel 版本 2.15 开始提供

chunk: 组件允许使用 Chunk 模板处理消息。这在使用 Templating 生成请求的响应时是理想的选择。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
<groupId>org.apache.camel</groupId>
<artifactId>camel-chunk</artifactId>
<version>x.x.x</version> <!-- use the same version as your Camel core version -->
</dependency>
```

62.1. URI 格式

```
chunk:templateName[?options]
```

其中 templateName 是要调用的模板的 classpath-local URI。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

62.2. 选项

Chunk 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
allowContextMapAll (producer)	设置上下文映射是否应该允许访问所有详情。默认情况下，只能访问消息正文和标头。可以启用此选项以完全访问当前 Exchange 和 CamelContext。这样做会带来潜在的安全风险，因为这会打开对 CamelContext API 的完整功能的访问。	false	布尔值
allowTemplateFromHeader (producer)	是否允许使用来自标头的资源模板（默认 false）。启用此选项具有安全原因。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。	false	布尔值

Chunk 端点使用 URI 语法进行配置：

```
chunk:resourceUri
```

使用以下路径和查询参数：

62.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceUri	资源 **所需的** 路径。您可以使用这些协议(classpath 为 default)使用： classpath、file、http、ref 或 bean. classpath、file 和 http 加载资源的前缀。ref 将查找 registry 中的资源。bean 将调用 bean 以供用作资源的方法。对于 bean，您可以在点后指定方法名称，如 bean:myBean.myMethod。		字符串

62.2.2. 查询参数(9 参数)：

Name	描述	默认值	类型
allowContextMapAll (producer)	设置上下文映射是否应该允许访问所有详情。默认情况下，只能访问消息正文和标头。可以启用此选项以完全访问当前 Exchange 和 CamelContext。这样做会带来潜在的安全风险，因为这会打开对 CamelContext API 的完整功能的访问。	false	布尔值
allowTemplateFromHeader (producer)	是否允许使用来自标头的资源模板（默认 false）。启用此选项具有安全原因。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。	false	布尔值
contentCache (producer)	设置是否使用资源内容缓存	false	布尔值
encoding (producer)	定义正文的编码		字符串
extension (producer)	定义模板的文件扩展		字符串
themeFolder (producer)	定义要扫描的文件夹		字符串

Name	描述	默认值	类型
themeLayer (producer)	将主题层定义为详细		字符串
themeSubfolder (producer)	定义要扫描的 themes 子文件夹		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

62.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.chunk.enabled	启用块组件	true	布尔值
camel.component.chunk.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

块组件将在 themes 文件夹中查找带有扩展 .chtml 或 _.cxml. _ _ 的特定模板。如果您需要指定不同的文件夹或扩展，您需要使用上面列出的特定选项。

62.4. 块上下文

Camel 将在 Chunk 上下文(just a Map)中提供交换信息。Exchange 作为以下内容传输：

key	value
交换	**Exchange** 本身。
exchange.properties	**Exchange** 属性.
标头	In 消息的标头。

key	value
camelContext	Camel 上下文.
Request （请求）	In 消息。
正文（body）	In 消息正文。
响应	Out 消息（仅适用于 InOut 消息交换模式）。

62.5. 动态模板

Camel 提供了两个标头，您可以为模板或模板内容本身定义不同的资源位置。如果设置了其中任何标头，则 Camel 会通过端点配置的资源使用此标头。这可让您在运行时提供动态模板。

标头	类型	描述	支持版本
ChunkConstants.CHUNK_RESOURCE_URI	字符串	要使用的模板资源的 URI，而不是配置的端点。	
ChunkConstants.CHUNK_TEMPLATE	字符串	要使用的模板而不是配置的端点。	

62.6. SAMPLES

例如，您可以使用类似如下的内容：

```
from("activemq:My.Queue").
to("chunk:template");
```

要使用 Chunk 模板为 InOut 消息交换发送发送响应（其中有一个 JMSReplyTo 标头）。

如果要使用 InOnly 并使用消息并将其发送到您可以使用的另一个目的地：

```
from("activemq:My.Queue").
to("chunk:template").
to("activemq:Another.Queue");
```

可以指定组件应该通过标头动态使用的模板，例如：

```
from("direct:in").
setHeader(ChunkConstants.CHUNK_RESOURCE_URI).constant("template").
to("chunk:dummy?allowTemplateFromHeader=true");
```

警告

启用 allowTemplateFromHeader 选项具有安全等级。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。

Chunk 组件选项使用的示例：

```
from("direct:in").
to("chunk:file_example?
themeFolder=template&themeSubfolder=subfolder&extension=chunk");
```

在本例中，Chunk 组件将在文件夹 template/subfolder 中查找 file_example.chunk。

62.7. 电子邮件示例

在本例中，我们希望使用 Chunk 模板进行订单确认电子邮件。电子邮件模板在 Chunk 中出现：

```
Dear {$headers.lastName}, {$headers.firstName}

Thanks for the order of {$headers.item}.

Regards Camel Riders Bookstore
{$body}
```

62.8. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 63 章 类组件

从 Camel 版本 2.4 开始提供

class: 组件将 Bean 绑定到 Camel 消息交换。它的工作方式与 Bean 组件相同，而不是从 Registry 查找 Bean，而是根据类名称创建 bean。

63.1. URI 格式

```
class:className[?options]
```

其中 className 是要创建和使用 Bean 的完全限定类名称。

63.2. 选项

Class 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
缓存 （高级）	如果启用，Camel 将缓存第一个 Registry 查找的结果。如果 Registry 中的 bean 定义为单例范围，则可以启用缓存。		布尔值
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Class 端点使用 URI 语法进行配置：

```
class:beanName
```

使用以下路径和查询参数：

63.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
beanName	**必需** 设置要调用的 bean 的名称		字符串

63.2.2. 查询参数(5 参数)：

Name	描述	默认值	类型
method (producer)	设置要在 bean 上调用的方法名称		字符串
缓存 （高级）	如果启用，Camel 将缓存第一个 Registry 查找的结果。如果 Registry 中的 bean 定义为单例范围，则可以启用缓存。		布尔值
multiParameterArray (advanced)	**弃用** How to treat 处理从消息正文传递的参数；如果为 true，消息正文应该是参数数组。注意：此选项由 Camel 内部使用，不面向最终用户使用。弃用备注：此选项由 Camel 内部使用，不适用于最终用户使用。	false	布尔值
参数 (advanced)	用于在 bean 中配置附加属性		Map
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

63.3. 使用

您只是像 Bean 组件一样使用 类 组件，而是通过指定完全限定类名称来替代。例如，要使用 MyFooBean，您必须执行以下操作：

```
from("direct:start").to("class:org.apache.camel.component.bean.MyFooBean").to("mock:result");
```

您也可以指定要在 MyFooBean 上调用的方法，例如 hello ：

```
from("direct:start").to("class:org.apache.camel.component.bean.MyFooBean?method=hello").to("mock:result");
```

63.4. 设置创建实例的属性

在 endpoint uri 中，您可以指定要在创建实例上设置的属性，例如，具有 setPrefix 方法：

```
// Camel 2.17 onwards
from("direct:start")
    .to("class:org.apache.camel.component.bean.MyPrefixBean?bean.prefix=Bye")
    .to("mock:result");

// Camel 2.16 and older
from("direct:start")
    .to("class:org.apache.camel.component.bean.MyPrefixBean?prefix=Bye")
    .to("mock:result");
```

此外，您也可以使用 # 语法来指代 Registry 中查找的属性。

```
// Camel 2.17 onwards
from("direct:start")
    .to("class:org.apache.camel.component.bean.MyPrefixBean?bean.cool=#foo")
    .to("mock:result");

// Camel 2.16 and older
from("direct:start")
    .to("class:org.apache.camel.component.bean.MyPrefixBean?cool=#foo")
    .to("mock:result");
```

该命令将从带有 id foo 的 Registry 查找 bean，并在 MyPrefixBean 类创建的实例上调用 setCool 方法。

TIP：请参阅 Bean 组件的更多详细信息，因为 类 组件的工作方式与类组件相同。

63.5. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- Bean

- ***Bean Binding***

- ***Bean 集成***

第 64 章 CMIS 组件

从 Camel 版本 2.11 开始提供

cmis 组件使用 Apache Chemistry 客户端 API，并允许您向 CMIS 兼容内容存储库添加/读取节点。

64.1. URI 格式

```
cmis://cmisServerUrl[?options]
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

64.2. CMIS 选项

CMIS 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
sessionFacadeFactory (common)	使用自定义 CMISSessionFacadeFactory 创建 CMISSessionFacade 实例		CMISSessionFacade Factory
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

CMIS 端点使用 URI 语法进行配置：

```
cmis:cmsUrl
```

使用以下路径和查询参数：

64.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cmsUrl	cmis 存储库 **所需的** URL		字符串

64.2.2. 查询参数(13 参数)：

Name	描述	默认值	类型
pageSize (common)	每个页面检索的节点数量	100	int
readContent (common)	如果设置为 true，则除属性外还会检索文档节点的内容	false	布尔值
readCount (common)	要读取的最大节点数		int
repositoryId (common)	要使用的存储库的 Id。如果没有指定第一个可用存储库		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
query (consumer)	要针对存储库执行的 cmis 查询。如果没有指定，消费者将通过递归迭代内容树来从内容存储库检索每个节点		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
queryMode (producer)	如果为 true，则会从消息正文执行 cmis 查询并返回结果，否则将在 cmis 仓库中创建节点	false	布尔值
sessionFacadeFactory (advanced)	使用自定义 CMISSessionFacadeFactory 创建 CMISSessionFacade 实例		CMISSessionFacade Factory
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
密码 (security)	cmis 存储库的密码		字符串
用户名 (security)	cmis 软件仓库的用户名		字符串

64.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.cmis.enabled	启用 cmis 组件	true	布尔值
camel.component.cmis.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.cmis.session-facade-factory	使用自定义 CMISSessionFacadeFactory 来创建 CMISSessionFacade 实例。选项是 org.apache.camel.component.cmis.CMISSessionFacadeFactory 类型。		字符串

64.4. 使用方法

64.4.1. 由制作者评估的消息标头

标头	默认值	描述
CamelCMISFolderPath	/	执行期间使用的当前文件夹。如果未指定，将使用根文件夹
CamelCMISRetrieveContent	**false**	在 **queryMode** 中，此标头将强制生成者检索文档节点的内容。
CamelCMISReadSize	**0**	要读取的最大节点数。
cmis:path	**null**	如果没有设置 **CamelCMISFolderPath**，它将尝试从这个 cmis 属性中查找节点的路径，它是 name
cmis:name	**null**	如果没有设置 **CamelCMISFolderPath**，它将尝试从这个 cmis 属性中查找节点的路径，它是路径

标头	默认值	描述
cmis:objectTypeId	**null**	节点的类型
cmis:contentStreamMimeType	**null**	为文档设置的 mimetype

64.4.2. 在查询 Producer 操作过程中设置的消息标头

标头	类型	描述
CamelCMISResultCount	**整数**	从查询返回的节点数。

消息正文将包含一个映射列表，映射中的每个条目都是 cmis 属性及其值。如果 CamelCMISRetrieveContent 标头设为 true，则带有键 CamelCMISContent 的映射中的一个额外条目将包含节点类型的 InputStream。

64.5. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-cmis</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 ${camel-version} 必须替换为 Camel 的实际版本(2.11 或更高版本)。

64.6. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 65 章 CM SMS GATEWAY 组件

从 Camel 版本 2.18 开始提供

camel-Cm-Sms 是 [CM SMS Gateway](https://www.cmtelecom.com)的 Apache Camel 组件。

它允许将应用程序中的 CM SMS API集成为 camel 组件。

您必须有一个有效的帐户。如需更多信息，请参阅 CM Telecom。

```
cm-sms://sgw01.cm.nl/gateway.ashx?
defaultFrom=DefaultSender&defaultMaxNumberOfParts=8&productToken=xxxxx
```

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
---
<dependency>
 <groupId>org.apache.camel</groupId>
 <artifactId>camel-cm-sms</artifactId>
 <version>x.x.x</version>
 <!-- use the same version as your Camel core version -->
</dependency>
---
```

65.1. 选项

CM SMS Gateway 组件没有选项。

CM SMS 网关端点使用 URI 语法进行配置：

```
cm-sms:host
```

使用以下路径和查询参数：

65.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
主机	使用方案 **需要** SMS Provider HOST		字符串

65.1.2. 查询参数(5 参数)：

Name	描述	默认值	类型
defaultFrom (producer)	这是发送者名称。最大长度为 11 个字符。		字符串)
defaultMaxNumberOfParts (producer)	如果它是多部分消息，会强制使用 max 数。可以截断消息。从技术上讲，网关会首先检查消息是否大于 160 个字符，如果是，则消息将划分为由这些参数限制的多个 153 个字符部分。	8	Max(8L)::Int)
productToken (producer)	**必需** 使用的唯一令牌		字符串)
testConnectionOnStartup (producer)	是否在启动时测试到 SMS 网关的连接	false	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

65.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.cm-sms.enabled	启用 cm-sms 组件	true	布尔值
camel.component.cm-sms.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

65.3. 示例

您可以尝试 此项目，查看如何在 camel 路由中集成 camel-cm-sms。

第 66 章 COAP 组件

从 Camel 版本 2.16 开始提供

Camel-CoAP 是一个 Apache Camel 组件，允许您使用 CoAP，它是一个轻量级的 REST 类型协议用于机器操作。CoAP, Constraed Application Protocol 是一个特殊的 Web 传输协议，用于限制的节点和受限网络，它基于 RFC 7252。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
 <groupId>org.apache.camel</groupId>
 <artifactId>camel-coap</artifactId>
 <version>x.x.x</version>
 <!-- use the same version as your Camel core version -->
</dependency>
```

66.1. 选项

CoAP 组件没有选项。

CoAP 端点使用 URI 语法进行配置：

```
coap:uri
```

使用以下路径和查询参数：

66.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
uri	CoAP 端点的 URI		URI

66.1.2. 查询参数(5 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
coapMethodRestrict (consumer)	CoAP 消费者将绑定到的以逗号分隔的方法列表。默认值为绑定到所有方法(DELETE、GET、POST、PUT)。		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

66.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.coap.enabled	启用 coap 组件	true	布尔值
camel.component.coap.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

66.3. 消息标头

Name	类型	描述
CamelCoapMethod	**字符串**	在调用目标 CoAP 服务器 URI 时，CoAP 生成者应使用的请求方法。有效选项包括 DELETE、GET、PING、POST 和 PUT。
CamelCoapResponseCode	**字符串**	外部服务器发送的 CoAP 响应代码。如需了解每个代码的含义的详细信息，请参阅 RFC 7252。
CamelCoapUri	**字符串**	要调用的 CoAP 服务器的 URI。将覆盖直接在端点上配置的任何现有 URI。

66.3.1. 配置 CoAP producer 请求方法

以下规则决定了 CoAP producer 将用来调用目标 URI 的请求方法：

1. *CamelCoapMethod 标头的值*
2. *GET 如果目标 CoAP 服务器 URI 上提供了查询字符串。*
3. *POST 如果消息交换正文不是 null。*
4. *GET 否则。*

第 67 章 恒定语言

从 Camel 版本 1.5 开始提供

Constant 表达式语言实际上只是将常量字符串指定为一种表达式类型的方法。

注意

这是一个固定的恒定值，在启动路由的过程中只设置一次，如果您希望在路由过程中使用动态值，请不要使用它。

67.1. 常数选项

Constant 语言支持 1 个选项，如下所列。

Name	默认值	Java 类型	描述
trim	**true**	布尔值	是否修剪值以移除前导和结尾的空格和换行符

67.2. 用法示例

Spring DSL 的 setHeader 元素可以使用一个恒定表达式，如下所示：

```
<route>
  <from uri="seda:a"/>
  <setHeader headerName="theHeader">
    <constant>the value</constant>
  </setHeader>
  <to uri="mock:b"/>
</route>
```

在这种情况下，来自 seda:a Endpoint 的 Message 会将 'theHeader' 标头设置为常量值 'the value'。

和使用 Java DSL 的同一示例：

```
from("seda:a")
  .setHeader("theHeader", constant("the value"))
  .to("mock:b");
```

67.3. 依赖项

Constant 语言是 camel-core 的一部分。

第 68 章 COMETD 组件

从 Camel 版本 2.0 开始提供

cometd: 组件是处理相较于 /bayeux 协议的 jetty 实施的传输。
将这个组件与 dojo 工具包库结合使用，可以使用基于 AJAX 机制将 Camel 消息直接推送到浏览器中。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-cometd</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

68.1. URI 格式

```
cometd://host:port/channelName[?options]
```

channelName 代表可由 Camel 端点订阅的主题。

68.2. 例子

```
cometd://localhost:8080/service/mychannel
cometds://localhost:8443/service/mychannel
```

where cometds: 代表 SSL 配置的端点。

68.3. 选项

CometD 组件支持 8 个选项，如下所列。

Name	描述	默认值	类型
sslKeyPassword (security)	使用 SSL 时密钥存储的密码。		字符串

Name	描述	默认值	类型
sslpassword (security)	使用 SSL 时的密码。		字符串
sslKeystore (security)	密钥存储的路径。		字符串
SecurityPolicy (security)	使用自定义配置的 SecurityPolicy 来控制授权		SecurityPolicy
extensions (common)	要使用自定义 BayeuxServer.Extension 列表，允许修改传入和传出请求。		list
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

CometD 端点使用 URI 语法进行配置：

```
cometd:host:port/channelName
```

使用以下路径和查询参数：

68.3.1. 路径参数(3 参数)：

Name	描述	默认值	类型
主机	**所需的** 主机名		字符串
port	**所需的主机** 端口号		int
channelName	**必需** The channelName 代表 Camel 端点可订阅的主题。		字符串

68.3.2. 查询参数(16 参数)：

Name	描述	默认值	类型
allowedOrigins (common)	如果 crosssOriginFilterOn 为 true，则支持跨的源域	*	字符串
baseResource (common)	Web 资源或 classpath 的根目录。使用协议 file: 或 classpath:，具体取决于组件是否希望从文件系统或 classpath 加载资源。OSGI 部署需要 classpath，其中资源被打包在 jar 中		字符串
crossOriginFilterOn (common)	如果为 true，服务器将支持跨域过滤	false	布尔值
filterPath (common)	如果 crosssOriginFilterOn 为 true，则 CrossOriginFilter 将使用 filterPath		字符串
interval (common)	客户端侧轮询超时（以毫秒为单位）。客户端在重新连接之间等待的时间		int
jsonCommented (common)	如果为 true，服务器将接受注释中嵌套的 JSON，并将生成 JSON 嵌套在注释中。这是对 AssumeRole Hijacking 的 defence。	true	布尔值
logLevel (common)	日志记录级别。0=None, 1=info, 2=debug.	1	int
maxInterval (common)	最大客户端侧轮询超时（以毫秒为单位）。如果此时没有收到连接，则将删除客户端。	30000	int
multiFrameInterval (common)	如果从同一浏览器检测到多个连接，客户端侧轮询超时。	1500	int
timeout (common)	服务器端轮询超时（以毫秒为单位）。这是服务器在响应前保存重新连接请求的时间。	240000	int
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
sessionHeadersEnabled (consumer)	在为传入请求创建 Camel 消息时，是否在 Camel 消息中包含服务器会话标头。	false	布尔值

Name	描述	默认值	类型
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
disconnectLocalSession (producer)	将消息发布到其频道后是否断开本地会话。需要断开本地会话，因为它们默认为 CometD，因此您可以耗尽内存。	false	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

68.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
camel.component.cometd.enabled	启用发出的组件	true	布尔值
camel.component.cometd.extensions	要使用自定义 BayeuxServer.Extension 列表，允许修改传入和传出请求。		list
camel.component.cometd.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.cometd.security-policy	使用自定义配置的 SecurityPolicy 来控制授权。选项是一个 org.cometd.bayeux.server.SecurityPolicy 类型。		字符串
camel.component.cometd.ssl-context-parameters	使用 SSLContextParameters 配置安全性。选项是 org.apache.camel.util.jsse.SSLContextParameters 类型。		字符串
camel.component.cometd.ssl-key-password	使用 SSL 时密钥存储的密码。		字符串

Name	描述	默认值	类型
camel.component.cometd.ssl-keystore	密钥存储的路径。		字符串
camel.component.cometd.ssl-password	使用 SSL 时的密码。		字符串
camel.component.cometd.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

以下是如何传递参数的一些示例

对于 file （用于 web 应用程序目录 -→ cometd://localhost:8080?resourceBase=file./webapp for classpath）中（例如，web 资源被打包在 webapp 文件夹中 -→ cometd://localhost:8080?resourceBase=classpath:webapp）

68.5. 身份验证

从 Camel 2.8 开始提供

您可以将自定义 SecurityPolicy 和 Extension's 配置为 'CometdComponent，它允许您使用身份验证，如下所示 http://cometd.org/documentation/howtos/authentication

68.6. 为 COMETD 组件设置 SSL

68.6.1. 使用 JSSE 配置实用程序

自 Camel 2.9 起，Cometd 组件通过 Camel JSSE 配置实用程序支持 SSL/TLS 配置。这个实用程序可大大减少您需要写入的组件特定代码量，并在端点和组件级别进行配置。以下示例演示了如何将 实用程序与 Cometd 组件一起使用。您需要在 CometdComponent 上配置 SSL。

组件的编程配置

```
KeyStoreParameters ksp = new KeyStoreParameters();
ksp.setResource("/users/home/server/keystore.jks");
ksp.setPassword("keystorePassword");

KeyManagersParameters kmp = new KeyManagersParameters();
kmp.setKeyStore(ksp);
kmp.setKeyPassword("keyPassword");

TrustManagersParameters tmp = new TrustManagersParameters();
tmp.setKeyStore(ksp);

SSLContextParameters scp = new SSLContextParameters();
scp.setKeyManagers(kmp);
scp.setTrustManagers(tmp);

CometdComponent commetdComponent = getContext().getComponent("cometds",
CometdComponent.class);
commetdComponent.setSslContextParameters(scp);
```

基于 Spring DSL 的端点配置

```
...
  <camel:sslContextParameters
      id="sslContextParameters">
    <camel:keyManagers
        keyPassword="keyPassword">
      <camel:keyStore
          resource="/users/home/server/keystore.jks"
          password="keystorePassword"/>
    </camel:keyManagers>
    <camel:trustManagers>
      <camel:keyStore
          resource="/users/home/server/keystore.jks"
          password="keystorePassword"/>
    </camel:keyManagers>
  </camel:sslContextParameters>...

  <bean id="cometd" class="org.apache.camel.component.cometd.CometdComponent">
    <property name="sslContextParameters" ref="sslContextParameters"/>
  </bean>
...
  <to uri="cometds://127.0.0.1:443/service/test?baseResource=file:./target/test-
classes/webapp&timeout=240000&interval=0&maxInterval=30000&multiFrameInterval=1500&jsonCom
mented=true&logLevel=2"/>...
```

68.7. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 69 章 CONSUL COMPONENT

从 Camel 版本 2.18 开始提供

Consul 组件是将应用程序与 Consul 集成的组件。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-consul</artifactId>
  <version>${camel-version}</version>
</dependency>
```

69.1. URI 格式

```
consul://domain?[options]
```

您可以使用以下格式将查询选项附加到 URI 中：

```
?option=value&option=value&...
```

69.2. 选项

Consul 组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
url (common)	Consul 代理 URL		字符串
DataCenter (common)	数据中心		字符串
sslContextParameters (common)	使用 org.apache.camel.util.jsse.SSLContextParameters 实例的 SSL 配置。		SSLContextParameters
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值

Name	描述	默认值	类型
aclToken (common)	设置要与 Consul 搭配使用的 ACL 令牌		字符串
username (common)	设置用于基本身份验证的用户名		字符串
password (common)	设置用于基本身份验证的密码		字符串
配置 （高级）	设置端点之间共享的通用配置		ConsulConfiguration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Consul 端点使用 URI 语法进行配置：

```
consul:apiEndpoint
```

使用以下路径和查询参数：

69.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
apiEndpoint	**所需的** API 端点		字符串

69.2.2. 查询参数(4 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值

Name	描述	默认值	类型
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

69.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 90 个选项，如下所列。

Name	描述	默认值	类型
camel.component .consul.acl-token	设置要与 Consul 搭配使用的 ACL 令牌		字符串
camel.component .consul.cluster.service.acl-token			字符串
camel.component .consul.cluster.service.attributes	自定义服务属性。		Map
camel.component .consul.cluster.service.block-seconds			整数
camel.component .consul.cluster.service.connect-timeout-millis			Long
camel.component .consul.cluster.service.consistency-mode			ConsistencyMode

Name	描述	默认值	类型
camel.component.consul.cluster.service.datacenter			字符串
camel.component.consul.cluster.service.enabled	设置 consul 集群服务是否应启用，则默认为 false。	false	布尔值
camel.component.consul.cluster.service.first-index			BigInteger
camel.component.consul.cluster.service.id	集群服务 ID		字符串
camel.component.consul.cluster.service.near-node			字符串
camel.component.consul.cluster.service.node-meta			list
camel.component.consul.cluster.service.order	服务查找顺序/优先级.		整数
camel.component.consul.cluster.service.password			字符串
camel.component.consul.cluster.service.ping-instance			布尔值
camel.component.consul.cluster.service.read-timeout-millis			Long
camel.component.consul.cluster.service.recursive			布尔值

Name	描述	默认值	类型
camel.component.consul.cluster.service.root-path			字符串
camel.component.consul.cluster.service.session-lock-delay			整数
camel.component.consul.cluster.service.session-refresh-interval			整数
camel.component.consul.cluster.service.session-ttl			整数
camel.component.consul.cluster.service.ssl-context-parameters			SSLContextParameters
camel.component.consul.cluster.service.tags			Set
camel.component.consul.cluster.service.url			字符串
camel.component.consul.cluster.service.user-name			字符串
camel.component.consul.cluster.service.write-timeout-millis			Long
camel.component.consul.configuration.acl-token	设置要与 Consul 搭配使用的 ACL 令牌		字符串
camel.component.consul.configuration.action	默认操作。CamelConsulAction 可以被 CamelConsulAction 覆盖		字符串

Name	描述	默认值	类型
camel.component.consul.configuration.block-seconds	第二个等待监视事件，默认为 10 秒		整数
camel.component.consul.configuration.connect-timeout-millis	为 OkHttpClient 连接超时		Long
camel.component.consul.configuration.consistency-mode	用于查询的 consistencyMode，默认的 ConsistencyMode.DEFAULT		ConsistencyMode
camel.component.consul.configuration.consul-client	对 registry 中的 **com.orbitz.consul.Consul** 的引用。		consul
camel.component.consul.configuration.datacenter	数据中心		字符串
camel.component.consul.configuration.first-index	监视的第一个索引，默认值为 0		BigInteger
camel.component.consul.configuration.key	默认密钥。CamelConsulKey 可以被 CamelConsulKey 覆盖		字符串
camel.component.consul.configuration.near-node	用于查询的接近节点。		字符串
camel.component.consul.configuration.node-meta	用于查询的 note meta-data。		list
camel.component.consul.configuration.password	设置用于基本身份验证的密码		字符串
camel.component.consul.configuration.ping-instance	配置 AgentClient 是否应该在返回 Consul 实例前尝试 ping		布尔值

Name	描述	默认值	类型
camel.component.consul.configuration.read-timeout-millis	OkHttpClient 读取超时		Long
camel.component.consul.configuration.recursive	递归监视，默认 false		布尔值
camel.component.consul.configuration.ssl-context-parameters	使用 org.apache.camel.util.jsse.SSLContextParameters 实例的 SSL 配置。		SSLContextParameters
camel.component.consul.configuration.tags	设置标签。您可以使用逗号分隔多个标签。		Set
camel.component.consul.configuration.url	Consul 代理 URL		字符串
camel.component.consul.configuration.user-name	设置用于基本身份验证的用户名		字符串
camel.component.consul.configuration.value-as-string	default 将从 Consul i.e. on KV 端点中检索的值转换为字符串。		布尔值
camel.component.consul.configuration.write-timeout-millis	OkHttpClient 的写超时		Long
camel.component.consul.datacenter	数据中心		字符串
camel.component.consul.enabled	启用 consul 组件	true	布尔值
camel.component.consul.health.check.repository.checks	定义要包含的检查。		list

Name	描述	默认值	类型
camel.component.consul.health.check.repository.configurations	健康检查配置。		Map
camel.component.consul.health.check.repository.enabled			布尔值
camel.component.consul.health.check.repository.failure-threshold			整数
camel.component.consul.health.check.repository.interval			字符串
camel.component.consul.password	设置用于基本身份验证的密码		字符串
camel.component.consul.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.consul.service-registry.acl-token			字符串
camel.component.consul.service-registry.attributes	自定义服务属性。		Map
camel.component.consul.service-registry.block-seconds			整数
camel.component.consul.service-registry.check-interval			整数

Name	描述	默认值	类型
camel.component.consul.service-registry.check-ttl			整数
camel.component.consul.service-registry.connect-timeout-millis			Long
camel.component.consul.service-registry.consistency-mode			ConsistencyMode
camel.component.consul.service-registry.datacenter			字符串
camel.component.consul.service-registry.deregister-after			整数
camel.component.consul.service-registry.deregister-services-on-stop			布尔值
camel.component.consul.service-registry.enabled	设置 consul 服务 registry 是否应启用，则默认为 false。	false	布尔值
camel.component.consul.service-registry.first-index			BigInteger
camel.component.consul.service-registry.id	Service Registry ID		字符串
camel.component.consul.service-registry.near-node			字符串

Name	描述	默认值	类型
camel.component.consul.service-registry.node-meta			list
camel.component.consul.service-registry.order	服务查找顺序/优先级.		整数
camel.component.consul.service-registry.override-service-host			布尔值
camel.component.consul.service-registry.password			字符串
camel.component.consul.service-registry.ping-instance			布尔值
camel.component.consul.service-registry.read-timeout-millis			Long
camel.component.consul.service-registry.recursive			布尔值
camel.component.consul.service-registry.service-host			字符串
camel.component.consul.service-registry.ssl-context-parameters			SSLContextParameters
camel.component.consul.service-registry.tags			Set

Name	描述	默认值	类型
camel.component.consul.service-registry.url			字符串
camel.component.consul.service-registry.user-name			字符串
camel.component.consul.service-registry.write-timeout-millis			Long
camel.component.consul.ssl-context-parameters	使用 org.apache.camel.util.jsse.SSLContextParameters 实例的 SSL 配置。选项是 org.apache.camel.util.jsse.SSLContextParameters 类型。		字符串
camel.component.consul.url	Consul 代理 URL		字符串
camel.component.consul.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值
camel.component.consul.user-name	设置用于基本身份验证的用户名		字符串
camel.component.consul.cluster.service.dc			字符串
camel.component.consul.configuration.dc	数据中心 @deprecated 替换为 {@link #setDatacenter (String)}（） }		字符串
camel.component.consul.service-registry.dc			字符串

69.4. HEADERS

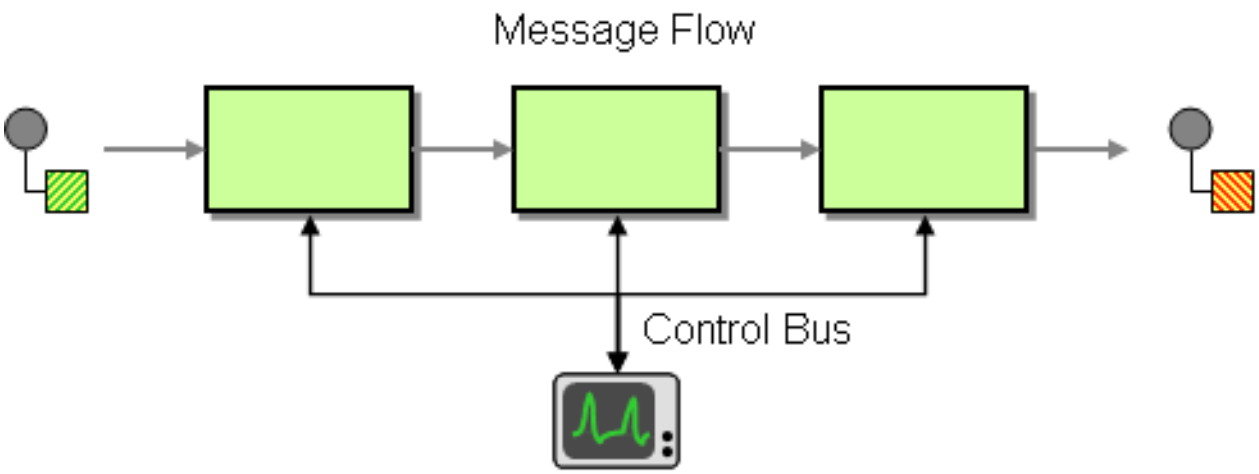
Name	类型	描述
Camel Consul Action	字符串	Producer 操作
Camel Consul Key	字符串	应应用该操作的密钥
Camel Consul EventId	字符串	event id （仅限使用者）
Camel Consul EventName	字符串	事件名称（仅限使用者）
Camel Consul EventLTime	Long	事件 LTime
Camel Consul NodeFilter	字符串	Node 过滤器
Camel Consul TagFilter	字符串	标签过滤器
Camel Consul SessionFilter	字符串	会话过滤器
Camel Consul Version	int	数据类型
Camel Consul Flags	Long	与值关联的标记
Camel Consul CreateIndex	Long	创建条目时代表的内部索引值

Name	类型	描述
Camel Consul LockIndex	Long	成功获取这个密钥的次数
Camel Consul ModifyIndex	Long	修改此密钥的最后索引
Camel Consul Options	对象	与请求关联的选项
Camel Consul Result	布尔值	如果响应的结果为 true
Camel Consul Session	字符串	会话 ID
Camel Consul ValueAsString	布尔值	将从 Consul i.e. on KV 端点中检索的值转换为字符串。

第 70 章 控制总线组件

从 Camel 版本 2.11 开始提供

EIP 模式中的控制 总线 允许从框架内监控和管理集成系统。



使用控制总线来管理企业集成系统。Control Bus 使用应用程序数据使用的相同消息传递机制，但使用单独的频道来传输与消息流涉及的组件管理相关的数据。

在 Camel 中，您可以使用 JMX 管理和监控，或使用 CamelContext 中的 Java API 或 org.apache.camel.api.management 软件包管理和监控，或者使用此处具有示例的事件通知程序。

从 Camel 2.11 开始，我们引入了一个新的 ControlBus 组件，可让您向相应响应的控制总线端点发送消息。

70.1. CONTROLBUS COMPONENT

从 Camel 2.11 开始提供

controlbus: 组件根据 控制总线 EIP 模式提供 Camel 应用程序轻松管理。例如，通过向 Endpoint 发送消息，您可以控制路由的生命周期或收集性能统计信息。

```
controlbus:command[?options]
```

其中 command 可以是任意字符串，以确定要使用的命令类型。

70.2. 命令

命令	描述
route	使用 **routeId** 和 **action** 参数控制路由。
language	允许您指定用于评估消息正文 的语言。如果评估中有任何结果，则结果将置于消息正文中。

70.3. 选项

Control Bus 组件没有选项。

Control Bus 端点使用 URI 语法进行配置：

```
controlbus:command:language
```

使用以下路径和查询参数：

70.3.1. 路径参数(2 参数)：

Name	描述	默认值	类型
命令	**所需的命令** 可以是 route 或 language		字符串
language	允许您指定用于评估消息正文的语言名称。如果评估中有任何结果，则结果将置于消息正文中。		语言

70.3.2. 查询参数(6 参数)：

Name	描述	默认值	类型

Name	描述	默认值	类型
action (producer)	表示可以：start、stop 或 status 的操作。要启动或停止路由，或者在消息正文中以输出形式获取路由的状态。您可以使用 suspend 并从 Camel 2.11.1 中恢复，以挂起或恢复路由。从 Camel 2.11.1 开始，您可以使用 stats 来以 XML 格式返回性能静态。如果未定义 routeId，则可使用 routeId 选项定义哪个路由来获取性能统计信息，如果未定义 routeId，则您会收到整个 CamelContext 的统计信息。restart 操作将重启路由。		字符串
async (producer)	是否异步执行控制总线任务。重要：如果启用了这个选项，则不会在交换上设置任务中的任何结果。这只有在同步执行任务时才可能。	false	布尔值
loggingLevel (producer)	任务完成后用于日志记录的日志记录级别，或者在处理任务过程中出现异常。	INFO	LoggingLevel
restartDelay (producer)	重启路由时要使用的延迟。	1000	int
routeId (producer)	通过 id 指定路由。special 关键字 current 指示当前路由。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

70.4. 使用 ROUTE 命令

route 命令允许您在给定路由上轻松执行常见任务，例如启动路由，您可以将空消息发送到此端点：

```
template.sendBody("controlbus:route?routeId=foo&action=start", null);
```

要获取路由的状态，您可以执行以下操作：

```
String status = template.requestBody("controlbus:route?routeId=foo&action=status", null,
String.class);
```

70.5. 获取性能统计

从 Camel 2.11.1 开始提供

这需要启用 JMX （默认情况下，每个路由的性能静态），或 CamelContext。例如，要获取名为 foo 的路由的静态命令，我们可以：

```
String xml = template.requestBody("controlbus:route?routeId=foo&action=stats", null,
String.class);
```

返回的静态采用 XML 格式。它与 JMX 获取的数据与 ManagedRouteMBean 上的 dumpRouteStatsAsXml 操作获取。

要获得整个 CamelContext 的静态项，您只省略了 routeId 参数，如下所示：

```
String xml = template.requestBody("controlbus:route?action=stats", null, String.class);
```

70.6. 使用简单语言

您可以将 Simple 语言与控制总线一起使用，例如停止特定路由，您可以向 "controlbus:language:simple" 端点发送消息，其中包含以下信息：

```
template.sendBody("controlbus:language:simple", "${camelContext.stopRoute('myRoute')}");
```

因此这是 void 操作，因此不会返回任何结果。但是，如果要路由状态，您可以执行以下操作：

```
String status = template.requestBody("controlbus:language:simple",
"${camelContext.getRouteStatus('myRoute')}", String.class);
```

使用 route 命令更轻松地控制路由的生命周期。language 命令允许您执行具有更强大的电源的语言脚本，如 Groovy 或扩展某些 简单 语言。

例如，要关闭 Camel 本身，您可以执行以下操作：

```
template.sendBody("controlbus:language:simple?async=true", "${camelContext.stop()}");
```

我们使用 async=true 异步停止 Camel，否则我们将尝试停止 Camel，同时在处理我们发送到控制总线组件的消息时尝试停止 Camel。

提示

您还可以使用其他语言，如 Groovy 等等。

第 71 章 COUCHBASE 组件

从 Camel 版本 2.19 开始提供

couchbase: 组件允许您将 CouchBase 实例视为生成者或消息使用者。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-couchbase</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

71.1. URI 格式

```
couchbase:url
```

71.2. 选项

Couchbase 组件没有选项。

Couchbase 端点使用 URI 语法进行配置：

```
couchbase:protocol:hostname:port
```

使用以下路径和查询参数：

71.2.1. 路径参数(3 参数)：

Name	描述	默认值	类型
protocol	要使用的协议		字符串
hostname	**必需** 使用的主机名		字符串

Name	描述	默认值	类型
port	要使用的端口号	8091	int

71.2.2. 查询参数(47 参数)：

Name	描述	默认值	类型
bucket (common)	要使用的存储桶		字符串
key (common)	要使用的密钥		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
consumerProcessedStrategy (consumer)	定义要使用的消费者流程策略	none	字符串
降序 (consumer)	定义此操作是否降序	false	布尔值
designDocumentName (consumer)	要使用的设计文档名称	Beer	字符串
limit (consumer)	要使用的输出限制	-1	int
rangeEndKey (consumer)	为 end 键定义范围		字符串
rangeStartKey (consumer)	为 start 键定义范围		字符串
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
skip (consumer)	定义要使用的跳过	-1	int
viewName (consumer)	要使用的视图名称	brewery_beers	字符串

Name	描述	默认值	类型
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
autoStartIdForInserts (producer)	定义我们在执行插入操作时是否希望自动启动 Id	false	布尔值
operation (producer)	要执行的操作	CCB_PUT	字符串
persistTo (producer)	持久数据的位置	0	int
producerRetryAttempts (producer)	定义重试尝试的数量	2	int
producerRetryPause (producer)	定义不同尝试之间重试暂停	5000	int
replicateTo (producer)	复制数据的位置	0	int
startingIdForInsertsFrom (producer)	定义我们进行插入操作的起始 Id		long
additionalHosts (advanced)	其他主机		字符串
maxReconnectDelay (advanced)	在重新连接过程中定义最大延迟	30000	long
obsPollInterval (advanced)	定义观察轮询间隔	400	long
obsTimeout (advanced)	定义观察超时	-1	long

Name	描述	默认值	类型
opQueueMaxBlockTime (advanced)	定义操作在队列中阻止的最大时间	10000	long
opTimeOut (advanced)	定义操作超时	2500	long
readBufferSize (advanced)	定义缓冲区大小	16384	int
shouldOptimize (advanced)	定义是否要在可能的情况下使用优化	false	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
timeoutExceptionThreshold (advanced)	定义抛出 timeout Exception 的阈值	998	int
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel

Name	描述	默认值	类型
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
密码 (security)	要使用的密码		字符串
用户名 (security)	要使用的用户名		字符串

71.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.couchbase.enabled	启用 couchbase 组件	true	布尔值
camel.component.couchbase.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 72 章 COUCHDB 组件

从 Camel 版本 2.11 开始提供

couchdb: 组件允许您将 CouchDB 实例视为生成者或消息使用者。使用轻量级 LightCouch API，这个 camel 组件具有以下功能：

- 作为消费者，monitor couch changesets 用于插入、更新和删除这些消息并将其发布为 camel 路由。
- 作为制作者，可以从 Camel 2.18 删除（使用带有 DELETE 值的 CouchDbMethod）文档和 Camel 2.22 获取文档（通过将 CouchDbMethod 与 GET 值搭配使用）来保存、更新。
- 可以根据需要支持多个端点，例如多个实例间的多个数据库。
- 只能删除事件触发器，只删除插入/更新或全部（默认）。
- 为 sequenceId、文档修订、文档 ID 和 HTTP 方法类型的标头。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-couchdb</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

72.1. URI 格式

```
couchdb:http://hostname[:port]/database?[options]
```

其中 hostname 是正在运行的 couchdb 实例的主机名。port 是可选的；如果没有指定，则默认为 5984。

72.2. 选项

CouchDB 组件没有选项。

CouchDB 端点使用 URI 语法进行配置：

```
couchdb:protocol:hostname:port/database
```

使用以下路径和查询参数：

72.2.1. 路径参数(4 参数)：

Name	描述	默认值	类型
protocol	**必需** 用于与数据库通信的协议。		字符串
hostname	正在运行的 couchdb 实例 **所需的** 主机名		字符串
port	正在运行的 couchdb 实例的端口号	5984	int
database	要使用的数据库 **所需的** 名称		字符串

72.2.2. 查询参数(12 参数)：

Name	描述	默认值	类型
createDatabase (common)	创建数据库（如果尚不存在）	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
删除 (consumer)	文档删除作为事件发布	true	布尔值
heartbeat (consumer)	发送空消息以保持套接字在 millis 中的频率	30000	long

Name	描述	默认值	类型
since (consumer)	在给定的更新序列后立即开始跟踪更改。默认 null 将从最新的序列开始监控。		字符串
style (consumer)	指定 changes 数组中返回的修订版本数量。默认的 main_only 将只返回当前的获奖修订；all_docs 将返回所有叶修订版本（包括冲突和删除的前冲突）。	main_only	字符串
updates (consumer)	文档插入/更新作为事件发布	true	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
密码 (security)	验证的数据库的密码		字符串
用户名 (security)	身份验证数据库的用户名		字符串

72.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.couchdb.enabled	启用 couchdb 组件	true	布尔值
camel.component.couchdb.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

72.4. HEADERS

在消息传输过程中，在交换上设置以下标头。

属性	value
CouchDbDatabase	消息来自的数据库
CouchDbSeq	update / delete 消息的 couchdb changeset 序列号
CouchDbId	couchdb 文档 id
CouchDbRev	couchdb 文档修订
CouchDbMethod	方法（删除 / update）

标头由消费者设置，在收到消息后。在插入/更新发生后，生成者还会为下游处理器设置标头。生成者前设置的任何标头都会被忽略。例如，如果您将 CouchDbId 设置为标头，它不会用作插入的 id，则文档的 id 仍会被使用。

72.5. 消息正文

组件将使用消息正文作为要插入的文档。如果正文是 String 实例，则在插入前，它将会被放入 GSON 对象中。这意味着字符串必须是有效的 JSON，否则插入 / 更新将失败。如果正文是 com.google.gson.JsonElement 的实例，则它将按原样插入。否则，生成者会抛出不支持的正文类型例外。

72.6. SAMPLES

例如，如果您想要消耗所有插入、更新和删除本地的 CouchDB 实例，在端口 9999 上，您可以使用以下内容：

```
from("couchdb:http://localhost:9999").process(someProcessor);
```

如果您只想删除，您可以使用以下内容

```
from("couchdb:http://localhost:9999?updates=false").process(someProcessor);
```

如果要将消息作为文档插入，则使用交换的正文

```
from("someProducingEndpoint").process(someProcessor).to("couchdb:http://localhost:9999")
```

第 73 章 CASSANDRA CQL 组件

从 Camel 版本 2.15 开始提供

Apache Cassandra 是一种开源 NoSQL 数据库，设计为在商业硬件上处理大量。与 Amazon 的 DynamoDB 一样，Cassandra 也有一个对等式和无主架构，以避免单一故障点和保证高可用性。与 Google 的 BigTable 一样，Cassandra 数据采用列系列结构，这些系列可通过 Thrift RPC API 或称为 CQL 的类 SQL API 进行访问。

此组件旨在使用 CQL3 API （而不是 Thrift API）集成 Cassandra 2.0+。它基于由 DataStax 提供的 Cassandra Java 驱动程序。

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-cassandraql</artifactId>
  <version>x.y.z</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

73.1. URI 格式

端点可以启动 Cassandra 连接，或者使用现有的连接。

URI	描述
cql:localhost/keyspace	单个主机、默认端口，通常用于测试
cql:host1,host2/keyspace	多主机, 默认端口
cql:host1,host2:9042/keyspace	多主机，自定义端口
cql:host1,host2	默认端口和键空间
cql:bean:sessionRef	提供的会话参考

URI	描述
cql:bean:clusterRef/keyspace	提供的集群参考

要微调 Cassandra 连接(SSL 选项、池选项、负载平衡策略、重试策略、重新连接策略...)，请创建自己的集群实例，并将其提供给 Camel 端点。

73.2. CASSANDRA 选项

Cassandra CQL 组件没有选项。

Cassandra CQL 端点使用 URI 语法进行配置：

```
cql:beanRef:hosts:port/keyspace
```

使用以下路径和查询参数：

73.2.1. 路径参数(4 参数)：

Name	描述	默认值	类型
beanRef	BeanRef 使用 bean:id 定义		字符串
主机	hostname (s) cassansdra server (s)。可以使用逗号分隔多个主机。		字符串
port	cassansdra 服务器的端口号		整数
keyspace	要使用的键空间		字符串

73.2.2. 查询参数(29 参数)：

Name	描述	默认值	类型
cluster (common)	要使用集群实例（您通常不使用这个选项）		Cluster
clusterName (common)	集群名称		字符串

Name	描述	默认值	类型
consistencyLevel (common)	使用的一致性级别		ConsistencyLevel
cql (common)	要执行的 CQL 查询。可以通过带有键 CamelCqlQuery 的 message 标头覆盖。		字符串
loadBalancingPolicy (common)	使用特定的 LoadBalancingPolicy		字符串
password (common)	用于会话验证的密码		字符串
prepareStatements (common)	使用 PreparedStatements 还是常规声明	true	布尔值
resultSetConversionStrategy (common)	使用一个实现将 ResultSet 转换为消息 body ALL, ONE, LIMIT_10, LIMIT_100... 的自定义类		字符串
session (common)	要使用 Session 实例（您通常不使用这个选项）		会话
username (common)	会话身份验证的用户名		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern

Name	描述	默认值	类型
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值

Name	描述	默认值	类型
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

73.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.cql.enabled	启用 cql 组件	true	布尔值
camel.component.cql.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

73.4. 消息

73.4.1. 传入消息

Camel Cassandra 端点需要一系列简单的对象(Object 或 Object[] 或 Collection<Object>)，它将绑定到 CQL 语句作为查询参数。如果消息正文为空，则将执行 CQL 查询而无需绑定参数。

headers:

- *CamelCqlQuery （可选，字符串 或 常规状态）：CQL 查询为普通字符串或使用 QueryBuilder 构建。*

73.4.2. 传出消息

根据 resultSetConversionStrategy，Camel Cassandra 端点生成一个或多个 Cassandra Row 对象：

- List<Row > 如果 resultSetConversionStrategy 为 ALL 或 LIMIT_[0-9]+
- 如果 resultSetConversionStrategy 为 ONE，则单一的 Row'
- 否则，如果 resultSetConversionStrategy 是 ResultSetConversionStrategy的自定义实现

73.5. 软件仓库

Cassandra 可用于存储幂等和聚合 EIP 的消息消息。

Cassandra 可能不是排队用例的最佳工具，但读取 Cassandra 反模式队列和队列，如数据集。建议您使用 LeveledCompaction 和一个小的 GC grace 设置来允许快速删除 tombstoned 行。

73.6. 幂等软件仓库

NamedCassandraIdempotentRepository 在 Cassandra 表中存储消息密钥，如下所示：

CAMEL_IDEMPOTENT.cql

```
CREATE TABLE CAMEL_IDEMPOTENT (
  NAME varchar,   -- Repository name
  KEY varchar,    -- Message key
  PRIMARY KEY (NAME, KEY)
) WITH compaction = {'class':'LeveledCompactionStrategy'}
  AND gc_grace_seconds = 86400;
```

此存储库实施使用轻量级事务（也称为 Compare 和 Set），并且需要 Cassandra 2.0.7+。

或者，CassandraIdempotentRepository 没有 NAME 列，并可扩展为使用不同的数据模型。

选项	默认	描述
table	**CAMEL_IDEMPOTENT**	表名称
pkColumns	**名称**,' 键'	主要键列

选项	默认	描述
name		存储库名称，用于 **NAME** 列的值
ttl		实时的关键时间
writeConsistencyLevel		用于插入/删除密钥的一致性级别：**ANY、ONE、TWO、QUORUM、LOCAL_QUORUM**...
readConsistencyLevel		用于读取/检查密钥的一致性级别：**ONE、TWO、QUORUM、LOCAL_QUORUM**...

73.7. 聚合存储库

NamedCassandraAggregationRepository 根据 correlation 键在 Cassandra 表中存储交换，如下所示：

CAMEL_AGGREGATION.cql

```
CREATE TABLE CAMEL_AGGREGATION (
  NAME varchar,       -- Repository name
  KEY varchar,        -- Correlation id
  EXCHANGE_ID varchar, -- Exchange id
  EXCHANGE blob,      -- Serialized exchange
  PRIMARY KEY (NAME, KEY)
) WITH compaction = {'class':'LeveledCompactionStrategy'}
  AND gc_grace_seconds = 86400;
```

或者，CassandraAggregationRepository 没有 NAME 列，并可扩展为使用不同的数据模型。

选项	默认	描述
table	**CAMEL_AGGREGATION**	表名称
pkColumns	**名称，键**	主要键列
exchangeIdColumn	**EXCHANGE_ID**	Exchange Id 列
exchangeColumn	**EXCHANGE**	Exchange content 列

选项	默认	描述
name		存储库名称，用于 **NAME** 列的值
ttl		交换生存时间
writeConsistencyLevel		用于插入/删除交换的一致性级别：任意、**ONE**、**TWO**、**QUORUM**、**LOCAL_QUORUM**...
readConsistencyLevel		用于读取/检查交换的一致性级别：**ONE**、**TWO**、**QUORUM**、**LOCAL_QUORUM**...

73.8. 例子

要在表中插入内容，您可以使用以下代码：

```
String CQL = "insert into camel_user(login, first_name, last_name) values (?, ?, ?)";
from("direct:input")
    .to("cql://localhost/camel_ks?cql=" + CQL);
```

此时，您应能够使用列表作为正文来插入数据

```
Arrays.asList("davsclaus", "Claus", "Ibsen")
```

相同的方法可用于更新或查询表。

第 74 章 CRYPTO (JCE)组件

从 Camel 版本 2.3 开始提供

使用 Camel 加密端点和 Java 的加密扩展，可轻松为交换创建数字签名。Camel 提供了一对灵活的端点，它们被用来在交换工作流的一个部分为交换创建签名，然后在工作流的后续部分中验证签名。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-crypto</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

74.1. 简介

数字签名利用 Asymmetric Cryptographic 技术为消息签名。从(very)高级别中，算法使用带有特殊属性的免费密钥对，数据通过一个密钥加密的属性只能与其他密钥进行解密。某个私钥（私钥）被严格保护，用于"注册"消息，而其他公钥则与有兴趣验证签名消息的任何人共享。消息通过使用私钥加密消息摘要来签名。这个加密摘要会与消息一起传输。在另一个端，verifier 重新计算消息摘要，并使用公钥解密签名中的摘要。如果两个摘要都与 verifier 知道只有私钥的拥有者才能创建签名。

Camel 使用 Java Cryptographic Extension 中的签名服务执行创建交换签名所需的所有重度加密滞后。以下是说明加密、消息摘要和数字签名机制的一些极佳资源，以及如何利用 JCE。

- Bruce Schneier 的 Applied Cryptography
- 以 Java 为 David Hook 开始加密
- 以前的 insightsful Wikipedia Digital_signatures

74.2. URI 格式

如前文所述，Camel 提供了一对加密端点来创建和验证签名

```
crypto:sign:name[?options]
crypto:verify:name[?options]
```

- crypto:sign 创建签名，并将其存储在由恒定的 org.apache.camel.component.crypto.DigitalSignatureConstants.SIGNATURE，即.e。"CamelDigitalSignature".

- crypto:verify 将在此标头的内容中读取，并执行验证计算。

为了正常工作，签名和验证过程需要共享一对密钥，签名需要 PrivateKey 和验证公钥（或包含 1 的证书）。使用 JCE 生成这些密钥对非常简单，但通常最好使用 KeyStore 向内部和共享您的密钥。DSL 对提供密钥的方式非常灵活，提供了多个机制。

请注意，一个 crypto:sign 端点通常在一个路由中定义，且免费的 crypto:verify 则在另一个路由后出现。它会说，签名和验证都应以相同方式配置。

74.3. 选项

Crypto (JCE)组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	使用共享 DigitalSignatureConfiguration 作为配置		数字签名配置
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Crypto (JCE)端点使用 URI 语法进行配置：

```
crypto:cryptoOperation:name
```

使用以下路径和查询参数：

74.3.1. 路径参数(2 参数)：

Name	描述	默认值	类型
cryptoOperation	**必需** 在端点 uri .g. crypto:sign 设置作为操作中的加密方案后提供的 Crypto 操作。		CryptoOperation
name	**必需** 此操作的逻辑名称。		字符串

74.3.2. 查询参数(19 参数)：

Name	描述	默认值	类型
algorithm (producer)	设置用于签名者的 Algorithm 的 JCE 名称。	SHA1WithDSA	字符串
alias (producer)	设置用于查询 KeyStore 的别名，并链接 java.security.cert.Certificate 证书以签名和验证交换。此值可以通过消息标头 org.apache.camel.component.crypto.DigitalSignatureConstantsKEYSTOREKEYSTORE_ALIAS 在运行时提供		字符串
certificateName (producer)	为 registry 中可以 fond 的 PrivateKey 设置引用名称。		字符串
keystore (producer)	设置 KeyStore，可以包含用于签名和验证交换的密钥和 Certficates。KeyStore 通常与别名一起使用，可以是 Route 定义中提供的，或者通过消息标头 CamelSignatureKeyStoreAlias 动态使用。如果没有提供别名，且在 Keystore 中只有一个条目，则会使用这个单个条目。		KeyStore
keystoreName (producer)	为 registry 中可以 fond 的密钥存储设置引用名称。		字符串
privatekey (producer)	设置用于签署交换的 PrivateKey		PrivateKey
privateKeyName (producer)	为 registry 中可以 fond 的 PrivateKey 设置引用名称。		字符串
provider (producer)	设置提供配置的签名算法的安全供应商的 ID。		字符串
publicKeyName (producer)	上下文更改时应解析的引用		字符串
secureRandomName (producer)	为 registry 中可以 fond 的 SecureRandom 设置引用名称。		字符串

Name	描述	默认值	类型
signatureHeaderName (producer)	设置用于存储 base64 编码签名的消息标头的名称。默认为 'CamelDigitalSignature'		字符串
bufferSize (advanced)	设置在 Exchange payload 数据中用于读取的缓冲区的大小。	2048	整数
证书 （高级）	设置应在交换中根据其有效负载验证签名的证书。		证书
clearHeaders (advanced)	决定在签名和验证后是否清除 Signature 特定标头。默认为 true，并且只应在您的极端考虑，因为密钥和密码等重要私有信息（如果未设置）可能会转义。	true	布尔值
keyStoreParameters (advanced)	设置 KeyStore，可以包含密钥和 Certficates，用于根据给定的 KeyStoreParameters 签名和验证交换。KeyStore 通常与别名一起使用，可以是 Route 定义中提供的，或者通过消息标头 CamelSignatureKeyStoreAlias 动态使用。如果没有提供别名，且在 Keystore 中只有一个条目，则会使用这个单个条目。		KeyStoreParameters
公钥(advanced)	设置应在交换中验证签名的 PublicKey。		PublicKey
SecureRandom (advanced)	设置用于初始化签名服务的 SecureRandom		SecureRandom
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
密码 (security)	设置用于访问 KeyStore 中别名的 PrivateKey 的密码。		字符串

74.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 33 选项，如下所列。

Name	描述	默认值	类型
camel.component.crypto.configuration.algorithm	设置用于签名者的 Algorithm 的 JCE 名称。	SHA1WithDSA	字符串

Name	描述	默认值	类型
camel.component.crypto.configuration.alias	设置用于查询 KeyStore 的别名，并链接 java.security.cert.Certificate 证书以签名和验证交换。这个值可以通过消息标头 org.apache.camel.component.crypto.DigitalSignatureConstants #KEYSTORE_ALIAS 在运行时提供		字符串
camel.component.crypto.configuration.buffer-size	设置在 Exchange payload 数据中用于读取的缓冲区的大小。	2048	整数
camel.component.crypto.configuration.certificate	设置应在交换中根据其有效负载验证签名的证书。		证书
camel.component.crypto.configuration.certificate-name	为 registry 中可以 fond 的 PrivateKey 设置引用名称。		字符串
camel.component.crypto.configuration.clear-headers	决定在签名和验证后是否清除 Signature 特定标头。默认为 true，并且只应在您的极端考虑，因为密钥和密码等重要私有信息（如果未设置）可能会转义。	true	布尔值
camel.component.crypto.configuration.crypto-operation	在 endpoint uri .g. crypto:sign set sign as the operation 中的 crypto scheme 后设置 Crypto 操作。		CryptoOperation
camel.component.crypto.configuration.key-store-parameters	设置 KeyStore，可以包含密钥和 Certficates，用于根据给定的 KeyStoreParameters 签名和验证交换。KeyStore 通常与别名一起使用，可以是 Route 定义中提供的，或者通过消息标头 CamelSignatureKeyStoreAlias 动态使用。如果没有提供别名，且在 Keystore 中只有一个条目，则会使用这个单个条目。		KeyStoreParameters
camel.component.crypto.configuration.keystore	设置 KeyStore，可以包含用于签名和验证交换的密钥和 Certficates。KeyStore 通常与别名一起使用，可以是 Route 定义中提供的，或者通过消息标头 CamelSignatureKeyStoreAlias 动态使用。如果没有提供别名，且在 Keystore 中只有一个条目，则会使用这个单个条目。		KeyStore
camel.component.crypto.configuration.keystore-name	为 registry 中可以 fond 的密钥存储设置引用名称。		字符串

Name	描述	默认值	类型
camel.component.crypto.configuration.name	此操作的逻辑名称。		字符串
camel.component.crypto.configuration.password	设置用于访问 KeyStore 中别名的 PrivateKey 的密码。		character[]
camel.component.crypto.configuration.private-key	设置用于签署交换的 PrivateKey		PrivateKey
camel.component.crypto.configuration.private-key-name	为 registry 中可以 fond 的 PrivateKey 设置引用名称。		字符串
camel.component.crypto.configuration.provider	设置提供配置的签名算法的安全供应商的 ID。		字符串
camel.component.crypto.configuration.public-key	设置应在交换中验证签名的 PublicKey。		PublicKey
camel.component.crypto.configuration.public-key-name	上下文更改时应解析的引用		字符串
camel.component.crypto.configuration.secure-random	设置用于初始化签名服务的 SecureRandom		SecureRandom
camel.component.crypto.configuration.secure-random-name	为 registry 中可以 fond 的 SecureRandom 设置引用名称。		字符串
camel.component.crypto.configuration.signature-header-name	设置用于存储 base64 编码签名的消息标头的名称。默认为 'CamelDigitalSignature'		字符串
camel.component.crypto.enabled	启用加密组件	true	布尔值

Name	描述	默认值	类型
camel.component.crypto.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.dataformat.crypto.algorithm	指示要使用的加密算法的 JCE 算法名称。默认为 DES/CBC/PKCS5Padding。	DES/CBC/PKCS5Padding	字符串
camel.dataformat.crypto.algorithm-parameter-ref	用于初始化 Cipher 的 JCE AlgorithmParameterSpec。将指定名称用作 java.security.spec.AlgorithmParameterSpec 类型来查找类型。		字符串
camel.dataformat.crypto.buffersize	签名进程中使用的缓冲区的大小。		整数
camel.dataformat.crypto.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.crypto.crypto-provider	应使用的 JCE Security Provider 的名称。		字符串
camel.dataformat.crypto.enabled	启用加密数据	true	布尔值
camel.dataformat.crypto.init-vector-ref	指的是包含 Initialization Vector 的字节数组，用于初始化 Cipher。		字符串
camel.dataformat.crypto.inline	表示配置的 IV 应该内联到加密的数据流的标志。默认为 false。	false	布尔值
camel.dataformat.crypto.key-ref	是指要从寄存器中查询的 secret 密钥。		字符串
camel.dataformat.crypto.mac-algorithm	指示消息身份验证算法的 JCE 算法名称。	HmacSHA1	字符串
camel.dataformat.crypto.should-append-h-m-a-c	表示消息身份验证代码应计算并附加到加密的数据的标志。	false	布尔值

74.5. 使用

74.5.1. 原始键

签署和验证交换的最基本方法就是 KeyPair，如下所示：

可使用对密钥的引用通过 Spring XML 扩展 来实现相同的操作

74.5.2. keystores 和 Aliases.

JCE 提供了非常灵活的密钥存储概念，用于托管私钥和证书对，保持加密和密码受到保护。可以通过将别名应用到检索 API 来检索它们。可以通过多种方式将密钥和证书放入密钥存储中，这通常是通过外部的"keytool"应用程序完成的。这是使用 keytool 使用自签名 Cert 和 Private 密钥创建 KeyStore 的良好示例。

示例使用由"bob"别名的密钥和证书的密钥存储。密钥存储的密码和密钥是 'letmein'

以下演示了如何通过 Fluent 构建器使用密钥存储，它还演示了如何加载和初始化密钥存储。

在 Spring a ref 中再次用于查找实际的密钥存储实例。

74.5.3. 更改 JCE 提供程序和算法

更改签名算法或安全供应商是指定其名称的简单关系。您还需要使用与您选择的算法兼容的密钥。

or

74.5.4. 更改签名消息标头

可能需要更改用于存储签名的消息标头。可以在路由定义中指定不同的标头名称，如下所示

or

74.5.5. 更改 buffersize

如果您需要更新 buffer... 的大小

or

74.5.6. 动态提供密钥。

当使用 Recipient 列表或类似的 EIP 时，交换的接收者可能会动态变化。在所有接收者间使用相同的密钥可能并不可行。可以根据每个交换来动态指定签名密钥。然后，在签名前，可以使用其目标接收者的密钥动态增强交换。为了便于此签名机制，允许通过以下消息标头动态提供密钥

- Exchange.SIGNATURE_PRIVATE_KEY, "CamelSignaturePrivateKey"
- Exchange.SIGNATURE_PUBLIC_KEY_OR_CERT, "CamelSignaturePublicKeyOrCert"

or

最好动态提供密钥存储别名。同样，别名可以在消息标头中提供

- Exchange.KEYSTORE_ALIAS, "CamelSignatureKeyStoreAlias"

or

标头将设置如下

```
Exchange unsigned = getMandatoryEndpoint("direct:alias-sign").createExchange();
unsigned.getIn().setBody(payload);
unsigned.getIn().setHeader(DigitalSignatureConstants.KEYSTORE_ALIAS, "bob");
unsigned.getIn().setHeader(DigitalSignatureConstants.KEYSTORE_PASSWORD,
"letmein".toCharArray());
template.send("direct:alias-sign", unsigned);
Exchange signed = getMandatoryEndpoint("direct:alias-sign").createExchange();
signed.getIn().copyFrom(unsigned.getOut());
signed.getIn().setHeader(KEYSTORE_ALIAS, "bob");
template.send("direct:alias-verify", signed);
```

74.6. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 75 章 CRYPTO CMS 组件

从 Camel 版本 2.20 开始提供

加密消息语法(CMS) 是签名和加密消息的良好标准。Apache Crypto CMS 组件支持此标准的以下部分：* Content Type "Enveloped Data" with Key Transport (asymmetric key), * Content Type "Signed Data"。您可以创建 CMS Enveloped Data 实例，解密 CMS Enveloped Data 实例，创建 CMS Signed Data 实例，并验证 CMS Signed Data 实例。

组件使用 Bouncy Castle 库 bcprov-jdk18on 和 bcpkix-jdk18on。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-crypto-cms</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

建议您在调用此组件的端点前在应用程序中注册 Bouncy Castle 安全供应商：

```
Security.addProvider(new BouncyCastleProvider());
```

如果没有注册 Bouncy Castle 安全供应商，则 Crypto CMS 组件将注册该提供程序。

75.1. 选项

Crypto CMS 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
signedDataVerifier 配置 (advanced)	要配置共享 SignedDataVerifierConfiguration，它将决定验证操作的 uri 参数。		SignedDataVerifier Configuration

Name	描述	默认值	类型
envelopedDataDecryptor Configuration (advanced)	要配置共享 EnvelopedDataDecryptorConfiguration，它将决定解密操作的 uri 参数。		EnvelopedDataDecryptor Configuration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Crypto CMS 端点使用 URI 语法进行配置：

```
crypto-cms:cryptoOperation:name
```

使用以下路径和查询参数：

75.1.1. 路径参数(2 参数)：

Name	描述	默认值	类型
cryptoOperation	**必需的** 在端点 uri.g. crypto-cms:sign 设置签名作为操作的加密方案后提供的 Crypto 操作。可能的值有：sign, verify, encrypt, 或 decrypt。		CryptoOperation
name	**必需** URI 中的 name 部分可由用户选择，以区分 camel 上下文中的不同 signer/verifier/encryptor/decryptor 端点。		字符串

75.1.2. 查询参数(15 参数)：

Name	描述	默认值	类型
keystore (common)	根据操作，密钥存储包含签名人私钥、verifier 公钥、加密密钥加密密钥、解密或私钥。使用此参数或参数 'keyStoreParameters'。		KeyStore
keyStoreParameters (common)	根据操作，包含签名人私钥、verifier 公钥、加密密钥加密密钥、解密器私钥的密钥存储。使用此参数或参数 'keystore'。		KeyStoreParameters
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

Name	描述	默认值	类型
密码 （解密）	设置私钥的密码。假设密钥存储中的所有私钥具有相同的密码。如果没有设置，则假定私钥的密码由 KeyStoreParameters 中指定的密钥存储密码提供。		char[]
fromBase64 (decrypt_verify)	如果为 true，则 CMS 消息采用 base 64 编码，必须在处理过程中被解码。默认值为 false。	false	布尔值
contentEncryptionAlgorithm (encrypt)	加密算法，如 DESede/CBC/PKCS5Padding。其它可能的值：DESede/CBC/PKCS5Padding, AES/CBC/PKCS5Padding, Camellia/CBC/PKCS5Padding, CAST5/CBC/PKCS5Padding.		字符串
originatorInformation Provider (encrypt)	原始器信息的供应商。请参阅 https://tools.ietf.org/html/rfc5652#section-6.1。默认值为 null。		OriginatorInformation Provider
接收者 （加密）	接收者信息：引用实现接口 org.apache.camel.component.crypto.cms.api.TransRecipientInfo 的 bean		list
secretKeyLength (encrypt)	用于内容加密的 secret 对称密钥的密钥长度。仅在指定的 content-encryption 算法允许不同大小的密钥时使用。如果 contentEncryptionAlgorithm=AES/CBC/PKCS5Padding 或 Camellia/CBC/PKCS5Padding 随后 128；如果 contentEncryptionAlgorithm=DESede/CBC/PKCS5Padding，则后续 192、128；如果对 AES/CBC/PKCS5Padding 和 Camellia/CBC/PKCS5Padding 也启用了密钥长度。		int
unprotectedAttributes GeneratorProvider (encrypt)	取消保护属性的生成器的提供商。默认值为 null，这意味着没有未保护的属性添加到 Enveloped Data 对象中。请参阅 https://tools.ietf.org/html/rfc5652#section-6.1。		AttributesGenerator Provider
toBase64 (encrypt_sign)	指明 Signed Data 或 Enveloped Data 实例是否应采用 base64 编码。默认值为 false。	false	布尔值
includeContent (sign)	指明签名的内容是否应包含在 Signed Data 实例中。如果为 false，则在标头 CamelCryptoCmsSignedData 中创建分离的 Signed Data 实例。	true	布尔值
签名者 （签名）	签名者信息：引用实现 org.apache.camel.component.crypto.cms.api.SignerInfo 的 bean		list

Name	描述	默认值	类型
signedDataHeaderBase64 (verify)	指明标头 CamelCryptoCmsSignedData 中的值是否采用 base64 编码。默认值为 false。只适用于分离的签名。在分离的签名情形中，标头包含 Signed Data 对象。	false	布尔值
verifySignaturesOfAll Signers (verify)	如果为 true，则会验证 Signed Data 对象中包含的所有签名者的签名。如果为 false，则只验证签名者信息与其中一个指定证书匹配的签名。默认值为 true。	true	布尔值

75.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.crypto-cms.enabled	是否启用 crypto-cms 组件的自动配置。这默认是启用的。		布尔值
camel.component.crypto-cms.enveloped-data-decryptor-configuration	要配置共享 EnvelopedDataDecryptorConfiguration，它将决定解密操作的 uri 参数。选项是 org.apache.camel.component.crypto.cms.crypt。EnvelopedDataDecryptorConfiguration 类型。		字符串
camel.component.crypto-cms.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.crypto-cms.signed-data-verifier-configuration	要配置共享 SignedDataVerifierConfiguration，它将决定验证操作的 uri 参数。选项是 org.apache.camel.component.crypto.cms.sig.SignedDataVerifierConfiguration 类型。		字符串

75.3. ENVELOPED DATA

*请注意，一个 **crypto-cms:encypt** 端点通常在一个路由中定义，而免费的 **crypto-cms:decrypt** 则在另一个路由中定义，尽管在另一个示例后显示它们的示例。*

以下示例演示了如何创建 Enveloped Data 消息，以及如何解密 Enveloped Data 消息。

Java DSL 中的基本示例

```
import org.apache.camel.util.jsse.KeyStoreParameters;
import org.apache.camel.component.crypto.cms.crypt.DefaultKeyTransRecipientInfo;
...
KeyStoreParameters keystore  = new KeyStoreParameters();
keystore.setType("JCEKS");
keystore.setResource("keystore/keystore.jceks);
keystore.setPassword("some_password"); // this password will also be used for accessing
the private key if not specified in the crypto-cms:decrypt endpoint

DefaultKeyTransRecipientInfo recipient1 = new DefaultKeyTransRecipientInfo();
recipient1.setCertificateAlias("rsa"); // alias of the public key used for the encryption
recipient1.setKeyStoreParameters(keystore);

simpleReg.put("keyStoreParameters", keystore); // register keystore in the registry
simpleReg.put("recipient1", recipient1); // register recipient info in the registry

from("direct:start")
   .to("crypto-cms:encrypt://testencrpyt?
toBase64=true&recipient=#recipient1&contentEncryptionAlgorithm=DESede/CBC/PKCS5Padd
ing&secretKeyLength=128")
   .to("crypto-cms:decrypt://testdecrypt?
fromBase64=true&keyStoreParameters=#keyStoreParameters")
   .to("mock:result");
```

Spring XML 中的基本示例

```
  <keyStoreParameters xmlns="http://camel.apache.org/schema/spring"
      id="keyStoreParameters1" resource="./keystore/keystore.jceks"
      password="some_password" type="JCEKS" />
   <bean id="recipient1"
      class="org.apache.camel.component.crypto.cms.crypt.DefaultKeyTransRecipientInfo">
      <property name="keyStoreParameters" ref="keyStoreParameters1" />
      <property name="certificateAlias" value="rsa" />
   </bean>
...
   <route>
      <from uri="direct:start" />
      <to uri="crypto-cms:encrypt://testencrpyt?
toBase64=true&amp;recipient=#recipient1&amp;contentEncryptionAlgorithm=DESede/CBC/PKCS5Pad
ding&amp;secretKeyLength=128" />
      <to uri="crypto-cms:decrypt://testdecrypt?
fromBase64=true&amp;keyStoreParameters=#keyStoreParameters1" />
       <to uri="mock:result" />
   </route>
```

Java DSL 中的两个接收者

```
import org.apache.camel.util.jsse.KeyStoreParameters;
import org.apache.camel.component.crypto.cms.crypt.DefaultKeyTransRecipientInfo;
...
KeyStoreParameters keystore  = new KeyStoreParameters();
keystore.setType("JCEKS");
keystore.setResource("keystore/keystore.jceks);
keystore.setPassword("some_password"); // this password will also be used for accessing
the private key if not specified in the crypto-cms:decrypt endpoint

DefaultKeyTransRecipientInfo recipient1 = new DefaultKeyTransRecipientInfo();
recipient1.setCertificateAlias("rsa"); // alias of the public key used for the encryption
recipient1.setKeyStoreParameters(keystore);

DefaultKeyTransRecipientInfo recipient2 = new DefaultKeyTransRecipientInfo();
recipient2.setCertificateAlias("dsa");
recipient2.setKeyStoreParameters(keystore);

simpleReg.put("keyStoreParameters", keystore); // register keystore in the registry
simpleReg.put("recipient1", recipient1); // register recipient info in the registry

from("direct:start")
   .to("crypto-cms:encrypt://testencrpyt?
toBase64=true&recipient=#recipient1&recipient=#recipient2&contentEncryptionAlgorithm=DE
Sede/CBC/PKCS5Padding&secretKeyLength=128")
   //the decryptor will automatically choose one of the two private keys depending which one
is in the decryptor keystore
   .to("crypto-cms:decrypt://testdecrypt?
fromBase64=true&keyStoreParameters=#keyStoreParameters")
   .to("mock:result");
```

Spring XML 中的两个接收者

```
  <keyStoreParameters xmlns="http://camel.apache.org/schema/spring"
      id="keyStoreParameters1" resource="./keystore/keystore.jceks"
      password="some_password" type="JCEKS" />
   <bean id="recipient1"
      class="org.apache.camel.component.crypto.cms.crypt.DefaultKeyTransRecipientInfo">
      <property name="keyStoreParameters" ref="keyStoreParameters1" />
      <property name="certificateAlias" value="rsa" />
   </bean>
   <bean id="recipient2"
      class="org.apache.camel.component.crypto.cms.crypt.DefaultKeyTransRecipientInfo">
      <property name="keyStoreParameters" ref="keyStoreParameters1" />
      <property name="certificateAlias" value="dsa" />
   </bean>
...
   <route>
      <from uri="direct:start" />
      <to uri="crypto-cms:encrypt://testencrpyt?
toBase64=true&amp;recipient=#recipient1&amp;recipient=#recipient2&amp;contentEncryptionAlgorithm
```

```
=DESede/CBC/PKCS5Padding&amp;secretKeyLength=128" />
        <!-- the decryptor will automatically choose one of the two private keys depending which
one is in the decryptor keystore -->
        <to uri="crypto-cms:decrypt://testdecrypt?
fromBase64=true&amp;keyStoreParameters=#keyStoreParameters1" />
         <to uri="mock:result" />
    </route>
```

75.4. 签名的数据

请注意，一个 crypto-cms:sign 端点通常在一个路由中定义，且免费的 crypto-cms:verify 在另一个路由后出现。

以下示例演示了如何创建 Signed Data 消息，以及如何验证 Signed Data 消息。

Java DSL 中的基本示例

```
import org.apache.camel.util.jsse.KeyStoreParameters;
import org.apache.camel.component.crypto.cms.sig.DefaultSignerInfo;
...
KeyStoreParameters keystore  = new KeyStoreParameters();
keystore.setType("JCEKS");
keystore.setResource("keystore/keystore.jceks);
keystore.setPassword("some_password"); // this password will also be used for accessing
the private key if not specified in the signerInfo1 bean

//Signer Information, by default the following signed attributes are included: contentType,
signingTime, messageDigest, and cmsAlgorithmProtect; by default no unsigned attribute is
included.
// If you want to add your own signed attributes or unsigned attributes, see methods
DefaultSignerInfo.setSignedAttributeGenerator and
DefaultSignerInfo.setUnsignedAttributeGenerator.
DefaultSignerInfo signerInfo1 = new DefaultSignerInfo();
signerInfo1.setIncludeCertificates(true); // if set to true then the certificate chain of the private
key will be added to the Signed Data object
signerInfo1.setSignatureAlgorithm("SHA256withRSA"); // signature algorithm; attention, the
signature algorithm must fit to the signer private key.
signerInfo1.setPrivateKeyAlias("rsa"); // alias of the private key used for the signing
signerInfo1.setPassword("private_key_pw".toCharArray()); // optional parameter, if not set
then the password of the KeyStoreParameters will be used for accessing the private key
signerInfo1.setKeyStoreParameters(keystore);

simpleReg.put("keyStoreParameters", keystore); //register keystore in the registry
simpleReg.put("signer1", signerInfo1); //register signer info in the registry

from("direct:start")
   .to("crypto-cms:sign://testsign?signer=#signer1&includeContent=true&toBase64=true")
```

```
    .to("crypto-cms:verify://testverify?
keyStoreParameters=#keyStoreParameters&fromBase64=true"")
    .to("mock:result");
```

Spring XML 中的基本示例

```
  <keyStoreParameters xmlns="http://camel.apache.org/schema/spring"
      id="keyStoreParameters1" resource="./keystore/keystore.jceks"
      password="some_password" type="JCEKS" />
   <bean id="signer1"
      class="org.apache.camel.component.crypto.cms.sig.DefaultSignerInfo">
      <property name="keyStoreParameters" ref="keyStoreParameters1" />
      <property name="privateKeyAlias" value="rsa" />
      <property name="signatureAlgorithm" value="SHA256withRSA" />
      <property name="includeCertificates" value="true" />
      <!-- optional parameter 'password', if not set then the password of the
KeyStoreParameters will be used for accessing the private key -->
      <property name="password" value="private_key_pw" />
   </bean>
...
   <route>
      <from uri="direct:start" />
      <to uri="crypto-cms:sign://testsign?
signer=#signer1&amp;includeContent=true&amp;toBase64=true" />
      <to uri="crypto-cms:verify://testverify?
keyStoreParameters=#keyStoreParameters1&amp;fromBase64=true" />
      <to uri="mock:result" />
   </route>
```

Java DSL 中有两个签名者示例

```
import org.apache.camel.util.jsse.KeyStoreParameters;
import org.apache.camel.component.crypto.cms.sig.DefaultSignerInfo;
...
KeyStoreParameters keystore  = new KeyStoreParameters();
keystore.setType("JCEKS");
keystore.setResource("keystore/keystore.jceks);
keystore.setPassword("some_password"); // this password will also be used for accessing
the private key if not specified in the signerInfo1 bean

//Signer Information, by default the following signed attributes are included: contentType,
signingTime, messageDigest, and cmsAlgorithmProtect; by default no unsigned attribute is
included.
// If you want to add your own signed attributes or unsigned attributes, see methods
DefaultSignerInfo.setSignedAttributeGenerator and
DefaultSignerInfo.setUnsignedAttributeGenerator.
DefaultSignerInfo signerInfo1 = new DefaultSignerInfo();
signerInfo1.setIncludeCertificates(true); // if set to true then the certificate chain of the private
key will be added to the Signed Data object
signerInfo1.setSignatureAlgorithm("SHA256withRSA"); // signature algorithm; attention, the
signature algorithm must fit to the signer private key.
```

```
signerInfo1.setPrivateKeyAlias("rsa"); // alias of the private key used for the signing
signerInfo1.setPassword("private_key_pw".toCharArray()); // optional parameter, if not set
then the password of the KeyStoreParameters will be used for accessing the private key
signerInfo1.setKeyStoreParameters(keystore);

DefaultSignerInfo signerInfo2 = new DefaultSignerInfo();
signerInfo2.setIncludeCertificates(true);
signerInfo2.setSignatureAlgorithm("SHA256withDSA");
signerInfo2.setPrivateKeyAlias("dsa");
signerInfo2.setKeyStoreParameters(keystore);


simpleReg.put("keyStoreParameters", keystore); //register keystore in the registry
simpleReg.put("signer1", signerInfo1); //register signer info in the registry
simpleReg.put("signer2", signerInfo2); //register signer info in the registry

from("direct:start")
   .to("crypto-cms:sign://testsign?signer=#signer1&signer=#signer2&includeContent=true")
   .to("crypto-cms:verify://testverify?keyStoreParameters=#keyStoreParameters")
   .to("mock:result");
```

在 Spring XML 中使用两个 Signers 的示例

```
  <keyStoreParameters xmlns="http://camel.apache.org/schema/spring"
      id="keyStoreParameters1" resource="./keystore/keystore.jceks"
      password="some_password" type="JCEKS" />
   <bean id="signer1"
      class="org.apache.camel.component.crypto.cms.sig.DefaultSignerInfo">
      <property name="keyStoreParameters" ref="keyStoreParameters1" />
      <property name="privateKeyAlias" value="rsa" />
      <property name="signatureAlgorithm" value="SHA256withRSA" />
      <property name="includeCertificates" value="true" />
      <!-- optional parameter 'password', if not set then the password of the
KeyStoreParameters will be used for accessing the private key -->
      <property name="password" value="private_key_pw" />
   </bean>
   <bean id="signer2"
      class="org.apache.camel.component.crypto.cms.sig.DefaultSignerInfo">
      <property name="keyStoreParameters" ref="keyStoreParameters1" />
      <property name="privateKeyAlias" value="dsa" />
      <property name="signatureAlgorithm" value="SHA256withDSA" />
      <!-- optional parameter 'password', if not set then the password of the
KeyStoreParameters will be used for accessing the private key -->
      <property name="password" value="private_key_pw2" />
   </bean>
...
   <route>
      <from uri="direct:start" />
      <to uri="crypto-cms:sign://testsign?
signer=#signer1&amp;signer=#signer2&amp;includeContent=true" />
      <to uri="crypto-cms:verify://testverify?keyStoreParameters=#keyStoreParameters1" />
      <to uri="mock:result" />
   </route>
```

Java DSL 中的分离签名示例

```
import org.apache.camel.util.jsse.KeyStoreParameters;
import org.apache.camel.component.crypto.cms.sig.DefaultSignerInfo;
...
KeyStoreParameters keystore  = new KeyStoreParameters();
keystore.setType("JCEKS");
keystore.setResource("keystore/keystore.jceks);
keystore.setPassword("some_password"); // this password will also be used for accessing
the private key if not specified in the signerInfo1 bean

//Signer Information, by default the following signed attributes are included: contentType,
signingTime, messageDigest, and cmsAlgorithmProtect; by default no unsigned attribute is
included.
// If you want to add your own signed attributes or unsigned attributes, see methods
DefaultSignerInfo.setSignedAttributeGenerator and
DefaultSignerInfo.setUnsignedAttributeGenerator.
DefaultSignerInfo signerInfo1 = new DefaultSignerInfo();
signerInfo1.setIncludeCertificates(true); // if set to true then the certificate chain of the private
key will be added to the Signed Data object
signerInfo1.setSignatureAlgorithm("SHA256withRSA"); // signature algorithm; attention, the
signature algorithm must fit to the signer private key.
signerInfo1.setPrivateKeyAlias("rsa"); // alias of the private key used for the signing
signerInfo1.setPassword("private_key_pw".toCharArray()); // optional parameter, if not set
then the password of the KeyStoreParameters will be used for accessing the private key
signerInfo1.setKeyStoreParameters(keystore);

simpleReg.put("keyStoreParameters", keystore); //register keystore in the registry
simpleReg.put("signer1", signerInfo1); //register signer info in the registry

from("direct:start")
    //with the option includeContent=false the SignedData object without the signed text will be
written into the header "CamelCryptoCmsSignedData"
   .to("crypto-cms:sign://testsign?signer=#signer1&includeContent=false&toBase64=true")
   //the verifier reads the Signed Data object form the header CamelCryptoCmsSignedData
and assumes that the signed content is in the message body
   .to("crypto-cms:verify://testverify?
keyStoreParameters=#keyStoreParameters&signedDataHeaderBase64=true")
   .to("mock:result");
```

Spring XML 中的分离签名示例

```
  <keyStoreParameters xmlns="http://camel.apache.org/schema/spring"
      id="keyStoreParameters1" resource="./keystore/keystore.jceks"
      password="some_password" type="JCEKS" />
   <bean id="signer1"
      class="org.apache.camel.component.crypto.cms.sig.DefaultSignerInfo">
      <property name="keyStoreParameters" ref="keyStoreParameters1" />
      <property name="privateKeyAlias" value="rsa" />
      <property name="signatureAlgorithm" value="SHA256withRSA" />
      <property name="includeCertificates" value="true" />
```

```
        <!-- optional parameter 'password', if not set then the password of the
KeyStoreParameters will be used for accessing the private key -->
        <property name="password" value="private_key_pw" />
    </bean>
...
    <route>
        <from uri="direct:start" />
        <!-- with the option includeContent=false the SignedData object without the signed text
will be written into the header "CamelCryptoCmsSignedData" -->
        <to uri="crypto-cms:sign://testsign?
signer=#signer1&amp;includeContent=false&amp;toBase64=true" />
        <!-- the verifier reads the Signed Data object form the header
CamelCryptoCmsSignedData and assumes that the signed content is in the message body -->
        <to uri="crypto-cms:verify://testverify?
keyStoreParameters=#keyStoreParameters1&amp;signedDataHeaderBase64=true" />
        <to uri="mock:result" />
    </route>
```

第 76 章 CRYPTO (JAVA CRYPTOGRAPHIC EXTENSION) DATAFORMAT

从 Camel 版本 2.3 开始提供

Crypto Data Format 将 Java Cryptographic Extension 集成到 Camel 中，允许使用 Camel 熟悉的 marshall 和 unmarshal 格式机制来简单而灵活的加密和解密信息。它假定要对 cyphertext 和 unmarshalling 进行加密，以意味着解密回原始的纯文本。此数据格式仅实施对称（共享密钥）加密和去除。

76.1. CRYPTODATAFORMAT OPTIONS

Crypto (Java Cryptographic Extension) dataformat 支持 10 个选项，如下所列。

Name	默认值	Java 类型	描述
algorithm	**DES/CBC/PKCS5Padding**	**字符串**	指示要使用的加密算法的 JCE 算法名称。默认为 DES/CBC/PKCS5Padding。
cryptoProvider		**字符串**	应使用的 JCE Security Provider 的名称。
keyRef		**字符串**	是指要从寄存器中查询的 secret 密钥。
initVectorRef		**字符串**	指的是包含 Initialization Vector 的字节数组，用于初始化 Cipher。
algorithmParameterRef		**字符串**	用于初始化 Cipher 的 JCE AlgorithmParameterSpec。将指定名称用作 java.security.spec.AlgorithmParameterSpec 类型来查找类型。
bufferSize		**整数**	签名进程中使用的缓冲区的大小。
macAlgorithm	**HmacSHA1**	**字符串**	指示消息身份验证算法的 JCE 算法名称。
shouldAppendHMAC	**false**	**布尔值**	表示消息身份验证代码应计算并附加到加密的数据的标志。
inline	**false**	**布尔值**	表示配置的 IV 应该内联到加密的数据流的标志。默认为 false。

Name	默认值	Java 类型	描述
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

76.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 33 选项，如下所列。

Name	描述	默认值	类型
camel.component.crypto.configuration.algorithm	设置用于签名者的 Algorithm 的 JCE 名称。	SHA1WithDSA	字符串
camel.component.crypto.configuration.alias	设置用于查询 KeyStore 的别名，并链接 java.security.cert.Certificate 证书以签名和验证交换。这个值可以通过消息标头 org.apache.camel.component.crypto.DigitalSignatureConstants #KEYSTORE_ALIAS 在运行时提供		字符串
camel.component.crypto.configuration.buffer-size	设置在 Exchange payload 数据中用于读取的缓冲区的大小。	2048	整数
camel.component.crypto.configuration.certificate	设置应在交换中根据其有效负载验证签名的证书。		证书
camel.component.crypto.configuration.certificate-name	为 registry 中可以 fond 的 PrivateKey 设置引用名称。		字符串
camel.component.crypto.configuration.clear-headers	决定在签名和验证后是否清除 Signature 特定标头。默认为 true，并且只应在您的极端考虑，因为密钥和密码等重要私有信息（如果未设置）可能会转义。	true	布尔值
camel.component.crypto.configuration.crypto-operation	在 endpoint uri .g. crypto:sign set sign as the operation 中的 crypto scheme 后设置 Crypto 操作。		CryptoOperation

Name	描述	默认值	类型
camel.component.crypto.configuration.key-store-parameters	设置 KeyStore，可以包含密钥和 Certficates，用于根据给定的 KeyStoreParameters 签名和验证交换。KeyStore 通常与别名一起使用，可以是 Route 定义中提供的，或者通过消息标头 CamelSignatureKeyStoreAlias 动态使用。如果没有提供别名，且在 Keystore 中只有一个条目，则会使用这个单个条目。		KeyStoreParameters
camel.component.crypto.configuration.keystore	设置 KeyStore，可以包含用于签名和验证交换的密钥和 Certficates。KeyStore 通常与别名一起使用，可以是 Route 定义中提供的，或者通过消息标头 CamelSignatureKeyStoreAlias 动态使用。如果没有提供别名，且在 Keystore 中只有一个条目，则会使用这个单个条目。		KeyStore
camel.component.crypto.configuration.keystore-name	为 registry 中可以 fond 的密钥存储设置引用名称。		字符串
camel.component.crypto.configuration.name	此操作的逻辑名称。		字符串
camel.component.crypto.configuration.password	设置用于访问 KeyStore 中别名的 PrivateKey 的密码。		character[]
camel.component.crypto.configuration.private-key	设置用于签署交换的 PrivateKey		PrivateKey
camel.component.crypto.configuration.private-key-name	为 registry 中可以 fond 的 PrivateKey 设置引用名称。		字符串
camel.component.crypto.configuration.provider	设置提供配置的签名算法的安全供应商的 ID。		字符串
camel.component.crypto.configuration.public-key	设置应在交换中验证签名的 PublicKey。		PublicKey

Name	描述	默认值	类型
camel.component.crypto.configuration.public-key-name	上下文更改时应解析的引用		字符串
camel.component.crypto.configuration.secure-random	设置用于初始化签名服务的 SecureRandom		SecureRandom
camel.component.crypto.configuration.secure-random-name	为 registry 中可以 fond 的 SecureRandom 设置引用名称。		字符串
camel.component.crypto.configuration.signature-header-name	设置用于存储 base64 编码签名的消息标头的名称。默认为 'CamelDigitalSignature'		字符串
camel.component.crypto.enabled	启用加密组件	true	布尔值
camel.component.crypto.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.dataformat.crypto.algorithm	指示要使用的加密算法的 JCE 算法名称。默认为 DES/CBC/PKCS5Padding。	DES/CBC/PKCS5Padding	字符串
camel.dataformat.crypto.algorithm-parameter-ref	用于初始化 Cipher 的 JCE AlgorithmParameterSpec。将指定名称用作 java.security.spec.AlgorithmParameterSpec 类型来查找类型。		字符串
camel.dataformat.crypto.buffersize	签名进程中使用的缓冲区的大小。		整数
camel.dataformat.crypto.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.crypto.crypto-provider	应使用的 JCE Security Provider 的名称。		字符串

Name	描述	默认值	类型
camel.dataformat.crypto.enabled	启用加密数据	true	布尔值
camel.dataformat.crypto.init-vector-ref	指的是包含 Initialization Vector 的字节数组，用于初始化 Cipher。		字符串
camel.dataformat.crypto.inline	表示配置的 IV 应该内联到加密的数据流的标志。默认为 false。	false	布尔值
camel.dataformat.crypto.key-ref	是指要从寄存器中查询的 secret 密钥。		字符串
camel.dataformat.crypto.mac-algorithm	指示消息身份验证算法的 JCE 算法名称。	HmacSHA1	字符串
camel.dataformat.crypto.should-append-h-m-a-c	表示消息身份验证代码应计算并附加到加密的数据的标志。	false	布尔值

ND

76.3. 基本用法

最基本的是加密/解密交换都是共享的 secret 密钥所必需的。如果使用此密钥配置了一个或多个 Crypto 数据格式实例，则格式可用于加密一个路由（或一部分）并在另一个路由中解密的有效负载。例如，使用 Java DSL，如下所示：

```
KeyGenerator generator = KeyGenerator.getInstance("DES");

CryptoDataFormat cryptoFormat = new CryptoDataFormat("DES", generator.generateKey());

from("direct:basic-encryption")
    .marshal(cryptoFormat)
    .to("mock:encrypted")
    .unmarshal(cryptoFormat)
    .to("mock:unencrypted");
```

在 Spring 中，dataformat 被首先配置，然后在路由中使用

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <dataFormats>
```

```
    <crypto id="basic" algorithm="DES" keyRef="desKey" />
  </dataFormats>
    ...
  <route>
    <from uri="direct:basic-encryption" />
    <marshal ref="basic" />
    <to uri="mock:encrypted" />
    <unmarshal ref="basic" />
    <to uri="mock:unencrypted" />
  </route>
</camelContext>
```

76.4. 指定加密算法

更改算法是提供 JCE 算法名称非常重要。如果更改了算法，则需要使用兼容密钥。

```
KeyGenerator generator = KeyGenerator.getInstance("DES");

CryptoDataFormat cryptoFormat = new CryptoDataFormat("DES", generator.generateKey());
cryptoFormat.setShouldAppendHMAC(true);
cryptoFormat.setMacAlgorithm("HmacMD5");

from("direct:hmac-algorithm")
    .marshal(cryptoFormat)
    .to("mock:encrypted")
    .unmarshal(cryptoFormat)
    .to("mock:unencrypted");
```

Java 7 中可用算法列表可以通过 Java Cryptography Architecture Standard Algorithm Name 文档获得。

76.5. 指定初始化向量

有些加密算法（特别是块算法）需要通过初始块（称为初始化向量）进行配置。在 JCE 中，当初始化 Cipher 时，它作为 AlgorithmParameterSpec 传递。要将此类向量与 CryptoDataFormat 搭配使用，您可以使用包含所需数据的 byte[] 来配置它，例如：

```
KeyGenerator generator = KeyGenerator.getInstance("DES");
byte[] initializationVector = new byte[] {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07};

CryptoDataFormat cryptoFormat = new CryptoDataFormat("DES/CBC/PKCS5Padding",
generator.generateKey());
cryptoFormat.setInitializationVector(initializationVector);

from("direct:init-vector")
    .marshal(cryptoFormat)
```

```
    .to("mock:encrypted")
    .unmarshal(cryptoFormat)
    .to("mock:unencrypted");
```

或使用 spring，为 byte[]提供引用

```
<crypto id="initvector" algorithm="DES/CBC/PKCS5Padding" keyRef="desKey"
initVectorRef="initializationVector" />
```

加密和解密阶段都需要同样的向量。由于不需要保留 IV a secret， DataFormat 允许将其内联到加密的数据中，然后读取在解密阶段来初始化 Cipher。若要内联 IV 设置 /oinline 标志。

```
KeyGenerator generator = KeyGenerator.getInstance("DES");
byte[] initializationVector = new byte[] {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07};
SecretKey key = generator.generateKey();

CryptoDataFormat cryptoFormat = new CryptoDataFormat("DES/CBC/PKCS5Padding", key);
cryptoFormat.setInitializationVector(initializationVector);
cryptoFormat.setShouldInlineInitializationVector(true);
CryptoDataFormat decryptFormat = new CryptoDataFormat("DES/CBC/PKCS5Padding", key);
decryptFormat.setShouldInlineInitializationVector(true);

from("direct:inline")
    .marshal(cryptoFormat)
    .to("mock:encrypted")
    .unmarshal(decryptFormat)
    .to("mock:unencrypted");
```

或使用 spring。

```
<crypto id="inline" algorithm="DES/CBC/PKCS5Padding" keyRef="desKey"
initVectorRef="initializationVector"
  inline="true" />
<crypto id="inline-decrypt" algorithm="DES/CBC/PKCS5Padding" keyRef="desKey" inline="true" />
```

有关使用初始化向量的更多信息，请参阅

- http://en.wikipedia.org/wiki/Initialization_vector
- http://www.herongyang.com/Cryptography/

- http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

76.6. 散列消息身份验证代码(HMAC)

为了避免对加密数据的攻击，而传输 CryptoDataFormat 也可以根据可配置的 MAC 算法计算加密交换内容的 Message Authentication Code。在加密后，计算的 HMAC 会附加到流中。它与解密阶段的流分开。根据传输的版本重新计算并验证的 MAC 以确保在传输中未被篡改。如需有关消息身份验证代码的更多信息，请参阅 http://en.wikipedia.org/wiki/HMAC

```
KeyGenerator generator = KeyGenerator.getInstance("DES");

CryptoDataFormat cryptoFormat = new CryptoDataFormat("DES", generator.generateKey());
cryptoFormat.setShouldAppendHMAC(true);

from("direct:hmac")
    .marshal(cryptoFormat)
    .to("mock:encrypted")
    .unmarshal(cryptoFormat)
    .to("mock:unencrypted");
```

或使用 spring。

```
<crypto id="hmac" algorithm="DES" keyRef="desKey" shouldAppendHMAC="true" />
```

默认情况下，HMAC 使用 HmacSHA1 mac 算法计算，但可以通过提供不同的算法名称来轻松地更改。请参阅此处了解如何通过配置的安全供应商检查哪些算法可用

```
KeyGenerator generator = KeyGenerator.getInstance("DES");

CryptoDataFormat cryptoFormat = new CryptoDataFormat("DES", generator.generateKey());
cryptoFormat.setShouldAppendHMAC(true);
cryptoFormat.setMacAlgorithm("HmacMD5");

from("direct:hmac-algorithm")
    .marshal(cryptoFormat)
    .to("mock:encrypted")
    .unmarshal(cryptoFormat)
    .to("mock:unencrypted");
```

或使用 spring。

```
<crypto id="hmac-algorithm" algorithm="DES" keyRef="desKey" macAlgorithm="HmacMD5"
shouldAppendHMAC="true" />
```

76.7. 动态提供密钥

当使用 Recipient 列表或类似的 EIP 时，交换的接收者可能会动态变化。在所有接收者中使用相同的密钥可能并不可行或必要。在每次交换基础上动态指定密钥会很有用。然后，可以在通过数据格式处理前，使用其目标接收者的密钥动态增强交换。为了便于实现这个 DataFormat 允许通过以下消息标头动态提供密钥

- CryptoDataFormat.KEY "CamelCryptoKey"

```
CryptoDataFormat cryptoFormat = new CryptoDataFormat("DES", null);
/**
 * Note: the header containing the key should be cleared after
 * marshalling to stop it from leaking by accident and
 * potentially being compromised. The processor version below is
 * arguably better as the key is left in the header when you use
 * the DSL leaks the fact that camel encryption was used.
 */
from("direct:key-in-header-encrypt")
    .marshal(cryptoFormat)
    .removeHeader(CryptoDataFormat.KEY)
    .to("mock:encrypted");

from("direct:key-in-header-decrypt").unmarshal(cryptoFormat).process(new Processor() {
    public void process(Exchange exchange) throws Exception {
        exchange.getIn().getHeaders().remove(CryptoDataFormat.KEY);
        exchange.getOut().copyFrom(exchange.getIn());
    }
}).to("mock:unencrypted");
```

或使用 spring。

```
<crypto id="nokey" algorithm="DES" />
```

76.8. 依赖项

要在 camel 路由中使用 Crypto dataformat，您需要将以下依赖项添加到 pom 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-crypto</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

76.9. 另请参阅

- *数据格式*
- *crypto (Digital Signatures)*
- *http://www.bouncycastle.org/java.html*

第 77 章 CSV DATAFORMAT

从 Camel 版本 1.3 开始提供

CSV 数据格式使用 Apache Commons CSV 处理 CSV 有效负载(Comma Sepaated Values)，如由 Excel 导出/导入的值。

77.1. 选项

CSV 数据支持 28 个选项，如下所列。

Name	默认值	Java 类型	描述
formatRef		字符串	要使用的引用格式，它将使用其它格式选项进行更新，默认值为 CSVFormat.DEFAULT
formatName		字符串	要使用的格式的名称，默认值为 CSVFormat.DEFAULT
commentMarkerDisabled	false	布尔值	禁用引用格式的注释标记。
commentMarker		字符串	设置参考格式的注释标记。
delimiter		字符串	设置要使用的分隔符。默认值为 ,(comma)
escapeDisabled	false	布尔值	使用 转义字符禁用
Escape		字符串	设置要使用的转义字符
headerDisabled	false	布尔值	用于禁用标头
header		list	配置 CSV 标头
allowMissingColumnNames	false	布尔值	是否允许缺少的列名称。
ignoreEmptyLines	false	布尔值	是否忽略空行。
ignoreSurroundingSpaces	false	布尔值	是否忽略周围的空格

Name	默认值	Java 类型	描述
nullStringDisabled	false	布尔值	用于禁用 null 字符串
nullString		字符串	设置 null 字符串
quoteDisabled	false	布尔值	用于禁用引号
quote		字符串	设置默认情况下的引号
recordSeparatorDisabled		字符串	用于禁用记录分隔符
recordSeparator		字符串	设置记录分隔符（也称为新行），默认为新行字符(CRLF)
skipHeaderRecord	false	布尔值	是否跳过输出中的标头记录
quoteMode		字符串	设置 quote 模式
ignoreHeaderCase	false	布尔值	设置在访问标头名称时是否忽略问题单。
trim	false	布尔值	设置是否修剪前导和尾随空白。
trailingDelimiter	false	布尔值	设置是否添加尾随分隔符。
lazyLoad	false	布尔值	unmarshalling 是否应该生成一个迭代器，该实用程序可即时读取行，或者是否必须读取所有行。
useMaps	false	布尔值	unmarshalling 是否应该为行值而不是列表生成映射(HashMap)。它需要有标头（定义或收集）。
useOrderedMaps	false	布尔值	unmarshalling 是否应该为行值而不是列表生成有序映射(LinkedHashMap)。它需要有标头（定义或收集）。
recordConverterRef		字符串	是指从 registry 中查询的自定义 CsvRecordConverter。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

77.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 29 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.csv.allow-missing-column-names	是否允许缺少的列名称。	false	布尔值
camel.dataformat.csv.comment-marker	设置参考格式的注释标记。		字符串
camel.dataformat.csv.comment-marker-disabled	禁用引用格式的注释标记。	false	布尔值
camel.dataformat.csv.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.csv.delimiter	设置要使用的分隔符。默认值为 ,(comma)		字符串
camel.dataformat.csv.enabled	enable csv dataformat	true	布尔值
camel.dataformat.csv.escape	设置要使用的转义字符		字符串
camel.dataformat.csv.escape-disabled	使用 转义字符禁用	false	布尔值
camel.dataformat.csv.format-name	要使用的格式的名称，默认值为 CSVFormat.DEFAULT		字符串
camel.dataformat.csv.format-ref	要使用的引用格式，它将使用其它格式选项进行更新，默认值为 CSVFormat.DEFAULT		字符串
camel.dataformat.csv.header	配置 CSV 标头		list
camel.dataformat.csv.header-disabled	用于禁用标头	false	布尔值

Name	描述	默认值	类型
camel.dataformat.csv.ignore-empty-lines	是否忽略空行。	false	布尔值
camel.dataformat.csv.ignore-header-case	设置在访问标头名称时是否忽略问题单。	false	布尔值
camel.dataformat.csv.ignore-surrounding-spaces	是否忽略周围的空格	false	布尔值
camel.dataformat.csv.lazy-load	unmarshalling 是否应该生成一个迭代器，该实用程序可即时读取行，或者是否必须读取所有行。	false	布尔值
camel.dataformat.csv.null-string	设置 null 字符串		字符串
camel.dataformat.csv.null-string-disabled	用于禁用 null 字符串	false	布尔值
camel.dataformat.csv.quote	设置默认情况下的引号		字符串
camel.dataformat.csv.quote-disabled	用于禁用引号	false	布尔值
camel.dataformat.csv.quote-mode	设置 quote 模式		字符串
camel.dataformat.csv.record-converter-ref	是指从 registry 中查询的自定义 CsvRecordConverter。		字符串
camel.dataformat.csv.record-separator	设置记录分隔符（也称为新行），默认为新行字符 (CRLF)		字符串
camel.dataformat.csv.record-separator-disabled	用于禁用记录分隔符		字符串

Name	描述	默认值	类型
camel.dataformat.csv.skip-header-record	是否跳过输出中的标头记录	false	布尔值
camel.dataformat.csv.trailing-delimiter	设置是否添加尾随分隔符。	false	布尔值
camel.dataformat.csv.trim	设置是否修剪前导和尾随空白。	false	布尔值
camel.dataformat.csv.use-maps	unmarshalling 是否应该为行值而不是列表生成映射(HashMap)。它需要有标头（定义或收集）。	false	布尔值
camel.dataformat.csv.use-ordered-maps	unmarshalling 是否应该为行值而不是列表生成有序映射(LinkedHashMap)。它需要有标头（定义或收集）。	false	布尔值

ND

77.3. 对 CSV 合并映射

组件允许您将 Java 映射（或其他可在映射中转换的消息类型）放入 CSV 有效负载中。

考虑以下正文

```
Map<String, Object> body = new LinkedHashMap<>();
body.put("foo", "abc");
body.put("bar", 123);
```

以及此 Java 路由定义

```
from("direct:start")
    .marshal().csv()
    .to("mock:result");
```

或者此 XML 路由定义

```
<route>
```

```
    <from uri="direct:start" />
    <marshal>
        <csv />
    </marshal>
    <to uri="mock:result" />
</route>
```

然后，它将生成

```
abc,123
```

77.4. 将 CSV 消息写入 JAVA 列表中

unmarshalling 将 CSV 条形转换为带有 CSV 文件行的 Java 列表（包含带有所有字段值的另一个列表）。

例如：我们有一个 CSV 文件，其名称为个人、IQ 及其当前活动。

```
Jack Dalton, 115, mad at Averell
Joe Dalton, 105, calming Joe
William Dalton, 105, keeping Joe from killing Averell
Averell Dalton, 80, playing with Rantanplan
Lucky Luke, 120, capturing the Daltons
```

现在，我们可以使用 CSV 组件 unmarshal 这个文件：

```
from("file:src/test/resources/?fileName=daltons.csv&noop=true")
    .unmarshal().csv()
    .to("mock:daltons");
```

生成的消息将包含一个 List<List<String>>，如…

```
List<List<String>> data = (List<List<String>>) exchange.getIn().getBody();
for (List<String> line : data) {
    LOG.debug(String.format("%s has an IQ of %s and is currently %s", line.get(0), line.get(1),
line.get(2)));
}
```

77.5. 将 LIST<MAP> 到 CSV 的 MARSHALLING A LIST<MAP>

从 Camel 2.1 开始提供

如果您要将多个数据行放入 CSV 格式，您现在可以将消息 payload 存储为 List<Map<String, Object>> 对象，其中列表包含每行的 Map。

77.6. CSV 文件 POLLER，然后 UNMARSHALING

给定一个可以处理传入的 data... 的 bean

MyCsvHandler.java

```
// Some comments here
public void doHandleCsvData(List<List<String>> csvData)
{
    // do magic here
}
```

i. 然后您的路由类似如下

```
<route>
      <!-- poll every 10 seconds -->
      <from uri="file:///some/path/to/pickup/csvfiles?delete=true&amp;consumer.delay=10000" />
      <unmarshal><csv /></unmarshal>
      <to uri="bean:myCsvHandler?method=doHandleCsvData" />
</route>
```

77.7. 使用管道作为分隔符进行 MARSHALING

考虑以下正文

```
Map<String, Object> body = new LinkedHashMap<>();
body.put("foo", "abc");
body.put("bar", 123);
```

以及此 Java 路由定义

```
// Camel version < 2.15
CsvDataFormat oldCSV = new CsvDataFormat();
oldCSV.setDelimiter("|");
from("direct:start")
    .marshal(oldCSV)
    .to("mock:result")
```

```
// Camel version >= 2.15
from("direct:start")
    .marshal(new CsvDataFormat().setDelimiter(&#39;|&#39;))
    .to("mock:result")
```

或者此 XML 路由定义

```
<route>
  <from uri="direct:start" />
  <marshal>
    <csv delimiter="|" />
  </marshal>
  <to uri="mock:result" />
</route>
```

然后，它将生成

```
abc|123
```

在 XML # DSL 中使用 autogenColumns、configRef 和 strategyRef 属性

从 Camel 2.9.2 / 2.10 开始，为 Camel 2.15 提供并删除 Camel 2.15

您可以自定义 CSV 数据格式，以使用自己的 CSVConfig 和/或 CSVStrategy。另请注意，autogen Columns 选项 的默认值为 true。下例演示了此自定义。

```
<route>
  <from uri="direct:start" />
  <marshal>
    <!-- make use of a strategy other than the default one which is
'org.apache.commons.csv.CSVStrategy.DEFAULT_STRATEGY' -->
    <csv autogenColumns="false" delimiter="|" configRef="csvConfig" strategyRef="excelStrategy" />
  </marshal>
  <convertBodyTo type="java.lang.String" />
  <to uri="mock:result" />
</route>

<bean id="csvConfig" class="org.apache.commons.csv.writer.CSVConfig">
  <property name="fields">
    <list>
      <bean class="org.apache.commons.csv.writer.CSVField">
        <property name="name" value="orderId" />
      </bean>
      <bean class="org.apache.commons.csv.writer.CSVField">
        <property name="name" value="amount" />
```

```
      </bean>
    </list>
  </property>
</bean>

<bean id="excelStrategy"
class="org.springframework.beans.factory.config.FieldRetrievingFactoryBean">
  <property name="staticField" value="org.apache.commons.csv.CSVStrategy.EXCEL_STRATEGY"
/>
</bean>
```

77.8. 在 UNMARSHALING 时使用 SKIPFIRSTLINE 选项

从 Camel 2.10 开始，为 Camel 2.15 提供并删除

您可以指示 CSV 数据格式跳过包含 CSV 标头的第一行。使用 Spring/XML DSL:

```
<route>
  <from uri="direct:start" />
  <unmarshal>
    <csv skipFirstLine="true" />
  </unmarshal>
  <to uri="bean:myCsvHandler?method=doHandleCsv" />
</route>
```

或 Java DSL：

```
CsvDataFormat csv = new CsvDataFormat();
csv.setSkipFirstLine(true);

from("direct:start")
  .unmarshal(csv)
.to("bean:myCsvHandler?method=doHandleCsv");
```

77.9. 使用管道作为分隔符的 UNMARSHALING

使用 Spring/XML DSL:

```
<route>
  <from uri="direct:start" />
  <unmarshal>
    <csv delimiter="|" />
  </unmarshal>
  <to uri="bean:myCsvHandler?method=doHandleCsv" />
</route>
```

或 Java DSL：

```
CsvDataFormat csv = new CsvDataFormat();
CSVStrategy strategy = CSVStrategy.DEFAULT_STRATEGY;
strategy.setDelimiter('|');
csv.setStrategy(strategy);

from("direct:start")
  .unmarshal(csv)
  .to("bean:myCsvHandler?method=doHandleCsv");
```

```
CsvDataFormat csv = new CsvDataFormat();
csv.setDelimiter("|");

from("direct:start")
  .unmarshal(csv)
  .to("bean:myCsvHandler?method=doHandleCsv");
```

```
CsvDataFormat csv = new CsvDataFormat();
CSVConfig csvConfig = new CSVConfig();
csvConfig.setDelimiter(";");
csv.setConfig(csvConfig);

from("direct:start")
  .unmarshal(csv)
  .to("bean:myCsvHandler?method=doHandleCsv");
```

CSVConfig 的问题

这似乎是这样

```
CSVConfig csvConfig = new CSVConfig();
csvConfig.setDelimiter(';');
```

不工作。您必须将分隔符设置为字符串！

77.10. 依赖项

要在 Camel 路由中使用 CSV，您需要对 camel-csv 添加依赖项，该依赖项实施此数据格式。

如果使用 Maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-csv</artifactId>
  <version>x.x.x</version>
</dependency>
```

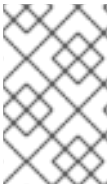
第 78 章 CXF

CXF 组件

cxf: 组件提供与 Apache CXF 集成，以连接到在 CXF 中托管的 JAX-WS 服务。

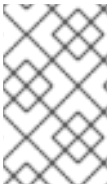
Maven 用户必须将此组件的 pom.xml 添加以下依赖项：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-cxf</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```



注意

如果要了解 CXF 依赖项，请参阅 WHICH-JARS 文本文件。



注意

在流模式中使用 CXF 时（请参阅 DataFormat 选项），然后读取 流缓存。

CAMEL ON EAP 部署

此组件由 EAP 上的 Camel (Wildfly Camel)框架支持，该框架在 Red Hat JBoss Enterprise Application Platform (JBoss EAP)容器上提供了简化的部署模型。

CXF 组件与也使用 Apache CXF 的 JBoss EAP webservices susbsystem 集成。如需更多信息，请参阅 JAX-WS。

注意

目前，EAP 子系统上的 Camel 不支持 CXF 或 Restlet 用户。但是，可以使用 CamelProxy 来模拟 CXF 消费者。

URI 格式

```
cxf:bean:cxfEndpoint[?options]
```

其中 cxfEndpoint 代表一个 Bean ID，它引用了 Spring bean registry 中的 bean。使用这个 URI 格式，大多数端点详情都在 bean 定义中指定。

```
cxf://someAddress[?options]
```

其中 someAddress 指定 CXF 端点的地址。使用这个 URI 格式，大多数端点详情都使用选项来指定。

对于上述任一样式，您可以按如下方式将选项附加到 URI：

```
cxf:bean:cxfEndpoint?wsdlURL=wsdl/hello_world.wsdl&dataFormat=PAYLOAD
```

选项

Name	必填	描述
wsdlURL	否	WSDL 的位置。WSDL 默认从端点地址获取。例如： **file://local/wsdl/hello.wsdl** 或 **wsdl/hello.wsdl**

serviceClass	是	SEI（服务端点接口）类的名称。此类可以有，但不需要 JSR181 注释。**从 2.0 开始，** 只有 POJO 模式需要这个选项。如果提供了 wsdlURL 选项，则 PAYLOAD 和 MESSAGE 模式不需要 serviceClass。当在没有 serviceClass 的情况下使用 wsdlURL 选项时，提供 serviceName 和 portName （用于 Spring 配置的endpointName）选项 **MUST**。 从 2.0 开始，可以使用 **\:** 表示法来引用 registry 中的 **serviceClass** 对象实例。 **建议引用的对象不能是一个代理 (Spring AOP Proxy is OK)，** 因为它依赖于 non Spring AOP Proxy 的 **Object.getClass （）.getName （）** 方法。 从 **2.8 开始，** 可以为 PAYLOAD 和 MESSAGE 模式省略 wsdlURL 和 serviceClass 选项。当忽略它们时，任意 XML 元素可以放在 CxfPayload 的正文(PAYLOAD 模式)中，以便于 CXF Dispatch Mode。 例如：**org.apache.camel.Hello**
serviceName	只有在 WSDL 中存在多个 **serviceName** 时	此服务实施的服务名称，它映射到 **wsdl:service@name**。例如： **{http://org.apache.camel}ServiceName**
endpointName	只有 **serviceName** 下有多个 **portName**，且 camel-cxf consumer 需要它，自 camel 2.2 起需要它	此服务实施的端口名称，它映射到 **wsdl:port@name**。例如： **{http://org.apache.camel}PortName**
dataFormat	否	CXF 端点支持的消息数据格式。可能的值有：**POJO （默认）**、**PAYLOAD**、**MESSAGE**。

relayHeaders	否	定义 CXF 端点中继路由中的标头。请参阅 "relayHeaders 选项的描述"一节。目前仅在 **dataFormat=POJO默认:true 示例,false**时才可用
嵌套	否	CXF 端点生成者调用的操作类型。可能的值有：**true**，**false**（**默认**）
wrappedStyle	否	**由于** 2.5.0 是 WSDL 样式，用于描述 SOAP 正文中如何表示参数。如果值为 **false**，则 CXF 会选择 Documentation-literal unwrapped 风格，如果值为 **true**，则 CXF 选择 document-literal 打包风格
setDefaultBus	否	**deprecated** ： 指定是否为这个端点使用默认的 CXF 总线。可能的值有：**true**，**false**（**默认**）这个选项已弃用。以后使用 Camel 2.16 **的默认Bus**。
defaultBus	否	**deprecated** ： 指定是否为这个端点使用默认的 CXF 总线。可能的值有：**true**，**false**（**默认**）这个选项已弃用。以后使用 Camel 2.16 **的默认Bus**。
bus	否	使用 \ Slack 表示法引用 registry 的总线对象，例如 **bus=\ problembusName**。所引用的对象必须是 **org.apache.cxf.Bus** 的实例。 默认情况下，使用 CXF Bus Factory 创建的默认总线。
cxfBinding	否	使用 **\#** 表示法引用来自 registry 的 CXF 绑定对象，例如：**cxfBinding=\"bindingName**。引用的对象必须是 **org.apache.camel.component.cxf.CxfBinding** 实例。
headerFilterStrategy	否	使用 \: 表示法引用来自 registry swig-wagon 的标头过滤器策略对象，例如：**headerFilterStrategy= \# strategyName**。所引用的对象必须是 **org.apache.camel.spi.HeaderFilterStrategy** 的实例。

loggingFeatureEnabled	否	在 2.3 中，这个选项启用 CXF Logging 功能，它将写入入站和出站 SOAP 消息进行日志记录。可能的值有：**true**，**false**（默认）
defaultOperationName	否	在 2.4 中，此选项设置由 CxfProducer 调用远程服务的 **CxfProducer** 使用的默认 **operationName**。例如： **defaultOperationName=greetMe**
defaultOperationNamespace	否	在 2.4 中，此选项设置 CxfProducer 使用的默认 operationNamespace，后者调用远程服务。例如： **defaultOperationNamespace=http://apache.org/hello_world_soap_http**
同步	否	在 2.5 中，这个选项可让 CXF 端点决定使用 sync 或 async API 进行底层工作。默认值为 **false**，这意味着 **camel-cxf** 端点会默认尝试使用 async API。
publishedEndpointUrl	否	2.5 中的新选项覆盖已发布的 WSDL 中出现的端点 URL，该端点使用服务地址 URL 和 **?wsdl** 访问。例如： **publishedEndpointUrl=http://example.com/service**

properties.propName	否	Camel 2.8： 允许您在端点 URI 中设置自定义 CXF 属性。例如，设置 **properties.mtom-enabled=true** 来启用 MTOM。为确保在开始调用时 CXF 不会切换线程，您可以设置 **properties.org.apache.cxf.interceptor.OneWayProcessorInterceptor.USE_ORIGINAL_THREAD=true**。
allowStreaming	否	2.8.2 中的新功能。这个选项控制 CXF 组件是否在 PAYLOAD 模式下运行时（请参阅以下），DOM 会将传入的消息解析为 DOM Elements，或者将有效负载保留为 **javax.xml.transform.Source** 对象，允许在某些情况下流。
skipFaultLogging	否	2.11 中的新功能。这个选项控制 PhaseInterceptorChain 是否跳过记录它捕获的 Fault。
cxfEndpointConfigurer	否	Camel 2.11 中的新功能。这个选项可以应用 **org.apache.camel.component.cxf.CxfEndpointConfigurer 的实现，它支持** 以编程方式配置 CXF 端点。自 Camel 2.15.0 **起，**用户可以通过实施 **CxfEndpointConfigurer** 的 configure{Server/Client} 方法来配置 CXF 服务器和客户端。
username	否	Camel 2.12.3 中的新选项用于为 CXF 客户端设置用户名的基本身份验证信息。
password	否	Camel 2.12.3 中的新选项用于为 CXF 客户端设置密码的基本身份验证信息。

continuationTimeout	否	Camel 2.14.0 中的新选项用于设置 CXF continuation 超时，当 CXF 服务器使用 Jetty 或 Servlet 传输时，默认可在 CxfConsumer 中使用。（在 Camel 2.14.0 之前，CxfConsumer 将 continuation 超时设置为 0，这意味着 continuation suspend 操作永远不会超时。） **默认** ： 30000**示例** ： continuation=80000

serviceName 和 portName 是 QNames，因此如果您为它们提供，请务必为其添加 {namespace} 前缀，如上例中所示。

DATAFORMATS 的描述

DataFormat	描述
POJO	POJO （纯 Java 对象）是目标服务器上调用的方法的 Java 参数。支持协议和逻辑 JAX-WS 处理程序。
PAYLOAD	**PAYLOAD** 是应用 CXF 端点消息配置后的消息有效负载(**soap:body**的内容)。仅支持协议 JAX-WS 处理程序。不支持逻辑 JAX-WS 处理程序。
MESSAGE	**MESSAGE** 是从传输层接收的原始消息。它不假设 touch 或 change Stream，如果您使用这种 DataFormat，则一些 CXF 拦截器会被删除，因此不支持 camel-cxf consumer 和 JAX-WS 处理程序后的任何 soap 标头。
CXF_MESSAGE	Camel 2.8.2 中的新功能，**CXF_MESSAGE** 允许通过将消息从传输层转换为原始 SOAP 消息来调用 CXF 拦截器的完整功能

您可以通过检索交换属性 CamelCXFDataFormat 来确定交换的数据格式模式。Exchange key 常量在 org.apache.camel.component.cxf.CxfConstants.DATA_FORMAT_PROPERTY 中定义。

使用 APACHE ARIES BLUEPRINT 配置 CXF 端点.

从 Camel 2.8 开始，支持对 CXF 端点使用 Aries 蓝图依赖项注入。模式与 Spring 模式非常相似，因此转换过程比较透明。

例如：

```
 <blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:cm="http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.0.0"
           xmlns:camel-cxf="http://camel.apache.org/schema/blueprint/cxf"
     xmlns:cxfcore="http://cxf.apache.org/blueprint/core"
           xsi:schemaLocation="http://www.osgi.org/xmlns/blueprint/v1.0.0
https://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd">

       <camel-cxf:cxfEndpoint id="routerEndpoint"
                    address="http://localhost:9001/router"
                    serviceClass="org.apache.servicemix.examples.cxf.HelloWorld">
         <camel-cxf:properties>
            <entry key="dataFormat" value="MESSAGE"/>
         </camel-cxf:properties>
       </camel-cxf:cxfEndpoint>

       <camel-cxf:cxfEndpoint id="serviceEndpoint"
    address="http://localhost:9000/SoapContext/SoapPort"
                    serviceClass="org.apache.servicemix.examples.cxf.HelloWorld">
     </camel-cxf:cxfEndpoint>

     <camelContext xmlns="http://camel.apache.org/schema/blueprint">
        <route>
           <from uri="routerEndpoint"/>
           <to uri="log:request"/>
        </route>
     </camelContext>

</blueprint>
```

目前，endpoint 元素是第一个支持的 CXF namespacehandler。

您还可以像 spring 中一样使用 bean 引用

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:cm="http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.0.0"
           xmlns:jaxws="http://cxf.apache.org/blueprint/jaxws"
           xmlns:cxf="http://cxf.apache.org/blueprint/core"
           xmlns:camel="http://camel.apache.org/schema/blueprint"
           xmlns:camelcxf="http://camel.apache.org/schema/blueprint/cxf"
           xsi:schemaLocation="
             http://www.osgi.org/xmlns/blueprint/v1.0.0
https://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd
             http://cxf.apache.org/blueprint/jaxws http://cxf.apache.org/schemas/blueprint/jaxws.xsd
             http://cxf.apache.org/blueprint/core http://cxf.apache.org/schemas/blueprint/core.xsd
             ">
```

```
    <camelcxf:cxfEndpoint id="reportIncident"
                address="/camel-example-cxf-blueprint/webservices/incident"
                wsdlURL="META-INF/wsdl/report_incident.wsdl"
                serviceClass="org.apache.camel.example.reportincident.ReportIncidentEndpoint">
    </camelcxf:cxfEndpoint>

    <bean id="reportIncidentRoutes"
class="org.apache.camel.example.reportincident.ReportIncidentRoutes" />

    <camelContext xmlns="http://camel.apache.org/schema/blueprint">
        <routeBuilder ref="reportIncidentRoutes"/>
    </camelContext>

</blueprint>
```

如何在 MESSAGE 模式中启用 CXF 的 LOGGINGOUTINTERCEPTOR

CXF 的 LoggingOutInterceptor 输出位于有线到日志记录系统(java.util.logging)的出站消息。由于 LoggingOutInterceptor 处于 PRE_STREAM 阶段（但 PRE_STREAM 阶段被删除为 MESSAGE 模式），因此您必须在 WRITE 阶段运行 LoggingOutInterceptor。以下是一个示例。

```
  <bean id="loggingOutInterceptor" class="org.apache.cxf.interceptor.LoggingOutInterceptor">
      <!--  it really should have been user-prestream but CXF does have such phase! -->
      <constructor-arg value="target/write"/>
  </bean>

<cxf:cxfEndpoint id="serviceEndpoint" address="http://localhost:9002/helloworld"
 serviceClass="org.apache.camel.component.cxf.HelloService">
 <cxf:outInterceptors>
    <ref bean="loggingOutInterceptor"/>
 </cxf:outInterceptors>
 <cxf:properties>
  <entry key="dataFormat" value="MESSAGE"/>
 </cxf:properties>
</cxf:cxfEndpoint>
```

RELAYHEADERS 选项的描述

有来自 JAXWS WSDL-first 开发者的 in-band 和 out-of-band on-the-wire 标头。

in-band 标头是标头，作为端点（如 SOAP 标头）的 WSDL 绑定合同的一部分。

带外 标头是通过线线序列化的标头，但不明确成为 WSDL 绑定合同的一部分。

标头中继/过滤是双向的。

当路由具有 CXF 端点并且开发人员需要有在线标头（如 SOAP 标头）时，将在另一个 JAXWS 端点消耗的路由中转发，然后将 relayHeaders 设置为 true，这是默认值。

仅在 POJO 模式中提供

relayHeaders=true 设置表示转发标头的意图。是否将给定标头中继的实际决定被委派给实现 MessageHeadersRelay 接口的可插拔实例。已查询 MessageHeadersRelay 的 concrete 实现，以决定是否需要转发标头。已有一个 SoapMessageHeadersRelay 的一个实现，它将自身绑定到已知的 SOAP 命名空间。目前，只过滤带外标头，当 relayHeaders=true 时始终转发带外的标头。如果有线上的标头，其命名空间在运行时未知，则使用 fall back DefaultMessageHeadersRelay，它只允许所有标头转发。

relayHeaders=false 设置表示所有标头（带外和带外）都会被丢弃。

您可以对自己的 MessageHeadersRelay 实现进行插件，或者在中继列表中添加其他内容。要覆盖预加载的中继实例，请确保您的 MessageHeadersRelay 实现服务与您要覆盖的命名空间相同。另请注意，覆盖中继必须将所有命名空间作为您要覆盖的命名空间提供服务，否则会抛出路由启动时的运行时异常，因为这会在命名空间中引入一种不确定性来转发实例映射。

```
<cxf:cxfEndpoint ...>
  <cxf:properties>
    <entry key="org.apache.camel.cxf.message.headers.relays">
      <list>
        <ref bean="customHeadersRelay"/>
      </list>
    </entry>
  </cxf:properties>
 </cxf:cxfEndpoint>
 <bean id="customHeadersRelay"
class="org.apache.camel.component.cxf.soap.headers.CustomHeadersRelay"/>
```

查看显示您如何转发/过滤标头的测试：

链接：https://svn.apache.org/repos/asf/camel/branches/camel-1.x/components/camel-cxf/src/test/java/org/apache/camel/component/cxf/soap/headers/CxfMessageHeadersRelayTest.java[https://svn.apache.org/repos/asf/camel/branches/camel-1.x/components/camel-cxf/src/test/java/org/apache/camel/component/cxf/soap/headers/CxfMessageHeadersRelayTest.java]

从版本 2.0 开始的更改

- 支持 POJO 和 PAYLOAD 模式。在 POJO 模式中，只有带外的消息标头可用于过滤，因为 CXF 从标头列表中删除。in-band 标头被合并到 POJO 模式的 MessageContentList 中。camel-cxf 组件会试图从 MessageContentList 中删除带外标头（如果需要过滤 in-band 标头），请使用 PAYLOAD 模式或插入(pretty directly) CXF 拦截器/JAXWS 处理程序到 CXF 端点。

- Message Header Relay 机制已合并到 CxfHeaderFilterStrategy 中。relayHeaders 选项、其语义和默认值保持不变，但它是 CxfHeaderFilterStrategy 的属性。以下是配置它的示例：

```
<bean id="dropAllMessageHeadersStrategy"
class="org.apache.camel.component.cxf.common.header.CxfHeaderFilterStrategy">

    <!-- Set relayHeaders to false to drop all SOAP headers -->
    <property name="relayHeaders" value="false"/>

</bean>
```

然后，您的端点可以引用 CxfHeaderFilterStrategy。

```
<route>
    <from uri="cxf:bean:routerNoRelayEndpoint?
headerFilterStrategy=#dropAllMessageHeadersStrategy"/>
    <to uri="cxf:bean:serviceNoRelayEndpoint?
headerFilterStrategy=#dropAllMessageHeadersStrategy"/>
</route>
```

- MessageHeadersRelay 接口稍有变化，并重命名为 MessageHeaderFilter。它是 CxfHeaderFilterStrategy 的属性。以下是配置用户定义的 Message Header Filters 的示例：

```
<bean id="customMessageFilterStrategy"
class="org.apache.camel.component.cxf.common.header.CxfHeaderFilterStrategy">
    <property name="messageHeaderFilters">
        <list>
            <!-- SoapMessageHeaderFilter is the built in filter. It can be removed by omitting it. --
>
            <bean
class="org.apache.camel.component.cxf.common.header.SoapMessageHeaderFilter"/>

            <!-- Add custom filter here -->
            <bean class="org.apache.camel.component.cxf.soap.headers.CustomHeaderFilter"/>
        </list>
    </property>
</bean>
```

- 除了 relayHeaders 外，还有可在 CxfHeaderFilterStrategy 中配置的新属性。

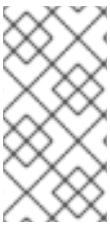
Name	描述	type	必需？	默认值
relayHeaders	所有消息标头都由 Message Header Filters 处理	**布尔值**	否	**true** (1.6.1 行为)
relayAllMessageHeaders	所有消息标头都会传播（无需由 Message Header Filters 处理）	**布尔值**	否	**false** (1.6.1 行为)
allowFilterNamespaceClash	在激活命名空间中处理重叠过滤器。如果值为 **true**，则最后一个值胜出。如果值为 **false**，它会抛出异常。	**布尔值**	否	**false** (1.6.1 行为)

使用 SPRING 配置 CXF 端点

您可以使用以下显示的 Spring 配置文件配置 CXF 端点，您也可以将端点嵌入到 camelContext 标签中。当您调用服务端点时，您可以将 operationName 和 operationNamespace 标头设置为明确您调用的操作的状态。

请注意，在 Camel 2.x 中，我们改为使用 http://camel.apache.org/schema/cxf 作为 CXF 端点的目标命名空间。

```
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:cxf="http://camel.apache.org/schema/cxf"
      xsi:schemaLocation="
      http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
      http://camel.apache.org/schema/cxf http://camel.apache.org/schema/cxf/camel-cxf.xsd
      http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd
">
 ...
```



注意

在 Apache Camel 2.x 中，http://activemq.apache.org/camel/schema/cxfEndpoint 命名空间被改为 http://camel.apache.org/schema/cxf。

务必包含 root Bean 元素上指定的 JAX-WS schemaLocation 属性。这允许 CXF 验证文件，并是必需的。另请注意 < cxf:cxfEndpoint/> 标签末尾的命名空间声明是必需的，因为此标签的属性值不支持合并

{namespace}localName 语法。

cxf:cxfEndpoint 元素支持很多附加属性：

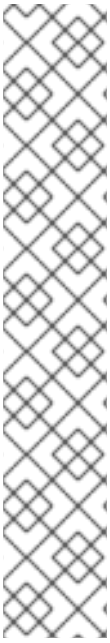
Name	值
portName	此服务实施的端点名称，它映射到 **wsdl:port@name**。在 **ns:PORT_NAME** 格式中，其中 **ns** 是在此范围内有效的命名空间前缀。
serviceName	此服务实施的服务名称，它映射到 **wsdl:service@name**。在 **ns:SERVICE_NAME** 格式中，其中 **ns** 是此范围有效的命名空间前缀。
wsdlURL	WSDL 的位置。可以在 classpath、文件系统上，也可以远程托管。
bindingId	要使用的服务模型的 **bindingId**。
address	服务发布地址。
bus	JAX-WS 端点中使用的总线名称。
serviceClass	可以具有 JSR181 注解或没有该类的 SEI （服务端点接口）类的名称。

它还支持许多子元素：

Name	值
cxf:inInterceptors	此端点的传入拦截器。< **bean> 或 &lt**; **ref>** 列表。
cxf:inFaultInterceptors	此端点的传入错误拦截器。< **bean> 或 &lt**; **ref>** 列表。
cxf:outInterceptors	此端点的传出拦截器。< **bean> 或 &lt**; **ref>** 列表。
cxf:outFaultInterceptors	此端点的传出错误拦截器。< **bean> 或 &lt**; **ref>** 列表。
cxf:properties	提供给 JAX-WS 端点的属性映射。
cxf:handlers	提供给 JAX-WS 端点的 JAX-WS 处理程序列表。

cxf:dataBinding	您可以指定端点中使用哪个 **DataBinding**。这可以通过 Spring < **bean class="MyDataBinding"/>** 语法提供。
cxf:binding	您可以指定要使用的此端点的 **BindingFactory**。这可以通过 Spring < **bean class="MyBindingFactory"/>** 语法提供。
cxf:features	保存此端点拦截器的功能。< **bean&gt**; s 或 < **ref>**s 列表
cxf:schemaLocations	要使用的端点的 schema 位置。< **schemaLocation> s**列表
cxf:serviceFactory	要使用此端点的服务工厂。这可以通过 Spring < **bean class="MyServiceFactory"/>** 语法提供

您可以在此处找到更详细的示例，它们演示了如何提供拦截器、属性和处理程序：
http://cwiki.apache.org/CXF20DOC/jax-ws-configuration.html



注意

您可以使用 CXF:properties 从 Spring 配置文件设置 CXF 端点的 dataFormat 和 setDefaultBus 属性，如下所示：

```
<cxf:cxfEndpoint id="testEndpoint" address="http://localhost:9000/router"
    serviceClass="org.apache.camel.component.cxf.HelloService"
    endpointName="s:PortName"
    serviceName="s:ServiceName"
    xmlns:s="http://www.example.com/test">
    <cxf:properties>
      <entry key="dataFormat" value="MESSAGE"/>
      <entry key="setDefaultBus" value="true"/>
    </cxf:properties>
  </cxf:cxfEndpoint>
```

如何将 CAMEL-CXF 组件使用 LOG4J 而不是 JAVA.UTIL.LOGGING

cxf 的默认日志记录器是 java.util.logging。如果要将其更改为 log4j，请按如下所示操作：在 classpath 中，创建一个名为 META-INF/cxf/org.apache.cxf.logger 的文件。此文件必须在一行中包含类的完全限定名称 org.apache.cxf.common.logging.Log4jLogger，且无注释。

如何使用 XML 启动文档让 CAMEL-CXF 响应消息

如果您使用一些 SOAP 客户端，如 PHP，可能会导致此类错误，因为 CXF 不会添加 XML 启动文档 <?xml version="1.0" encoding="utf-8"?>。

```
Error:sendSms: SoapFault exception: [Client] looks like we got no XML document in [...]
```

要解决这个问题，您只需要告诉 StaxOutInterceptor 为您编写 XML 启动文档。

```
public class WriteXmlDeclarationInterceptor extends AbstractPhaseInterceptor<SoapMessage> {
    public WriteXmlDeclarationInterceptor() {
        super(Phase.PRE_STREAM);
        addBefore(StaxOutInterceptor.class.getName());
    }

    public void handleMessage(SoapMessage message) throws Fault {
        message.put("org.apache.cxf.stax.force-start-document", Boolean.TRUE);
    }

}
```

您可以添加客户拦截器，并将其配置为 camel-cxf endpont

```
<cxf:cxfEndpoint id="routerEndpoint"
address="http://localhost:${CXFTestSupport.port2}/CXFGreeterRouterTest/CamelContext/RouterPort"

  serviceClass="org.apache.hello_world_soap_http.GreeterImpl"
  skipFaultLogging="true">
    <cxf:outInterceptors>
        <!-- This interceptor forces the CXF server send the XML start document to client -->
        <bean class="org.apache.camel.component.cxf.WriteXmlDeclarationInterceptor"/>
    </cxf:outInterceptors>
    <cxf:properties>
        <!-- Set the publishedEndpointUrl which could override the service address from generated
WSDL as you want -->
        <entry key="publishedEndpointUrl" value="http://www.simple.com/services/test" />
    </cxf:properties>
 </cxf:cxfEndpoint>
```

或者，如果您正在使用 Camel 2.4，则为它添加一个消息标头。

```
// set up the response context which force start document
Map<String, Object> map = new HashMap<String, Object>();
map.put("org.apache.cxf.stax.force-start-document", Boolean.TRUE);
exchange.getOut().setHeader(Client.RESPONSE_CONTEXT, map);
```

如何以 POJO 数据格式消耗来自 CAMEL-CXF 端点的消息

camel-cxf 端点消费者 POJO 数据格式基于 cxf 调用器，因此消息标题具有名为 CxfConstants.OPERATION_NAME 的属性，消息正文是 SEI 方法参数的列表。

```
public class PersonProcessor implements Processor {

    private static final transient Logger LOG = LoggerFactory.getLogger(PersonProcessor.class);

    @SuppressWarnings("unchecked")
    public void process(Exchange exchange) throws Exception {
        LOG.info("processing exchange in camel");

        BindingOperationInfo boi =
(BindingOperationInfo)exchange.getProperty(BindingOperationInfo.class.toString());
        if (boi != null) {
            LOG.info("boi.isUnwrapped" + boi.isUnwrapped());
        }
        // Get the parameters list which element is the holder.
        MessageContentsList msgList = (MessageContentsList)exchange.getIn().getBody();
        Holder<String> personId = (Holder<String>)msgList.get(0);
        Holder<String> ssn = (Holder<String>)msgList.get(1);
        Holder<String> name = (Holder<String>)msgList.get(2);

        if (personId.value == null || personId.value.length() == 0) {
            LOG.info("person id 123, so throwing exception");
            // Try to throw out the soap fault message
            org.apache.camel.wsdl_first.types.UnknownPersonFault personFault =
                new org.apache.camel.wsdl_first.types.UnknownPersonFault();
            personFault.setPersonId("");
            org.apache.camel.wsdl_first.UnknownPersonFault fault =
                new org.apache.camel.wsdl_first.UnknownPersonFault("Get the null value of person name",
personFault);
            // Since camel has its own exception handler framework, we can't throw the exception to
trigger it
            // We just set the fault message in the exchange for camel-cxf component handling and return
            exchange.getOut().setFault(true);
            exchange.getOut().setBody(fault);
            return;
        }

        name.value = "Bonjour";
        ssn.value = "123";
        LOG.info("setting Bonjour as the response");
        // Set the response message, first element is the return value of the operation,
        // the others are the holders of method parameters
        exchange.getOut().setBody(new Object[] {null, personId, ssn, name});
    }

}
```

如何以 POJO 数据格式为 CAMEL-CXF 端点准备消息

camel-cxf 端点生成者基于 cxf 客户端 API。首先，您需要在消息标头中指定操作名称，然后将方法参

数添加到列表中，然后使用此参数列表初始化消息。响应消息的正文是 **messageContentsList**，您可以从该列表获取结果。

如果您没有在消息标头中指定操作名称，**CxfProducer** 会尝试使用 **CxfEndpoint** 中的 **defaultOperationName**。如果 **CxfEndpoint** 上没有设置 **defaultOperationName**，它会从操作列表中选择第一个操作名称。

如果要从消息正文获取对象数组，您可以使用 **message.getbody (Object[].class)** 获取正文，如下所示：

```
Exchange senderExchange = new DefaultExchange(context, ExchangePattern.InOut);
final List<String> params = new ArrayList<String>();
// Prepare the request message for the camel-cxf procedure
params.add(TEST_MESSAGE);
senderExchange.getIn().setBody(params);
senderExchange.getIn().setHeader(CxfConstants.OPERATION_NAME, ECHO_OPERATION);

Exchange exchange = template.send("direct:EndpointA", senderExchange);

org.apache.camel.Message out = exchange.getOut();
// The response message's body is an MessageContentsList which first element is the return value of
the operation,
// If there are some holder parameters, the holder parameter is filled in the reset of List.
// The result is extract from the MessageContentsList with the String class type
MessageContentsList result = (MessageContentsList)out.getBody();
LOG.info("Received output text: " + result.get(0));
Map<String, Object> responseContext = CastUtils.cast((Map<?, ?
>)out.getHeader(Client.RESPONSE_CONTEXT));
assertNotNull(responseContext);
assertEquals("The response context", "UTF-8",
responseContext.get(org.apache.cxf.message.Message.ENCODING));
assertEquals("Reply body on Camel is wrong", "echo " + TEST_MESSAGE, result.get(0));
```

如何以 PAYLOAD 数据格式处理 CAMEL-CXF 端点的消息

在 **Apache Camel 2.0** 中：**CxfMessage.getBody （）** 返回 **org.apache.camel.component.cxf.CxfPayload** 对象，它具有 **SOAP** 消息标头和 **Body** 元素的 **getter**。这个更改支持将原生 **CXF** 消息与 **Apache Camel** 消息分离。

```
protected RouteBuilder createRouteBuilder() {
    return new RouteBuilder() {
        public void configure() {
            from(SIMPLE_ENDPOINT_URI + "&dataFormat=PAYLOAD").to("log:info").process(new
Processor() {
                @SuppressWarnings("unchecked")
                public void process(final Exchange exchange) throws Exception {
                    CxfPayload<SoapHeader> requestPayload =
exchange.getIn().getBody(CxfPayload.class);
```

```
                List<Source> inElements = requestPayload.getBodySources();
                List<Source> outElements = new ArrayList<Source>();
                // You can use a customer toStringConverter to turn a CxfPayLoad message into String
as you want
                String request = exchange.getIn().getBody(String.class);
                XmlConverter converter = new XmlConverter();
                String documentString = ECHO_RESPONSE;

                Element in = new XmlConverter().toDOMElement(inElements.get(0));
                // Just check the element namespace
                if (!in.getNamespaceURI().equals(ELEMENT_NAMESPACE)) {
                    throw new IllegalArgumentException("Wrong element namespace");
                }
                if (in.getLocalName().equals("echoBoolean")) {
                    documentString = ECHO_BOOLEAN_RESPONSE;
                    checkRequest("ECHO_BOOLEAN_REQUEST", request);
                } else {
                    documentString = ECHO_RESPONSE;
                    checkRequest("ECHO_REQUEST", request);
                }
                Document outDocument = converter.toDOMDocument(documentString);
                outElements.add(new DOMSource(outDocument.getDocumentElement()));
                // set the payload header with null
                CxfPayload<SoapHeader> responsePayload = new CxfPayload<SoapHeader>(null,
outElements, null);
                exchange.getOut().setBody(responsePayload);
            }
        });
    }
};
}
```

如何在 POJO 模式中获取和设置 SOAP 标头

POJO 表示，当 CXF 端点生成或使用 Camel 交换时，数据格式是 Java 对象的列表。尽管 Apache Camel 在此模式中将消息正文公开为 POJO，但 CXF 组件仍然提供对读取和写入 SOAP 标头的访问。但是，由于 CXF 拦截器在处理后从标头列表中删除带外 SOAP 标头，因此只有 POJO 模式才提供带外 SOAP 标头。

以下示例演示了如何获取/设置 SOAP 标头。假设我们有一个路由，它从一个 CXF 端点转发到另一个端点。也就是说，SOAP Client → Apache Camel → CXF 服务。在请求进入 CXF 服务之前，我们可以附加两个处理器以在(1)获取/插入 SOAP 标头，然后再返回到 SOAP 客户端。本例中的 processor (1)和(2)是 InsertRequestOutHeaderProcessor 和 InsertResponseOutHeaderProcessor。我们的路由类似如下：

```
<route>
    <from uri="cxf:bean:routerRelayEndpointWithInsertion"/>
    <process ref="InsertRequestOutHeaderProcessor" />
    <to uri="cxf:bean:serviceRelayEndpointWithInsertion"/>
    <process ref="InsertResponseOutHeaderProcessor" />
</route>
```

在 2.x SOAP 标头中，会传播到 Apache Camel 消息标头。Apache Camel 消息标头名称是 org.apache.cxf.headers.Header.list，它是 CXF 中定义的常量 (org.apache.cxf.headers.Header.Header.HEADER_LIST)。标头值是 CXF SoapHeader 对象 (org.apache.cxf.binding.soap.SoapHeader)的 List <>。以下片段是 InsertResponseOutHeaderProcessor （它会在响应消息中插入一个新的 SOAP 标头）。在 InsertResponseOutHeaderProcessor 和 InsertRequestOutHeaderProcessor 中访问 SOAP 标头的方式实际相同。两个处理器之间的唯一区别是设置插入 SOAP 标头的方向。

```
public static class InsertResponseOutHeaderProcessor implements Processor {

    @SuppressWarnings("unchecked")
    public void process(Exchange exchange) throws Exception {
        // If exchange is routed from camel-cxf endpoint, this is the header
        List<SoapHeader> soapHeaders = CastUtils.cast((List<?
>)exchange.getIn().getHeader(Header.HEADER_LIST));
        if (soapHeaders == null) {
            // we just create a new soap headers in case the header is null
            soapHeaders = new ArrayList<SoapHeader>();
        }

        // Insert a new header
        String xml = "<?xml version=\"1.0\" encoding=\"utf-8\"?><outofbandHeader "
            + "xmlns=\"http://cxf.apache.org/outofband/Header\" hdrAttribute=\"testHdrAttribute\" "
            + "xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\" soap:mustUnderstand=\"1\">"
            + "<name>New_testOobHeader</name><value>New_testOobHeaderValue</value>
</outofbandHeader>";
        SoapHeader newHeader = new SoapHeader(soapHeaders.get(0).getName(),
                DOMUtils.readXml(new StringReader(xml)).getDocumentElement());
        // make sure direction is OUT since it is a response message.
        newHeader.setDirection(Direction.DIRECTION_OUT);
        //newHeader.setMustUnderstand(false);
        soapHeaders.add(newHeader);

    }

}
```

如何在 PAYLOAD 模式中获取和设置 SOAP 标头

我们已演示了如何以 PAYLOAD 模式访问 SOAP 消息(CxfPayload 对象)（请参阅 如何处理 PAYLOAD 数据格式的 camel-cxf 端点的消息）。

获取 CxfPayload 对象后，您可以调用 CxfPayload.getHeaders （） 方法，该方法返回 DOM Elements (SOAP 标头) 列表。

```
from(getRouterEndpointURI()).process(new Processor() {
    @SuppressWarnings("unchecked")
    public void process(Exchange exchange) throws Exception {
        CxfPayload<SoapHeader> payload = exchange.getIn().getBody(CxfPayload.class);
```

```
        List<Source> elements = payload.getBodySources();
        assertNotNull("Elements here", elements);
        assertEquals("Get the wrong elements size", 1, elements.size());

        Element el = new XmlConverter().toDOMElement(elements.get(0));
        elements.set(0, new DOMSource(el));
        assertEquals("Get the wrong namespace URI", "http://camel.apache.org/pizza/types",
                el.getNamespaceURI());

        List<SoapHeader> headers = payload.getHeaders();
        assertNotNull("Headers here", headers);
        assertEquals("Get the wrong headers size", headers.size(), 1);
        assertEquals("Get the wrong namespace URI",
                ((Element)(headers.get(0).getObject())).getNamespaceURI(),
                "http://camel.apache.org/pizza/types");
    }

})
.to(getServiceEndpointURI());
```

自 Camel 2.16.0 起，您可以使用 "如何在 POJO 模式中获取和设置 SOAP 标头"一节 中所述的方法设置或获取 SOAP 标头。现在，您可以使用 org.apache.cxf.headers.Header.list 标头来获取和设置 SOAP 标头列表。这意味着，如果您有一个路由从一个 Camel CXF 端点转发到另一个(SOAP Client → Camel → CXF 服务)，则 SOAP 客户端发送的 SOAP 标头现在也转发到 CXF 服务。如果您不希望转发标头，请从 org.apache.cxf.headers.Header.list Camel 标头中删除它们。

SOAP 标头在 MESSAGE 模式中不可用

在跳过 SOAP 处理时，在 MESSAGE 模式中不提供 SOAP 标头。

如何从 APACHE CAMEL 抛出 SOAP 故障

如果您使用 CXF 端点来消耗 SOAP 请求，您可能需要从 camel 上下文抛出 SOAP Fault。基本上，您可以使用 throwFault DSL 来执行此操作；它适用于 POJO、PAYLOAD 和 MESSAGE 数据格式。您可以定义 soap 错误，如下所示：

```
SOAP_FAULT = new SoapFault(EXCEPTION_MESSAGE, SoapFault.FAULT_CODE_CLIENT);
Element detail = SOAP_FAULT.getOrCreateDetail();
Document doc = detail.getOwnerDocument();
Text tn = doc.createTextNode(DETAIL_TEXT);
detail.appendChild(tn);
```

然后按以下方式抛出它：

```
from(routerEndpointURI).setFaultBody(constant(SOAP_FAULT));
```

如果您的 CXF 端点以 MESSAGE 数据格式工作，您可以在消息正文中设置 SOAP Fault 消息，并在消息标头中设置响应代码。

```
from(routerEndpointURI).process(new Processor() {

    public void process(Exchange exchange) throws Exception {
        Message out = exchange.getOut();
        // Set the message body with the
        out.setBody(this.getClass().getResourceAsStream("SoapFaultMessage.xml"));
        // Set the response code here
        out.setHeader(org.apache.cxf.message.Message.RESPONSE_CODE, new Integer(500));
    }

});
```

对于 POJO 数据格式来说也是如此。您可以在 Out 正文上设置 SOAP Fault，并通过调用 Message.setFault (true) 来指示其存在故障，如下所示：

```
from("direct:start").onException(SoapFault.class).maximumRedeliveries(0).handled(true)
    .process(new Processor() {
        public void process(Exchange exchange) throws Exception {
            SoapFault fault = exchange
                .getProperty(Exchange.EXCEPTION_CAUGHT, SoapFault.class);
            exchange.getOut().setFault(true);
            exchange.getOut().setBody(fault);
        }

    }).end().to(serviceURI);
```

如何传播 CXF 端点的请求和响应上下文

CXF 客户端 API 提供了使用请求和响应上下文调用操作的方法。如果您使用 CXF 端点制作者调用外部 Web 服务，您可以设置请求上下文，并使用以下代码获取响应上下文：

```
        CxfExchange exchange = (CxfExchange)template.send(getJaxwsEndpointUri(), new
Processor() {
            public void process(final Exchange exchange) {
                final List<String> params = new ArrayList<String>();
                params.add(TEST_MESSAGE);
                // Set the request context to the inMessage
                Map<String, Object> requestContext = new HashMap<String, Object>();
                requestContext.put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
JAXWS_SERVER_ADDRESS);
                exchange.getIn().setBody(params);
                exchange.getIn().setHeader(Client.REQUEST_CONTEXT , requestContext);
                exchange.getIn().setHeader(CxfConstants.OPERATION_NAME,
GREET_ME_OPERATION);
            }
```

```
        });
        org.apache.camel.Message out = exchange.getOut();
        // The output is an object array, the first element of the array is the return value
        Object\[\] output = out.getBody(Object\[\].class);
        LOG.info("Received output text: " + output\[0\]);
        // Get the response context form outMessage
        Map<String, Object> responseContext =
CastUtils.cast((Map)out.getHeader(Client.RESPONSE_CONTEXT));
        assertNotNull(responseContext);
        assertEquals("Get the wrong wsdl opertion name", "
{http://apache.org/hello_world_soap_http}greetMe",
                responseContext.get("javax.xml.ws.wsdl.operation").toString());
```

附加支持

POJO 模式 ： 支持带有 Attachment 和 MTOM 的 SOAP （请参阅 Payload Mode 为启用 MTOM 的示例）。无论没有测试，带有 Attachment 的 SOAP 不会被测试。因为 2.1.Since 附加功能被推广到 Camel 消息的附件。Since 附加到 POJO，用户通常不需要处理附件。

```
DataHandler Message.getAttachment(String id)
```

.

有效负载模式： 自 2.1 起支持 MTOM。附加功能可以通过上述 Camel 消息 API 检索。不支持使用 Attachment 的 SOAP，因为此模式中没有 SOAP 处理。

要启用 MTOM，将 CXF 端点属性 "mtom_enabled" 设置为 true。（我相信您只能通过 Spring 执行此操作。）

```
<cxf:cxfEndpoint id="routerEndpoint"
address="http://localhost:${CXFTestSupport.port1}/CxfMtomRouterPayloadModeTest/jaxws-
mtom/hello"
        wsdlURL="mtom.wsdl"
        serviceName="ns:HelloService"
        endpointName="ns:HelloPort"
        xmlns:ns="http://apache.org/camel/cxf/mtom_feature">

    <cxf:properties>
        <!--  enable mtom by setting this property to true -->
        <entry key="mtom-enabled" value="true"/>

        <!--  set the camel-cxf endpoint data fromat to PAYLOAD mode -->
        <entry key="dataFormat" value="PAYLOAD"/>
    </cxf:properties>
```

您可以在 Payload 模式中生成带有附件的 Camel 消息，以发送到 CXF 端点。

```
Exchange exchange = context.createProducerTemplate().send("direct:testEndpoint", new
Processor() {

    public void process(Exchange exchange) throws Exception {
        exchange.setPattern(ExchangePattern.InOut);
        List&lt;Source> elements = new ArrayList&lt;Source>();
        elements.add(new DOMSource(DOMUtils.readXml(new
StringReader(MtomTestHelper.REQ_MESSAGE)).getDocumentElement()));
        CxfPayload<SoapHeader> body = new CxfPayload<SoapHeader>(new ArrayList<SoapHeader>
(),
            elements, null);
        exchange.getIn().setBody(body);
        exchange.getIn().addAttachment(MtomTestHelper.REQ_PHOTO_CID,
            new DataHandler(new ByteArrayDataSource(MtomTestHelper.REQ_PHOTO_DATA,
"application/octet-stream")));

        exchange.getIn().addAttachment(MtomTestHelper.REQ_IMAGE_CID,
            new DataHandler(new ByteArrayDataSource(MtomTestHelper.requestJpeg, "image/jpeg")));

    }

});

// process response

CxfPayload<SoapHeader> out = exchange.getOut().getBody(CxfPayload.class);
Assert.assertEquals(1, out.getBody().size());

Map<String, String> ns = new HashMap<String, String>();
ns.put("ns", MtomTestHelper.SERVICE_TYPES_NS);
ns.put("xop", MtomTestHelper.XOP_NS);

XPathUtils xu = new XPathUtils(ns);
Element oute = new XmlConverter().toDOMElement(out.getBody().get(0));
Element ele = (Element)xu.getValue("//ns:DetailResponse/ns:photo/xop:Include", oute,
                        XPathConstants.NODE);
String photoId = ele.getAttribute("href").substring(4); // skip "cid:"

ele = (Element)xu.getValue("//ns:DetailResponse/ns:image/xop:Include", oute,
                        XPathConstants.NODE);
String imageId = ele.getAttribute("href").substring(4); // skip "cid:"

DataHandler dr = exchange.getOut().getAttachment(photoId);
Assert.assertEquals("application/octet-stream", dr.getContentType());
MtomTestHelper.assertEquals(MtomTestHelper.RESP_PHOTO_DATA,
IOUtils.readBytesFromStream(dr.getInputStream()));

dr = exchange.getOut().getAttachment(imageId);
Assert.assertEquals("image/jpeg", dr.getContentType());
```

```
BufferedImage image = ImageIO.read(dr.getInputStream());
Assert.assertEquals(560, image.getWidth());
Assert.assertEquals(300, image.getHeight());
```

您还可以在 Payload 模式中使用从 CXF 端点接收的 Camel 消息。

```
public static class MyProcessor implements Processor {

    @SuppressWarnings("unchecked")
    public void process(Exchange exchange) throws Exception {
        CxfPayload<SoapHeader> in = exchange.getIn().getBody(CxfPayload.class);

        // verify request
        assertEquals(1, in.getBody().size());

        Map<String, String> ns = new HashMap<String, String>();
        ns.put("ns", MtomTestHelper.SERVICE_TYPES_NS);
        ns.put("xop", MtomTestHelper.XOP_NS);

        XPathUtils xu = new XPathUtils(ns);
        Element body = new XmlConverter().toDOMElement(in.getBody().get(0));
        Element ele = (Element)xu.getValue("//ns:Detail/ns:photo/xop:Include", body,
                                   XPathConstants.NODE);
        String photoId = ele.getAttribute("href").substring(4); // skip "cid:"
        assertEquals(MtomTestHelper.REQ_PHOTO_CID, photoId);

        ele = (Element)xu.getValue("//ns:Detail/ns:image/xop:Include", body,
                                   XPathConstants.NODE);
        String imageId = ele.getAttribute("href").substring(4); // skip "cid:"
        assertEquals(MtomTestHelper.REQ_IMAGE_CID, imageId);

        DataHandler dr = exchange.getIn().getAttachment(photoId);
        assertEquals("application/octet-stream", dr.getContentType());
        MtomTestHelper.assertEquals(MtomTestHelper.REQ_PHOTO_DATA,
IOUtils.readBytesFromStream(dr.getInputStream()));

        dr = exchange.getIn().getAttachment(imageId);
        assertEquals("image/jpeg", dr.getContentType());
        MtomTestHelper.assertEquals(MtomTestHelper.requestJpeg,
IOUtils.readBytesFromStream(dr.getInputStream()));

        // create response
        List&lt;Source> elements = new ArrayList&lt;Source>();
        elements.add(new DOMSource(DOMUtils.readXml(new
StringReader(MtomTestHelper.RESP_MESSAGE)).getDocumentElement()));
        CxfPayload&lt;SoapHeader> sbody = new CxfPayload&lt;SoapHeader>(new
ArrayList&lt;SoapHeader>(),
            elements, null);
        exchange.getOut().setBody(sbody);
        exchange.getOut().addAttachment(MtomTestHelper.RESP_PHOTO_CID,
            new DataHandler(new ByteArrayDataSource(MtomTestHelper.RESP_PHOTO_DATA,
"application/octet-stream")));

        exchange.getOut().addAttachment(MtomTestHelper.RESP_IMAGE_CID,
```

```
            new DataHandler(new ByteArrayDataSource(MtomTestHelper.responseJpeg, "image/jpeg")));

    }
}
```

Message Mode: Attachments 不支持，因为它根本不处理消息。

CXF_MESSAGE Mode： 支持 MTOM，而附件则可通过上述 Camel 消息 API 检索。

注意

当收到多部分（即 MTOM）时，将默认的 SOAPMessage 传给 String 转换器在正文上提供完整的多部分有效负载。如果您只需要 SOAP XML 作为 String，您可以使用 message.getSOAPPart （） 设置消息正文，并且 Camel 转换可以为您做剩余工作。

如何传播堆栈追踪信息

可以配置 CXF 端点，以便在服务器端引发 Java 异常时，异常的堆栈追踪会被放入故障消息并返回到客户端。要启用这种情况，请将 dataFormat 设置为 PAYLOAD，并在 cxfEndpoint 元素中将 faultStackTraceEnabled 属性设置为 true，如下所示：

```
<cxf:cxfEndpoint id="router" address="http://localhost:9002/TestMessage"
    wsdlURL="ship.wsdl"
    endpointName="s:TestSoapEndpoint"
    serviceName="s:TestService"
    xmlns:s="http://test">
  <cxf:properties>
    <!-- enable sending the stack trace back to client; the default value is false-->
    <entry key="faultStackTraceEnabled" value="true" /> <entry key="dataFormat" value="PAYLOAD"
/>
  </cxf:properties>
</cxf:cxfEndpoint>
```

为安全起见，堆栈跟踪不包括导致异常（即，由 导致的堆栈追踪的一部分）。如果要在堆栈追踪中包含导致异常，在 cxfEndpoint 元素中将 exceptionMessageCauseEnabled 属性设置为 true，如下所示：

```
<cxf:cxfEndpoint id="router" address="http://localhost:9002/TestMessage"
    wsdlURL="ship.wsdl"
    endpointName="s:TestSoapEndpoint"
    serviceName="s:TestService"
    xmlns:s="http://test">
  <cxf:properties>
    <!-- enable to show the cause exception message and the default value is false -->
    <entry key="exceptionMessageCauseEnabled" value="true" />
```

```
    <!-- enable to send the stack trace back to client, the default value is false-->
    <entry key="faultStackTraceEnabled" value="true" />
    <entry key="dataFormat" value="PAYLOAD" />
  </cxf:properties>
</cxf:cxfEndpoint>
```

警告

仅为测试和诊断目的启用 **exceptionMessageCauseEnabled** 标志。对于服务器而言，正常做法是让恶意用户探测到服务器的最初原因。

PAYLOAD 模式流支持

在 2.8.2 中，camel-cxf 组件现在支持在使用 PAYLOAD 模式时流传输传入的消息。在以前的版本中，传入的信息会被完全解析 DOM。对于大型消息，这非常耗时，使用大量内存。从 2.8.2 开始，在路由时，传入的消息可以保留为 javax.xml.transform.Source，如果没有修改有效负载，则可以直接流传输到目标目的地。对于常见的"简单代理"用例（例如： from ("cxf:...").to ("cxf:...")），这可以提供非常显著的性能增加，并显著降低内存要求。

然而，在有些情况下流可能不合适或需要的。由于流性质，在稍后处理链中可能无法发现无效的传入的 XML。此外，某些操作可能要求消息是 DOM 解析的任何方式（如 WS-Security 或消息追踪等），在这种情况下，流的优势是有限的。此时，可以通过两种方式控制流：

- endpoint 属性：您可以添加 "allowStreaming=false" 作为端点属性，来打开/关闭流。
- 组件属性：CxfComponent 对象也具有 allowStreaming 属性，可以为从该组件创建的端点设置默认值。
- 全局系统属性：如果关闭，您可以添加 org.apache.camel.component.cxf.streaming 的系统属性，以关闭。这会设置全局默认值，但设置上面的 endpoint 属性会覆盖该端点的值。

使用通用 CXF DISPATCH 模式

从 2.8.0，camel-cxf 组件支持通用 CXF 分配模式，它可以传输任意结构的消息（例如，不绑定到特定 XML 模式）。要使用此模式，您只需要省略指定 CXF 端点的 wsdlURL 和 serviceClass 属性。

```
<cxf:cxfEndpoint id="testEndpoint" address="http://localhost:9000/SoapContext/SoapAnyPort">
  <cxf:properties>
    <entry key="dataFormat" value="PAYLOAD"/>
  </cxf:properties>
</cxf:cxfEndpoint>
```

请注意，默认的 CXF 分配客户端不会发送特定的 SOAPAction 标头。因此，当目标服务需要特定的 SOAPAction 值时，会使用键 SOAPAction （不区分大小写）在 Camel 标头中提供。

78.1. WILDFLY 上的 CXF 用户

WildFly 上的 camel-cxf 用户配置与独立 Camel 的配置不同。生产者端点可以正常运行。

在 WildFly 上，camel-cxf 用户利用容器提供的默认 Undertow HTTP 服务器。服务器在 undertow 子系统配置中定义。以下是 standalone.xml 中默认配置的摘录：

```
<subsystem xmlns="urn:jboss:domain:undertow:4.0">
    <buffer-cache name="default" />
    <server name="default-server">
        <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true" />
        <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-
http2="true" />
        <host name="default-host" alias="localhost">
            <location name="/" handler="welcome-content" />
            <filter-ref name="server-header" />
            <filter-ref name="x-powered-by-header" />
            <http-invoker security-realm="ApplicationRealm" />
        </host>
    </server>
</subsystem>
```

在本实例中，Undertow 配置为侦听 http 和 https socket-binding 指定的接口/端口。默认情况下，对于 http，对于端口 8080，https 为 8443。

例如，如果您使用不同的主机或端口组合配置端点使用者，服务器日志文件中会显示警告。例如，以下主机和端口配置将被忽略：

```
<cxf:rsServer id="cxfRsConsumer"
              address="http://somehost:1234/path/to/resource"
              serviceClass="org.example.ServiceClass" />
```

```
<cxf:cxfEndpoint id="cxfWsConsumer"
                 address="http://somehost:1234/path/to/resource"
                 serviceClass="org.example.ServiceClass" />
```

> *[org.wildfly.extension.camel] (pool-2-thread-1) Ignoring configured host: http://somehost:1234/path/to/resource*

但是，消费者在默认主机和端口 localhost:8080 或 localhost:8443 上仍然可用。

注意

使用 camel-cxf 使用者 的应用程序必须 打包为 WAR。在以前的 WildFly-Camel 版本中，允许 JAR 等其他类型的存档，但不再受支持。

78.1.1. 配置替代端口

如果要接受替代端口，则必须通过 WildFly 子系统配置它们。这在服务器文档中解释：

https://access.redhat.com/documentation/zh-cn/red_hat_jboss_enterprise_application_platform/7.1/html/configuration_guide/configuring_the_web_server_undertow

78.1.2. 配置 SSL

要配置 SSL，请参阅 WildFly SSL 配置指南：

https://access.redhat.com/documentation/zh-cn/red_hat_jboss_enterprise_application_platform/7.1/html-single/how_to_configure_server_security/#configure_one_way_and_two_way_ssl_tls_for_application

78.1.3. 使用 Elytron 配置安全性

WildFly-Camel 支持使用 Elytron 安全框架保护 camel-cxf 消费者端点。

78.1.3.1. 配置安全域

要使用 Elytron 保护 WildFly-Camel 应用程序，需要在 WAR 部署的 WEB-INF/jboss-web.xml 中引用应用程序安全域：

```
<jboss-web>
  <security-domain>my-application-security-domain</security-domain>
</jboss-web>
```

& lt;security-domain& gt; 配置引用 Undertow 子系统定义的 < application-security-domain > 的名称。例如，Undertow 子系统 &lt ;application-security-domain > 在 WildFly 服务器 standalone.xml 配置文件中配置，如下所示：

```
<subsystem xmlns="urn:jboss:domain:undertow:6.0">
    ...
    <application-security-domains>
        <application-security-domain name="my-application-security-domain" http-authentication-factory="application-http-authentication"/>
    </application-security-domains>
</subsystem>
```

& lt;http-authentication-factory > application-http-authentication 在 Elytron 子系统中定义。application-http-authentication 默认在 standalone.xml 和 standalone-full.xml 服务器配置文件中可用。例如：

```
<subsystem xmlns="urn:wildfly:elytron:1.2">
    ...
    <http>
        ...
        <http-authentication-factory name="application-http-authentication" http-server-mechanism-factory="global" security-domain="ApplicationDomain">
            <mechanism-configuration>
                <mechanism mechanism-name="BASIC">
                    <mechanism-realm realm-name="Application Realm" />
                </mechanism>
                <mechanism mechanism-name="FORM" />
            </mechanism-configuration>
        </http-authentication-factory>
        <provider-http-server-mechanism-factory name="global" />
    </http>
    ...
</subsystem>
```

& lt;http-authentication-factory > 名为 application-http-authentication，包含对名为 ApplicationDomain 的 Elytron 安全域的引用。

有关如何配置 Elytron 子系统的更多信息，请参阅 Elytron 文档。

78.1.3.2. 配置安全约束、身份验证方法和安全角色

安全约束、camel-cxf 消费者端点的身份验证方法和安全角色可以在 WAR 部署 WEB-INF/web.xml

中配置。例如，配置 BASIC 身份验证：

```
<web-app>
 <security-constraint>
  <web-resource-collection>
   <web-resource-name>secure</web-resource-name>
   <url-pattern>/webservices/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
   <role-name>my-role</role-name>
  </auth-constraint>
 </security-constraint>
 <security-role>
  <description>The role that is required to log in to /webservices/*</description>
  <role-name>my-role</role-name>
 </security-role>
 <login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>my-realm</realm-name>
 </login-config>
</web-app>
```

请注意，由 Servlet 规范定义的 <url-pattern > 相对于 web 应用程序的上下文路径。如果您的应用程序被打包为 my-app.war，Wild 将在上下文路径 /my-app 下访问它，< url-patternpattern&gt ; /webservices GovCloud 将应用到相对于 /my-app 的路径。

例如，针对 http://my-server/my-app/webservices/my-endpoint 的请求将与 /webservices netobserv 模式匹配，而 http://my-server/webservices/my-endpoint 不匹配。

这很重要，因为 WildFly-Camel 允许创建 camel-cxf 端点消费者，其基本路径位于主机 Web 应用上下文路径之外。例如，可以为 my-app.war 中的 http://my-server/webservices/my-endpoint 创建 camel-cxf 使用者。

要为此类上下文端点定义安全约束，Wild-Camel 支持自定义非标准 & lt; url-pattern& gt; 约定，其中为带有三个正斜杠 /// 前缀的模式作为服务器主机名。例如，要在 my-app.war 内保护 http://my-server/webservices/my-endpoint，您可以将以下配置添加到 web.xml 中：

```
<web-app>
 <security-constraint>
  <web-resource-collection>
   <web-resource-name>secure</web-resource-name>
   <url-pattern>///webservices/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
   <role-name>my-role</role-name>
  </auth-constraint>
 </security-constraint>
```

```
  <security-role>
    <description>The role that is required to log in to /webservices/*</description>
    <role-name>my-role</role-name>
  </security-role>
  <login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>my-realm</realm-name>
  </login-config>
</web-app>
```

第 79 章 CXF-RS COMPONENT

从 Camel 版本 2.0 开始提供

cxfrs: 组件提供与 Apache CXF 集成，以连接到 CXF 中托管的 JAX-RS 1.1 和 2.0 服务。

当使用 CXF 作为消费者时，CXF Bean 组件 允许您考虑从其处理中作为 RESTful 或 SOAP web 服务接收消息有效负载的方式。这可能会使用多语言传输来使用 Web 服务。bean 组件的配置也更为简单，提供了使用 Camel 和 CXF 实施 Web 服务的最快方法。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-cxf</artifactId>
  <version>x.x.x</version>  <!-- use the same version as your Camel core version -->
</dependency>
```

79.1. URI 格式

```
cxfrs://address?options
```

其中 address 代表 CXF 端点的地址

```
cxfrs:bean:rsEndpoint
```

其中 rsEndpoint 代表 spring bean 的名称，其显示 CXFRS 客户端或服务器

对于上述任一样式，您可以按如下方式将选项附加到 URI：

```
cxfrs:bean:cxfEndpoint?resourceClasses=org.apache.camel.rs.Example
```

79.2. 选项

CXF-RS 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

CXF-RS 端点使用 URI 语法进行配置：

```
cxfrs:beanId:address
```

使用以下路径和查询参数：

79.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
beanId	要查找现有配置的 CxfRsEndpoint。必须使用 bean: 作为前缀。		字符串
address	服务发布地址。		字符串

79.2.2. 查询参数(30 参数)：

Name	描述	默认值	类型
features (common)	将功能列表设置为 CxfRs 端点。		list
loggingFeatureEnabled (common)	这个选项启用 CXF Logging 功能，将入站和出站 REST 消息写入日志。	false	布尔值
loggingSizeLimit (common)	为限制启用日志记录功能时，日志记录器将输出的字节数。		int

Name	描述	默认值	类型
modelRef (common)	此选项用于指定对没有注解的资源类有用的模型文件。使用这个选项时，可以省略服务类，以模拟仅文档的端点		字符串
providers (common)	设置自定义 JAX-RS 提供程序列表到 CxfRs 端点。您可以在以逗号分开的 registy 中指定要查询的供应商列表指定字符串。		字符串
resourceClasses (common)	要导出为 REST 服务的资源类。可以使用逗号分隔多个类。		list
schemaLocations (common)	设置架构的位置，可用于验证传入的 XML 或 JAXB 驱动的 JSON。		list
skipFaultLogging (common)	这个选项控制 PhaseInterceptorChain 是否跳过记录它捕获的 Fault。	false	布尔值
bindingStyle (consumer)	设置请求和响应如何映射到/来自 Camel。可能有两个值： SimpleConsumer：此绑定样式处理请求参数、多部分等，并将它们映射到 IN 标头、IN附件和消息正文。它旨在消除 org.apache.cxf.message.MessageContentsList 的低级别处理。它还为响应映射增加了更大的灵活性和简洁性。仅适用于消费者。Default ：默认风格。对于消费者，这会将 MessageContentsList 传递给路由，需要在路由中处理低级处理。这是传统的绑定样式，只需要将 org.apache.cxf.message.MessageContentsList 转储到 IN 消息正文。然后，用户负责根据 JAX-RS 方法签名所定义的合同进行处理。custom ：允许您通过 binding 选项指定自定义绑定。	default	BindingStyle
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
publishedEndpointUrl (consumer)	这个选项可以覆盖从 WADL 发布的 endpointUrl，该端点可以使用资源地址 url 和 _wadl 访问		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler

Name	描述	默认值	类型
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
serviceBeans (consumer)	要导出为 REST 服务的服务 Bean。可以使用逗号分隔多个 Bean。		list
cookieHandler (producer)	配置 Cookie 处理程序，以维护 HTTP 会话		CookieHandler
hostnameVerifier (producer)	要使用的主机名验证器。使用 # 表示法引用 registry 中的 HostnameVerifier。		HostnameVerifier
sslContextParameters (producer)	Camel SSL 设置参考。使用 # 表示法引用 SSL 上下文。		SSLContextParameters
throwExceptionOnFailure (producer)	此选项告知 CxfRsProducer 检查返回代码，并在返回码大于 207 时生成一个例外。	true	布尔值
httpClientAPI (producer)	如果为 true，CxfRsProducer 将使用 HttpClientAPI 调用该服务。如果是 false，CxfRsProducer 将使用 ProxyClientAPI 调用服务	true	布尔值
ignoreDeleteMethodMessageBody (producer)	此选项用于告知 CxfRsProducer 在使用 HTTP API 时忽略 DELETE 方法的消息正文。	false	布尔值
maxClientCacheSize (producer)	这个选项允许您配置缓存的最大大小。实现在 CxfProvider 和 CxfRsProvider 中缓存 CXF 客户端或 ClientFactoryBean。	10	int
绑定 (advanced)	使用自定义 CxfBinding 控制 Camel 消息和 CXF 消息之间的绑定。		CxfRsBinding
总线 (advanced)	使用自定义配置的 CXF 总线。		Bus
continuationTimeout (advanced)	这个选项用于设置 CXF continuation 超时，当 CXF 服务器使用 Jetty 或 Servlet 传输时，默认可在 CxfConsumer 中使用它。	30000	long
cxfRsEndpointConfigurer (advanced)	这个选项可以应用 org.apache.camel.component.cxf.jaxrs.CxfRsEndpointConfigurer 的实现，它支持以编程方式配置 CXF 端点。用户可以通过实施 CxfEndpointConfigurer 的 configureServer/Client 方法来配置 CXF 服务器和客户端。		CxfRsEndpointConfigurer

Name	描述	默认值	类型
defaultBus (advanced)	当 CXF 端点自行创建总线时，将设置默认总线	false	布尔值
headerFilterStrategy (advanced)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
performInvocation (advanced)	当选项为 true 时，Camel 将执行资源类实例的调用，并将响应对象放入交换中以进一步处理。	false	布尔值
propagateContexts (advanced)	当选项为 true 时，JAXRS UriInfo, HttpHeaders, Request 和 SecurityContext 上下文将可用于自定义 CXFRS 处理器，作为类型的 Camel Exchange 属性。这些上下文可用于利用 JAX-RS API 分析当前请求。	false	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

79.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.cxfrs.enabled	启用 cxfrs 组件	true	布尔值
camel.component.cxfrs.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		字符串
camel.component.cxfrs.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.cxfrs.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

您还可以通过 spring 配置配置 CXF REST 端点。由于 CXF REST 客户端和 CXF REST 服务器之间存在许多区别，因此我们为它们提供不同的配置。如需更多信息，请查看架构文件和 CXF JAX-RS 文档。

79.4. 如何在 CAMEL 中配置 REST 端点

在 camel-cxf 模式文件 中，REST 端点定义有两个元素。cxf:rsServer 用于 REST consumer， cxf:rsClient 用于 REST producer。
您可以在此处找到 Camel REST 服务路由配置示例。

79.5. 如何从消息标头覆盖 CXF PRODUCER 地址

camel-cxfrs 生成者支持通过设置消息 "CamelDestinationOverrideUrl" 键来覆盖服务地址。

```
// set up the service address from the message header to override the setting of CXF endpoint
 exchange.getIn().setHeader(Exchange.DESTINATION_OVERRIDE_URL,
constant(getServiceAddress()));
```

79.6. 消耗 REST 请求 - SIMPLE BINDING STYLE

从 Camel 2.11 开始提供

Default 绑定风格是低级的，要求用户手动处理来自路由的 MessageContentsList 对象。因此，它将路由逻辑与 JAX-RS 操作的方法签名和参数索引紧密耦合。微不足道、困难和容易出错。

相反，SimpleConsumer 绑定样式执行以下映射，以便在 Camel 消息内 使请求数据更易访问 ：

- JAX-RS 参数(@HeaderParam、@QueryParam 等)注入 IN 消息标头。标头名称与注解值匹配。
- 请求实体(POJO 或其他类型)成为 IN 消息正文。如果无法在 JAX-RS 方法签名中识别单个实体， 它将回退到原始的 MessageContentsList。
- 二进制 @Multipart 正文部分成为 IN 消息附件，支持 DataHandler、InputStream、DataSource 和 CXF 的 Attachment 类。
- 非二进制 @Multipart 正文部分映射为 IN 消息标头。标头名称与 Body Part 名称匹配。

另外，以下规则适用于 Response 映射 ：

- 如果消息正文类型与 javax.ws.rs.core.Response (user-built response)不同，则创建新的响应，并且消息正文设置为实体（因此其不是 null）。响应状态代码从 Exchange.HTTP_RESPONSE_CODE 标头获取，如果不存在，则默认为 200 OK。

- 如果消息正文类型等于 javax.ws.rs.core.Response，这表示用户已构建自定义响应，因此它会被遵守，并成为最终响应。

- 在所有情况下，自定义或默认 HeaderFilterStrategy 允许的 Camel 标头都添加到 HTTP 响应中。

79.6.1. 启用 Simple Binding Style

通过将消费者端点中的 bindingStyle 参数设置为 SimpleConsumer，可以激活这个绑定风格：

```
from("cxfrs:bean:rsServer?bindingStyle=SimpleConsumer")
  .to("log:TEST?showAll=true");
```

79.6.2. 使用不同方法签名的请求绑定示例

以下是方法签名列表以及来自 Simple 绑定的预期结果。

公共响应 doAction (BusinessObject 请求);
请求有效负载放置在 IN 消息正文中，替换原始的 MessageContentsList。

Public Response doAction (BusinessObject request, @HeaderParam ("abcd") String abcd, @QueryParam ("defg") String defg"); Request payload put in IN message body, 替换原始 MessageContentsList。两个请求参数都映射为 IN 消息标头，其名称为 abcd 和 defg。

公共响应 doAction (@HeaderParam ("abcd") String abcd, @QueryParam ("defg") String defg); Both request params mapped as IN message headers with name abcd and defg.原始 MessageContentsList 被保留，即使它只包含 2 参数。

公共响应 doAction (@Multipart (value="body1") BusinessObject 请求, @Multipart (value="body2") BusinessObject request2); 第一个参数作为带有名称 body1 的标头传输，第二个参

数被映射为标头 body2。原始 MessageContentsList 保留为 IN message body。

公共响应 doAction (InputStream abcd); InputStream is unwrapped from the MessageContentsList，并保留为 IN 消息正文。

Public Response doAction (DataHandler abcd); DataHandler is unwrapped from the MessageContentsList，并保留为 IN 消息正文。

79.6.3. Simple Binding Style 的更多示例

使用此方法给定 JAX-RS 资源类：

```
@POST @Path("/customers/{type}")
public Response newCustomer(Customer customer, @PathParam("type") String type,
@QueryParam("active") @DefaultValue("true") boolean active) {
    return null;
}
```

通过以下路由提供服务：

```
from("cxfrs:bean:rsServer?bindingStyle=SimpleConsumer")
    .recipientList(simple("direct:${header.operationName}"));

from("direct:newCustomer")
    .log("Request: type=${header.type}, active=${header.active}, customerData=${body}");
```

以下带有 XML 有效负载的 HTTP 请求（代表客户 DTO 是 JAXB-annotated）：

```
POST /customers/gold?active=true

Payload:
<Customer>
  <fullName>Raul Kripalani</fullName>
  <country>Spain</country>
  <project>Apache Camel</project>
</Customer>
```

将打印消息：

```
Request: type=gold, active=true, customerData=<Customer.toString() representation>
```

有关如何处理请求和写入响应的更多示例，请访问 https://svn.apache.org/repos/asf/camel/trunk/components/camel-cxf/src/test/java/org/apache/camel/component/cxf/jaxrs/simplebinding/。

79.7. 消耗 REST 请求 - 默认绑定 STYLE

CXF JAXRS 前端实施 JAX-RS (rust-311) API，因此我们可以将资源类导出为 REST 服务。我们利用 CXF Invoker API 将 REST 请求转换为普通的 Java 对象方法调用。与 Camel Restlet 组件不同，您不需要在端点中指定 URI 模板，CXF 根据 JSR-311 规范，处理 REST 请求 URI 到资源类方法映射。您需要在 Camel 中完成的所有操作，将此方法请求委托给正确的处理器或端点。

以下是 CXFRS route... 的示例

```
private static final String CXF_RS_ENDPOINT_URI =
        "cxfrs://http://localhost:" + CXT + "/rest?
resourceClasses=org.apache.camel.component.cxf.jaxrs.testbean.CustomerServiceResource
";
private static final String CXF_RS_ENDPOINT_URI2 =
        "cxfrs://http://localhost:" + CXT + "/rest2?
resourceClasses=org.apache.camel.component.cxf.jaxrs.testbean.CustomerService";
private static final String CXF_RS_ENDPOINT_URI3 =
        "cxfrs://http://localhost:" + CXT + "/rest3?"
        +
"resourceClasses=org.apache.camel.component.cxf.jaxrs.testbean.CustomerServiceNoAnnot
ations&"
        +
"modelRef=classpath:/org/apache/camel/component/cxf/jaxrs/CustomerServiceModel.xml";
private static final String CXF_RS_ENDPOINT_URI4 =
        "cxfrs://http://localhost:" + CXT + "/rest4?"
        +
"modelRef=classpath:/org/apache/camel/component/cxf/jaxrs/CustomerServiceDefaultHandler
Model.xml";
private static final String CXF_RS_ENDPOINT_URI5 =
        "cxfrs://http://localhost:" + CXT + "/rest5?"
        + "propagateContexts=true&"
        +
"modelRef=classpath:/org/apache/camel/component/cxf/jaxrs/CustomerServiceDefaultHandler
Model.xml";
protected RouteBuilder createRouteBuilder() throws Exception {
    final Processor testProcessor = new TestProcessor();
    final Processor testProcessor2 = new TestProcessor2();
    final Processor testProcessor3 = new TestProcessor3();
    return new RouteBuilder() {
        public void configure() {
            errorHandler(new NoErrorHandlerBuilder());
            from(CXF_RS_ENDPOINT_URI).process(testProcessor);
            from(CXF_RS_ENDPOINT_URI2).process(testProcessor);
            from(CXF_RS_ENDPOINT_URI3).process(testProcessor);
            from(CXF_RS_ENDPOINT_URI4).process(testProcessor2);
```

```
            from(CXF_RS_ENDPOINT_URI5).process(testProcessor3);
        }
    };
}
```

以及用于配置 endpoint... 的对应资源类

INFO:*Note about resource class*

默认情况下，JAX-RS 资源类仅*用于配置 JAX-RS 属性。在将信息路由到端点期间，将执行方法。相反，它负责执行所有处理的路由。

请注意，从 Camel 2.15 开始，仅提供一个接口，而不是默认模式的 no-op 服务实施类。

从 Camel 2.15 开始，如果启用了 performInvocation 选项，将首先调用服务实施，会在 Camel 交换上设置响应，并且路由执行将照常进行。这对于将现有的 JAX-RS 实现集成到 Camel 路由以及自定义处理器中的后处理 JAX-RS 响应时非常有用。

```
@Path("/customerservice/")
public interface CustomerServiceResource {

    @GET
    @Path("/customers/{id}/")
    Customer getCustomer(@PathParam("id") String id);

    @PUT
    @Path("/customers/")
    Response updateCustomer(Customer customer);

    @Path("/{id}")
    @PUT()
    @Consumes({ "application/xml", "text/plain",
                "application/json" })
    @Produces({ "application/xml", "text/plain",
                "application/json" })
    Object invoke(@PathParam("id") String id,
                String payload);
}
```

79.8. 如何通过 CAMEL-CXFRS PRODUCER 调用 REST 服务

CXF JAXRS 前端 实施 基于代理的客户端 API，通过此 API，您可以通过代理调用远程 REST 服务。camel-cxfrs 生成者基于 此代理 API。您只需要在消息标头中指定操作名称，并在消息正文中准备参

数，camel-cxfrs 生成者将为您生成正确的 REST 请求。

下面是一个示例：

```
Exchange exchange = template.send("direct://proxy", new Processor() {
    public void process(Exchange exchange) throws Exception {
        exchange.setPattern(ExchangePattern.InOut);
        Message inMessage = exchange.getIn();
        // set the operation name
        inMessage.setHeader(CxfConstants.OPERATION_NAME, "getCustomer");
        // using the proxy client API
        inMessage.setHeader(CxfConstants.CAMEL_CXF_RS_USING_HTTP_API,
Boolean.FALSE);
        // set a customer header
        inMessage.setHeader("key", "value");
        // setup the accept content type
        inMessage.setHeader(Exchange.ACCEPT_CONTENT_TYPE, "application/json");
        // set the parameters , if you just have one parameter
        // camel will put this object into an Object[] itself
        inMessage.setBody("123");
    }
});

// get the response message
Customer response = (Customer) exchange.getOut().getBody();

assertNotNull("The response should not be null ", response);
assertEquals("Get a wrong customer id ", 123, response.getId());
assertEquals("Get a wrong customer name", "John", response.getName());
assertEquals("Get a wrong response code", 200,
exchange.getOut().getHeader(Exchange.HTTP_RESPONSE_CODE));
assertEquals("Get a wrong header value", "value", exchange.getOut().getHeader("key"));
```

CXF JAXRS 前端还提供以 http 为中心的客户端 API。您也可以从 camel-cxfrs 生成者调用此 API。您需要指定 HTTP_PATH 和 HTTP_METHOD，并使用 URI 选项 httpClientAPI 或设置消息标头 CxfConstants.CAMEL_CXF_RS_USING_HTTP_API 使用 http 中心客户端 API。您可以将响应对象转换为使用消息标头 CxfConstants.CAMEL_CXF_RS_RESPONSE_CLASS 指定的类型类。

```
Exchange exchange = template.send("direct://http", new Processor() {
    public void process(Exchange exchange) throws Exception {
        exchange.setPattern(ExchangePattern.InOut)
        Message inMessage = exchange.getIn();
        // using the http central client API
        inMessage.setHeader(CxfConstants.CAMEL_CXF_RS_USING_HTTP_API, Boolean.TRUE);
        // set the Http method
        inMessage.setHeader(Exchange.HTTP_METHOD, "GET");
        // set the relative path
        inMessage.setHeader(Exchange.HTTP_PATH, "/customerservice/customers/123");
        // Specify the response class , cxfrs will use InputStream as the response object type
        inMessage.setHeader(CxfConstants.CAMEL_CXF_RS_RESPONSE_CLASS,
```

```
Customer.class);
        // set a customer header
        inMessage.setHeader("key", "value");
        // since we use the Get method, so we don't need to set the message body
        inMessage.setBody(null);
    }
});
```

在 Camel 2.1 中，我们还支持为 CXFRS http 以 http 中心的客户端指定来自 cxfrs URI 的查询参数。

```
Exchange exchange = template.send("cxfrs://http://localhost:9003/testQuery?
httpClientAPI=true&q1=12&q2=13"
```

要支持 Dynamical 路由，您可以使用 CxfConstants.CAMEL_CXF_RS_QUERY_MAP 标头覆盖 URI 的查询参数，以为其设置参数映射。

```
Map<String, String> queryMap = new LinkedHashMap<>();
queryMap.put("q1", "new");
queryMap.put("q2", "world");
inMessage.setHeader(CxfConstants.CAMEL_CXF_RS_QUERY_MAP, queryMap);
```

79.9. CXF 的 CAMEL 传输是什么

在 CXF 中，您通过定义其地址来提供或消耗 web 服务。地址的第一个部分指定要使用的协议。例如，端点配置中 address="http://localhost:9000" 意味着您的服务将使用 http 协议在 localhost 的端口 9000 上提供。当您将 Camel Tranport 集成到 CXF 中时，您将获得新的传输"camel"。因此，您可以指定 address="camel://direct:MyEndpointName" 将 CXF 服务地址绑定到 camel 直接端点。

从技术上讲，对于 CXF 的 Camel 传输是一个使用 Camel 内核库实现 CXF 传输 API 的组件。这可让您与 CXF 服务一起轻松使用 Camel 的路由引擎和集成模式支持。

79.10. 将 CAMEL 集成到 CXF 传输层

要将 Camel Tranport 包含在使用 CamelTransportFactory 的 CXF 总线中。您可以在 Java 和 Spring 中执行此操作。

79.10.1. 在 Spring 中设置 Camel 传输

如果要配置任何特殊配置，您可以在 applicationcontext 中使用以下代码片段。如果您只想激活 camel 传输，则不必在应用程序上下文中进行任何操作。当您在应用程序中包含 camel-cxf-transport jar（或 camel-cxf.jar 小于 2.7.x）后，cxf 将扫描 jar 并为您加载 CamelTransportFactory。

```
<!-- you don't need to specify the CamelTransportFactory configuration as it is auto load by
CXF bus -->
<bean class="org.apache.camel.component.cxf.transport.CamelTransportFactory">
  <property name="bus" ref="cxf" />
  <property name="camelContext" ref="camelContext" />
  <!-- checkException new added in Camel 2.1 and Camel 1.6.2 -->
  <!-- If checkException is true , CamelDestination will check the outMessage's
     exception and set it into camel exchange. You can also override this value
     in CamelDestination's configuration. The default value is false.
     This option should be set true when you want to leverage the camel's error
     handler to deal with fault message -->
  <property name="checkException" value="true" />
  <property name="transportIds">
    <list>
      <value>http://cxf.apache.org/transports/camel</value>
    </list>
  </property>
</bean>
```

79.10.2. 以编程方式集成 Camel 传输

Camel 传输提供了一个 setContext 方法，可用于将 Camel 上下文设置为传输工厂。如果您希望此工厂生效，您需要将工厂注册到 CXF 总线中。以下是您的完整示例：

```
import org.apache.cxf.Bus;
import org.apache.cxf.BusFactory;
import org.apache.cxf.transport.ConduitInitiatorManager;
import org.apache.cxf.transport.DestinationFactoryManager;
...

BusFactory bf = BusFactory.newInstance();
Bus bus = bf.createBus();
CamelTransportFactory camelTransportFactory = new CamelTransportFactory();
// set up the CamelContext which will be use by the CamelTransportFactory
camelTransportFactory.setCamelContext(context)
// if you are using CXF higher then 2.4.x the
camelTransportFactory.setBus(bus);

// if you are lower CXF, you need to register the ConduitInitiatorManager and
DestinationFactoryManager like below
// register the conduit initiator
ConduitInitiatorManager cim = bus.getExtension(ConduitInitiatorManager.class);
cim.registerConduitInitiator(CamelTransportFactory.TRANSPORT_ID,
camelTransportFactory);
// register the destination factory
DestinationFactoryManager dfm = bus.getExtension(DestinationFactoryManager.class);
dfm.registerDestinationFactory(CamelTransportFactory.TRANSPORT_ID,
camelTransportFactory);
// set or bus as the default bus for cxf
BusFactory.setDefaultBus(bus);
```

79.11. 使用 SPRING 配置 DESTINATION 和 CONDUIT

79.11.1. Namespace

用于配置 Camel 传输端点的元素在命名空间 http://cxf.apache.org/transports/camel 中定义。它通常被称为前缀 camel。要使用 Camel 传输配置元素，您需要将下面显示的行添加到端点配置文件的 Bean 元素中。另外，您需要将配置元素的命名空间添加到 xsi:schemaLocation 属性中。

添加配置命名空间

```
<beans ...
      xmlns:camel="http://cxf.apache.org/transports/camel
      ...
      xsi:schemaLocation="...
                          http://cxf.apache.org/transports/camel
                          http://cxf.apache.org/transports/camel.xsd
                          ...>
```

79.11.2. destination 元素

您可以使用 camel:destination 元素及其子项配置 Camel 传输服务器端点。camel:destination 元素采用单个属性 name，它指定与端点对应的 WSDL 端口元素。name 属性的值采用 portQName'.camel-destination' 形式。以下示例显示 camel:destination 元素，它将用于为 WSDL 片段 < port binding="widgetSOAPBinding" name="widgetSOAPPort">如果端点的 目标命名空间是 http://widgets.widgetvendor.net 指定的端点添加配置。

camel:destination Element

```
...
  <camel:destination name="{http://widgets/widgetvendor.net}widgetSOAPPort.http-
destination>
    <camelContext id="context" xmlns="http://activemq.apache.org/camel/schema/spring">
        <route>
          <from uri="direct:EndpointC" />
          <to uri="direct:EndpointD" />
        </route>
    </camelContext>
  </camel:destination>

  <!-- new added feature since Camel 2.11.x
  <camel:destination name="{http://widgets/widgetvendor.net}widgetSOAPPort.camel-
destination" camelContextId="context" />

...
```

Spring 的 camel:destination 元素有多个指定配置信息的子元素。它们如下所述。

元素

描述

camel-spring:camelContext

您可以在 camel 目的地中指定 camel 上下文

camel:camelContextRef

要注入 camel 目的地的 camel 上下文 ID

79.11.3. conduit 元素

您可以使用 camel:conduit 元素及其子项配置 Camel 传输客户端。camel:conduit 元素采用单个属性 name，它指定与端点对应的 WSDL 端口元素。name 属性的值采用 portQName'.camel-conduit' 形式。例如，下面的代码显示 camel:conduit 元素，它用于为 WSDL 片段 < port binding="widgetSOAPBinding" name="widgetSOAPPort "> （如果端点的目标命名空间是 http://widgets.widgetvendor.net ）指定的端点配置。

http-conf:conduit Element

```
...
 <camelContext id="conduit_context" xmlns="http://activemq.apache.org/camel/schema/spring">
     <route>
         <from uri="direct:EndpointA" />
         <to uri="direct:EndpointB" />
     </route>
 </camelContext>

 <camel:conduit name="{http://widgets/widgetvendor.net}widgetSOAPPort.camel-conduit">
   <camel:camelContextRef>conduit_context</camel:camelContextRef>
 </camel:conduit>

 <!-- new added feature since Camel 2.11.x
 <camel:conduit name="{http://widgets/widgetvendor.net}widgetSOAPPort.camel-conduit"
```

```
camelContextId="conduit_context" />


  <camel:conduit name="*.camel-conduit">
  <!-- you can also using the wild card to specify the camel-conduit that you want to configure
-->
    ...
  </camel:conduit>
...
```

camel:conduit 元素有多个指定配置信息的子元素。它们如下所述。

元素	描述
camel-spring:camelContext	您可以在 camel conduit 中指定 camel 上下文
camel:camelContextRef	要注入 camel conduit 的 camel 上下文 ID

79.12. 使用蓝图配置 DESTINATION 和 CONDUIT

从 Camel 2.11.x 中，支持 Camel 传输配置蓝图。

如果使用蓝图，您应该使用命名空间 http://cxf.apache.org/transports/camel/blueprint 并导入类似 blow 的 schema。

为蓝图添加配置命名空间

```
<beans ...
      xmlns:camel="http://cxf.apache.org/transports/camel/blueprint"
      ...
```

```
    xsi:schemaLocation="...
                http://cxf.apache.org/transports/camel/blueprint
                http://cxf.apache.org/schmemas/blueprint/camel.xsd
               ...>
```

在蓝图 camel:conduit camel:destination 中只有一个 camelContextId 属性，它们不支持在 camel 目的地中指定 camel 上下文。

```
<camel:conduit id="*.camel-conduit" camelContextId="camel1" />
<camel:destination id="*.camel-destination" camelContextId="camel1" />
```

79.13. 使用 CAMEL 作为 CXF 负载均衡器的示例

本例演示了如何在 CXF 中使用 camel 负载均衡功能。您需要在 CXF 中加载配置文件，并在地址 "camel://direct:EndpointA" 和 "camel://direct:EndpointB" 上发布端点

79.14. 将 CAMEL 附加到 CXF 的完整 HOWTO 和 EXAMPLE

使用 Apache Camel 为 CXF Webservice 更好地 JMS 传输

第 80 章 数据格式组件

从 Camel 版本 2.12 开始提供

dataformat: 组件允许使用 数据格式作为 Camel 组件。

80.1. URI 格式

```
dataformat:name:(marshal|unmarshal)[?options]
```

其中 name 是数据格式的名称。然后后根一个操作，需要是 marshal 或 unmarshal。这些选项用于配置 使用的数据格式。有关它支持的选项，请参阅数据格式文档。

80.2. DATAFORMAT 选项

Data Format 组件没有选项。

Data Format 端点使用 URI 语法进行配置：

```
dataformat:name:operation
```

使用以下路径和查询参数：

80.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
name	数据格式 **必需** 名称		字符串
operation	**必需** Operation 使用 marshal 或 unmarshal		字符串

80.2.2. 查询参数(1 参数)：

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

Name	描述	默认值	类型

80.3. SAMPLES

例如，要使用 JAXB 数据格式，我们可以按如下方式执行：

```
from("activemq:My.Queue").
  to("dataformat:jaxb:unmarshal?contextPath=com.acme.model").
  to("mqseries:Another.Queue");
```

在 XML DSL 中，您可以进行：

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="activemq:My.Queue"/>
    <to uri="dataformat:jaxb:unmarshal?contextPath=com.acme.model"/>
    <to uri="mqseries:Another.Queue"/>
  </route>
</camelContext>
```

第 81 章 DATASET 组件

从 Camel 版本 1.3 开始提供

测试分布式和异步处理非常困难。Mock、Test 和 DataSet 端点与 Camel 测试框架协同工作，从而通过使用 企业集成模式 和 Camel 的大量组件以及强大的 Bean 集成来简化您的单元和集成测试。

DataSet 组件提供了一种可轻松执行系统的负载和 soak 测试的机制。它的工作原理是允许您创建 DataSet 实例作为 消息来源，并作为接收数据集的方式。

Camel 将在发送 数据集时使用吞吐量日志记录器。

81.1. URI 格式

```
dataset:name[?options]
```

其中 name 用于查找 Registry 中的 DataSet 实例

Camel 附带了一个 org.apache.camel.component.dataset.DataSet (org.apache.camel.component.dataset.DataSetSupport 类)的支持实现，它可用作实施您自己的 DataSet 的基础。Camel 还附带一些可用于测试的实现： org.apache.camel.component.dataset.SimpleDataSet,org.apache.camel.component.dataset.ListDataSet 和 org.apache.camel.component.dataset.FileDataSet，所有扩展 DataSetSupport。

81.2. 选项

Dataset 组件没有选项。

Dataset 端点使用 URI 语法进行配置：

```
dataset:name
```

使用以下路径和查询参数：

81.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	在 registry 中查找的 DataSet **必需** 的名称		DataSet

81.2.2. 查询参数(19 参数)：

Name	描述	默认值	类型
dataSetIndex (common)	控制 CamelDataSetIndex 标头的行为。对于 Consumers: - off = 标头不会被设置 - strict/lenient = 将设置标头。对于 Producers: - off = 不会被验证，如果标头值不存在，则不会设置它，如果标头值不存在，它将会被验证 = lenient = 它会验证标头值（如果不存在），则不会设置它，如果不存在，则不会设置标头值。	lenient	字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外，该处理程序记录在 WARN/ERROR 级别并忽略。	false	布尔值
initialDelay (consumer)	开始发送消息前的 millis 中等待的时间。	1000	long
minRate (consumer)	等待 DataSet 至少包含这个数量的信息	0	int
preloadSize (consumer)	设置在路由完成初始化前预加载(sent)多少消息	0	long
produceDelay (consumer)	允许指定延迟，它会在消费者发送消息时造成延迟（以模拟缓慢的处理）	3	long
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，该例外记录在 WARN/ERROR 级别并忽略。		ExceptionHandler
exchangePattern (consumer)	在创建交换时设置默认交换模式。		ExchangePattern

Name	描述	默认值	类型
assertPeriod (producer)	设置一个宽限期，之后 mock 端点将重新断言，以确保初始断言仍然有效。例如，这只用于表示多个消息到达的 assert。例如，如果 expectedMessageCount (int)设为 5，则当 5 个或更多消息到达时，会满足断言。为确保准确 5 个消息到达，您需要稍等片刻，以确保进一步的消息到达。这是您可以对此使用 setAssertPeriod （长）方法的内容。默认情况下禁用这个周期。	0	long
consumeDelay (producer)	允许指定延迟，这会导致生成者使用消息时造成延迟（以模拟缓慢的处理）	0	long
expectedCount (producer)	指定此端点应接收的消息交换数量。注意：如果要预期 0 个消息，请在测试启动时做额外的小心，以 0 匹配，因此您需要设置一个 assert 周期时间以便让测试在一段时间内运行，以确保仍然没有消息到达；对于 setAssertPeriod （长）。另一种方法是使用 NotifyBuilder，并在调用 mocks 上的 assertIsSatisfied （）方法前，使用 notifier 知道 Camel 何时进行路由。这可让您不使用固定的 assert 周期来加快测试时间。如果您要断言，其中第 n 个消息到达这个 mock 端点，则还要查看 setAssertPeriod (long)方法了解更多详情。	-1	int
reportGroup (producer)	用于根据大小组打开吞吐量日志记录的数字。		int
resultMinimumWaitTime (producer)	设置 assertIsSatisfied （）的最小预期时间（单位为 millis）将等待 latch，直到它满足为止	0	long
resultWaitTime (producer)	设置 assertIsSatisfied （）将等待的最大时间量，直到它满足为止	0	long
retainFirst (producer)	指定只保留收到的前 n 个接收交换数。这在测试大数据时，通过不存储此模拟端点接收的每一个交换的副本来减少内存消耗。重要： 使用此限制时，getReceivedCounter （）仍然会返回接收的交换的实际数量。例如，如果我们收到 5000 Exchanges，并且配置为仅保留前 10 个交换，那么 getReceivedCounter （） 仍然会返回 5000，但 getExchanges （）中只有前 10 Exchanges （） 和 getReceivedExchanges （）方法。使用此方法时，不支持其他一些预期的方法，例如 expectedBodiesReceived (Object...)在收到的第一个正文数上设置预期。您可以配置 setRetainFirst (int)和 setRetainLast (int)方法，以限制第一个和最后一个收到。	-1	int

Name	描述	默认值	类型
retainLast (producer)	指定只保留最后第 n 个接收交换的数量。这在测试大数据时，通过不存储此模拟端点接收的每一个交换的副本来减少内存消耗。重要： 使用此限制时，getReceivedCounter （）仍然会返回接收的交换的实际数量。例如，如果我们收到 5000 Exchanges，并且配置为仅保留最后 20 个交换，那么 getReceivedCounter （） 仍然会返回 5000，但 getExchanges （） 中只有最后 20 个交换程序）和 getReceivedExchanges （）方法。使用此方法时，不支持其他一些预期的方法，例如 expectedBodiesReceived (Object...)在收到的第一个正文数上设置预期。您可以配置 setRetainFirst (int)和 setRetainLast (int)方法，以限制第一个和最后一个收到。	-1	int
sleepForEmptyTest (producer)	使用零调用 expectedMessageCount (int)时，允许指定 sleep 来检查此端点是否确实为空	0	long
copyOnExchange (producer)	设置在这个模拟端点收到传入 Exchange 时是否进行深度副本。默认为 true。	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

81.3. 配置 DATASET

Camel 将在 Registry 中查找实施 DataSet 接口的 bean。以便您可以将自己的 DataSet 注册为：

```
<bean id="myDataSet" class="com.mycompany.MyDataSet">
  <property name="size" value="100"/>
</bean>
```

81.4. 示例

例如，要测试一组信息是否发送到队列，然后从队列使用，而不丢失任何消息：

```
// send the dataset to a queue
from("dataset:foo").to("activemq:SomeQueue");
```

```
// now lets test that the messages are consumed correctly
from("activemq:SomeQueue").to("dataset:foo");
```

以上命令会在 Registry 中查找用于创建消息的 foo DataSet 实例。

然后，您可以创建一个 DataSet 实现，如使用 SimpleDataSet，配置如下内容，如数据集的大型程度以及消息类似等。

81.5. DATASETSUPPORT (ABSTRACT 类)

DataSetSupport abstract 类是新 DataSets 的良好起点，为派生类提供了一些有用的功能。

81.5.1. DataSetSupport 的属性

属性	类型	Default（默认）	描述
defaultHeaders	**Map<String,Object>**	**null**	指定默认消息正文。对于 SimpleDataSet，它是一个恒定的有效负载，但如果要为每个消息创建自定义有效负载，请创建自己的 **DataSetSupport**ation。
outputTransformer	**org.apache.camel.Processor**	null	
size	**long**	**10**	指定要发送/恢复多少个消息。
reportCount	**long**	**-1**	指定报告进度前要接收的消息数量。可用于显示大型负载测试的进度。如果 < 0，则 **大小** / 5，如果为 0，则 **大小**，否则设置为 **reportCount** 值。

81.6. SIMPLEDATASET

SimpleDataSet 扩展 DataSetSupport，并添加默认正文。

81.6.1. SimpleDataSet 上的附加属性

属性	类型	Default（默认）	描述
defaultBody	对象	**<hello>world!</hello>**	指定默认消息正文。默认情况下，**SimpleDataSet** 为每个交换生成相同的常量有效负载。如果要为每个交换自定义有效负载，创建一个 Camel **Processor**，通过设置 **outputTransformer** 属性将 **SimpleDataSet** 配置为使用它。

81.7. LISTDATASET

自 Camel 2.17 起可用

List'DataSet' 扩展 DataSetSupport，并添加默认正文列表。

81.7.1. ListDataSet 上的额外属性

属性	类型	Default（默认）	描述
defaultBodies	**List<Object>**	**空 LinkedList<Object>**	指定默认消息正文。默认情况下，**ListDataSet** 使用 **CamelDataSetIndex** 从 **defaultBodies** 列表中选择一个恒定有效负载。如果要自定义有效负载，创建一个 Camel **Processor**，并通过设置 **outputTransformer** 属性将 **ListDataSet** 配置为使用它。
size	**long**	defaultBodies 列表的大小	指定要发送/恢复多少个消息。这个值可能与 **defaultBodies** 列表的大小不同。如果该值小于 **defaultBodies** 列表的大小，则不会使用一些列表元素。如果值大于 **defaultBodies** 列表的大小，则会使用 CamelDataSetIndex 的 modulus 以及 **defaultBodies** 列表的大小（如 **CamelDataSetIndex % defaultBodies.size （）** ）来选择交换的有效负载。

81.8. FILEDATASET

自 Camel 2.17 起可用

FileDataSet 扩展 ListDataSet，并添加从文件中加载正文的支持。

81.8.1. FileDataSet 中的其他属性

属性	类型	Default（默认）	描述
sourceFile	**File**	null	指定有效负载的源文件
delimiter	**字符串**	\z	指定 **java.util.Scanner** 用来将文件分成多个有效负载的分隔符模式。

第 82 章 DIGITALOCEAN COMPONENT

从 Camel 版本 2.19 开始提供

DigitalOcean 组件允许您通过封装 [digitalocean-api-java](https://www.digitalocean.com/community/projects/api-client-in-java)管理 DigitalOcean 云中的 Droplets 和资源。您在 DigitalOcean 控制面板中熟悉的所有功能都可通过此 Camel 组件获得。

82.1. 先决条件

您必须具有有效的 DigitalOcean 帐户和有效的 OAuth 令牌。您可以通过访问您帐户的 DigitalOcean 控制面板的 [Apps & API](https://cloud.digitalocean.com/settings/applications)部分来生成 OAuth 令牌。

82.2. URI 格式

DigitalOcean 组件 使用以下 URI 格式：

```
digitalocean://endpoint?[options]
```

其中 endpoint 是 DigitalOcean 资源类型。

示例：列出您的 droplets：

```
digitalocean://droplets?operation=list&oAuthToken=XXXXXX&page=1&perPage=10
```

DigitalOcean 组件只支持生成者端点，因此您不能在路由开始时使用这个组件来侦听频道中的消息。

82.3. 选项

DigitalOcean 组件没有选项。

DigitalOcean 端点使用 URI 语法进行配置：

digitalocean:operation

使用以下路径和查询参数：

82.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
operation	对给定资源执行的操作。		DigitalOceanOperations

82.3.2. 查询参数(10 parameters):

Name	描述	默认值	类型
page (producer)	使用 进行分页。强制页面号。	1	整数
perPage (producer)	使用 进行分页。设置每个请求的项目数量。每个页面的最大结果数为 200。	25	整数
resource (producer)	**必需** 执行操作的 DigitalOcean 资源类型。		DigitalOceanResources
digitalOceanClient (advanced)	使用现有配置的 DigitalOceanClient 作为客户端		DigitalOceanClient
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
httpProxyHost (proxy)	如果需要，设置代理主机		字符串
httpProxyPassword (proxy)	如果需要，设置代理密码		字符串
httpProxyPort (proxy)	如果需要，设置代理端口		整数
httpProxyUser (proxy)	如果需要，设置代理主机		字符串
oAuthToken (security)	DigitalOcean OAuth 令牌		字符串

82.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.digitalocean.enabled	启用 digitalocean 组件	true	布尔值
camel.component.digitalocean.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

您必须为每个端点提供操作值，以及 operation URI 选项或 CamelDigitalOceanOperation 消息标头。

所有 操作值 都在 DigitalOceanOperations enumeration 中定义。

组件使用的所有 标头名称 都在 DigitalOceanHeaders enumeration 中定义。

82.5. 消息正文结果

返回的所有消息正文都使用 digitalocean-api-java 库提供的对象。

82.6. API 速率限制

由 camel-digitalocean 组件封装的 DigitalOcean REST API 受到 API 速率限制的影响。您可以在 [API Rate Limits 文档](https://developers.digitalocean.com/documentation/v2/#rate-limit)中找到每个方法限制。

82.7. 帐户端点

| operation | Description | Headers | Result | | ------ | ---- | ------- | ---------- | | get | get account info | | com.myjeeva.digitalocean.pojo.Account |

82.8. BLOCKSTORAGES 端点

*| operation | Description | Result | Result | | ------ | ---- | ------- | ---------- | | list | list of the Block Storage volumes | | List<com.myjeeva.digitalocean.pojo.Volume > | | get | show information about a Block Storage volume| CamelDigitalOceanId Integer | com.myjeeva.digitalocean.pojo. Volume | | | get | show about a Block Storage volume by name | CamelDigitalOcean Name String
'CamelDigitalOceanRegion 'String | com.myjeeva.digitalocean.pojo. Volume | | list Snapshots | 检索从卷创建的快照 | CamelDigitalOceanId Integer | List<com.myjeeva.digitalocean.pojo.Snapshot> | | create | create a new volume | CamelDigitalO ceanVolumeSizeGigabytesInteger
'CamelDigitalOcean Name' String
'CamelDigitalOcean Description'* String
'CamelDigitalOcean Region' * String | com.myjeeva.digitalocean. pojo.Volume | | delete | delete a Block Storage volume , 销毁所有数据并从 您的帐户中删除它| CamelDigital Oce anId Integer | com.myjeeva.digitalocean.pojo.Delete | | delete | delete a Block Storage volume by name| CamelDigital OceanName String
'CamelDigitalOce ananRegion' String | com.myjeeva.digitaloce.poean.pojoe.Delete | Attach | 将 块存储卷附加到 Droplet| Camel Digital OceanId Integer
'CamelDigitalOce anDropletId' Integer
'CamelDigitalOce anDrop letRegion' String | com.myjeeva.digitaloce an.pojo.Action | attach | attach a Block Storage volume by name| CamelDigitalOceanName String
'CamelDigitalO ceanDropletId' Integer
'CamelDigitalO cean DropletRegion' String | com.myjeeva.digitaloc e an.pojo.Action | detach | detach a Block Storage volume from a Droplet | CamelDigitalOceanId Integer
'CamelDigital OceanDropletId' Integer
'CamelDigital Oce anDropletRegion' String | com.myjeeva.digital oc e an.pojo.Action | attach | detach a Block Storage volume by name| CamelDigitalOceanName String
'Camel DigitalOceanDropletId' Integer
'Camel DigitalO ceanDropletRegion' String | com.myjeeva.digit al oc ean.pojo.Action | resize a Block Storage volume | CamelDigitalOceanVolumeSizeGigabytes Integer <br&gt ;' CamelDigitalOceanRegion' String | com.myjeeva. digital oce an.pojo.Action | | listActions | 检索在 卷上执行的所有操作 | CamelDigitalOceanId Integer | List<com.myjeeva.digitalocean.pojo.Action> |*

82.9. DROPLETS 端点

*| operation | Description | Result | Result | | ------ | ---- | ------- | ---------- | | list | list all Droplets in your account | | List< com.myjeeva.digitalocean.pojo.Droplet > | | get | show a individual droplet | CamelDigitalOceanId Integer | com.myjeeva.digitalocean.pojo.Drop let | | create | create a new Droplet | CamelDigitalOcean Name String
'CamelDigitalOceanDropletImage ' String
'CamelDigitalOceanRegion ' String
'CamelDigitalOceanDropletSize ' String
'CamelDigitalOceanDropletSSHKeys '* List\<String\>
'CamelDigitalOceanDropletEnableBackups ' * Boolean
'CamelDigitalOceanDropletEnableIpv6 ' * Boolean
'CamelDigitalOceanDropletEnablePrivateNetworking'* 布尔值
'CamelDigitalOceanDroplet UserData '* String
'CamelDigitalOceanDroplet Volumes'* List\<String\>
'CamelDigitalOceanDrop letTags' List\<String\> | com.myjeeva.digitalocean.pojo.Droplet | create | create multiple Droplets | CamelDigitalO ceanNames List \<String\>
'CamelDigitalOcean Drop letImage' String
'CamelDigitalO ceanRegion' String
'CamelDigitalOceanDropletSize' String
'CamelDigitalOceanDrop letKeys'] List\<String\>
'CamelDigitalOceanDrop let EnableBackups'* Boolean
'CamelDigitalOceanDropletEnableIpv6'* Boolean
'CamelDigitalOceanDropletEnablePrivateNetworking'* Boolean
'CamelDigitalOcean DropletUserData'* String
'CamelDigitalOceanDropletVolumes'* List\<String\>
'CamelDigitalOceanDropletTags' List\<String\> | com.myjeeva.digitalocean.pojo.Droplet | | delete | delete a Droplet, | CamelDigitalOceanId Integer | com.myjeeva.digitalocean.pojo.Delete | enableBackups | enable backup on an existing Droplet | Camel DigitalOceanId Integer | com.myjeeva.digital ocean.pojo.Action | | disableBackups | 禁用现有 Droplet 上的 备份| CamelDigitalOceanId Integer | com.myjeeva.digit alocean.pojo.Action | | enableIpv6 | enable Ipv6 | 在现有 Drop let| CamelDigitalOceanId Integer |*

*com.myjeeva.digitalocean.pojoean.pojoean.pojoean | enablePrivateNetworking | 在现有 Droplet | CamelDigitalOceanId Integer | com.myjeeva. digitalocean.pojo. Action | reboot | reboot a Droplet | CamelDigitalOceanId Integer | com.myjeeva. digital oce an.pojo.Action | | powerCycle | Power cycle aDrop let | CamelDigitalOceanId Integer | com.myjeeva.digitalocean.pojo .Action | shutdown | shutdown aDrop let | CamelDigitalOceanId Integer | com.myjeeva.digit al ocean.pojo.Action | powerOff | power offa Droplet | CamelDigitalOceanId Integer | com.myjee va.digitalocean.pojo.Action | | powerOn | power on a Droplet | CamelDigitalOceanId Integer | com.myjeeva . digitalocean.pojo.Action | restore |shutdown a Droplet | CamelDigitalOceanId Integer
'CamelDigitalOceanImageId' Integer | com.myjee va. digitalocean.pojo.Action | passwordReset | reset the password for a Droplet | CamelDigitalOceanId Integer | com.myjeeva.digitalocean.pojo.Action | resize | resize a Droplet | CamelDigitalOceanId Integer
'CamelDigitalOceanDropletSize' String | com.myjeeva.digitalocean.pojo.Action | | rebuild | rebuild a Droplet | CamelDigitalOceanId Integer &lt ; br>'CamelDigitalOceanImageId' Integer | com.myjeeva.digitalocean.pojo.Action | | rename | rename a Droplet | CamelDigitalOceanId Integer & lt;br>'CamelDigitalOceanName' String | com.myjeeva.digitalocean.pojo. Action| changeKernel | 更改 Droplet | CamelDigitalOceanId Integer & lt;br>'CamelDigitalOceanKernelId' Integer | com .my jeeva.digitalocean.pojo.Action | takeSnapshot | snapshot a Droplet | CamelDigitalOceanId Integer
'CamelDigitalOceanName'* String | com.myjeeva.digitalocean.pojo. Action | | tag | tag a Droplet | CamelDigitalOceanId Integer
'CamelDigitalOceanName' String | com.myjeeva.digitalocean.pojo.Response | | untag | untag a Droplet | CamelDigitalOcean IdInteger
'CamelDigitalOceanName' String | com.myjeeva.digitalocean.pojo.Response | | listKernel s |检索 Droplet 所有可用内核的列表 | CamelDigitalOceanId Integer | List <com. myjeeva.digitalocean.pojo .Kernel> | | list Snapshots | 检索从 Droplet 创建的快照 | CamelDigitalOceanId Integer | List<com .my jeeva.digitaloce an.pojo.Snapshot& gt; | | listBackups | 检索与 Droplet 关联的备份 | CamelDigitalOceanId Integer | List<com .my jeeva.digitalocean. pojo.Backup> | | listActions | 检索所有操作 这已在 Droplet 上执行 | CamelDigitalOceanId Integer | List< com.myjeeva.digitalocean.pojo .Action> | | list Neighbors | 检索在同一物理服务器上运行的丢弃程序列表 | CamelDigitalOceanId Integer | List< com.myjeeva. digitalocean.pojo.Droplet > | | listAllNeighbors | 检索在同一物理硬件上运行的任何 droplets 列表 | | List<com.myjeeva.myjeeva.digitalocean.pojo.Droplet> |*

82.10. 镜像端点

| operation | Description | Result | Result | | ------ | ---- | ------- | ----------- | | list | list images on your account | CamelDigitalOceanType DigitalOceanImageTypes | List< com.myjeeva.digitalocean.pojo.Image > | ownList | retrieve 仅用户的专用镜像 | | List<com.myjeeva.digitalocean.pojo. Image> | | list Actions | 检索在镜像上执行的所有操作 | CamelDigitalOcean Id Integer | List<com.myjeeva.digitalocean.pojo. Action> | get | get | retrieve a image (public or private) by id| CamelDigitalOce anId Integer | com.myjeeva.digitalocean.pojo.Image | get | get | retrieve about an public image by slug| CamelDigitalOceanDrop letImage String | com.myjeeva.digitalocean.pojoean.pojo.Image . | update | update a image| CamelDigitalO ce anId Integer
'CamelDigitalOce anName ' String | com.myjeeva.digitalocean. pojo. Image | | delete | delete a image| CamelDigitalO ce an Id Integer | com.myjeeva.digitalocean. pojo . Delete | transfer | transfer a image to another region| CamelDigital O ceanId Integer
'CamelDigitalO cean Region' String | com.myjeeva.digitalocean.pojo . Action | | convert | convert a image, 例如， 备份快照| Camel Digital O ceanId Integer | com.myjeeva.digitaloce an.pojo.Action |*

82.11. 快照端点

| operation | Description | Result | Result | | ------ | ---- | ------- | ----------- | | list | list of the

account | CamelDigitalOceanType DigitalOceanSnapshotTypes | List<com.myjeeva.digitalocean.pojo.Snapshot > | get |检索有关快照的信息 | CamelDigitalOcean IdInteger | com.myjeeva.digitalocean.pojo. Snapshot | | delete | delete an snapshot | CamelDigitalOcean Id Integer | com.myjeeva.digitalocean.pojo. Delete |*

82.12. KEY ENDPOINT

*| operation | Description | Result | Result | | ------ | ---- | ------- | ---------- | | list | list of the account | List<com.myjeeva.digitalocean.pojo.Key > | | get | retrieve a key by id| CamelDigitalOceanId Integer | com.myjeeva.digitalocean.pojo. Key | | get | retrieve a key by fingerprint | CamelDigitalOceanKeyFinger printString | com.myjeeva.digitalocean.pojo. Key | | update | update a key by id | CamelDigitalOcean Id Integer
'CamelDigitalOcean Name' String | com.myjeeva.digitalocean.pojo.Key | | update | update a key by fingerprint| CamelDigitalOceanKeyFi nger print String
'CamelDigitalOcean Name' String | com.myjeeva.digitalocean.pojo. Key | delete | delete a key by id| CamelDigitalO ce an Id Integer | com.myjeeva.digitalocean. pojo. Delete | | delete | delete a key by fingerprint| CamelDigitalOceanKey Fi nger print String | com.myjeeva.digitalocean. pojo.Delete |*

82.13. 区域端点

| operation | Description | Result | Result | | ------ | ---- | ------- | ---------- | | list | list of available | List<com.myjeeva.digitalocean.pojo.Region > |

82.14. 大小端点

| operation | Description | Result | Result | | ------ | ---- | ------- | ---------- | | list | list of available size of available | List<com.myjeeva.digitalocean.pojo.Size > |

82.15. 浮动 IP 端点

*| operation | Description | Result | Result | | ------ | ---- | ------- | ---------- | | list | list of the a Droplet available the all of the list< com.myjeeva.digitalocean.pojo.FloatingIP > | create | create | create a new Floating IP assigned to a Droplet | CamelDigitalOcean Id Integer | List<com.myjeeva.digitalocean.pojo.Floating IP> | | create | create | create a new Floating IP assigned to a Region | CamelDigitalO ceanRegion String | List<com.myjeeva.digitalocean. pojo.FloatingIP> | get | 检索有关浮动 IP 的信息| CamelDigitalOcean FloatingIPAddress String | com.myjeeva.digitalocean.pojo.Key | | delete | delete a Floating IP，并将它从您的帐户| CamelDigitalOcean FloatingIPAddress String | com.myjeeva.digitalocean.pojoean.pojo.Delete | 将浮动 IP 分配给 Droplet| CamelDigitalOce an FloatingIPAddress String
'CamelDigitalOceanDrop letId ' Integer | com.myjeeva.digitalocean. pojo . Action | unassign | unassign a Floating IP | CamelDigitalOce an Floating IPAddress String | com.myjeeva.digitalocean. pojo. Action | listActions | 检索 在 Floating IP | CamelDigitalO ce an FloatingIPAddress String | List<com.myjeeva.digitalocean.pojo.Action> |*

82.16. 标签端点

| operation | Description | Result | Result | | ------ | ---- | ------- | ---------- | | list | list all of your tags | List<com.myjeeva.digitalocean.pojo.Tag > | create | create a Tag | CamelDigitalOceanName String | com.myjeeva.digitalocean.pojo.Tag | get | 检索单个标签 | CamelDigitalOcean Name String | com.myjeeva.digitalocean.pojo. Tag | | delete | delete a tag | CamelDigitalOcean Name String | com.myjeeva.digitalocean.pojo. Delete | | update | update | update a update a tag | CamelDigitalOce an Name String
'CamelDigitalOceanNew Name' String | com.myjeeva.digitalocean.pojo.Tag |

82.17. 例子

获取您的帐户信息

```
from("direct:getAccountInfo")
    .setHeader(DigitalOceanConstants.OPERATION, constant(DigitalOceanOperations.get))
    .to("digitalocean:account?oAuthToken=XXXXXX")
```

创建一个 droplet

```
from("direct:createDroplet")
    .setHeader(DigitalOceanConstants.OPERATION, constant("create"))
    .setHeader(DigitalOceanHeaders.NAME, constant("myDroplet"))
    .setHeader(DigitalOceanHeaders.REGION, constant("fra1"))
    .setHeader(DigitalOceanHeaders.DROPLET_IMAGE, constant("ubuntu-14-04-x64"))
    .setHeader(DigitalOceanHeaders.DROPLET_SIZE, constant("512mb"))
    .to("digitalocean:droplet?oAuthToken=XXXXXX")
```

列出所有 droplets

```
from("direct:getDroplets")
    .setHeader(DigitalOceanConstants.OPERATION, constant("list"))
    .to("digitalocean:droplets?oAuthToken=XXXXXX")
```

检索 Droplet 的信息(dropletId = 34772987)

```
from("direct:getDroplet")
    .setHeader(DigitalOceanConstants.OPERATION, constant("get"))
    .setHeader(DigitalOceanConstants.ID, 34772987)
    .to("digitalocean:droplet?oAuthToken=XXXXXX")
```

Droplet 的关闭信息(dropletId = 34772987)

```
from("direct:shutdown")
    .setHeader(DigitalOceanConstants.ID, 34772987)
    .to("digitalocean:droplet?operation=shutdown&oAuthToken=XXXXXX")
```

第 83 章 直接组件

从 Camel 版本 1.0 开始提供

direct: 组件在生成者发送消息交换时，提供任何消费者的直接同步调用。
此端点可用于连接 同一 camel 上下文中的现有路由。

提示

异步 的 SEDA 组件在生成者发送消息交换时提供任何消费者的异步调用。

提示

连接到其他 camel 上下文。 VM 组件提供 Camel 上下文之间的连接，只要它们在同一个 JVM 中运行。

83.1. URI 格式

```
direct:someName[?options]
```

其中 someName 可以是任意字符串来唯一标识端点

83.2. 选项

Direct 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
block (producer)	如果向直接端点发送没有活跃消费者的消息，我们可以告诉生成者阻止并等待消费者变为活动状态。	true	布尔值
timeout (producer)	如果启用了块，要使用的超时值。	30000	long
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Direct 端点使用 URI 语法进行配置：

```
direct:name
```

使用以下路径和查询参数：

83.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	**所需** 直接端点的名称		字符串

83.2.2. 查询参数(7 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外，该处理程序记录在 WARN/ERROR 级别并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，该例外记录在 WARN/ERROR 级别并忽略。		ExceptionHandler
exchangePattern (consumer)	在创建交换时设置默认交换模式。		ExchangePattern
block (producer)	如果向直接端点发送没有活跃消费者的消息，我们可以告诉生成者阻止并等待消费者变为活动状态。	true	布尔值
failIfNoConsumers (producer)	当发送到没有活跃用户的 DIRECT 端点时，生成者是否应该会抛出异常。	false	布尔值
timeout (producer)	如果启用了块，要使用的超时值。	30000	long
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

83.3. SAMPLES

在以下路由中，我们使用直接组件将两个路由链接在一起：

```
from("activemq:queue:order.in")
    .to("bean:orderServer?method=validate")
    .to("direct:processOrder");

from("direct:processOrder")
    .to("bean:orderService?method=process")
    .to("activemq:queue:order.out");
```

使用 spring DSL 的示例：

```
<route>
 <from uri="activemq:queue:order.in"/>
 <to uri="bean:orderService?method=validate"/>
 <to uri="direct:processOrder"/>
</route>

<route>
 <from uri="direct:processOrder"/>
 <to uri="bean:orderService?method=process"/>
 <to uri="activemq:queue:order.out"/>
</route>
```

另请参阅 SEDA 组件中的示例，以及如何一起使用它们。

83.4. 另请参阅

- SEDA
- VM

第 84 章 直接虚拟机组件

从 Camel 版本 2.10 开始提供

direct-vm: 组件在生成者发送消息交换时，提供 JVM 中任何消费者的直接、同步调用。
此端点可用于连接同一 camel 上下文中的现有路由，以及来自同一 JVM 中的其他 camel 上下文。

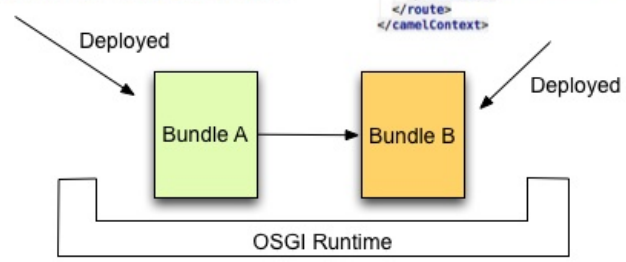
此组件与 Direct 组件不同，即 Direct-VM 支持 CamelContext 实例之间的通信 - 因此，您可以使用此机制在 Web 应用程序之间进行通信（只要 camel-core.jar 位于 system/boot 类路径上）。

在运行时，您可以通过停止现有消费者并启动新用户来在新消费者中进行交换。
但在任何给定时间，对于一个给定端点最多只能有一个活跃的消费者。

此组件还能够连接在不同 OSGI Bundles 中部署的路由，因为您可以在此看到。即使它们在不同捆绑包中运行，camel 路由也会使用
相同的线程。这会自动使用 Transactions - Tx 开发应用程序。

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="timer:foo?period=5000"/>
    <log message=">> Here is a message from direct-vm (bundle1)"/>
    <log message=">> Using ${threadName} of bundle 1"/>
    <to uri="direct-vm:hello"/>
    <log message=">> Response received from bundle 2 : ${body}"/>
  </route>
</camelContext>
```

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="direct-vm:hello"/>
    <log message=">> Prepare response from bundle 2"/>
    <log message=">> Using ${threadName} of bundle 2"/>
    <transform>
        <method ref="helloBean" method="hello"/>
    </transform>
  </route>
</camelContext>
```




```
INFO  | 14 - timer://foo | route7  >> Here is a message from direct-vm (bundle1)
INFO  | 14 - timer://foo | route7  >> Using Camel (89-camel-23) thread #14 - timer://foo of bundle 1
INFO  | 14 - timer://foo | route8  >> Prepare response from bundle 2
INFO  | 14 - timer://foo | route8  >> Using Camel (89-camel-23) thread #14 - timer://foo of bundle 2
INFO  | 14 - timer://foo | route7  >> Response received from bundle 2 : Hi from Camel direct-vm at 2012-06-21 15:21:22
```

84.1. URI 格式

```
direct-vm:someName
```

其中 someName 可以是任意字符串来唯一标识端点

84.2. 选项

Direct VM 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
block (producer)	如果向直接端点发送没有活跃消费者的消息，我们可以告诉生成者阻止并等待消费者变为活动状态。	true	布尔值
timeout (producer)	如果启用了块，要使用的超时值。	30000	long
headerFilterStrategy (advanced)	设置一个 HeaderFilterStrategy，它仅适用于生成者端点（在 directions: request 和 response 中）。默认值：none。		HeaderFilterStrategy
propagateProperties (advanced)	是否从生产者一侧传播属性到消费者，反之亦然。默认值为：true。	true	布尔值
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Direct VM 端点使用 URI 语法进行配置：

```
direct-vm:name
```

使用以下路径和查询参数：

84.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	direct-vm 端点 **必需** 名称		字符串

84.2.2. 查询参数(9 参数)：

Name	描述	默认值	类型

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外，该处理程序记录在 WARN/ERROR 级别并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，该例外记录在 WARN/ERROR 级别并忽略。		ExceptionHandler
exchangePattern (consumer)	在创建交换时设置默认交换模式。		ExchangePattern
block (producer)	如果向直接端点发送没有活跃消费者的消息，我们可以告诉生成者阻止并等待消费者变为活动状态。	true	布尔值
failIfNoConsumers (producer)	当发送到没有活跃用户的 Direct-VM 端点时，生成者是否应该通过抛出异常。	false	布尔值
timeout (producer)	如果启用了块，要使用的超时值。	30000	long
headerFilterStrategy (producer)	设置一个 HeaderFilterStrategy，它仅适用于生成者端点（在 directions: request 和 response 中）。默认值：none。		HeaderFilterStrategy
propagateProperties (advanced)	是否从生产者一侧传播属性到消费者，反之亦然。默认值为：true。	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

84.3. SAMPLES

在以下路由中，我们使用直接组件将两个路由链接在一起：

```
from("activemq:queue:order.in")
    .to("bean:orderServer?method=validate")
    .to("direct-vm:processOrder");
```

现在在另一个 CamelContext 中，如另一个 OSGi 捆绑包。

-

```
from("direct-vm:processOrder")
    .to("bean:orderService?method=process")
    .to("activemq:queue:order.out");
```

使用 spring DSL 的示例：

```
<route>
 <from uri="activemq:queue:order.in"/>
 <to uri="bean:orderService?method=validate"/>
 <to uri="direct-vm:processOrder"/>
</route>

<route>
 <from uri="direct-vm:processOrder"/>
 <to uri="bean:orderService?method=process"/>
 <to uri="activemq:queue:order.out"/>
</route>
```

84.4. 另请参阅

- direct
- SEDA
- VM

第 85 章 DISRUPTOR 组件

从 Camel 版本 2.12 开始提供

disruptor: 组件提供异步 SEDA 行为，但使用 Disruptor 而不是标准 SEDA 使用的 BlockingQueue。或者，a

disruptor-vm：此组件支持端点，提供标准 虚拟机 的替代选择。与 SEDA 组件一样，破坏者的缓冲区：端点仅在 单个 CamelContext 中可见，不提供对持久性或恢复的支持。disruptor-vm: 端点的缓冲区也支持 CamelContexts 实例之间的通信，因此您可以使用此机制在 Web 应用之间进行通信（提供 camel-disruptor.jar 在系统/启动 类路径上）。

选择在生成者和/或多播或并发消费者之间有高竞争时，使用 SEDA 或 VM 组件相比，使用 Disruptor 组件或虚拟机组件的主要优点是性能。在这些情况下，可以看到大量吞吐量并缩短延迟。在没有竞争的情况下的性能与 SEDA 和虚拟机组件相当。

Disruptor 实施，目的是尽可能模拟 SEDA 和 VM 组件的行为和选项。与它们的主要区别如下：

- 使用的缓冲区的大小始终绑定（默认的 1024 个交换）。
- 因为缓冲区总是被滥用，因此 Disruptor 的默认行为是在缓冲区已满而不是抛出异常时阻止。此默认行为可以在组件上配置（请参阅选项）。
- Disruptor enpoints 不实施 BrowsableEndpoint 接口。因此，目前无法在 Disruptor 中检索的交换，只有交换量。
- Disruptor 要求其消费者（多播或其他）需要静态配置。即时添加或删除用户，需要完全刷新 Disruptor 中的所有待处理交换。
- 重新配置结果：通过 Disruptor 发送的数据将直接处理，如果至少有一个消费者，则后接接者仅获得新的交换程序在加入后发布新的交换。
- Disruptor 组件不支持 pollTimeout 选项。

- 当完全 Disruptor 上的生成者块时，它不会响应线程中断。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-disruptor</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

85.1. URI 格式

```
disruptor:someName[?options]
```

or

```
disruptor-vm:someName[?options]
```

其中 someName 可以是在当前 CamelContext 中唯一标识端点的任何字符串（如果 disruptor-vm:），则跨上下文。
您可以使用以下格式将查询选项附加到 URI 中：

```
?option=value&option=value&...
```

85.2. 选项

所有以下选项都对 disruptor: 和 disruptor-vm: 组件都有效。

Disruptor 组件支持 8 个选项，如下所列。

Name	描述	默认值	类型
defaultConcurrent Consumers (consumer)	配置默认的并发消费者数	1	int

Name	描述	默认值	类型
defaultMultipleConsumers (consumer)	为多个消费者配置默认值	false	布尔值
defaultProducerType (producer)	要为 DisruptorProducerType 配置默认值，默认值为 Multi。	multi	DisruptorProducerType
defaultWaitStrategy (consumer)	要为 DisruptorWaitStrategy 配置默认值，默认值为 Blocking。	Blocking	DisruptorWaitStrategy
defaultBlockWhenFull (producer)	当 full 的默认值为 true 时，要为 block 配置默认值。	true	布尔值
queueSize (common)	**弃用** 以配置环缓冲大小		int
bufferSize (common)	配置环缓冲大小	1024	int
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Disruptor 端点使用 URI 语法进行配置：

```
disruptor:name
```

使用以下路径和查询参数：

85.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	**所需** 队列的名称		字符串

85.2.2. 查询参数(12 参数)：

Name	描述	默认值	类型

Name	描述	默认值	类型
size (common)	Disruptors 环缓冲的最大容量将有效增加到最接近 2 的指数。注意：如果使用这个选项，则使用队列名称创建的第一个端点，则决定大小。为确保所有端点都使用相同的大小，请在所有这些端点或创建的第一个端点上配置 size 选项。	1024	int
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
concurrentConsumers (consumer)	并发线程处理交换的数量。	1	int
multipleConsumers (consumer)	指定是否允许多个消费者。如果启用，您可以将 Disruptor 用于 Publish-Subscribe 消息传递。也就是说，您可以向队列发送消息，并让每个使用者收到邮件的副本。启用后，应在每个消费者端点上指定此选项。	false	布尔值
waitStrategy (consumer)	定义消费者线程用于在发布新交换时等待的策略。允许的选项有：Blocking、Sleeping、BusySpin 和 Yielding。	Blocking	DisruptorWaitStrategy
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
blockWhenFull (producer)	将消息发送到完全 Disruptor 的线程将阻止，直到环缓冲区的容量不再耗尽为止。默认情况下，调用线程将阻止并等待消息被接受。禁用这个选项后，会抛出一个例外，表示队列已满。	false	布尔值
producerType (producer)	定义在 Disruptor 上允许的制作者。允许的选项有：Multi to allow multiple producers 和 Single，仅在一个并发制作者（在一个线程或其他同步）处于活跃状态时才允许进行某些优化。	multi	DisruptorProducerType
timeout (producer)	生成者停止等待异步任务完成前的超时（以毫秒为单位）。您可以使用 0 或负值禁用超时。	30000	long

Name	描述	默认值	类型
waitForTaskToComplete (producer)	选项指定调用者是否应该等待 async 任务在继续前等待。支持以下三个选项： Always、Never 或 IfReplyExpected。前两个值为 self-explanatory。最后的值如果为 Request ReplyExpected，只有在消息是 Request Reply based 时等待。	IfReplyExpected	WaitForTaskToComplete
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

85.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 18 个选项，如下所列。

Name	描述	默认值	类型
camel.component.disruptor-vm.buffer-size	配置环缓冲大小	1024	整数
camel.component.disruptor-vm.default-block-when-full	当 full 的默认值为 true 时，要为 block 配置默认值。	true	布尔值
camel.component.disruptor-vm.default-concurrent-consumers	配置默认的并发消费者数	1	整数
camel.component.disruptor-vm.default-multiple-consumers	为多个消费者配置默认值	false	布尔值
camel.component.disruptor-vm.default-producer-type	要为 DisruptorProducerType 配置默认值，默认值为 Multi。		DisruptorProducerType
camel.component.disruptor-vm.default-wait-strategy	要为 DisruptorWaitStrategy 配置默认值，默认值为 Blocking。		DisruptorWaitStrategy

Name	描述	默认值	类型
camel.component.disruptor-vm.enabled	启用 disruptor-vm 组件	true	布尔值
camel.component.disruptor-vm.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.disruptor.buffer-size	配置环缓冲大小	1024	整数
camel.component.disruptor.default-block-when-full	当 full 的默认值为 true 时，要为 block 配置默认值。	true	布尔值
camel.component.disruptor.default-concurrent-consumers	配置默认的并发消费者数	1	整数
camel.component.disruptor.default-multiple-consumers	为多个消费者配置默认值	false	布尔值
camel.component.disruptor.default-producer-type	要为 DisruptorProducerType 配置默认值，默认值为 Multi。		DisruptorProducerType
camel.component.disruptor.default-wait-strategy	要为 DisruptorWaitStrategy 配置默认值，默认值为 Blocking。		DisruptorWaitStrategy
camel.component.disruptor.enabled	启用 disruptor 组件	true	布尔值
camel.component.disruptor.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.disruptor-vm.queue-size	配置环缓冲大小		整数

Name	描述	默认值	类型
camel.component.disruptor.queue-size	配置环缓冲大小		整数

85.4. 等待策略

等待策略会影响当前等待下一个交换发布的消费者线程的等待类型。可以选择以下策略：

Name	描述	建议
Blocking	阻塞使用 lock 和 condition 变量的策略来等待 Consumers 等待 barrier。	当吞吐量和低延迟不重要时，可以使用此策略。
sleeping	休眠最初启动的策略，然后使用 Thread.yield（），最终作为 OS 和 JVM 的纳纳数，而 JVM 将允许 Consumers 在 barrier 上等待。	此策略在性能和 CPU 资源之间是很好的折衷。在静默周期后，延迟激增可能会发生。
BusySpin	Spin 策略使用忙碌的旋转循环，用于等待障碍的 Consumers。	此策略将使用 CPU 资源来避免系统调用，从而引入延迟。当线程可以绑定到特定 CPU 内核时，最好使用它。
创建	生成使用 Thread.yield （）在初始旋转后等待 Consumers 的策略。	此策略在性能和 CPU 资源之间是很好的折衷，而不会造成显著的延迟激增。

85.5. 重新使用请求

Disruptor 组件支持使用 Request Reply，调用者将等待 Async 路由完成。例如：

```
from("mina:tcp://0.0.0.0:9876?textline=true&sync=true").to("disruptor:input");
from("disruptor:input").to("bean:processInput").to("bean:createResponse");
```

在上面的路由中，我们在端口 9876 上有一个 TCP 侦听器，它接受传入的请求。请求被路由到 disruptor:input 缓冲。当它是 Request Reply 消息时，我们会等待响应。当 disruptor:input 缓冲区上的消费者完成后，它会将响应复制到原始消息响应。

85.6. 并发消费者

默认情况下，Disruptor 端点使用单一消费者线程，但您可以将它配置为使用并发消费者线程。因此，您可以使用的线程池而不是线程池：

```
from("disruptor:stageName?concurrentConsumers=5").process(...)
```

与两者之间的差别一样，请注意，根据负载，线程池可以在运行时动态增加/shrink，而并发消费者的数量始终被固定，且由 Disruptor 支持，因此性能会更高。

85.7. 线程池

请注意，通过执行以下操作将线程池添加到 Disruptor 端点：

```
from("disruptor:stageName").thread(5).process(...)
```

可以利用 Disruptor 添加与 Disruptor 配合使用的普通 BlockingQueue，从而有效地给性能带来的一部分，从而利用 Disruptor 实现了性能收益。相反，建议您使用 concurrentConsumers 选项直接在处理 Disruptor 端点上处理消息的线程数量。

85.8. 示例

在以下路由中，我们使用 Disruptor 将请求发送到此 async 队列，以便能够向另一个线程中进一步处理发送一个 fire-and-get 消息，并将此线程中的恒定回复返回给原始调用者。

```
public void configure() throws Exception {
    from("direct:start")
        // send it to the disruptor that is async
        .to("disruptor:next")
        // return a constant response
        .transform(constant("OK"));

    from("disruptor:next").to("mock:result");
}
```

在这里，我们发送 Hello World 消息，并期望回复是 OK。

```
Object out = template.requestBody("direct:start", "Hello World");
assertEquals("OK", out);
```

"Hello World"消息将从另一个线程中使用 Disruptor，以进行进一步处理。由于这来自单元测试，因此它将发送到一个模拟端点，在单元测试中我们可以进行断言。

85.9. 使用 MULTIPLECONSUMERS

在这个示例中，我们定义了两个消费者，并将其注册为 spring Bean。

```
<!-- define the consumers as spring beans -->
<bean id="consumer1" class="org.apache.camel.spring.example.FooEventConsumer"/>

<bean id="consumer2"
class="org.apache.camel.spring.example.AnotherFooEventConsumer"/>

<camelContext xmlns="http://camel.apache.org/schema/spring">
    <!-- define a shared endpoint which the consumers can refer to instead of using url -->
    <endpoint id="foo" uri="disruptor:foo?multipleConsumers=true"/>
</camelContext>
```

由于在 Disruptor foo 端点上指定了 multipleConsumers=true，所以我们可以使这两个或者多个消费者接收自己的消息副本作为 pub-sub 风格的消息传递。因为 Bean 是单元测试的一部分，它们只是将消息发送到模拟端点，但注意我们可以如何使用 @Consume 从 Disruptor 使用。

```
public class FooEventConsumer {

    @EndpointInject(uri = "mock:result")
    private ProducerTemplate destination;

    @Consume(ref = "foo")
    public void doSomething(String body) {
        destination.sendBody("foo" + body);
    }

}
```

85.10. 提取中断器信息

如果需要，可以在不使用 JMX 的情况下获得缓冲区大小等信息：

```
DisruptorEndpoint disruptor = context.getEndpoint("disruptor:xxxx");
int size = disruptor.getBufferSize();
```

第 86 章 DNS 组件

从 Camel 版本 2.7 开始提供

这是 Camel 使用 DNSJava 运行 DNS 查询的其他组件。组件是 DNSJava 顶部的精简层。
组件提供以下操作：

- ip，根据其 ip 解析域
- 查找有关域的信息
- dig，运行 DNS 查询

INFO:*Requires SUN JVM* DNSJava 库需要在 SUN JVM 上运行。
如果使用 Apache ServiceMix 或 Apache Karaf，则需要调整 etc/jre.properties 文件，将 sun.net.spi.nameservice 添加到导出的 Java 平台软件包列表中。服务器将在此更改生效之前重新启动。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-dns</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

86.1. URI 格式

DNS 组件的 URI 方案如下

```
dns://operation[?options]
```

此组件仅支持生成者。

86.2. 选项

DNS 组件没有选项。

DNS 端点使用 URI 语法进行配置：

```
dns:dnsType
```

使用以下路径和查询参数：

86.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
dnsType	**必需** 查找的类型。		DnsType

86.2.2. 查询参数(1 参数)：

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

86.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.dns.enabled	启用 dns 组件	true	布尔值
camel.component.dns.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

86.4. HEADERS

标头	类型	操作	描述
dns.domain	字符串	ip	域名。必需。
dns.name	字符串	lookup	要查找的名称。必需。
dns.type		lookup, dig	查找的类型。应该与 **org.xbill.dns.Type** 的值匹配。可选。
dns.class		lookup, dig	查找的 DNS 类。应该与 **org.xbill.dns.DClass** 的值匹配。可选。
dns.query	字符串	dig	查询本身。必需。
dns.server	字符串	dig	特定于查询的服务器。如果未指定，则使用由操作系统指定的默认值。可选。

86.5. 例子

86.5.1. IP 查找

```
<route id="IPCheck">
    <from uri="direct:start"/>
    <to uri="dns:ip"/>
</route>
```

这将查找域的 IP。例如： www.example.com 解析为 192.0.32.10。
要查询的 IP 地址必须在标头中提供，键为 "dns.domain "。

86.5.2. DNS 查找

```
<route id="IPCheck">
    <from uri="direct:start"/>
    <to uri="dns:lookup"/>
</route>
```

这会返回与域关联的一组 DNS 记录。
要查找的名称必须在标头中提供，键为 "dns.name "。

86.5.3. DNS Dig

dig 是运行 DNS 查询的 Unix 命令行工具。

```
<route id="IPCheck">
    <from uri="direct:start"/>
    <to uri="dns:dig"/>
</route>
```

查询必须在标头中提供，键为 "dns.query "。

86.6. DNS 激活策略

DnsActivationPolicy 可以用来根据 dns 状态动态启动和停止路由。

如果您的同一组件在不同区域中运行的实例，您可以在每个区域中配置路由，使其仅在 dns 指向其区域时激活。

例如，您可能在 NYC 中有一个实例，以及 SFO 中的实例。您可以将服务 CNAME service.example.com 配置为指向 nyc-service.example.com，使 NYC 实例启动并关闭 SFO 实例。当您将 CNAME service.example.com 更改为指向 sfo-service.example.com swig-wagonnyc 实例时，将停止其路由，sfo 将启动其路由。这可让您在不重启实际组件的情况下切换区域。

```
<bean id="dnsActivationPolicy"
class="org.apache.camel.component.dns.policy.DnsActivationPolicy">
 <property name="hostname" value="service.example.com" />
 <property name="resolvesTo" value="nyc-service.example.com" />
 <property name="ttl" value="60000" />
 <property name="stopRoutesOnException" value="false" />
</bean>

<route id="routeId" autoStartup="false" routePolicyRef="dnsActivationPolicy">
</route>
```

第 87 章 DOCKER 组件

从 Camel 版本 2.15 开始提供

用于与 Docker 通信的 Camel 组件。

Docker Camel 组件通过 Docker 远程 API 利用 docker-java。

87.1. URI 格式

```
docker://[operation]?[options]
```

其中 operation 是对 Docker 执行的特定操作。

87.2. 常规选项

Docker 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
配置（高级）	使用共享 docker 配置		DockerConfiguration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Docker 端点使用 URI 语法进行配置：

```
docker:operation
```

使用以下路径和查询参数：

87.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
operation	**必需的** Which 操作		DockerOperation

87.2.2. 查询参数(20 参数)：

Name	描述	默认值	类型
email (common)	与用户关联的电子邮件地址		字符串
host (common)	**所需的** Docker 主机	localhost	字符串
port (common)	**所需的** Docker 端口	2375	整数
requestTimeout (common)	请求超时响应（以秒为单位）		整数
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
cmdExecFactory (advanced)	要使用的 DockerCmdExecFactory 实现的完全限定类名称	com.github.dockerjava.netty.NettyDockerCmdExecFactory	字符串
followRedirectFilter (advanced)	是否跟踪重定向过滤器	false	布尔值
loggingFilter (advanced)	是否使用日志记录过滤器	false	布尔值

Name	描述	默认值	类型
maxPerRouteConnections (advanced)	最大路由连接	100	整数
maxTotalConnections (advanced)	最大连接总数	100	整数
serverAddress (advanced)	docker registry 的服务器地址。	https://index.docker.io/v1/	字符串
socket (advanced)	套接字连接模式	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
certpath (security)	包含 SSL 证书链的位置		字符串
密码 (security)	要进行身份验证的密码		字符串
安全 （安全）	使用 HTTPS 通信	false	布尔值
tlsVerify (security)	检查 TLS	false	布尔值
用户名 (security)	要进行身份验证的用户名		字符串

87.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 20 个选项，如下所列。

Name	描述	默认值	类型
camel.component.docker.configuration.cert-path	包含 SSL 证书链的位置		字符串

Name	描述	默认值	类型
camel.component.docker.configuration.cmd-exec-factory	要使用的 DockerCmdExecFactory 实现的完全限定类名称	com.github.dockerjava.netty.NettyDockerCmdExecFactory	字符串
camel.component.docker.configuration.email	与用户关联的电子邮件地址		字符串
camel.component.docker.configuration.follow-redirect-filter	是否跟踪重定向过滤器	false	布尔值
camel.component.docker.configuration.host	Docker 主机	localhost	字符串
camel.component.docker.configuration.logging-filter	是否使用日志记录过滤器	false	布尔值
camel.component.docker.configuration.max-per-route-connections	最大路由连接	100	整数
camel.component.docker.configuration.max-total-connections	最大连接总数	100	整数
camel.component.docker.configuration.operation	要使用哪个操作		DockerOperation
camel.component.docker.configuration.parameters	其他配置参数作为键/值对		Map

Name	描述	默认值	类型
camel.component.docker.configuration.password	要进行身份验证的密码		字符串
camel.component.docker.configuration.port	Docker 端口	2375	整数
camel.component.docker.configuration.request-timeout	请求超时响应（以秒为单位）		整数
camel.component.docker.configuration.secure	使用 HTTPS 通信	false	布尔值
camel.component.docker.configuration.server-address	docker registry 的服务器地址。	https://index.docker.io/v1/	字符串
camel.component.docker.configuration.socket	套接字连接模式	true	布尔值
camel.component.docker.configuration.tls-verify	检查 TLS	false	布尔值
camel.component.docker.configuration.username	要进行身份验证的用户名		字符串
camel.component.docker.enabled	启用 docker 组件	true	布尔值
camel.component.docker.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

87.4. 标头策略

所有 URI 选项都可以作为标头属性传递。消息标头中找到的值优先于 URI 参数。标头属性采用前缀 CamelDocker 的 URI 选项形式，如下所示

URI 选项	标头属性
containerId	CamelDockerContainerId

87.5. 例子

以下示例消耗 Docker 中的事件：

```
from("docker://events?host=192.168.59.103&port=2375").to("log:event");
```

以下示例查询 Docker 以获取系统范围信息

```
from("docker://info?host=192.168.59.103&port=2375").to("log:info");
```

87.6. 依赖项

要在 Camel 路由中使用 Docker，您需要添加对 camel-docker 的依赖，该依赖项实施该组件。

如果使用 Maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-docker</artifactId>
  <version>x.x.x</version>
</dependency>
```

第 88 章 DOZER 组件

从 Camel 版本 2.15 开始提供

dozer: 组件提供从 Camel 2.15.0 起使用 Dozer 映射框架在 Java Bean 之间映射的功能。Camel 还支持将 Dozer 映射触发 为类型转换器 的功能。使用 Dozer 端点和 Dozer 转换器之间的主要区别是：

- 能够基于每个端点，通过转换器注册表管理 Dozer 映射配置与全局配置。
- Dozer 端点可以使用 Camel 数据格式配置 marshal/unmarshal 输入和输出数据，以支持单个、任何到任何转换端点
- Dozer 组件允许 Dozer 的精细集成和扩展来支持额外的功能（如映射字面值、使用表达式进行映射等）。

要使用 Dozer 组件，Maven 用户需要将以下依赖项添加到其 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-dozer</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

88.1. URI 格式

Dozer 组件仅支持生成者端点。

```
dozer:endpointId[?options]
```

其中 endpointId 是一个用于唯一标识 Dozer 端点配置的名称。

Dozer 端点 URI 示例：

```
from("direct:orderInput").
  to("dozer:transformOrder?
```

```
mappingFile=orderMapping.xml&targetModel=example.XYZOrder").
  to("direct:orderOutput");
```

88.2. 选项

Dozer 组件没有选项。

Dozer 端点使用 URI 语法进行配置：

```
dozer:name
```

使用以下路径和查询参数：

88.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	**需要** 映射的人类可读名称。		字符串

88.2.2. 查询参数(7 参数)：

Name	描述	默认值	类型
mappingConfiguration (producer)	Camel registry 中 DozerBeanMapperConfiguration bean 的名称，该 registry 应用于配置 Dozer 映射。这是用于精细控制 Dozer 的 mappingFile 选项的替代选择。记住在值中使用 # 前缀，表示 bean 位于 Camel registry 中（例如 #myDozerConfig）。		DozerBeanMapper 配置
mappingFile (producer)	Dozer 配置文件的位置。默认情况下，该文件从 classpath 加载，但您可以使用 file:、classpath: 或 http: 从特定位置加载配置。	dozerBeanMapping.xml	字符串
marshalId (producer)	Camel 上下文中定义的 dataFormat 的 id，用于将映射输出放入非 Java 类型。		字符串
sourceModel (producer)	映射中使用的源类型的完全限定域名。如果指定，在使用 Dozer 映射前，到映射的输入将转换为指定的类型。		字符串
targetModel (producer)	映射中使用的目标类型 **需要** Fully-qualified class 名称。		字符串

Name	描述	默认值	类型
unmarshalId (producer)	Camel 上下文中定义的 dataFormat 的 id，用于从非 Java 类型设置映射输入。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

88.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.dozer.enabled	启用 dozer 组件	true	布尔值
camel.component.dozer.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

88.4. 在 DOZER 中使用数据格式

Dozer 不支持用于映射的非 Java 源和目标，因此无法自行将 XML 文档映射到 Java 对象。幸运的是，Camel 具有广泛支持，支持 Java 和使用数据格式的 各种格式。Dozer 组件通过允许您指定在通过 Dozer 处理之前通过数据格式传递的输入和输出数据，从而利用这种支持。您始终可以自行对 Dozer 执行此操作，但直接在 Dozer 组件中支持它，您可以使用单个端点在 Camel 中配置任何对任意转换。

例如，假设您要使用 Dozer 组件在 XML 数据结构和 JSON 数据结构之间进行映射。如果您在 Camel 上下文中定义以下数据格式：

```
<dataFormats>
  <json library="Jackson" id="myjson"/>
  <jaxb contextPath="org.example" id="myjaxb"/>
</dataFormats>
```

然后，您可以将 Dozer 端点配置为使用 JAXB 数据格式的输入 XML，并使用 Jackson 对映射输出进行提取。

```
<endpoint uri="dozer:xml2json?
marshalId=myjson&amp;unmarshalId=myjaxb&amp;targetModel=org.example.Order"/>
```

88.5. 配置 DOZER

所有 Dozer 端点都需要一个 Dozer 映射配置文件，该文件定义源和目标对象之间的映射。如果端点上没有指定 mappingFile 或 mappingConfiguration 选项，则组件将默认为 META-INF/dozerBeanMapping.xml 的位置。如果您需要为单个端点提供多个映射文件，或者指定额外的配置选项（如事件监听程序、自定义转换器等），您可以使用 org.apache.camel.converter.dozer.Dozer.BeanMapperConfiguration 实例。

```
<bean id="mapper" class="org.apache.camel.converter.dozer.DozerBeanMapperConfiguration">
  <property name="mappingFiles">
    <list>
      <value>mapping1.xml</value>
      <value>mapping2.xml</value>
    </list>
  </property>
</bean>
```

88.6. 映射扩展

Dozer 组件将多个扩展实施到 Dozer 映射框架，作为自定义转换器。这些转换器实施直接由 Dozer 本身支持的映射功能。

88.6.1. 变量映射

通过变量映射，您可以将 Dozer 配置中的变量定义值映射到 target 字段，而不使用 source 字段的值。这等同于在其他映射框架中持续映射，您可以在其中为目标字段分配字面值。要使用变量映射，只需在映射配置中定义一个变量，然后将 VariableMapper 类映射到您选择的目标字段中：

```
<mappings xmlns="http://dozermapper.github.io/schema/bean-mapping"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://dozermapper.github.io/schema/bean-mapping
http://dozermapper.github.io/schema/bean-mapping.xsd">
  <configuration>
    <variables>
      <variable name="CUST_ID">ACME-SALES</variable>
    </variables>
  </configuration>
  <mapping>
    <class-a>org.apache.camel.component.dozer.VariableMapper</class-a>
    <class-b>org.example.Order</class-b>
    <field custom-converter-id="_variableMapping" custom-converter-param="${CUST_ID}">
      <a>literal</a>
      <b>custId</b>
```

```
    </field>
  </mapping>
</mappings>
```

88.6.2. 自定义映射

通过自定义映射，您可以为源字段映射到 target 字段定义您自己的逻辑。它们的功能与 Dozer 客户转换器类似，有两个值得注意的区别：

- 您可以在带有自定义映射的单一类中有多个转换器方法。
- 不需要使用自定义映射实施特定于 Dozer 的接口。

使用映射配置中内置的 '_customMapping' converter 来声明自定义映射。此转换器的参数具有以下语法：

```
[class-name][,method-name]
```

方法名称是可选的 - Dozer 组件将搜索与映射所需的输入和输出类型匹配的方法。以下是自定义映射和配置示例。

```
public class CustomMapper {
    // All customer ids must be wrapped in "[ ]"
    public Object mapCustomer(String customerId) {
        return "[" + customerId + "]";
    }
}
```

```
<mappings xmlns="http://dozermapper.github.io/schema/bean-mapping"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://dozermapper.github.io/schema/bean-mapping
http://dozermapper.github.io/schema/bean-mapping.xsd">
  <mapping>
    <class-a>org.example.A</class-a>
    <class-b>org.example.B</class-b>
    <field custom-converter-id="_customMapping"
        custom-converter-param="org.example.CustomMapper,mapCustomer">
      <a>header.customerNum</a>
      <b>custId</b>
    </field>
  </mapping>
</mappings>
```

88.6.3. 表达式映射

表达式映射允许您使用 Camel 的强大 语言 功能来评估表达式，并将结果分配给映射中的目标字段。Camel 支持的任何语言都可以在表达式映射中使用。表达式的基本示例包括可以将 Camel 消息标头或交换属性映射到目标字段，或将多个源字段连接到目标字段中。映射表达式的语法为：

```
[language]:[expression]
```

将消息标头映射到目标字段的示例：

```
<mappings xmlns="http://dozermapper.github.io/schema/bean-mapping"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://dozermapper.github.io/schema/bean-mapping
http://dozermapper.github.io/schema/bean-mapping.xsd">
  <mapping>
    <class-a>org.apache.camel.component.dozer.ExpressionMapper</class-a>
    <class-b>org.example.B</class-b>
    <field custom-converter-id="_expressionMapping" custom-converter-
param="simple:\${header.customerNumber}">
      <a>expression</a>
      <b>custId</b>
    </field>
  </mapping>
</mappings>
```

请注意，表达式中的任何属性都必须使用 "\" 进行转义，以防止 Dozer 尝试解析使用 EL 定义的变量值时出现错误。

第 89 章 深度组件

从 Camel 版本 2.19 开始提供

deep: 组件可让您查询 Apache Drill Cluster

deep 是一个 Apache 开源 SQL 查询引擎，用于大型数据探索。深入的设计旨在支持半结构化和快速发展来自现代大数据应用程序的高性能分析，同时仍然提供 ANSI SQL 的熟悉和生态系统，这是行业标准查询语言。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-drill</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

89.1. URI 格式

```
drill://host[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

89.2. 深入的生成者

生成者使用 CamelDrillQuery 标头执行查询，并将结果放入正文。

89.3. 选项

Drill 组件没有选项。

Drill 端点使用 URI 语法进行配置：

```
drill:host
```

使用以下路径和查询参数：

89.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
主机	**所需的** ZooKeeper 主机名或 IP 地址。使用 local 而不是主机名或 IP 地址连接到本地 Drillbit		字符串

89.3.2. 查询参数(5 参数)：

Name	描述	默认值	类型
clusterid (producer)	集群 ID https://drill.apache.org/docs/using-the-jdbc-driver/#determining-the-cluster-id		字符串
directory (producer)	ZooKeeper 中的深入目录		字符串
mode (producer)	连接模式：zk: Zookeeper drillbit: Drillbit Direct connection https://drill.apache.org/docs/using-the-jdbc-driver/	ZK	DrillConnectionMode
port (producer)	ZooKeeper 端口号		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

89.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.drill.enabled	启用深入组件	true	布尔值
camel.component.drill.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

89.5. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 90 章 DROPBOX 组件

从 Camel 版本 2.14 开始提供

dropbox: 组件允许您将 Dropbox 远程文件夹视为生成者或消息使用者。使用 Dropbox Java Core API （此组件的参考版本为 1.7.x），这个 camel 组件具有以下功能：

- 作为消费者，通过查询下载文件和搜索文件
- 作为制作者，下载文件，在远程目录之间移动文件，删除文件/目录，通过查询上传文件和搜索文件

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-dropbox</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

90.1. URI 格式

```
dropbox://[operation]?[options]
```

其中 operation 是对 Dropbox 远程文件夹执行的特定操作（通常是 CRUD 操作）。

90.2. 操作

操作	描述
del	删除 Dropbox 上的文件或目录
get	从 Dropbox 下载文件
Move	从 Dropbox 上的文件夹移动文件
put	上传 Dropbox 上的文件

操作	描述
search	根据字符串查询在 Dropbox 上搜索文件

操作 需要额外的选项才能工作，某些选项对于特定操作是必需的。

90.3. 选项

要使用 Dropbox API，您需要获取 accessToken 和 clientIdentifier。您可以参考说明如何获取它们的 Dropbox 文档。

Dropbox 组件没有选项。

Dropbox 端点使用 URI 语法进行配置：

```
dropbox:operation
```

使用以下路径和查询参数：

90.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
operation	**必需** 的特定操作（通常是 CRUD 操作）才能对 Dropbox 远程文件夹执行。		DropboxOperation

90.3.2. 查询参数(12 参数)：

Name	描述	默认值	类型
accessToken (common)	**必需的** 访问令牌，以为特定 Dropbox 用户发出 API 请求		字符串
client (common)	使用现有的 DbxClient 实例作为 DropBox 客户端。		DbxClientV2
clientIdentifier (common)	注册用于发出 API 请求的应用程序名称		字符串

Name	描述	默认值	类型
localPath (common)	可选文件夹或文件，以在本地文件系统的 Dropbox 上上传。如果还没有配置这个选项，则会将消息正文用作要上传的内容。		字符串
newRemotePath (common)	目标文件或文件夹		字符串
query (common)	要搜索的、以空格分隔的子字符串列表。只有文件包含所有子字符串时，才会匹配。如果没有设置这个选项，则会匹配所有文件。		字符串
remotePath (common)	要移动的原始文件或文件夹		字符串
uploadMode (common)	如果 dropbox 上已存在具有相同名称的文件，则要上传的模式。如果 dropbox 上已存在具有相同名称的文件，则这将被覆盖。		DropboxUploadMode
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

90.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.dropbox.enabled	启用 dropbox 组件	true	布尔值

Name	描述	默认值	类型
camel.component.dropbox.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

90.5. DEL 操作

删除 Dropbox 上的文件。

只能作为 Camel producer 工作。

下面列出了此操作的选项：

属性	必需	描述
remotePath	**true**	要删除在 Dropbox 上的文件夹或文件

90.5.1. Samples

```
from("direct:start")
  .to("dropbox://del?accessToken=XXX&clientIdentifier=XXX&remotePath=/root/folder1")
  .to("mock:result");

from("direct:start")
  .to("dropbox://del?
accessToken=XXX&clientIdentifier=XXX&remotePath=/root/folder1/file1.tar.gz")
  .to("mock:result");
```

90.5.2. 结果消息标头

在消息结果上设置以下标头：

属性	value
DELETED_PATH	dropbox 中删除的路径名称

90.5.3. 结果消息正文

以下对象在消息正文结果上设置：

对象类型	描述
字符串	dropbox 中删除的路径名称

90.6. GET (DOWNLOAD)操作

从 Dropbox 下载文件。

作为 Camel producer 或 Camel 使用者工作。

下面列出了此操作的选项：

属性	必需	描述
remotePath	**true**	从 Dropbox 下载的文件夹或文件

90.6.1. Samples

```
from("direct:start")
  .to("dropbox://get?
accessToken=XXX&clientIdentifier=XXX&remotePath=/root/folder1/file1.tar.gz")
  .to("file:///home/kermit/?fileName=file1.tar.gz");

from("direct:start")
  .to("dropbox://get?accessToken=XXX&clientIdentifier=XXX&remotePath=/root/folder1")
  .to("mock:result");

from("dropbox://get?accessToken=XXX&clientIdentifier=XXX&remotePath=/root/folder1")
  .to("file:///home/kermit/");
```

90.6.2. 结果消息标头

在消息结果上设置以下标头：

属性	value
DOWNLOADED_FILE	如果单个文件下载，则下载的远程文件的路径
DOWNLOADED_FILES	如果有多个文件下载，则下载的远程文件的路径

90.6.3. 结果消息正文

以下对象在消息正文结果上设置：

对象类型	描述
ByteArrayOutputStream	如果单个文件下载，代表下载的文件的流
Map<String, ByteArrayOutputStream>	如果有多个文件下载，则映射作为下载远程文件的路径的映射，作为代表下载文件的流的值

90.7. 移动操作

将 Dropbox 上的文件移动到另一个文件夹。

只能作为 Camel producer 工作。

下面列出了此操作的选项：

属性	必需	描述
remotePath	**true**	要移动的原始文件或文件夹
newRemotePath	**true**	目标文件或文件夹

90.7.1. Samples

```
from("direct:start")
  .to("dropbox://move?
accessToken=XXX&clientIdentifier=XXX&remotePath=/root/folder1&newRemotePath=/root/folder2")
  .to("mock:result");
```

90.7.2. 结果消息标头

在消息结果上设置以下标头：

属性	value
MOVED_PATH	dropbox 上移动的路径名称

90.7.3. 结果消息正文

以下对象在消息正文结果上设置：

对象类型	描述
字符串	dropbox 上移动的路径名称

90.8. 放置（上传）操作

上传 Dropbox 上的文件。

作为 Camel producer 工作。

下面列出了此操作的选项：

属性	必需	描述
uploadMode	**true**	添加或强制此选项指定在 dropbox 上应如何保存文件：如果 dropbox 上已存在具有相同名称的文件，则会重命名新文件。如果 dropbox 上已存在具有相同名称的文件，则这将被覆盖。
localPath	**false**	文件夹或文件，以在本地文件系统上上传 Dropbox。如果已经配置了这个选项，则优先于将内容上传为单个文件，其中包含来自 Camel 消息正文的内容（消息正文将转换为字节数数组）。

属性	必需	描述
remotePath	**false**	Dropbox 上的文件夹目的地。如果没有设置属性，则组件将在与本地路径相等的远程路径中上传该文件。对于 Windows 或没有绝对 localPath，您可以运行类似如下的例外： 原因原因：java.lang.IllegalArgumentException: 'path': bad path: must start with "/": "C:/My/File" OR Caused by: java.lang.IllegalArgumentException: 'path': bad path: must start with "/": "MyFile"

90.8.1. Samples

```
from("direct:start").to("dropbox://put?
accessToken=XXX&clientIdentifier=XXX&uploadMode=add&localPath=/root/folder1")
  .to("mock:result");

from("direct:start").to("dropbox://put?
accessToken=XXX&clientIdentifier=XXX&uploadMode=add&localPath=/root/folder1&remotePa
th=/root/folder2")
  .to("mock:result");
```

和 上传包含消息正文内容的单个文件

```
from("direct:start")
   .setHeader(DropboxConstants.HEADER_PUT_FILE_NAME, constant("myfile.txt"))
   .to("dropbox://put?
accessToken=XXX&clientIdentifier=XXX&uploadMode=add&remotePath=/root/folder2")
   .to("mock:result");
```

文件的名称可以在标题 DropboxConstants.HEADER_PUT_FILE_NAME 或 Exchange.FILE_NAME 中按优先顺序提供。如果没有提供标头，则使用消息 id (uuid)作为文件名。

90.8.2. 结果消息标头

在消息结果上设置以下标头：

属性	value
UPLOADED_FILE	如果单个文件上传，则远程路径的路径上传

属性	value
UPLOADED_FILES	如果有多个文件上传，则上传了远程路径的字符串

90.8.3. 结果消息正文

以下对象在消息正文结果上设置：

对象类型	描述
字符串	如果单个文件上传，则上传操作(OK 或 KO)
Map<String, DropboxResultCode>	如果有多个文件上传，使用 作为上传远程文件的路径的映射，以及上传操作的结果(OK 或 KO)的结果

90.9. 搜索操作

在远程 Dropbox 文件夹中搜索，包括其子目录。

作为 Camel producer 和 Camel 使用者工作。

下面列出了此操作的选项：

属性	必需	描述
remotePath	**true**	要搜索的 Dropbox 上的文件夹。
query	**true**	要搜索的、以空格分隔的子字符串列表。只有文件包含所有子字符串时，才会匹配。如果没有设置这个选项，则会匹配所有文件。需要在端点配置中或作为 Camel 消息上的标头 **CamelDropboxQuery** 提供查询。

90.9.1. Samples

```
from("dropbox://search?
accessToken=XXX&clientIdentifier=XXX&remotePath=/XXX&query=XXX")
  .to("mock:result");

from("direct:start")
```

```
.setHeader("CamelDropboxQuery", constant("XXX"))
.to("dropbox://search?accessToken=XXX&clientIdentifier=XXX&remotePath=/XXX")
.to("mock:result");
```

90.9.2. 结果消息标头

在消息结果上设置以下标头：

属性	value
FOUNDED_FILES	找到的文件路径列表

90.9.3. 结果消息正文

以下对象在消息正文结果上设置：

对象类型	描述
List<DbxEntry>	找到的文件路径列表。有关此对象的更多信息，请参阅 Dropbox 文档，

第 91 章 EHCACHE 组件

从 Camel 版本 2.18 开始提供

ehcache 组件允许您使用 Ehcache 3 作为缓存实施来执行缓存操作。

此组件支持基于制作者和事件的消费者端点。

缓存消费者是基于事件的消费者，可用于侦听和响应特定的缓存活动。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-ehcache</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

91.1. URI 格式

```
ehcache://cacheName[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option =value&option=""beanRef&...

91.2. 选项

Ehcache 组件支持 7 个选项，如下所列。

Name	描述	默认值	类型
配置（高级）	设置全局组件配置		EhcacheConfiguration
cacheManager (common)	缓存管理器		CacheManager

Name	描述	默认值	类型
cacheManager 配置 (common)	缓存管理器配置		配置
cacheConfiguration (common)	用于创建缓存的默认缓存配置。		CacheConfiguration
cacheConfigurations (common)	用于创建缓存的缓存配置映射。		Map
cacheConfigurationUri (common)	指向 Ehcache XML 配置文件位置的 URI		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Ehcache 端点使用 URI 语法进行配置：

```
ehcache:cacheName
```

使用以下路径和查询参数：

91.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	**所需的** 缓存名称		字符串

91.2.2. 查询参数(17 参数)：

Name	描述	默认值	类型
cacheManager (common)	缓存管理器		CacheManager
cacheManagerConfiguration (common)	缓存管理器配置		配置

Name	描述	默认值	类型
configurationUri (common)	指向 Ehcache XML 配置文件位置的 URI		字符串
createCacheIfNotExist (common)	如果缓存存在或无法预先配置，则配置是否需要创建缓存。	true	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
eventFiring (consumer)	设置交付模式（同步、异步）	异步	EventFiring
eventOrdering (consumer)	设置交付模式（排序、未排序）	排序	EventOrdering
eventTypes (consumer)	设置要侦听的事件类型	EVICTED,EXPIRED,REMOVED,CREATED,UPDATED	Set
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
action (producer)	配置默认缓存操作。如果在消息标头中设置了操作，则标题中的操作将具有优先权。		字符串
key (producer)	配置默认操作密钥。如果在消息标头中设置了键，则标题中的键具有优先权。		对象
配置 （高级）	用于创建缓存的默认缓存配置。		CacheConfiguration
配置 （高级）	用于创建缓存的缓存配置映射。		Map

Name	描述	默认值	类型
keyType (advanced)	缓存密钥类型，默认 java.lang.Object	java.lang.Object	字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
valueType (advanced)	缓存值类型，默认为 java.lang.Object	java.lang.Object	字符串

91.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 25 个选项，如下所列。

Name	描述	默认值	类型
camel.component.ehcache.cache-configuration	用于创建缓存的默认缓存配置。选项是一个 org.ehcache.config.CacheConfiguration<?,?> 类型。		字符串
camel.component.ehcache.cache-configuration-uri	指向 Ehcache XML 配置文件位置的 URI		字符串
camel.component.ehcache.cache-manager	缓存管理器。选项是 org.ehcache.CacheManager 类型。		字符串
camel.component.ehcache.cache-manager-configuration	缓存管理器配置。选项是 org.ehcache.config.Configuration 类型。		字符串
camel.component.ehcache.caches-configurations	用于创建缓存的缓存配置映射。		Map
camel.component.ehcache.configuration.action	配置默认缓存操作。如果在消息标头中设置了操作，则标题中的操作将具有优先权。		字符串

Name	描述	默认值	类型
camel.component.ehcache.configuration.cache-manager	缓存管理器		CacheManager
camel.component.ehcache.configuration.cache-manager-configuration	缓存管理器配置		配置
camel.component.ehcache.configuration.configuration	用于创建缓存的默认缓存配置。		CacheConfiguration
camel.component.ehcache.configuration.configuration-uri	指向 Ehcache XML 配置文件位置的 URI		字符串
camel.component.ehcache.configuration.configurations	用于创建缓存的缓存配置映射。		Map
camel.component.ehcache.configuration.create-cache-if-not-exist	如果缓存存在或无法预先配置，则配置是否需要创建缓存。	true	布尔值
camel.component.ehcache.configuration.event-firing	设置交付模式（同步、异步）		EventFiring
camel.component.ehcache.configuration.event-ordering	设置交付模式（排序、未排序）		EventOrdering
camel.component.ehcache.configuration.event-types	设置要侦听的事件类型		Set
camel.component.ehcache.configuration.key	配置默认操作密钥。如果在消息标头中设置了键，则标题中的键具有优先权。		对象

Name	描述	默认值	类型
camel.component.ehcache.configuration.key-type	缓存密钥类型，默认 java.lang.Object	java.lang.Object	字符串
camel.component.ehcache.configuration.value-type	缓存值类型，默认为 java.lang.Object	java.lang.Object	字符串
camel.component.ehcache.customizer.cache-configuration.enabled	启用或禁用 cache-configuration 自定义器。	true	布尔值
camel.component.ehcache.customizer.cache-configuration.mode	配置（如果已添加了缓存配置），或者必须替换已在组件上配置的缓存配置。		CacheConfiguration CustomizerConfiguration$ Mode
camel.component.ehcache.customizer.cache-manager.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component.ehcache.customizer.cache-manager.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component.ehcache.enabled	启用 ehcache 组件	true	布尔值
camel.component.ehcache.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.ehcache.configuration.config-uri (DEPRECATED - Use **set configurationUri (String)** instead.)	指向 Ehcache XML 配置文件位置的 URI		字符串

91.3.1. Message Headers Camel

标头	类型	描述
CamelEhcacheAction	**字符串**	在缓存中执行 perfomed 的操作，有效选项有： * CLEAR * PUT * PUT_ALL * PUT_IF_ABSENT * GET * GET_ALL * REMOVE * REMOVE_ALL * REPLACE
CamelEhcacheActionHasResult	布尔值	如果操作的结果为 true
CamelEhcacheActionSucceeded	**布尔值**	如果 actionsuccedded 为 true
CamelEhcacheKey	对象	用于操作的缓存密钥
CamelEhcacheKeys	set<Object>	使用的密钥列表 * PUT_ALL * GET_ALL * REMOVE_ALL
CamelEhcacheValue	对象	放入缓存或操作结果的值
CamelEhcacheOldValue	对象	对于 PUT_IF_ABSENT 或用于与 REPLACE 等操作进行比较的对象，与键关联的旧值（如 PUT_IF_ABSENT）或对象
CamelEhcacheEventType	EventType	接收的事件类型

91.4. 基于 EHCACHE 的幂等存储库示例：

```
CacheManager manager = CacheManagerBuilder.newCacheManager(new
XmlConfiguration("ehcache.xml"));
EhcacheIdempotentRepository repo = new EhcacheIdempotentRepository(manager,
"idempotent-cache");

from("direct:in")
    .idempotentConsumer(header("messageId"), idempotentRepo)
    .to("mock:out");
```

91.5. 基于 EHCACHE 的聚合存储库示例：

```
public class EhcacheAggregationRepositoryRoutesTest extends CamelTestSupport {
    private static final String ENDPOINT_MOCK = "mock:result";
    private static final String ENDPOINT_DIRECT = "direct:one";
    private static final int[] VALUES = generateRandomArrayOfInt(10, 0, 30);
    private static final int SUM = IntStream.of(VALUES).reduce(0, (a, b) -> a + b);
    private static final String CORRELATOR = "CORRELATOR";

    @EndpointInject(uri = ENDPOINT_MOCK)
    private MockEndpoint mock;

    @Produce(uri = ENDPOINT_DIRECT)
    private ProducerTemplate producer;

    @Test
    public void checkAggregationFromOneRoute() throws Exception {
        mock.expectedMessageCount(VALUES.length);
        mock.expectedBodiesReceived(SUM);

        IntStream.of(VALUES).forEach(
            i -> producer.sendBodyAndHeader(i, CORRELATOR, CORRELATOR)
        );

        mock.assertIsSatisfied();
    }

    private Exchange aggregate(Exchange oldExchange, Exchange newExchange) {
        if (oldExchange == null) {
            return newExchange;
        } else {
            Integer n = newExchange.getIn().getBody(Integer.class);
            Integer o = oldExchange.getIn().getBody(Integer.class);
            Integer v = (o == null ? 0 : o) + (n == null ? 0 : n);

            oldExchange.getIn().setBody(v, Integer.class);

            return oldExchange;
        }
    }

    @Override
    protected RoutesBuilder createRouteBuilder() throws Exception {
        return new RouteBuilder() {
            @Override
            public void configure() throws Exception {
                from(ENDPOINT_DIRECT)
                    .routeId("AggregatingRouteOne")
                    .aggregate(header(CORRELATOR))
                    .aggregationRepository(createAggregateRepository())
                    .aggregationStrategy(EhcacheAggregationRepositoryRoutesTest.this::aggregate)
                    .completionSize(VALUES.length)

.to("log:org.apache.camel.component.ehcache.processor.aggregate.level=INFO&showAll=tru
e&mulltiline=true")
```

```
                .to(ENDPOINT_MOCK);
            }
        };
    }

    protected EhcacheAggregationRepository createAggregateRepository() throws Exception {
        CacheManager cacheManager = CacheManagerBuilder.newCacheManager(new
XmlConfiguration("ehcache.xml"));
        cacheManager.init();

        EhcacheAggregationRepository repository = new EhcacheAggregationRepository();
        repository.setCacheManager(cacheManager);
        repository.setCacheName("aggregate");

        return repository;
    }
}
```

第 92 章 EJB 组件

从 Camel 版本 2.4 开始提供

ejb: 组件将 EJB 绑定到 Camel 消息交换。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-ejb</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

92.1. URI 格式

```
ejb:ejbName[?options]
```

其中 ejbName 可以是用于在 Application Server JNDI Registry 中查找 EJB 的任何字符串

92.2. 选项

EJB 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
context (producer)	用于查找 EJB 的上下文		Context
properties (producer)	如果没有配置上下文，则创建 javax.naming.Context 的属性。		Properties
缓存 （高级）	如果启用，Camel 将缓存第一个 Registry 查找的结果。如果 Registry 中的 bean 定义为单例范围，则可以启用缓存。		布尔值
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

EJB 端点使用 URI 语法进行配置：

```
ejb:beanName
```

使用以下路径和查询参数：

92.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
beanName	**必需** 设置要调用的 bean 的名称		字符串

92.2.2. 查询参数(5 参数)：

Name	描述	默认值	类型
method (producer)	设置要在 bean 上调用的方法名称		字符串
缓存 （高级）	如果启用，Camel 将缓存第一个 Registry 查找的结果。如果 Registry 中的 bean 定义为单例范围，则可以启用缓存。		布尔值
multiParameterArray (advanced)	**弃用的** How to treat 处理从消息 body.true 传递的参数，表示消息正文应该是参数数组。弃用备注：此选项由 Camel 内部使用，不适用于最终用户使用。弃用备注：此选项由 Camel 内部使用，不适用于最终用户使用。	false	布尔值
参数 (advanced)	用于在 bean 中配置附加属性		Map
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

92.3. BEAN BINDING

要调用的 bean 方法是如何被选择（如果没有通过 method 参数明确指定），以及从 Message 构建的参数值是如何由 Camel 中所有不同 Bean 集成机制使用的 Bean Binding 机制定义的。

92.4. 例子

在以下示例中，我们使用定义如下的 Greater EJB：

GreaterLocal.java

```
public interface GreaterLocal {

    String hello(String name);

    String bye(String name);

}
```

和实施

GreaterImpl.java

```
@Stateless
public class GreaterImpl implements GreaterLocal {

    public String hello(String name) {
        return "Hello " + name;
    }

    public String bye(String name) {
        return "Bye " + name;
    }

}
```

92.4.1. 使用 Java DSL

在本例中，我们将要在 EJB 上调用 hello 方法。由于本示例基于使用 Apache Openejb 的单元测试，我们必须使用 Openejb 设置在 EJB 组件上设置 JndiContext。

```
@Override
protected CamelContext createCamelContext() throws Exception {
    CamelContext answer = new DefaultCamelContext();

    // enlist EJB component using the JndiContext
    EjbComponent ejb = answer.getComponent("ejb", EjbComponent.class);
    ejb.setContext(createEjbContext());

    return answer;
}
```

```
private static Context createEjbContext() throws NamingException {
    // here we need to define our context factory to use OpenEJB for our testing
    Properties properties = new Properties();
    properties.setProperty(Context.INITIAL_CONTEXT_FACTORY,
"org.apache.openejb.client.LocalInitialContextFactory");

    return new InitialContext(properties);
}
```

然后，我们已准备好在 Camel 路由中使用 EJB：

```
from("direct:start")
    // invoke the greeter EJB using the local interface and invoke the hello method
    .to("ejb:GreaterImplLocal?method=hello")
    .to("mock:result");
```

在实际的应用服务器中

在实际的应用服务器中，您可能不必在 EJB 组件上设置 JndiContext，因为它将在与应用服务器相同的 JVM 上创建默认的 JndiContext，这通常允许它访问 JNDI 注册表并查找 EJB。但是，如果您需要访问远程 JVM 或类似的应用服务器，您必须事先准备属性。

92.4.2. 使用 Spring XML

这是使用 Spring XML 的同一示例：

同样，这基于单元测试，因此我们需要设置 EJB 组件：

```
<!-- setup Camel EJB component -->
<bean id="ejb" class="org.apache.camel.component.ejb.EjbComponent">
    <property name="properties" ref="jndiProperties"/>
</bean>

<!-- use OpenEJB context factory -->
<p:properties id="jndiProperties">
    <prop
key="java.naming.factory.initial">org.apache.openejb.client.LocalInitialContextFactory</prop>
</p:properties>
```

在 Camel 路由中使用 EJB 之前：

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
    <route>
```

```
        <from uri="direct:start"/>
        <to uri="ejb:GreaterImplLocal?method=hello"/>
        <to uri="mock:result"/>
    </route>
</camelContext>
```

92.5. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- Bean
- Bean Binding
- Bean 集成

第 93 章 ELASTICSEARCH 组件（已弃用）

从 Camel 版本 2.11 开始提供

ElasticSearch 组件允许您与 ElasticSearch 服务器进行接口。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-elasticsearch</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

93.1. URI 格式

```
elasticsearch://clusterName[?options]
```

93.2. 端点选项

Elasticsearch 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
客户端 (advanced)	使用现有的配置的 Elasticsearch 客户端，而不是为每个端点创建客户端。		客户端
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Elasticsearch 端点使用 URI 语法进行配置：

```
elasticsearch:clusterName
```

使用以下路径和查询参数：

93.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
clusterName	**集群所需的** 名称，或使用 local 用于本地模式		字符串

93.2.2. 查询参数(11 参数)：

Name	描述	默认值	类型
clientTransportSniff (producer)	是允许 sniff 集群的其余部分或非监控（默认为 true）的客户端。此设置映射到 client.transport.sniff 设置。	true	布尔值
consistencyLevel (producer)	用于 INDEX 和 BULK 操作的写入一致性级别（可以是 ONE、QUORUM、All 或 DEFAULT）	DEFAULT	WriteConsistencyLevel
data (producer)	是允许为其分配数据（分片）的节点。此设置映射到 node.data 设置。		布尔值
indexName (producer)	要针对的索引的名称		字符串
indexType (producer)	要针对的索引类型		字符串
ip (producer)	要使用的 TransportClient 远程主机 ip		字符串
operation (producer)	要执行的操作		字符串
pathHome (producer)	ElasticSearch 配置的 path.home 属性。您需要提供有效的路径，否则将使用默认值 $user.home/.elasticsearch。	${user.home}/.elasticsearch	字符串
port (producer)	要使用的 TransportClient 远程端口（默认为 9300）	9300	int
transportAddresses (producer)	以逗号分隔的带有 ip:port 格式的远程传输地址的列表。ip 和 port 选项必须留空，才能考虑 transportAddresses。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

93.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.elasticsearch.client	使用现有的配置的 Elasticsearch 客户端，而不是为每个端点创建客户端。选项是一个 org.elasticsearch.client.Client 类型。		字符串
camel.component.elasticsearch.enabled	启用 elasticsearch 组件	true	布尔值
camel.component.elasticsearch.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

93.4. 本地测试

如果要针对本地（在 JVM/classloader）Elastic 服务器运行，只需将 URI 中的 clusterName 值设置为 "local"。如需了解更多详细信息，请参阅 客户端指南。

93.5. 消息操作

目前支持以下 ElasticSearch 操作。只需设置端点 URI 选项或交换标头，其键为"operation"，值设为以下之一：有些操作还需要设置其他参数或消息正文。

operation	消息正文	description
INDEX	将, String, byte[] 或 XContentBuilder 内容映射到索引	将内容添加到索引中，并在正文中返回内容的 indexId。**Camel 2.15，** 您可以通过使用键 "indexId" 设置消息标头来设置 indexId。
GET_BY_ID	要检索的内容的索引 ID	检索指定的索引，并在正文中返回一个 GetResult 对象

operation	消息正文	description
DELETE	要删除的内容的索引 ID	删除指定的 indexId，并在正文中返回一个 DeleteResult 对象
BULK_INDEX	已接受的任何类型的 **列表** 或 **集合** (XContentBuilder, Map, byte[], String)	*Camel 2.14,* 将内容添加到索引中，并返回正文中成功索引文档的 id 列表
BULK	已接受的任何类型的 **列表** 或 **集合** (XContentBuilder, Map, byte[], String)	**Camel 2.15：** 将内容添加到索引中，并返回正文中的 BulkResponse 对象
搜索	map 或 SearchRequest Object	**Camel 2.15：** 使用查询字符串映射搜索内容
MULTIGET	MultigetRequest.Item 对象列表	**Camel 2.17:** 检索指定的索引、类型等，在 MultigetRequest 中返回正文中的 MultigetResponse 对象
MULTISEARCH	SearchRequest 对象列表	**Camel 2.17:** 搜索 MultiSearchRequest 中指定的参数，并在正文中返回一个 MultiSearchResponse 对象
EXISTS	索引名称作为标头	**Camel 2.17:** 在正文中返回一个布尔值对象

operation	消息正文	description
更新	map, String, byte[] 或 XContentBuilder 内容以更新	Camel 2.17: 将内容更新至索引，并返回正文中的内容 indexId。

93.6. 索引示例

以下是一个简单的 INDEX 示例

```
from("direct:index")
.to("elasticsearch://local?operation=INDEX&indexName=twitter&indexType=tweet");
```

```
<route>
  <from uri="direct:index" />
  <to uri="elasticsearch://local?operation=INDEX&indexName=twitter&indexType=tweet"/>
</route>
```

客户端只需要将包含映射的正文消息传递给路由。结果正文包含创建的 indexId。

```
Map<String, String> map = new HashMap<String, String>();
map.put("content", "test");
String indexId = template.requestBody("direct:index", map, String.class);
```

93.7. 如需更多信息，请参阅这些资源

Elasticsearch 主站点

ElasticSearch Java API

93.8. 另请参阅

- 配置 Camel

- *组件*
- *端点*
- *开始使用*

第 94 章 ELASTICSEARCH5 组件（已弃用）

从 Camel 版本 2.19 开始提供

ElasticSearch 组件允许您使用 ElasticSearch 5.x API 进行接口。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-elasticsearch5</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

94.1. URI 格式

```
elasticsearch5://clusterName[?options]
```

94.2. 端点选项

Elasticsearch5 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
客户端 (advanced)	使用现有的配置的 Elasticsearch 客户端，而不是为每个端点创建客户端。这允许使用特定设置自定义客户端。		TransportClient
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Elasticsearch5 端点使用 URI 语法进行配置：

```
elasticsearch5:clusterName
```

使用以下路径和查询参数：

94.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
clusterName	**集群所需的** 名称		字符串

94.2.2. 查询参数(16 参数)：

Name	描述	默认值	类型
clientTransportSniff (producer)	是允许对群集的其余部分进行嗅探的客户端。此设置映射到 client.transport.sniff 设置。	false	布尔值
indexName (producer)	要针对的索引的名称		字符串
indexType (producer)	要针对的索引类型		字符串
ip (producer)	要使用的 TransportClient 远程主机 ip		字符串
operation (producer)	要执行的操作		ElasticsearchOperation
pingSchedule (producer)	客户端 ping 集群的时间（单位）。	5s	字符串
pingTimeout (producer)	来自节点的 ping 响应的时间（单位）手动返回。	5s	字符串
port (producer)	要使用的 TransportClient 远程端口（默认为 9300）	9300	int
tcpCompress (producer)	如果所有节点之间都启用了压缩(LZF)，则为 true。	false	布尔值
tcpConnectTimeout (producer)	等待连接超时的时间（单位）	30s	字符串
transportAddresses (producer)	以逗号分隔的带有 ip:port 格式的远程传输地址的列表。ip 和 port 选项必须留空，才能考虑 transportAddresses。		字符串
waitForActiveShards (producer)	索引创建等待分片的写入一致性数量可用	1	int
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

Name	描述	默认值	类型
启用SSL (security)	启用 SSL。在类路径上需要 XPack 客户端 jar	false	布尔值
密码 （身份验证）	用于对集群进行身份验证的密码。在类路径上需要 XPack 客户端 jar		字符串
用户 （身份验证）	用于对集群进行身份验证的用户。需要 transport_client 角色才能访问集群。在类路径上需要 XPack 客户端 jar		字符串

94.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.elasticsearch5.client	使用现有的配置的 Elasticsearch 客户端，而不是为每个端点创建客户端。这允许使用特定设置自定义客户端。选项是一个 org.elasticsearch.client.transport.TransportClient 类型。		字符串
camel.component.elasticsearch5.enabled	启用 elasticsearch5 组件	true	布尔值
camel.component.elasticsearch5.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

94.4. 消息操作

目前支持以下 ElasticSearch 操作。只需设置端点 URI 选项或交换标头，其键为"operation"，值设为以下之一：有些操作还需要设置其他参数或消息正文。

operation	消息正文	description

operation	消息正文	description
INDEX	将, String, byte[] 或 XContentBuilder 内容映射到索引	将内容添加到索引中，并在正文中返回内容的 indexId。您可以使用键 "indexId" 设置消息标头来设置 indexId。
GET_BY_ID	要检索的内容的索引 ID	检索指定的索引，并在正文中返回一个 GetResult 对象
DELETE	要删除的索引名称和类型	删除指定的 indexName 和 indexType，并返回正文中的 DeleteResponse 对象
DELETE_INDEX	要删除的内容的索引名称	删除指定的 indexName，并在正文中返回一个 DeleteIndexResponse 对象
BULK_INDEX	已接受的任何类型的 **列表** 或 **集合** (XContentBuilder, Map, byte[], String)	将内容添加到索引中，并返回正文中成功索引文档的 id 列表
BULK	已接受的任何类型的 **列表** 或 **集合** (XContentBuilder, Map, byte[], String)	将内容添加到索引中，并返回正文中的 BulkResponse 对象

operation	消息正文	description
搜索	Map, String 或 SearchRequest Object	使用查询字符串映射搜索内容
MULTIGET	MultigetRequest.Item 对象列表	检索指定的索引、类型等，在 MultigetRequest 中返回正文中的 MultigetResponse 对象
MULTISEARCH	SearchRequest 对象列表	搜索 MultiSearchRequest 中指定的参数，并在正文中返回一个 MultiSearchResponse 对象
EXISTS	索引名称作为标头	检查索引是否存在或不返回正文中的布尔值标志
更新	map, String, byte[] 或 XContentBuilder 内容以更新	将内容更新为索引，并返回正文中内容的 indexId。

94.5. 索引示例

以下是一个简单的 INDEX 示例

```
from("direct:index")
.to("elasticsearch5://elasticsearch?operation=INDEX&indexName=twitter&indexType=tweet");
```

```
<route>
  <from uri="direct:index" />
  <to uri="elasticsearch5://elasticsearch?operation=INDEX&indexName=twitter&indexType=tweet"/>
</route>
```

客户端只需要将包含映射的正文消息传递给路由。结果正文包含创建的 indexId。

```
Map<String, String> map = new HashMap<String, String>();
map.put("content", "test");
String indexId = template.requestBody("direct:index", map, String.class);
```

94.6. 如需更多信息，请参阅这些资源

Elastic Main Site

ElasticSearch Java API

94.7. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 95 章 ELASTICHSEARCH REST 组件

从 Camel 版本 2.21 开始提供

ElasticSearch 组件允许您使用 REST 客户端库与 ElasticSearch 6.x API 进行接口。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-elasticsearch-rest</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

95.1. URI 格式

```
elasticsearch-rest://clusterName[?options]
```

95.2. 端点选项

Elastichsearch Rest 组件支持 12 个选项，如下所列。

Name	描述	默认值	类型
客户端 (advanced)	使用现有的配置的 Elasticsearch 客户端，而不是为每个端点创建客户端。这允许使用特定设置自定义客户端。		RESTClient
hostAddresses (advanced)	以逗号分隔的带有 ip:port 格式的远程传输地址的列表。ip 和 port 选项必须留空，才能考虑 hostAddresses。		字符串
socketTimeout (advanced)	套接字超时前等待的超时时间。	30000	int
connectionTimeout (advanced)	在连接超时前等待的时间(ms)。	30000	int
用户 （安全）	基本验证用户		字符串
密码 (security)	用于验证的密码		字符串

Name	描述	默认值	类型
启用SSL (security)	启用 SSL	false	布尔值
maxRetryTimeout (advanced)	重试前的 ms 的时间	30000	int
enableSniffer (advanced)	启用从正在运行的 Elasticsearch 集群中自动发现节点	false	布尔值
snifferInterval (advanced)	连续的普通嗅探执行间隔（以毫秒为单位）。当 sniffOnFailure 被禁用或连续的 sniffOnFailure 之间没有故障时，将隐藏	300000	int
sniffAfterFailureDelay (advanced)	失败后调度的 sniff 执行的延迟（以毫秒为单位）	60000	int
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Elastichsearch Rest 端点使用 URI 语法进行配置：

```
elasticsearch-rest:clusterName
```

使用以下路径和查询参数：

95.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
clusterName	**集群所需的** 名称		字符串

95.2.2. 查询参数(11 参数)：

Name	描述	默认值	类型
connectionTimeout (producer)	在连接超时前等待的时间(ms)。	30000	int

Name	描述	默认值	类型
disconnect (producer)	在调用制作者完成后断开连接	false	布尔值
enableSSL (producer)	启用 SSL	false	布尔值
hostAddresses (producer)	**必需的** Comma separated list with ip:port formatted remote transport address use.		字符串
indexName (producer)	要针对的索引的名称		字符串
indexType (producer)	要针对的索引类型		字符串
maxRetryTimeout (producer)	重试前的 ms 的时间	30000	int
operation (producer)	要执行的操作		ElasticsearchOperation
socketTimeout (producer)	套接字超时前等待的超时时间。	30000	int
waitForActiveShards (producer)	索引创建等待分片的写入一致性数量可用	1	int
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

95.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 13 个选项，如下所列。

Name	描述	默认值	类型
camel.component.elasticsearch-rest.client	使用现有的配置的 Elasticsearch 客户端，而不是为每个端点创建客户端。这允许使用特定设置自定义客户端。选项是一个 org.elasticsearch.client.RestClient 类型。		字符串

Name	描述	默认值	类型
camel.component.elasticsearch-rest.connection-timeout	在连接超时前等待的时间(ms)。	30000	整数
camel.component.elasticsearch-rest.enable-s-s-l	启用 SSL	false	布尔值
camel.component.elasticsearch-rest.enable-sniffer	启用从正在运行的 Elasticsearch 集群中自动发现节点	false	布尔值
camel.component.elasticsearch-rest.enabled	是否启用 elasticsearch-rest 组件的自动配置。这默认是启用的。		布尔值
camel.component.elasticsearch-rest.host-addresses	以逗号分隔的带有 ip:port 格式的远程传输地址的列表。ip 和 port 选项必须留空，才能考虑 hostAddresses。		字符串
camel.component.elasticsearch-rest.max-retry-timeout	重试前的 ms 的时间	30000	整数
camel.component.elasticsearch-rest.password	用于验证的密码		字符串
camel.component.elasticsearch-rest.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.elasticsearch-rest.sniff-after-failure-delay	失败后调度的 sniff 执行的延迟（以毫秒为单位）	60000	整数
camel.component.elasticsearch-rest.sniffer-interval	连续的普通嗅探执行间隔（以毫秒为单位）。当 sniffOnFailure 被禁用或连续的 sniffOnFailure 之间没有故障时，将隐藏	300000	整数

Name	描述	默认值	类型
camel.component.elasticsearch-rest.socket-timeout	套接字超时前等待的超时时间。	30000	整数
camel.component.elasticsearch-rest.user	基本验证用户		字符串

95.4. 消息操作

目前支持以下 ElasticSearch 操作。只需设置端点 URI 选项或交换标头，其键为"operation"，值设为以下之一：有些操作还需要设置其他参数或消息正文。

operation	消息正文	description
索引	将 ,**String**, **byte[]**, **XContentBuilder** 或 **IndexRequest** 内容 **映射到** index	将内容添加到索引中，并在正文中返回内容的 indexId。您可以使用键 "indexId" 设置消息标头来设置 indexId。
GetById	要检索的内容的字符串或 **GetRequest** index id	检索指定的索引，并在正文中返回一个 GetResult 对象
删除	**字符串** 或 **DeleteRequest** 索引名称和要删除的内容类型	删除指定的 indexName 和 indexType，并返回正文中的 DeleteResponse 对象

operation	消息正文	description
DeleteIndex	要删除的索引的字符串或 **DeleteRequest** 索引名称	删除指定的 indexName 并返回正文状态代码
BulkIndex	已接受的任何类型的 **列表**、**BulkRequest** 或 **Collection**（XContentBuilder、Map、byte[]、String）	将内容添加到索引中，并返回正文中成功索引文档的 id 列表
bulk	已接受的任何类型的 **列表**、**BulkRequest** 或 **Collection**（XContentBuilder、Map、byte[]、String）	将内容添加到索引中，并返回正文中的 BulkItemResponse[] 对象
搜索	**映射**,String 或 **SearchRequest**	使用查询字符串映射搜索内容

operation	消息正文	description
MultiSearch	**MultiSearchRequest**	一个中的多个搜索
Exists	索引名称 (indexName) 作为标头	检查索引是否存在或不返回正文中的布尔值标志
Update（更新）	**映射,UpdateRequest,String,byte[] 或 XContentBuilder** 内容以更新	将内容更新为索引，并返回正文中内容的 indexId。
ping	None	对远程 Elasticsearch 集群发出 ping 命令，如果 ping 成功返回 true，否则为 false
info	None	获取 Elasticsearch 集群的信息，并将它们返回为 MainResponse 类实例

95.5. 配置组件并启用基本身份验证

要使用 Elasticsearch 组件，必须配置最小配置。

```
ElasticsearchComponent elasticsearchComponent = new ElasticsearchComponent();
elasticsearchComponent.setHostAddresses("myelkhost:9200");
camelContext.addComponent("elasticsearch-rest", elasticsearchComponent);
```

对于使用 elasticsearch 的基本身份验证，或在 elasticsearch 集群前面使用反向代理代理，只需在组件上设置基本身份验证和 SSL，如下例所示

```
ElasticsearchComponent elasticsearchComponent = new ElasticsearchComponent();
elasticsearchComponent.setHostAddresses("myelkhost:9200");
elasticsearchComponent.setUser("elkuser");
elasticsearchComponent.setPassword("secure!!");
```

```
elasticsearchComponent.setEnableSSL(true);

camelContext.addComponent("elasticsearch-rest", elasticsearchComponent);
```

95.6. 索引示例

以下是一个简单的 INDEX 示例

```
from("direct:index")
  .to("elasticsearch-rest://elasticsearch?
operation=Index&indexName=twitter&indexType=tweet");
```

```
<route>
    <from uri="direct:index" />
    <to uri="elasticsearch-rest://elasticsearch?
operation=Index&indexName=twitter&indexType=tweet"/>
</route>
```

对于此操作，您需要指定一个 indexId 标头。

客户端只需要将包含映射的正文消息传递给路由。结果正文包含创建的 indexId。

```
Map<String, String> map = new HashMap<String, String>();
map.put("content", "test");
String indexId = template.requestBody("direct:index", map, String.class);
```

95.7. 搜索示例

搜索特定字段和值使用 Operation 'Search'。传递查询 JSON 字符串或映射

```
from("direct:search")
  .to("elasticsearch-rest://elasticsearch?
operation=Search&indexName=twitter&indexType=tweet");
```

```
<route>
    <from uri="direct:search" />
    <to uri="elasticsearch-rest://elasticsearch?
operation=Search&indexName=twitter&indexType=tweet"/>
</route>
```

```
String query = "{\"query\":{\"match\":{\"content\":\"new release of ApacheCamel\"}}}";
SearchHits response = template.requestBody("direct:search", query, SearchHits.class);
```

使用 Map 搜索特定字段。

```
Map<String, Object> actualQuery = new HashMap<>();
actualQuery.put("content", "new release of ApacheCamel");

Map<String, Object> match = new HashMap<>();
match.put("match", actualQuery);

Map<String, Object> query = new HashMap<>();
query.put("query", match);
SearchHits response = template.requestBody("direct:search", query, SearchHits.class);
```

95.8. MULTISEARCH 示例

MultiSearching on specific field (s)和 value 使用 Operation 'MultiSearch'。传递 MultiSearchRequest 实例

```
from("direct:multiSearch")
  .to("elasticsearch-rest://elasticsearch?operation=MultiSearch");
```

```
<route>
    <from uri="direct:multiSearch" />
    <to uri="elasticsearch-rest://elasticsearch?operation=MultiSearch"/>
</route>
```

MultiSearch on specific field (s)

```
SearchRequest req = new SearchRequest();
req.indices("twitter");
req.types("tweet");
SearchRequest req1 = new SearchRequest();
req.indices("twitter");
req.types("tweets");
MultiSearchRequest request = new MultiSearchRequest().add(req1).add(req);
Item[] response = template.requestBody("direct:search", request, Item[].class);
```

第 96 章 ELSQL COMPONENT

从 Camel 版本 2.16 开始提供

elsql: 组件是现有 SQL 组件的扩展，使用 ElSql 定义 SQL 查询。

此组件在后台使用 spring-jdbc 进行实际的 SQL 处理。

此组件可用作 事务客户端。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-elsql</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

SQL 组件使用以下端点 URI 表示法：

```
sql:elSqlName:resourceUri[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

SQL 查询的参数在 elsql 映射文件中命名参数，并在给定优先级中映射到 Camel 消息中的对应键：

1. Camel 2.16.1： 如果 简单 表达式，来自消息正文。
2. 如果来自消息标头的 'java.util.Map'3.

如果无法解析指定参数，则抛出异常。

96.1. 选项

ElSQL 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
databaseVendor (common)	使用特定于供应商的 com.opengamma.elsql.ElSqlConfig		ElSqlDatabaseVendor
DataSource (common)	设置 DataSource 用于与数据库通信。		DataSource
elSqlConfig (advanced)	使用特定的配置的 ElSqlConfig。最好使用 databaseVendor 选项。		ElSqlConfig
resourceUri (common)	包含要使用的 elsql SQL 语句的资源文件。您可以指定用逗号分开的多个资源。默认情况下，资源在 classpath 上载入，您可以使用 file: 前缀来从文件系统中加载。请注意，您可以在组件上设置这个选项，然后您不必在端点上配置这个选项。		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

ElSQL 端点使用 URI 语法进行配置：

```
elsql:elsqlName:resourceUri
```

使用以下路径和查询参数：

96.1.1. 路径参数(2 参数)：

Name	描述	默认值	类型
elsqlName	***必需*** 要使用的 elsql 的名称(elsql 文件中的 NAMED)		字符串
resourceUri	包含要使用的 elsql SQL 语句的资源文件。您可以指定用逗号分开的多个资源。默认情况下，资源在 classpath 上载入，您可以使用 file: 前缀来从文件系统中加载。请注意，您可以在组件上设置这个选项，然后您不必在端点上配置这个选项。		字符串

96.1.2. 查询参数(47 参数)：

Name	描述	默认值	类型
allowNamedParameters (common)	是否允许在查询中使用命名参数。	true	布尔值
databaseVendor (common)	使用特定于供应商的 com.opengamma.elsql.ElSqlConfig		ElSqlDatabaseVendor
DataSource (common)	设置 DataSource 用于与数据库通信。		DataSource
dataSourceRef (common)	**弃用** 将引用设置为从 registry 中查询的 DataSource，以用于与数据库通信。		字符串
outputClass (common)	指定在 outputType=SelectOne 时使用的完整软件包和类名称。		字符串
outputHeader (common)	将查询结果存储在标头中而不是消息正文。默认情况下，outputHeader == null，查询结果存储在消息正文中，消息正文中的任何现有内容都会被丢弃。如果设置了 outputHeader，则值将用作标头名称来存储查询结果，并保留原始消息正文。		字符串
outputType (common)	将使用者或制作者的输出设置为 SelectList 作为 Map 列表，或者选择One 作为单个 Java 对象：a）如果查询只有一个列，则返回 JDBC Column 对象。（如 SELECT COUNT （）FROM PROJECT 将会返回 Long object.b）如果查询具有多个列，则它将返回该结果的映射。然后，它将调用与列名称匹配的所有集合将查询结果转换为 Java bean 对象。It 将假定您的类具有默认的构造器来创建实例 with.d）。如果查询导致多个行，它会抛出一个非唯一结果异常。StreamList 会处理使用 Iterator 的查询结果的结果。这可与流模式的 Splitter EIP 一起使用，以流传输方式处理 ResultSet。	SelectList	SqlOutputType
分隔符 (common)	当使用命名参数，则在从消息正文（如果正文是 String 类型）获取参数值时使用分隔符。如果使用了命名参数，则使用 Map 类型。默认值为 comma	,	char
breakBatchOnConsumeFail (consumer)	如果 onConsume 失败，则设置为是否中断批处理。	false	布尔值

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
expectedUpdateCount (consumer)	设置预期的更新计数，以便在使用 onConsume 时进行验证。	-1	int
maxMessagesPerPoll (consumer)	设置要轮询的最大消息数		int
onConsume (consumer)	处理每行后，可以执行此查询，如果 Exchange 成功处理，例如将行标记为已处理。查询可以有参数。		字符串
onConsumeBatchComplete (consumer)	处理整个批处理后，可以执行此查询以批量更新行等。查询不能有参数。		字符串
onConsumeFailed (consumer)	处理每行后，可以执行此查询，如果 Exchange 失败，例如将行标记为失败。查询可以有参数。		字符串
routeEmptyResultSet (consumer)	设置是否允许空 resultset 发送到下一跃点。默认为 false。因此，空结果集将被过滤掉。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
transacted (consumer)	启用或禁用事务。如果启用，如果处理交换失败，则消费者破坏了任何进一步交换的处理，从而导致回滚。	false	布尔值
useIterator (consumer)	设置如何将 resultset 传送到 route。将交付表示为列表或单个对象。默认为 true。	true	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern

Name	描述	默认值	类型
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
processingStrategy (consumer)	当消费者处理 rows/batch 时，允许插件使用自定义 org.apache.camel.component.sql.SqlProcessingStrategy 执行查询。		SqlProcessingStrategy
batch (producer)	启用或禁用批处理模式	false	布尔值
noop (producer)	如果设置，将忽略 SQL 查询的结果，并使用现有 IN 消息作为继续处理的 OUT 消息	false	布尔值
useMessageBodyForSql (producer)	是否将消息正文用作 SQL，然后是参数的标头。如果启用了这个选项，则不会使用 uri 中的 SQL。	false	布尔值
alwaysPopulateStatement (advanced)	如果启用了，则始终调用 org.apache.camel.component.sql.SqlPrepareStatementStrategy 的 populateStatement 方法，也如果没有准备预期的参数。当这是 false 时，只有在设置了 1 个或多个预期参数时，才会调用 populateStatement；例如，这样可避免读取没有参数的 SQL 查询的消息正文/标题。	false	布尔值
elSqlConfig (advanced)	使用特定的配置的 ElSqlConfig。最好使用 databaseVendor 选项。		ElSqlConfig
parametersCount (advanced)	如果设置大于零，则 Camel 将使用此计数的参数值替换，而不是通过 JDBC 元数据 API 查询。如果 JDBC 供应商无法返回正确的参数数，则这非常有用，用户可能会改为覆盖。		int
占位符 （高级）	指定在 SQL 查询中替换到的字符。请注意，它只是一个简单的 String.replaceAll （）操作，且不会涉及 SQL 解析（加引号的字符串也会改变）。	#	字符串
prepareStatementStrategy (advanced)	允许插件使用自定义 org.apache.camel.component.sql.SqlPrepareStatementStrategy 来控制查询和准备语句的准备。		SqlPrepareStatement 策略
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
templateOptions (advanced)	使用映射中的键/值配置 Spring JdbcTemplate		Map

Name	描述	默认值	类型
usePlaceholder (advanced)	设置是否使用占位符，并将所有占位符字符替换为 SQL 查询中的 符号。	true	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit

Name	描述	默认值	类型
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

96.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
camel.component.elsql.data-source	设置 DataSource 用于与数据库通信。选项是 javax.sql.DataSource 类型。		字符串
camel.component.elsql.database-vendor	使用特定于供应商的 com.opengamma.elsql.ElSqlConfig		ElSqlDatabaseVendor
camel.component.elsql.el-sql-config	使用特定的配置的 ElSqlConfig。最好使用 databaseVendor 选项。选项是一个 com.opengamma.elsql.ElSqlConfig 类型。		字符串
camel.component.elsql.enabled	启用 elsql 组件	true	布尔值
camel.component.elsql.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.elsql.resource-uri	包含要使用的 elsql SQL 语句的资源文件。您可以指定用逗号分开的多个资源。默认情况下，资源在 classpath 上载入，您可以使用 file: 前缀来从文件系统中加载。请注意，您可以在组件上设置这个选项，然后您不必在端点上配置这个选项。		字符串

96.3. 查询的结果

对于 选择 操作，结果是 List<Map<String, Object>& gt; 类型的实例，如 JdbcTemplate.queryForList （）方法返回。对于 更新操作，结果是更新的行数，返回为 Integer。

默认情况下，结果放置在消息正文中。如果设置了 outputHeader 参数，则结果将放在标头中。这是使用完整消息增强模式来添加标头的替代选择，它为查询序列或其它小值提供了一个简洁的语法。最好一起使用 outputHeader 和 outputType ：

96.4. 标头值

在执行更新操作时，SQL 组件将更新计数存储在以下消息标头中：

标头	描述
CamelSqlUpdateCount	更新更新操作的行数，返回为 Integer 对象。
CamelSqlRowCount	选择 操作返回的行数，返回为 Integer 对象。

96.4.1. 示例

在下面的给定路由中，我们想要从 projects 表中获取所有项目。请注意，SQL 查询具有 2 个命名参数，：#lic 和 :#min。

然后，Camel 将从消息正文或消息标头中查找这些参数。请注意，在上面的示例中，我们为命名参数设置两个带有恒定值
的标头：

```
from("direct:projects")
  .setHeader("lic", constant("ASF"))
  .setHeader("min", constant(123))
  .to("elsql:projects:com/foo/orders.elsql")
```

和 elsql 映射文件

```
@NAME(projects)
  SELECT *
  FROM projects
  WHERE license = :lic AND id > :min
  ORDER BY id
```

尽管消息正文是 java.util.Map，但命名的参数将从正文中获取。

```
from("direct:projects")
  .to("elsql:projects:com/foo/orders.elsql")
```

96.5. 在生成者中使用表达式参数

在 Camel 2.16.1 中，您还可以使用 Simple 表达式，它允许使用 OGNL，如消息正文上的标记，其中假设具有 getLicense 和 getMinimum 方法：

```
@NAME(projects)
  SELECT *
  FROM projects
  WHERE license = :${body.license} AND id > :${body.minimum}
  ORDER BY id
```

96.5.1. 在消费者中使用表达式参数

从 Camel 2.23 开始提供

当使用 ElSql 组件作为消费者时，您现在可以使用表达式参数（简单语言）构建动态查询参数，如在 bean 上调用方法来检索 id、date 或 something。

例如，在以下示例中，我们调用 bean myIdGenerator 中的 nextId 方法：

```
@NAME(projectsByIdBean)
  SELECT *
  FROM projects
  WHERE id = :${bean#myIdGenerator.nextId}
```

重要

注意在上面的 bean 语法中，必须在简单表达式中使用 # 而不是 :。这是因为 Spring 查询参数解析器正在使用中，它将在冒号上分隔参数。另外请注意，Spring 查询解析器将为每个查询调用 bean 两次。

和 bean 具有以下方法：

```
public static class MyIdGenerator {

  private int id = 1;
```

```
public int nextId() {
    // spring will call this twice, one for initializing query and 2nd for actual value
    id++;
    return id / 2;
}
```

请注意，没有带有消息正文和标头的现有 Exchange，因此您可以在消费者中使用的简单表达式最可用于调用 bean 方法，如本例中所示。

第 97 章 ETCD 组件

从 Camel 版本 2.18 开始提供

camel etcd 组件允许您使用 Etcd，它是一个分布式的键值存储。

97.1. URI 格式

```
etcd:namespace/path[?options]
```

97.2. URI 选项

etcd 组件支持 7 个选项，如下所列。

Name	描述	默认值	类型
URI (common)	要设置客户端连接的 URI，请执行以下操作：		字符串
sslContextParameters (common)	使用 SSLContextParameters 配置安全性。		SSLContextParameters
username (common)	用于基本身份验证的用户名。		字符串
password (common)	用于基本身份验证的密码。		字符串
配置 （高级）	设置端点之间共享的通用配置		EtcdConfiguration
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

etcd 端点使用 URI 语法进行配置：

```
etcd:namespace/path
```

使用以下路径和查询参数：

97.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
namespace	**必需** 使用的 API 命名空间		EtcdNamespace
path	端点引用的路径		字符串

97.2.2. 查询参数(29 参数)：

Name	描述	默认值	类型
递归 (common)	要以递归方式应用操作：	false	布尔值
servicePath (common)	查找服务发现的路径	/services/	字符串
timeout (common)	要设置操作可以完成的最长时间。		Long
URI (common)	要设置客户端连接的 URI，请执行以下操作：	http://localhost:2379,http://localhost:4001	字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
sendEmptyExchangeOnTimeout (consumer)	要在出现超时监视密钥时发送空消息。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值

Name	描述	默认值	类型
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
fromIndex (consumer)	要监视的索引	0	Long
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
timeToLive (producer)	以毫秒为单位设置键的期限。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long

Name	描述	默认值	类型
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
密码 (security)	用于基本身份验证的密码。		字符串
sslContextParameters (security)	使用 SSLContextParameters 配置安全性。		SSLContextParameters
用户名 (security)	用于基本身份验证的用户名。		字符串

97.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 17 个选项，如下所列。

Name	描述	默认值	类型
camel.component.etcd.configuration.from-index	要监视的索引	0	Long
camel.component.etcd.configuration.password	用于基本身份验证的密码。		字符串

Name	描述	默认值	类型
camel.component.etcd.configuration.recursive	要以递归方式应用操作：	false	布尔值
camel.component.etcd.configuration.send-empty-exchange-on-timeout	要在出现超时监视密钥时发送空消息。	false	布尔值
camel.component.etcd.configuration.service-path	查找服务发现的路径	/services/	字符串
camel.component.etcd.configuration.ssl-context-parameters	使用 SSLContextParameters 配置安全性。		SSLContextParameters
camel.component.etcd.configuration.time-to-live	以毫秒为单位设置键的期限。		整数
camel.component.etcd.configuration.timeout	要设置操作可以完成的最长时间。		Long
camel.component.etcd.configuration.uris	要设置客户端连接的 URI，请执行以下操作：	http://localhost:2379,http://localhost:4001	字符串
camel.component.etcd.configuration.user-name	用于基本身份验证的用户名。		字符串
camel.component.etcd.enabled	启用 etcd 组件	true	布尔值
camel.component.etcd.password	用于基本身份验证的密码。		字符串
camel.component.etcd.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Name	描述	默认值	类型
camel.component.etcd.ssl-context-parameters	使用 SSLContextParameters 配置安全性。选项是 org.apache.camel.util.jsse.SSLContextParameters 类型。		字符串
camel.component.etcd.uris	要设置客户端连接的 URI，请执行以下操作：		字符串
camel.component.etcd.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值
camel.component.etcd.user-name	用于基本身份验证的用户名。		字符串

第 98 章 OSGI EVENTADMIN COMPONENT

从 Camel 版本 2.6 开始提供

eventadmin 组件可以在 OSGi 环境中使用，以接收 OSGi EventAdmin 事件并处理它们。

98.1. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml中

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-eventadmin</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 ${camel-version} 必须替换为 Camel 的实际版本(2.6.0 或更高版本)。

98.2. URI 格式

```
eventadmin:topic[?options]
```

其中 topic 是要侦听的主题的名称。

98.3. URI 选项

OSGi EventAdmin 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
bundleContext (common)	OSGi BundleContext 由 Camel 自动注入		BundleContext
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

OSGi EventAdmin 端点使用 URI 语法进行配置：

```
eventadmin:topic
```

使用以下路径和查询参数：

98.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
topic	要侦听或发送到的主题名称		字符串

98.3.2. 查询参数(5 参数)：

Name	描述	默认值	类型
send (common)	是否使用 'send' 还是 'synchronous'。默认错误(async 交付)	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

98.4. 消息标头

Name	类型	消息
描述		

98.5. 消息正文

消息正文 中的 将设置为接收的事件。

98.6. 用法示例

```
<route>
    <from uri="eventadmin:*"/>
    <to uri="stream:out"/>
</route>
```

第 99 章 EXEC 组件

从 Camel 版本 2.3 开始提供

exec 组件可用于执行系统命令。

99.1. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml中

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-exec</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 ${camel-version} 必须替换为 Camel 的实际版本(2.3.0 或更高版本)。

99.2. URI 格式

```
exec://executable[?options]
```

其中 executable 是要执行的系统命令的名称或文件路径。如果使用可执行名称（如 exec:java），则可执行文件必须在系统路径中。

99.3. URI 选项

Exec 组件没有选项。

Exec 端点使用 URI 语法进行配置：

```
exec:executable
```

使用以下路径和查询参数：

99.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
可执行	**必需** 设置要执行的可执行文件。可执行文件不能为空或为空。		字符串

99.3.2. 查询参数(8 参数)：

Name	描述	默认值	类型
args (producer)	参数可以是一个或多个空格分隔的令牌。		字符串
binding (producer)	对 Registry 中的 org.apache.commons.exec.ExecBinding 的引用。		ExecBinding
commandExecutor (producer)	对 registry 中自定义命令执行的 org.apache.commons.exec.ExecCommandExecutor 的引用。默认命令 executor 使用 commons-exec 库，它会为每个执行的命令添加一个 shutdown hook。		ExecCommandExecutor
outfile (producer)	由可执行文件创建的文件的名称，应被视为其输出。如果没有设置 outFile，则会使用可执行文件的标准输出(stdout)。		字符串
timeout (producer)	应该终止可执行文件的超时时间（以毫秒为单位）。如果在超时时间内还没有完成执行，则组件将发送终止请求。		long
useStderrOnEmptyStdout (producer)	指示 stdout 为空的布尔值，此组件将填充 Camel 消息正文和 stderr。默认禁用此行为(false)。	false	布尔值
workingDir (producer)	应该在其中执行命令的目录。如果为 null，则使用当前进程的工作目录。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

99.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.exec.enabled	启用 exec 组件	true	布尔值

Name	描述	默认值	类型
camel.component.exec.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

99.5. 消息标头

支持的标头在 org.apache.camel.component.exec.ExecBinding 中定义。

Name	类型	消息	描述
ExecBinding.EXEC_COMMAND_EXECUTABLE	**字符串**	**in**	将要执行的系统命令的名称。覆盖 URI 中的 **可执行文件**。
ExecBinding.EXEC_COMMAND_ARGS	**java.util.List<String>**	**in**	传递给执行进程的命令行参数。参数按字面使用 - 不应用引用。覆盖 **URI** 中的任何现有参数。
ExecBinding.EXEC_COMMAND_ARGS	**字符串**	**in**	Camel 2.5： 可执行文件的参数作为单一字符串，其中每个参数都是空格分开的（请参阅 URI 选项中的 **args** ）。参数按字面使用，不会应用引用。覆盖 **URI** 中的任何现有参数。
ExecBinding.EXEC_COMMAND_OUT_FILE	**字符串**	**in**	由可执行文件创建的文件的名称，应被视为其输出。覆盖 URI 中的任何现有 **outFile**。

Name	类型	消息	描述
ExecBinding.EXEC_COMMAND_TIMEOUT	**long**	**in**	应该终止可执行文件的超时时间（以毫秒为单位）。覆盖 URI 中的任何现有 **超时**。
ExecBinding.EXEC_COMMAND_WORKING_DIR	**字符串**	**in**	应该在其中执行命令的目录。覆盖 URI 中的任何现有 **workingDir**。
ExecBinding.EXEC_EXIT_VALUE	**int**	**out**	此标头的值是可执行文件的 *退出值*。非零退出值通常表示异常终止。请注意，退出值与 OS 无关。
ExecBinding.EXEC_STDERR	**java.io.InputStream**	**out**	此标头的值指向可执行文件的标准错误流(stderr)。如果没有写入 stderr，则值为 **null**。
ExecBinding.EXEC_USE_STDERR_ON_EMPTY_STDOUT	**布尔值**	**in**	表示 **stdout** 为空时，此组件将使用 **stderr** 填充 Camel 消息正文。默认情况下禁用此行为(**false**)。

99.6. 消息正文

如果 Exec 组件收到 可以转换为 java.io.InputStream 的消息正文中的消息正文，它用于通过 stdin 将输入源到可执行文件。执行后，消息正文 是执行的结果，即 org.apache.camel.components.exec.ExecResult 实例，其中包含 stdout、stderr、exit 值和 out 文件。为方便起见，这个组件支持以下 ExecResult 类型转换器 ：

from	to
ExecResult	java.io.InputStream
ExecResult	字符串
ExecResult	byte []
ExecResult	org.w3c.dom.Document

如果指定了 out 文件（通过 outFile 指定端点或通过 ExecBinding.EXEC_COMMAND_OUT_FILE）返回 out 文件的内容。如果没有使用文件，则此组件会将进程的 stdout 转换为目标类型。详情请参考下面的 用法示例。

99.7. 使用示例

99.7.1. 执行字计数(Linux)

以下示例执行 wc （字数、Linux）来统计文件 /usr/share/dict/words 中的词语。单词 count （输出）写入 wc 的标准输出流。

```
from("direct:exec")
.to("exec:wc?args=--words /usr/share/dict/words")
.process(new Processor() {
    public void process(Exchange exchange) throws Exception {
      // By default, the body is ExecResult instance
      assertIsInstanceOf(ExecResult.class, exchange.getIn().getBody());
      // Use the Camel Exec String type converter to convert the ExecResult to String
      // In this case, the stdout is considered as output
      String wordCountOutput = exchange.getIn().getBody(String.class);
      // do something with the word count
    }
});
```

99.7.2. 执行 java

以下示例执行带有 2 参数： -server 和 -version 的 java，只要 java 在系统路径中。

```
from("direct:exec")
.to("exec:java?args=-server -version")
```

以下示例在 c:\temp 中使用 3 参数执行 java ： -server,-version 和 sytem property user.name。

```
from("direct:exec")
.to("exec:c:/program files/jdk/bin/java?args=-server -version -
Duser.name=Camel&workingDir=c:/temp")
```

99.7.3. 执行 Ant 脚本

以下示例使用构建文件 CamelExecBuildFile.xml 执行 Apache Ant （仅限 Windows），只要 ant.bat 在系统路径中，并且 CamelExecBuildFile.xml 位于当前目录中。

```
from("direct:exec")
.to("exec:ant.bat?args=-f CamelExecBuildFile.xml")
```

在下一个示例中，t.bat 命令将其输出重定向到 带有 -l 的 CamelExecOutFile.txt。文件 CamelExecOutFile.txt 用作 outFile=CamelExecOutFile.txt 文件。该示例假定 ant.bat 位于系统路径中，并且 CamelExecBuildFile.xml 位于当前目录中。

```
from("direct:exec")
.to("exec:ant.bat?args=-f CamelExecBuildFile.xml -l
CamelExecOutFile.txt&outFile=CamelExecOutFile.txt")
.process(new Processor() {
    public void process(Exchange exchange) throws Exception {
      InputStream outFile = exchange.getIn().getBody(InputStream.class);
      assertIsInstanceOf(InputStream.class, outFile);
      // do something with the out file here
    }
  });
```

99.7.4. 执行 echo (Windows)

echo 和 dir 等命令只能通过操作系统的命令解释器来执行。本例演示了如何在 Windows 中执行此类命令 - echo -。

```
from("direct:exec").to("exec:cmd?args=/C echo echoString")
```

99.8. 另请参阅

- 配置 Camel
- 组件

- *端点*
- *开始使用*

第 100 章 FACEBOOK COMPONENT

从 Camel 版本 2.14 开始提供

Facebook 组件提供使用 Facebook4J 访问所有bookbook API 的访问权限。它允许生成消息来检索、添加和删除文章，如注释、评论、图片、视频、照片、检查位置、链接等。它还支持 API，允许轮询后期、用户、检查、组、位置等。

Facebook 需要对所有客户端应用程序身份验证使用 OAuth。为了在您的账户中使用 camel-facebook，您需要在 https://developers.facebook.com/apps 的 Facebook 中创建新应用程序，并授予应用程序对您的帐户的访问权限。bookbook 应用程序的 id 和 secret 允许访问在无需当前用户的情况下，Facebook API。需要登录用户的 API 需要用户访问令牌。有关获取用户访问令牌的更多信息，请访问 https://developers.facebook.com/docs/facebook-login/access-tokens/。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-facebook</artifactId>
  <version>${camel-version}</version>
</dependency>
```

100.1. URI 格式

```
facebook://[endpoint]?[options]
```

100.2. FACEBOOKCOMPONENT

facebook 组件可以通过Facebook 帐户设置进行配置，这是强制性的。这些值可以使用类型为 org.apache.camel.component.facebook.config.FacebookConfiguration 的 bean 属性配置 提供给组件。oAuthAccessToken 选项可能是 ommited，它将只允许访问应用程序 API。

bookbook 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	使用共享配置		FacebookConfiguration

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

bookbook 端点使用 URI 语法进行配置：

```
facebook:methodName
```

使用以下路径和查询参数：

100.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
methodName	**需要** 执行什么操作		字符串

100.2.2. 查询参数(102 参数)：

Name	描述	默认值	类型
achievementURL (common)	实现的唯一 URL		URL
albumId (common)	album ID		字符串
albumUpdate (common)	要创建或更新的 facebook Album		AlbumUpdate
appId (common)	Facebook 应用程序的 ID		字符串
center (common)	位置 latitude 和 longitude		地理位置
checkinId (common)	检查 ID		字符串
checkinUpdate (common)	**弃用** 要创建的检查。弃用，而是创建带有附加位置的 Post		CheckinUpdate

Name	描述	默认值	类型
clientURL (common)	acebook4J API 客户端 URL		字符串
clientVersion (common)	Facebook4J 客户端 API 版本		字符串
commentId (common)	注释 ID		字符串
commentUpdate (common)	要创建或更新的 facebook Comment		CommentUpdate
debugEnabled (common)	启用 deubg 输出。只适用于嵌入的日志记录器	false	布尔值
description (common)	描述文本		字符串
distance (common)	量表中的距离		整数
domainId (common)	域 ID		字符串
domainName (common)	域名		字符串
domainNames (common)	域名		list
eventId (common)	事件 ID		字符串
eventUpdate (common)	创建或更新的事件		EventUpdate
friendId (common)	friend ID		字符串
friendlistId (common)	friend 列表 ID		字符串
friendlistName (common)	friend 列表名称		字符串
friendUserId (common)	friend 用户 ID		字符串

Name	描述	默认值	类型
GroupId (common)	组 ID		字符串
gzipEnabled (common)	使用Facebook GZIP 编码	true	布尔值
httpConnectionTimeout (common)	HTTP 连接超时（以毫秒为单位）	20000	整数
httpDefaultMaxPerRoute (common)	每个路由的 HTTP 最大连接	2	整数
httpMaxTotalConnections (common)	HTTP 最大连接总数	20	整数
httpReadTimeout (common)	HTTP 读取超时（以毫秒为单位）	120000	整数
httpRetryCount (common)	HTTP 重试次数	0	整数
httpRetryIntervalSeconds (common)	HTTP 重试间隔（以秒为单位）	5	整数
httpStreamingReadTimeout (common)	HTTP 流读取超时（以毫秒为单位）	40000	整数
id (common)	用户的 ID		list
inBody (common)	设置要在交换 In Body 中传递的参数名称		字符串
includeRead (common)	启用用户未读取的通知		布尔值
isHidden (common)	隐藏		布尔值
jsonStoreEnabled (common)	如果设置为 true，则原始 JSON 表单将存储在 DataObjectFactory 中	false	布尔值
link (common)	链接 URL		URL

Name	描述	默认值	类型
linkId (common)	链接 ID		字符串
locale (common)	所需的 FQL 区域		Locale
mbeanEnabled (common)	如果设置为 true，Facebook4J mbean 将注册	false	布尔值
message (common)	消息文本		字符串
messageId (common)	消息 ID		字符串
指标 (common)	指标名称		字符串
milestoneId (common)	milestone id		字符串
name (common)	测试用户名，必须是格式为 'first last'		字符串
noteId (common)	请注意 ID		字符串
notificationId (common)	通知 ID		字符串
objectId (common)	insight 对象 ID		字符串
offerId (common)	优惠 ID		字符串
optionDescription (common)	问题的回答选项描述		字符串
pageId (common)	页面 ID		字符串
permissionName (common)	权限名称		字符串
权限 (common)	以 perm1,perm2,... 格式测试用户权限		字符串
photoId (common)	照片 ID		字符串
pictureId (common)	图片 ID		整数

Name	描述	默认值	类型
pictureId2 (common)	picture2 id		整数
pictureSize (common)	图片大小		PictureSize
placeId (common)	原位 ID		字符串
postId (common)	后 ID		字符串
postUpdate (common)	创建或更新的 post		PostUpdate
prettyDebugEnabled (common)	如果设为 true，则 Prettify JSON debug 输出	false	布尔值
query (common)	FQL 查询		Map
query (common)	FQL 查询或搜索搜索端点的术语		字符串
questionId (common)	问题 ID		字符串
Read (common)	可选读取参数。请参阅 Reading Options (#reading)		读取
readingOptions (common)	使用映射中的键/值对配置读取。		Map
restBaseURL (common)	API 基本 URL	https://graph.facebook.com/	字符串
scoreValue (common)	数字分数带有值		整数
size (common)	图片大小，大、普通、小或方括号之一		PictureSize
source (common)	来自 java.io.File 或 java.io.Inputstream 的介质内容		media
subject (common)	主题的备注		字符串
tabId (common)	选项卡 ID		字符串

Name	描述	默认值	类型
tagUpdate (common)	Photo 标签信息		TagUpdate
testUser1 (common)	测试用户 1		TestUser
testUser2 (common)	测试用户 2		TestUser
testUserId (common)	测试用户的 ID		字符串
title (common)	标题文本		字符串
toUserId (common)	要标记的用户 ID		字符串
toUserIds (common)	要标记的用户 ID		list
userId (common)	bookbook 用户 ID		字符串
userId1 (common)	用户 1 的 ID		字符串
userId2 (common)	用户 2 的 ID		字符串
userIds (common)	邀请事件的用户 ID		list
userLocale (common)	测试用户区域设置		字符串
useSSL (common)	使用 SSL	true	布尔值
videoBaseURL (common)	视频 API 基本 URL	https://graph-video.facebook.com/	字符串
videoId (common)	视频 ID		字符串

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
httpProxyHost (proxy)	HTTP 代理服务器主机名		字符串
httpProxyPassword (proxy)	HTTP 代理服务器密码		字符串
httpProxyPort (proxy)	HTTP 代理服务器端口		整数
httpProxyUser (proxy)	HTTP 代理服务器用户名		字符串
OAuthAccessToken (security)	用户访问令牌		字符串
oAuthAccessTokenURL (security)	OAuth 访问令牌 URL	https://graph.facebook.com/oauth/access_token	字符串
oAuthAppId (security)	应用程序 Id		字符串
oAuthAppSecret (security)	应用程序 Secret		字符串

Name	描述	默认值	类型
oAuthAuthorizationURL (security)	OAuth 授权 URL	https://www.facebook.com/dialog/oauth	字符串
oAuthPermissions (security)	默认 OAuth 权限。以逗号分隔的权限名称。详情请查看 https://developers.facebook.com/docs/reference/login/#permissions		字符串

100.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 29 个选项，如下所列。

Name	描述	默认值	类型
camel.component.facebook.configuration.client-u-r-l	acebook4J API 客户端 URL		字符串
camel.component.facebook.configuration.client-version	Facebook4J 客户端 API 版本		字符串
camel.component.facebook.configuration.debug-enabled	启用 deubg 输出。只适用于嵌入的日志记录器	false	布尔值
camel.component.facebook.configuration.gzip-enabled	使用Facebook GZIP 编码	true	布尔值
camel.component.facebook.configuration.http-connection-timeout	HTTP 连接超时（以毫秒为单位）	20000	整数

Name	描述	默认值	类型
camel.component.facebook.configuration.http-default-max-per-route	每个路由的 HTTP 最大连接	2	整数
camel.component.facebook.configuration.http-max-total-connections	HTTP 最大连接总数	20	整数
camel.component.facebook.configuration.http-proxy-host	HTTP 代理服务器主机名		字符串
camel.component.facebook.configuration.http-proxy-password	HTTP 代理服务器密码		字符串
camel.component.facebook.configuration.http-proxy-port	HTTP 代理服务器端口		整数
camel.component.facebook.configuration.http-proxy-user	HTTP 代理服务器用户名		字符串
camel.component.facebook.configuration.http-read-timeout	HTTP 读取超时（以毫秒为单位）	120000	整数
camel.component.facebook.configuration.http-retry-count	HTTP 重试次数	0	整数
camel.component.facebook.configuration.http-retry-interval-seconds	HTTP 重试间隔（以秒为单位）	5	整数

Name	描述	默认值	类型
camel.component.facebook.configuration.http-streaming-read-timeout	HTTP 流读取超时（以毫秒为单位）	40000	整数
camel.component.facebook.configuration.json-store-enabled	如果设置为 true，则原始 JSON 表单将存储在 DataObjectFactory 中	false	布尔值
camel.component.facebook.configuration.mbean-enabled	如果设置为 true，Facebook4J mbean 将注册	false	布尔值
camel.component.facebook.configuration.o-auth-access-token	用户访问令牌		字符串
camel.component.facebook.configuration.o-auth-access-token-u-r-l	OAuth 访问令牌 URL	https://graph.facebook.com/oauth/access_token	字符串
camel.component.facebook.configuration.o-auth-app-id	应用程序 Id		字符串
camel.component.facebook.configuration.o-auth-app-secret	应用程序 Secret		字符串
camel.component.facebook.configuration.o-auth-authorization-u-r-l	OAuth 授权 URL	https://www.facebook.com/dialog/oauth	字符串

Name	描述	默认值	类型
camel.component.facebook.configuration.o-auth-permissions	默认 OAuth 权限。以逗号分隔的权限名称。详情请查看 https://developers.facebook.com/docs/reference/login/#permissions		字符串
camel.component.facebook.configuration.pretty-debug-enabled	如果设为 true，则 Prettify JSON debug 输出	false	布尔值
camel.component.facebook.configuration.rest-base-u-r-l	API 基本 URL	https://graph.facebook.com/	字符串
camel.component.facebook.configuration.use-s-s-l	使用 SSL	true	布尔值
camel.component.facebook.configuration.video-base-u-r-l	视频 API 基本 URL	https://graph-video.facebook.com/	字符串
camel.component.facebook.enabled	启用 facebook 组件	true	布尔值
camel.component.facebook.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

100.4. 生产者端点：

生产者端点可以使用下表中的端点名称和选项。端点也可以使用不带 get 或 搜索 前缀的短名称，但由于 get Checkin 和 searchCheckin 之间的模糊检查。未强制的端点选项由 [] 表示。

生成者端点也可以使用特殊选项 inBody，它应包含端点选项的名称，其值将包含在 Camel Exchange In 消息中。例如，以下路由的 facebook 端点检索传入消息正文中用户 id 值的活动。

```
from("direct:test").to("facebook://activities?inBody=userId")...
```

任何端点选项都可以在端点 URI 中提供，或者在消息标头中动态提供。消息标头名称必须是 CamelFacebook.https://cwiki.apache.org/confluence/pages/createpage.action?spaceKey=CAMEL&title=option&linkCreation=true&fromPageId=34020899[option] 格式。例如，在消息标头 CamelFacebook. userId 中可能会提供上一个路由中的 userId 选项值。请注意，inBody 选项会覆盖消息标头，如 Body=user 中的端点 选项会覆盖 CamelFacebook.userId 标头。

返回字符串为创建或修改实体返回 Id 的端点，如 addAlbumPhoto 返回新的 album Id。返回布尔值的端点，如果成功返回 true，否则返回 false。如果是Facebook API 错误，端点将引发一个 RuntimeCamelException，并带有 facebook4j.FacebookException 原因。

100.5. 消费者端点：

所有取而成器 端点的制作者 端点都可以用作消费者端点。轮询使用者使用 since 和 until 字段在轮询间隔内获取响应。除了其他读取字段外，还可在 端点中为第一次轮询提供初始值。

camel-facebook 根据每个返回的对象创建一个路由交换，而不是通过一个路由交换返回列表（或 facebook）返回列表（或 facebook ）返回列表。例如，如果 "facebook://home" 结果为五条，则路由将执行五次（每个 Post 一次一次）。

100.6. 读取选项

类型的 read 选项 facebook4j.Reading 添加了对读取参数的支持，允许选择特定的字段，限制结果数量等。如需更多信息，请参阅 Graph API这个问题- Facebook Developers。

消费者端点也用于轮询 Facebook 数据，以避免在轮询之间发送重复的消息。

reading 选项可以是 type facebook4j.Reading 的参考或值，也可以使用端点 URI 中的以下读取选项来指定，或使用 CamelFacebook. 前缀交换标头来指定。

100.7. 消息标头

任何 URI 选项都可在 带有 CamelFacebook. 前缀的制作者端点的消息标头中提供。

100.8. 消息正文

所有结果消息正文都使用 Facebook4J API 提供的对象。生产者端点可以在 inBody 端点参数中指定传入消息正文的选项名称。

对于返回数组或 facebook4j.ResponseList 或 java.util.List 的端点，消费者端点会将列表中的每个元素映射到不同的消息。

100.9. 使用案例

要在Facebook 配置集中创建文章，请发送此制作者一个 facebook4j.PostUpdate 正文。

```
from("direct:foo")
    .to("facebook://postFeed/inBody=postUpdate);
```

要轮询，每 5 sec （您可以通过添加前缀 " consumer" 来设置轮询 消费者选项），您的主页源上的所有状态：

```
from("facebook://home?consumer.delay=5000")
    .to("bean:blah");
```

使用带有标头中动态选项的制作者进行搜索。

在条条标题中，我们有bookbook 搜索字符串，以便在公共文章中执行，因此我们需要为 CamelFacebook.query 标头分配这个值。

```
from("direct:foo")
    .setHeader("CamelFacebook.query", header("bar"))
    .to("facebook://posts");
```

第 101 章 FHIR 组件

从 Camel 版本 2.23 开始提供

FHIR 组件与 HAPI-FHIR 库集成，这是 Java 中 FHIR (Fast Healthcare Interoperability Resources) 规范的开源实现。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-fhir</artifactId>
  <version>${camel-version}</version>
</dependency>
```

101.1. URI 格式

FHIR 组件使用以下 URI 格式：

```
fhir://endpoint-prefix/endpoint?[options]
```

端点前缀可以是以下之一：

- 功能
- create
- delete
- history
- load-page

- *meta*
- *operation*
- *patch*
- *读取*
- *search*
- *transaction*
- *update*
- *validate*

FHIR 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	使用共享配置		FhirConfiguration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

FHIR 端点使用 URI 语法进行配置：

```
fhir:apiName/methodName
```

使用以下路径和查询参数：

101.1.1. 路径参数(2 参数)：

Name	描述	默认值	类型
apiName	**需要** 执行什么操作		FhirApiName
methodName	**必需的** 所选操作使用哪些子操作		字符串

101.1.2. 查询参数(26 参数)：

Name	描述	默认值	类型
encoding (common)	用于所有请求的编码		字符串
fhirVersion (common)	要使用的 FHIR 版本	DSTU3	字符串
inBody (common)	设置要在交换 In Body 中传递的参数名称		字符串
log (common)	将记录每个请求和响应	false	布尔值
prettyPrint (common)	用户打印所有请求	false	布尔值
serverUrl (common)	FHIR 服务器基本 URL		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
compress (advanced)	将传出(POST/PUT)内容压缩到 GZIP 格式	false	布尔值

Name	描述	默认值	类型
connectionTimeout (advanced)	尝试和建立初始 TCP 连接的时间（以 ms 为单位）	10000	整数
deferModelScanning (advanced)	设置此选项后，在实际访问给定类型的子列表之前，不会扫描模型类。	false	布尔值
fhirContext (advanced)	FhirContext 是一个昂贵的创建对象。为避免创建多个实例，可以直接设置它。		FhirContext
forceConformanceCheck (advanced)	强制检查	false	布尔值
sessionCookie (advanced)	要添加到每个请求的 HTTP 会话 Cookie		字符串
socketTimeout (advanced)	阻止单个读/写操作的时长（以 ms 为单位）	10000	整数
Summary (advanced)	请求服务器使用 _summary 参数修改响应		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
validationMode (advanced)	当 Camel 应该验证 FHIR 服务器一致性语句时	ONCE	字符串
proxyHost (proxy)	代理主机		字符串
proxyPassword (proxy)	代理密码		字符串
proxyPort (proxy)	代理端口		整数
proxyUser (proxy)	代理用户名		字符串
accessToken (security)	OAuth 访问令牌		字符串
密码 (security)	用于基本身份验证的用户名		字符串
用户名 (security)	用于基本身份验证的用户名		字符串

101.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 23 个选项，如下所列。

Name	描述	默认值	类型
camel.component.fhir.configuration.access-token	OAuth 访问令牌		字符串
camel.component.fhir.configuration.api-name	要执行的操作类型		FhirApiName
camel.component.fhir.configuration.client	使用自定义客户端		IGenericClient
camel.component.fhir.configuration.client-factory	使用自定义客户端工厂		IRestfulClientFactory
camel.component.fhir.configuration.compress	将传出(POST/PUT)内容压缩到 GZIP 格式	false	布尔值
camel.component.fhir.configuration.connection-timeout	尝试和建立初始 TCP 连接的时间（以 ms 为单位）	10000	整数
camel.component.fhir.configuration.defer-model-scanning	设置此选项后，在实际访问给定类型的子列表之前，不会扫描模型类。	false	布尔值
camel.component.fhir.configuration.fhir-context	FhirContext 是一个昂贵的创建对象。为避免创建多个实例，可以直接设置它。		FhirContext
camel.component.fhir.configuration.force-conformance-check	强制检查	false	布尔值
camel.component.fhir.configuration.log	将记录每个请求和响应	false	布尔值

Name	描述	默认值	类型
camel.component.fhir.configuration.method-name	用于所选操作的子操作		字符串
camel.component.fhir.configuration.password	用于基本身份验证的用户名		字符串
camel.component.fhir.configuration.pretty-print	用户打印所有请求	false	布尔值
camel.component.fhir.configuration.proxy-host	代理主机		字符串
camel.component.fhir.configuration.proxy-password	代理密码		字符串
camel.component.fhir.configuration.proxy-port	代理端口		整数
camel.component.fhir.configuration.proxy-user	代理用户名		字符串
camel.component.fhir.configuration.server-url	FHIR 服务器基本 URL		字符串
camel.component.fhir.configuration.session-cookie	要添加到每个请求的 HTTP 会话 Cookie		字符串
camel.component.fhir.configuration.socket-timeout	阻止单个读/写操作的时长（以 ms 为单位）	10000	整数
camel.component.fhir.configuration.username	用于基本身份验证的用户名		字符串
camel.component.fhir.enabled	是否启用 fhir 组件的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.component.fhir.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 102 章 FHIR JSON DATAFORMAT

从 Camel 版本 2.21开始，提供 Camel 版本 2.21

FHIR-JSON 数据格式利用 HAPI-FHIR 的 JSON 解析器来解析到/从 JSON 格式解析到/从 HAPI-FHIR 的 IBaseResource 中解析。

102.1. FHIR JSON 格式选项

FHIR JSon dataformat 支持 14 个选项，如下所列。

Name	默认值	Java 类型	描述
fhirVersion	DSTU3	字符串	要使用的 FHIR 版本。可能的值有：DSTU2,DSTU2_HL7ORG,DSTU2_1,DSTU3,R4
prettyPrint	false	布尔值	设置用户友善打印标志，这意味着解析器将使用元素间人类可读的空间和换行符对资源进行编码，而不是尽可能地压缩输出。
serverBaseUrl		字符串	设置此解析器使用的服务器基本 URL。如果设置了值，则资源引用将转换为相对引用（如果它们作为绝对 URL 提供），但基本与给定基础匹配。
omitResourceId	false	布尔值	如果设置为 true （默认为 false），则被编码的任何资源的 ID 都不会包含在输出中。请注意，这不适用于包含资源，仅适用于根资源。换句话说，如果设为 true，则包含的资源仍将具有本地 ID，但外部/包含 ID 不会具有 ID。
encodeElementsAppliesToResourceTypes		Set	如果提供，告诉解析哪个资源类型以应用链接 #setEncodeElements (Set)编码元素。这里没有指定的资源类型都会被完全编码，但没有排除元素。
encodeElementsAppliesToChildResourcesOnly	false	布尔值	如果设置为 true （默认为 false），则提供给 setEncodeElements (Set)的值不会应用到根资源（通常是 Bundle），而是应用到该捆绑包中包含的任何子资源（例如，在该捆绑包中的搜索结果）
encodeElements		Set	如果提供，请指定应编码的元素，以排除所有其他内容。此字段的有效值包括： Patient - Encode patient 及其子级 Patient.name - Encode 只包括病人的名称 Patient.name.family - Encode only the patient's family name .text - Encode the text element on any resource (only the first position may contains a wildcard). （必需） - 导致任何强制字段（只在 0 编码）进行编码。

Name	默认值	Java 类型	描述
dontEncodeElements		Set	如果提供，指定不应编码的元素。此字段的有效值包括： Patient - Don't encode to and its children Patient.name - Don't encode the patient's name Patient.name.family - Don't encode the program's family name .text - Don't encode the text element on any resource (only the first position may contains a wildcard) DSTU2 备注： 请注意，包括 meta 的值，如 Patient.meta 将适用于 DSTU2 解析器，但带有元数据上的子元素的值（如 Patient.meta.lastUpdated 只会在 DSTU3 模式下工作）。
stripVersionsFromReferences	false	布尔值	如果设置为 true （默认值），则包含版本的资源引用将在资源被编码时删除版本。这通常很好，因为在大多数情况下，从一个资源到另一个资源的引用应该通过 ID 而不是 ID 和版本来指向资源。然而，在某些情况下，可能需要在资源链接中保留版本。在这种情况下，这个值应设置为 false。这个方法提供了全局禁用引用编码的功能。如果需要更精细的控制，请使用 setDontStripVersionsFromReferencesAtPaths (List)
overrideResourceIdWithBundleEntryFullUrl	false	布尔值	如果设置为 true （默认值），Bundle.entry.fullUrl 将覆盖 Bundle.entry.resource 的资源 ID （如果定义了 fullUrl）。当将源数据解析为 Bundle 对象时，会发生此行为。如果这不是所需的行为（例如，客户端代码希望在 fullUrl 和资源 ID 之间执行额外的验证检查），则将其设置为 false。
summaryMode	false	布尔值	如果设置为 true （默认为 false），则只会包含由 FHIR 规格标记为 summary 元素的元素。
suppressNarratives	false	布尔值	如果设置为 true （默认为 false），则不会将 narratives 包含在编码的值中。
dontStripVersionsFromReferencesAtPaths		list	如果提供的值，则指定路径中的任何资源引用都会编码其资源版本，而不是在编码过程中自动剥离。此设置对解析过程没有影响。这个方法提供比 setStripVersionsFromReferences (Boolean)提供的更精细的控制级别，即使 setStripVersionsFromReferences (Boolean)已设置为 true （这是默认设置）
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

102.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 15 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.fhirjson.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.fhirjson.dont-encode-elements	如果提供，指定不应编码的元素。此字段的有效值包括： Patient - Don't encode to and its children Patient.name - Don't encode the patient's name Patient.name.family - Don't encode the program's family name .text - Don't encode the text element on any resource (only the first position may contains a wildcard) DSTU2 备注： 请注意，包括 meta 的值，如 Patient.meta 将适用于 DSTU2 解析器，但带有元数据上的子元素的值（如 Patient.meta.lastUpdated 只会在 DSTU3 模式下工作）。		Set
camel.dataformat.fhirjson.dont-strip-versions-from-references-at-paths	如果提供的值，则指定路径中的任何资源引用都会编码其资源版本，而不是在编码过程中自动剥离。此设置对解析过程没有影响。这个方法提供比 setStripVersionsFromReferences (Boolean)提供的更精细的控制级别，即使 setStripVersionsFromReferences (Boolean)已设置为 true （这是默认设置）		list
camel.dataformat.fhirjson.enabled	是否启用 fhirJson 数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.fhirjson.encode-elements	如果提供，请指定应编码的元素，以排除所有其他内容。此字段的有效值包括： Patient - Encode patient 及其子级 Patient.name - Encode 只包括病人的名称 Patient.name.family - Encode only the patient's family name .text - Encode the text element on any resource (only the first position may contains a wildcard). （必需）- 导致任何强制字段（只在 0 编码）进行编码。		Set
camel.dataformat.fhirjson.encode-elements-applies-to-child-resources-only	如果设置为 true （默认为 false），则提供给 setEncodeElements (Set)的值不会应用到根资源（通常是 Bundle），而是应用到该捆绑包中包含的任何子资源（例如，在该捆绑包中的搜索结果）	false	布尔值
camel.dataformat.fhirjson.encode-elements-applies-to-resource-types	如果提供，告诉解析哪个资源类型以应用链接 #setEncodeElements (Set)编码元素。这里没有指定的资源类型都会被完全编码，但没有排除元素。		Set

Name	描述	默认值	类型
camel.dataformat.fhirjson.fhir-version	要使用的 FHIR 版本。可能的值有：DSTU2,DSTU2_HL7ORG,DSTU2_1,DSTU3,R4	DSTU3	字符串
camel.dataformat.fhirjson.omit-resource-id	如果设置为 true （默认为 false），则被编码的任何资源的 ID 都不会包含在输出中。请注意，这不适用于包含资源，仅适用于根资源。换句话说，如果设为 true，则包含的资源仍将具有本地 ID，但外部/包含 ID 不会具有 ID。	false	布尔值
camel.dataformat.fhirjson.override-resource-id-with-bundle-entry-full-url	如果设置为 true （默认值），Bundle.entry.fullUrl 将覆盖 Bundle.entry.resource 的资源 ID （如果定义了 fullUrl）。当将源数据解析为 Bundle 对象时，会发生此行为。如果这不是所需的行为（例如，客户端代码希望在 fullUrl 和资源 ID 之间执行额外的验证检查），则将其设置为 false。	false	布尔值
camel.dataformat.fhirjson.pretty-print	设置用户友善打印标志，这意味着解析器将使用元素间人类可读的空间和换行符对资源进行编码，而不是尽可能地压缩输出。	false	布尔值
camel.dataformat.fhirjson.server-base-url	设置此解析器使用的服务器基本 URL。如果设置了值，则资源引用将转换为相对引用（如果它们作为绝对 URL 提供），但基本与给定基础匹配。		字符串
camel.dataformat.fhirjson.strip-versions-from-references	如果设置为 true （默认值），则包含版本的资源引用将在资源被编码时删除版本。这通常很好，因为在大多数情况下，从一个资源到另一个资源的引用应该通过 ID 而不是 ID 和版本来指向资源。然而，在某些情况下，可能需要在资源链接中保留版本。在这种情况下，这个值应设置为 false。这个方法提供了全局禁用引用编码的功能。如果需要更精细的控制，请使用 setDontStripVersionsFromReferencesAtPaths (List)	false	布尔值
camel.dataformat.fhirjson.summary-mode	如果设置为 true （默认为 false），则只会包含由 FHIR 规格标记为 summary 元素的元素。	false	布尔值
camel.dataformat.fhirjson.suppress-narratives	如果设置为 true （默认为 false），则不会将 narratives 包含在编码的值中。	false	布尔值

第 103 章 FHIR XML DATAFORMAT

从 Camel 版本 2.21开始，提供 Camel 版本 2.21

FHIR-XML 数据格式利用 HAPI-FHIR 的 XML 解析器向 HAPI-FHIR 的 IBaseResource 解析/从 XML 格式解析到/从 XML 格式进行解析。

103.1. FHIR XML 格式选项

FHIR XML 格式支持 14 个选项，如下所列。

Name	默认值	Java 类型	描述
fhirVersion	DSTU3	字符串	要使用的 FHIR 版本。可能的值有：DSTU2,DSTU2_HL7ORG,DSTU2_1,DSTU3,R4
prettyPrint	false	布尔值	设置用户友善打印标志，这意味着解析器将使用元素间人类可读的空间和换行符对资源进行编码，而不是尽可能地压缩输出。
serverBaseUrl		字符串	设置此解析器使用的服务器基本 URL。如果设置了值，则资源引用将转换为相对引用（如果它们作为绝对 URL 提供），但基本与给定基础匹配。
omitResourceId	false	布尔值	如果设置为 true （默认为 false），则被编码的任何资源的 ID 都不会包含在输出中。请注意，这不适用于包含资源，仅适用于根资源。换句话说，如果设为 true，则包含的资源仍将具有本地 ID，但外部/包含 ID 不会具有 ID。
encodeElementsAppliesToResourceTypes		Set	如果提供，告诉解析哪个资源类型以应用链接 #setEncodeElements (Set)编码元素。这里没有指定的资源类型都会被完全编码，但没有排除元素。
encodeElementsAppliesToChildResourcesOnly	false	布尔值	如果设置为 true （默认为 false），则提供给 setEncodeElements (Set)的值不会应用到根资源（通常是 Bundle），而是应用到该捆绑包中包含的任何子资源（例如，在该捆绑包中的搜索结果）
encodeElements		Set	如果提供，请指定应编码的元素，以排除所有其他内容。此字段的有效值包括： Patient - Encode patient 及其子级 Patient.name - Encode 只包括病人的名称 Patient.name.family - Encode only the patient's family name .text - Encode the text element on any resource (only the first position may contains a wildcard). （必需） - 导致任何强制字段（只在 0 编码）进行编码。

Name	默认值	Java 类型	描述
dontEncodeElements		Set	如果提供，指定不应编码的元素。此字段的有效值包括： Patient - Don't encode to and its children Patient.name - Don't encode the patient's name Patient.name.family - Don't encode the program's family name .text - Don't encode the text element on any resource (only the first position may contains a wildcard) DSTU2 备注： 请注意，包括 meta 的值，如 Patient.meta 将适用于 DSTU2 解析器，但带有元数据上的子元素的值（如 Patient.meta.lastUpdated 只会在 DSTU3 模式下工作）。
stripVersionsFromReferences	false	布尔值	如果设置为 true （默认值），则包含版本的资源引用将在资源被编码时删除版本。这通常很好，因为在大多数情况下，从一个资源到另一个资源的引用应该通过 ID 而不是 ID 和版本来指向资源。然而，在某些情况下，可能需要在资源链接中保留版本。在这种情况下，这个值应设置为 false。这个方法提供了全局禁用引用编码的功能。如果需要更精细的控制，请使用 setDontStripVersionsFromReferencesAtPaths (List)
overrideResourceIdWithBundleEntryFullUrl	false	布尔值	如果设置为 true （默认值），Bundle.entry.fullUrl 将覆盖 Bundle.entry.resource 的资源 ID （如果定义了 fullUrl）。当将源数据解析为 Bundle 对象时，会发生此行为。如果这不是所需的行为（例如，客户端代码希望在 fullUrl 和资源 ID 之间执行额外的验证检查），则将其设置为 false。
summaryMode	false	布尔值	如果设置为 true （默认为 false），则只会包含由 FHIR 规格标记为 summary 元素的元素。
suppressNarratives	false	布尔值	如果设置为 true （默认为 false），则不会将 narratives 包含在编码的值中。
dontStripVersionsFromReferencesAtPaths		list	如果提供的值，则指定路径中的任何资源引用都会编码其资源版本，而不是在编码过程中自动剥离。此设置对解析过程没有影响。这个方法提供比 setStripVersionsFromReferences (Boolean)提供的更精细的控制级别，即使 setStripVersionsFromReferences (Boolean)已设置为 true （这是默认设置）
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

103.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 15 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.fhirxml.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.fhirxml.dont-encode-elements	如果提供，指定不应编码的元素。此字段的有效值包括： Patient - Don't encode to and its children Patient.name - Don't encode the patient's name Patient.name.family - Don't encode the program's family name .text - Don't encode the text element on any resource (only the first position may contains a wildcard) DSTU2 备注： 请注意，包括 meta 的值，如 Patient.meta 将适用于 DSTU2 解析器，但带有元数据上的子元素的值（如 Patient.meta.lastUpdated 只会在 DSTU3 模式下工作）。		Set
camel.dataformat.fhirxml.dont-strip-versions-from-references-at-paths	如果提供的值，则指定路径中的任何资源引用都会编码其资源版本，而不是在编码过程中自动剥离。此设置对解析过程没有影响。这个方法提供比 setStripVersionsFromReferences (Boolean)提供的更精细的控制级别，即使 setStripVersionsFromReferences (Boolean)已设置为 true （这是默认设置）		list
camel.dataformat.fhirxml.enabled	是否启用 fhirXml 数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.fhirxml.encode-elements	如果提供，请指定应编码的元素，以排除所有其他内容。此字段的有效值包括： Patient - Encode patient 及其子级 Patient.name - Encode 只包括病人的名称 Patient.name.family - Encode only the patient's family name .text - Encode the text element on any resource (only the first position may contains a wildcard). （必需）- 导致任何强制字段（只在 0 编码）进行编码。		Set
camel.dataformat.fhirxml.encode-elements-applies-to-child-resources-only	如果设置为 true （默认为 false），则提供给 setEncodeElements (Set)的值不会应用到根资源（通常是 Bundle），而是应用到该捆绑包中包含的任何子资源（例如，在该捆绑包中的搜索结果）	false	布尔值
camel.dataformat.fhirxml.encode-elements-applies-to-resource-types	如果提供，告诉解析哪个资源类型以应用链接 #setEncodeElements (Set)编码元素。这里没有指定的资源类型都会被完全编码，但没有排除元素。		Set

Name	描述	默认值	类型
camel.dataformat.fhirxml.fhir-version	要使用的 FHIR 版本。可能的值有：DSTU2,DSTU2_HL7ORG,DSTU2_1,DSTU3,R4	DSTU3	字符串
camel.dataformat.fhirxml.omit-resource-id	如果设置为 true （默认为 false），则被编码的任何资源的 ID 都不会包含在输出中。请注意，这不适用于包含资源，仅适用于根资源。换句话说，如果设为 true，则包含的资源仍将具有本地 ID，但外部/包含 ID 不会具有 ID。	false	布尔值
camel.dataformat.fhirxml.override-resource-id-with-bundle-entry-full-url	如果设置为 true （默认值），Bundle.entry.fullUrl 将覆盖 Bundle.entry.resource 的资源 ID （如果定义了 fullUrl）。当将源数据解析为 Bundle 对象时，会发生此行为。如果这不是所需的行为（例如，客户端代码希望在 fullUrl 和资源 ID 之间执行额外的验证检查），则将其设置为 false。	false	布尔值
camel.dataformat.fhirxml.pretty-print	设置用户友善打印标志，这意味着解析器将使用元素间人类可读的空间和换行符对资源进行编码，而不是尽可能地压缩输出。	false	布尔值
camel.dataformat.fhirxml.server-base-url	设置此解析器使用的服务器基本 URL。如果设置了值，则资源引用将转换为相对引用（如果它们作为绝对 URL 提供），但基本与给定基础匹配。		字符串
camel.dataformat.fhirxml.strip-versions-from-references	如果设置为 true （默认值），则包含版本的资源引用将在资源被编码时删除版本。这通常很好，因为在大多数情况下，从一个资源到另一个资源的引用应该通过 ID 而不是 ID 和版本来指向资源。然而，在某些情况下，可能需要在资源链接中保留版本。在这种情况下，这个值应设置为 false。这个方法提供了全局禁用引用编码的功能。如果需要更精细的控制，请使用 setDontStripVersionsFromReferencesAtPaths (List)	false	布尔值
camel.dataformat.fhirxml.summary-mode	如果设置为 true （默认为 false），则只会包含由 FHIR 规格标记为 summary 元素的元素。	false	布尔值
camel.dataformat.fhirxml.suppress-narratives	如果设置为 true （默认为 false），则不会将 narratives 包含在编码的值中。	false	布尔值

第 104 章 文件组件

从 Camel 版本 1.0 开始提供

文件组件提供对文件系统的访问，允许由任何其他 Camel 组件或来自其他组件的消息处理文件到磁盘。

104.1. URI 格式

```
file:directoryName[?options]
```

or

```
file:///directoryName[?options]
```

其中 directoryName 代表底层文件目录。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

仅目录

Camel 仅支持通过起始目录配置的端点。因此，directoryName 必须是目录。
如果只想使用单个文件，您可以使用 fileName 选项，例如通过设置 fileName=thefilename。
另外，起始目录不得包含带有 $\{ } 占位符的动态表达式。再次使用 fileName 选项指定文件名的动态部分。

警告

避免读取当前由另一个应用程序写入的文件，使 JDK File IO API 在检测另一个应用程序当前是否正在写入/复制文件时受到某种限制。此外，实施可能还视操作系统平台而不同。这可能会导致 Camel 认为文件不会被另一个进程锁定，并开始使用该文件。因此，您必须自行调查您的环境套件。为了帮助此 Camel 提供不同的 readLock 选项和 doneFileName 选项，您可以使用它们。另请参阅 节，其中存储了其他人直接丢弃文件的文件夹。

104.2. URI 选项

File 组件没有选项。

File 端点使用 URI 语法进行配置：

```
file:directoryName
```

使用以下路径和查询参数：

104.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
directoryName	**所需的** 起始目录		File

104.2.2. 查询参数(87 参数)：

Name	描述	默认值	类型
charset (common)	此选项用于指定文件的编码。您可以在消费者上使用此选项，指定文件的编码，允许 Camel 了解在访问文件内容时应加载文件内容。在编写文件时，您可以使用此选项指定写入该文件的 charset。请记住，在编写文件 Camel 时，可能需要将消息内容读取到内存中，以便能将数据转换为配置的 charset，因此如果您有大型消息，请不要使用它。		字符串

Name	描述	默认值	类型
doneFileName (common)	生产者：如果提供，则 Camel 将在写入原始文件时写入第 2 个 done 文件。完成的文件将为空。这个选项配置要使用的文件名。您可以指定一个固定名称。或者，您可以使用动态占位符。完成的文件始终将写入与原始文件相同的文件夹中。consumer：如果提供，Camel 仅在文件存在时使用文件。这个选项配置要使用的文件名。您可以指定一个固定名称。或者您可以使用动态占位符。已完成的文件始终预期位于与原始文件相同的文件夹中。仅支持 $file.name 和 $file.name.noext 作为动态占位符。		字符串
filename (common)	使用 File Language 等表达式来动态设置文件名。对于消费者，它将用作文件名过滤器。对于生成者，它用于评估要写入的文件名。如果设置了表达式，它将优先于 CamelFileName 标头。（注：标题本身也可以是表达式）。表达式选项支持 String 和 Expression 类型。如果表达式是 String 类型，则始终使用 File 语言来评估它。如果表达式是 Expression 类型，则使用指定的 Expression 类型 - 例如，允许您使用 OGNL 表达式。对于消费者，您可以使用它过滤文件名，因此您可以使用 File Language 语法使用当前的文件：mydata-$date:now:yyyyMMdd.txt。生产者支持 CamelOverruleFileName 标头，它优先于任何现有的 CamelFileName 标头；CamelOverruleFileName 是一个只使用一次的标头，它可让您更轻松地临时存储 CamelFileName，且之后必须恢复它。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外，该处理程序记录在 WARN/ERROR 级别并忽略。	false	布尔值
delete (consumer)	如果为 true，则在成功处理后会删除该文件。	false	布尔值
moveFailed (consumer)	根据简单语言设置移动失败表达式。例如，要将文件移到 .error 子目录中，请使用： .error。注意：将文件移到故障位置 Camel 时，将处理错误，并且不会再次获取该文件。		字符串
noop (consumer)	如果为 true，则文件不会以任何方式移动或删除。这个选项对只读数据或者 ETL 类型要求很好。如果 noop=true，Camel 也设置 idempotent=true，以避免再次消耗相同的文件。	false	布尔值

Name	描述	默认值	类型
preMove (consumer)	表达式（如文件语言）用于在处理之前动态设置文件名。例如，要将 in-progress 文件移到 order 目录中，将此值设置为 order。		字符串
preSort (consumer)	启用预先排序后，消费者将在轮询期间对文件和目录名称进行排序，该名称是从文件系统检索的。如果需要按排序顺序对文件进行操作，您可能需要执行此操作。指示器在消费者开始过滤前执行，并接受 Camel 处理的文件。这个选项是 default=false 表示禁用。	false	布尔值
递归 (consumer)	如果某个目录，将查找所有子目录中的文件。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
directoryMustExist (consumer)	与 startingDirectoryMustExist 类似，但这在轮询递归子目录期间应用。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，该例外记录在 WARN/ERROR 级别并忽略。		ExceptionHandler
exchangePattern (consumer)	在创建交换时设置默认交换模式。		ExchangePattern
extendedAttributes (consumer)	定义感兴趣的文件属性。与 posix:permissions,posix:owner,basic:lastAccessTime 类似，它支持基本通配符，如 posix:, basic:lastAccessTime		字符串
inProgressRepository (consumer)	可插拔式 in-progress 存储库 org.apache.camel.spi.IdempotentRepository。in-progress 存储库用于考虑被消耗的当前正在进行的文件。默认使用基于内存的存储库。		IdempotentRepository
localWorkDirectory (consumer)	在消耗时，可以使用本地工作目录直接将远程文件内容存储在本地文件中，以避免将内容加载到内存中。这很有用，如果您使用非常大的远程文件，因此可以节省内存。		字符串
onCompletionException Handler (consumer)	要使用自定义 org.apache.camel.spi.ExceptionHandler 来处理在完成过程中发生的任何抛出异常，消费者在其中执行提交或回滚。默认实施会将任何异常记录在 WARN 级别并忽略。		ExceptionHandler

Name	描述	默认值	类型
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。换句话说，在轮询收集信息时发生错误，例如，对文件网络的访问失败，因此 Camel 无法访问它来扫描文件。默认实施将记录 WARN 级别导致的异常并忽略它。		PollingConsumerPoll Strategy
probeContentType (consumer)	是否启用内容类型探测。如果启用，则消费者使用 Files SerialprobeContentType (java.nio.file.Path)来决定文件的 content-type，并在 Message 上存储为带有密钥 ExchangeacFILE_CONTENT_TYPE 的标头。	false	布尔值
processStrategy (consumer)	可插拔 org.apache.camel.component.file.GenericFileProcessStrategy 允许您实施自己的 readLock 选项或类似选项。也可以在使用文件之前满足特殊条件时使用，如存在特殊的就绪文件。如果设置了这个选项，则不会应用 readLock 选项。		GenericFileProcess s Strategy
startingDirectoryMustExist (consumer)	起始目录是否必须存在。请注意，autoCreate 选项是默认启用的，这意味着如果起始目录不存在，则通常会自动创建该目录。您可以禁用 autoCreate 并启用它，以确保起始目录必须存在。如果目录 不存在，将抛出异常。	false	布尔值
fileExist (producer)	如果文件已存在具有相同名称的文件，该怎么办。override （默认值）替换现有文件。Append - 在现有文件中添加内容。fail - 抛出 GenericFileOperationException，表示已存在文件。ignore - 静默忽略问题，且不会覆盖现有文件，但假设一切都正常。Move - 选项需要使用 moveExisting 选项也可以配置。选项 eagerDeleteTargetFile 可用于控制在移动文件时执行什么操作，并且已存在文件，否则会导致移动操作失败。Move 选项将在编写目标文件之前移动任何现有文件。只有在使用 tempFileName 选项时，TryRename 才适用。这允许尝试将文件从临时名称重命名为实际名称，而无需执行任何存在的检查。在某些文件系统（特别是 FTP 服务器中）中，这个检查可能会更快。	override	GenericFileExist
flatten (producer)	flatten 用于扁平化任何前导路径的文件名路径，因此只用于文件名。这可让您以递归方式消耗到子目录中，但当您将文件写入另一个目录时，它们将写入单个目录。在生成者中将其设置为 true 强制执行 CamelFileName 标头中的任何文件名都会被剥离任何前导路径。	false	布尔值

Name	描述	默认值	类型
jailStartingDirectory (producer)	用于存放（限制）将文件写入起始目录（和子目录）。这默认是启用的，不允许 Camel 将文件写入外部目录（从而更加安全）。您可以关闭此项，以允许将文件写入起始目录之外的目录，如父目录或 root 文件夹。	true	布尔值
moveExisting (producer)	当配置了 fileExist=Move 时，用于计算文件名的表达式（如 File Language）。要将文件移动到备份子目录中，请输入 backup。这个选项只支持以下 File Language 令牌：file:name, file:name.ext, file:name.noext, file:onlyname, file:onlyname.noext, file:ext, 和 file:parent。请注意，FTP 组件不支持 file:parent，因为 FTP 组件只能将任何现有文件移到基于当前 dir 的相对目录中。		字符串
tempFileName (producer)	与 tempPrefix 选项相同，但对临时文件名的命名提供更精细的控制，因为它使用 File Language。		字符串
tempPrefix (producer)	此选项用于使用临时名称写入文件，然后在写入完成后将其重命名为实际名称。可用于识别正在写入的文件，并避免消费者（不使用专用读取锁定）读取进度文件。上传大型文件时，FTP 通常使用。		字符串
allowNullBody (producer)	用于指定在文件写入过程中是否允许 null 正文。如果设置为 true，则会创建一个空文件，当设为 false 时，并尝试向文件组件发送 null 正文，则 'Cannot null body to file.' 的 GenericFileWriteException 将会被抛出。如果将 fileExist 选项设置为 'Override'，则文件将被截断，如果设置为附加文件，则文件将保持不变。	false	布尔值
chmod (producer)	指定生成者发送的文件权限，chmod 值必须在 000 和 777 之间；如果 0755 中有前导数字，我们将忽略它。		字符串
chmodDirectory (producer)	指定制作者创建缺失目录时使用的目录权限，chmod 值必须在 000 和 777 之间；如果前一个数字，如 0755，我们将忽略它。		字符串
eagerDeleteTargetFile (producer)	是否强制删除任何现有目标文件。这个选项只适用于使用 fileExists=Override 和 tempFileName 选项。您可以使用它来禁用（将其设置为 false）在写入临时文件前删除目标文件。例如，您可以写入大文件，并且希望写入 temp 文件期间存在目标文件。这样可确保目标文件仅在最后一次删除之前，直到 temp 文件被重命名为目标文件名之前。这个选项还用于控制在启用了 fileExist=Move 时是否删除任何现有文件，并且存在现有的文件。如果此选项 copyAndDeleteOnRenameFails false，则会在移动操作前现有文件存在时抛出异常。	true	布尔值

Name	描述	默认值	类型
forceWrites (producer)	是否强制同步对文件系统写入。如果您不希望此级别的保证，例如写入日志/审计日志等，则可以关闭此项。	true	布尔值
keepLastModified (producer)	将保留源文件的最后修改的时间戳（若有）。将使用 Exchange.FILE_LAST_MODIFIED 标头来定位时间戳。此标头可以包含带有时间戳的 java.util.Date 或 long。如果存在时间戳，并且启用了 选项，它将在写入的文件上设置此时间戳。注：此选项仅适用于文件制作者。您不能将此选项与任何 ftp 生成者一起使用。	false	布尔值
moveExistingFileStrategy (producer)	策略(Custom Strategy)用于在配置 fileExist=Move 时使用，使用特殊命名令牌来移动文件。默认情况下，如果没有提供自定义策略，则使用实施		FileMoveExisting 策略
auto create (advanced)	在文件的路径名称中自动创建缺少的目录。对于文件消费者，这意味着创建起始目录。对于生成者文件，这意味着文件应当要写入的目录。	true	布尔值
bufferSize (advanced)	写入缓冲区大小（以字节为单位）。	131072	int
copyAndDeleteOnRenameFail (advanced)	whether to fallback 和 do a copy and delete 文件，如果无法直接重命名该文件。此选项不适用于 FTP 组件。	true	布尔值
renameUsingCopy (advanced)	使用 copy 和 delete 策略执行重命名操作。这主要用于常规重命名操作不可靠（例如跨不同的文件系统或网络）。这个选项优先于 copyAndDeleteOnRenameFail 参数，该参数将自动回退到 copy 和 delete 策略，但仅在额外的延迟后自动回退到。	false	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
antExclude (filter)	Ant 风格过滤器排除。如果使用 antInclude 和 antExclude，则 antExclude 优先于 antInclude。可以使用逗号分隔的格式指定多个排除项。		字符串
antFilterCaseSensitive (filter)	在t 过滤器上设置问题单敏感标志	true	布尔值
antInclude (filter)	Ant 风格过滤器包括：可以使用逗号分隔的格式指定多个包含。		字符串

Name	描述	默认值	类型
eagerMaxMessagesPerPoll (filter)	允许控制 maxMessagesPerPoll 的限制是 eager 还是 not。如果 eager，则限制是在扫描文件时。其中 false 会扫描所有文件，然后执行排序。将此选项设置为 false 可首先对所有文件进行排序，然后限制轮询。请注意，这需要更高的内存用量，因为所有文件详细信息都在内存中执行排序。	true	布尔值
exclude (filter)	用于排除文件，如果文件名与 regex 模式匹配（匹配是 in-senstive）。请注意，如果您使用加号等符号，如果将其配置为端点 uri，则需要使用 RAW （）语法进行配置。有关配置端点 uris 时，请参阅更多详情		字符串
Filter (filter)	可插拔过滤器为 org.apache.camel.component.file.GenericFileFilter 类。如果过滤在 accept （）方法中返回 false，则会跳过文件。		GenericFileFilter
filterDirectory (filter)	根据简单语言过滤目录。例如，要过滤当前日期，您可以使用简单的日期模式，如 $date:now:yyMMdd		字符串
filterFile (filter)	根据简单语言过滤文件。例如，要过滤文件大小，您可以使用 $file:size 5000		字符串
idempotent (filter)	使用 Idempotent Consumer EIP 模式的选项，让 Camel 跳过已处理的文件。默认情况下，将使用基于内存的 LRUCache，其中包含 1000 个条目。如果 noop=true 然后会启用幂等，以避免再次消耗同样的文件。	false	布尔值
idempotentKey (filter)	使用自定义幂等密钥。默认情况下使用文件的绝对路径。您可以使用 File Language，例如使用文件名和文件大小，您可以执行以下操作：idempotentKey=$file:name-$file:size		字符串
idempotentRepository (filter)	一个可插拔式存储库 org.apache.camel.spi.IdempotentRepository，如果未指定 none，则默认使用 MemoryMessageIdRepository，并且幂等性为 true。		IdempotentRepository
include (filter)	用于包含文件，如果文件名匹配 regex 模式（匹配不区分大小写）。请注意，如果您使用加号等符号，如果将其配置为端点 uri，则需要使用 RAW （）语法进行配置。有关配置端点 uris 时，请参阅更多详情		字符串
maxDepth (filter)	递归处理目录时要遍历的最大深度。	2147483647	int

Name	描述	默认值	类型
maxMessagesPerPoll (filter)	定义每个轮询要收集的最大消息。默认不设置最大值。可用于设置限制，例如 1000，以避免在启动有数千个文件的服务器时避免。设置一个 0 或 negative 值来禁用它。注意：如果使用这个选项，则 File 和 FTP 组件将在任何排序之前限制。例如，如果您有 100000 文件并使用 maxMessagesPerPoll=500，则只有第一个 500 文件会被获取，然后排序。您可以使用 eagerMaxMessagesPerPoll 选项，并将其设置为 false 以允许先扫描所有文件，然后随后排序。		int
minDepth (filter)	在递归处理目录时开始处理的最小深度。使用 minDepth=1 表示基础目录。使用 minDepth=2 表示第一个子目录。		int
Move (filter)	表达式（如简单语言）用于在处理后动态设置文件名。要将文件移动到 .done 子目录中，只需输入 .done。		字符串
exclusiveReadLockStrategy (lock)	可插拔 read-lock 作为 org.apache.camel.component.file.GenericFileExclusiveReadLockStrategy 实施。		GenericFileExclusive ReadLockStrategy

Name	描述	默认值	类型
readLock (lock)	供消费者使用，只有在文件上具有独占的 read-lock 时才轮询文件（例如，该文件没有正在进行中或被写入）。Camel 将等到授予文件锁定为止。这个选项提供了 build in strategy: none - No read lock is in use markerFile - Camel 创建一个标记文件 (fileName.camelLock)，然后保存锁定。这个选项不适用于 FTP 组件更改 - 更改方法是使用文件长度/修改时间戳来检测当前是否复制该文件。至少将使用 1 sec 确定此选项，因此此选项无法像其他进程一样快速使用文件，但可以更可靠，因为 JDK IO API 始终无法确定文件目前是否被其他进程使用。可以使用 readLockCheckInterval 选项来设置检查频率。 fileLock - 用于使用 java.nio.channels.FileLock。这个选项不适用于 FTP 组件。当通过 mount/share 访问远程文件系统时，应该避免使用这种方法，除非文件系统支持分布式文件锁定。rename 是使用尝试重命名该文件的测试，如果我们可以获得专用只读锁。幂等的 - （仅限文件组件）幂等性是将 idempotentRepository 用作读锁。这允许使用支持集群读取锁定（如果幂等存储库实现支持集群）。idid-changed - （仅限文件组件）幂等更改，使用 idempotentRepository 并更改作为组合的 read-lock。这允许使用读取锁定，如果幂等存储库实施支持集群。jid-rename - （仅限文件组件） idempotent-rename 使用 idempotentRepository，并重命名作为组合的 read-lock。这允许使用支持集群的读取锁定（如果幂等存储库实现支持）。注意：各种读取锁定并不适用于在集群模式下工作，其中不同节点上的并发用户与共享文件系统上的相同文件竞争。markerFile 使用接近 atomic 操作来创建空的标记文件，但不保证在集群中工作。fileLock 可能会更好地工作，但文件系统需要支持分布式文件锁定，等等。如果幂等性存储库支持集群，如 Hazelcast 组件或 Infinispan，则使用幂等读取锁定支持集群。	none	字符串
readLockCheckInterval (lock)	读锁在 millis 中，如果读锁定支持，则间隔为 millis。这个间隔用于在尝试获取读锁定之间休眠。例如，在使用更改的读锁定时，您可以将更高的间隔周期设置为 cater 以慢速写入。如果生成者非常慢，则默认 1 sec. 可能太快。注意：对于 FTP，默认的 readLockCheckInterval 为 5000。readLockTimeout 值必须高于 readLockCheckInterval，但 thumb 规则是有一个超过 readLockCheckInterval 的 2 次或多次的超时。这是为了确保在达到超时前尝试获取锁进程允许随时读取的时间。	1000	long

Name	描述	默认值	类型
readLockDeleteOrphanLock Files (lock)	如果 Camel 没有正确关闭（如 JVM 崩溃），则应在启动时读取带有标记文件的读取锁定文件（可能已留在文件系统上）读取锁定。如果将此选项设置为 false，则任何孤立锁定文件将导致 Camel 不尝试选择该文件，这可能是因为另一个节点同时从同一个共享目录读取文件。	true	布尔值
readLockIdempotentRelease Async (lock)	延迟发布任务应该是同步还是异步的。请参阅 readLockIdempotentReleaseDelay 选项的详情。	false	布尔值
readLockIdempotentRelease AsyncPoolSize (lock)	使用异步发行任务时，调度线程池中的线程数量。在几乎所有用例中，使用默认值 1 个内核线程应足够足够，只有在更新幂等存储库缓慢或处理大量文件时才将其设置为更高的值。如果您使用共享线程池，通过配置 readLockIdempotentReleaseExecutorService 选项，这个选项不会被使用。请参阅 readLockIdempotentReleaseDelay 选项的详情。		int
readLockIdempotentRelease Delay (lock)	是否将发行任务延迟为 millis。当文件被视为具有共享幂等存储库的主动/主动集群场景时，这可用于延迟发行任务扩展窗口，以确保其他节点因为竞争条件而可能扫描和获取同一文件。通过扩展发行任务的 time-window 有助于防止这些情况。只有在将 readLockRemoveOnCommit 配置为 true 时才需要注意延迟。		int
readLockIdempotentRelease ExecutorService (lock)	使用自定义和共享线程池进行异步发行任务。请参阅 readLockIdempotentReleaseDelay 选项的详情。		ScheduledExecutor Service
readLockLogging Level (lock)	无法获取读取锁定时使用的日志记录级别。默认情况下，会记录 WARN。您可以更改此级别，例如将 OFF 更改为没有任何日志记录。这个选项只适用于 readLock 类型： changed, fileLock, idempotent, idempotent-changed, idempotent-rename, rename。	DEBUG	LoggingLevel
readLockMarkerFile (lock)	是否将标志文件与 changed、rename 或 exclusive read 锁定类型一起使用。默认情况下，还使用标记文件来保护获取相同文件的其他进程。通过将这个选项设置为 false 来关闭此行为。例如，如果您不希望将标记文件写入 Camel 应用程序的文件系统。	true	布尔值

Name	描述	默认值	类型
readLockMinAge (lock)	这个选项只适用于 readLock=changed。它可指定文件在尝试获取读取锁定前必须经过的最短期限。例如，使用 readLockMinAge=300s 来要求文件最新 5 分钟。这可加快更改的读锁定速度，因为它将只尝试获取至少给定年龄的文件。	0	long
readLockMinLength (lock)	这个选项只适用于 readLock=changed。它允许您配置最小文件长度。默认情况下，Camel 预期文件包含数据，因此默认值为 1。您可以将这个选项设置为零，以允许消耗零长度文件。	1	long
readLockRemoveOnCommit (lock)	这个选项只适用于 readLock=idempotent。它允许在处理文件成功并且执行提交时指定是否从幂等存储库中删除文件名条目。默认情况下，文件不会被删除，以确保不会发生任何竞争条件，因此另一个活动节点可能会尝试获取该文件。相反，幂等存储库可能支持驱除策略，在 X 分钟后用于驱除文件名条目 - 这样可保证竞争条件没有问题。请参阅 readLockIdempotentReleaseDelay 选项的详情。	false	布尔值
readLockRemoveOnRollback (lock)	这个选项只适用于 readLock=idempotent。它允许在处理文件失败时指定是否从幂等存储库中删除文件名条目，并发生回滚。如果此选项为 false，则文件名称条目将被确认（就像文件确实是提交一样）。	true	布尔值
readLockTimeout (lock)	read-lock 的 millis 中可选超时（如果 read-lock 支持）。如果无法授予 read-lock，并且触发超时，则 Camel 将跳过该文件。在下一次轮询 Camel 时，将再次尝试文件，此时可能会授予 read-lock。使用 0 或较低值来指示永久。目前，fileLock，改变并重命名支持超时。注意：对于 FTP，默认的 readLockTimeout 值为 20000 而不是 10000。readLockTimeout 值必须高于 readLockCheckInterval，但 thumb 规则是有一个超过 readLockCheckInterval 的 2 次或多次的超时。这是为了确保在达到超时前尝试获取锁进程允许随时读取的时间。	10000	long
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int

Name	描述	默认值	类型
delay (scheduler)	下一次轮询前的时间（毫秒）。默认值为 500。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。默认值为 1000。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。此选项允许您在多个消费者之间共享线程池。		ScheduledExecutor Service
scheduler (scheduler)	允许插件自定义 org.apache.camel.spi.ScheduledPollConsumerScheduler，以用作在轮询消费者运行时触发的调度程序。默认实施使用 ScheduledExecutorService，有一个 Quartz2 和 Spring，它支持 CRON 表达式。注意：如果使用自定义调度程序，则不能使用 initialDelay 的选项，则使用FixedDelay、timeUnit 和 scheduledExecutorService。使用文本 quartz2 来引用使用 Quartz2 调度程序；并使用文本 Spring 来使用基于 Spring 的；并使用文本 #myScheduler 来引用 registry 中的 id 的自定义调度程序。如需示例，请参阅 Quartz2 页。	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
Shuff le (sort)	影响文件列表（以随机顺序排序）	false	布尔值
排序(排序)	使用文件语言进行内置排序.支持嵌套排序，以便按文件名排序，并根据修改的日期作为第二组排序。		字符串

Name	描述	默认值	类型
排序 （排序）	可插拔排序器作为 java.util.Comparator 类。		比较器

提示

文件生成者的默认行为 将默认覆盖任何现有文件（如果名称相同）。

104.3. 移动和删除操作

任何移动或删除操作都会在(post 命令)完成后执行；因此，在处理 Exchange 时，该文件仍然位于 inbox 文件夹中。

让我们用一个示例来说明这一点：

```
from("file://inbox?move=.done").to("bean:handleOrder");
```

当文件在 inbox 文件夹中丢弃时，文件使用者会注意到这个内容，并创建一个新的 FileExchange，它被路由到 handleOrder bean。然后，bean 会处理 File 对象。此时，该文件仍然位于 inbox 文件夹中。在 bean 完成后，路由完成，因此文件使用者将执行 move 操作，并将文件移到 .done 子文件夹。

move 和 preMove 选项被视为目录名称（但是，如果您使用表达式，如 File Language ），或 简单，表达式评估的结果是要使用的文件名 - 例如，如果您设置了，则表达式评估的结果是要使用的文件名。

```
move=../backup/copy-of-${file:name}
```

然后，使用 文件语言 返回要使用的文件名，可以是相对或绝对的。如果相对，则目录创建为来自文件的文件夹中的子文件夹，其中使用了该文件。

默认情况下，Camel 会将消耗的文件移到相对于使用文件的目录的 .camel 子文件夹。

如果要在处理后删除该文件，则路由应该是：

```
from("file://inobox?delete=true").to("bean:handleOrder");
```

我们引入了一个新的 预 移动操作，以便在处理文件 之前 移动文件。这样，您可以在处理前标记哪些文件已扫描到这个子文件夹中。

```
from("file://inbox?preMove=inprogress").to("bean:handleOrder");
```

您可以组合 预 移动和常规移动：

```
from("file://inbox?preMove=inprogress&move=.done").to("bean:handleOrder");
```

因此，在这种情况下，文件在处理后处于 inprogress 文件夹中，它被移到 .done 文件夹。

104.4. 对 MOVE 和 PREMOVE 选项的精细控制

move 和 preMove 选项基于 Expression，因此我们拥有 文件语言 的完整电源，可以执行目录和名称模式的高级配置。
实际上，Camel 将内部将您输入的目录名称转换为 File Language 表达式。因此，当输入 move=.done Camel 会将此转换为 ${file:parent}/.done/${'file:onlyname} 时。这只有在 Camel 检测到您自己没有在 option 值中提供 $\{ } 时完成。因此，当您输入 $\{ } Camel 时，不会 转换它，因此您有完整的电源。

因此，如果我们希望将文件移到具有当前日期作为模式的备份文件夹中，我们可以：

```
move=backup/${date:now:yyyyMMdd}/${file:name}
```

104.5. 关于 MOVEFAILED

moveFailed 选项允许您将 无法处理的 文件移到其他位置，如您选择的错误文件夹。例如，要移动带有时间戳的错误文件夹中的文件，您可以使用 moveFailed=/error/${file:name.noext}-${date:now:yyyyMMddHHmmssSSS}.${"file:ext}。

请参阅 文件语言的更多示例

104.6. 消息标头

此组件支持以下标头：

104.6.1. 仅限文件制作者

标头	描述
CamelFileName	指定要写入的文件的名称（相对于端点目录）。此名称可以是 **String** ；一个字符串，带有 File Language 或 Simple 表达式；或 Expression 对象。**如果是 null，则 Camel 将根据消息唯一 ID 自动生成文件名。**
CamelFileNameProduced	所编写的输出文件的实际绝对文件路径（路径 + 名称）。此标头由 Camel 设置，其目的是为最终用户提供所写入文件的名称。
CamelOverruleFileName	Camel 2.11： 用于覆盖 **CamelFileName** 标头并使用值（但仅一次，因为生成者将在编写文件后删除此标头）。该值只能是一个 String。请注意，如果配置了选项 **fileName**，则仍然会被评估。

104.6.2. 仅限文件消费者

标头	描述
CamelFileName	使用的文件的名称作为相对文件路径，且来自端点上配置的起始目录偏移。
CamelFileNameOnly	只有文件名（没有前导路径的名称）。
CamelFileAbsolute	指定消耗的文件是否表示绝对路径 **的布尔值** 选项。对于相对路径，通常应该为 **false**。通常不应使用绝对路径，但我们被添加到 move 选项中，以允许将文件移动到绝对路径中。但也可以在其他位置使用。
CamelFileAbsolutePath	文件的绝对路径。对于相对文件，此路径包含相对路径。
CamelFilePath	文件路径。对于相对文件，这是起始目录 + 相对文件名。对于绝对文件，这是绝对路径。

标头	描述
CamelFileRelativePath	相对路径。
CamelFileParent	父路径。
CamelFileLength	包含文件大小 **的**长 值。
CamelFileLastModified	包含文件最后一次修改时间戳的 **Long** 值。在 Camel 2.10.3 **中，** 类型是 **Date**。

104.7. BATCH CONSUMER

*这个组件实现了 **Batch Consumer。***

104.8. 交换属性，仅限文件消费者

由于文件使用者实施 BatchConsumer，它支持批处理它轮询的文件。通过批处理意味着 Camel 将在交换中添加以下额外属性，因此您知道轮询文件的数量、当前索引以及批处理是否已完成。

属性	描述
CamelBatchSize	在此批处理中轮询的文件总数。
CamelBatchIndex	批处理的当前索引。从 0 开始。
CamelBatchComplete	指示批处理中最后一个交换的**布尔值**。仅对最后一个条目是 **true**。

这样，您可以让实例知道此批处理中存在多少个文件，而实例则让 Aggregator2 聚合这个数量的文件。

104.9. 使用 CHARSET

对于 Camel 2.9.3，
charset 选项允许在消费者和生成者端点上配置文件的编码。例如，如果您读取 utf-8 文件，并希望将文件转换为 iso-8859-1，您可以执行以下操作：

```
from("file:inbox?charset=utf-8")
  .to("file:outbox?charset=iso-8859-1")
```

您还可以在路由中使用 convertBodyTo。在以下示例中，我们仍然以 utf-8 格式输入文件，但我们希望将文件内容转换为 iso-8859-1 格式的字节阵列。然后让 bean 进程处理数据。在使用当前的 charset 将内容写入 outbox 文件夹之前。

```
from("file:inbox?charset=utf-8")
  .convertBodyTo(byte[].class, "iso-8859-1")
  .to("bean:myBean")
  .to("file:outbox");
```

如果您在消费者端点上省略 charset，则 Camel 不知道文件的 charset，默认情况下会使用 "UTF-8"。但是，您可以配置 JVM 系统属性，以覆盖和使用键 org.apache.camel.default.charset 的不同默认编码。

在以下示例中，如果文件不在 UTF-8 编码中，这可能会造成问题，这是读取文件的默认编码。
在编写文件时，内容已转换为字节阵列，因此将内容直接写为原样（无需进一步编码）。

```
from("file:inbox")
  .convertBodyTo(byte[].class, "iso-8859-1")
  .to("bean:myBean")
  .to("file:outbox");
```

您还可以通过使用 Exchange.CHARSET_NAME 在交换上设置属性，在编写文件时覆盖和控制编码动态。例如，在下面的路由中，我们将属性设置为消息标头中的值。

```
from("file:inbox")
  .convertBodyTo(byte[].class, "iso-8859-1")
  .to("bean:myBean")
  .setProperty(Exchange.CHARSET_NAME, header("someCharsetHeader"))
  .to("file:outbox");
```

我们建议更加简单，因此，如果您选择具有相同编码的文件，并希望使用端点上的 charset 选项写入文件，则最好使用端点上的 charset 选项。

请注意，如果您明确在端点上配置了 charset 选项，则使用该配置，而不考虑 Exchange.CHARSET_NAME 属性。

如果您有一些问题，您可以在 org.apache.camel.component.file 上启用 DEBUG 日志记录，并在使用特定的 charset 读取/写文件时，启用 Camel 日志。
例如，以下路由会记录以下内容：

```
from("file:inbox?charset=utf-8")
  .to("file:outbox?charset=iso-8859-1")
```

和日志：

```
DEBUG GenericFileConverter          - Read file /Users/davsclaus/workspace/camel/camel-
core/target/charset/input/input.txt with charset utf-8
DEBUG FileOperations                - Using Reader to write file: target/charset/output.txt with charset:
iso-8859-1
```

104.10. COMMON GOTCHAS WITH FOLDER 和 FILENAMES

当 Camel 生成文件（写入文件）时，有几个影响了如何设置您选择的文件名。默认情况下，Camel 将使用消息 ID 作为文件名，并且消息 ID 通常是唯一的 ID，因此您将以文件名结尾，例如：ID-MACHINENAME-2443-1211718892437-1-0。如果不需要此类文件名，则必须在 CamelFileName 消息标头中提供文件名。也可使用常量 Exchange.FILE_NAME。

以下示例代码使用消息 ID 作为文件名生成文件：

```
from("direct:report").to("file:target/reports");
```

使用 report.txt 作为您需要做的文件名：

```
from("direct:report").setHeader(Exchange.FILE_NAME, constant("report.txt")).to(
"file:target/reports");
```

i. 与以上相同，但使用 CamelFileName ：

```
from("direct:report").setHeader("CamelFileName", constant("report.txt")).to(
"file:target/reports");
```

以及使用 fileName URI 选项在端点上设置文件名的语法。

```
from("direct:report").to("file:target/reports/?fileName=report.txt");
```

104.11. 文件名表达式

filename 可以使用 expression 选项或字符串在 CamelFileName 标头中的基于 文件语言 表达式进行设置。如需语法和样本，请参阅 文件语言。

104.12. 从其他人直接丢弃文件的文件夹中消耗文件

请注意，如果您消耗其他应用程序直接将文件写入的文件夹中的文件。查看不同的 readLock 选项，以查看适合您的用例的内容。最佳方法是写入另一个文件夹，并在写入移动 drop 文件夹中的文件之后。但是，如果您将文件直接写入 drop 文件夹，则选项 changed 可能会更好地检测文件当前是否被写入/复制，因为它使用文件更改的算法来查看在一段时间内是否更改文件大小/修改。其他 readLock 选项依赖于 Java File API，在检测到这一点时并非始终很好。您可能还希望查看 doneFileName 选项，该选项在文件完成后使用标记文件（目标文件）来表示信号。

104.13. 使用完成的文件

从 Camel 2.6 开始提供

另请参阅在下面 编写文件 的部分。

如果您只想在文件存在时使用文件，您可以在端点上使用 doneFileName 选项。

```
from("file:bar?doneFileName=done");
```

如果在目标文件的同一个目录中存在一个 done 文件，将仅使用 bar 文件夹中的文件。在完成使用文件或，Camel 将自动删除 done 文件。如果配置了 noop=true，则 Camel 上的 Camel 2.9.3 不会自动删除 done 文件。

但是，每个目标文件都有一个 done 文件 更为常见。这意味着有一个 1:1 相关性。要做到这一点，您必须在 doneFileName 选项中使用动态占位符。目前，Camel 支持以下两个动态令牌： file:name 和

file:name.noext，它必须包括在 $\{ } 中。消费者仅支持 done 文件名 的静态部分作为前缀或后缀（不支持两者）。

```
from("file:bar?doneFileName=${file:name}.done");
```

本例中只有文件轮询，如果存在名为 . done 的文件。例如：

- hello.txt - 是要消耗的文件
- hello.txt.done - 是关联的 done 文件

您还可以将前缀用于 done 文件，例如：

```
from("file:bar?doneFileName=ready-${file:name}");
```

- hello.txt - 是要消耗的文件
- ready-hello.txt - 是关联的 done 文件

104.14. 编写完成的文件

从 Camel 2.6 开始提供

在编写完文件后，您可能希望将额外的 done文件 写为标记类型，以指示文件已完成并已写入。为此，您可以在文件制作者端点上使用 doneFileName 选项。

```
.to("file:bar?doneFileName=done");
```

只需在与目标文件相同的目录中创建一个名为 done 的文件。

但是，每个目标文件都有一个 done 文件更为常见。这意味着有一个 1:1 相关性。要做到这一点，您必须在 doneFileName 选项中使用动态占位符。目前，Camel 支持以下两个动态令牌： file:name 和 file:name.noext，它必须包括在 $\{ } 中。

-

```
.to("file:bar?doneFileName=done-${file:name}");
```

如果目标文件是在与目标文件所在的相同目录中的 foo.txt 文件，则会创建一个名为 done-foo.txt 的文件。

```
.to("file:bar?doneFileName=${file:name}.done");
```

如果目标文件是在与目标文件所在的相同目录中的 foo.txt 文件，则会创建一个名为 foo.txt.done 的文件。

```
.to("file:bar?doneFileName=${file:name.noext}.done");
```

如果目标文件是在与目标文件所在的相同目录中的 foo.txt 文件，则会创建一个名为 foo.done 的文件。

104.15. SAMPLES

#=== 从目录中读取并写入另一个目录

```
from("file://inputdir/?delete=true").to("file://outputdir")
```

104.15.1. 从目录读取并使用 overrule 动态名称写入另一个目录

```
from("file://inputdir/?delete=true").to("file://outputdir?overruleFile=copy-of-${file:name}")
```

侦听目录，并为那里丢弃的每个文件创建一条消息。将内容复制到 outputdir，再删除 inputdir 中的文件。

104.15.2. 从目录中递归读取并写入另一个目录

```
from("file://inputdir/?recursive=true&delete=true").to("file://outputdir")
```

侦听目录，并为那里丢弃的每个文件创建一条消息。将内容复制到 outputdir，再删除 inputdir 中的文件。将递归扫描到子目录中。会将文件放入与 input dir 相同的目录结构中，包括任何子目录。

```
inputdir/foo.txt
inputdir/sub/bar.txt
```

将导致以下输出布局：

```
outputdir/foo.txt
outputdir/sub/bar.txt
```

104.16. 使用 FLATTEN

如果要将文件存储在同一目录中的 outputdir 目录中，请忽略源目录布局（例如，扁平化路径），只需在文件制作者端添加 flatten=true 选项：

```
from("file://inputdir/?recursive=true&delete=true").to("file://outputdir?flatten=true")
```

将导致以下输出布局：

```
outputdir/foo.txt
outputdir/bar.txt
```

104.17. 从目录和默认移动操作读取

默认情况下，Camel 会将任何已处理的文件移到文件所消耗的目录中的 .camel 子目录。

```
from("file://inputdir/?recursive=true&delete=true").to("file://outputdir")
```

按如下所示影响布局：
before

```
inputdir/foo.txt
inputdir/sub/bar.txt
```

after

```
inputdir/.camel/foo.txt
inputdir/sub/.camel/bar.txt
outputdir/foo.txt
outputdir/sub/bar.txt
```

104.18. 从目录读取并处理 JAVA 中的消息

```
from("file://inputdir/").process(new Processor() {
```

```
  public void process(Exchange exchange) throws Exception {
    Object body = exchange.getIn().getBody();
    // do some business logic with the input body
  }
});
```

正文将是一个 File 对象，它指向刚刚放入到 inputdir 目录中的文件。

104.19. 写入文件

Camel 也可以编写文件，即生成文件。在以下示例中，我们会收到有关在 SEDA 队列写入到目录之前所处理的报告。

104.19.1. 使用 Exchange.FILE_NAME写入子目录

使用单一路由时，可以将文件写入到任意数量的子目录。如果您有路由设置，如下所示：

```
<route>
  <from uri="bean:myBean"/>
  <to uri="file:/rootDirectory"/>
</route>
```

您可以将 myBean 将标头 Exchange.FILE_NAME 设置为值，例如：

```
Exchange.FILE_NAME = hello.txt => /rootDirectory/hello.txt
Exchange.FILE_NAME = foo/bye.txt => /rootDirectory/foo/bye.txt
```

这可让您有一个路由将文件写入多个目的地。

104.19.2. 通过相对于最终目的地的临时目录写入文件

有时，您需要临时将文件写入相对于目标目录的一些目录。当某些具有有限过滤功能的外部进程从您要写入的目录中读取时，通常会发生这种情况。在以下示例中，将写入 /var/myapp/filesInProgress 目录，并在进行数据传输后，它们将被整体移到' /var/myapp/finalDirectory 'directory 中。

```
from("direct:start").
  to("file:///var/myapp/finalDirectory?tempPrefix=/../filesInProgress/");
```

104.20. 对文件名使用表达式

在本例中，我们想使用今天的日期作为子文件夹名称将消耗的文件移到备份文件夹中：

```
from("file://inbox?move=backup/${date:now:yyyyMMdd}/${file:name}").to("...");
```

如需了解更多示例，请参阅 文件语言。

104.21. 避免多次读取同一文件（验证消费者）

Camel 支持在组件内直接支持 Idempotent Consumer，以便跳过已处理的文件。通过设置 idempotent=true 选项可以启用此功能。

```
from("file://inbox?idempotent=true").to("...");
```

Camel 使用绝对文件名作为幂等键，来检测重复的文件。从 Camel 2.11 开始，您可以使用 idempotentKey 选项中的表达式来自定义这个密钥。例如，使用名称和文件大小作为密钥

```
<route>
  <from uri="file://inbox?idempotent=true&amp;idempotentKey=${file:name}-${file:size}"/>
  <to uri="bean:processInbox"/>
</route>
```

默认情况下，Camel 使用基于内存的存储来跟踪消耗的文件，它会使用最少使用最多 1000 个条目的缓存。您可以使用值中的 idempotentRepository 选项使用 idempotentRepository 选项来插件您自己的存储实现，以指示它在带有指定 id 的 Registry 中的 bean 的引用。

```
<!-- define our store as a plain spring bean -->
<bean id="myStore" class="com.mycompany.MyIdempotentStore"/>

<route>
  <from uri="file://inbox?idempotent=true&amp;idempotentRepository=#myStore"/>
  <to uri="bean:processInbox"/>
</route>
```

如果一个文件在 DEBUG 级别被使用，则 Camel 将在 DEBUG 级别记录，因为它已在前面被使用：

```
DEBUG FileConsumer is idempotent and the file has been consumed before. Will skip this file:
target\idempotent\report.txt
```

104.22. 使用基于文件的幂等存储库

在本节中，我们将使用基于文件的幂等存储库 **org.apache.camel.processor.idempotent.FileIdempotentRepository**，而不是基于内存的，用作默认值。
此仓库使用第一级缓存以避免读取文件存储库。它将仅使用文件存储库来存储第一级别缓存的内容。因此，存储库可以在服务器重启后保留。它将在启动时将文件的内容加载到第一级缓存中。文件结构非常简单，因为它将密钥存储在文件的单独行中。默认情况下，文件存储的大小限制为 1mb。当文件增长更大的 Camel 时，将通过将第一级缓存刷新到一个新的空文件来截断文件存储。

我们使用 Spring XML 配置我们的存储库，创建文件幂等存储库，并定义文件消费者来使用我们的带有 idempotentRepository 的存储库，以指示 Registry 查找：

104.23. 使用基于 JPA 的幂等存储库

在本节中，我们将使用基于 JPA 的幂等存储库，而不是以默认方式为基础的内存。

首先，我们需要 META-INF/persistence.xml 中的 persistence-unit，其中我们需要使用类 **org.apache.camel.processor.idempotent.jpa.MessageProcessed** 作为模型。

```
<persistence-unit name="idempotentDb" transaction-type="RESOURCE_LOCAL">
  <class>org.apache.camel.processor.idempotent.jpa.MessageProcessed</class>

  <properties>
    <property name="openjpa.ConnectionURL" value="jdbc:derby:target/idempotentTest;create=true"/>
    <property name="openjpa.ConnectionDriverName"
value="org.apache.derby.jdbc.EmbeddedDriver"/>
    <property name="openjpa.jdbc.SynchronizeMappings" value="buildSchema"/>
    <property name="openjpa.Log" value="DefaultLevel=WARN, Tool=INFO"/>
    <property name="openjpa.Multithreaded" value="true"/>
  </properties>
</persistence-unit>
```

接下来，我们还可在 spring XML 文件中创建 JPA idempotent 存储库：

```
<!-- we define our jpa based idempotent repository we want to use in the file consumer -->
<bean id="jpaStore" class="org.apache.camel.processor.idempotent.jpa.JpaMessageIdRepository">
    <!-- Here we refer to the entityManagerFactory -->
    <constructor-arg index="0" ref="entityManagerFactory"/>
    <!-- This 2nd parameter is the name  (= a category name).
         You can have different repositories with different names -->
    <constructor-arg index="1" value="FileConsumer"/>
</bean>
```

然后，我们只需要使用 # 语法选项在文件消费者端点中使用 idempotentRepository 来指代文件消费者端点中的 jpaStore bean ：

```
<route>
  <from uri="file://inbox?idempotent=true&amp;idempotentRepository=#jpaStore"/>
  <to uri="bean:processInbox"/>
</route>
```

104.24. 使用 ORG.APACHE.CAMEL.COMPONENT.FILE.GENERICFILEFILTER 进行过滤

Camel 支持可插拔过滤策略。然后，您可以使用这样的过滤器配置端点，以跳过被处理的某些文件。

在示例中，我们构建了自己的过滤器，它会跳过其文件名是以 skip 开始的文件：

然后，我们可以使用 filter 属性配置我们的路由，以引用我们在 spring XML 文件中定义的过滤（使用 # 表示法）：

```
<!-- define our filter as a plain spring bean -->
<bean id="myFilter" class="com.mycompany.MyFileFilter"/>

<route>
  <from uri="file://inbox?filter=#myFilter"/>
  <to uri="bean:processInbox"/>
</route>
```

104.25. 使用 ANT 路径匹配器进行过滤

ANT 路径匹配器在 camel-spring jar 中提供出厂。因此，如果您使用 Maven，则需要依赖 camel-spring。
原因在于，我们利用 Spring 的 AntPathMatcher 进行实际匹配。

文件路径与以下规则匹配：

- ? 匹配一个字符
- * 匹配零个或多个字符
- ** 匹配路径中的零个或多个目录

提示

现在，Camel 2.10 的新选项有 antInclude 和 antExclude 选项，以便轻松指定 ANT 风格 include/exclude，而无需定义过滤器。如需更多信息，请参阅上面的 URI 选项。

以下示例演示了如何使用它：

104.25.1. 使用比较器排序

Camel 支持可插拔排序策略。这可确保它在 Java 中的 java.util.Comparator 中使用构建。然后，您可以使用这样的比较器配置端点，并在处理前让 Camel 排序文件。

在示例中，我们构建了自己的比较器，仅按文件名排序：

然后，我们可以使用 sorter 选项配置路由以引用我们的排序器(我的Sorter)在 spring XML 文件中定义：

```
<!-- define our sorter as a plain spring bean -->
<bean id="mySorter" class="com.mycompany.MyFileSorter"/>

<route>
  <from uri="file://inbox?sorter=#mySorter"/>
  <to uri="bean:processInbox"/>
</route>
```

提示

URI 选项可以使用上述的 # 语法来引用 Bean，请注意，我们可以通过将 id 前缀为 # 来引用 Registry 中的 Bean。因此，编写 sorter=stepsmySorter 将指示 Camel 在 Registry 中查找 ID 为 mySorter 的 bean。

104.25.2. 使用 sortBy 排序

Camel 支持可插拔排序策略。此策略使用 文件语言 来配置排序。sortBy 选项配置如下：

```
sortBy=group 1;group 2;group 3;...
```

其中每个组用分号分隔。在简单情况下，您只需要使用一个组，因此一个简单的示例可以是：

```
sortBy=file:name
```

这将按文件名排序，您可以通过为组反反对顺序 相反，从而使排序现在为 Z..A：

```
sortBy=reverse:file:name
```

我们拥有 文件语言 的完整功能，我们可以使用其他一些参数，因此如果我们希望按文件大小排序：

```
sortBy=file:length
```

您可以将 配置为忽略大小写，使用 ignoreCase: 进行字符串比较，因此如果您想要使用文件名排序，但忽略大小写，然后我们做：

```
sortBy=ignoreCase:file:name
```

您可以组合忽略的大小写和反向，但必须首先指定 reverse ：

```
sortBy=reverse:ignoreCase:file:name
```

在以下示例中，我们希望按上次修改的文件排序，因此我们这样做：

```
sortBy=file:modified
```

然后，我们希望根据名称对第 2 个选项进行分组，以便具有相同 modifcation 的文件按名称排序：

```
sortBy=file:modified;file:name
```

现在，这里有一个问题，您可以发现它吗？文件修改的时间戳在毫秒内过于正常，但如果我们只想按日期排序，然后按名称对子组进行排序会怎样？
我们拥有 文件语言 的真正能力，我们可以使用支持模式的 date 命令。这可以通过以下方式解决：

```
sortBy=date:file:yyyyMMdd;file:name
```

是的，它非常强大，您可以采用按组反向使用的方式 oh，因此我们可以撤销文件名：

```
sortBy=date:file:yyyyMMdd;reverse:file:name
```

104.26. 使用 GENERICFILEPROCESSSTRATEGY

选项 processStrategy 可用于使用自定义 GenericFileProcessStrategy，允许您自行实施 开始、提交 和回滚 逻辑。
例如，假设系统在应该消耗的文件夹中写入一个文件。但是，在写入另一个 就绪 文件之前，您不应该开始消耗该文件。

因此，通过实施自己的 GenericFileProcessStrategy，我们可以做到这一点：

- 在 begin （） 方法中，我们可以测试特殊 就绪 文件是否存在。begin 方法返回 布尔值，以指示我们可以使用文件或不用。
- 在 abort （） 方法(Camel 2.10)中，可以在 begin 操作返回 false 时执行特殊逻辑，例如清理资源等。
- 在 commit （） 方法中，我们可以移动实际文件，同时删除 ready 文件。

104.27. 使用过滤器

filter 选项允许您通过实施 org.apache.camel.component.file.GenericFileFilter 接口在 Java 代码中实施自定义过滤器。此接口具有返回布尔值的 accept 方法。返回 true 以包含该文件，而 false 以跳过该文件。从 Camel 2.10 开始，在 GenericFile 上有一个 isDirectory 方法，无论该文件是否为目录。这可让您过滤不需要的目录，以避免遍历不需要的目录。

例如，要跳过名称中以 "skip" 开头的目录，可以实施如下：

104.28. 使用 CONSUMER.BRIDGEERRORHANDLER

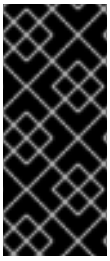
从 Camel 2.10 开始提供

如果要使用 Camel Error Handler 处理文件消费者中发生的任何异常，您可以启用

consumer.bridgeErrorHandler 选项，如下所示：

```
// to handle any IOException being thrown
onException(IOException.class)
    .handled(true)
    .log("IOException occurred due: ${exception.message}")
    .transform().simple("Error ${exception.message}")
    .to("mock:error");

// this is the file route that pickup files, notice how we bridge the consumer to use the Camel routing error handler
// the exclusiveReadLockStrategy is only configured because this is from an unit test, so we use that to simulate exceptions
from("file:target/nospace?consumer.bridgeErrorHandler=true")
    .convertBodyTo(String.class)
    .to("mock:result");
```

因此，您只需要启用这个选项，路由中的错误处理程序将从那里获取它。



重要

在使用 consumer.bridgeErrorHandler 时，在使用 consumer.bridgeErrorHandler 时很重要，然后 interceptors, OnCompletions 不适用。Exchange 由 Camel Error Handler 直接处理，不允许在编译时执行以前的操作，如拦截器。

104.29. 调试日志记录

此组件具有日志级别 TRACE，如果您遇到问题，这非常有用。

104.30. 另请参阅

- 文件语言
- FTP
- polling Consumer

第 105 章 文件语言

从 Camel 版本 1.1 开始提供

INFO:*File 语言现在从 Camel 2.2 开始与 Simple language 合并，文件语言现在使用 Simple 语言合并，这意味着您可以在简单语言中直接使用所有文件语法。

文件表达式语言是简单语言 的扩展，添加文件相关功能。这些功能与处理文件路径和名称的常见用例相关。目标是允许表达式与 File 和 FTP 组件一起使用，以便为消费者和生成者设置动态文件模式。

105.1. 文件语言选项

文件语言支持 2 个选项，如下所列。

Name	默认值	Java 类型	描述
resultType		字符串	设置结果类型的类名称（输出中的类型）
trim	true	布尔值	是否修剪值以移除前导和结尾的空格和换行符

105.2. 语法

此语言是简单语言的扩展，因此也应用 Simple 语法。因此，下表仅列出附加内容。与 简单 语言 文件语言 相反还支持 Constant 表达式，以便您可以输入固定文件名。

所有文件令牌都使用与 java.io.File 对象上方法相同的表达式名称，例如 file:absolute 指的是 java.io.File.getAbsolute （） 方法。请注意，当前交换都支持所有表达式。例如，FTP 组件支持一些选项，因为 File 组件支持所有这些选项。

表达式	类型	File Consumer	文件生成	FTP Consumer	FTP Producer	描述
file:name	字符串	是	否	是	否	指的是文件名（相对于起始目录，请参阅以下内容）

表达式	类型	File Consumer	文件生成	FTP Consumer	FTP Producer	描述
file:name.ext	字符串	是	否	是	否	**Camel 2.3：** 仅引用文件扩展名
file:name.ext.single	字符串	是	否	是	否	**Camel 2.14.4/2.15.3:** 是指文件扩展名。如果文件扩展有多个圆点，则此表达式剥离，仅返回最后一个部分。
file:name.noext	字符串	是	否	是	否	指的是没有扩展名的文件名（相对于起始目录，请参考下面的备注）
file:name.noext.single	字符串	是	否	是	否	**Camel 2.14.4/2.15.3:** 是指不带扩展名的文件名（相对于起始目录，请参见以下注释）。如果文件扩展有多个点，则此表达式只剥离最后一个部分，而保留其他部分。
file:onlyname	字符串	是	否	是	否	只引用没有前导路径的文件名。
file:onlyname.noext	字符串	是	否	是	否	只引用文件名，没有扩展名，且没有前导路径。
file:onlyname.noext.single	字符串	是	否	是	否	*Camel 2.14.4/2.15.3:* 只参考文件名，没有扩展名，且没有前导路径。如果文件扩展有多个点，则此表达式只剥离最后一个部分，而保留其他部分。
file:ext	字符串	是	否	是	否	仅引用文件扩展名
file:parent	字符串	是	否	是	否	指的是文件父级
file:path	字符串	是	否	是	否	引用文件路径
file:absolute	布尔值	是	否	否	否	指的是该文件是否被视为绝对还是相对

表达式	类型	File Consumer	文件生成	FTP Consumer	FTP Producer	描述
file:absolute.path	字符串	是	否	否	否	指的是绝对路径
file:length	Long	是	否	是	否	将文件长度指代为 Long type
file:size	Long	是	否	是	否	**Camel 2.5：** 将文件长度指代为 Long 类型
file:modified	Date	是	否	是	否	将上次修改的文件称为 Date 类型
date:_command:pattern_	字符串	是	是	是	是	对于使用 **java.text.SimpleDateFormat** 模式的日期格式。是 简单语言的扩展。**其他命令为：文件（仅限消费者）用于文件最后一次修改的时间戳。注意：也可以使用 Simple 语言中的所有命令。**

105.3. 文件令牌示例

105.3.1. 相对路径

在以下 相对 目录中，我们有一个 hello.txt 文件的 java.io.File handle： .\filelanguage\test。我们会将端点配置为使用此起始目录 .\filelanguage。文件令牌将返回：

èi¨è¾¾å¼	返回
file:name	test\hello.txt
file:name.ext	txt
file:name.noext	test\hello
file:onlyname	hello.txt

èi¨è¾¾å¼	返回
file:onlyname.noext	您好
file:ext	txt
file:parent	filelanguage\test
file:path	filelanguage\test\hello.txt
file:absolute	false
file:absolute.path	\workspace\camel\camel-core\target\filelanguage\test\hello.txt

105.3.2. 绝对路径

在以下 绝对 目录中，我们有一个 hello.txt 文件的 java.io.File handle： \workspace\camel\camel-core\target\filelanguage\test。我们将端点配置为使用绝对起始目录 \workspace\camel\camel-core\target\filelanguage。文件令牌将返回：

èi¨è¾¾å¼	返回
file:name	test\hello.txt
file:name.ext	txt
file:name.noext	test\hello
file:onlyname	hello.txt
file:onlyname.noext	您好
file:ext	txt
file:parent	\workspace\camel\camel-core\target\filelanguage\test
file:path	\workspace\camel\camel-core\target\filelanguage\test\hello.txt
file:absolute	true
file:absolute.path	\workspace\camel\camel-core\target\filelanguage\test\hello.txt

105.4. SAMPLES

您可以输入固定的 Constant 表达式，如 myfile.txt ：

```
fileName="myfile.txt"
```

我们假设我们使用文件使用者来读取文件，并希望将读取文件移到以当前日期作为子文件夹的备份文件夹。这可以使用类似以下的表达式进行归档：

```
fileName="backup/${date:now:yyyyMMdd}/${file:name.noext}.bak"
```

也支持相对文件夹名称，因此假设备份文件夹应为同级文件夹，然后您可以附加 .., 如下所示：

```
fileName="../backup/${date:now:yyyyMMdd}/${file:name.noext}.bak"
```

由于这是我们从此语言访问所有好事的扩展，因此在这种用例中，我们想在动态表达式中使用 in.header.type 作为参数： ???

```
fileName="../backup/${date:now:yyyyMMdd}/type-${in.header.type}/backup-of-${file:name.noext}.bak"
```

如果您在表达式中使用了自定义日期，则 Camel 支持从消息标头中检索日期。

```
fileName="orders/order-${in.header.customerId}-${date:in.header.orderDate:yyyyMMdd}.xml"
```

最后，我们还可以使用 bean 表达式来调用 POJO 类，该类生成一些字符串输出（或转换为 String）：

```
fileName="uniquefile-${bean:myguidgenerator.generateid}.txt"
```

当然，这些内容都可以在一个表达式中合并，您可以在一个组合表达式中使用 文件语言、简单 和 Bean 语言。对于这些常用文件路径模式，这非常强大。

105.5. 将 SPRING PROPERTYPLACEHOLDERCONFIGURER 与 FILE 组件一起使用

在 Camel 中，您可以直接使用 Simple 语言的 File Language，从而使基于内容的路由器更容易在 Spring XML 中实现，我们可以基于文件扩展进行路由，如下所示：

```
-
```

```
<from uri="file://input/orders"/>
  <choice>
    <when>
      <simple>${file:ext} == 'txt'</simple>
      <to uri="bean:orderService?method=handleTextFiles"/>
    </when>
    <when>
      <simple>${file:ext} == 'xml'</simple>
      <to uri="bean:orderService?method=handleXmlFiles"/>
    </when>
    <otherwise>
      <to uri="bean:orderService?method=handleOtherFiles"/>
    </otherwise>
  </choice>
```

如果您使用 File 端点上的 fileName 选项使用 File 语言设置动态文件名，请确保使用替代语法（在 Camel 2.5 中获得）以避免与 Springs PropertyPlaceholderConfigurer 冲突。???

bundle-context.xml

```
<bean id="propertyPlaceholder"
class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="location" value="classpath:bundle-context.cfg" />
</bean>

<bean id="sampleRoute" class="SampleRoute">
    <property name="fromEndpoint" value="${fromEndpoint}" />
    <property name="toEndpoint" value="${toEndpoint}" />
</bean>
```

bundle-context.cfg

```
fromEndpoint=activemq:queue:test
toEndpoint=file://fileRoute/out?fileName=test-$simple{date:now:yyyyMMdd}.txt
```

注意我们在上面的 toEndpoint 中使用 $simple\{ } 语法。如果没有这样做，则有一个 clash，Spring 会抛出异常，如

```
org.springframework.beans.factory.BeanDefinitionStoreException:
Invalid bean definition with name 'sampleRoute' defined in class path resource
[bundle-context.xml]:
Could not resolve placeholder 'date:now:yyyyMMdd'
```

105.6. 依赖项

File 语言是 camel-core 的一部分。

第 106 章 FLATPACK 组件

从 Camel 版本 1.4 开始提供

Flatpack 组件支持通过 FlatPack 库进行固定宽度和分隔的文件解析。
注意： 此组件只支持从 flatpack 文件到对象模型消耗。您无法(yet)从 Object model 写入 flatpack 格式。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-flatpack</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

106.1. URI 格式

```
flatpack:[delim|fixed]:flatPackConfig.pzmap.xml[?options]
```

或对于不使用任何配置文件的分隔文件处理程序

```
flatpack:someName[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

106.2. URI 选项

Flatpack 组件没有选项。

Flatpack 端点使用 URI 语法进行配置：

```
flatpack:type:resourceUri
```

使用以下路径和查询参数：

106.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
type	是否使用固定还是分隔符	delim	FlatpackType
resourceUri	从 classpath 或文件系统中加载 flatpack 映射文件 **所需的** URL		字符串

106.2.2. 查询参数(25 参数)：

Name	描述	默认值	类型
allowShortLines (common)	允许行少于预期，并忽略额外的字符	false	布尔值
delimiter (common)	分隔文件的默认字符分隔符。	,	char
ignoreExtraColumns (common)	允许行大于预期，并忽略额外的字符	false	布尔值
ignoreFirstRecord (common)	对于分隔的文件（列标题）是否忽略第一行。	true	布尔值
splitRows (common)	设置组件以在解析后将每行作为单独的交换发送	true	布尔值
textQualifier (common)	分隔文件的文本限定符。		char
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler

Name	描述	默认值	类型
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map

Name	描述	默认值	类型
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

106.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 12 个选项，如下所列。

Name	描述	默认值	类型
camel.component.flatpack.enabled	启用 flatpack 组件	true	布尔值
camel.component.flatpack.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.dataformat.flatpack.allow-short-lines	允许行少于预期，并忽略额外的字符	false	布尔值
camel.dataformat.flatpack.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.flatpack.definition	flatpack pzmap 配置文件。在更简单的情况下可以省略，但首选使用 pzmap。		字符串
camel.dataformat.flatpack.delimiter	分隔符字符（正为，为，或类似字符）	,	字符串
camel.dataformat.flatpack.enabled	启用 flatpack dataformat	true	布尔值
camel.dataformat.flatpack.fixed	分隔或固定。默认为 false = 分隔	false	布尔值

Name	描述	默认值	类型
camel.dataformat.flatpack.ignore-extra-columns	允许行大于预期，并忽略额外的字符。	false	布尔值
camel.dataformat.flatpack.ignore-first-record	对于分隔的文件（列标题）是否忽略第一行。默认为 true。	true	布尔值
camel.dataformat.flatpack.parser-factory-ref	对 registry 中查询的自定义解析器工厂的引用		字符串
camel.dataformat.flatpack.text-qualifier	如果文本通过 字符限定。默认使用 quote 字符。		字符串

106.4. 例子

- ***flatpack:fixed:foo.pzmap.xml 使用 foo.pzmap.xml 文件配置创建一个固定宽度端点。***
- ***flatpack:delim:bar.pzmap.xml 使用 bar.pzmap.xml 文件配置创建一个分隔的端点。***
- ***flatpack:foo 创建一个分隔的端点，名为 foo，没有文件配置。***

106.5. 消息标头

Camel 将在 IN 信息上存储以下标头：

标头	描述
CamelFlatpackCounter	当前行索引。对于 **splitRows=false**，计数器是行总数。

106.6. 消息正文

*组件以 **org.apache.camel.component.flatpack.DataSetList** 对象的形式提供 **IN** 信息中的数据，该对象具有 **java.util.Map** 或 **java.util.List** 的转换器。*
通常，如果您一次处理一行，则您希望 映射 (splitRows=true)。将 List 用于整个内容

(splitRows=false)，其中列表中的每个元素都是 映射。
每个 映射 包含列名称的密钥及其对应的值。

例如，从以下示例中获取 firstname：

```
Map row = exchange.getIn().getBody(Map.class);
String firstName = row.get("FIRSTNAME");
```

但是，您也可以将其作为 列表 （即使 splitRows=true也是如此）。同一示例：

```
List data = exchange.getIn().getBody(List.class);
Map row = (Map)data.get(0);
String firstName = row.get("FIRSTNAME");
```

106.7. 标头和垃圾记录

支持 Flatpack 中的标头和垃圾语的概念。但是，您必须使用 固定记录 ID：

- 标头 记录（必须为小写）
- trailer 记录（必须为小写）

以下示例演示了这个事实，我们有一个标题和垃圾语。如果不需要，可以省略其中一个或两个。

```
<RECORD id="header" startPosition="1" endPosition="3" indicator="HBT">
   <COLUMN name="INDICATOR" length="3"/>
   <COLUMN name="DATE" length="8"/>
</RECORD>

<COLUMN name="FIRSTNAME" length="35" />
<COLUMN name="LASTNAME" length="35" />
<COLUMN name="ADDRESS" length="100" />
<COLUMN name="CITY" length="100" />
<COLUMN name="STATE" length="2" />
<COLUMN name="ZIP" length="5" />

<RECORD id="trailer" startPosition="1" endPosition="3" indicator="FBT">
   <COLUMN name="INDICATOR" length="3"/>
   <COLUMN name="STATUS" length="7"/>
</RECORD>
```

106.8. 使用端点

常见的用例是将文件发送到此端点，以便在单独的路由中进一步处理。例如：

```
<camelContext xmlns="http://activemq.apache.org/camel/schema/spring">
  <route>
    <from uri="file://someDirectory"/>
    <to uri="flatpack:foo"/>
  </route>

  <route>
    <from uri="flatpack:foo"/>
    ...
  </route>
</camelContext>
```

您还可以将创建的每个消息的有效负载转换为 映射 以方便 Bean 集成

106.9. FLATPACK DATAFORMAT

Flatpack 组件附带了 Flatpack 数据格式，可用于格式化固定宽度或分隔的文本消息到以 映射 形式为行列表。

- marshal = from List<Map<String, Object >> to OutputStream （可以转换为 String）
- unmarshal = 从 java.io.InputStream （如 File 或 String）到 java.util.List 作为 org.apache.camel.component.flatpack.DataSetList 实例。
  操作的结果将包含所有数据。如果您需要逐一处理每行，您可以使用 Splitter 来分割交换。

注意： Flatpack 库目前不支持 marshal 操作的标头和跟踪器。

106.10. 选项

数据格式有以下选项：

选项	默认	描述
定义	null	flatpack pzmap 配置文件。在更简单的情况下可以省略，但首选使用 pzmap。

选项	默认	描述
已修复	**false**	分隔或固定。
ignoreFirstRecord	**true**	对于分隔的文件（列标题）是否忽略第一行。
textQualifier	**"**	如果文本通过字符（如 **"** ）授权。
delimiter	**,**	分隔符字符（正为，为， 或类似字符）
parserFactory	**null**	使用默认的 Flatpack 解析器工厂。
allowShortLines	**false**	Camel 2.9.7 和 2.10.5 以后 ：允许将行小于预期，并忽略额外的字符。
ignoreExtraColumns	**false**	Camel 2.9.7 和 2.10.5 以后 ：允许将行长于预期，并忽略额外的字符。

106.11. 使用方法

要使用数据格式，只需实例化实例并在路由构建器中调用 marshal 或 unmarshal 操作：

```
FlatpackDataFormat fp = new FlatpackDataFormat();
fp.setDefinition(new ClassPathResource("INVENTORY-Delimited.pzmap.xml"));
...
from("file:order/in").unmarshal(df).to("seda:queue:neworder");
```

上面的示例将读取来自 order/in 文件夹的文件，并使用 Flatpack 配置文件 INVENTORY-Delimited.pzmap.xml （配置文件结构）来读取输入。结果是我们存储在 SEDA 队列中的 DataSetList 对象。

```
FlatpackDataFormat df = new FlatpackDataFormat();
df.setDefinition(new ClassPathResource("PEOPLE-FixedLength.pzmap.xml"));
df.setFixed(true);
```

```
df.setIgnoreFirstRecord(false);

from("seda:people").marshal(df).convertBodyTo(String.class).to("jms:queue:people");
```

在上面的代码中，我们将对象表示中的数据作为 映射 行 列表。Map 所在的行包含列名称，键和对应的值。此结构可以在 Java 代码中创建，例如处理器。我们根据 Flatpack 格式提取数据，并将结果转换为 String 对象并将其存储在 JMS 队列中。

106.12. 依赖项

要在 camel 路由中使用 Flatpack，您需要添加对实现此数据格式的 camel-flatpack 的依赖。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-flatpack</artifactId>
  <version>x.x.x</version>
</dependency>
```

106.13. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 107 章 FLATPACK DATAFORMAT

从 Camel 版本 2.1 开始提供

Flatpack 组件附带了 Flatpack 数据格式，可用于格式化固定宽度或分隔的文本消息到以 映射 形式为行列表。

- *marshal = from List<Map<String, Object >> to OutputStream （可以转换为 String）*
- *unmarshal = 从 java.io.InputStream （如 File 或 String）到 java.util.List 作为 org.apache.camel.component.flatpack.DataSetList 实例。*
  操作的结果将包含所有数据。如果您需要逐一处理每行，您可以使用 Splitter 来分割交换。

注意： Flatpack 库目前不支持 marshal 操作的标头和跟踪器。

107.1. 选项

Flatpack dataformat 支持 9 个选项，如下所列。

Name	默认值	Java 类型	描述
定义		字符串	flatpack pzmap 配置文件。在更简单的情况下可以省略，但首选使用 pzmap。
已修复	false	布尔值	分隔或固定。默认为 false = 分隔
ignoreFirstRecord	true	布尔值	对于分隔的文件（列标题）是否忽略第一行。默认为 true。
textQualifier		字符串	如果文本通过 字符限定。默认使用 quote 字符。
delimiter	,	字符串	分隔符字符（正为，为，或类似字符）
allowShortLines	false	布尔值	允许行少于预期，并忽略额外的字符
ignoreExtraColumns	false	布尔值	允许行大于预期，并忽略额外的字符。
parserFactoryRef		字符串	对 registry 中查询的自定义解析器工厂的引用

Name	默认值	Java 类型	描述
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

107.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 12 个选项，如下所列。

Name	描述	默认值	类型
camel.component.flatpack.enabled	启用 flatpack 组件	true	布尔值
camel.component.flatpack.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.dataformat.flatpack.allow-short-lines	允许行少于预期，并忽略额外的字符	false	布尔值
camel.dataformat.flatpack.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.flatpack.definition	flatpack pzmap 配置文件。在更简单的情况下可以省略，但首选使用 pzmap。		字符串
camel.dataformat.flatpack.delimiter	分隔符字符（正为，为，或类似字符）	,	字符串
camel.dataformat.flatpack.enabled	启用 flatpack dataformat	true	布尔值
camel.dataformat.flatpack.fixed	分隔或固定。默认为 false = 分隔	false	布尔值
camel.dataformat.flatpack.ignore-extra-columns	允许行大于预期，并忽略额外的字符。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.flatpack.ignore-first-record	对于分隔的文件（列标题）是否忽略第一行。默认为 true。	true	布尔值
camel.dataformat.flatpack.parser-factory-ref	对 registry 中查询的自定义解析器工厂的引用		字符串
camel.dataformat.flatpack.text-qualifier	如果文本通过 字符限定。默认使用 quote 字符。		字符串

ND

107.3. 使用方法

要使用数据格式，只需实例化实例并在路由构建器中调用 marshal 或 unmarshal 操作：

```
FlatpackDataFormat fp = new FlatpackDataFormat();
fp.setDefinition(new ClassPathResource("INVENTORY-Delimited.pzmap.xml"));
...
from("file:order/in").unmarshal(df).to("seda:queue:neworder");
```

上面的示例将读取来自 order/in 文件夹的文件，并使用 Flatpack 配置文件 INVENTORY-Delimited.pzmap.xml （配置文件结构）来读取输入。结果是我们存储在 SEDA 队列中的 DataSetList 对象。

```
FlatpackDataFormat df = new FlatpackDataFormat();
df.setDefinition(new ClassPathResource("PEOPLE-FixedLength.pzmap.xml"));
df.setFixed(true);
df.setIgnoreFirstRecord(false);

from("seda:people").marshal(df).convertBodyTo(String.class).to("jms:queue:people");
```

在上面的代码中，我们将对象表示中的数据作为 映射 行 列表。Map 所在的行包含列名称，键和对应的值。此结构可以在 Java 代码中创建，例如处理器。我们根据 Flatpack 格式提取数据，并将结果转换为 String 对象并将其存储在 JMS 队列中。

107.4. 依赖项

要在 camel 路由中使用 Flatpack，您需要添加对实现此数据格式的 camel-flatpack 的依赖。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-flatpack</artifactId>
  <version>x.x.x</version>
</dependency>
```

第 108 章 APACHE FLINK 组件

从 Camel 版本 2.18 开始提供

本文档页面涵盖了 Apache Camel 的 Apache Flink 组件。camel-flink 组件在 Camel 连接器和 Flink 任务之间提供一个桥接。
此 Camel Flink 连接器提供了一种从各种传输路由消息的方法，动态选择要执行的 flink 任务，使用传入消息作为任务的输入数据，最后将结果传送回 Camel 管道。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-flink</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

108.1. URI 格式

目前，Flink 组件只支持 Producers。可以创建 DataSet、DataStream 作业。

```
flink:dataset?dataset=#myDataSet&dataSetCallback=#dataSetCallback
flink:datastream?datastream=#myDataStream&dataStreamCallback=#dataStreamCallback
```

FlinkEndpoint Options

Apache Flink 端点使用 URI 语法进行配置：

```
flink:endpointType
```

使用以下路径和查询参数：

108.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
endpointType	端点 **所需的** 类型(dataset、datastream)。		EndpointType

108.1.2. 查询参数(6 参数)：

Name	描述	默认值	类型
collect (producer)	指明是否应收集或计算结果。	true	布尔值
dataset (producer)	到计算的数据集。		DataSet
dataSetCallback (producer)	对 DataSet 执行操作的功能。		DataSetCallback
datastream (producer)	到计算的数据流。		DataStream
dataStreamCallback (producer)	对 DataStream 执行操作的功能。		DataStreamCallback
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

108.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
camel.component.flink.data-set	到计算的数据集。选项是 org.apache.flink.api.java.DataSet 类型。		字符串
camel.component.flink.data-set-callback	对 DataSet 执行操作的功能。选项是 org.apache.camel.component.flink.DataSetCallback 类型。		字符串
camel.component.flink.data-stream	到计算的数据流。选项是 org.apache.flink.streaming.api.datastream.DataStream 类型。		字符串
camel.component.flink.data-stream-callback	对 DataStream 执行操作的功能。选项是 org.apache.camel.component.flink.DataStreamCallback 类型。		字符串

Name	描述	默认值	类型
camel.component.flink.enabled	启用 flink 组件	true	布尔值
camel.component.flink.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

108.3. FLINKCOMPONENT OPTIONS

Apache Flink 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
dataset (producer)	到计算的数据集。		DataSet
datastream (producer)	到计算的数据流。		DataStream
dataSetCallback (producer)	对 DataSet 执行操作的功能。		DataSetCallback
dataStreamCallback (producer)	对 DataStream 执行操作的功能。		DataStreamCallback
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

108.4. FLINK DATASET CALLBACK

```
@Bean
public DataSetCallback<Long> dataSetCallback() {
    return new DataSetCallback<Long>() {
        public Long onDataSet(DataSet dataSet, Object... objects) {
            try {
                dataSet.print();
                return new Long(0);
            } catch (Exception e) {
                return new Long(-1);
            }
        }
    };
}
```

108.5. FLINK DATASTREAM CALLBACK

```
@Bean
public VoidDataStreamCallback dataStreamCallback() {
   return new VoidDataStreamCallback() {
      @Override
      public void doOnDataStream(DataStream dataStream, Object... objects) throws Exception
{
         dataStream.flatMap(new Splitter()).print();

         environment.execute("data stream test");
      }
   };
}
```

108.6. CAMEL-FLINK PRODUCER 调用

```
CamelContext camelContext = new SpringCamelContext(context);

String pattern = "foo";

try {
   ProducerTemplate template = camelContext.createProducerTemplate();
   camelContext.start();
   Long count = template.requestBody("flink:dataSet?
dataSet=#myDataSet&dataSetCallback=#countLinesContaining", pattern, Long.class);
   } finally {
      camelContext.stop();
   }
```

108.7. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 109 章 FOP 组件

从 Camel 版本 2.10 开始提供

FOP 组件允许您使用 Apache FOP 将消息呈现为不同的输出格式。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-fop</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

109.1. URI 格式

```
fop://outputFormat?[options]
```

109.2. 输出格式

主要输出格式是 PDF，但也支持其他 输出格式 ：

name	output Format	description
PDF	application/pdf	便携式文件格式
PS	application/postscript	adobe Postscript
PCL	application/x-pcl	打印机控制语言
PNG	image/png	PNG 镜像

name	output Format	description
JPEG	image/jpeg	JPEG 镜像
SVG	image/svg+xml	可扩展向量图形
XML	application/X-fop-areatree	区域树表示
MIF	application/mif	FrameMaker's MIF
RTF	application/rtf	丰富的文本格式
TXT	text/plain	文本

有效输出格式的完整列表 可在此处找到

109.3. 端点选项

FOP 组件没有选项。

FOP 端点使用 URI 语法进行配置：

```
fop:outputType
```

使用以下路径和查询参数：

109.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
outputType	**必需** ：主输出格式是 PDF，但也支持其他输出格式。		FopOutputType

109.3.2. 查询参数(3 参数)：

Name	描述	默认值	类型
fopFactory (producer)	允许使用 org.apache.fop.apps.FopFactory 的自定义配置或实现。		FopFactory
userConfigURL (producer)	可以从 classpath 或 filesystem 加载的配置文件的位置。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

109.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.fop.enabled	启用 fop 组件	true	布尔值
camel.component.fop.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

带有以下 结构的 配置文件的位置：默认情况下，从 Camel 2.12 开始从 classpath 加载。您可以使用 file: 或 classpath: 作为前缀，从 file 或 classpath 加载资源。在以前的版本中，该文件始终从文件系统中加载。

fopFactory

允许使用 org.apache.fop.apps.FopFactory 的自定义配置或实施。

109.5. 消息操作

name	默认值	description
CamelFop.Output.Format		覆盖该消息的输出格式
CamelFop.Encrypt.userPassword		PDF 用户密码
CamelFop.Encrypt.ownerPassword		PDF 所有者密码
CamelFop.Encrypt.allowPrint	true	允许打印 PDF
CamelFop.Encrypt.allowCopyContent	true	允许复制 PDF 的内容
CamelFop.Encrypt.allowEditContent	true	允许编辑 PDF 的内容
CamelFop.Encrypt.allowEditAnnotations	true	允许编辑 PDF 的注解
CamelFop.Render.producer	Apache FOP	生成文档的系统/软件的元数据元素

name	默认值	description
CamelFop.Render.creator		创建文档的用户的元数据元素
CamelFop.Render.creationDate		创建日期
CamelFop.Render.author		文档内容的作者
CamelFop.Render.title		文档的标题
CamelFop.Render.subject		文档的主题
CamelFop.Render.keywords		适用于本文档的关键字集

109.6. EXAMPLE

以下是一个示例路由，它呈现来自 xml 数据和 xslt 模板的 PDF 文件，并在目标文件夹中保存 PDF 文件：

```
from("file:source/data/xml")
    .to("xslt:xslt/template.xsl")
    .to("fop:application/pdf")
    .to("file:target/data");
```

如需更多信息，请参阅这些资源...

109.7. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 110 章 FREEMARKER 组件

从 Camel 版本 2.10 开始提供

freemarker: 组件允许使用 FreeMarker 模板处理消息。这在使用 Templating 生成请求的响应时是理想的选择。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-freemarker</artifactId>
  <version>x.x.x</version> <!-- use the same version as your Camel core version -->
</dependency>
```

110.1. URI 格式

```
freemarker:templateName[?options]
```

其中 templateName 是要调用的模板的 classpath-local URI，或者远程模板的完整 URL （例如：file://folder/myfile.ftl）。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

110.2. 选项

Freemarker 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
allowContextMapAll (producer)	设置上下文映射是否应该允许访问所有详情。默认情况下，只能访问消息正文和标头。可以启用此选项以完全访问当前 Exchange 和 CamelContext。这样做会带来潜在的安全风险，因为这会打开对 CamelContext API 的完整功能的访问。	false	布尔值

Name	描述	默认值	类型
allowTemplateFromHeader (producer)	是否允许使用来自标头的资源模板（默认 false）。**启用此选项具有安全原因。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。**	false	布尔值
配置 （高级）	使用现有的 freemarker.template.Configuration 实例作为配置。		配置
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Freemarker 端点使用 URI 语法进行配置：

```
freemarker:resourceUri
```

使用以下路径和查询参数：

110.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceUri	资源 **所需的** 路径。您可以使用这些协议(classpath 为 default)使用： classpath、file、http、ref 或 bean. classpath、file 和 http 加载资源的前缀。ref 将查找 registry 中的资源。bean 将调用 bean 以供用作资源的方法。对于 bean，您可以在点后指定方法名称，如 bean:myBean.myMethod。		字符串

110.2.2. 查询参数(7 参数)：

Name	描述	默认值	类型
allowContextMapAll (producer)	设置上下文映射是否应该允许访问所有详情。默认情况下，只能访问消息正文和标头。可以启用此选项以完全访问当前 Exchange 和 CamelContext。这样做会带来潜在的安全风险，因为这会打开对 CamelContext API 的完整功能的访问。	false	布尔值

Name	描述	默认值	类型
allowTemplateFromHeader (producer)	是否允许使用来自标头的资源模板（默认 false）。**启用此选项具有安全原因。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。**	false	布尔值
配置 （生成者）	设置要使用的 Freemarker 配置		配置
contentCache (producer)	设置是否使用资源内容缓存	false	布尔值
encoding (producer)	设置用于加载模板文件的编码。		字符串
templateUpdateDelay (producer)	加载模板资源将保留在缓存中的秒数。		int
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

110.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.freemarker.configuration	使用现有的 freemarker.template.Configuration 实例作为配置。选项是一个 freemarker.template.Configuration 类型。		字符串
camel.component.freemarker.enabled	启用 freemarker 组件	true	布尔值
camel.component.freemarker.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

110.4. HEADERS

在 FreeMarker 评估过程中设置的标头会返回到消息，并作为标头添加。这提供了 FreeMarker 组件向 Message 返回值的机制。

例如：在 FreeMarker 模板中设置标头值：

```
${request.setHeader('fruit', 'Apple')}
```

标头 fruit 现在可从 message.out.headers 访问。

110.5. FREEMARKER CONTEXT

Camel 将在 FreeMarker 上下文(just a Map)中提供交换信息。Exchange 作为以下内容传输：

key	value
交换	**Exchange** 本身。
exchange.properties	**Exchange** 属性.
标头	In 消息的标头。
camelContext	Camel 上下文.
Request （请求）	In 消息。
正文（body）	In 消息正文。
响应	Out 消息（仅适用于 InOut 消息交换模式）。

在 Camel 2.14 中，您可以在消息标头中设置自定义 FreeMarker 上下文，其键为 "CamelFreemarkerDataModel" like this

```
Map<String, Object> variableMap = new HashMap<String, Object>();
variableMap.put("headers", headersMap);
variableMap.put("body", "Monday");
variableMap.put("exchange", exchange);
exchange.getIn().setHeader("CamelFreemarkerDataModel", variableMap);
```

110.6. 热重新载入

FreeMarker 模板资源默认为 file 和 classpath 资源(expanded jar) 不 热重新加载。如果您设置了 contentCache=false，则 Camel 不会缓存资源，从而启用了热重新加载。此方案可用于开发。

110.7. 动态模板

Camel 提供了两个标头，您可以为模板或模板内容本身定义不同的资源位置。如果设置了其中任何标头，则 Camel 会通过端点配置的资源使用此标头。这可让您在运行时提供动态模板。

标头	类型	描述	支持版本
FreemarkerConstants.FREEMARKER_RESOURCE	org.springframework.core.io.Resource	模板资源	⇐ 2.1
FreemarkerConstants.FREEMARKER_RESOURCE_URI	字符串	要使用的模板资源的 URI，而不是配置的端点。	>= 2.1
FreemarkerConstants.FREEMARKER_TEMPLATE	字符串	要使用的模板而不是配置的端点。	>= 2.1

110.8. SAMPLES

例如，您可以使用类似如下的内容：

```
from("activemq:My.Queue").
  to("freemarker:com/acme/MyResponse.ftl");
```

要使用 FreeMarker 模板模拟对 InOut 消息交换的消息的响应（其中有一个 JMSReplyTo 标头）。

如果要使用 InOnly 并使用消息并将其发送到您可以使用的另一个目的地：

```
from("activemq:My.Queue").
  to("freemarker:com/acme/MyResponse.ftl").
  to("activemq:Another.Queue");
```

和 要禁用内容缓存，例如，对于应该热重新加载 .ftl 模板的开发使用情况：

```
from("activemq:My.Queue").
  to("freemarker:com/acme/MyResponse.ftl?contentCache=false").
  to("activemq:Another.Queue");
```

和基于文件的资源：

```
from("activemq:My.Queue").
  to("freemarker:file://myfolder/MyResponse.ftl?contentCache=false").
  to("activemq:Another.Queue");
```

在 Camel 2.1 中，可以指定组件应该通过标头动态使用的模板，例如：

```
from("direct:in").

setHeader(FreemarkerConstants.FREEMARKER_RESOURCE_URI).constant("path/to/my/template.ftl").
  to("freemarker:dummy?allowTemplateFromHeader=true");
```

警告

启用 allowTemplateFromHeader 选项具有安全等级。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。

110.9. 电子邮件示例

在本例中，我们希望使用 FreeMarker 模板进行订单确认电子邮件。电子邮件模板在 FreeMarker 中布局，如下所示：

```
Dear ${headers.lastName}, ${headers.firstName}

Thanks for the order of ${headers.item}.
```

```
Regards Camel Riders Bookstore
${body}
```

和 java 代码：

110.10. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 111 章 FTP 组件

从 Camel 版本 1.1 开始提供

这个组件可让您通过 FTP 和 SFTP 协议访问远程文件系统。

当从远程 FTP 服务器 消耗时，请确保在下面阅读标题为 Default 的小节，以了解与使用文件相关的详细信息。

不支持 绝对路径。Camel 2.16 将通过修剪 directoryname 中的所有前导斜杠将绝对路径转换为相对路径。日志中会输出 WARN 消息。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-ftp</artifactId>
  <version>x.x.x</version>See the documentation of the Apache Commons
  <!-- use the same version as your Camel core version -->
</dependency>
```

111.1. URI 格式

```
ftp://[username@]hostname[:port]/directoryname[?options]
sftp://[username@]hostname[:port]/directoryname[?options]
ftps://[username@]hostname[:port]/directoryname[?options]
```

其中 directoryname 代表底层目录。目录名称是相对路径。不支持 绝对路径。相对路径可以包含嵌套文件夹，如 /inbox/us。

对于 Camel 2.16 之前的 Camel 版本，directoryName 必须已存在，因为此组件不支持 autoCreate 选项（文件组件的作用）。其原因是，其 FTP 管理员(FTP 服务器)任务用于正确设置用户帐户，以及具有正确文件权限的主目录等。

对于 Camel 2.16，支持autoCreate 选项。当消费者启动时，在调度轮询前，需要执行额外的 FTP 操作来创建为端点配置的目录。autoCreate 的默认值为 true。

如果未提供 用户名，则将尝试使用任何密码进行 匿名 登录。
如果没有提供 端口号，Camel 将根据协议(ftp = 21、sftp = 22、ftps = 2222)提供默认值。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

该组件对实际 FTP 工作使用两个不同的库。FTP 和 FTPS 使用 Apache Commons Net，而 SFTP 使用 JCraft JSCH。

FTPS 组件仅适用于 Camel 2.2 或更新版本。
FTPS （也称为 FTP Secure）是 FTP 的扩展，它增加了对传输层安全(TLS)和安全套接字层(SSL)加密协议的支持。

111.2. URI 选项

以下选项专用于 FTP 组件。

FTP 组件没有选项。

FTP 端点使用 URI 语法进行配置：

```
ftp:host:port/directoryName
```

使用以下路径和查询参数：

111.2.1. 路径参数(3 参数)：

Name	描述	默认值	类型
主机	需要的 FTP 服务器的主机名		字符串
port	FTP 服务器的端口		int
directoryName	起始目录		字符串

111.2.2. 查询参数(110 参数)：

Name	描述	默认值	类型
binary (common)	指定文件传输模式 BINARY 或 ASCII。默认为 ASCII (false)。	false	布尔值
charset (common)	此选项用于指定文件的编码。您可以在消费者上使用此选项，指定文件的编码，允许 Camel 了解在访问文件内容时应加载文件内容。在编写文件时，您可以使用此选项指定写入该文件的 charset。请记住，在编写文件 Camel 时，可能需要将消息内容读取到内存中，以便能将数据转换为配置的 charset，因此如果您有大型消息，请不要使用它。		字符串
disconnect (common)	使用后是否从远程 FTP 服务器断开连接。断开连接将仅断开当前与 FTP 服务器的连接。如果您有要停止的消费者，则需要停止消费者/路由。	false	布尔值
doneFileName (common)	生产者：如果提供，则 Camel 将在写入原始文件时写入第 2 个 done 文件。完成的文件将为空。这个选项配置要使用的文件名。您可以指定一个固定名称。或者，您可以使用动态占位符。完成的文件始终将写入与原始文件相同的文件夹中。consumer：如果提供，Camel 仅在文件存在时使用文件。这个选项配置要使用的文件名。您可以指定一个固定名称。或者您可以使用动态占位符。已完成的文件始终预期位于与原始文件相同的文件夹中。仅支持 $file.name 和 $file.name.noext 作为动态占位符。		字符串
filename (common)	使用 File Language 等表达式来动态设置文件名。对于消费者，它将用作文件名过滤器。对于生成者，它用于评估要写入的文件名。如果设置了表达式，它将优先于 CamelFileName 标头。（注：标题本身也可以是表达式）。表达式选项支持 String 和 Expression 类型。如果表达式是 String 类型，则始终使用 File 语言来评估它。如果表达式是 Expression 类型，则使用指定的 Expression 类型 - 例如，允许您使用 OGNL 表达式。对于消费者，您可以使用它过滤文件名，因此您可以使用 File Language 语法使用当前的文件：mydata-$date:now:yyyyMMdd.txt。生产者支持 CamelOverruleFileName 标头，它优先于任何现有的 CamelFileName 标头；CamelOverruleFileName 是一个只使用一次的标头，它可让您更轻松地临时存储 CamelFileName，且之后必须恢复它。		字符串
passiveMode (common)	设置被动模式连接。默认为活跃的模式连接。	false	布尔值
分隔符 (common)	设置要使用的路径分隔符。Unix = 使用 unix 样式路径分隔符 Windows = 使用窗口样式路径分隔符 Auto = （默认）在文件名中使用现有路径分隔符	UNIX	PathSeparator

Name	描述	默认值	类型
transferLoggingInterval Seconds (common)	配置在记录上传和下载操作的进度时使用的时间间隔（以秒为单位）。这在操作需要更长的时间时用于记录进度。	5	int
transferLoggingLevel (common)	配置在记录上传和下载操作的进度时使用的日志记录级别。	DEBUG	LoggingLevel
transferLoggingVerbose (common)	配置上传和下载操作进度的执行详细(ine grained)日志记录。	false	布尔值
fastExistsCheck (common)	如果将此选项设置为 true，camel-ftp 将直接使用列表文件来检查该文件是否存在。由于某些 FTP 服务器可能不支持直接列出文件，如果选项为 false，camel-ftp 将使用旧方式列出该目录并检查该文件是否存在。这个选项还影响 readLock=changed，以控制它是否执行快速检查来更新文件信息。如果 FTP 服务器包含大量文件，则可以使用它来加快进程。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
delete (consumer)	如果为 true，则在成功处理后会删除该文件。	false	布尔值
moveFailed (consumer)	根据简单语言设置移动失败表达式。例如，要将文件移到 .error 子目录中，请使用： .error。注意：将文件移到故障位置 Camel 时，将处理错误，并且不会再次获取该文件。		字符串
noop (consumer)	如果为 true，则文件不会以任何方式移动或删除。这个选项对只读数据或者 ETL 类型要求很好。如果 noop=true，Camel 也设置 idempotent=true，以避免再次消耗相同的文件。	false	布尔值
preMove (consumer)	表达式（如文件语言）用于在处理之前动态设置文件名。例如，要将 in-progress 文件移到 order 目录中，将此值设置为 order。		字符串
preSort (consumer)	启用预先排序后，消费者将在轮询期间对文件和目录名称进行排序，该名称是从文件系统检索的。如果需要按排序顺序对文件进行操作，您可能需要执行此操作。指示器在消费者开始过滤前执行，并接受 Camel 处理的文件。这个选项是 default=false 表示禁用。	false	布尔值
递归 (consumer)	如果某个目录，将查找所有子目录中的文件。	false	布尔值

Name	描述	默认值	类型
resumeDownload (consumer)	配置是否启用了恢复下载。这必须被 FTP 服务器支持（几乎所有 FTP 服务器都支持它）。此外，必须配置 localWorkDirectory 选项，以便下载的文件存储在本地目录中，且必须启用选项二进制文件，以支持恢复下载。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
streamDownload (consumer)	设置在使用本地工作目录时要使用的下载方法。如果设置为 true，则远程文件会在读取时流到路由。当设置为 false 时，远程文件会在发送到路由之前加载到内存中。	false	布尔值
directoryMustExist (consumer)	与 startingDirectoryMustExist 类似，但这在轮询递归子目录期间应用。	false	布尔值
download (consumer)	FTP 使用者是否应下载 文件。如果此选项设为 false，则消息正文将为空，但消费者仍会触发具有文件详情的 Camel Exchange，如文件名、文件大小等。只是不会下载该文件。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
handleDirectoryParserAbsoluteResult (consumer)	如果目录解析器生成了绝对路径，则允许使用者如何处理路径中子文件夹和文件的方式。这样做的原因是，有些 FTP 服务器可能会返回具有绝对路径的文件名，如果是，FTP 组件需要处理这个路径。	false	布尔值
ignoreFileNotFoundOrPermissionError (consumer)	是否忽略 when （尝试列出目录中的文件或下载文件时），但不存在，还是因为权限错误。默认情况下，当目录或文件不存在或权限不足时，会抛出异常。将这个选项设置为 true 可忽略它。	false	布尔值
inProgressRepository (consumer)	可插拔式 in-progress 存储库 org.apache.camel.spi.IdempotentRepository。in-progress 存储库用于考虑被消耗的当前正在进行的文件。默认使用基于内存的存储库。		IdempotentRepository

Name	描述	默认值	类型
localWorkDirectory (consumer)	在消耗时，可以使用本地工作目录直接将远程文件内容存储在本地文件中，以避免将内容加载到内存中。这很有用，如果您使用非常大的远程文件，因此可以节省内存。		字符串
onCompletionException Handler (consumer)	要使用自定义 org.apache.camel.spi.ExceptionHandler 来处理在完成过程中发生的任何抛出异常，消费者在其中执行提交或回滚。默认实施会将任何异常记录在 WARN 级别并忽略。		ExceptionHandler
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
processStrategy (consumer)	可插拔 org.apache.camel.component.file.GenericFileProcessStrategy 允许您实施自己的 readLock 选项或类似选项。也可以在使用文件之前满足特殊条件时使用，如存在特殊的就绪文件。如果设置了这个选项，则不会应用 readLock 选项。		GenericFileProcess s Strategy
receiveBufferSize (consumer)	仅由 FTPClient 使用的接收（下载）缓冲大小	32768	int
startingDirectoryMustExist (consumer)	起始目录是否必须存在。请注意，autoCreate 选项是默认启用的，这意味着如果起始目录不存在，则通常会自动创建该目录。您可以禁用 autoCreate 并启用它，以确保起始目录必须存在。如果目录 不存在，将抛出异常。	false	布尔值
useList (consumer)	下载文件时是否允许使用 LIST 命令。默认为 true。在某些情况下，您可能想要下载特定的文件，且不允许使用 LIST 命令，因此您可以将此选项设置为 false。请注意，在使用此选项时，要下载的特定文件不包括诸如文件大小、时间戳、权限等元数据数据信息，因为这些信息只能在 LIST 命令使用时检索。	true	布尔值

Name	描述	默认值	类型
fileExist (producer)	如果文件已存在具有相同名称的文件，该怎么办。override （默认值）替换现有文件。Append - 在现有文件中添加内容。fail - 抛出 GenericFileOperationException，表示已存在文件。ignore - 静默忽略问题，且不会覆盖现有文件，但假设一切都正常。Move - 选项需要使用 moveExisting 选项也可以配置。选项 eagerDeleteTargetFile 可用于控制在移动文件时执行什么操作，并且已存在文件，否则会导致移动操作失败。Move 选项将在编写目标文件之前移动任何现有文件。只有在使用 tempFileName 选项时，TryRename 才适用。这允许尝试将文件从临时名称重命名为实际名称，而无需执行任何存在的检查。在某些文件系统（特别是 FTP 服务器中）中，这个检查可能会更快。	override	GenericFileExist
flatten (producer)	flatten 用于扁平化任何前导路径的文件名路径，因此只用于文件名。这可让您以递归方式消耗到子目录中，但当您将文件写入另一个目录时，它们将写入单个目录。在生成者中将其设置为 true 强制执行 CamelFileName 标头中的任何文件名都会被剥离任何前导路径。	false	布尔值
jailStartingDirectory (producer)	用于存放（限制）将文件写入起始目录（和子目录）。这默认是启用的，不允许 Camel 将文件写入外部目录（从而更加安全）。您可以关闭此项，以允许将文件写入起始目录之外的目录，如父目录或 root 文件夹。	true	布尔值
moveExisting (producer)	当配置了 fileExist=Move 时，用于计算文件名的表达式（如 File Language）。要将文件移动到备份子目录中，请输入 backup。这个选项只支持以下 File Language 令牌：file:name, file:name.ext, file:name.noext, file:onlyname, file:onlyname.noext, file:ext, 和 file:parent。请注意，FTP 组件不支持 file:parent，因为 FTP 组件只能将任何现有文件移到基于当前 dir 的相对目录中。		字符串
tempFileName (producer)	与 tempPrefix 选项相同，但对临时文件名的命名提供更精细的控制，因为它使用 File Language。		字符串
tempPrefix (producer)	此选项用于使用临时名称写入文件，然后在写入完成后将其重命名为实际名称。可用于识别正在写入的文件，并避免消费者（不使用专用读取锁定）读取进度文件。上传大型文件时，FTP 通常使用。		字符串

Name	描述	默认值	类型
allowNullBody (producer)	用于指定在文件写入过程中是否允许 null 正文。如果设置为 true，则会创建一个空文件，当设为 false 时，并尝试向文件组件发送 null 正文，则 'Cannot null body to file.' 的 GenericFileWriteException 将会被抛出。如果将 fileExist 选项设置为 'Override'，则文件将被截断，如果设置为附加文件，则文件将保持不变。	false	布尔值
chmod (producer)	允许您在存储的文件上设置 chmod。例如 chmod=640。		字符串
disconnectOnBatchComplete (producer)	在 Batch upload 完成后是否断开与远程 FTP 服务器的连接。disconnectOnBatchComplete 只会断开当前与 FTP 服务器的连接。	false	布尔值
eagerDeleteTargetFile (producer)	是否强制删除任何现有目标文件。这个选项只适用于使用 fileExists=Override 和 tempFileName 选项。您可以使用它来禁用（将其设置为 false）在写入临时文件前删除目标文件。例如，您可以写入大文件，并且希望写入 temp 文件期间存在目标文件。这样可确保目标文件仅在最后一次删除之前，直到 temp 文件被重命名为目标文件名之前。这个选项还用于控制在启用了 fileExist=Move 时是否删除任何现有文件，并且存在现有的文件。如果此选项 copyAndDeleteOnRenameFails false，则会在移动操作前现有文件存在时抛出异常。	true	布尔值
keepLastModified (producer)	将保留源文件的最后修改的时间戳（若有）。将使用 Exchange.FILE_LAST_MODIFIED 标头来定位时间戳。此标头可以包含带有时间戳的 java.util.Date 或 long。如果存在时间戳，并且启用了 选项，它将在写入的文件上设置此时间戳。注：此选项仅适用于文件制作者。您不能将此选项与任何 ftp 生成者一起使用。	false	布尔值
moveExistingFileStrategy (producer)	策略(Custom Strategy)用于在配置 fileExist=Move 时使用，使用特殊命名令牌来移动文件。默认情况下，如果没有提供自定义策略，则使用实施		FileMoveExisting 策略
sendNoop (producer)	在将文件上传到 FTP 服务器之前，是否将 noop 命令作为预写检查发送。这默认是启用的，因为连接的验证仍然有效，这允许静默重新连接才能上传该文件。但是，如果这会导致问题，您可以关闭这个选项。	true	布尔值
activePortRange (advanced)	以活跃模式设置客户端侧端口范围。语法为：minPort-maxPort Both 端口号，例如 10000-19999，包含所有 1xxxx 端口。		字符串
auto create (advanced)	在文件的路径名称中自动创建缺少的目录。对于文件消费者，这意味着创建起始目录。对于生成者文件，这意味着文件应当要写入的目录。	true	布尔值

Name	描述	默认值	类型
bufferSize (advanced)	写入缓冲区大小（以字节为单位）。	131072	int
connectTimeout (advanced)	设置连接超时，以等待 FTPClient 和 JSCH 使用连接	10000	int
ftpClient (advanced)	使用自定义 FTPClient 实例		FTPClient
ftpClientConfig (advanced)	要使用 FTPClientConfig 的自定义实例来配置端点应使用的 FTP 客户端。		FTPClientConfig
ftpClientConfigParameters (advanced)	FtpComponent 用于为 FTPClientConfig 提供其他参数		Map
ftpClientParameters (advanced)	FtpComponent 用于为 FTPClient 提供其他参数		Map
maximumReconnectAttempts (advanced)	指定在尝试连接到远程 FTP 服务器时执行的最大重新连接尝试 Camel。使用 0 来禁用此行为。		int
reconnectDelay (advanced)	millis Camel 中的延迟将在执行重新连接尝试前等待。		long
siteCommand (advanced)	设置在成功登录后要执行的可选 site 命令。可以使用新行字符来分隔多个站点命令。		字符串
soTimeout (advanced)	仅为 FTPClient 设置 so 超时	300000	int
stepwise (advanced)	设置我们是否应在下载文件时遍历文件结构的同时，还是将文件上传到目录时逐步更改目录。例如，如果您因为安全原因无法在 FTP 服务器上更改目录时禁用此功能。	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
throwExceptionOnConnect Failed (advanced)	如果连接失败（忽略）时，应该抛出异常，默认异常不会被抛出，并记录 WARN。您可以使用它来启用异常，并处理 org.apache.camel.spi.PollingConsumerPollStrategy rollback 方法所引发的异常。	false	布尔值

Name	描述	默认值	类型
超时 (advanced)	设置用于仅由 FTPClient 使用的数据超时	30000	int
antExclude (filter)	Ant 风格过滤器排除。如果使用 antInclude 和 antExclude，则 antExclude 优先于 antInclude。可以使用逗号分隔的格式指定多个排除项。		字符串
antFilterCaseSensitive (filter)	在t 过滤器上设置问题单敏感标志	true	布尔值
antInclude (filter)	Ant 风格过滤器包括：可以使用逗号分隔的格式指定多个包含。		字符串
eagerMaxMessagesPerPoll (filter)	允许控制 maxMessagesPerPoll 的限制是 eager 还是 not。如果 eager，则限制是在扫描文件时。其中 false 会扫描所有文件，然后执行排序。将此选项设置为 false 可首先对所有文件进行排序，然后限制轮询。请注意，这需要更高的内存用量，因为所有文件详细信息都在内存中执行排序。	true	布尔值
exclude (filter)	用于排除文件，如果文件名与 regex 模式匹配（匹配是 in-senstive）。请注意，如果您使用加号等符号，如果将其配置为端点 uri，则需要使用 RAW （）语法进行配置。有关配置端点 uris 时，请参阅更多详情		字符串
Filter (filter)	可插拔过滤器为 org.apache.camel.component.file.GenericFileFilter 类。如果过滤在 accept （）方法中返回 false，则会跳过文件。		GenericFileFilter
filterDirectory (filter)	根据简单语言过滤目录。例如，要过滤当前日期，您可以使用简单的日期模式，如 $date:now:yyMMdd		字符串
filterFile (filter)	根据简单语言过滤文件。例如，要过滤文件大小，您可以使用 $file:size 5000		字符串
idempotent (filter)	使用 Idempotent Consumer EIP 模式的选项，让 Camel 跳过已处理的文件。默认情况下，将使用基于内存的 LRUCache，其中包含 1000 个条目。如果 noop=true 然后会启用幂等，以避免再次消耗同样的文件。	false	布尔值
idempotentKey (filter)	使用自定义幂等密钥。默认情况下使用文件的绝对路径。您可以使用 File Language，例如使用文件名和文件大小，您可以执行以下操作：idempotentKey=$file:name-$file:size		字符串

Name	描述	默认值	类型
idempotentRepository (filter)	一个可插拔式存储库 org.apache.camel.spi.IdempotentRepository，如果未指定 none，则默认使用 MemoryMessageIdRepository，并且幂等性为 true。		IdempotentRepository
include (filter)	用于包含文件，如果文件名匹配 regex 模式（匹配不区分大小写）。请注意，如果您使用加号等符号，如果将其配置为端点 uri，则需要使用 RAW （）语法进行配置。有关配置端点 uris 时，请参阅更多详情		字符串
maxDepth (filter)	递归处理目录时要遍历的最大深度。	2147483647	int
maxMessagesPerPoll (filter)	定义每个轮询要收集的最大消息。默认不设置最大值。可用于设置限制，例如 1000，以避免在启动有数千个文件的服务器时避免。设置一个 0 或 negative 值来禁用它。注意：如果使用这个选项，则 File 和 FTP 组件将在任何排序之前限制。例如，如果您有 100000 文件并使用 maxMessagesPerPoll=500，则只有第一个 500 文件会被获取，然后排序。您可以使用 eagerMaxMessagesPerPoll 选项，并将其设置为 false 以允许先扫描所有文件，然后随后排序。		int
minDepth (filter)	在递归处理目录时开始处理的最小深度。使用 minDepth=1 表示基础目录。使用 minDepth=2 表示第一个子目录。		int
Move (filter)	表达式（如简单语言）用于在处理后动态设置文件名。要将文件移动到 .done 子目录中，只需输入 .done。		字符串
exclusiveReadLockStrategy (lock)	可插拔 read-lock 作为 org.apache.camel.component.file.GenericFileExclusiveReadLockStrategy 实施。		GenericFileExclusiveReadLockStrategy

Name	描述	默认值	类型
readLock (lock)	供消费者使用，只有在文件上具有独占的 read-lock 时才轮询文件（例如，该文件没有正在进行中或被写入）。Camel 将等到授予文件锁定为止。这个选项提供了 build in strategy: none - No read lock is in use markerFile - Camel 创建一个标记文件 (fileName.camelLock)，然后保存锁定。这个选项不适用于 FTP 组件更改 - 更改方法是使用文件长度/修改时间戳来检测当前是否复制该文件。至少将使用 1 sec 确定此选项，因此此选项无法像其他进程一样快速使用文件，但可以更可靠，因为 JDK IO API 始终无法确定文件目前是否被其他进程使用。可以使用 readLockCheckInterval 选项来设置检查频率。 fileLock - 用于使用 java.nio.channels.FileLock。这个选项在 Windows 或 FTP 组件上不可用。当通过 mount/share 访问远程文件系统时，应该避免使用这种方法，除非文件系统支持分布式文件锁定。rename 是使用尝试重命名该文件的测试，如果我们可以获得专用只读锁。幂等的 - （仅限文件组件）幂等性是将 idempotentRepository 用作读锁。这允许使用支持集群读取锁定（如果幂等存储库实现支持集群）。idid-changed - （仅限文件组件）幂等更改，使用 idempotentRepository 并更改作为组合的 read-lock。这允许使用读取锁定，如果幂等存储库实施支持集群。jid-rename - （仅限文件组件） idempotent-rename 使用 idempotentRepository，并重命名作为组合的 read-lock。这允许使用支持集群的读取锁定（如果幂等存储库实现支持）。注意：各种读取锁定并不适用于在集群模式下工作，其中不同节点上的并发用户与共享文件系统上的相同文件竞争。markerFile 使用接近 atomic 操作来创建空的标记文件，但不保证在集群中工作。fileLock 可能会更好地工作，但文件系统需要支持分布式文件锁定，等等。如果幂等性存储库支持集群，如 Hazelcast 组件或 Infinispan，则使用幂等读取锁定支持集群。	none	字符串
readLockCheckInterval (lock)	读锁在 millis 中，如果读锁定支持，则间隔为 millis。这个间隔用于在尝试获取读锁定之间休眠。例如，在使用更改的读锁定时，您可以将更高的间隔周期设置为 cater 以慢速写入。如果生成者非常慢，则默认 1 sec. 可能太快。注意：对于 FTP，默认的 readLockCheckInterval 为 5000。readLockTimeout 值必须高于 readLockCheckInterval，但 thumb 规则是有一个超过 readLockCheckInterval 的 2 次或多次的超时。这是为了确保在达到超时前尝试获取锁进程允许随时读取的时间。	1000	long

Name	描述	默认值	类型
readLockDeleteOrphanLock Files (lock)	如果 Camel 没有正确关闭（如 JVM 崩溃），则应在启动时读取带有标记文件的读取锁定文件（可能已留在文件系统上）读取锁定。如果将此选项设置为 false，则任何孤立锁定文件将导致 Camel 不尝试选择该文件，这可能是因为另一个节点同时从同一个共享目录读取文件。	true	布尔值
readLockLogging Level (lock)	无法获取读取锁定时使用的日志记录级别。默认情况下，会记录 WARN。您可以更改此级别，例如将 OFF 更改为没有任何日志记录。这个选项只适用于 readLock 类型： changed, fileLock, idempotent, idempotent-changed, idempotent-rename, rename。	DEBUG	LoggingLevel
readLockMarkerFile (lock)	是否将标志文件与 changed、rename 或 exclusive read 锁定类型一起使用。默认情况下，还使用标记文件来保护获取相同文件的其他进程。通过将这个选项设置为 false 来关闭此行为。例如，如果您不希望将标记文件写入 Camel 应用程序的文件系统。	true	布尔值
readLockMinAge (lock)	这个选项只适用于 readLock=changed。它可指定文件在尝试获取读取锁定前必须经过的最短期限。例如，使用 readLockMinAge=300s 来要求文件最新 5 分钟。这可加快更改的读锁定速度，因为它将只尝试获取至少给定年龄的文件。	0	long
readLockMinLength (lock)	这个选项只适用于 readLock=changed。它允许您配置最小文件长度。默认情况下，Camel 预期文件包含数据，因此默认值为 1。您可以将这个选项设置为零，以允许消耗零长度文件。	1	long
readLockRemoveOnCommit (lock)	这个选项只适用于 readLock=idempotent。它允许在处理文件成功并且执行提交时指定是否从幂等存储库中删除文件名条目。默认情况下，文件不会被删除，以确保不会发生任何竞争条件，因此另一个活动节点可能会尝试获取该文件。相反，幂等存储库可能支持驱除策略，在 X 分钟后用于驱除文件名条目 - 这样可保证竞争条件没有问题。请参阅 readLockIdempotentReleaseDelay 选项的详情。	false	布尔值
readLockRemoveOnRollback (lock)	这个选项只适用于 readLock=idempotent。它允许在处理文件失败时指定是否从幂等存储库中删除文件名条目，并发生回滚。如果此选项为 false，则文件名称条目将被确认（就像文件确实是提交一样）。	true	布尔值

Name	描述	默认值	类型
readLockTimeout (lock)	read-lock 的 millis 中可选超时（如果 read-lock 支持）。如果无法授予 read-lock，并且触发超时，则 Camel 将跳过该文件。在下一次轮询 Camel 时，将再次尝试文件，此时可能会授予 read-lock。使用 0 或较低值来指示永久。目前，fileLock，改变并重命名支持超时。注意：对于 FTP，默认的 readLockTimeout 值为 20000 而不是 10000。readLockTimeout 值必须高于 readLockCheckInterval，但 thumb 规则是有一个超过 readLockCheckInterval 的 2 次或多次的超时。这是为了确保在达到超时前尝试获取锁进程允许随时读取的时间。	10000	long
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map

Name	描述	默认值	类型
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
Shuff le (sort)	影响文件列表（以随机顺序排序）	false	布尔值
排序(排序)	使用文件语言进行内置排序.支持嵌套排序，以便按文件名排序，并根据修改的日期作为第二组排序。		字符串
排序 （排序）	可插拔排序器作为 java.util.Comparator 类。		比较器
帐户 （安全）	用于登录的帐户		字符串
密码 (security)	用于登录的密码		字符串
用户名 (security)	用于登录的用户名		字符串

111.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.ftp.enabled	启用 ftp 组件	true	布尔值
camel.component.ftp.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

111.4. FTPS 组件默认信任存储

当使用与 FTPS 组件 SSL 相关的 ftpClient. 属性时，信任存储接受所有证书。如果您只想信任选择的证书，您必须使用 ftpClient.trustStore.xxx 选项或配置自定义的 ftpClient 选项来配置信任存储。

使用 sslContextParameters 时，信任存储由提供的 SSLContextParameters 实例的配置来管理。

您可以使用 ftpClient. 或 ftpClientConfig. 前缀直接在 URI 上配置 ftpClient 和 ftpClientConfig 上的附加选项。

例如，要将 FTPClient 上的 setDataTimeout 设置为 30 秒，您可以执行以下操作：

```
from("ftp://foo@myserver?password=secret&ftpClient.dataTimeout=30000").to("bean:foo");
```

您可以混合和匹配并使用两个前缀，例如配置日期格式或时区。

```
from("ftp://foo@myserver?
password=secret&ftpClient.dataTimeout=30000&ftpClientConfig.serverLanguageCode=fr").to(
"bean:foo");
```

您可以根据需要拥有任意数量的选项。

有关可能选项和更多详情，请参阅 Apache Commons FTP FTPClientConfig 的文档。和用于 Apache Commons FTP FTPClient。

如果您不想在 url 中有多个和长的配置，您可以通过让 Registry 中的 Camel 查找来指代 ftpClient 或 ftpClientConfig 要使用的 ftpClient 或 ftpClientConfig。

例如：

```
<bean id="myConfig" class="org.apache.commons.net.ftp.FTPClientConfig">
   <property name="lenientFutureDates" value="true"/>
   <property name="serverLanguageCode" value="fr"/>
</bean>
```

然后，当您在 url 中使用 # 表示法时，请让 Camel 查找这个 bean。

```
from("ftp://foo@myserver?password=secret&ftpClientConfig=#myConfig").to("bean:foo");
```

111.5. 例子

ftp://someone@someftpserver.com/public/upload/images/holiday2008?password=secret&binary=true

ftp://someoneelse@someotherftpserver.co.uk:12049/reports/2008/password=secret&binary=false
ftp://publicftpserver.com/download

111.6. 并发

FTP Consumer 不支持并发

FTP 使用者（具有相同端点）不支持并发（支持的 FTP 客户端不支持线程安全）。
您可以使用多个 FTP 用户从不同的端点轮询。它只是一个不支持并发用户的端点。

FTP 生成者 没有 此问题， 它支持并发性。

111.7. 更多信息

此组件是 File 组件的扩展。因此， 在 File 组件页面中有更多示例和详情。

111.8. 使用文件时的默认

默认情况下， FTP 使用者将保留使用的文件在远程 FTP 服务器上不动。如果要删除文件或将其移动到另一个位置，则必须明确进行配置。例如， 您可以使用 delete=true 删除文件， 或使用 move=.done 将文件移到隐藏完成的子目录中。

常规文件使用者不同，因为它默认会将文件移到 .camel 子目录中。默认情况下， Camel 不对 FTP 使用者执行此操作是，默认情况下它可能会缺少权限，从而能够移动或删除文件。

111.8.1. 限制

可以使用 readLock 选项强制 Camel 不 消耗当前正在写入的文件。但是，默认关闭这个选项，因为它要求用户具有写入访问权限。有关读取锁定的详情， 请查看 File2 中的选项表。
还有其他解决方案可以避免使用当前通过 FTP 编写的文件；例如， 您可以写入临时目标， 并在写入文件后移动该文件。

使用 move 或 preMove 选项移动文件时，文件仅限于 FTP_ROOT 文件夹。这样可防止您移动 FTP 区域之外的文件。如果要将文件移动到另一个区域，您可以使用 软链接并将文件移动到软链接文件夹中。

111.9. 消息标头

以下消息标头可用于影响组件的行为

标头	描述
CamelFileName	指定输出文件名（相对于端点目录），用于在发送到端点时用于输出消息。如果这不存在且没有表达式，则生成的消息 ID 将用作文件名。
CamelFileNameProduced	所写入的输出文件的实际文件路径（路径 + 名称）。此标头由 Camel 设置，其目的是最终用户提供所写入文件的名称。
CamelFileIndex	当前索引超出此批处理中消耗的文件总数。
CamelFileSize	此批处理中消耗的文件总数。
CamelFileHost	远程主机名。
CamelFileLocalWorkPath	如果使用本地工作目录的路径。

此外，FTP/FTPS 使用者和制作者将使用以下标头增强 Camel 消息

标头	描述
CamelFtpReplyCode	Camel 2.11.1： FTP 客户端回复代码（类型是一个整数）
CamelFtpReplyString	Camel 2.11.1 ： FTP 客户端回复字符串

111.10. 关于超时

两组库（请参阅 top）具有不同的 API 来设置超时。您可以使用 connectTimeout 选项来设置 millis 中的超时来建立网络连接。还可以在 FTP/FTPS 上设置单独的 soTimeout，对应于使用 ftpClient.soTimeout。请注意，SFTP 将自动使用 connectTimeout 作为其 soTimeout。timeout 选项只适用于 FTP/FTSP 作为数据超时，它与 ftpClient.dataTimeout 值对应。所有超时值都是 millis。

111.11. 使用本地工作目录

Camel 支持从远程 FTP 服务器消耗，并将文件直接下载到本地工作目录中。这可避免将整个远程文件内容读取到内存中，因为它使用 FileOutputStream 直接传输到本地文件。

在下载文件时，Camel 将存储到名称与远程文件相同的本地文件，但 .inprogress 作为扩展名。之后，该文件被重命名为删除 .inprogress 后缀。最后，当 Exchange 完成本地文件后，将删除。

因此，如果您要从远程 FTP 服务器下载文件并将其存储为文件，那么您需要路由到文件端点，例如：

```
from("ftp://someone@someserver.com?
password=secret&localWorkDirectory=/tmp").to("file://inbox");
```

提示

以上路由效率更高，因为它可避免将整个文件内容读入内存中。它将直接下载远程文件到本地文件流。然后，java.io.File 处理被用作 Exchange body。文件生成者利用此事实，可以直接处理工作文件 java.io.File 处理，并对目标文件名执行 java.io.File.rename。由于 Camel 知道它是一个本地工作文件，它可以优化并使用重命名而不是文件副本，因为工作文件应该被任何删除。

111.12. 逐步更改目录

在消耗文件（例如下载）或生成文件（例如上传）时，Camel FTP 可在遍历目录方面以两种模式运行。

- stepwise
- 不分步

您可能希望根据您的情况和安全问题选择其中之一。有些 Camel 最终用户只能在文件使用步骤时下载文件，而其他用户只能下载文件（如果文件则不能进行下载）。您至少可以选择选择（从 Camel 2.6 开始）。

在 Camel 2.0 - 2.5 中，只有一个模式，它：

- *2.5 之前没有步骤*

- *2.5 stepwise*

现在，从 Camel 2.6 开始，您可以使用一个选项来控制行为。

请注意，在大多数情况下，目录更改操作才会在大多数情况下才能工作，而当主目录报告为 "/" 时，才起作用。

其中两个之间的差别最好地展示了一个示例。假设我们在远程 FTP 服务器上有以下目录结构，我们需要遍历和下载文件：

```
/
/one
/one/two
/one/two/sub-a
/one/two/sub-b
```

我们在每个子(a.txt)和子b (b.txt)文件夹中都有一个文件。

111.12.1. 使用 stepwise=true （默认模式）

```
TYPE A
200 Type set to A
PWD
257 "/" is current directory.
CWD one
250 CWD successful. "/one" is current directory.
CWD two
250 CWD successful. "/one/two" is current directory.
SYST
215 UNIX emulated by FileZilla
PORT 127,0,0,1,17,94
200 Port command successful
LIST
150 Opening data channel for directory list.
226 Transfer OK
CWD sub-a
250 CWD successful. "/one/two/sub-a" is current directory.
PORT 127,0,0,1,17,95
200 Port command successful
LIST
150 Opening data channel for directory list.
```

```
226 Transfer OK
CDUP
200 CDUP successful. "/one/two" is current directory.
CWD sub-b
250 CWD successful. "/one/two/sub-b" is current directory.
PORT 127,0,0,1,17,96
200 Port command successful
LIST
150 Opening data channel for directory list.
226 Transfer OK
CDUP
200 CDUP successful. "/one/two" is current directory.
CWD /
250 CWD successful. "/" is current directory.
PWD
257 "/" is current directory.
CWD one
250 CWD successful. "/one" is current directory.
CWD two
250 CWD successful. "/one/two" is current directory.
PORT 127,0,0,1,17,97
200 Port command successful
RETR foo.txt
150 Opening data channel for file transfer.
226 Transfer OK
CWD /
250 CWD successful. "/" is current directory.
PWD
257 "/" is current directory.
CWD one
250 CWD successful. "/one" is current directory.
CWD two
250 CWD successful. "/one/two" is current directory.
CWD sub-a
250 CWD successful. "/one/two/sub-a" is current directory.
PORT 127,0,0,1,17,98
200 Port command successful
RETR a.txt
150 Opening data channel for file transfer.
226 Transfer OK
CWD /
250 CWD successful. "/" is current directory.
PWD
257 "/" is current directory.
CWD one
250 CWD successful. "/one" is current directory.
CWD two
250 CWD successful. "/one/two" is current directory.
CWD sub-b
250 CWD successful. "/one/two/sub-b" is current directory.
PORT 127,0,0,1,17,99
200 Port command successful
RETR b.txt
150 Opening data channel for file transfer.
226 Transfer OK
CWD /
```

```
250 CWD successful. "/" is current directory.
QUIT
221 Goodbye
disconnected.
```

正如您在启用 stepwise 时看到，它将使用 CD xxx 遍历目录结构。

111.12.2. 使用 stepwise=false

```
230 Logged on
TYPE A
200 Type set to A
SYST
215 UNIX emulated by FileZilla
PORT 127,0,0,1,4,122
200 Port command successful
LIST one/two
150 Opening data channel for directory list
226 Transfer OK
PORT 127,0,0,1,4,123
200 Port command successful
LIST one/two/sub-a
150 Opening data channel for directory list
226 Transfer OK
PORT 127,0,0,1,4,124
200 Port command successful
LIST one/two/sub-b
150 Opening data channel for directory list
226 Transfer OK
PORT 127,0,0,1,4,125
200 Port command successful
RETR one/two/foo.txt
150 Opening data channel for file transfer.
226 Transfer OK
PORT 127,0,0,1,4,126
200 Port command successful
RETR one/two/sub-a/a.txt
150 Opening data channel for file transfer.
226 Transfer OK
PORT 127,0,0,1,4,127
200 Port command successful
RETR one/two/sub-b/b.txt
150 Opening data channel for file transfer.
226 Transfer OK
QUIT
221 Goodbye
disconnected.
```

正如您在不使用 stepwise 时所见，根本没有调用 CD 操作。

111.13. SAMPLES

在以下示例中，我们将 Camel 设置为每小时(60 分钟)一次从 FTP 服务器下载所有报告，作为 BINARY 内容，并将它作为文件存储在本地文件系统中。

和使用 Spring DSL 的路由：

```
<route>
  <from uri="ftp://scott@localhost/public/reports?
password=tiger&amp;binary=true&amp;delay=60000"/>
  <to uri="file://target/test-reports"/>
</route>
```

111.13.1. 使用远程 FTPS 服务器（简化 SSL）和客户端身份验证

```
from("ftps://admin@localhost:2222/public/camel?
password=admin&securityProtocol=SSL&isImplicit=true
    &ftpClient.keyStore.file=./src/test/resources/server.jks
    &ftpClient.keyStore.password=password&ftpClient.keyStore.keyPassword=password")
  .to("bean:foo");
```

111.13.2. 消耗远程 FTPS 服务器（临时 TLS）和自定义信任存储配置

```
from("ftps://admin@localhost:2222/public/camel?
password=admin&ftpClient.trustStore.file=./src/test/resources/server.jks&ftpClient.trustStore.
password=password")
  .to("bean:foo");
```

111.14. 使用 ORG.APACHE.CAMEL.COMPONENT.FILE.GENERICFILEFILTER进行过滤

Camel 支持可插拔过滤策略。这可确保它在 Java 中的 org.apache.camel.component.file.GenericFileFilter 中使用构建。然后，您可以使用这样的过滤器配置端点，以在处理前跳过某些过滤器。

在示例中，我们构建了自己的过滤器，该过滤器仅接受文件名中以报告开头的文件。

然后，我们可以使用 filter 属性配置我们的路由，以引用我们在 spring XML 文件中定义的过滤（使用 # 表示法）：

```
<!-- define our sorter as a plain spring bean -->
<bean id="myFilter" class="com.mycompany.MyFileFilter"/>
```

```
<route>
  <from uri="ftp://someuser@someftpserver.com?password=secret&amp;filter=#myFilter"/>
  <to uri="bean:processInbox"/>
</route>
```

111.15. 使用 ANT 路径匹配器进行过滤

ANT 路径匹配器是一个过滤器，它在 camel-spring jar 中提供出厂。因此，如果您使用 Maven，则需要依赖 camel-spring。
原因在于，我们利用 Spring 的 AntPathMatcher 进行实际匹配。

文件路径与以下规则匹配：

- ? 匹配一个字符
- * 匹配零个或多个字符
- ** 匹配路径中的零个或多个目录

以下示例演示了如何使用它：

111.16. 在 SFTP 中使用代理

要使用 HTTP 代理连接到远程主机，您可以使用以下方法配置路由：

```
<!-- define our sorter as a plain spring bean -->
<bean id="proxy" class="com.jcraft.jsch.ProxyHTTP">
  <constructor-arg value="localhost"/>
  <constructor-arg value="7777"/>
</bean>

<route>
  <from uri="sftp://localhost:9999/root?username=admin&password=admin&proxy=#proxy"/>
  <to uri="bean:processFile"/>
</route>
```

如果需要，您还可以将用户名和密码分配给代理。请参阅有关 com.jcraft.jsch.Proxy 的文档，以发现所有选项。

111.17. 设置首选 SFTP 验证方法

如果要明确指定 sftp 组件应使用的身份验证方法列表，请使用 preferredAuthentications 选项。例如，当没有公钥可用时，您希望 Camel 试图使用私有/公共 SSH 密钥进行身份验证，并在没有公钥可用时回退到用户/密码身份验证，请使用以下路由配置：

```
from("sftp://localhost:9999/root?
username=admin&password=admin&preferredAuthentications=publickey,password").
  to("bean:processFile");
```

111.18. 使用固定名称消耗单个文件

当您想下载单个文件并知道文件名时，您可以使用 fileName=myFileName.txt 告知 Camel 要下载的文件名。默认情况下，使用者仍会执行 FTP LIST 命令，以便进行目录列表，然后根据 fileName 选项过滤这些文件。虽然在这种情况下，可能需要通过设置 useList=false 来关闭目录列表。例如，用于登录 FTP 服务器的用户帐户可能没有执行 FTP LIST 命令的权限。因此，您可以使用 useList=false 关闭，然后提供文件的固定名称来下载 fileName=myFileName.txt，然后 FTP 使用者仍然可以下载该文件。如果出于某种原因的文件不存在，则 Camel 默认会抛出异常，您可以通过设置 ignoreFileNotFoundOrPermissionError=true 来忽略这个情况。

例如，要有一个选择单个文件的 Camel 路由，并在使用后将其删除。

```
from("ftp://admin@localhost:21/nolist/?
password=admin&stepwise=false&useList=false&ignoreFileNotFoundOrPermissionError=true
&fileName=report.txt&delete=true")
  .to("activemq:queue:report");
```

请注意，我们使用了上述所有选项。

您也可以将其与 ConsumerTemplate 搭配使用。例如，要下载单个文件（如果存在）并作为 String 类型获取文件内容：

```
String data = template.retrieveBodyNoWait("ftp://admin@localhost:21/nolist/?
password=admin&stepwise=false&useList=false&ignoreFileNotFoundOrPermissionError=true
&fileName=report.txt&delete=true", String.class);
```

111.19. 调试日志记录

此组件具有日志级别 TRACE，如果您遇到问题，这非常有用。

111.20. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*
- *File2*

第 112 章 FTPS 组件

从 Camel 版本 2.2 开始提供

这个组件可让您通过 FTP 和 SFTP 协议访问远程文件系统。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-ftp</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

如需更多信息，您可以查看 FTP 组件

112.1. URI 选项

以下选项专用于 FTPS 组件。

FTPS 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

FTPS 端点使用 URI 语法进行配置：

```
ftps:host:port/directoryName
```

使用以下路径和查询参数：

112.1.1. 路径参数(3 参数)：

Name	描述	默认值	类型
主机	需要的 FTP 服务器的主机名		字符串
port	FTP 服务器的端口		int
directoryName	起始目录		字符串

112.1.2. 查询参数(122 参数)：

Name	描述	默认值	类型
binary (common)	指定文件传输模式 BINARY 或 ASCII。默认为 ASCII (false)。	false	布尔值
charset (common)	此选项用于指定文件的编码。您可以在消费者上使用此选项，指定文件的编码，允许 Camel 了解在访问文件内容时应加载文件内容。在编写文件时，您可以使用此选项指定写入该文件的 charset。请记住，在编写文件 Camel 时，可能需要将消息内容读取到内存中，以便能将数据转换为配置的 charset，因此如果您有大型消息，请不要使用它。		字符串
disconnect (common)	使用后是否从远程 FTP 服务器断开连接。断开连接将仅断开当前与 FTP 服务器的连接。如果您有要停止的消费者，则需要停止消费者/路由。	false	布尔值
doneFileName (common)	生产者：如果提供，则 Camel 将在写入原始文件时写入第 2 个 done 文件。完成的文件将为空。这个选项配置要使用的文件名。您可以指定一个固定名称。或者，您可以使用动态占位符。完成的文件始终将写入与原始文件相同的文件夹中。consumer：如果提供，Camel 仅在文件存在时使用文件。这个选项配置要使用的文件名。您可以指定一个固定名称。或者您可以使用动态占位符。已完成的文件始终预期位于与原始文件相同的文件夹中。仅支持 $file.name 和 $file.name.noext 作为动态占位符。		字符串

Name	描述	默认值	类型
filename (common)	使用 File Language 等表达式来动态设置文件名。对于消费者，它将用作文件名过滤器。对于生成者，它用于评估要写入的文件名。如果设置了表达式，它将优先于 CamelFileName 标头。（注：标题本身也可以是表达式）。表达式选项支持 String 和 Expression 类型。如果表达式是 String 类型，则始终使用 File 语言来评估它。如果表达式是 Expression 类型，则使用指定的 Expression 类型 - 例如，允许您使用 OGNL 表达式。对于消费者，您可以使用它过滤文件名，因此您可以使用 File Language 语法使用当前的文件：mydata-$date:now:yyyyMMdd.txt。生产者支持 CamelOverruleFileName 标头，它优先于任何现有的 CamelFileName 标头；CamelOverruleFileName 是一个只使用一次的标头，它可让您更轻松地临时存储 CamelFileName，且之后必须恢复它。		字符串
passiveMode (common)	设置被动模式连接。默认为活跃的模式连接。	false	布尔值
分隔符 (common)	设置要使用的路径分隔符。Unix = 使用 unix 样式路径分隔符 Windows = 使用窗口样式路径分隔符 Auto =（默认）在文件名中使用现有路径分隔符	UNIX	PathSeparator
transferLoggingInterval Seconds (common)	配置在记录上传和下载操作的进度时使用的时间间隔（以秒为单位）。这在操作需要更长的时间时用于记录进度。	5	int
transferLoggingLevel (common)	配置在记录上传和下载操作的进度时使用的日志记录级别。	DEBUG	LoggingLevel
transferLoggingVerbose (common)	配置上传和下载操作进度的执行详细(ine grained)日志记录。	false	布尔值
fastExistsCheck (common)	如果将此选项设置为 true，camel-ftp 将直接使用列表文件来检查该文件是否存在。由于某些 FTP 服务器可能不支持直接列出文件，如果选项为 false，camel-ftp 将使用旧方式列出该目录并检查该文件是否存在。这个选项还影响 readLock=changed，以控制它是否执行快速检查来更新文件信息。如果 FTP 服务器包含大量文件，则可以使用它来加快进程。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
delete (consumer)	如果为 true，则在成功处理后会删除该文件。	false	布尔值

Name	描述	默认值	类型
moveFailed (consumer)	根据简单语言设置移动失败表达式。例如，要将文件移到 .error 子目录中，请使用： .error。注意：将文件移到故障位置 Camel 时，将处理错误，并且不会再次获取该文件。		字符串
noop (consumer)	如果为 true，则文件不会以任何方式移动或删除。这个选项对只读数据或者 ETL 类型要求很好。如果 noop=true，Camel 也设置 idempotent=true，以避免再次消耗相同的文件。	false	布尔值
preMove (consumer)	表达式（如文件语言）用于在处理之前动态设置文件名。例如，要将 in-progress 文件移到 order 目录中，将此值设置为 order。		字符串
preSort (consumer)	启用预先排序后，消费者将在轮询期间对文件和目录名称进行排序，该名称是从文件系统检索的。如果需要按排序顺序对文件进行操作，您可能需要执行此操作。指示器在消费者开始过滤前执行，并接受 Camel 处理的文件。这个选项是 default=false 表示禁用。	false	布尔值
递归 (consumer)	如果某个目录，将查找所有子目录中的文件。	false	布尔值
resumeDownload (consumer)	配置是否启用了恢复下载。这必须被 FTP 服务器支持（几乎所有 FTP 服务器都支持它）。此外，必须配置 localWorkDirectory 选项，以便下载的文件存储在本地目录中，且必须启用选项二进制文件，以支持恢复下载。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
streamDownload (consumer)	设置在使用本地工作目录时要使用的下载方法。如果设置为 true，则远程文件会在读取时流到路由。当设置为 false 时，远程文件会在发送到路由之前加载到内存中。	false	布尔值
directoryMustExist (consumer)	与 startingDirectoryMustExist 类似，但这在轮询递归子目录期间应用。	false	布尔值
download (consumer)	FTP 使用者是否应下载 文件。如果此选项设为 false，则消息正文将为空，但消费者仍会触发具有文件详情的 Camel Exchange，如文件名、文件大小等。只是不会下载该文件。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler

Name	描述	默认值	类型
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
handleDirectoryParserAbsoluteResult (consumer)	如果目录解析器生成了绝对路径，则允许使用者如何处理路径中子文件夹和文件的方式。这样做的原因是，有些 FTP 服务器可能会返回具有绝对路径的文件名，如果是，FTP 组件需要处理这个路径。	false	布尔值
ignoreFileNotFoundOrPermissionError (consumer)	是否忽略 when （尝试列出目录中的文件或下载文件时），但不存在，还是因为权限错误。默认情况下，当目录或文件不存在或权限不足时，会抛出异常。将这个选项设置为 true 可忽略它。	false	布尔值
inProgressRepository (consumer)	可插拔式 in-progress 存储库 org.apache.camel.spi.IdempotentRepository。in-progress 存储库用于考虑被消耗的当前正在进行的文件。默认使用基于内存的存储库。		IdempotentRepository
localWorkDirectory (consumer)	在消耗时，可以使用本地工作目录直接将远程文件内容存储在本地文件中，以避免将内容加载到内存中。这很有用，如果您使用非常大的远程文件，因此可以节省内存。		字符串
onCompletionException Handler (consumer)	要使用自定义 org.apache.camel.spi.ExceptionHandler 来处理在完成过程中发生的任何抛出异常，消费者在其中执行提交或回滚。默认实施会将任何异常记录在 WARN 级别并忽略。		ExceptionHandler
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
processStrategy (consumer)	可插拔 org.apache.camel.component.file.GenericFileProcessStrategy 允许您实施自己的 readLock 选项或类似选项。也可以在使用文件之前满足特殊条件时使用，如存在特殊的就绪文件。如果设置了这个选项，则不会应用 readLock 选项。		GenericFileProcess s Strategy
receiveBufferSize (consumer)	仅由 FTPClient 使用的接收（下载）缓冲大小	32768	int

Name	描述	默认值	类型
startingDirectoryMustExist (consumer)	起始目录是否必须存在。请注意，autoCreate 选项是默认启用的，这意味着如果起始目录不存在，则通常会自动创建该目录。您可以禁用 autoCreate 并启用它，以确保起始目录必须存在。如果目录 不存在，将抛出异常。	false	布尔值
useList (consumer)	下载文件时是否允许使用 LIST 命令。默认为 true。在某些情况下，您可能想要下载特定的文件，且不允许使用 LIST 命令，因此您可以将此选项设置为 false。请注意，在使用此选项时，要下载的特定文件不包括诸如文件大小、时间戳、权限等元数据数据信息，因为这些信息只能在 LIST 命令使用时检索。	true	布尔值
fileExist (producer)	如果文件已存在具有相同名称的文件，该怎么办。override （默认值）替换现有文件。Append - 在现有文件中添加内容。fail - 抛出 GenericFileOperationException，表示已存在文件。ignore - 静默忽略问题，且不会覆盖现有文件，但假设一切都正常。Move - 选项需要使用 moveExisting 选项也可以配置。选项 eagerDeleteTargetFile 可用于控制在移动文件时执行什么操作，并且已存在文件，否则会导致移动操作失败。Move 选项将在编写目标文件之前移动任何现有文件。只有在使用 tempFileName 选项时，TryRename 才适用。这允许尝试将文件从临时名称重命名为实际名称，而无需执行任何存在的检查。在某些文件系统（特别是 FTP 服务器中）中，这个检查可能会更快。	override	GenericFileExist
flatten (producer)	flatten 用于扁平化任何前导路径的文件名路径，因此只用于文件名。这可让您以递归方式消耗到子目录中，但当您将文件写入另一个目录时，它们将写入单个目录。在生成者中将其设置为 true 强制执行 CamelFileName 标头中的任何文件名都会被剥离任何前导路径。	false	布尔值
jailStartingDirectory (producer)	用于存放（限制）将文件写入起始目录（和子目录）。这默认是启用的，不允许 Camel 将文件写入外部目录（从而更加安全）。您可以关闭此项，以允许将文件写入起始目录之外的目录，如父目录或 root 文件夹。	true	布尔值
moveExisting (producer)	当配置了 fileExist=Move 时，用于计算文件名的表达式（如 File Language）。要将文件移动到备份子目录中，请输入 backup。这个选项只支持以下 File Language 令牌：file:name, file:name.ext, file:name.noext, file:onlyname, file:onlyname.noext, file:ext, 和 file:parent。请注意，FTP 组件不支持 file:parent，因为 FTP 组件只能将任何现有文件移到基于当前 dir 的相对目录中。		字符串

Name	描述	默认值	类型
tempFileName (producer)	与 tempPrefix 选项相同，但对临时文件名的命名提供更精细的控制，因为它使用 File Language。		字符串
tempPrefix (producer)	此选项用于使用临时名称写入文件，然后在写入完成后将其重命名为实际名称。可用于识别正在写入的文件，并避免消费者（不使用专用读取锁定）读取进度文件。上传大型文件时，FTP 通常使用。		字符串
allowNullBody (producer)	用于指定在文件写入过程中是否允许 null 正文。如果设置为 true，则会创建一个空文件，当设为 false 时，并尝试向文件组件发送 null 正文，则 'Cannot null body to file.' 的 GenericFileWriteException 将会被抛出。如果将 fileExist 选项设置为 'Override'，则文件将被截断，如果设置为附加文件，则文件将保持不变。	false	布尔值
chmod (producer)	允许您在存储的文件上设置 chmod。例如 chmod=640。		字符串
disconnectOnBatchComplete (producer)	在 Batch upload 完成后是否断开与远程 FTP 服务器的连接。disconnectOnBatchComplete 只会断开当前与 FTP 服务器的连接。	false	布尔值
eagerDeleteTargetFile (producer)	是否强制删除任何现有目标文件。这个选项只适用于使用 fileExists=Override 和 tempFileName 选项。您可以使用它来禁用（将其设置为 false）在写入临时文件前删除目标文件。例如，您可以写入大文件，并且希望写入 temp 文件期间存在目标文件。这样可确保目标文件仅在最后一次删除之前，直到 temp 文件被重命名为目标文件名之前。这个选项还用于控制在启用了 fileExist=Move 时是否删除任何现有文件，并且存在现有的文件。如果此选项 copyAndDeleteOnRenameFails false，则会在移动操作前现有文件存在时抛出异常。	true	布尔值
keepLastModified (producer)	将保留源文件的最后修改的时间戳（若有）。将使用 Exchange.FILE_LAST_MODIFIED 标头来定位时间戳。此标头可以包含带有时间戳的 java.util.Date 或 long。如果存在时间戳，并且启用了 选项，它将在写入的文件上设置此时间戳。注：此选项仅适用于文件制作者。您不能将此选项与任何 ftp 生成者一起使用。	false	布尔值
moveExistingFileStrategy (producer)	策略(Custom Strategy)用于在配置 fileExist=Move 时使用，使用特殊命名令牌来移动文件。默认情况下，如果没有提供自定义策略，则使用实施		FileMoveExisting 策略

Name	描述	默认值	类型
sendNoop (producer)	在将文件上传到 FTP 服务器之前，是否将 noop 命令作为预写检查发送。这默认是启用的，因为连接的验证仍然有效，这允许静默重新连接才能上传该文件。但是，如果这会导致问题，您可以关闭这个选项。	true	布尔值
activePortRange (advanced)	以活跃模式设置客户端侧端口范围。语法为：minPort-maxPort Both 端口号，例如 10000-19999，包含所有 1xxxx 端口。		字符串
auto create (advanced)	在文件的路径名称中自动创建缺少的目录。对于文件消费者，这意味着创建起始目录。对于生成者文件，这意味着文件应当要写入的目录。	true	布尔值
bufferSize (advanced)	写入缓冲区大小（以字节为单位）。	131072	int
connectTimeout (advanced)	设置连接超时，以等待 FTPClient 和 JSCH 使用连接	10000	int
ftpClient (advanced)	使用自定义 FTPClient 实例		FTPClient
ftpClientConfig (advanced)	要使用 FTPClientConfig 的自定义实例来配置端点应使用的 FTP 客户端。		FTPClientConfig
ftpClientConfigParameters (advanced)	FtpComponent 用于为 FTPClientConfig 提供其他参数		Map
ftpClientParameters (advanced)	FtpComponent 用于为 FTPClient 提供其他参数		Map
maximumReconnectAttempts (advanced)	指定在尝试连接到远程 FTP 服务器时执行的最大重新连接尝试 Camel。使用 0 来禁用此行为。		int
reconnectDelay (advanced)	millis Camel 中的延迟将在执行重新连接尝试前等待。		long
siteCommand (advanced)	设置在成功登录后要执行的可选 site 命令。可以使用新行字符来分隔多个站点命令。		字符串
soTimeout (advanced)	仅为 FTPClient 设置 so 超时	300000	int

Name	描述	默认值	类型
stepwise (advanced)	设置我们是否应在下载文件时遍历文件结构的同时，还是将文件上传到目录时逐步更改目录。例如，如果您因为安全原因无法在 FTP 服务器上更改目录时禁用此功能。	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
throwExceptionOnConnect Failed (advanced)	如果连接失败（忽略）时，应该抛出异常，默认异常不会被抛出，并记录 WARN。您可以使用它来启用异常，并处理 org.apache.camel.spi.PollingConsumerPollStrategy rollback 方法所引发的异常。	false	布尔值
超时 (advanced)	设置用于仅由 FTPClient 使用的数据超时	30000	int
antExclude (filter)	Ant 风格过滤器排除。如果使用 antInclude 和 antExclude，则 antExclude 优先于 antInclude。可以使用逗号分隔的格式指定多个排除项。		字符串
antFilterCaseSensitive (filter)	在t 过滤器上设置问题单敏感标志	true	布尔值
antInclude (filter)	Ant 风格过滤器包括：可以使用逗号分隔的格式指定多个包含。		字符串
eagerMaxMessagesPerPoll (filter)	允许控制 maxMessagesPerPoll 的限制是 eager 还是 not。如果 eager，则限制是在扫描文件时。其中 false 会扫描所有文件，然后执行排序。将此选项设置为 false 可首先对所有文件进行排序，然后限制轮询。请注意，这需要更高的内存用量，因为所有文件详细信息都在内存中执行排序。	true	布尔值
exclude (filter)	用于排除文件，如果文件名与 regex 模式匹配（匹配是 in-senstive）。请注意，如果您使用加号等符号，如果将其配置为端点 uri，则需要使用 RAW （）语法进行配置。有关配置端点 uris 时，请参阅更多详情		字符串
Filter (filter)	可插拔过滤器为 org.apache.camel.component.file.GenericFileFilter 类。如果过滤在 accept （）方法中返回 false，则会跳过文件。		GenericFileFilter
filterDirectory (filter)	根据简单语言过滤目录。例如，要过滤当前日期，您可以使用简单的日期模式，如 $date:now:yyMMdd		字符串

Name	描述	默认值	类型
filterFile (filter)	根据简单语言过滤文件。例如，要过滤文件大小，您可以使用 $file:size 5000		字符串
idempotent (filter)	使用 Idempotent Consumer EIP 模式的选项，让 Camel 跳过已处理的文件。默认情况下，将使用基于内存的 LRUCache，其中包含 1000 个条目。如果 noop=true 然后会启用幂等，以避免再次消耗同样的文件。	false	布尔值
idempotentKey (filter)	使用自定义幂等密钥。默认情况下使用文件的绝对路径。您可以使用 File Language，例如使用文件名和文件大小，您可以执行以下操作：idempotentKey=$file:name-$file:size		字符串
idempotentRepository (filter)	一个可插拔式存储库 org.apache.camel.spi.IdempotentRepository，如果未指定 none，则默认使用 MemoryMessageIdRepository，并且幂等性为 true。		IdempotentRepository
include (filter)	用于包含文件，如果文件名匹配 regex 模式（匹配不区分大小写）。请注意，如果您使用加号等符号，如果将其配置为端点 uri，则需要使用 RAW （）语法进行配置。有关配置端点 uris 时，请参阅更多详情		字符串
maxDepth (filter)	递归处理目录时要遍历的最大深度。	2147483647	int
maxMessagesPerPoll (filter)	定义每个轮询要收集的最大消息。默认不设置最大值。可用于设置限制，例如 1000，以避免在启动有数千个文件的服务器时避免。设置一个 0 或 negative 值来禁用它。注意：如果使用这个选项，则 File 和 FTP 组件将在任何排序之前限制。例如，如果您有 100000 文件并使用 maxMessagesPerPoll=500，则只有第一个 500 文件会被获取，然后排序。您可以使用 eagerMaxMessagesPerPoll 选项，并将其设置为 false 以允许先扫描所有文件，然后随后排序。		int
minDepth (filter)	在递归处理目录时开始处理的最小深度。使用 minDepth=1 表示基础目录。使用 minDepth=2 表示第一个子目录。		int
Move (filter)	表达式（如简单语言）用于在处理后动态设置文件名。要将文件移动到 .done 子目录中，只需输入 .done。		字符串
exclusiveReadLockStrategy (lock)	可插拔 read-lock 作为 org.apache.camel.component.file.GenericFileExclusiveReadLockStrategy 实施。		GenericFileExclusiveReadLockStrategy

Name	描述	默认值	类型
readLock (lock)	供消费者使用，只有在文件上具有独占的 read-lock 时才轮询文件（例如，该文件没有正在进行中或被写入）。Camel 将等到授予文件锁定为止。这个选项提供了 build in strategy: none - No read lock is in use markerFile - Camel 创建一个标记文件 (fileName.camelLock)，然后保存锁定。这个选项不适用于 FTP 组件更改 - 更改方法是使用文件长度/修改时间戳来检测当前是否复制该文件。至少将使用 1 sec 确定此选项，因此此选项无法像其他进程一样快速使用文件，但可以更可靠，因为 JDK IO API 始终无法确定文件目前是否被其他进程使用。可以使用 readLockCheckInterval 选项来设置检查频率。 fileLock - 用于使用 java.nio.channels.FileLock。这个选项在 Windows 或 FTP 组件上不可用。当通过 mount/share 访问远程文件系统时，应该避免使用这种方法，除非文件系统支持分布式文件锁定。rename 是使用尝试重命名该文件的测试，如果我们可以获得专用只读锁。幂等的 - （仅限文件组件）幂等性是将 idempotentRepository 用作读锁。这允许使用支持集群读取锁定（如果幂等存储库实现支持集群）。idid-changed - （仅限文件组件）幂等更改，使用 idempotentRepository 并更改作为组合的 read-lock。这允许使用读取锁定，如果幂等存储库实施支持集群。jid-rename - （仅限文件组件） idempotent-rename 使用 idempotentRepository，并重命名作为组合的 read-lock。这允许使用支持集群的读取锁定（如果幂等存储库实现支持）。注意：各种读取锁定并不适用于在集群模式下工作，其中不同节点上的并发用户与共享文件系统上的相同文件竞争。markerFile 使用接近 atomic 操作来创建空的标记文件，但不保证在集群中工作。fileLock 可能会更好地工作，但文件系统需要支持分布式文件锁定，等等。如果幂等性存储库支持集群，如 Hazelcast 组件或 Infinispan，则使用幂等读取锁定支持集群。	none	字符串
readLockCheckInterval (lock)	读锁在 millis 中，如果读锁定支持，则间隔为 millis。这个间隔用于在尝试获取读锁定之间休眠。例如，在使用更改的读锁定时，您可以将更高的间隔周期设置为 cater 以慢速写入。如果生成者非常慢，则默认 1 sec. 可能太快。注意：对于 FTP，默认的 readLockCheckInterval 为 5000。readLockTimeout 值必须高于 readLockCheckInterval，但 thumb 规则是有一个超过 readLockCheckInterval 的 2 次或多次的超时。这是为了确保在达到超时前尝试获取锁进程允许随时读取的时间。	1000	long
readLockDeleteOrphanLock Files (lock)	如果 Camel 没有正确关闭（如 JVM 崩溃），则应在启动时读取带有标记文件的读取锁定文件（可能已留在文件系统上）读取锁定。如果将此选项设置为 false，则任何孤立锁定文件将导致 Camel 不尝试选择该文件，这可能是因为另一个节点同时从同一个共享目录读取文件。	true	布尔值

Name	描述	默认值	类型
readLockIdempotentRelease Async (lock)	延迟发布任务应该是同步还是异步的。请参阅 readLockIdempotentReleaseDelay 选项的详情。	false	布尔值
readLockIdempotentRelease AsyncPoolSize (lock)	使用异步发行任务时，调度线程池中的线程数量。在几乎所有用例中，使用默认值 1 个内核线程应足够足够，只有在更新幂等存储库缓慢或处理大量文件时才将其设置为更高的值。如果您使用共享线程池，通过配置 readLockIdempotentReleaseExecutorService 选项，这个选项不会被使用。请参阅 readLockIdempotentReleaseDelay 选项的详情。		int
readLockIdempotentRelease Delay (lock)	是否将发行任务延迟为 millis。当文件被视为具有共享幂等存储库的主动/主动集群场景时，这可用于延迟发行任务扩展窗口，以确保其他节点因为竞争条件而可能扫描和获取同一文件。通过扩展发行任务的 time-window 有助于防止这些情况。只有在将 readLockRemoveOnCommit 配置为 true 时才需要注意延迟。		int
readLockIdempotentRelease ExecutorService (lock)	使用自定义和共享线程池进行异步发行任务。请参阅 readLockIdempotentReleaseDelay 选项的详情。		ScheduledExecutor Service
readLockLoggingLevel (lock)	无法获取读取锁定时使用的日志记录级别。默认情况下，会记录 WARN。您可以更改此级别，例如将 OFF 更改为没有任何日志记录。这个选项只适用于 readLock 类型： changed, fileLock, idempotent, idempotent-changed, idempotent-rename, rename。	DEBUG	LoggingLevel
readLockMarkerFile (lock)	是否将标志文件与 changed、rename 或 exclusive read 锁定类型一起使用。默认情况下，还使用标记文件来保护获取相同文件的其他进程。通过将这个选项设置为 false 来关闭此行为。例如，如果您不希望将标记文件写入 Camel 应用程序的文件系统。	true	布尔值
readLockMinAge (lock)	这个选项只适用于 readLock=changed。它可指定文件在尝试获取读取锁定前必须经过的最短期限。例如，使用 readLockMinAge=300s 来要求文件最新 5 分钟。这可加快更改的读锁定速度，因为它将只尝试获取至少给定年龄的文件。	0	long
readLockMinLength (lock)	这个选项只适用于 readLock=changed。它允许您配置最小文件长度。默认情况下，Camel 预期文件包含数据，因此默认值为 1。您可以将这个选项设置为零，以允许消耗零长度文件。	1	long

Name	描述	默认值	类型
readLockRemoveOnCommit (lock)	这个选项只适用于 readLock=idempotent。它允许在处理文件成功并且执行提交时指定是否从幂等存储库中删除文件名条目。默认情况下，文件不会被删除，以确保不会发生任何竞争条件，因此另一个活动节点可能会尝试获取该文件。相反，幂等存储库可能支持驱除策略，在 X 分钟后用于驱除文件名条目 - 这样可保证竞争条件没有问题。请参阅 readLockIdempotentReleaseDelay 选项的详情。	false	布尔值
readLockRemoveOnRollback (lock)	这个选项只适用于 readLock=idempotent。它允许在处理文件失败时指定是否从幂等存储库中删除文件名条目，并发生回滚。如果此选项为 false，则文件名称条目将被确认（就像文件确实是提交一样）。	true	布尔值
readLockTimeout (lock)	read-lock 的 millis 中可选超时（如果 read-lock 支持）。如果无法授予 read-lock，并且触发超时，则 Camel 将跳过该文件。在下一次轮询 Camel 时，将再次尝试文件，此时可能会授予 read-lock。使用 0 或较低值来指示永久。目前，fileLock，改变并重命名支持超时。注意：对于 FTP，默认的 readLockTimeout 值为 20000 而不是 10000。readLockTimeout 值必须高于 readLockCheckInterval，但 thumb 规则是有一个超过 readLockCheckInterval 的 2 次或多次的超时。这是为了确保在达到超时前尝试获取锁进程允许随时读取的时间。	10000	long
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long

Name	描述	默认值	类型
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
Shuffle (sort)	影响文件列表（以随机顺序排序）	false	布尔值
排序(排序)	使用文件语言进行内置排序.支持嵌套排序，以便按文件名排序，并根据修改的日期作为第二组排序。		字符串
排序 （排序）	可插拔排序器作为 java.util.Comparator 类。		比较器
帐户 （安全）	用于登录的帐户		字符串
disableSecureDataChannel 默认值 (security)	使用这个选项在使用安全数据频道时禁用默认选项。这可让您完全控制应使用 execPbsz 和 execProt 设置的内容。默认值为 false	false	布尔值
execPbsz (security)	使用安全数据频道时，您可以设置 exec 保护缓冲区大小		Long
execProt (security)	exec 保护级别 PROT 命令。c - Clear S - 仅 Safe (SSL 协议) E - 保密(SSL 协议) P - Private		字符串
ftpClientKeyStore 参数 (security)	设置密钥存储参数		Map

Name	描述	默认值	类型
ftpClientTrustStore 参数 (security)	设置信任存储参数		Map
isImplicit (security)	设置安全模式(Implicit/Explicit). true - Implicit Mode / False - Explicit Mode	false	布尔值
密码 (security)	用于登录的密码		字符串
securityProtocol (security)	设置底层安全协议。	TLS	字符串
sslContextParameters (security)	获取 JSSE 配置，该配置覆盖 FtpsEndpoint SerialftpClientKeyStoreParameters、ftpClientTrustStoreParameters 和 FtpsConfiguration SerialgetSecurityProtocol （）中的任何设置。		SSLContextParameters
用户名 (security)	用于登录的用户名		字符串

112.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.ftps.enabled	启用 ftps 组件	true	布尔值
camel.component.ftps.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.ftps.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

第 113 章 GANGLIA 组件

从 Camel 版本 2.15 开始提供

提供向 Ganglia 监控系统发送值（消息正文）的机制。使用 gmetric4j 库。可与标准的 Ganglia 和 JMXetric 一起使用，通过单一平台监控来自操作系统、JVM 和业务流程的指标。

您应该有一个 Ganglia gmond 代理在 JVM 运行的机器上运行。gmond 向 Ganglia 基础架构发送心跳，camel-ganglia 目前无法发送 heartbeat 本身。

在大多数 Linux 系统上(Debian、Ubuntu、Ubuntu、Fedora 和 RHEL/CentOS)，您只能安装 Ganglia 代理软件包，并使用多播配置自动运行。如果您愿意，您可以将它配置为使用常规 UDP 单播。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-ganglia</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

113.1. URI 格式

```
ganglia:address:port[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

113.2. GANGLIA 组件和端点 URI 选项

Ganglia 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
配置（高级）	使用共享配置		GangliaConfiguration

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Ganglia 端点使用 URI 语法进行配置：

```
ganglia:host:port
```

使用以下路径和查询参数：

113.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
主机	Ganglia 服务器的主机名	239.2.11 .71	字符串
port	Ganglia 服务器的端口	8649	int

113.2.2. 查询参数(13 参数)：

Name	描述	默认值	类型
dmax (producer)	Ganglia 将清除指标值（如果其过期）前的最短时间（以秒为单位）。设置为 0，并且该值将无限期地保留在 Ganglia 中，直到 gmond 代理重启为止。	0	int
groupname (producer)	指标所属的组。	java	字符串
metricName (producer)	用于指标的名称。	metric	字符串
mode (producer)	使用 MULTICAST 或 UNICAST 发送 UDP 指标数据包	多播	UDPAddressingM ode
prefix (producer)	使用这个字符串和下划线作为指标名称的前缀。		字符串
slope (producer)	slope	两者都	GMetricSlope

Name	描述	默认值	类型
spoofHostname (producer)	欺骗信息 IP:hostname		字符串
tmax (producer)	值可被视为当前的最大时间（以秒为单位）。之后，Ganglia 认为值已过期。	60	int
TTL (producer)	如果使用多播，请设置数据包的 TTL	5	int
type (producer)	值的类型	字符串	GMetricType
units (producer)	任何计算指标的测量单位，如小部件、litres、bytes。不要包含前缀，如 k (kilo)或 m (milli)，其他工具可能会在以后扩展单元。该值应该不会被扩展。		字符串
wireFormat31x (producer)	使用 Ganglia 3.1.0 及更新版本的线格式。把它设置为 false 以使用 Ganglia 3.0.x 或更早版本。	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

113.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 16 个选项，如下所列。

Name	描述	默认值	类型
camel.component.ganglia.configuration.dmax	Ganglia 将清除指标值（如果其过期）前的最短时间（以秒为单位）。设置为 0，并且该值将无限期地保留在 Ganglia 中，直到 gmond 代理重启为止。	0	整数
camel.component.ganglia.configuration.group-name	指标所属的组。	java	字符串
camel.component.ganglia.configuration.host	Ganglia 服务器的主机名	239.2.11.71	字符串
camel.component.ganglia.configuration.metric-name	用于指标的名称。	metric	字符串

Name	描述	默认值	类型
camel.component.ganglia.configuration.mode	使用 MULTICAST 或 UNICAST 发送 UDP 指标数据包		GMetric$UDPAddressingMode
camel.component.ganglia.configuration.port	Ganglia 服务器的端口	8649	整数
camel.component.ganglia.configuration.prefix	使用这个字符串和下划线作为指标名称的前缀。		字符串
camel.component.ganglia.configuration.slope	slope		GMetricSlope
camel.component.ganglia.configuration.spoof-hostname	欺骗信息 IP:hostname		字符串
camel.component.ganglia.configuration.tmax	值可被视为当前的最大时间（以秒为单位）。之后，Ganglia 认为值已过期。	60	整数
camel.component.ganglia.configuration.ttl	如果使用多播，请设置数据包的 TTL	5	整数
camel.component.ganglia.configuration.type	值的类型		GMetricType
camel.component.ganglia.configuration.units	任何计算指标的测量单位，如小部件、litres、bytes。不要包含前缀，如 k (kilo)或 m (milli)，其他工具可能会在以后扩展单元。该值应该不会被扩展。		字符串
camel.component.ganglia.configuration.wire-format31x	使用 Ganglia 3.1.0 及更新版本的线格式。把它设置为 false 以使用 Ganglia 3.0.x 或更早版本。	true	布尔值
camel.component.ganglia.enabled	启用 ganglia 组件	true	布尔值
camel.component.ganglia.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

113.4. 消息正文

正文中的任何值（如字符串或数字类型）都发送到 Ganglia 系统。

113.5. 返回值/响应

Ganglia 使用单向 UDP 或多播发送指标。邮件正文没有响应或更改。

113.6. 例子

113.6.1. 发送 String 指标

消息正文将转换为 String，并作为指标值发送。与数字指标不同，字符串值无法图表，但 Ganglia 使它们可用于报告。每个 Ganglia 主机页面顶部的 os_version 字符串是一个 String 指标的示例。

```
from("direct:string.for.ganglia")
    .setHeader(GangliaConstants.METRIC_NAME, simple("my_string_metric"))
    .setHeader(GangliaConstants.METRIC_TYPE, GMetricType.STRING)
    .to("direct:ganglia.tx");

from("direct:ganglia.tx")
    .to("ganglia:239.2.11.71:8649?mode=MULTICAST&prefix=test");
```

113.6.2. 发送数字指标

```
from("direct:value.for.ganglia")
    .setHeader(GangliaConstants.METRIC_NAME, simple("widgets_in_stock"))
    .setHeader(GangliaConstants.METRIC_TYPE, GMetricType.UINT32)
    .setHeader(GangliaConstants.METRIC_UNITS, simple("widgets"))
    .to("direct:ganglia.tx");

from("direct:ganglia.tx")
    .to("ganglia:239.2.11.71:8649?mode=MULTICAST&prefix=test");
```

第 114 章 GEOCODER 组件

从 Camel 版本 2.12 开始提供

geocoder: 组件用于查找给定地址或反向查找的 geocodes (latitude 和 longitde)。组件使用 Java API 用于 Google Geocoder 库。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-geocoder</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

114.1. URI 格式

```
geocoder:address:name[?options]
geocoder:latlng:latitude,longitude[?options]
```

114.2. 选项

Geocoder 组件没有选项。

Geocoder 端点使用 URI 语法进行配置：

```
geocoder:address:latlng
```

使用以下路径和查询参数：

114.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
address	应该带有 address 前缀的 geo 地址：		字符串
latlng	应该使用 latlng 为 geo latitude 和 longitude 前缀：		字符串

114.2.2. 查询参数(13 参数)：

Name	描述	默认值	类型
headersOnly (producer)	是否只增强带有标头的 Exchange，并将正文保留原样。	false	布尔值
language (producer)	要使用的语言。	en	字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
proxyAuthDomain (proxy)	代理 NTML 身份验证的域		字符串
proxyAuthHost (proxy)	用于代理 NTML 身份验证的可选主机		字符串
proxyAuthMethod (proxy)	代理的身份验证方法，可以是 Basic、Digest 或 NTLM。		字符串
proxyAuthPassword (proxy)	用于代理身份验证的密码		字符串
proxyAuthUsername (proxy)	用于代理身份验证的用户名		字符串
proxyHost (proxy)	代理主机名		字符串
proxyPort (proxy)	代理端口号		整数
apiKey (security)	使用 google apiKey		字符串
clientId (security)	对这个客户端 ID 使用 google Premium		字符串
clientKey (security)	在这个客户端密钥中使用 google Premium		字符串

114.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.geocoder.enabled	启用 geocoder 组件	true	布尔值
camel.component.geocoder.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

114.4. 交换数据格式

Camel 将以 com.google.code.geocoder.model.GeocodeResponse 类型的形式提供正文。如果地址为 "current"，则响应是一个 String 类型，带有当前位置的 JSON 表示。

如果选项 headersOnly 设为 true，则消息正文将按原样保留，并且只有标头将添加到 Exchange。

114.5. 消息标头

标头	描述
CamelGeoCoderStatus	必需。来自 geocoder 库的状态代码。如果 status 是 **GeocoderStatus.OK**，则额外标头会增强
CamelGeoCoderAddress	格式的地址
CamelGeoCoderLat	位置的 latitude。
CamelGeoCoderLng	位置的长度。
CamelGeoCoderLatlng	位置的 latitude 和 longitude。用逗号分开。
CamelGeoCoderCity	城市长名称。
CamelGeoCoderRegionCode	区域代码。
CamelGeoCoderRegionName	区域名称。
CamelGeoCoderCountryLong	国家的长名称。
CamelGeoCoderCountryShort	国家/地区短名称。

标头	描述
CamelGeoCoderPostalCode	邮政编码。

请注意，并非所有标头都可能会根据使用的可用数据和模式（地址与 latlng）提供。

114.6. SAMPLES

在以下示例中，我们获得巴斯州立法学分和长长

```
from("direct:start")
  .to("geocoder:address:Paris, France")
```

如果您使用 CamelGeoCoderAddress 提供一个标头，然后覆盖端点配置，以便获取 Copenhagen 的位置，Denmark 我们可以发送一条带有标头的消息，如下所示：

```
template.sendBodyAndHeader("direct:start", "Hello", GeoCoderConstants.ADDRESS,
"Copenhagen, Denmark");
```

要获得 latitude 和 longitude 的地址，我们可以：

```
from("direct:start")
  .to("geocoder:latlng:40.714224,-73.961452")
  .log("Location ${header.CamelGeocoderAddress} is at lat/lng:
${header.CamelGeocoderLatlng} and in country ${header.CamelGeoCoderCountryShort}")
```

哪个将记录

```
Location 285 Bedford Avenue, Brooklyn, NY 11211, USA is at lat/lng:
40.71412890,-73.96140740 and in country US
```

要获得当前位置，您可以使用 "current" 作为地址，如下所示：

```
from("direct:start")
  .to("geocoder:address:current")
```

第 115 章 GIT 组件

从 Camel 版本 2.16 开始提供

git: 组件允许您处理通用 Git 存储库。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-git</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

URI 格式

```
git://localRepositoryPath[?options]
```

115.1. URI 选项

生产者允许在特定存储库上执行操作。
使用者允许在特定存储库上消耗提交、标签和分支。

Git 组件没有选项。

Git 端点使用 URI 语法进行配置：

```
git:localPath
```

使用以下路径和查询参数：

115.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
localPath	所需的 本地存储库路径		字符串

115.1.2. 查询参数(13 参数)：

Name	描述	默认值	类型
branchName (common)	要处理的分支名称		字符串
password (common)	远程存储库密码		字符串
remoteName (common)	特定操作中使用的远程存储库名称，如 pull		字符串
remotePath (common)	远程存储库路径		字符串
tagName (common)	要处理的标签名称		字符串
username (common)	远程存储库用户名		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
type (consumer)	消费者类型		GitType
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
allowEmpty (producer)	管理空 git 提交的标志	true	布尔值
operation (producer)	对存储库执行的操作		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

115.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.git.enabled	启用 git 组件	true	布尔值
camel.component.git.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

115.3. 消息标头

Name	默认值	类型	Context	描述
CamelGitOperation	**null**	字符串	制作者	如果未作为端点选项指定，则在存储库上执行的操作
CamelGitFilename	**null**	字符串	制作者	add 操作中的文件名
CamelGitCommitMessage	**null**	字符串	制作者	提交操作中相关的提交消息
CamelGitCommitUsername	**null**	字符串	制作者	提交操作中的提交用户名
CamelGitCommitEmail	**null**	字符串	制作者	提交操作中的提交电子邮件
CamelGitCommitId	**null**	字符串	制作者	提交 ID
CamelGitAllowEmpty	**null**	布尔值	制作者	管理空 git 提交的标志

115.4. 生成者示例

以下是制作者的示例路由，它将文件 test.java 添加到本地存储库，使用 master 分支上的特定消息提交该文件，然后将其推送到远程存储库。

```
from("direct:start")
        .setHeader(GitConstants.GIT_FILE_NAME, constant("test.java"))
        .to("git:///tmp/testRepo?operation=add")
        .setHeader(GitConstants.GIT_COMMIT_MESSAGE, constant("first commit"))
        .to("git:///tmp/testRepo?operation=commit")
        .to("git:///tmp/testRepo?
operation=push&remotePath=https://foo.com/test/test.git&username=xxx&password=xxx")
        .to("git:///tmp/testRepo?operation=createTag&tagName=myTag")
        .to("git:///tmp/testRepo?operation=pushTag&tagName=myTag&remoteName=origin")
```

115.5. 消费者示例

以下是消耗提交的消费者路由示例：

```
from("git:///tmp/testRepo?type=commit")
                    .to(....)
```

第 116 章 GITHUB 组件

从 Camel 版本 2.15 开始提供

GitHub 组件通过封装 egit-github 与 GitHub API 交互。它目前为新的拉取请求、拉取请求注释、标签和提交提供轮询。它还可以在拉取请求上生成注释，并完全关闭拉取请求。

此端点依赖于简单的轮询，而不是 Webhook。原因包括：

- 可靠性/状态的问题
- 我们正在轮询的有效负载类型通常不大（以及分页在 API 中提供）
- 需要支持在有些地方无法公开运行的应用程序，其中 Webhook 会失败

请注意，GitHub API 相当大。因此，此组件可轻松扩展，以提供额外的交互。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-github</artifactId>
  <version>${camel-version}</version>
</dependency>
```

116.1. URI 格式

```
github://endpoint[?options]
```

116.2. 强制选项：

请注意，它们可以直接通过端点配置。

GitHub 组件没有选项。

GitHub 端点使用 URI 语法进行配置：

```
github:type/branchName
```

使用以下路径和查询参数：

116.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
type	**需要执行** 哪些 git 操作		GitHubType
branchName	分支名称		字符串

116.2.2. 查询参数(12 参数)：

Name	描述	默认值	类型
oauthToken (common)	GitHub OAuth 令牌，除非提供用户名和密码		字符串
password (common)	GitHub 密码，除非提供了 oauthToken		字符串
repoName (common)	**所需的** GitHub 存储库名称		字符串
repoOwner (common)	**所需的** GitHub 存储库所有者（机构）		字符串
username (common)	GitHub 用户名，除非提供了 oauthToken		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值

Name	描述	默认值	类型
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
encoding (producer)	在获取 git commit 文件时使用给定的编码		字符串
state (producer)	设置 git commit status 状态		字符串
targetUrl (producer)	要设置 git commit status 目标 url		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

116.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.github.enabled	启用 github 组件	true	布尔值
camel.component.github.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

116.4. 消费者端点：

端点	Context	正文类型
pullRequest	轮询	org.eclipse.egit.github.core.PullRequest
pullRequestComment	轮询	org.eclipse.egit.github.core.Comment （针对常规拉取请求讨论）或 org.eclipse.egit.github.core.CommitComment （在拉取请求 diff 中注释）

端点	Context	正文类型
tag	轮询	org.eclipse.egit.github.core.RepositoryTag
commit	轮询	org.eclipse.egit.github.core.RepositoryCommit

116.5. 生产者端点：

端点	Body	消息标头
pullRequestComment	字符串（文本）	- GitHubPullRequest （整数）(REQUIRED): Pull request number。 - GitHubInResponseTo （整数）：如果响应拉取请求 diff 的另一个内联注释，则需要此项。如果保留，则假定拉取请求讨论的一个常规注释。
closePullRequest	none	- GitHubPullRequest （整数）(REQUIRED): Pull request number。
createIssue (From Camel 2.18)	字符串（签发正文文本）	- GitHubIssueTitle （字符串）(REQUIRED)：问题标题。

第 117 章 GZIP DATAFORMAT

从 Camel 版本 2.0 开始提供

GZip Data Format 是消息压缩和解压缩格式。它使用与 Zip DataFormat 时使用的相同保护算法，但提供了一些额外的标头。这个格式由常用的 gzip/gunzip 工具生成。使用 GZip 压缩的消息可以像在端点中使用 GZip 解压缩前使用 GZip 解压缩。当您处理基于 XML 和文本的有效负载时，压缩功能非常有用，或者在您之前使用 gzip 工具读取消息时。

117.1. 选项

GZip dataformat 支持 1 个选项，如下所列。

Name	默认值	Java 类型	描述
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

117.2. MARSHAL

在本例中，我们将常规文本/XML 有效负载分成压缩有效负载使用 gzip 压缩格式，并将其发送到名为 MY_QUEUE 的 ActiveMQ 队列。

```
from("direct:start").marshal().gzip().to("activemq:queue:MY_QUEUE");
```

117.3. UNMARSHAL

在本例中，我们从名为 MY_QUEUE 的 ActiveMQ 队列到其原始格式，并将它转发到 UnGZippedMessageProcessor。

```
from("activemq:queue:MY_QUEUE").unmarshal().gzip().process(new
UnGZippedMessageProcessor());
```

117.4. 依赖项

此数据格式以 camel-core 提供，因此不需要额外的依赖项。

第 118 章 GOOGLE BIGQUERY 组件

从 Camel 版本 2.20 开始提供

118.1. 组件描述

Google Bigquery 组件通过 Google Client Services API 访问 Cloud BigQuery 基础架构。

当前实现不使用 gRPC。

当前实施不支持查询 BigQuery，即仅生成者。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-google-bigquery</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

118.2. 身份验证配置

Google BigQuery 组件身份验证用于 GCP 服务帐户。如需更多信息，请参阅 Google Cloud Platform Auth 指南

Google 安全凭证可以通过以下两个选项之一显式设置：

- 服务帐户电子邮件和服务帐户密钥(PEM 格式)
- GCP 凭证文件位置

如果两者都设置了，则服务帐户电子邮件/密钥将具有优先权。

或隐式，连接工厂将回退到 应用程序默认凭证。

俄克斯！默认凭据文件的位置可以通过 GOOGLE_APPLICATION_CREDENTIALS 环境变量进行配置。

Service Account Email 和 Service Account Key 可以在 GCP JSON 凭证文件中分别作为 client_email 和 private_key 找到。

118.3. URI 格式

```
google-bigquery://project-id:datasetId[:tableId]?[options]
```

118.4. 选项

Google BigQuery 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
projectId (producer)	Google Cloud Project Id		字符串
datasetId (producer)	bigquery Dataset Id		字符串
ConnectionFactory (producer)	ConnectionFactory 以获取与 Bigquery 服务的连接。如果非提供默认，则使用默认值		GoogleBigQueryConnectionFactory
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Google BigQuery 端点使用 URI 语法进行配置：

```
google-bigquery:projectId:datasetId:tableName
```

使用以下路径和查询参数：

118.4.1. 路径参数(3 参数)：

Name	描述	默认值	类型
projectId	**所需的** Google Cloud Project Id		字符串
datasetId	**所需的** BigQuery Dataset Id		字符串
tableId	bigquery 表 ID		字符串

118.4.2. 查询参数(3 参数)：

Name	描述	默认值	类型
ConnectionFactory (producer)	ConnectionFactory 以获取与 Bigquery 服务的连接。如果未提供默认值，则将使用默认值。		GoogleBigQueryConnectionFactory
useAsInsertId (producer)	用作插入 id 的字段名称		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

118.5. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.google-bigquery.dataset-id	bigquery Dataset Id		字符串
camel.component.google-bigquery.enabled	是否启用 google-bigquery 组件的自动配置。这默认是启用的。		布尔值
camel.component.google-bigquery.project-id	Google Cloud Project Id		字符串

Name	描述	默认值	类型
camel.component.google-bigquery.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

118.6. 消息标头

Name	类型	描述
CamelGoogleBigQuery.TableSuffix	**字符串**	插入数据时使用的表后缀
CamelGoogleBigQuery.InsertId	**字符串**	inserting data 时使用的 InsertId
CamelGoogleBigQuery.PartitionDecorator	**字符串**	分区 decorator 表示在插入数据时要使用的分区
CamelGoogleBigQuery.TableId	**字符串**	表 id，将在哪里提交数据。如果指定，将覆盖端点配置

118.7. 制作者端点

生产者端点可以接受并传送到 BigQuery 个人和分组的交换。分组的交换设置了 Exchange.GROUPED_EXCHANGE 属性。

Goole BigQuery 生成者将在单一 api 调用中发送分组的交换，除非指定了不同的表后缀或分区 decorator，否则它将中断它，以确保数据使用正确的后缀或分区 decorator 编写。

Google BigQuery 端点需要有效负载可以是映射或映射列表。包含映射的有效负载将插入一行，以及一个有效负载，其中包含映射列表的映射列表，将为列表中的每个条目插入一行。

118.8. 模板表

参考资料：https://cloud.google.com/bigquery/streaming-data-into-bigquery#template-tables

templated 表可以使用 GoogleBigQueryConstants.TABLE_SUFFIX 标头来指定。

例如，以下路由将创建表并每天插入记录分片：

```
from("direct:start")
.header(GoogleBigQueryConstants.TABLE_SUFFIX, "_${date:now:yyyyMMdd}")
.to("google-bigquery:sampleDataset:sampleTable")
```

请注意，建议在这个用例中使用分区。

118.9. 分区

参考资料：https://cloud.google.com/bigquery/docs/creating-partitioned-tables

创建表时指定分区，如果设置数据将自动分区到单独的表中。在插入数据时，可以通过在交换上设置 GoogleBigQueryConstants.PARTITION_DECORATOR 标头来指定特定分区。

118.10. 确保数据一致性

参考资料：https://cloud.google.com/bigquery/streaming-data-into-bigquery#dataconsistency

可以在带有标头 GoogleBigQueryConstants.INSERT_ID 或指定查询参数 useAsInsertId 的交换上设置插入 id。当有效负载是一个列表 - 如果有效负载是 GoogleBigQueryConstants.INSERT_ID 将会被忽略时，则无法使用插入插入的插入标头时，无法使用插入的 id。在这种情况下，使用查询参数 useAsInsertId。

第 119 章 GOOGLE CALENDAR 组件

从 Camel 版本 2.15 开始提供

Google Calendar 组件可以通过 Google Calendar Web API 访问 Google Calendar 组件。

Google Calendar 使用 OAuth 2.0 协议 来验证 Google 帐户并授权对用户数据的访问权限。在可以使用此组件之前，您需要 创建帐户并生成 OAuth 凭据。凭证由 clientId、clientSecret 和 refreshToken 组成。用于生成较长的 refreshToken 的方便资源是 OAuth 播放。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-google-calendar</artifactId>
    <version>2.15.0</version>
</dependency>
```

119.1. 1.GOOGLE CALENDAR 选项

Google Calendar 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	使用共享配置		GoogleCalendar 配置
clientFactory (advanced)	使用 GoogleCalendarClientFactory 作为创建客户端的工厂。默认情况下将使用 BatchGoogleCalendarClientFactory		GoogleCalendarClient Factory
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Google Calendar 端点使用 URI 语法进行配置：

```
google-calendar:apiName/methodName
```

使用以下路径和查询参数：

119.1.1. 路径参数(2 参数)：

Name	描述	默认值	类型
apiName	**需要** 执行什么操作		GoogleCalendarApiName
methodName	**必需的** 所选操作使用哪些子操作		字符串

119.1.2. 查询参数(14 参数)：

Name	描述	默认值	类型
accessToken (common)	OAuth 2 访问令牌。这通常在一小时后过期，因此建议使用 refreshToken 进行长期使用。		字符串
applicationName (common)	Google 日历应用程序名称。示例为 camel-google-calendar/1.0		字符串
clientId (common)	日历应用程序的客户端 ID		字符串
clientSecret (common)	日历应用程序的客户端 secret		字符串
emailAddress (common)	Google Service Account 的 emailAddress。		字符串
inBody (common)	设置要在交换 In Body 中传递的参数名称		字符串
p12FileName (common)	包含用于 Google 服务帐户的私钥的 p12 文件的名称。		字符串
refreshToken (common)	OAuth 2 刷新令牌。使用这个，Google Calendar 组件可以在当前过期时获取新的 accessToken - 如果应用程序长期到期，则需要获得新的 accessToken。		字符串
scopes (common)	指定您希望日历应用程序具有用户帐户的权限级别。您可以使用逗号分隔多个范围。如需更多信息，请参阅 https://developers.google.com/google-apps/calendar/auth。	https://www.googleapis.com/auth/calendar	字符串

Name	描述	默认值	类型
user (common)	应用程序试图在服务帐户流中模拟的用户的电子邮件地址		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

119.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 14 个选项，如下所列。

Name	描述	默认值	类型
camel.component.google-calendar.client-factory	使用 GoogleCalendarClientFactory 作为创建客户端的工厂。默认情况下，将使用 BatchGoogleCalendarClientFactory。选项是 org.apache.camel.component.google.calendar.GoogleCalendarClientFactory 类型。		字符串
camel.component.google-calendar.configuration.access-token	OAuth 2 访问令牌。这通常在一小时后过期，因此建议使用 refreshToken 进行长期使用。		字符串
camel.component.google-calendar.configuration.api-name	要执行的操作类型		GoogleCalendarApiName

Name	描述	默认值	类型
camel.component.google-calendar.configuration.application-name	Google 日历应用程序名称。示例为 camel-google-calendar/1.0		字符串
camel.component.google-calendar.configuration.client-id	日历应用程序的客户端 ID		字符串
camel.component.google-calendar.configuration.client-secret	日历应用程序的客户端 secret		字符串
camel.component.google-calendar.configuration.email-address	Google Service Account 的 emailAddress。		字符串
camel.component.google-calendar.configuration.method-name	用于所选操作的子操作		字符串
camel.component.google-calendar.configuration.p12-file-name	包含用于 Google 服务帐户的私钥的 p12 文件的名称。		字符串
camel.component.google-calendar.configuration.refresh-token	OAuth 2 刷新令牌。使用这个，Google Calendar 组件可以在当前过期时获取新的 accessToken - 如果应用程序长期到期，则需要获得新的 accessToken。		字符串
camel.component.google-calendar.configuration.scopes	指定您希望日历应用程序具有用户帐户的权限级别。您可以使用逗号分隔多个范围。如需更多信息，请参阅 https://developers.google.com/google-apps/calendar/auth。	https://www.googleapis.com/auth/calendar	字符串

Name	描述	默认值	类型
camel.component.google-calendar.configuration.user	应用程序试图在服务帐户流中模拟的用户的电子邮件地址		字符串
camel.component.google-calendar.enabled	启用 google-calendar 组件	true	布尔值
camel.component.google-calendar.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

119.3. URI 格式

GoogleCalendar 组件使用以下 URI 格式：

```
google-calendar://endpoint-prefix/endpoint?[options]
```

端点前缀可以是以下之一：

- ***acl***
- *日历*
- ***Channels***
- ***colors***
- ***events***
- ***freebusy***

- *list*
- *设置*

119.4. 制作者端点

生产者端点可以使用端点前缀，后跟端点名称和关联的选项。可以将简写别名用于某些端点。端点 URI MUST 包含前缀。

未强制的端点选项由 [] 表示。当端点没有强制选项时，必须提供其中一组 [] 选项。生成者端点也可以使用特殊选项 inBody，它应包含端点选项的名称，其值将包含在 Camel Exchange In 消息中。

任何端点选项都可以在端点 URI 中提供，或者在消息标头中动态提供。消息标头名称的格式必须是 CamelGoogleCalendar.<option>。请注意，inBody 选项会覆盖消息标头，即 Body=option 中的端点选项会覆盖 CamelGoogleCalendar.option 标头。

119.5. 消费者端点

任何制作者端点都可以用作消费者端点。消费者端点可以使用 Scheduled Poll Consumer Options 和 consumer. 前缀来调度端点调用。返回数组或集合的消费者端点将为每个元素生成一个交换，它们的路由将为每个交换执行一次。

119.6. 消息标头

任何 URI 选项都可以在带有 CamelGoogleCalendar. 前缀的制作者端点的消息标头中提供。

119.7. 消息正文

所有结果消息正文都使用 GoogleCalendarComponent 所使用的底层 API 提供的对象。生产者端点可以在 inBody 端点 URI 参数中为传入消息正文指定选项名称。对于返回数组或集合的端点，消费者端点会将每个元素映射到不同的消息。

第 120 章 GOOGLE CALENDAR STREAM COMPONENT

从 Camel 版本 2.23 开始提供

Google Calendar 组件可以通过 Google Calendar Web API 访问 Calendar。

Google Calendar 使用 OAuth 2.0 协议 来验证 Google 帐户并授权对用户数据的访问权限。在可以使用此组件之前，您需要 创建帐户并生成 OAuth 凭据。凭证由 clientId、clientSecret 和 refreshToken 组成。用于生成较长的 refreshToken 的方便资源是 OAuth 播放。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-google-calendar</artifactId>
    <version>2.23.0</version>
</dependency>
```

120.1. URI 格式

Google Calendar 组件使用以下 URI 格式：

```
google-calendar-stream://index?[options]
```

120.2. GOOGLECALENDARSTREAMCOMPONENT

Google Calendar Stream 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
配置（高级）	配置		GoogleCalendarStream Configuration
clientFactory (advanced)	客户端工厂		GoogleCalendarClient Factory

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Google Calendar Stream 端点使用 URI 语法进行配置：

```
google-calendar-stream:index
```

使用以下路径和查询参数：

120.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
index	为端点指定索引		字符串

120.2.2. 查询参数(30 参数)：

Name	描述	默认值	类型
accessToken (consumer)	OAuth 2 访问令牌。这通常在一小时后过期，因此建议使用 refreshToken 进行长期使用。		字符串
applicationName (consumer)	Google Calendar 应用程序名称。示例为 camel-google-calendar/1.0		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
calendarId (consumer)	要使用的 calendarId	primary	字符串
clientId (consumer)	日历应用程序的客户端 ID		字符串
clientSecret (consumer)	日历应用程序的客户端 secret		字符串

Name	描述	默认值	类型
considerLastUpdate (consumer)	考虑最后一个事件轮询的 lastUpdate 作为下一次轮询的开始日期	false	布尔值
consumeFromNow (consumer)	从现在开始消耗所选日历中的事件	true	布尔值
maxResults (consumer)	要返回的最大结果	10	int
query (consumer)	要在日历上执行的查询		字符串
refreshToken (consumer)	OAuth 2 刷新令牌。使用这个，Google Calendar 组件可以在当前过期时获取新的 accessToken - 如果应用程序长期到期，则需要获得新的 accessToken。		字符串
scopes (consumer)	指定您希望日历应用程序具有用户帐户的权限级别。如需更多信息，请参阅 https://developers.google.com/calendar/auth。		list
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int

Name	描述	默认值	类型
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

120.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 15 个选项，如下所列。

Name	描述	默认值	类型
camel.component.google-calendar-stream.client-factory	客户端工厂。选项是 org.apache.camel.component.google.calendar.GoogleCalendarClientFactory 类型。		字符串
camel.component.google-calendar-stream.configuration.access-token	OAuth 2 访问令牌。这通常在一小时后过期，因此建议使用 refreshToken 进行长期使用。		字符串
camel.component.google-calendar-stream.configuration.application-name	Google Calendar 应用程序名称。示例为 camel-google-calendar/1.0		字符串
camel.component.google-calendar-stream.configuration.calendar-id	要使用的 calendarId	primary	字符串
camel.component.google-calendar-stream.configuration.client-id	日历应用程序的客户端 ID		字符串
camel.component.google-calendar-stream.configuration.client-secret	日历应用程序的客户端 secret		字符串
camel.component.google-calendar-stream.configuration.consider-last-update	考虑最后一个事件轮询的 lastUpdate 作为下一次轮询的开始日期	false	布尔值
camel.component.google-calendar-stream.configuration.consume-from-now	从现在开始消耗所选日历中的事件	true	布尔值
camel.component.google-calendar-stream.configuration.index	为端点指定索引		字符串

Name	描述	默认值	类型
camel.component.google-calendar-stream.configuration.max-results	要返回的最大结果	10	整数
camel.component.google-calendar-stream.configuration.query	要在日历上执行的查询		字符串
camel.component.google-calendar-stream.configuration.refresh-token	OAuth 2 刷新令牌。使用这个，Google Calendar 组件可以在当前过期时获取新的 accessToken - 如果应用程序长期到期，则需要获得新的 accessToken。		字符串
camel.component.google-calendar-stream.configuration.scopes	指定您希望日历应用程序具有用户帐户的权限级别。如需更多信息，请参阅 https://developers.google.com/calendar/auth。		list
camel.component.google-calendar-stream.enabled	是否启用 google-calendar-stream 组件的自动配置。这默认是启用的。		布尔值
camel.component.google-calendar-stream.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

120.4. 消费者

默认情况下，使用者将轮询，maxResults 等于 5。

例如：

```
from("google-calendar-stream://test?
markAsRead=true&delay=5000&maxResults=5").to("mock:result");
```

此路由将消耗从轮询日期开始的下一个五个事件。

第 121 章 GOOGLE DRIVE 组件

从 Camel 版本 2.14 开始提供

Google Drive 组件通过 Google Drive Web API 提供对 Google Drive 文件存储服务的访问。

Google Drive 使用 OAuth 2.0 协议 来验证 Google 帐户并授权对用户数据的访问。在可以使用此组件之前，您需要 创建帐户并生成 OAuth 凭据。凭证由 clientId、clientSecret 和 refreshToken 组成。用于生成较长的 refreshToken 的方便资源是 OAuth 播放。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-google-drive</artifactId>
    <version>2.14-SNAPSHOT</version>
</dependency>
```

121.1. URI 格式

GoogleDrive 组件使用以下 URI 格式：

```
google-drive://endpoint-prefix/endpoint?[options]
```

端点前缀可以是以下之一：

- drive-about
- drive-apps
- drive-changes

- *drive-channels*
- *drive-children*
- *drive-comments*
- *drive-files*
- *drive-parents*
- *drive-permissions*
- *drive-properties*
- *drive-realtime*
- *drive-replies*
- *drive-revisions*

121.2. GOOGLEDRIVECOMPONENT

Google Drive 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	使用共享配置		GoogleDrive 配置
clientFactory (advanced)	使用 GoogleCalendarClientFactory 作为创建客户端的工厂。默认情况下将使用 BatchGoogleDriveClientFactory		GoogleDriveClient Factory

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Google Drive 端点使用 URI 语法进行配置：

```
google-drive:apiName/methodName
```

使用以下路径和查询参数：

121.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
apiName	**需要** 执行什么操作		GoogleDriveApiName
methodName	**必需的** 所选操作使用哪些子操作		字符串

121.2.2. 查询参数(12 参数)：

Name	描述	默认值	类型
accessToken (common)	OAuth 2 访问令牌。这通常在一小时后过期，因此建议使用 refreshToken 进行长期使用。		字符串
applicationName (common)	Google 驱动器应用程序名称。示例为 camel-google-drive/1.0		字符串
clientFactory (common)	使用 GoogleCalendarClientFactory 作为创建客户端的工厂。默认情况下将使用 BatchGoogleDriveClientFactory		GoogleDriveClientFactory
clientId (common)	驱动器应用程序的客户端 ID		字符串
clientSecret (common)	驱动器应用程序的客户端 secret		字符串
inBody (common)	设置要在交换 In Body 中传递的参数名称		字符串

Name	描述	默认值	类型
refreshToken (common)	OAuth 2 刷新令牌。使用这个，Google Calendar 组件可以在当前过期时获取新的 accessToken - 如果应用程序长期到期，则需要获得新的 accessToken。		字符串
scopes (common)	指定您希望驱动器应用程序对用户帐户具有的权限级别。如需更多信息，请参阅 https://developers.google.com/drive/web/scopes。		list
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

121.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 11 个选项，如下所列。

Name	描述	默认值	类型
camel.component.google-drive.client-factory	使用 GoogleCalendarClientFactory 作为创建客户端的工厂。默认情况下，将使用 BatchGoogleDriveClientFactory。选项是 org.apache.camel.component.google.drive.GoogleDriveClientFactory 类型。		字符串
camel.component.google-drive.configuration.access-token	OAuth 2 访问令牌。这通常在一小时后过期，因此建议使用 refreshToken 进行长期使用。		字符串

Name	描述	默认值	类型
camel.component.google-drive.configuration.api-name	要执行的操作类型		GoogleDriveApiName
camel.component.google-drive.configuration.application-name	Google 驱动器应用程序名称。示例为 camel-google-drive/1.0		字符串
camel.component.google-drive.configuration.client-id	驱动器应用程序的客户端 ID		字符串
camel.component.google-drive.configuration.client-secret	驱动器应用程序的客户端 secret		字符串
camel.component.google-drive.configuration.method-name	用于所选操作的子操作		字符串
camel.component.google-drive.configuration.refresh-token	OAuth 2 刷新令牌。使用这个，Google Calendar 组件可以在当前过期时获取新的 accessToken - 如果应用程序长期到期，则需要获得新的 accessToken。		字符串
camel.component.google-drive.configuration.scopes	指定您希望驱动器应用程序对用户帐户具有的权限级别。如需更多信息，请参阅 https://developers.google.com/drive/web/scopes。		list
camel.component.google-drive.enabled	启用 google-drive 组件	true	布尔值
camel.component.google-drive.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

121.4. 制作者端点

生产者端点可以使用端点前缀，后跟端点名称和关联的选项。可以将简写别名用于某些端点。端点 URI MUST 包含前缀。

未强制的端点选项由 [] 表示。当端点没有强制选项时，必须提供其中一组 [] 选项。生成者端点也可以使用特殊选项 inBody，它应包含端点选项的名称，其值将包含在 Camel Exchange In 消息中。

任何端点选项都可以在端点 URI 中提供，或者在消息标头中动态提供。消息标头名称的格式必须是 CamelGoogleDrive.<option>。请注意，inBody 选项会覆盖消息标头，即 Body=option 中的端点 选项会覆盖 CamelGoogleDrive.option 标头。

有关端点和选项的更多信息，请参阅 API 文档，网址为：
https://developers.google.com/drive/v2/reference/

121.5. 消费者端点

任何制作者端点都可以用作消费者端点。消费者端点可以使用 Scheduled Poll Consumer Options 和 consumer. 前缀来调度端点调用。返回数组或集合的消费者端点将为每个元素生成一个交换，它们的路由将为每个交换执行一次。

121.6. 消息标头

任何 URI 选项都可以在带有 CamelGoogleDrive. 前缀的制作者端点的消息标头中提供。

121.7. 消息正文

所有结果消息正文都使用 GoogleDriveComponent 使用的基本 API 提供的对象。生产者端点可以在 inBody 端点 URI 参数中为传入消息正文指定选项名称。对于返回数组或集合的端点，消费者端点会将每个元素映射到不同的消息。

第 122 章 GOOGLE MAIL COMPONENT

从 Camel 版本 2.15 开始提供

Google Mail 组件通过 Google Mail Web API 提供对 Gmail 的访问。

Google Mail 使用 OAuth 2.0 协议 来验证 Google 帐户并授权对用户数据的访问权限。在可以使用此组件之前，您需要 创建帐户并生成 OAuth 凭据。凭证由 clientId、clientSecret 和 refreshToken 组成。用于生成较长的 refreshToken 的方便资源是 OAuth 播放。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-google-mail</artifactId>
    <version>2.15-SNAPSHOT</version>
</dependency>
```

122.1. URI 格式

GoogleMail 组件使用以下 URI 格式：

```
google-mail://endpoint-prefix/endpoint?[options]
```

端点前缀可以是以下之一：

- attachments
- 草案
- history
- labels

- ***messages***
- ***threads***
- ***users***

122.2. GOOGLEMAILCOMPONENT

Google Mail 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	使用共享配置		GoogleMailConfiguration
clientFactory (advanced)	使用 GoogleCalendarClientFactory 作为创建客户端的工厂。默认情况下将使用 BatchGoogleMailClientFactory		GoogleMailClientFactory
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Google Mail 端点使用 URI 语法进行配置：

```
google-mail:apiName/methodName
```

使用以下路径和查询参数：

122.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
apiName	**需要** 执行什么操作		GoogleMailApiName
methodName	**必需的** 所选操作使用哪些子操作		字符串

122.2.2. 查询参数(10 parameters):

Name	描述	默认值	类型
accessToken (common)	OAuth 2 访问令牌。这通常在一小时后过期，因此建议使用 refreshToken 进行长期使用。		字符串
applicationName (common)	Google 邮件应用程序名称。示例为 camel-google-mail/1.0		字符串
clientId (common)	邮件应用的客户端 ID		字符串
clientSecret (common)	邮件应用的客户端机密		字符串
inBody (common)	设置要在交换 In Body 中传递的参数名称		字符串
refreshToken (common)	OAuth 2 刷新令牌。使用这个，Google Calendar 组件可以在当前过期时获取新的 accessToken - 如果应用程序长期到期，则需要获得新的 accessToken。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

122.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
camel.component.google-mail.client-factory	使用 GoogleCalendarClientFactory 作为创建客户端的工厂。默认情况下，将使用 BatchGoogleMailClientFactory。选项是 org.apache.camel.component.google.mail.GoogleMailClientFactory 类型。		字符串
camel.component.google-mail.configuration.access-token	OAuth 2 访问令牌。这通常在一小时后过期，因此建议使用 refreshToken 进行长期使用。		字符串
camel.component.google-mail.configuration.api-name	要执行的操作类型		GoogleMailApiName
camel.component.google-mail.configuration.application-name	Google 邮件应用程序名称。示例为 camel-google-mail/1.0		字符串
camel.component.google-mail.configuration.client-id	邮件应用的客户端 ID		字符串
camel.component.google-mail.configuration.client-secret	邮件应用的客户端机密		字符串
camel.component.google-mail.configuration.method-name	用于所选操作的子操作		字符串
camel.component.google-mail.configuration.refresh-token	OAuth 2 刷新令牌。使用这个，Google Calendar 组件可以在当前过期时获取新的 accessToken - 如果应用程序长期到期，则需要获得新的 accessToken。		字符串
camel.component.google-mail.enabled	启用 google-mail 组件	true	布尔值

Name	描述	默认值	类型
camel.component.google-mail.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

122.4. 制作者端点

生产者端点可以使用端点前缀，后跟端点名称和关联的选项。可以将简写别名用于某些端点。端点 URI MUST 包含前缀。

未强制的端点选项由 [] 表示。当端点没有强制选项时，必须提供其中一组 [] 选项。生成者端点也可以使用特殊选项 inBody，它应包含端点选项的名称，其值将包含在 Camel Exchange In 消息中。

任何端点选项都可以在端点 URI 中提供，或者在消息标头中动态提供。消息标头名称的格式必须是 CamelGoogleMail.<option>。请注意，inBody 选项会覆盖消息标头，即 Body=option 中的端点 选项会覆盖 CamelGoogleMail.option 标头。

有关端点和选项的更多信息，请参阅 API 文档，网址为：
https://developers.google.com/gmail/api/v1/reference/

122.5. 消费者端点

任何制作者端点都可以用作消费者端点。消费者端点可以使用 Scheduled Poll Consumer Options 和 consumer. 前缀来调度端点调用。返回数组或集合的消费者端点将为每个元素生成一个交换，它们的路由将为每个交换执行一次。

122.6. 消息标头

任何 URI 选项都可以在带有 CamelGoogleMail. 前缀的制作者端点的消息标头中提供。

122.7. 消息正文

所有结果消息正文都使用 GoogleMailComponent 使用的基本 API 提供的对象。生产者端点可以在 inBody 端点 URI 参数中为传入消息正文指定选项名称。对于返回数组或集合的端点，消费者端点会将每个元素映射到不同的消息。

第 123 章 GOOGLE MAIL STREAM COMPONENT

从 Camel 版本 2.22 开始提供

Google Mail 组件通过 Google Mail Web API 提供对 Gmail 的访问。

Google Mail 使用 OAuth 2.0 协议来验证 Google 帐户并授权对用户数据的访问权限。在可以使用此组件之前，您需要创建帐户并生成 OAuth 凭据。凭证由 clientId、clientSecret 和 refreshToken 组成。用于生成较长的 refreshToken 的方便资源是 OAuth 播放。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-google-mail</artifactId>
    <version>2.22-SNAPSHOT</version>
</dependency>
```

123.1. URI 格式

GoogleMail 组件使用以下 URI 格式：

```
google-mail-stream://index?[options]
```

123.2. GOOGLEMAILSTREAMCOMPONENT

Google Mail Stream 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
配置（高级）	配置		GoogleMailStream Configuration
clientFactory (advanced)	客户端工厂		GoogleMailClient Factory
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Google Mail Stream 端点使用 URI 语法进行配置：

```
google-mail-stream:index
```

使用以下路径和查询参数：

123.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
index	为端点指定索引		字符串

123.2.2. 查询参数(28 参数)：

Name	描述	默认值	类型
accessToken (consumer)	OAuth 2 访问令牌。这通常在一小时后过期，因此建议使用 refreshToken 进行长期使用。		字符串
applicationName (consumer)	Google 邮件应用程序名称。示例为 camel-google-mail/1.0		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
clientId (consumer)	邮件应用的客户端 ID		字符串
clientSecret (consumer)	邮件应用的客户端机密		字符串
标签 (consumer)	要考虑的以逗号分隔的标签列表		字符串
markAsRead (consumer)	将消息标记为读取后，将其标记为读取	false	布尔值
maxResults (consumer)	要返回的最大结果	10	long

Name	描述	默认值	类型
query (consumer)	要在 gmail 框中执行的查询	is:unread	字符串
refreshToken (consumer)	OAuth 2 刷新令牌。使用这个，Google Calendar 组件可以在当前过期时获取新的 accessToken - 如果应用程序长期到期，则需要获得新的 accessToken。		字符串
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值

Name	描述	默认值	类型
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

123.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 13 个选项，如下所列。

Name	描述	默认值	类型
camel.component.google-mail-stream.client-factory	客户端工厂。选项是 org.apache.camel.component.google.mail.GoogleMailClientFactory 类型。		字符串
camel.component.google-mail-stream.configuration.access-token	OAuth 2 访问令牌。这通常在一小时后过期，因此建议使用 refreshToken 进行长期使用。		字符串

Name	描述	默认值	类型
camel.component.google-mail-stream.configuration.application-name	Google 邮件应用程序名称。示例为 camel-google-mail/1.0		字符串
camel.component.google-mail-stream.configuration.client-id	邮件应用的客户端 ID		字符串
camel.component.google-mail-stream.configuration.client-secret	邮件应用的客户端机密		字符串
camel.component.google-mail-stream.configuration.index	为端点指定索引		字符串
camel.component.google-mail-stream.configuration.labels	要考虑的以逗号分隔的标签列表		字符串
camel.component.google-mail-stream.configuration.mark-as-read	将消息标记为读取后，将其标记为读取	false	布尔值
camel.component.google-mail-stream.configuration.max-results	要返回的最大结果	10	Long
camel.component.google-mail-stream.configuration.query	要在 gmail 框中执行的查询	is:unread	字符串
camel.component.google-mail-stream.configuration.refresh-token	OAuth 2 刷新令牌。使用这个，Google Calendar 组件可以在当前过期时获取新的 accessToken - 如果应用程序长期到期，则需要获得新的 accessToken。		字符串

Name	描述	默认值	类型
camel.component.google-mail-stream.enabled	是否启用 google-mail-stream 组件的自动配置。这默认是启用的。		布尔值
camel.component.google-mail-stream.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

123.4. 消费者

默认情况下，消费者将轮询，查询 "is:unread" 和 maxResults 等于 10。

例如：

```
from("google-mail-stream://test?
markAsRead=true&delay=5000&maxResults=5&labels=GitHub,Apache").to("mock:result");
```

此路由将使用带有标签 Github 和 Apache 的未读取消息，它将消息标记为 read。

第 124 章 GOOGLE PUBSUB 组件

从 Camel 版本 2.19 开始提供

Google Pubsub 组件通过 Google Client Services API 提供对 Cloud Pub/Sub Infrastructure 的访问。

当前实现不使用 gRPC。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-google-pubsub</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

124.1. URI 格式

Google Pubsub 组件使用以下 URI 格式：

```
google-pubsub://project-id:destinationName?[options]
```

目的地名称可以是主题或订阅名称。

124.2. 选项

Google Pubsub 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
ConnectionFactory (common)	设置要使用的连接工厂：提供明确管理连接凭证的能力： - 密钥文件的路径 - 服务帐户密钥/电子邮件对		GooglePubsubConnection Factory

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Google Pubsub 端点使用 URI 语法进行配置：

```
google-pubsub:projectId:destinationName
```

使用以下路径和查询参数：

124.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
projectId	**所需的项目** ID		字符串
destinationName	**所需的** 目标名称		字符串

124.2.2. 查询参数(9 参数)：

Name	描述	默认值	类型
ackMode (common)	AUTO = Exchange get ack'ed/nack'ed on completion.NONE = 下游进程必须明确使用 ack/nack	AUTO	AckMode
concurrentConsumers (common)	从订阅中消耗的并行流数量	1	整数
ConnectionFactory (common)	ConnectionFactory 获取到 PubSub Service 的连接。如果未提供默认值，则将使用默认值。		GooglePubsubConnection Factory
loggerId (common)	当匹配所需的父路由时，要使用的日志记录器 ID		字符串
maxMessagesPerPoll (common)	在单个 API 调用中接收服务器的最大消息数	1	整数

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

124.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.google-pubsub.enabled	启用 google-pubsub 组件	true	布尔值
camel.component.google-pubsub.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

124.4. 制作者端点

生产者端点可以接受并传送到 PubSub 个人，并分组交换器。分组的交换设置了 Exchange.GROUPED_EXCHANGE 属性。

Google PubSub 预期有效负载为 byte[] 数组，Producer 端点将发送：

- 字符串正文，使用 byte[] 编码为 UTF-8
- byte[] body as as
- 其他所有内容都被序列化为 byte[] 数组

Map set as message header GooglePubsubConstants.ATTRIBUTES 将作为 PubSub 属性发送。发送到 PubSub 的交换后，PubSub Message ID 将分配给标题 GooglePubsubConstants.MESSAGE_ID。

124.5. 消费者端点

如果在设置为订阅的配置选项的时间周期内未确认，Google PubSub 将恢复该消息。

在交换处理完成后，组件将确认消息。

如果路由抛出异常，则交换会被标记为失败，组件将 NACK 的消息 - 它将立即重新设计。

要 ack/nack，组件使用 Acknowledgement ID 存储为标头 GooglePubsubConstants.ACK_ID 的消息。如果删除了或篡改标头，则 ack 将失败，并在 ack deadline 后再次重新设计消息。

124.6. 消息标头

由消费者端点设置的标头：

- GooglePubsubConstants.MESSAGE_ID
- GooglePubsubConstants.ATTRIBUTES
- GooglePubsubConstants.PUBLISH_TIME

- *GooglePubsubConstants.ACK_ID*

124.7. 消息正文

消费者端点返回消息的内容为 byte[] - 与底层系统发送完全相同。它是用于转换/unmarshall 内容的路由。

124.8. 身份验证配置

Google Pubsub 组件身份验证用于 GCP 服务帐户。如需更多信息，请参阅 Google Cloud Platform Auth 指南

Google 安全凭证可以通过以下两个选项之一显式设置：

- *服务帐户电子邮件和服务帐户密钥(PEM 格式)*
- *GCP 凭证文件位置*

如果两者都设置了，则服务帐户电子邮件/密钥将具有优先权。

或隐式，连接工厂将回退到 应用程序默认凭证。

俄克斯！默认凭据文件的位置可以通过 GOOGLE_APPLICATION_CREDENTIALS 环境变量进行配置。

Service Account Email 和 Service Account Key 可以在 GCP JSON 凭证文件中分别作为 client_email 和 private_key 找到。

124.9. 回滚和重新发送

Google PubSub 的回滚依赖于 Acknowledgement Deadline - Google PubSub 预期接收确认的时间周期。如果尚未收到确认，则会重新发送该消息。

Google 提供了一个 API，用于扩展消息的截止时间。

Google PubSub Documentation中的更多信息

因此，回滚实际上是一个截止时间扩展 API 调用，且现在达到了截止时间，并且消息可以被重新提供给下一消费者。

通过将消息标头 GooglePubsubConstants.ACK_DEADLINE 设置为以秒为单位，可以明确为回滚设置确认期限来延迟消息重新发送。

第 125 章 GOOGLE SHEETS 组件

从 Camel 版本 2.23 开始提供

Google Sheets 组件通过 Google Sheets Web API 提供对 Google Sheets 的访问。

Google Sheets 使用 OAuth 2.0 协议 来验证 Google 帐户并授权对用户数据的访问权限。在可以使用此组件之前，您需要 创建帐户并生成 OAuth 凭据。凭证由 clientId、clientSecret 和 refreshToken 组成。用于生成较长的 refreshToken 的方便资源是 OAuth 播放。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-google-sheets</artifactId>
    <version>2.23.0</version>
</dependency>
```

125.1. URI 格式

GoogleSheets 组件使用以下 URI 格式：

```
google-sheets://endpoint-prefix/endpoint?[options]
```

端点前缀可以是以下之一：

- 电子表格
- data

125.2. GOOGLESHEETSCOMPONENT

Google Sheets 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	使用共享配置		GoogleSheets 配置
clientFactory (advanced)	使用 GoogleSheetsClientFactory 作为创建客户端的工厂。默认情况下将使用 BatchGoogleSheetsClientFactory		GoogleSheetsClient Factory
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Google Sheets 端点使用 URI 语法进行配置：

```
google-sheets:apiName/methodName
```

使用以下路径和查询参数：

125.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
apiName	**需要** 执行什么操作		GoogleSheetsApiName
methodName	**必需的** 所选操作使用哪些子操作		字符串

125.2.2. 查询参数(10 parameters):

Name	描述	默认值	类型
accessToken (common)	OAuth 2 访问令牌。这通常在一小时后过期，因此建议使用 refreshToken 进行长期使用。		字符串
applicationName (common)	Google Sheets 应用程序名称。示例为 camel-google-sheets/1.0		字符串
clientId (common)	sheets 应用程序的客户端 ID		字符串
clientSecret (common)	sheets 应用程序的客户端 secret		字符串

Name	描述	默认值	类型
inBody (common)	设置要在交换 In Body 中传递的参数名称		字符串
refreshToken (common)	OAuth 2 刷新令牌。使用这个，Google Sheets 组件可以在当前过期时获取新的 accessToken - 如果应用程序是长期的。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

125.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
camel.component.google-sheets.client-factory	使用 GoogleSheetsClientFactory 作为创建客户端的工厂。默认情况下，将使用 BatchGoogleSheetsClientFactory。选项是 org.apache.camel.component.google.sheets.GoogleSheetsClientFactory 类型。		字符串
camel.component.google-sheets.configuration.access-token	OAuth 2 访问令牌。这通常在一小时后过期，因此建议使用 refreshToken 进行长期使用。		字符串
camel.component.google-sheets.configuration.api-name	要执行的操作类型		GoogleSheetsApiName

Name	描述	默认值	类型
camel.component.google-sheets.configuration.application-name	Google Sheets 应用程序名称。示例为 camel-google-sheets/1.0		字符串
camel.component.google-sheets.configuration.client-id	sheets 应用程序的客户端 ID		字符串
camel.component.google-sheets.configuration.client-secret	sheets 应用程序的客户端 secret		字符串
camel.component.google-sheets.configuration.method-name	用于所选操作的子操作		字符串
camel.component.google-sheets.configuration.refresh-token	OAuth 2 刷新令牌。使用这个，Google Sheets 组件可以在当前过期时获取新的 accessToken - 如果应用程序是长期的。		字符串
camel.component.google-sheets.enabled	是否启用 google-sheets 组件的自动配置。这默认是启用的。		布尔值
camel.component.google-sheets.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

125.4. 制作者端点

生产者端点可以使用端点前缀，后跟端点名称和关联的选项。可以将简写别名用于某些端点。端点 URI MUST 包含前缀。

未强制的端点选项由 [] 表示。当端点没有强制选项时，必须提供其中一组 [] 选项。生成者端点也可以使用特殊选项 inBody，它应包含端点选项的名称，其值将包含在 Camel Exchange In 消息中。

任何端点选项都可以在端点 URI 中提供，或者在消息标头中动态提供。消息标头名称的格式必须是

CamelGoogleSheets.<option>。请注意，inBody 选项会覆盖消息标头，即 Body=option 中的端点选项会覆盖 CamelGoogleSheets.option 标头。

有关端点和选项的更多信息，请参阅 API 文档，网址为：
https://developers.google.com/sheets/api/reference/rest/

125.5. 消费者端点

任何制作者端点都可以用作消费者端点。消费者端点可以使用 Scheduled Poll Consumer Options 和 consumer. 前缀来调度端点调用。返回数组或集合的消费者端点将为每个元素生成一个交换，它们的路由将为每个交换执行一次。

125.6. 消息标头

任何 URI 选项可以在带有 CamelGoogleSheets. 前缀的制作者端点的消息标头中提供。

125.7. 消息正文

所有结果消息正文都使用 GoogleSheetsComponent 所使用的底层 API 提供的对象。生产者端点可以在 inBody 端点 URI 参数中为传入消息正文指定选项名称。对于返回数组或集合的端点，消费者端点会将每个元素映射到不同的消息。

第 126 章 GOOGLE SHEETS STREAM 组件

从 Camel 版本 2.23 开始提供

Google Sheets 组件通过 Google Sheets Web API 提供对 Sheets 的访问。

Google Sheets 使用 OAuth 2.0 协议 来验证 Google 帐户并授权对用户数据的访问权限。在可以使用此组件之前，您需要 创建帐户并生成 OAuth 凭据。凭证由 clientId、clientSecret 和 refreshToken 组成。用于生成较长的 refreshToken 的方便资源是 OAuth 播放。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-google-sheets</artifactId>
    <version>2.23.0</version>
</dependency>
```

126.1. URI 格式

Google Sheets 组件使用以下 URI 格式：

```
google-sheets-stream://apiName?[options]
```

126.2. GOOGLESHEETSSTREAMCOMPONENT

Google Sheets Stream 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
configuration (consumer)	使用共享配置		GoogleSheetsStream Configuration
clientFactory (advanced)	使用 GoogleSheetsClientFactory 作为创建客户端的工厂。默认情况下将使用 BatchGoogleSheetsClientFactory		GoogleSheetsClient Factory

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Google Sheets Stream 端点使用 URI 语法进行配置：

```
google-sheets-stream:apiName
```

使用以下路径和查询参数：

126.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
apiName	设置 apiName。		字符串

126.2.2. 查询参数(31 参数)：

Name	描述	默认值	类型
accessToken (consumer)	OAuth 2 访问令牌。这通常在一小时后过期，因此建议使用 refreshToken 进行长期使用。		字符串
applicationName (consumer)	Google sheets 应用程序名称。示例为 camel-google-sheets/1.0		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
clientId (consumer)	sheets 应用程序的客户端 ID		字符串
clientSecret (consumer)	sheets 应用程序的客户端 secret		字符串
includeGridData (consumer)	如果应该返回网格数据，则为 true。	false	布尔值

Name	描述	默认值	类型
majorDimension (consumer)	指定结果应使用的主要维度。	ROWS	字符串
maxResults (consumer)	指定返回的结果的最大数量。这将限制返回值范围数据集中的行数或批处理请求中返回的值范围的数量。	10	int
range (consumer)	指定表中要从中获取数据的行和列范围。		字符串
refreshToken (consumer)	OAuth 2 刷新令牌。使用这个，Google Calendar 组件可以在当前过期时获取新的 accessToken - 如果应用程序长期到期，则需要获得新的 accessToken。		字符串
scopes (consumer)	指定您希望 sheets 应用程序具有用户帐户的权限级别。如需更多信息，请参阅 https://developers.google.com/identity/protocols/googlescopes。		list
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
spreadsheetId (consumer)	指定用于标识要获取的目标的电子表格标识符。		字符串
valueRenderOption (consumer)	决定在输出中如何呈现值。	FORMATTED_VALUE	字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

Name	描述	默认值	类型
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

126.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 16 个选项，如下所列。

Name	描述	默认值	类型
camel.component.google-sheets-stream.client-factory	使用 GoogleSheetsClientFactory 作为创建客户端的工厂。默认情况下，将使用 BatchGoogleSheetsClientFactory。选项是 org.apache.camel.component.google.sheets.GoogleSheetsClientFactory 类型。		字符串
camel.component.google-sheets-stream.configuration.access-token	OAuth 2 访问令牌。这通常在一小时后过期，因此建议使用 refreshToken 进行长期使用。		字符串
camel.component.google-sheets-stream.configuration.api-name	设置 apiName。		字符串
camel.component.google-sheets-stream.configuration.application-name	Google sheets 应用程序名称。示例为 camel-google-sheets/1.0		字符串
camel.component.google-sheets-stream.configuration.client-id	sheets 应用程序的客户端 ID		字符串
camel.component.google-sheets-stream.configuration.client-secret	sheets 应用程序的客户端 secret		字符串
camel.component.google-sheets-stream.configuration.include-grid-data	如果应该返回网格数据，则为 true。	false	布尔值
camel.component.google-sheets-stream.configuration.major-dimension	指定结果应使用的主要维度。	ROWS	字符串

Name	描述	默认值	类型
camel.component.google-sheets-stream.configuration.max-results	指定返回的结果的最大数量。这将限制返回值范围数据集中的行数或批处理请求中返回的值范围的数量。	0	整数
camel.component.google-sheets-stream.configuration.range	指定表中要从中获取数据的行和列范围。		字符串
camel.component.google-sheets-stream.configuration.refresh-token	OAuth 2 刷新令牌。使用这个，Google Calendar 组件可以在当前过期时获取新的 accessToken - 如果应用程序长期到期，则需要获得新的 accessToken。		字符串
camel.component.google-sheets-stream.configuration.scopes	指定您希望 sheets 应用程序具有用户帐户的权限级别。如需更多信息，请参阅 https://developers.google.com/identity/protocols/googlescopes。		list
camel.component.google-sheets-stream.configuration.spreadsheet-id	指定用于标识要获取的目标的电子表格标识符。		字符串
camel.component.google-sheets-stream.configuration.value-render-option	决定在输出中如何呈现值。	FORMATTED_VALUE	字符串
camel.component.google-sheets-stream.enabled	是否启用 google-sheets-stream 组件的自动配置。这默认是启用的。		布尔值
camel.component.google-sheets-stream.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

126.4. 消费者

默认情况下，使用者将轮询， ***maxResults*** *等于* ***5。***

例如：

```
from("google-sheets-stream://data?
range=A:B&delay=5000&maxResults=5").to("mock:result");
```

此路由将消耗从轮询日期开始的下一个十个事件。

第 127 章 GROOVY 语言

从 Camel 版本 1.3 开始提供

Camel 支持通过 Groovy 和其他脚本语言，以允许在 DSL 或 Xml 配置中使用 Expression 或 Predicate。

要使用 Groovy 表达式，请使用以下 Java 代码

```
... groovy("someGroovyExpression") ...
```

例如，您可以使用 groovy 功能在 Message Filter 中创建一个 Predicate，或作为 Recipient List 的 Expression

127.1. GROOVY 选项

Groovy 语言支持 1 个选项，如下所列。

Name	默认值	Java 类型	描述
trim	**true**	**布尔值**	是否修剪值以移除前导和结尾的空格和换行符

127.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.language.groovy.enabled	启用 groovy 语言	true	布尔值
camel.language.groovy.trim	是否修剪值以移除前导和结尾的空格和换行符	true	布尔值

127.3. 自定义 GROOVY SHELL

有时，您可能需要在 Groovy 表达式中使用自定义 GroovyShell 实例。要提供自定义 GroovyShell，请将 org.apache.camel.language.groovy.GroovyShellFactory SPI 接口的实现添加到 Camel registry 中。例如，在将以下 bean 添加到 Spring context... 后

```
public class CustomGroovyShellFactory implements GroovyShellFactory {

  public GroovyShell createGroovyShell(Exchange exchange) {
    ImportCustomizer importCustomizer = new ImportCustomizer();
    importCustomizer.addStaticStars("com.example.Utils");
    CompilerConfiguration configuration = new CompilerConfiguration();
    configuration.addCompilationCustomizers(importCustomizer);
    return new GroovyShell(configuration);
  }

}
```

...Camel 将使用您的自定义 GroovyShell 实例（包含您的自定义静态导入），而不是默认导入。

127.4. EXAMPLE

```
// lets route if a line item is over $100
from("queue:foo").filter(groovy("request.lineItems.any { i -> i.value > 100 }")).to("queue:bar")
```

和 Spring DSL:

```
<route>
  <from uri="queue:foo"/>
  <filter>
    <groovy>request.lineItems.any { i -> i.value > 100 }</groovy>
    <to uri="queue:bar"/>
  </filter>
</route>
```

127.5. SCRIPTCONTEXT

JSR-223 脚本语言 ScriptContext 预先配置了以下属性，它们都通过 ENGINE_SCOPE 设置：

属性	类型	value

属性	类型	value
context	**org.apache.camel.CamelContext**	Camel 上下文（它不能用于 groovy）
camelContext	**org.apache.camel.CamelContext**	Camel 上下文
交换	**org.apache.camel.Exchange**	当前的交换
request	**org.apache.camel.Message**	消息(IN message)
response	**org.apache.camel.Message**	**弃用**: OUT 消息。如果默认为空，则 OUT 消息。改为使用 IN message。
属性	**org.apache.camel.builder.script.PropertiesFunction**	Camel 2.9： 带有 **解析** 方法的功能，以便更轻松地使用脚本中的 Camels Properties 组件。请参阅以下示例。

有关支持显式 DSL 的语言列表，请参阅脚本语言。

127.6. SCRIPTINGENGINE 的额外参数

从 Camel 2.8 开始提供

您可以使用 Camel 消息上的标头和密钥 CamelScriptArguments 为 ScriptingEngine 提供额外的参数。
请参见以下示例：

127.7. 使用属性功能

从 Camel 2.9 开始可用

如果您需要使用脚本中的 Properties 组件来查找属性占位符，则其太繁琐。例如，使用属性占位符中的值设置标头名称 myHeader，该键在名为 "foo" 的标头中提供。

```
.setHeader("myHeader").groovy(""context.resolvePropertyPlaceholders( + '{{' +
request.headers.get(&#39;foo&#39;) + '}}' + ")")
```

从 Camel 2.9 开始，您现在可以使用属性功能，同一示例更为简单：

```
.setHeader("myHeader").groovy("properties.resolve(request.headers.get(&#39;foo&#39;))")
```

127.8. 从外部资源载入脚本

从 Camel 2.11 开始提供

您可以对脚本进行外部化，并让 Camel 从资源（如 "classpath:"、"file:" 或 "http:" ）加载它。这可以通过以下语法完成：" resource:scheme:location"，例如引用您可以进行的类路径上的文件：

```
.setHeader("myHeader").groovy("resource:classpath:mygroovy.groovy")
```

127.9. 如何从多个语句脚本获得结果

从 Camel 2.14 开始提供

因为 scripteengine evale 方法只需运行多个统计数据脚本，则返回 Null。Camel 现在通过使用来自值集的"result"的键查找脚本结果值。如果您有多个语句脚本，则需要确保设置 result 变量的值，因为脚

本返回值。

```
bar = "baz";
# some other statements ...
# camel take the result value as the script evaluation result
result = body * 2 + 1
```

127.10. 依赖项

要在 camel 路由中使用脚本语言，您需要添加对 camel-groovy 的依赖。

如果使用 Maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-groovy</artifactId>
  <version>x.x.x</version>
</dependency>
```

第 128 章 GRPC COMPONENT

从 Camel 版本 2.19 开始提供

gRPC 组件允许您通过 HTTP/2 传输使用 协议缓冲器(protobuf) 交换格式调用或公开远程过程调用(RPC)服务。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-grpc</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

从 Camel 2.22 gRCP 开始，需要足够拥有 Google Guava 版本。需要将以下配置属性添加到 maven 文件中

```
<properties>
  <google-guava-version>${grpc-guava-version}</google-guava-version>
<properties>
```

128.1. URI 格式

```
grpc://service[?options]
```

128.2. 端点选项

gRPC 组件没有选项。

gRPC 端点使用 URI 语法进行配置：

```
grpc:host:port/service
```

使用以下路径和查询参数：

128.2.1. 路径参数(3 参数)：

Name	描述	默认值	类型
主机	**必需的** gRPC 服务器主机名。在使用生成者时，这是 localhost 或 0.0.0.0，当作为消费者或远程服务器主机名时。		字符串
port	**需要** gRPC 本地或远程服务器端口		int
service	从协议缓冲区描述符文件（软件包点服务定义名称）中需要完全限定的服务名称。		字符串

128.2.2. 查询参数(25 参数)：

Name	描述	默认值	类型
flowControlWindow (common)	HTTP/2 流控制窗口大小(MiB)	1048576	int
maxMessageSize (common)	允许接收/sent (MiB)的最大消息大小。	4194304	int
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
consumerStrategy (consumer)	此选项指定在流传输模式中处理服务请求和响应的顶级策略。如果选择了聚合策略，则会在列表中累积所有请求，然后传送到流，累积的响应将发送到发送者。如果选择了传播策略，请求将发送到流，并且响应会立即发送到发送者。	传播	GrpcConsumerStrategy
forwardOnCompleted (consumer)	确定是否应该将 onCompleted 事件推送到 Camel 路由。	false	布尔值
forwardOnError (consumer)	决定 onError 事件是否应该推送到 Camel 路由。例外将设置为消息正文。	false	布尔值
maxConcurrentCallsPerConnection (consumer)	每个传入服务器连接允许的最大并发调用数	2147483647	int

Name	描述	默认值	类型
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
method (producer)	gRPC 方法名称		字符串
producerStrategy (producer)	用于与远程 gRPC 服务器通信的模式。在 SIMPLE 模式中，单个交换转换为远程过程调用。在 STREAMING 模式中，所有交换都将在同一请求中发送（接收者 gRPC 服务的输入和输出必须是 'stream'）。	SIMPLE	GrpcProducerStrategy
streamRepliesTo (producer)	使用 STREAMING 客户端模式时，它指示应转发响应的端点。		字符串
userAgent (producer)	传递给服务器的用户代理标头		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
authenticationType (security)	验证方法首先进入 SSL/TLS 协商	NONE	GrpcAuthType
JWTAlgorithm (security)	JSON Web 令牌签名算法	HMAC256	JwtAlgorithm
JWTIssuer (security)	JSON Web 令牌签发者		字符串
jwtSecret (security)	JSON Web 令牌 secret		字符串
JWTSubject (security)	JSON Web 令牌主题		字符串
keyCertChainResource (security)	PEM 格式的 X.509 证书链文件资源		字符串
keyPassword (security)	PKCS#8 私钥文件密码		字符串

Name	描述	默认值	类型
keyResource (security)	PEM 格式的 PKCS#8 私钥文件资源		字符串
negotiationType (security)	标识用于 HTTP/2 通信的安全协商类型	PLAIN TEXT	NegotiationType
serviceAccountResource (security)	Google Cloud SDK 支持的 JSON 格式的服务帐户密钥文件		字符串
trustCertCollectionResource (security)	PEM 格式的可信证书集合文件资源以验证远程端点的证书		字符串

128.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.grpc.enabled	启用 grpc 组件	true	布尔值
camel.component.grpc.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

128.4. 传输安全性和身份验证支持（可从 CAMEL 2.20获得）

以下 身份验证机制 内置了 gRPC，在此组件中可用：

- *SSL/TLS：gRPC 具有 SSL/TLS 集成，并推广使用 SSL/TLS 验证服务器，并加密客户端和服务器之间交换的所有数据。客户端可以使用可选机制为 mutual 身份验证提供证书。*
- *使用 Google 的基于令牌的身份验证：gRPC 提供了将基于元数据的凭证附加到请求和响应的通用机制。提供了在通过 gRPC 访问 Google API 时获取访问令牌的额外支持。通常，这个机制必须被用作频道上的 SSL/TLS。*

要启用这些功能，必须配置以下组件属性组合：

num.	选项	参数	value	必填/选填
1	**SSL/TLS**	negotiationType	TLS	必填
		keyCertChainResource		必填
		keyResource		必填
		keyPassword		选填
		trustCertCollectionResource		选填
2	**使用 Google API 的基于令牌的身份验证**	authenticationType	GOOGLE	必填
		negotiationType	TLS	必填
		serviceAccountResource		必填
3	**自定义 JSON Web 令牌实施身份验证**	authenticationType	JWT	必填
		negotiationType	NONE 或 TLS	可选。TLS/SSL 不检查此类型，但强烈建议使用。
		jwtAlgorithm	HMAC256 (default)或 (HMAC384, HMAC512)	选填
		jwtSecret		必填
		jwtIssuer		选填
		jwtSubject		选填

目前，使用 OpenSSL 的 TLS 是通过 TLS 组件使用 gRPC 的建议方法。将 JDK 用于 ALPN 通常较慢，可能不支持 HTTP2 所需的密码。此功能没有在组件中实施。

128.5. GRPC PRODUCER 资源类型映射

下表显示了消息正文中的对象类型，具体取决于传入和传出参数的类型（简单或流），以及调用样式（同步或异步）。请注意，不允许以异步样式调用带有传入流参数的流程。

调用风格	请求类型	响应类型	请求正文类型	结果正文类型
同步	simple	simple	对象	对象
同步	simple	流	对象	List<Object>
同步	流	simple	不允许	不允许
同步	流	流	不允许	不允许
异步	simple	simple	对象	List<Object>
异步	simple	流	对象	List<Object>
异步	流	simple	对象或列表<Object>	List<Object>
异步	流	流	对象或列表<Object>	List<Object>

128.6. GRPC 消费者标头（将在消费者调用后安装）

标头名称	描述	可能的值
CamelGrpcMethodName	由消费者服务处理的方法名称	
CamelGrpcEventType	从发送请求接收的事件类型	onNext, onCompleted 或 onError
CamelGrpcUserAgent	如果提供，则给定代理将预先填充 gRPC 库的用户代理信息	

128.7. 例子

以下是使用主机和端口参数调用的简单同步方法

```
from("direct:grpc-sync")
.to("grpc://remotehost:1101/org.apache.camel.component.grpc.PingPong?
method=sendPing&synchronous=true");
```

```
<route>
```

```
    <from uri="direct:grpc-sync" />
    <to uri="grpc://remotehost:1101/org.apache.camel.component.grpc.PingPong?
method=sendPing&synchronous=true"/>
</route>
```

一个异步方法调用

```
from("direct:grpc-async")
.to("grpc://remotehost:1101/org.apache.camel.component.grpc.PingPong?
method=pingAsyncResponse");
```

带有传播消费者策略的 gRPC 服务消费者

```
from("grpc://localhost:1101/org.apache.camel.component.grpc.PingPong?
consumerStrategy=PROPAGATION")
.to("direct:grpc-service");
```

带有流生成者策略的 gRPC 服务生成者（需要一个使用 "stream" 模式作为输入和输出的服务）

```
from("direct:grpc-request-stream")
.to("grpc://remotehost:1101/org.apache.camel.component.grpc.PingPong?
method=PingAsyncAsync&producerStrategy=STREAMING&streamRepliesTo=direct:grpc-
response-stream");

from("direct:grpc-response-stream")
.log("Response received: ${body}");
```

gRPC 服务消费者 TLS/SLL 安全协商启用

```
from("grpc://localhost:1101/org.apache.camel.component.grpc.PingPong?
consumerStrategy=PROPAGATION&negotiationType=TLS&keyCertChainResource=file:src/te
st/resources/certs/server.pem&keyResource=file:src/test/resources/certs/server.key&trustCert
CollectionResource=file:src/test/resources/certs/ca.pem")
.to("direct:tls-enable")
```

带有自定义 JSON Web Token 实现身份验证的 gRPC 服务制作者

```
from("direct:grpc-jwt")
.to("grpc://localhost:1101/org.apache.camel.component.grpc.PingPong?
method=pingSyncSync&synchronous=true&authenticationType=JWT&jwtSecret=supersecure
dsecret");
```

128.8. 配置

建议您使用 Maven 协议缓冲插件调用协议缓冲区编译器(protoc)工具，从自定义项目的 .proto （协议缓冲区定义）文件中生成 Java 源文件。此插件也会生成流程请求和响应类、其构建器和 gRPC 流程存根类。

以下步骤是必需的：

在项目 pom.xml 的 < build> 标签中插入操作系统和 CPU 架构检测扩展，或者手动设置 ${os.detected.classifier} 参数

```
<extensions>
  <extension>
    <groupId>kr.motd.maven</groupId>
    <artifactId>os-maven-plugin</artifactId>
    <version>1.4.1.Final</version>
  </extension>
</extensions>
```

插入 gRPC 和 protobuf Java 代码生成器插件 & lt;plugins > 项目 pom.xml 标签

```
<plugin>
  <groupId>org.xolstice.maven.plugins</groupId>
  <artifactId>protobuf-maven-plugin</artifactId>
  <version>0.5.0</version>
  <configuration>
    <protocArtifact>com.google.protobuf:protoc:${protobuf-
version}:exe:${os.detected.classifier}</protocArtifact>
    <pluginId>grpc-java</pluginId>
    <pluginArtifact>io.grpc:protoc-gen-grpc-java:${grpc-
version}:exe:${os.detected.classifier}</pluginArtifact>
  </configuration>
  <executions>
    <execution>
      <goals>
        <goal>compile</goal>
        <goal>compile-custom</goal>
        <goal>test-compile</goal>
        <goal>test-compile-custom</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

128.9. 如需更多信息，请参阅这些资源

gRPC 项目站点

Maven 协议缓冲器插件

128.10. 另请参阅

- *开始使用*
- *配置 Camel*
- *组件*
- *端点*
- *协议缓冲数据格式*

第 129 章 GUAVA EVENTBUS COMPONENT

从 Camel 版本 2.10 开始提供

Google Guava EventBus 允许组件之间的发布订阅式通信，而无需组件明确相互注册（从而知道彼此）。guava-eventbus: 组件在 Camel 和 Google Guava EventBus 基础架构之间提供集成网桥。使用后者组件时，与 Guava EventBus 交换的消息可以透明地转发到 Camel 路由。EventBus 组件还允许将 Camel 交换的正文路由到 Guava EventBus。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-guava-eventbus</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

129.1. URI 格式

```
guava-eventbus:busName[?options]
```

其中 busName 代表 Camel registry 中的 com.google.common.eventbus.EventBus 实例的名称。

129.2. 选项

Guava EventBus 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
eventBus (common)	使用给定的 Guava EventBus 实例		EventBus
listenerInterface (common)	带有 Subscribe 注解标记的方法接口。动态代理将通过接口创建，以便将其注册为 EventBus 侦听器。在创建多事件监听程序和正确处理 DeadEvent 时，特别有用。这个选项不能与 eventClass 选项一同使用。		类
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Guava EventBus 端点使用 URI 语法进行配置：

```
guava-eventbus:eventBusRef
```

使用以下路径和查询参数：

129.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
eventBusRef	从带有给定名称的 registry 中查找 Guava EventBus		字符串

129.2.2. 查询参数(6 参数)：

Name	描述	默认值	类型
eventClass (common)	如果用于路由的消费者端，会将从 EventBus 收到的事件过滤到类的实例和 eventClass 的 superclasses。这个选项的 null 值等于将其设置为 java.lang.Object i.e。消费者将捕获传入事件总线的所有消息。这个选项不能与 listenerInterface 选项一同使用。		类
listenerInterface (common)	带有 Subscribe 注解标记的方法接口。动态代理将通过接口创建，以便将其注册为 EventBus 侦听器。在创建多事件监听程序和正确处理 DeadEvent 时，特别有用。这个选项不能与 eventClass 选项一同使用。		类
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

129.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.guava-eventbus.enabled	启用 guava-eventbus 组件	true	布尔值
camel.component.guava-eventbus.event-bus	使用给定的 Guava EventBus 实例。选项是一个 com.google.common.eventbus.EventBus 类型。		字符串
camel.component.guava-eventbus.listener-interface	带有 Subscribe 注解标记的方法接口。动态代理将通过接口创建，以便将其注册为 EventBus 侦听器。在创建多事件监听程序和正确处理 DeadEvent 时，特别有用。这个选项不能与 eventClass 选项一同使用。		类
camel.component.guava-eventbus.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

129.4. 使用方法

在路由的消费者端使用 guava-eventbus 组件将捕获发送到 Guava EventBus 的消息并将其转发到 Camel 路由。guava EventBus 消费者 异步 处理传入的信息。

```
SimpleRegistry registry = new SimpleRegistry();
EventBus eventBus = new EventBus();
registry.put("busName", eventBus);
CamelContext camel = new DefaultCamelContext(registry);

from("guava-eventbus:busName").to("seda:queue");

eventBus.post("Send me to the SEDA queue.");
```

在路由的生成者一侧使用 guava-eventbus 组件会将 Camel 交换的正文转发到 Guava EventBus 实例。

```
SimpleRegistry registry = new SimpleRegistry();
EventBus eventBus = new EventBus();
registry.put("busName", eventBus);
```

```
CamelContext camel = new DefaultCamelContext(registry);

from("direct:start").to("guava-eventbus:busName");

ProducerTemplate producerTemplate = camel.createProducerTemplate();
producer.sendBody("direct:start", "Send me to the Guava EventBus.");

eventBus.register(new Object(){
  @Subscribe
  public void messageHander(String message) {
    System.out.println("Message received from the Camel: " + message);
  }
});
```

129.5. DEADEVENT 注意事项

请记住，由于 Guava EventBus 设计的限制，您无法指定监听器接收的事件类，而无需创建带有 @Subscribe 方法的类。这个限制意味着指定 eventClass 选项的端点实际侦听所有可能的事件 (java.lang.Object)，并在运行时以编程方式过滤适当的消息。以下 snipped 演示了 Camel 代码库中的相应摘录。

```
@Subscribe
public void eventReceived(Object event) {
  if (eventClass == null || eventClass.isAssignableFrom(event.getClass())) {
    doEventReceived(event);
...
```

这种方法的这种缺点是，Camel 使用的 EventBus 实例永远不会生成 com.google.common.eventbus.DeadEvent 通知。如果您希望 Camel 仅侦听精确指定的事件（因此启用 DeadEvent 支持），请使用 listenerInterface 端点选项。Camel 将在您使用 latter 选项指定的接口创建动态代理，并仅侦听由接口处理器方法指定的消息。以下是仅处理 SpecificEvent 实例的监听程序接口的示例。

```
package com.example;

public interface CustomListener {

  @Subscribe
  void eventReceived(SpecificEvent event);

}
```

以上显示的监听程序可以在端点定义中使用，如下所示：

```
from("guava-eventbus:busName?
listenerInterface=com.example.CustomListener").to("seda:queue");
```

129.6. 消耗多种类型的事件

为了定义由 Guava EventBus 使用者使用 listenerInterface endpoint 选项所消耗的多种事件类型，因为监听器接口可以提供标有 @Subscribe 注释的多个方法。

```
package com.example;

public interface MultipleEventsListener {

  @Subscribe
  void someEventReceived(SomeEvent event);

  @Subscribe
  void anotherEventReceived(AnotherEvent event);

}
```

以上显示的监听程序可以在端点定义中使用，如下所示：

```
from("guava-eventbus:busName?
listenerInterface=com.example.MultipleEventsListener").to("seda:queue");
```

第 130 章 HAWTDB （已弃用）

从 Camel 2.3 开始提供

HawtDB 是一个轻量级且可嵌入的键值数据库。它允许与 Camel 结合使用，为各种 Camel 功能（如聚合器）提供持久支持。

它提供的当前功能：

- HawtDBAggregationRepository

已弃用

HawtDB 项目已弃用，由 leveldb 替代，作为轻量级和可嵌入的键值数据库。为了便于使用 leveldb，为此有一个 leveldbjni 项目。Apache ActiveMQ 项目正在计划将 leveldb 用作其基于主文件的主文件存储，以取代 kahadb。

我们建议使用 camel-leveldb 组件而不是它。

HawtDB 1.4 或更早版本的问题

HawtDB 1.4 或更早版本中有一个错误，这意味着 filestore 不会释放未使用的空间。这意味着文件不断增长。这个问题已在 HawtDB 1.5 中解决，它随 Camel 2.5 开始提供。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-hawtdb</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

它提供的当前功能：

- *HawtDBAggregationRepository*

130.1. 使用 HAWTDBAGGREGATIONREPOSITORY

HawtDBAggregationRepository 是一个 AggregationRepository，实时保留聚合的消息。这样可确保您不会松散消息，因为默认聚合器将使用仅在内存中的 AggregationRepository。

它有以下选项：

选项	类型	描述
repositoryName	字符串	必需的仓库名称。允许您将 shared **HawtDBFile** 用于多个存储库。
persistentFileName	字符串	持久性存储的文件名。如果启动文件时不存在任何文件，则创建新文件。
bufferSize	int	映射到文件存储的内存片段的大小。默认情况下，其 8mb。该值以字节为单位。
sync	布尔值	**HawtDBFile** 是否应在写入时同步。默认为 **true**。通过同步写入，可以确保始终等待所有写入假脱机到磁盘，因此不会造成松散更新。如果您禁用这个选项，则 HawtDB 会在批处理了多个写入时自动同步。
pageSize	short	内存页面的大小。默认情况下，其 512 字节。该值以字节为单位。
hawtDBFile	HawtDBFile	使用现有的 **org.apache.camel.component.hawtdb.HawtDBFile** 实例。
returnOldExchange	布尔值	get 操作是否应该返回旧的现有 Exchange （如果存在）。默认情况下，此选项为 **false**，以优化，因为在聚合时我们不需要旧的交换。
useRecovery	布尔值	是否启用恢复。此选项默认为 **true**。启用 Camel Aggregator 自动恢复失败的聚合交换，并重新提交。
recoveryInterval	long	如果启用了恢复，则每次 x 次运行后台任务，以扫描失败的交换以恢复并重新提交。默认情况下，这个间隔为 5000 millis。

选项	类型	描述
maximumRedeliveries	int	允许您限制恢复的交换的最大重新发送尝试次数。如果启用，如果所有重新发送尝试都失败，则交换将移到死信频道。默认情况下禁用这个选项。如果使用这个选项，则必须提供 **deadLetterUri** 选项。
deadLetterUri	字符串	一个死信频道的端点 uri，将移动耗尽恢复的交换。如果使用这个选项，则必须提供 **maximumRedeliveries** 选项。
optimisticLocking	**false**	Camel 2.12 ： 要打开最佳锁定，在多个 Camel 应用程序共享基于 HawtDB 的聚合存储库的集群环境中通常需要这种锁定。

必须提供 repositoryName 选项。然后，必须提供 persistentFileName 或 hawtDBFile。

130.1.1. 持久性时保留的内容

HawtDBAggregationRepository 将仅保留任何 Serializable 兼容数据类型。如果数据类型不是这样一个类型，则会丢弃它，并记录 WARN。它只会保留消息正文和 Message 标头。Exchange 属性 不会被保留。

130.1.2. 恢复

HawtDBAggregationRepository 默认恢复任何失败的 Exchange。它通过在持久性存储中扫描失败的交换的后台任务来实现此目的。您可以使用 checkInterval 选项设置此任务运行的频率。恢复充当事务处理，可确保 Camel 会尝试恢复并恢复失败的交换。发现恢复的所有交换都将从持久存储恢复并重新提交并再次发送出。

当交换被恢复/冗余时，会设置以下标头：

标头	类型	描述
Exchange.REDELIVERED	布尔值	设置为 true，表示 Exchange 正在重新设计。

标头	类型	描述
Exchange.REDELIVERY_COUNTER	整数	从 1 开始重新发送尝试。

只有在成功处理交换时，才会将其标记为 complete，只有在 AggregationRepository 上调用 confirm 方法时才会发生这种情况。这意味着，如果同一交换再次失败，它将被重试，直到成功为止。

您可以使用选项 maximumRedeliveries 来限制给定恢复的交换的最大重新发送尝试次数。您还必须设置 deadLetterUri 选项，以便 Camel 知道在 max Redeliveries 正在命中时要发送交换的位置。

您可以在 camel-hawtdb 的单元测试中看到一些示例，例如 此测试。

130.1.2.1. 在 Java DSL 中使用 HawtDBAggregationRepository

在本例中，我们希望在 target/data/hawtdb.dat 文件中持久保留聚合的消息。

130.1.2.2. 在 Spring XML 中使用 HawtDBAggregationRepository

同一示例，但使用 Spring XML：

130.1.3. 依赖项

要在 camel 路由中使用 HawtDB，您需要添加对 camel-hawtdb 的依赖。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-hawtdb</artifactId>
  <version>2.3.0</version>
</dependency>
```

130.1.4. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*
- *聚合器*
- *组件*

第 131 章 HAZELCAST 组件

从 Camel 版本 2.7 开始提供

hazelcast- 组件允许您使用 Hazelcast 分布式数据网格/缓存。Hazelcast 是内存数据网格中的一个，完全使用 Java 编写（单一 jar）。它提供映射、多映射（与键、n 值）、队列、列表和原子数等不同数据存储的绝佳面板。使用 Hazelcast 的主要原因是其简单集群支持。如果您在网络上启用了多播，您可以运行有 hundred 节点的集群，且没有额外的配置。Hazelcast 只需配置为在节点间添加 n 个副本（默认为 1）、缓存持久性、网络配置（如果需要）、近缓存、信点等。如需更多信息，请参阅 http://www.hazelcast.com/docs.jsp 的 Hazelcast 文档。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-hazelcast</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

131.1. HAZELCAST 组件

请参阅以下每个组件用法：* map * multi map * queue * topic * list * seda * set * atomic number * cluster support (instance) * replicatedmap * ringbuffer

131.2. 使用 HAZELCAST 参考

131.2.1. 按名称

```
<bean id="hazelcastLifecycle" class="com.hazelcast.core.LifecycleService"
      factory-bean="hazelcastInstance" factory-method="getLifecycleService"
      destroy-method="shutdown" />

<bean id="config" class="com.hazelcast.config.Config">
  <constructor-arg type="java.lang.String" value="HZ.INSTANCE" />
</bean>

<bean id="hazelcastInstance" class="com.hazelcast.core.Hazelcast" factory-
method="newHazelcastInstance">
  <constructor-arg type="com.hazelcast.config.Config" ref="config"/>
</bean>
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <route id="testHazelcastInstanceBeanRefPut">
    <from uri="direct:testHazelcastInstanceBeanRefPut"/>
    <setHeader headerName="CamelHazelcastOperationType">
```

```
            <constant>put</constant>
        </setHeader>
        <to uri="hazelcast-map:testmap?hazelcastInstanceName=HZ.INSTANCE"/>
    </route>

    <route id="testHazelcastInstanceBeanRefGet">
        <from uri="direct:testHazelcastInstanceBeanRefGet" />
        <setHeader headerName="CamelHazelcastOperationType">
            <constant>get</constant>
        </setHeader>
        <to uri="hazelcast-map:testmap?hazelcastInstanceName=HZ.INSTANCE"/>
        <to uri="seda:out" />
    </route>
</camelContext>
```

131.2.2. 按实例

```
<bean id="hazelcastInstance" class="com.hazelcast.core.Hazelcast"
      factory-method="newHazelcastInstance" />
<bean id="hazelcastLifecycle" class="com.hazelcast.core.LifecycleService"
      factory-bean="hazelcastInstance" factory-method="getLifecycleService"
      destroy-method="shutdown" />

<camelContext xmlns="http://camel.apache.org/schema/spring">
    <route id="testHazelcastInstanceBeanRefPut">
        <from uri="direct:testHazelcastInstanceBeanRefPut"/>
        <setHeader headerName="CamelHazelcastOperationType">
            <constant>put</constant>
        </setHeader>
        <to uri="hazelcast-map:testmap?hazelcastInstance=#hazelcastInstance"/>
    </route>

    <route id="testHazelcastInstanceBeanRefGet">
        <from uri="direct:testHazelcastInstanceBeanRefGet" />
        <setHeader headerName="CamelHazelcastOperationType">
            <constant>get</constant>
        </setHeader>
        <to uri="hazelcast-map:testmap?hazelcastInstance=#hazelcastInstance"/>
        <to uri="seda:out" />
    </route>
</camelContext>
```

131.3. 将 HAZELCAST 实例作为 OSGI 服务发布

如果在 OSGI 容器中运行，并且您希望在同一容器中的所有捆绑包间使用一个 hazelcast 实例。您可以使用缓存要求将实例发布为 OSGI 服务和捆绑包，以引用 hazelcast 端点中的服务。

131.3.1. 捆绑包 A 创建实例并将其发布为 OSGI 服务

```
<bean id="config" class="com.hazelcast.config.FileSystemXmlConfig">
    <argument type="java.lang.String" value="${hazelcast.config}"/>
```

```
</bean>

<bean id="hazelcastInstance" class="com.hazelcast.core.Hazelcast" factory-
method="newHazelcastInstance">
    <argument type="com.hazelcast.config.Config" ref="config"/>
</bean>

<!-- publishing the hazelcastInstance as a service -->
<service ref="hazelcastInstance" interface="com.hazelcast.core.HazelcastInstance" />
```

131.3.2. 捆绑包 B 使用实例

```
<!-- referencing the hazelcastInstance as a service -->
<reference ref="hazelcastInstance" interface="com.hazelcast.core.HazelcastInstance" />

<camelContext xmlns="http://camel.apache.org/schema/blueprint">
    <route id="testHazelcastInstanceBeanRefPut">
        <from uri="direct:testHazelcastInstanceBeanRefPut"/>
        <setHeader headerName="CamelHazelcastOperationType">
            <constant>put</constant>
        </setHeader>
        <to uri="hazelcast-map:testmap?hazelcastInstance=#hazelcastInstance"/>
    </route>

    <route id="testHazelcastInstanceBeanRefGet">
        <from uri="direct:testHazelcastInstanceBeanRefGet" />
        <setHeader headerName="CamelHazelcastOperationType">
            <constant>get</constant>
        </setHeader>
        <to uri="hazelcast-map:testmap?hazelcastInstance=#hazelcastInstance"/>
        <to uri="seda:out" />
    </route>
</camelContext>
```

第 132 章 HAZELCAST ATOMIC NUMBER 组件

从 Camel 版本 2.7 开始提供

Hazelcast 原子数组件是 Camel Hazelcast 组件之一，允许您访问 Hazelcast 原子编号。原子数是一种只提供网格数量（长）的对象。

此端点没有消费者！

132.1. 选项

Hazelcast Atomic Number 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
hazelcastInstance (advanced)	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		HazelcastInstance
hazelcastMode (advanced)	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Hazelcast Atomic Number 端点使用 URI 语法进行配置：

```
hazelcast-atomicvalue:cacheName
```

使用以下路径和查询参数：

132.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	**所需的** 缓存的名称		字符串

132.1.2. 查询参数(10 parameters):

Name	描述	默认值	类型
reliable (common)	定义端点是否使用可靠的主题结构。	false	布尔值
defaultOperation (producer)	如果不提供操作标头，则指定要使用的默认操作。		HazelcastOperation
hazelcastInstance (producer)	hazelcast 实例引用，可用于 hazelcast 端点。		HazelcastInstance
hazelcastInstanceName (producer)	hazelcast 实例引用名称，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
concurrentConsumers (seda)	使用 SEDA 队列中的并发用户轮询。	1	int
onErrorDelay (seda)	发生错误后，消费者在消费者继续轮询前的时间（毫秒）。	1000	int
pollTimeout (seda)	从 SEDA 队列消耗时使用的超时。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让使用者在关闭时更快地做出反应。	1000	int
Transacted (seda)	如果设置为 true，则消费者以事务模式运行，则只有事务提交时，seda 队列中的消息才会被删除。	false	布尔值
transferExchange (seda)	如果设置为 true，则会传输整个 Exchange。如果标头或正文包含不序列化的对象，则会跳过它们。	false	布尔值

132.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
camel.component.hazelcast-atomicvalue.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值

Name	描述	默认值	类型
camel.component.hazelcast-atomicvalue.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component.hazelcast-atomicvalue.enabled	启用 hazelcast-atomicvalue 组件	true	布尔值
camel.component.hazelcast-atomicvalue.hazelcast-instance	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。选项是一个 com.hazelcast.core.HazelcastInstance 类型。		字符串
camel.component.hazelcast-atomicvalue.hazelcast-mode	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串
camel.component.hazelcast-atomicvalue.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

132.3. ATOMIC NUMBER PRODUCER - TO"HAZELCAST-ATOMICVALUE:FOO"）

此制作者的操作包括：* setvalue （使用给定值设置号）* get * increase (+1)* 减少(-1)* destroy

请求消息的标头变量：

Name	类型	描述
CamelHazelcastOperationType	**字符串**	有效值有：setvalue, get, increase, decrease, destroy

132.3.1. 设置 示例：

Java DSL:

```
from("direct:set")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.SET_VALUE))
.toF("hazelcast-%sfoo", HazelcastConstants.ATOMICNUMBER_PREFIX);
```

Spring DSL:

```
<route>
   <from uri="direct:set" />
      <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
   <setHeader headerName="hazelcast.operation.type">
      <constant>setvalue</constant>
   </setHeader>
   <to uri="hazelcast-atomicvalue:foo" />
</route>
```

提供要在消息正文中设置的值（其中值为 10）： template.sendBody ("direct:set", 10);

132.3.2. get 的示例：

Java DSL:

```
from("direct:get")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.GET))
.toF("hazelcast-%sfoo", HazelcastConstants.ATOMICNUMBER_PREFIX);
```

Spring DSL:

```
<route>
   <from uri="direct:get" />
      <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
   <setHeader headerName="hazelcast.operation.type">
      <constant>get</constant>
   </setHeader>
   <to uri="hazelcast-atomicvalue:foo" />
</route>
```

您可以使用 较长的 body = template.requestBody ("direct:get", null, Long.class) 获取数字，。

132.3.3. 递增 示例：

Java DSL:

```
from("direct:increment")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.INCREMENT))
.toF("hazelcast-%sfoo", HazelcastConstants.ATOMICNUMBER_PREFIX);
```

Spring DSL:

```
<route>
   <from uri="direct:increment" />
      <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
   <setHeader headerName="hazelcast.operation.type">
      <constant>increment</constant>
   </setHeader>
   <to uri="hazelcast-atomicvalue:foo" />
</route>
```

实际值（递增后）将在消息正文中提供。

132.3.4. 减少 示例：

Java DSL:

```
from("direct:decrement")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.DECREMENT))
.toF("hazelcast-%sfoo", HazelcastConstants.ATOMICNUMBER_PREFIX);
```

Spring DSL:

```
<route>
   <from uri="direct:decrement" />
      <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
   <setHeader headerName="hazelcast.operation.type">
      <constant>decrement</constant>
   </setHeader>
   <to uri="hazelcast-atomicvalue:foo" />
</route>
```

实际值（减少后）将在消息正文中提供。

132.3.5. destroy的示例

Java DSL:

```
from("direct:destroy")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.DESTROY))
.toF("hazelcast-%sfoo", HazelcastConstants.ATOMICNUMBER_PREFIX);
```

Spring DSL:

```
<route>
   <from uri="direct:destroy" />
      <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
   <setHeader headerName="hazelcast.operation.type">
      <constant>destroy</constant>
   </setHeader>
   <to uri="hazelcast-atomicvalue:foo" />
</route>
```

第 133 章 HAZELCAST 实例组件

从 Camel 版本 2.7 开始提供

Hazelcast 实例组件是 Camel Hazelcast 组件之一，可让您在集群中消耗缓存实例的 join/leave 事件。Hazelcast 在单一"服务器节点"中有意义，但在集群环境中非常强大。

此端点不提供任何制作者！

133.1. 选项

Hazelcast 实例组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
hazelcastInstance (advanced)	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		HazelcastInstance
hazelcastMode (advanced)	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Hazelcast 实例端点使用 URI 语法进行配置：

```
hazelcast-instance:cacheName
```

使用以下路径和查询参数：

133.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	**所需的** 缓存的名称		字符串

133.1.2. 查询参数(16 参数)：

Name	描述	默认值	类型
reliable (common)	定义端点是否使用可靠的主题结构。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
defaultOperation (consumer)	如果不提供操作标头，则指定要使用的默认操作。		HazelcastOperation
hazelcastInstance (consumer)	hazelcast 实例引用，可用于 hazelcast 端点。		HazelcastInstance
hazelcastInstanceName (consumer)	hazelcast 实例引用名称，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		字符串
pollingTimeout (consumer)	在 Poll 模式中定义 Queue consumer 的轮询超时	10000	long
poolSize (consumer)	定义 Queue Consumer Executor 的池大小	1	int
queueConsumerMode (consumer)	定义 Queue Consumer 模式： Listen 或 Poll	listen	HazelcastQueueConsumer Mode
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
concurrentConsumers (seda)	使用 SEDA 队列中的并发用户轮询。	1	int
onErrorDelay (seda)	发生错误后，消费者在消费者继续轮询前的时间（毫秒）。	1000	int

Name	描述	默认值	类型
pollTimeout (seda)	从 SEDA 队列消耗时使用的超时。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让使用者在关闭时更快地做出反应。	1000	int
Transact ed (seda)	如果设置为 true，则消费者以事务模式运行，则只有事务提交时，seda 队列中的消息才会被删除。	false	布尔值
transferExchange (seda)	如果设置为 true，则会传输整个 Exchange。如果标头或正文包含不序列化的对象，则会跳过它们。	false	布尔值

133.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 26 个选项，如下所列。

Name	描述	默认值	类型
camel.component .hazelcast-atomicvalue.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component .hazelcast-atomicvalue.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component .hazelcast-instance.enabled	启用 hazelcast-instance 组件	true	布尔值
camel.component .hazelcast-instance.hazelcast-instance	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。选项是一个 com.hazelcast.core.HazelcastInstance 类型。		字符串
camel.component .hazelcast-instance.hazelcast-mode	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串

Name	描述	默认值	类型
camel.component.hazelcast-instance.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.hazelcast-list.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component.hazelcast-list.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component.hazelcast-map.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component.hazelcast-map.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component.hazelcast-multimap.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component.hazelcast-multimap.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component.hazelcast-queue.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值

Name	描述	默认值	类型
camel.component.hazelcast-queue.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component.hazelcast-replicatedmap.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component.hazelcast-replicatedmap.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component.hazelcast-ringbuffer.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component.hazelcast-ringbuffer.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component.hazelcast-seda.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component.hazelcast-seda.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component.hazelcast-set.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值

Name	描述	默认值	类型
camel.component.hazelcast-set.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component.hazelcast-topic.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component.hazelcast-topic.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component.hazelcast-topic.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component.hazelcast-topic.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值

133.3. 实例消费者 - FROM ("HAZELCAST-INSTANCE:FOO")

> ***The instance consumer fires if a new cache instance will join or leave the cluster.***

下面是一个示例：

```
fromF("hazelcast-%sfoo", HazelcastConstants.INSTANCE_PREFIX)
.log("instance...")
.choice()

.when(header(HazelcastConstants.LISTENER_ACTION).isEqualTo(HazelcastConstants.ADDED))
        .log("...added")
        .to("mock:added")
```

```
    .otherwise()
        .log("...removed")
        .to("mock:removed");
```

每个事件在消息标头中提供以下信息：

响应消息中的标头变量：

Name	类型	描述
CamelHazelcastListenerTime	**Long**	millis 事件的时间
CamelHazelcastListenerType	**字符串**	映射消费者设置此处的 "instancelistener"
CamelHazelcastListenerAction	**字符串**	事件类型 - 这里 **添加或删除了 。**
CamelHazelcastInstanceHost	**字符串**	实例的主机名
CamelHazelcastInstancePort	**整数**	实例的端口号

第 134 章 HAZELCAST LIST 组件

从 Camel 版本 2.7 开始提供

Hazelcast List 组件是 Camel Hazelcast 组件之一，允许您访问 Hazelcast 分布式列表。

134.1. 选项

Hazelcast List 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
hazelcastInstance (advanced)	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		HazelcastInstance
hazelcastMode (advanced)	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Hazelcast List 端点使用 URI 语法进行配置：

```
hazelcast-list:cacheName
```

使用以下路径和查询参数：

134.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	**所需的** 缓存的名称		字符串

134.1.2. 查询参数(16 参数)：

Name	描述	默认值	类型
defaultOperation (common)	如果不提供操作标头，则指定要使用的默认操作。		HazelcastOperation
hazelcastInstance (common)	hazelcast 实例引用，可用于 hazelcast 端点。		HazelcastInstance
hazelcastInstanceName (common)	hazelcast 实例引用名称，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		字符串
reliable (common)	定义端点是否使用可靠的主题结构。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
pollingTimeout (consumer)	在 Poll 模式中定义 Queue consumer 的轮询超时	10000	long
poolSize (consumer)	定义 Queue Consumer Executor 的池大小	1	int
queueConsumerMode (consumer)	定义 Queue Consumer 模式： Listen 或 Poll	listen	HazelcastQueueConsumer Mode
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
concurrentConsumers (seda)	使用 SEDA 队列中的并发用户轮询。	1	int
onErrorDelay (seda)	发生错误后，消费者在消费者继续轮询前的时间（毫秒）。	1000	int

Name	描述	默认值	类型
pollTimeout (seda)	从 SEDA 队列消耗时使用的超时。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让使用者在关闭时更快地做出反应。	1000	int
Transact ed (seda)	如果设置为 true，则消费者以事务模式运行，则只有事务提交时，seda 队列中的消息才会被删除。	false	布尔值
transferExchange (seda)	如果设置为 true，则会传输整个 Exchange。如果标头或正文包含不序列化的对象，则会跳过它们。	false	布尔值

134.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
camel.component .hazelcast-list.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component .hazelcast-list.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component .hazelcast-list.enabled	启用 hazelcast-list 组件	true	布尔值
camel.component .hazelcast-list.hazelcast-instance	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。选项是一个 com.hazelcast.core.HazelcastInstance 类型。		字符串
camel.component .hazelcast-list.hazelcast-mode	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串

Name	描述	默认值	类型
camel.component.hazelcast-list.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

134.3. LIST PRODUCER - TO ("HAZELCAST-LIST:FOO")

list producer 提供 7 个操作： add * addAll * set * get * removevalue * removeAll * clear*

134.3.1. 添加 的示例：

```
from("direct:add")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.ADD))
.toF("hazelcast-%sbar", HazelcastConstants.LIST_PREFIX);
```

134.3.2. get 的示例：

```
from("direct:get")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.GET))
.toF("hazelcast-%sbar", HazelcastConstants.LIST_PREFIX)
.to("seda:out");
```

134.3.3. setvalue 示例：

```
from("direct:set")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.SET_VALUE))
.toF("hazelcast-%sbar", HazelcastConstants.LIST_PREFIX);
```

134.3.4. removevalue 示例：

```
from("direct:removevalue")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.REMOVE_VALUE))
.toF("hazelcast-%sbar", HazelcastConstants.LIST_PREFIX);
```

请注意，CamelHazelcastObjectIndex 标头用于索引目的。

134.4. LIST CONSUMER - FROM ("HAZELCAST-LIST:FOO")

列表消费者提供 2 个操作： add * remove*

```
fromF("hazelcast-%smm", HazelcastConstants.LIST_PREFIX)
   .log("object...")
   .choice()

.when(header(HazelcastConstants.LISTENER_ACTION).isEqualTo(HazelcastConstants.ADDE
D))
        .log("...added")
                .to("mock:added")

.when(header(HazelcastConstants.LISTENER_ACTION).isEqualTo(HazelcastConstants.REMO
VED))
        .log("...removed")
                .to("mock:removed")
           .otherwise()
                .log("fail!");
```

第 135 章 HAZELCAST MAP 组件

从 Camel 版本 2.7 开始提供

Hazelcast Map 组件是 Camel Hazelcast 组件之一，允许您访问 Hazelcast 分布式映射。

135.1. 选项

Hazelcast Map 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
hazelcastInstance (advanced)	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		HazelcastInstance
hazelcastMode (advanced)	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Hazelcast Map 端点使用 URI 语法进行配置：

```
hazelcast-map:cacheName
```

使用以下路径和查询参数：

135.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	**所需的** 缓存的名称		字符串

135.1.2. 查询参数(16 参数)：

Name	描述	默认值	类型
defaultOperation (common)	如果不提供操作标头，则指定要使用的默认操作。		HazelcastOperation
hazelcastInstance (common)	hazelcast 实例引用，可用于 hazelcast 端点。		HazelcastInstance
hazelcastInstanceName (common)	hazelcast 实例引用名称，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		字符串
reliable (common)	定义端点是否使用可靠的主题结构。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
pollingTimeout (consumer)	在 Poll 模式中定义 Queue consumer 的轮询超时	10000	long
poolSize (consumer)	定义 Queue Consumer Executor 的池大小	1	int
queueConsumerMode (consumer)	定义 Queue Consumer 模式： Listen 或 Poll	listen	HazelcastQueueConsumer Mode
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
concurrentConsumers (seda)	使用 SEDA 队列中的并发用户轮询。	1	int
onErrorDelay (seda)	发生错误后，消费者在消费者继续轮询前的时间（毫秒）。	1000	int

Name	描述	默认值	类型
pollTimeout (seda)	从 SEDA 队列消耗时使用的超时。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让使用者在关闭时更快地做出反应。	1000	int
Transact ed (seda)	如果设置为 true，则消费者以事务模式运行，则只有事务提交时，seda 队列中的消息才会被删除。	false	布尔值
transferExchange (seda)	如果设置为 true，则会传输整个 Exchange。如果标头或正文包含不序列化的对象，则会跳过它们。	false	布尔值

135.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
camel.component .hazelcast-map.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component .hazelcast-map.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component .hazelcast-map.enabled	启用 hazelcast-map 组件	true	布尔值
camel.component .hazelcast-map.hazelcast-instance	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。选项是一个 com.hazelcast.core.HazelcastInstance 类型。		字符串
camel.component .hazelcast-map.hazelcast-mode	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串

Name	描述	默认值	类型
camel.component.hazelcast-map.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

135.3. 映射 CACHE PRODUCER - TO ("HAZELCAST-MAP:FOO")

如果要在映射中存储值，您可以使用映射缓存制作者。

映射缓存制作者提供 CamelHazelcastOperationType 标头指定的后续操作：

- *put*
- *putIfAbsent*
- *get*
- *getAll*
- *keySet*
- *containsKey*
- *containsValue*
- *delete*
- *update*

- ***query***
- ***clear***
- ***Evict***
- ***evictAll***

所有操作都在 "hazelcast.operation.type" 头变量内提供。在 Java DSL 中，您可以使用 org.apache.camel.component.hazelcast.HazelcastOperation 中的常量。

请求消息的标头变量：

Name	类型	描述
CamelHazelcastOperationType	**字符串**	如前文所述。
CamelHazelcastObjectId	**字符串**	在缓存中存储 / 查找对象的对象 ID （查询操作不需要）

放置 和放置操作 提供驱除机制：

Name	类型	描述
CamelHazelcastObjectTtlValue	**整数**	TTL 值.

Name	类型	描述
CamelHazelcastObjectTtlUnit	**java.util.concurrent.TimeUnit**	时间单位（DAYS / HOURS / MINUTES / ....

您可以使用以下方法调用示例：

```
template.sendBodyAndHeader("direct:[put|get|update|delete|query|evict]", "my-foo",
HazelcastConstants.OBJECT_ID, "4711");
```

135.3.1. 放置 的示例：

Java DSL:

```
from("direct:put")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.PUT))
.toF("hazelcast-%sfoo", HazelcastConstants.MAP_PREFIX);
```

Spring DSL:

```
<route>
   <from uri="direct:put" />
      <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
   <setHeader headerName="hazelcast.operation.type">
      <constant>put</constant>
   </setHeader>
   <to uri="hazelcast-map:foo" />
</route>
```

使用驱除 放置 的示例：

Java DSL:

```
from("direct:put")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.PUT))
.setHeader(HazelcastConstants.TTL_VALUE, constant(Long.valueOf(1)))
.setHeader(HazelcastConstants.TTL_UNIT, constant(TimeUnit.MINUTES))
.toF("hazelcast-%sfoo", HazelcastConstants.MAP_PREFIX);
```

-

Spring DSL:

```
<route>
  <from uri="direct:put" />
    <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
  <setHeader headerName="hazelcast.operation.type">
    <constant>put</constant>
  </setHeader>
  <setHeader headerName="HazelcastConstants.TTL_VALUE">
    <simple resultType="java.lang.Long">1</simple>
  </setHeader>
  <setHeader headerName="HazelcastConstants.TTL_UNIT">
    <simple resultType="java.util.concurrent.TimeUnit">TimeUnit.MINUTES</simple>
  </setHeader>
  <to uri="hazelcast-map:foo" />
</route>
```

135.3.2. get 的示例 :

Java DSL:

```
from("direct:get")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.GET))
.toF("hazelcast-%sfoo", HazelcastConstants.MAP_PREFIX)
.to("seda:out");
```

Spring DSL:

```
<route>
  <from uri="direct:get" />
    <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
  <setHeader headerName="hazelcast.operation.type">
    <constant>get</constant>
  </setHeader>
  <to uri="hazelcast-map:foo" />
  <to uri="seda:out" />
</route>
```

135.3.3. 更新 示例 :

Java DSL:

```
from("direct:update")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.UPDATE))
.toF("hazelcast-%sfoo", HazelcastConstants.MAP_PREFIX);
```

Spring DSL:

```
<route>
   <from uri="direct:update" />
      <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
   <setHeader headerName="hazelcast.operation.type">
      <constant>update</constant>
   </setHeader>
   <to uri="hazelcast-map:foo" />
</route>
```

135.3.4. 删除 示例 :

Java DSL:

```
from("direct:delete")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.DELETE))
.toF("hazelcast-%sfoo", HazelcastConstants.MAP_PREFIX);
```

Spring DSL:

```
<route>
   <from uri="direct:delete" />
      <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
   <setHeader headerName="hazelcast.operation.type">
      <constant>delete</constant>
   </setHeader>
   <to uri="hazelcast-map:foo" />
</route>
```

135.3.5. 查询示例

Java DSL:

```
from("direct:query")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.QUERY))
.toF("hazelcast-%sfoo", HazelcastConstants.MAP_PREFIX)
.to("seda:out");
```

Spring DSL:

```
<route>
   <from uri="direct:query" />
      <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
   <setHeader headerName="hazelcast.operation.type">
      <constant>query</constant>
   </setHeader>
   <to uri="hazelcast-map:foo" />
   <to uri="seda:out" />
</route>
```

对于查询操作 Hazelcast，提供了类似语法的 SQL 来查询您的分布式映射。

```
String q1 = "bar > 1000";
template.sendBodyAndHeader("direct:query", null, HazelcastConstants.QUERY, q1);
```

135.4. 映射缓存消费者 - FROM ("HAZELCAST-MAP:FOO")

Hazelcast 提供其数据网格的事件监听程序。如果您要在操作缓存时获得通知，您可以使用映射消费者。有 4 个事件：放置、update、delete 和 envict。事件类型将存储在 "hazelcast.listener.action" header 变量中。映射使用者在这些变量中提供一些额外的信息：

响应消息中的标头变量：

Name	类型	描述
CamelHazelcastListenerTime	**Long**	millis 事件的时间
CamelHazelcastListenerType	**字符串**	映射消费者设置此处的 "cachelistener"

Name	类型	描述
CamelHazelcastListenerAction	**字符串**	事件类型 - 这里 **添加了**、**更新**、通知 **和删除** 的。
CamelHazelcastObjectId	**字符串**	对象没有
CamelHazelcastCacheName	**字符串**	缓存的名称 - 例如 "foo"
CamelHazelcastCacheType	**字符串**	cache - here map 的类型

对象值将存储在消息正文中的 放置 和更新 操作中。

下面是一个示例：

```
fromF("hazelcast-%sfoo", HazelcastConstants.MAP_PREFIX)
.log("object...")
.choice()

.when(header(HazelcastConstants.LISTENER_ACTION).isEqualTo(HazelcastConstants.ADDED))
        .log("...added")
        .to("mock:added")

.when(header(HazelcastConstants.LISTENER_ACTION).isEqualTo(HazelcastConstants.ENVICTED))
        .log("...envicted")
        .to("mock:envicted")

.when(header(HazelcastConstants.LISTENER_ACTION).isEqualTo(HazelcastConstants.UPDATED))
        .log("...updated")
        .to("mock:updated")

.when(header(HazelcastConstants.LISTENER_ACTION).isEqualTo(HazelcastConstants.REMO
```

```
VED))
        .log("...removed")
        .to("mock:removed")
    .otherwise()
        .log("fail!");
```

第 136 章 HAZELCAST MULTIMAP 组件

从 Camel 版本 2.7 开始提供

Hazelcast Multimap 组件是 Camel Hazelcast 组件之一，允许您访问 Hazelcast 分布式多映射。

136.1. 选项

Hazelcast Multimap 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
hazelcastInstance (advanced)	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		HazelcastInstance
hazelcastMode (advanced)	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Hazelcast Multimap 端点使用 URI 语法进行配置：

```
hazelcast-multimap:cacheName
```

使用以下路径和查询参数：

136.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	**所需的** 缓存的名称		字符串

136.1.2. 查询参数(16 参数)：

Name	描述	默认值	类型
defaultOperation (common)	如果不提供操作标头，则指定要使用的默认操作。		HazelcastOperation
hazelcastInstance (common)	hazelcast 实例引用，可用于 hazelcast 端点。		HazelcastInstance
hazelcastInstanceName (common)	hazelcast 实例引用名称，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		字符串
reliable (common)	定义端点是否使用可靠的主题结构。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
pollingTimeout (consumer)	在 Poll 模式中定义 Queue consumer 的轮询超时	10000	long
poolSize (consumer)	定义 Queue Consumer Executor 的池大小	1	int
queueConsumerMode (consumer)	定义 Queue Consumer 模式： Listen 或 Poll	listen	HazelcastQueueConsumer Mode
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
concurrentConsumers (seda)	使用 SEDA 队列中的并发用户轮询。	1	int
onErrorDelay (seda)	发生错误后，消费者在消费者继续轮询前的时间（毫秒）。	1000	int

Name	描述	默认值	类型
pollTimeout (seda)	从 SEDA 队列消耗时使用的超时。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让使用者在关闭时更快地做出反应。	1000	int
Transact ed (seda)	如果设置为 true，则消费者以事务模式运行，则只有事务提交时，seda 队列中的消息才会被删除。	false	布尔值
transferExchange (seda)	如果设置为 true，则会传输整个 Exchange。如果标头或正文包含不序列化的对象，则会跳过它们。	false	布尔值

136.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
camel.component .hazelcast-multimap.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component .hazelcast-multimap.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component .hazelcast-multimap.enabled	启用 hazelcast-multimap 组件	true	布尔值
camel.component .hazelcast-multimap.hazelcast-instance	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。选项是一个 com.hazelcast.core.HazelcastInstance 类型。		字符串
camel.component .hazelcast-multimap.hazelcast-mode	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串

Name	描述	默认值	类型
camel.component.hazelcast-multimap.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

136.3. MULTIMAP CACHE PRODUCER - TO"HAZELCAST-MULTIMAP:FOO"）

多映射是一个缓存，您可以将 n 个值存储到一个键。多图生成者提供 4 个操作(put, get, removevalue, delete)。

请求消息的标头变量：

Name	类型	描述
CamelHazelcastOperationType	**字符串**	有效值为： put, get, removevalue, delete **From Camel 2.16:** clear.
CamelHazelcastObjectId	**字符串**	在缓存中存储 / 的对象 ID

136.3.1. 放置 的示例：

Java DSL:

```
from("direct:put")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.PUT))
.to(String.format("hazelcast-%sbar", HazelcastConstants.MULTIMAP_PREFIX));
```

Spring DSL:

```
<route>
   <from uri="direct:put" />
   <log message="put.."/>
      <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
```

```
>
    <setHeader headerName="hazelcast.operation.type">
        <constant>put</constant>
    </setHeader>
    <to uri="hazelcast-multimap:foo" />
</route>
```

136.3.2. removevalue 示例：

Java DSL:

```
from("direct:removevalue")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.REMOVE_VALUE))
.toF("hazelcast-%sbar", HazelcastConstants.MULTIMAP_PREFIX);
```

Spring DSL:

```
<route>
    <from uri="direct:removevalue" />
    <log message="removevalue..."/>
        <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
    <setHeader headerName="hazelcast.operation.type">
        <constant>removevalue</constant>
    </setHeader>
    <to uri="hazelcast-multimap:foo" />
</route>
```

要删除一个值，您必须提供要在消息正文中删除的值。如果您有多映射对象 \{key: "4711" values: { "my-foo", "my-bar"}}，您必须在消息正文中放置 "my-foo" 值来删除 "my-foo" 值。

136.3.3. get 的示例：

Java DSL:

```
from("direct:get")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.GET))
.toF("hazelcast-%sbar", HazelcastConstants.MULTIMAP_PREFIX)
.to("seda:out");
```

Spring DSL:

```
<route>
```

```
    <from uri="direct:get" />
    <log message="get.."/>
       <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
    <setHeader headerName="hazelcast.operation.type">
       <constant>get</constant>
    </setHeader>
    <to uri="hazelcast-multimap:foo" />
    <to uri="seda:out" />
</route>
```

136.3.4. 删除 示例：

Java DSL:

```
from("direct:delete")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.DELETE))
.toF("hazelcast-%sbar", HazelcastConstants.MULTIMAP_PREFIX);
```

Spring DSL:

```
<route>
    <from uri="direct:delete" />
    <log message="delete.."/>
       <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
    <setHeader headerName="hazelcast.operation.type">
       <constant>delete</constant>
    </setHeader>
    <to uri="hazelcast-multimap:foo" />
</route>
```

您可以在测试类中调用它们：

```
template.sendBodyAndHeader("direct:[put|get|removevalue|delete]", "my-foo",
HazelcastConstants.OBJECT_ID, "4711");
```

136.4. MULTIMAP CACHE CONSUMER - FROM("HAZELCAST-MULTIMAP:FOO")

对于多图缓存，此组件提供与映射缓存消费者相同的监听程序 / 变量（更新和信封监听程序除外）。唯一的区别是 URI 中的 multimap 前缀。下面是一个示例：

```
fromF("hazelcast-%sbar", HazelcastConstants.MULTIMAP_PREFIX)
.log("object...")
.choice()
```

```
.when(header(HazelcastConstants.LISTENER_ACTION).isEqualTo(HazelcastConstants.ADDED))
        .log("...added")
            .to("mock:added")

//.when(header(HazelcastConstants.LISTENER_ACTION).isEqualTo(HazelcastConstants.ENVICTED))
        //      .log("...envicted")
        //      .to("mock:envicted")

.when(header(HazelcastConstants.LISTENER_ACTION).isEqualTo(HazelcastConstants.REMOVED))
            .log("...removed")
            .to("mock:removed")
        .otherwise()
            .log("fail!");
```

响应消息中的标头变量：

Name	类型	描述
CamelHazelcastListenerTime	**Long**	millis 事件的时间
CamelHazelcastListenerType	**字符串**	映射消费者设置此处的 "cachelistener"
CamelHazelcastListenerAction	**字符串**	事件类型 - 这里 **添加和删除** （并很快 为）
CamelHazelcastObjectId	**字符串**	对象没有
CamelHazelcastCacheName	**字符串**	缓存的名称 - 例如 "foo"

Name	类型	描述
CamelHazelcastCacheType	**字符串**	缓存的类型 - here multimap

第 137 章 HAZELCAST QUEUE 组件

从 Camel 版本 2.7 开始提供

Hazelcast Queue 组件是 Camel Hazelcast 组件之一，允许您访问 Hazelcast 分布式队列。

137.1. 选项

Hazelcast Queue 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
hazelcastInstance (advanced)	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		HazelcastInstance
hazelcastMode (advanced)	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Hazelcast Queue 端点使用 URI 语法进行配置：

```
hazelcast-queue:cacheName
```

使用以下路径和查询参数：

137.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	**所需的** 缓存的名称		字符串

137.1.2. 查询参数(16 参数)：

Name	描述	默认值	类型
defaultOperation (common)	如果不提供操作标头，则指定要使用的默认操作。		HazelcastOperation
hazelcastInstance (common)	hazelcast 实例引用，可用于 hazelcast 端点。		HazelcastInstance
hazelcastInstanceName (common)	hazelcast 实例引用名称，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		字符串
reliable (common)	定义端点是否使用可靠的主题结构。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
pollingTimeout (consumer)	在 Poll 模式中定义 Queue consumer 的轮询超时	10000	long
poolSize (consumer)	定义 Queue Consumer Executor 的池大小	1	int
queueConsumerMode (consumer)	定义 Queue Consumer 模式： Listen 或 Poll	listen	HazelcastQueueConsumer Mode
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
concurrentConsumers (seda)	使用 SEDA 队列中的并发用户轮询。	1	int
onErrorDelay (seda)	发生错误后，消费者在消费者继续轮询前的时间（毫秒）。	1000	int

Name	描述	默认值	类型
pollTimeout (seda)	从 SEDA 队列消耗时使用的超时。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让使用者在关闭时更快地做出反应。	1000	int
Transact ed (seda)	如果设置为 true，则消费者以事务模式运行，则只有事务提交时，seda 队列中的消息才会被删除。	false	布尔值
transferExchange (seda)	如果设置为 true，则会传输整个 Exchange。如果标头或正文包含不序列化的对象，则会跳过它们。	false	布尔值

137.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
camel.component .hazelcast-queue.customizer .hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component .hazelcast-queue.customizer .hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component .hazelcast-queue.enabled	启用 hazelcast-queue 组件	true	布尔值
camel.component .hazelcast-queue.hazelcast-instance	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。选项是一个 com.hazelcast.core.HazelcastInstance 类型。		字符串
camel.component .hazelcast-queue.hazelcast-mode	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串

Name	描述	默认值	类型
camel.component.hazelcast-queue.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

137.3. QUEUE PRODUCER - TO ("HAZELCAST-QUEUE:FOO")

queue producer 提供 10 个操作： add * put * poll * CUSTOM * ahead * remove value * remaining capacity * remove all * remove all * remove all if * drain to * retain all

137.3.1. 添加 的示例：

```
from("direct:add")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.ADD))
.toF("hazelcast-%sbar", HazelcastConstants.QUEUE_PREFIX);
```

137.3.2. 放置 的示例：

```
from("direct:put")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.PUT))
.toF("hazelcast-%sbar", HazelcastConstants.QUEUE_PREFIX);
```

137.3.3. 轮询 示例：

```
from("direct:poll")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.POLL))
.toF("hazelcast:%sbar", HazelcastConstants.QUEUE_PREFIX);
```

137.3.4. 示例 ：

```
from("direct:peek")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.PEEK))
.toF("hazelcast:%sbar", HazelcastConstants.QUEUE_PREFIX);
```

137.3.5. 提供 示例：

```
from("direct:offer")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.OFFER))
.toF("hazelcast:%sbar", HazelcastConstants.QUEUE_PREFIX);
```

137.3.6. removevalue 示例：

```
from("direct:removevalue")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.REMOVE_VALUE))
.toF("hazelcast-%sbar", HazelcastConstants.QUEUE_PREFIX);
```

137.3.7. 剩余容量 示例：

```
from("direct:remaining-capacity").setHeader(HazelcastConstants.OPERATION,
constant(HazelcastOperation.REMAINING_CAPACITY)).to(
String.format("hazelcast-%sbar", HazelcastConstants.QUEUE_PREFIX));
```

137.3.8. 删除所有 的示例：

```
from("direct:removeAll").setHeader(HazelcastConstants.OPERATION,
constant(HazelcastOperation.REMOVE_ALL)).to(
String.format("hazelcast-%sbar", HazelcastConstants.QUEUE_PREFIX));
```

137.3.9. 如果删除的示例 ：

```
from("direct:removeIf").setHeader(HazelcastConstants.OPERATION,
constant(HazelcastOperation.REMOVE_IF)).to(
String.format("hazelcast-%sbar", HazelcastConstants.QUEUE_PREFIX));
```

137.3.10. 排空至 的示例：

```
from("direct:drainTo").setHeader(HazelcastConstants.OPERATION,
constant(HazelcastOperation.DRAIN_TO)).to(
String.format("hazelcast-%sbar", HazelcastConstants.QUEUE_PREFIX));
```

137.3.11. 示例 获取 ：

```
from("direct:take").setHeader(HazelcastConstants.OPERATION,
constant(HazelcastOperation.TAKE)).to(
String.format("hazelcast-%sbar", HazelcastConstants.QUEUE_PREFIX));
```

137.3.12. 保留所有 的示例：

```
from("direct:retainAll").setHeader(HazelcastConstants.OPERATION,
constant(HazelcastOperation.RETAIN_ALL)).to(
String.format("hazelcast-%sbar", HazelcastConstants.QUEUE_PREFIX));
```

137.4. QUEUE CONSUMER - FROM ("HAZELCAST-QUEUE:FOO")

队列使用者提供两种不同的模式：

- *Poll*
- *listen*

Poll 模式示例

```
fromF("hazelcast-%sfoo?queueConsumerMode=Poll",
HazelcastConstants.QUEUE_PREFIX)).to("mock:result");
```

这样，消费者将轮询队列并在超时后返回队列或 null 的头。

在 Listen 模式中，使用者将侦听队列上的事件。

Listen 模式中的队列使用者提供 2 个操作： add * remove*

Listen 模式示例

```
fromF("hazelcast-%smm", HazelcastConstants.QUEUE_PREFIX)
  .log("object...")
  .choice()

.when(header(HazelcastConstants.LISTENER_ACTION).isEqualTo(HazelcastConstants.ADDED))
        .log("...added")
      .to("mock:added")

.when(header(HazelcastConstants.LISTENER_ACTION).isEqualTo(HazelcastConstants.REMOVED))
      .log("...removed")
      .to("mock:removed")
  .otherwise()
      .log("fail!");
```

第 138 章 HAZELCAST REPLICATED MAP 组件

从 Camel 版本 2.16 开始提供

Hazelcast 实例组件是 Camel Hazelcast 组件之一，可让您在集群中消耗缓存实例的 join/leave 事件。复制映射是一个弱一致性的分布式键值数据结构，没有数据分区。

138.1. 选项

Hazelcast Replicated Map 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
hazelcastInstance (advanced)	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		HazelcastInstance
hazelcastMode (advanced)	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Hazelcast Replicated Map 端点使用 URI 语法进行配置：

```
hazelcast-replicatedmap:cacheName
```

使用以下路径和查询参数：

138.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	**所需的** 缓存的名称		字符串

138.1.2. 查询参数(16 参数)：

Name	描述	默认值	类型
defaultOperation (common)	如果不提供操作标头，则指定要使用的默认操作。		HazelcastOperation
hazelcastInstance (common)	hazelcast 实例引用，可用于 hazelcast 端点。		HazelcastInstance
hazelcastInstanceName (common)	hazelcast 实例引用名称，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		字符串
reliable (common)	定义端点是否使用可靠的主题结构。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
pollingTimeout (consumer)	在 Poll 模式中定义 Queue consumer 的轮询超时	10000	long
poolSize (consumer)	定义 Queue Consumer Executor 的池大小	1	int
queueConsumerMode (consumer)	定义 Queue Consumer 模式： Listen 或 Poll	listen	HazelcastQueueConsumer Mode
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
concurrentConsumers (seda)	使用 SEDA 队列中的并发用户轮询。	1	int
onErrorDelay (seda)	发生错误后，消费者在消费者继续轮询前的时间（毫秒）。	1000	int

Name	描述	默认值	类型
pollTimeout (seda)	从 SEDA 队列消耗时使用的超时。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让使用者在关闭时更快地做出反应。	1000	int
Transact ed (seda)	如果设置为 true，则消费者以事务模式运行，则只有事务提交时，seda 队列中的消息才会被删除。	false	布尔值
transferExchange (seda)	如果设置为 true，则会传输整个 Exchange。如果标头或正文包含不序列化的对象，则会跳过它们。	false	布尔值

138.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
camel.component .hazelcast-replicatedmap.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component .hazelcast-replicatedmap.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component .hazelcast-replicatedmap.enabled	启用 hazelcast-replicatedmap 组件	true	布尔值
camel.component .hazelcast-replicatedmap.hazelcast-instance	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。选项是一个 com.hazelcast.core.HazelcastInstance 类型。		字符串
camel.component .hazelcast-replicatedmap.hazelcast-mode	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串

Name	描述	默认值	类型
camel.component.hazelcast-replicatedmap.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

138.3. REPLICATEDMAP 缓存制作者

replicatedmap producer 提供 4 个操作： put * get * delete * clear*

请求消息的标头变量：

Name	类型	描述
CamelHazelcastOperationType	**字符串**	有效值为： put, get, removevalue, delete
CamelHazelcastObjectId	**字符串**	在缓存中存储 / 的对象 ID

138.3.1. 放置 的示例：

Java DSL:

```
from("direct:put")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.PUT))
.to(String.format("hazelcast-%sbar", HazelcastConstants.REPLICATEDMAP_PREFIX));
```

Spring DSL:

```
<route>
    <from uri="direct:put" />
    <log message="put.."/>
        <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
```

```
    <setHeader headerName="hazelcast.operation.type">
      <constant>put</constant>
    </setHeader>
    <to uri="hazelcast-replicatedmap:foo" />
</route>
```

138.3.2. get 的示例 :

Java DSL:

```
from("direct:get")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.GET))
.toF("hazelcast-%sbar", HazelcastConstants.REPLICATEDMAP_PREFIX)
.to("seda:out");
```

Spring DSL:

```
<route>
    <from uri="direct:get" />
    <log message="get.."/>
      <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
    <setHeader headerName="hazelcast.operation.type">
      <constant>get</constant>
    </setHeader>
    <to uri="hazelcast-replicatedmap:foo" />
    <to uri="seda:out" />
</route>
```

138.3.3. 删除 示例 :

Java DSL:

```
from("direct:delete")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.DELETE))
.toF("hazelcast-%sbar", HazelcastConstants.REPLICATEDMAP_PREFIX);
```

Spring DSL:

```
<route>
    <from uri="direct:delete" />
    <log message="delete.."/>
      <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
    <setHeader headerName="hazelcast.operation.type">
```

```
        <constant>delete</constant>
    </setHeader>
    <to uri="hazelcast-replicatedmap:foo" />
</route>
```

您可以在测试类中调用它们：

```
template.sendBodyAndHeader("direct:[put|get|delete|clear]", "my-foo",
HazelcastConstants.OBJECT_ID, "4711");
```

138.4. REPLICATEDMAP 缓存消费者

对于多图缓存，此组件提供与映射缓存消费者相同的监听程序 / 变量（更新和信封监听程序除外）。唯一的区别是 URI 中的 multimap 前缀。下面是一个示例：

```
fromF("hazelcast-%sbar", HazelcastConstants.MULTIMAP_PREFIX)
.log("object...")
.choice()

.when(header(HazelcastConstants.LISTENER_ACTION).isEqualTo(HazelcastConstants.ADDED))
        .log("...added")
            .to("mock:added")

//.when(header(HazelcastConstants.LISTENER_ACTION).isEqualTo(HazelcastConstants.ENVICTED))
        //      .log("...envicted")
        //      .to("mock:envicted")

.when(header(HazelcastConstants.LISTENER_ACTION).isEqualTo(HazelcastConstants.REMOVED))
            .log("...removed")
            .to("mock:removed")
        .otherwise()
            .log("fail!");
```

响应消息中的标头变量：

Name	类型	描述
CamelHazelcastListenerTime	**Long**	millis 事件的时间

Name	类型	描述
CamelHazelcastListenerType	**字符串**	映射消费者设置此处的 "cachelistener"
CamelHazelcastListenerAction	**字符串**	事件类型 - 这里 **添加和删除** （并很快 为）
CamelHazelcastObjectId	**字符串**	对象没有
CamelHazelcastCacheName	**字符串**	缓存的名称 - 例如 "foo"
CamelHazelcastCacheType	**字符串**	cache - here replicatedmap 的类型

第 139 章 HAZELCAST RINGBUFFER 组件

从 Camel 版本 2.16 开始提供

可以被 Camel 2.16 访问

Hazelcast 环缓冲组件是 Camel Hazelcast 组件之一，允许您访问 Hazelcast 环缓冲。Ringbuffer 是一个分布式数据结构，其中数据存储在类似环形结构中。您可以将它视为具有特定容量的圆形数组。

139.1. 选项

Hazelcast Ringbuffer 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
hazelcastInstance (advanced)	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		HazelcastInstance
hazelcastMode (advanced)	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Hazelcast Ringbuffer 端点使用 URI 语法进行配置：

```
hazelcast-ringbuffer:cacheName
```

使用以下路径和查询参数：

139.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	**所需的** 缓存的名称		字符串

139.1.2. 查询参数(10 parameters):

Name	描述	默认值	类型
reliable (common)	定义端点是否使用可靠的主题结构。	false	布尔值
defaultOperation (producer)	如果不提供操作标头，则指定要使用的默认操作。		HazelcastOperation
hazelcastInstance (producer)	hazelcast 实例引用，可用于 hazelcast 端点。		HazelcastInstance
hazelcastInstanceName (producer)	hazelcast 实例引用名称，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
concurrentConsumers (seda)	使用 SEDA 队列中的并发用户轮询。	1	int
onErrorDelay (seda)	发生错误后，消费者在消费者继续轮询前的时间（毫秒）。	1000	int
pollTimeout (seda)	从 SEDA 队列消耗时使用的超时。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让使用者在关闭时更快地做出反应。	1000	int
Transacted (seda)	如果设置为 true，则消费者以事务模式运行，则只有事务提交时，seda 队列中的消息才会被删除。	false	布尔值
transferExchange (seda)	如果设置为 true，则会传输整个 Exchange。如果标头或正文包含不序列化的对象，则会跳过它们。	false	布尔值

139.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
camel.component.hazelcast-ringbuffer.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值

Name	描述	默认值	类型
camel.component.hazelcast-ringbuffer.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component.hazelcast-ringbuffer.enabled	启用 hazelcast-wagonfer 组件	true	布尔值
camel.component.hazelcast-ringbuffer.hazelcast-instance	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。选项是一个 com.hazelcast.core.HazelcastInstance 类型。		字符串
camel.component.hazelcast-ringbuffer.hazelcast-mode	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串
camel.component.hazelcast-ringbuffer.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

139.3. RINGBUFFER 缓存制作者

ringbuffer producer 提供 5 个操作：* add * readonceHead * readonceTail * remainingCapacity * capacity

请求消息的标头变量：

Name	类型	描述
CamelHazelcastOperationType	**字符串**	有效值为： put, get, removevalue, delete

Name	类型	描述
CamelHazelcastObjectId	**字符串**	在缓存中存储 / 的对象 ID

139.3.1. 放置 的示例：

Java DSL:

```
from("direct:put")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.ADD))
.to(String.format("hazelcast-%sbar", HazelcastConstants.RINGBUFFER_PREFIX));
```

Spring DSL:

```
<route>
   <from uri="direct:put" />
   <log message="put.."/>
      <!-- If using version 2.8 and above set headerName to "CamelHazelcastOperationType" --
>
   <setHeader headerName="hazelcast.operation.type">
      <constant>add</constant>
   </setHeader>
   <to uri="hazelcast-ringbuffer:foo" />
</route>
```

139.3.2. 来自头读概念 的示例：

Java DSL:

```
from("direct:get")
.setHeader(HazelcastConstants.OPERATION,
constant(HazelcastOperation.READ_ONCE_HEAD))
.toF("hazelcast-%sbar", HazelcastConstants.RINGBUFFER_PREFIX)
.to("seda:out");
```

第 140 章 HAZELCAST SEDA 组件

从 Camel 版本 2.7 开始提供

Hazelcast SEDA 组件是 Camel Hazelcast 组件之一，允许您访问 Hazelcast BlockingQueue。SEDA 组件与提供的其他组件不同。它实现了一个 work-queue，以支持异步 SEDA 架构，类似于核心"SEDA"组件。

140.1. 选项

Hazelcast SEDA 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
hazelcastInstance (advanced)	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		HazelcastInstance
hazelcastMode (advanced)	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Hazelcast SEDA 端点使用 URI 语法进行配置：

```
hazelcast-seda:cacheName
```

使用以下路径和查询参数：

140.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	**所需的** 缓存的名称		字符串

140.1.2. 查询参数(16 参数)：

Name	描述	默认值	类型
defaultOperation (common)	如果不提供操作标头，则指定要使用的默认操作。		HazelcastOperation
hazelcastInstance (common)	hazelcast 实例引用，可用于 hazelcast 端点。		HazelcastInstance
hazelcastInstanceName (common)	hazelcast 实例引用名称，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		字符串
reliable (common)	定义端点是否使用可靠的主题结构。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
pollingTimeout (consumer)	在 Poll 模式中定义 Queue consumer 的轮询超时	10000	long
poolSize (consumer)	定义 Queue Consumer Executor 的池大小	1	int
queueConsumerMode (consumer)	定义 Queue Consumer 模式： Listen 或 Poll	listen	HazelcastQueueConsumer Mode
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
concurrentConsumers (seda)	使用 SEDA 队列中的并发用户轮询。	1	int
onErrorDelay (seda)	发生错误后，消费者在消费者继续轮询前的时间（毫秒）。	1000	int

Name	描述	默认值	类型
pollTimeout (seda)	从 SEDA 队列消耗时使用的超时。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让使用者在关闭时更快地做出反应。	1000	int
Transact ed (seda)	如果设置为 true，则消费者以事务模式运行，则只有事务提交时，seda 队列中的消息才会被删除。	false	布尔值
transferExchange (seda)	如果设置为 true，则会传输整个 Exchange。如果标头或正文包含不序列化的对象，则会跳过它们。	false	布尔值

140.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
camel.component .hazelcast-seda.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component .hazelcast-seda.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component .hazelcast-seda.enabled	启用 hazelcast-seda 组件	true	布尔值
camel.component .hazelcast-seda.hazelcast-instance	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。选项是一个 com.hazelcast.core.HazelcastInstance 类型。		字符串
camel.component .hazelcast-seda.hazelcast-mode	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串

Name	描述	默认值	类型
camel.component.hazelcast-seda.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

140.3. SEDA PRODUCER - TO"HAZELCAST-SEDA:FOO"）

SEDA 生成者不提供任何操作。您仅将数据发送到指定的队列。

Java DSL：

```
from("direct:foo")
.to("hazelcast-seda:foo");
```

Spring DSL:

```
<route>
  <from uri="direct:start" />
  <to uri="hazelcast-seda:foo" />
</route>
```

140.4. SEDA CONSUMER - FROM ("HAZELCAST-SEDA:FOO")

SEDA 使用者不提供任何操作。您仅从指定的队列检索数据。

Java DSL：

```
from("hazelcast-seda:foo")
.to("mock:result");
```

Spring DSL:

```
<route>
  <from uri="hazelcast-seda:foo" />
```

```
  <to uri="mock:result" />
</route>
```

第 141 章 HAZELCAST 设置组件

从 Camel 版本 2.7 开始提供

Hazelcast Set 组件是 Camel Hazelcast 组件之一，允许您访问 Hazelcast 分布式集合。

141.1. 选项

Hazelcast Set 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
hazelcastInstance (advanced)	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		HazelcastInstance
hazelcastMode (advanced)	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Hazelcast Set 端点使用 URI 语法进行配置：

```
hazelcast-set:cacheName
```

使用以下路径和查询参数：

141.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	**所需的** 缓存的名称		字符串

141.1.2. 查询参数(16 参数)：

Name	描述	默认值	类型
defaultOperation (common)	如果不提供操作标头，则指定要使用的默认操作。		HazelcastOperation
hazelcastInstance (common)	hazelcast 实例引用，可用于 hazelcast 端点。		HazelcastInstance
hazelcastInstanceName (common)	hazelcast 实例引用名称，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		字符串
reliable (common)	定义端点是否使用可靠的主题结构。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
pollingTimeout (consumer)	在 Poll 模式中定义 Queue consumer 的轮询超时	10000	long
poolSize (consumer)	定义 Queue Consumer Executor 的池大小	1	int
queueConsumerMode (consumer)	定义 Queue Consumer 模式： Listen 或 Poll	listen	HazelcastQueueConsumer Mode
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
concurrentConsumers (seda)	使用 SEDA 队列中的并发用户轮询。	1	int
onErrorDelay (seda)	发生错误后，消费者在消费者继续轮询前的时间（毫秒）。	1000	int

Name	描述	默认值	类型
pollTimeout (seda)	从 SEDA 队列消耗时使用的超时。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让使用者在关闭时更快地做出反应。	1000	int
Transact ed (seda)	如果设置为 true，则消费者以事务模式运行，则只有事务提交时，seda 队列中的消息才会被删除。	false	布尔值
transferExchange (seda)	如果设置为 true，则会传输整个 Exchange。如果标头或正文包含不序列化的对象，则会跳过它们。	false	布尔值

141.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
camel.component .hazelcast-set.customizer.hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component .hazelcast-set.customizer.hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component .hazelcast-set.enabled	启用 hazelcast-set 组件	true	布尔值
camel.component .hazelcast-set.hazelcast-instance	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。选项是一个 com.hazelcast.core.HazelcastInstance 类型。		字符串
camel.component .hazelcast-set.hazelcast-mode	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串

Name	描述	默认值	类型
camel.component.hazelcast-set.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 142 章 HAZELCAST 主题组件

从 Camel 版本 2.15 开始提供

Hazelcast 主题组件是 Camel Hazelcast 组件之一，允许您访问 Hazelcast 分布式主题。

142.1. 选项

Hazelcast 主题组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
hazelcastInstance (advanced)	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		HazelcastInstance
hazelcastMode (advanced)	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Hazelcast 主题端点使用 URI 语法进行配置：

```
hazelcast-topic:cacheName
```

使用以下路径和查询参数：

142.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	**所需的** 缓存的名称		字符串

142.1.2. 查询参数(16 参数)：

Name	描述	默认值	类型
defaultOperation (common)	如果不提供操作标头，则指定要使用的默认操作。		HazelcastOperation
hazelcastInstance (common)	hazelcast 实例引用，可用于 hazelcast 端点。		HazelcastInstance
hazelcastInstanceName (common)	hazelcast 实例引用名称，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。		字符串
reliable (common)	定义端点是否使用可靠的主题结构。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
pollingTimeout (consumer)	在 Poll 模式中定义 Queue consumer 的轮询超时	10000	long
poolSize (consumer)	定义 Queue Consumer Executor 的池大小	1	int
queueConsumerMode (consumer)	定义 Queue Consumer 模式： Listen 或 Poll	listen	HazelcastQueueConsumer Mode
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
concurrentConsumers (seda)	使用 SEDA 队列中的并发用户轮询。	1	int
onErrorDelay (seda)	发生错误后，消费者在消费者继续轮询前的时间（毫秒）。	1000	int

Name	描述	默认值	类型
pollTimeout (seda)	从 SEDA 队列消耗时使用的超时。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让使用者在关闭时更快地做出反应。	1000	int
Transact ed (seda)	如果设置为 true，则消费者以事务模式运行，则只有事务提交时，seda 队列中的消息才会被删除。	false	布尔值
transferExchange (seda)	如果设置为 true，则会传输整个 Exchange。如果标头或正文包含不序列化的对象，则会跳过它们。	false	布尔值

142.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 8 个选项，如下所列。

Name	描述	默认值	类型
camel.component .hazelcast-topic.customizer. hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component .hazelcast-topic.customizer. hazelcast-instance.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component .hazelcast-topic.customizer. hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component .hazelcast-topic.customizer. hazelcast-instance.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component .hazelcast-topic.enabled	启用 hazelcast-topic 组件	true	布尔值

Name	描述	默认值	类型
camel.component.hazelcast-topic.hazelcast-instance	hazelcast 实例引用，可用于 hazelcast 端点。如果您没有指定实例引用，camel 将使用 camel-hazelcast 实例中的默认 hazelcast 实例。选项是一个 com.hazelcast.core.HazelcastInstance 类型。		字符串
camel.component.hazelcast-topic.hazelcast-mode	应使用 hazelcast 模式引用的实例类型。如果没有指定模式，节点模式将为默认模式。	node	字符串
camel.component.hazelcast-topic.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

142.3. TOPIC PRODUCER - TO ("HAZELCAST-TOPIC:FOO")

主题生成者仅提供一个操作（发布）。

142.3.1. 发布 示例：

```
from("direct:add")
.setHeader(HazelcastConstants.OPERATION, constant(HazelcastOperation.PUBLISH))
.toF("hazelcast-%sbar", HazelcastConstants.PUBLISH_OPERATION);
```

142.4. 主题 CONSUMER - FROM ("HAZELCAST-TOPIC:FOO")

主题使用者仅提供一个操作（接收）。这个组件应该会象涉及主题时支持多个消耗，因此您可以自由地拥有同一 hazelcast 主题所需的消费者。

```
fromF("hazelcast-%sfoo", HazelcastConstants.TOPIC_PREFIX)
  .choice()

.when(header(HazelcastConstants.LISTENER_ACTION).isEqualTo(HazelcastConstants.RECEIVED))
      .log("...message received")
   .otherwise()
      .log("...this should never have happened")
```

第 143 章 HBASE 组件

从 Camel 版本 2.10 开始提供

此组件为 Apache HBase 提供 idemptotent 存储库、生成者和消费者。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-hbase</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

143.1. APACHE HBASE 概述

HBase 是一个开源、分布式、版本化的、面向列的存储模型，在 Google 的 Bigtable 后建模：用于结构化数据的分布式存储系统。当您需要随机的、实时读/写访问您的大数据时，您可以使用 HBase。如需更多信息，请参阅 Apache HBase。

143.2. CAMEL 和 HBASE

在 camel 路由中使用 datasotre 时，始终存在指定 camel 消息将存储到数据存储中的 chalenge。在基于文档的存储方面，消息正文可以直接映射到文档。在关系数据库中，ORM 解决方案可用于将属性映射到列等。在基于列中的存储方面更具挑战性，因为没有执行这种映射的标准方法。

HBase 添加了两个额外的挑战：

- HBase 组列到系列中，因此只需使用名称惯例将属性映射到列则不够。
- HBase 没有类型的概念，这意味着它将所有内容存储为 byte[]，并且不知道 byte[] 代表 String、数字、序列化 Java 对象或二进制数据。

为克服这些挑战，camel-hbase 使用消息标头来指定消息到 HBase 列的映射。它还能够使用一些 camel-hbase 提供的类来建模 HBase 数据，并可轻松转换为 xml/json 等。

最后，它允许用户实施和使用自己的映射策略。

无论映射策略 camel-hbase 都将消息转换为 org.apache.camel.component.hbase.model.HBaseData 对象，并将该对象用于其内部操作。

143.3. 配置组件

HBase 组件可以作为属性提供自定义 HBaseConfiguration 对象，或者可以根据在 classpath 上找到的 HBase 相关资源自行创建 HBase 配置对象。

```
<bean id="hbase" class="org.apache.camel.component.hbase.HBaseComponent">
    <property name="configuration" ref="config"/>
</bean>
```

如果没有为组件提供配置对象，则组件会创建一个。创建的配置将搜索 hbase-site.xml 文件的类路径，该文件将从中提取配置。您可以找到有关如何配置 HBase 客户端的更多信息，请访问：HBase 客户端配置和依赖项

143.4. HBASE PRODUCER

如上所述，camel 为 HBase 提供 produers 端点。这可让您使用 camel 路由从 HBase 存储、删除、检索或查询数据。

```
hbase://table[?options]
```

其中 table 是表名称。

支持的操作有：

- put
- Get
- 删除

- *扫描*

143.4.1. 支持的 URI 选项

HBase 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	使用共享配置		配置
poolMaxSize (common)	HTable 池中为每个表保留的最大引用数。默认值为 10。	10	int
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

HBase 端点使用 URI 语法进行配置：

```
hbase:tableName
```

使用以下路径和查询参数：

143.4.2. 路径参数(1 参数)：

Name	描述	默认值	类型
tableName	**必需** 表的名称		字符串

143.4.3. 查询参数(16 参数)：

Name	描述	默认值	类型
cellMappingStrategyFactory (common)	使用负责映射单元的自定义 CellMappingStrategyFactory。		CellMappingStrategy Factory
filters (common)	要使用的过滤器列表。		list

Name	描述	默认值	类型
mappingStrategyClassName (common)	自定义映射策略实施的类名称。		字符串
mappingStrategyName (common)	用于将 Camel 消息映射到 HBase 列的策略。支持的值有： header 或 body。		字符串
rowMapping (common)	将 key/values 从 Map 映射到 HBaseRow。支持以下键： rowId - 行的 id。这有有限的使用，因为通常更改每个 Exchange 的行。rowType - type to covert row id to.支持的操作：CamelHBaseScan. family - 列系列。支持引用多个列的数字后缀。限定符 - 列限定符。支持引用多个列的数字后缀。value - 值。支持引用多个列 valueType - 值类型的数字后缀。支持引用多个列的数字后缀。支持的操作：CamelHBaseGet 和 CamelHBaseScan。		Map
rowModel (common)	org.apache.camel.component.hbase.model.HBaseRow 的实例，它描述了每行如何建模		HBaseRow
userGroupInformation (common)	定义与 HBase 进行通信的特权，如使用 kerberos。		UserGroupInformation
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
maxMessagesPerPoll (consumer)	获取最大消息数，作为每次轮询轮询的限制。默认为无限限制，但使用 0 或负数来禁用它。		int
operation (consumer)	执行的 HBase 操作		字符串
remove (consumer)	如果选项为 true，则 Camel HBase Consumer 将删除其进程所在的行。	true	布尔值
removeHandler (consumer)	要使用在要删除的行时执行的自定义 HBaseRemoveHandler。		HBaseRemoveHandler
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler

Name	描述	默认值	类型
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
maxResults (producer)	要扫描的最大行数。	100	int
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

143.5. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.hbase.configuration	使用共享配置。选项是 org.apache.hadoop.conf.Configuration 类型。		字符串
camel.component.hbase.enabled	启用 hbase 组件	true	布尔值
camel.component.hbase.pool-max-size	HTable 池中为每个表保留的最大引用数。默认值为 10。	10	整数
camel.component.hbase.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

143.5.1. Put Operations （放置操作）

HBase 是基于列的存储，允许您将数据存储在特定行的特定列中。列分组为系列，因此要指定一个列，您需要指定列系列和该列的限定符。要将数据存储到特定的列中，您需要同时指定列和行。

将数据存储在 HBase 中的 camel 路由中的最简单场景是将消息正文的一部分存储到指定的 HBase 列。

```
    <route>
      <from uri="direct:in"/>
```

```
        <!-- Set the HBase Row -->
        <setHeader headerName="CamelHBaseRowId">
          <el>${in.body.id}</el>
        </setHeader>
        <!-- Set the HBase Value -->
        <setHeader headerName="CamelHBaseValue">
          <el>${in.body.value}</el>
        </setHeader>
        <to uri="hbase:mytable?
operation=CamelHBasePut&amp;family=myfamily&amp;qualifier=myqualifier"/>
      </route>
```

上面的路由假定消息正文包含一个具有 id 和 value 属性的对象，并将值的内容存储在 id 指定的行中的 HBase 列 myfamily:myqualifier 中。如果我们需要指定多个列/值对，我们可以仅指定额外的列映射。请注意，您必须使用结果上的第 2 个标头中的数字，如 RowId2、RowId3、RowId4 等。只有第一代标题没有数字 1。

```
      <route>
        <from uri="direct:in"/>
        <!-- Set the HBase Row 1st column -->
        <setHeader headerName="CamelHBaseRowId">
          <el>${in.body.id}</el>
        </setHeader>
        <!-- Set the HBase Row 2nd column -->
        <setHeader headerName="CamelHBaseRowId2">
          <el>${in.body.id}</el>
        </setHeader>
        <!-- Set the HBase Value for 1st column -->
        <setHeader headerName="CamelHBaseValue">
          <el>${in.body.value}</el>
        </setHeader>
        <!-- Set the HBase Value for 2nd column -->
        <setHeader headerName="CamelHBaseValue2">
          <el>${in.body.othervalue}</el>
        </setHeader>
        <to uri="hbase:mytable?
operation=CamelHBasePut&amp;family=myfamily&amp;qualifier=myqualifier&amp;family2=myfamily&a
p;qualifier2=myqualifier2"/>
      </route>
```

务必要记住，您可以使用 uri 选项、消息标头或这两者的组合。建议将常量指定为 uri 和 dynamic 值的一部分，作为标头的一部分。如果某个项同时定义为标头和作为 uri 的一部分，则将使用标头。

143.5.2. 获取运营.

Get Operation 是一个操作，用于从指定的 HBase 行中检索一个或多个值。要指定您要检索的值，您只需将它们指定为 uri 的一部分，或作为消息标头的一部分。

```
      <route>
```

```
        <from uri="direct:in"/>
        <!-- Set the HBase Row of the Get -->
        <setHeader headerName="CamelHBaseRowId">
          <el>${in.body.id}</el>
        </setHeader>
        <to uri="hbase:mytable?
operation=CamelHBaseGet&amp;family=myfamily&amp;qualifier=myqualifier&amp;valueType=java.lang
Long"/>
        <to uri="log:out"/>
      </route>
```

在上例中，get 操作的结果将存储为名为 CamelHBaseValue 的标头。

143.5.3. 删除操作.

您还可以 camel-hbase 来执行 HBase delete 操作。删除操作将删除一行。所有需要指定的一行或多行作为消息标头的一部分。

```
      <route>
        <from uri="direct:in"/>
        <!-- Set the HBase Row of the Get -->
        <setHeader headerName="CamelHBaseRowId">
          <el>${in.body.id}</el>
        </setHeader>
        <to uri="hbase:mytable?operation=CamelHBaseDelete"/>
      </route>
```

143.5.4. 扫描操作.

扫描操作等同于 HBase 中的查询。您可以使用扫描操作来检索多行。要指定哪些列应该是结果的一部分，并指定如何将值转换为对象，您可以使用 uri 选项或标头。

```
      <route>
        <from uri="direct:in"/>
        <to uri="hbase:mytable?
operation=CamelHBaseScan&amp;family=myfamily&amp;qualifier=myqualifier&amp;valueType=java.la
g.Long&amp;rowType=java.lang.String"/>
        <to uri="log:out"/>
      </route>
```

在这种情况下，您还需要指定一个过滤器列表来限制结果。您可以将过滤器列表指定为 uri 的一部分，camel 只返回满足 ALL 过滤器的行。
要使过滤器可以了解消息一部分的信息，camel 定义了 ModelAwareFilter。这样，您的过滤器可以考虑消息和映射策略定义的模型。
当使用 ModelAwareFilter camel-hbase 时，会将所选映射策略应用到 in 消息，会创建一个模型映射的对象，并将该对象传递给 Filter。

例如，要使用 作为消息标头的条件来执行扫描，您可以使用 ModelAwareColumnMatchingFilter，如下所示。

```
<route>
    <from uri="direct:scan"/>
    <!-- Set the Criteria -->
    <setHeader headerName="CamelHBaseFamily">
        <constant>name</constant>
    </setHeader>
    <setHeader headerName="CamelHBaseQualifier">
        <constant>first</constant>
    </setHeader>
    <setHeader headerName="CamelHBaseValue">
        <el>in.body.firstName</el>
    </setHeader>
    <setHeader headerName="CamelHBaseFamily2">
        <constant>name</constant>
    </setHeader>
    <setHeader headerName="CamelHBaseQualifier2">
        <constant>last</constant>
    </setHeader>
    <setHeader headerName="CamelHBaseValue2">
        <el>in.body.lastName</el>
    </setHeader>
    <!-- Set additional fields that you want to be return by skipping value -->
    <setHeader headerName="CamelHBaseFamily3">
        <constant>address</constant>
    </setHeader>
    <setHeader headerName="CamelHBaseQualifier3">
        <constant>country</constant>
    </setHeader>
    <to uri="hbase:mytable?operation=CamelHBaseScan&amp;filters=#myFilterList"/>
</route>

<bean id="myFilters" class="java.util.ArrayList">
    <constructor-arg>
        <list>
            <bean
class="org.apache.camel.component.hbase.filters.ModelAwareColumnMatchingFilter"/>
        </list>
    </constructor-arg>
</bean>
```

上面的路由假定 pojo 具有属性 firstName 和 lastName 作为消息正文传递，它会采用这些属性，并将它们添加为消息标头的一部分。默认映射策略会创建一个模型对象，它将把标头映射到 HBase 列，并传递模型 ModelAwareColumnMatchingFilter。过滤器将过滤掉任何行，它们不包含与模型匹配的列。类似于通过 example 的查询。

143.6. HBASE CONSUMER

Camel HBase Consumer 将对指定的 HBase 表执行重复扫描，并将返回扫描结果作为消息的一部分。您可以指定标头映射（默认）或正文映射。稍后，将仅添加 org.apache.camel.component.hbase.model.HBaseData 作为消息正文的一部分。

```
hbase://table[?options]
```

您可以将要返回的列及其类型指定为 uri 选项的一部分：

```
hbase:mutable?
family=name&qualifer=first&valueType=java.lang.String&family=address&qualifer=number&v
alueType2=java.lang.Integer&rowType=java.lang.Long
```

上面的示例将创建一个由指定字段组成的模型对象，扫描结果将使用值填充 model 对象。最后，映射策略将用于将此模型映射到 camel 消息。

143.7. HBASE IDEMPOTENT 存储库

camel-hbase 组件还提供幂等存储库，可在您要确保仅处理一次时使用这些存储库。HBase idempotent 存储库配置有表、列系列和一个列限定符，并将为每个消息创建一个行。

```
HBaseConfiguration configuration = HBaseConfiguration.create();
HBaseIdempotentRepository repository = new HBaseIdempotentRepository(configuration,
tableName, family, qualifier);

from("direct:in")
  .idempotentConsumer(header("messageId"), repository)
  .to("log:out);
```

143.8. HBASE MAPPING

上面提到了默认的映射策略是 标头和正文 映射。
您可以在下面找到有关每个映射策略的工作方式的详细示例。

143.8.1. HBase Header 映射示例

标头映射是默认的映射。将值 "myvalue" 放入 HBase row "myrow"，列 "myfamily:mycolum"，消息应包含以下标头：

标头	value
Camel HBase RowId	myrow
Camel HBase Family	myfamily
Camel HBase Qualifier	myqualifier
Camel HBase Value	myValue

要为不同的列和 / 或不同的行放置更多值，您可以使用标头的索引指定附加标头后缀，例如：

标头	value
Camel HBase RowId	myrow
Camel HBase Family	myfamily
Camel HBase Qualifier	myqualifier
Camel HBase Value	myValue
Camel HBase RowId2	myrow2

标头	value
CamelHBaseFamily2	myfamily
CamelHBaseQualifier2	myqualifier
CamelHBaseValue2	myvalue2

如果检索操作，如 get 或 scan，您也可以为每个列指定您要将数据转换为的类型。对于考试：

标头	value
CamelHBaseFamily	myfamily
CamelHBaseQualifier	myqualifier
CamelHBaseValueType	Long

请注意，为了避免对所有消息保持恒定的样板标头，您也可以将它们指定为端点 uri 的一部分，如下所示。

143.8.2. 正文映射示例

要使用正文映射策略，您必须将选项 mappingStrategy 指定为 uri 的一部分，例如：

```
hbase:mytable?mappingStrategyName=body
```

要使用正文映射策略，正文需要包含 org.apache.camel.component.hbase.model.HBaseData 实例。您可以构建 t

```
HBaseData data = new HBaseData();
HBaseRow row = new HBaseRow();
row.setId("myRowId");
HBaseCell cell = new HBaseCell();
cell.setFamily("myfamily");
cell.setQualifier("myqualifier");
cell.setValue("myValue");
row.getCells().add(cell);
data.addRows().add(row);
```

以上对象可用于例如在放置操作中，并导致使用 id myRowId 创建或更新行，并将值 myfamily:myqualifier 添加到列 myfamily:myqualifier 中。
正文映射策略最初看似没有非常满意。它相对于标头映射策略的好处是，可以将 HBaseData 对象轻松转换为 xml/json。

143.9. 另请参阅

- polling Consumer
- Apache HBase

第 144 章 HDFS 组件（已弃用）

从 Camel 版本 2.8 开始提供

hdfs 组件可让您读取和写入来自/到 HDFS 文件系统的消息。HDFS 是分布式文件系统，是 Hadoop 的核心。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-hdfs</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

144.1. URI 格式

```
hdfs://hostname[:port][/path][?options]
```

您可以在 URI 中附加查询选项，格式为 ?option =value&option=value&...
The path is treated:

1. 作为消费者，如果文件是文件，则仅读取该文件，否则它将代表一个目录，它会扫描满足配置模式的路径下的所有文件。该目录下的所有文件必须相同类型。

2. 作为制作者，如果至少定义了一个分割策略，则路径被视为目录，并在该目录下，生成者使用配置的 UuidGenerator 为每个分割创建不同的文件。

备注

当从 hdfs 使用时，在正常模式下，文件被分成块，生成每个块的消息。您可以使用 chunkSize 选项配置块大小。如果要从 hdfs 读取并使用文件组件写入常规文件，您可以使用 fileMode=Append 将每个块一起附加。

144.2. 选项

HDFS 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
jAASConfiguration (common)	将给定的配置用于 JAAS 的安全性。		配置
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

HDFS 端点使用 URI 语法进行配置：

```
hdfs:hostName:port/path
```

使用以下路径和查询参数：

144.2.1. 路径参数(3 参数)：

Name	描述	默认值	类型
hostName	**要使用的** HDFS 主机		字符串
port	要使用的 HDFS 端口	8020	int
path	***必需*** 使用的目录路径		字符串

144.2.2. 查询参数(38 参数)：

Name	描述	默认值	类型
connectOnStartup (common)	是否在启动 producer/consumer 时连接到 HDFS 文件系统。如果为 false，则按需创建连接。请注意，HDFS 可能需要 15 分钟才能建立连接，因为它已重新传送硬编码 45 x 20 sec。通过将此选项设置为 false 可让您的应用程序启动，并且不会阻止 till 15 分钟。	true	布尔值
fileSystemType (common)	设置为 LOCAL 不使用 HDFS，而是使用本地 java.io.File。	HDFS	HdfsFileSystemType

Name	描述	默认值	类型
filetype (common)	要使用的文件类型。如需了解更多详细信息，请参阅关于各种文件类型的 Hadoop HDFS 文档。	NORMAL_FILE	HdfsFileType
keyType (common)	序列或映射文件时的键的类型。	NULL	WritableType
owner (common)	文件所有者必须与此所有者匹配，才能选择文件。否则会跳过该文件。		字符串
valueType (common)	如果序列或映射文件，键的类型	BYTES	WritableType
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
delay (consumer)	目录扫描之间的间隔（毫秒）。	1000	long
initialDelay (consumer)	对于消费者，在开始扫描目录前等待多少（毫秒）。		long
pattern (consumer)	用于扫描目录的模式	*	字符串
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy

Name	描述	默认值	类型
append (producer)	将 附加到现有文件。请注意，并非所有 HDFS 文件系统都支持 append 选项。	false	布尔值
overwrite (producer)	是否覆盖具有相同名称的现有文件	true	布尔值
blockSize (advanced)	HDFS 块的大小	67108864	long
bufferSize (advanced)	HDFS 使用的缓冲区大小	4096	int
checkIdleInterval (advanced)	运行空闲检查器后台任务的频率（以 millis ）。只有在 splitter 策略是 IDLE 时，这个选项才被使用。	500	int
CHUNKSIZE (advanced)	在读取普通文件时，这会分割为每块生成消息的块。	4096	int
compressionCodec (advanced)	要使用的压缩代码	DEFAULT	HdfsCompressionCodec
compressionType (advanced)	要使用的压缩类型（默认值不使用）	NONE	CompressionType
openSuffix (advanced)	打开用于读取/写入文件时使用后缀重命名的文件时，以避免在编写阶段读取该文件。	打开	字符串
readSuffix (advanced)	使用此后缀重命名文件后，以避免再次读取该文件。	读取	字符串
复制 （高级）	HDFS 复制因素	3	short
splitStrategy (advanced)	在当前版本的 Hadoop 中，Hadoop 在 append 模式中打开一个文件被禁用，因为它不可靠。因此，目前只能创建新文件。Camel HDFS 端点尝试以这种方式解决此问题：如果定义了 split 策略选项，则 hdfs 路径将用作目录，使用配置的 UuidGenerator 创建文件。每次满足分割条件时，都会创建一个新文件。 splitStrategy 选项被定义为一个字符串，语法如下： splitStrategy=ST:value,ST:value,... where ST can be be: BYTES a new file is created file, and the old is closed when the written bytes is more than value MESSAGES a new file is created, old is closed when the written messages is more than a new file is more than a new file is in a new file, in last milliseconds		字符串

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

144.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.hdfs.enabled	启用 hdfs 组件	true	布尔值
camel.component.hdfs.j-a-a-s-configuration	将给定的配置用于 JAAS 的安全性。选项是 javax.security.auth.login.Configuration 类型。		字符串
camel.component.hdfs.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

144.3.1. keyType 和 ValueType

- ***NULL 表示缺少键或值***
- ***用于编写字节的 BYTE，java Byte 类映射到 BYTE***
- ***用于编写一系列字节的 BYTES。它映射 java ByteBuffer 类***
- ***INT 用于编写 java 整数***
- ***用于编写 java 浮点的 FLOAT***
- ***用于编写 java 长的 LONG***
- ***用于编写 java double 的 DOUBLE***
- ***写入 java 字符串的 TEXT***

BYTES 也与所有其他对象一起使用，例如，在 Camel 中，文件作为 InputStream 发送，在内，这个情况会以一个序列形式编写，或映射文件作为一系列字节数。

144.4. 分割策略

在当前版本的 Hadoop 中，Hadoop 在 append 模式中打开一个文件被禁用，因为它不可靠。因此，目前只能创建新文件。Camel HDFS 端点尝试以这种方式解决这个问题：

- 如果定义了 split 策略选项，hdfs 路径将用作目录，使用配置的 UuidGenerator 创建文件
- 每次满足分割条件时，都会创建一个新文件。
  splitStrategy 选项被定义为一个字符串，语法如下：
  splitStrategy=<ST>:<value>,<ST>:<value>,*

其中 <ST> 可以是：

- BYTES 创建了一个新文件，当写入字节数超过 <value> 时，旧的将关闭
- MESSAGES 创建了一个新文件，当写入的消息数量超过 <value> 时，旧旧会被关闭
- IDLE 创建了一个新文件，当最后一个 <value> 毫秒中没有写时，旧的会被关闭

备注

请注意，这个策略目前需要设置 IDLE 值，或者将 HdfsConstants.HDFS_CLOSE 标头设置为 false 来使用 BYTES/MESSAGES configuration...otherwise，该文件将关闭每个消息

例如：

```
hdfs://localhost/tmp/simple-file?splitStrategy=IDLE:1000,BYTES:5
```

这意味着，当新文件闲置超过 1 秒或写入了 5 字节时，就会创建一个新文件。因此，运行 hadoop fs -ls /tmp/simple-file 您会看到已创建了多个文件。

144.5. 消息标头

此组件支持以下标头：

144.5.1. 仅限生成者

标头	描述
CamelFileName	Camel 2.13 ： 指定要写入的文件的名称（相对于端点路径）。名称可以是 **String** 或 Expression 对象。仅在不使用分割策略时相关。

144.6. 控制关闭文件流

从 Camel 2.10.4 开始提供

当在没有 分割策略时使用 HDFS producer 时，文件输出流默认在写入后关闭。但是，您可能希望使流保持打开，以后只明确关闭流。为此，您可以使用标头 HdfsConstants.HDFS_CLOSE (value = "CamelHdfsClose")来控制它。将此值设置为布尔值后，您可以明确控制流是否应关闭。

请注意，如果您使用分割策略，则这不适用，因为有各种策略可以控制流何时关闭。

144.7. 在 OSGI 中使用此组件

这个组件在 OSGi 环境中完全正常工作，但它需要用户的一些操作。Hadoop 使用线程上下文类加载程序来加载资源。通常，线程上下文类加载器将是包含路由的捆绑包的捆绑包类加载程序。因此，默认的配置文件需要在捆绑包类加载程序中可见。处理它的典型方法是在捆绑包 root 中保留 core-default.xml 副本。该文件可以在 hasoop-common.jar 中找到。

第 145 章 HDFS2 组件

从 Camel 版本 2.14 开始提供

hdfs2 组件可让您使用 Hadoop 2.x 从/向 HDFS 文件系统读取和写入消息。HDFS 是分布式文件系统，是 Hadoop 的核心。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-hdfs2</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

145.1. URI 格式

```
hdfs2://hostname[:port][/path][?options]
```

您可以在 URI 中附加查询选项，格式为 ?option =value&option=value&... The path is treated:

1. 作为消费者，如果文件是文件，则仅读取该文件，否则它将代表一个目录，它会扫描满足配置模式的路径下的所有文件。该目录下的所有文件必须相同类型。

2. 作为制作者，如果至少定义了一个分割策略，则路径被视为目录，并在该目录下，生成者使用配置的 UuidGenerator 为每个分割创建不同的文件。

当从 hdfs2 消耗时，在正常模式中，文件被分成块，生成每个块的消息。您可以使用 chunkSize 选项配置块大小。如果要从 hdfs 读取并使用文件组件写入常规文件，您可以使用 fileMode=Append 将每个块一起附加。

145.2. 选项

HDFS2 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
jAASConfiguration (common)	将给定的配置用于 JAAS 的安全性。		配置
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

HDFS2 端点使用 URI 语法进行配置：

```
hdfs2:hostName:port/path
```

使用以下路径和查询参数：

145.2.1. 路径参数(3 参数)：

Name	描述	默认值	类型
hostName	**要使用的** HDFS 主机		字符串
port	要使用的 HDFS 端口	8020	int
path	**必需** 使用的目录路径		字符串

145.2.2. 查询参数(38 参数)：

Name	描述	默认值	类型
connectOnStartup (common)	是否在启动 producer/consumer 时连接到 HDFS 文件系统。如果为 false，则按需创建连接。请注意，HDFS 可能需要 15 分钟才能建立连接，因为它已重新传送硬编码 45 x 20 sec。通过将此选项设置为 false 可让您的应用程序启动，并且不会阻止 till 15 分钟。	true	布尔值
fileSystemType (common)	设置为 LOCAL 不使用 HDFS，而是使用本地 java.io.File。	HDFS	HdfsFileSystemType
filetype (common)	要使用的文件类型。如需了解更多详细信息，请参阅关于各种文件类型的 Hadoop HDFS 文档。	NORMAL_FILE	HdfsFileType

Name	描述	默认值	类型
keyType (common)	序列或映射文件时的键的类型。	NULL	WritableType
owner (common)	文件所有者必须与此所有者匹配，才能选择文件。否则会跳过该文件。		字符串
valueType (common)	如果序列或映射文件，键的类型	BYTES	WritableType
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
pattern (consumer)	用于扫描目录的模式	*	字符串
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
append (producer)	将 附加到现有文件。请注意，并非所有 HDFS 文件系统都支持 append 选项。	false	布尔值
overwrite (producer)	是否覆盖具有相同名称的现有文件	true	布尔值
blockSize (advanced)	HDFS 块的大小	67108864	long

Name	描述	默认值	类型
bufferSize (advanced)	HDFS 使用的缓冲区大小	4096	int
checkIdleInterval (advanced)	运行空闲检查器后台任务的频率（以 millis ）。只有在 splitter 策略是 IDLE 时，这个选项才被使用。	500	int
CHUNKSIZE (advanced)	在读取普通文件时，这会分割为每块生成消息的块。	4096	int
compressionCodec (advanced)	要使用的压缩代码	DEFAULT	HdfsCompressionCodec
compressionType (advanced)	要使用的压缩类型（默认值不使用）	NONE	CompressionType
openSuffix (advanced)	打开用于读取/写入文件时使用后缀重命名的文件时，以避免在编写阶段读取该文件。	打开	字符串
readSuffix (advanced)	使用此后缀重命名文件后，以避免再次读取该文件。	读取	字符串
复制 （高级）	HDFS 复制因素	3	short
splitStrategy (advanced)	在当前版本的 Hadoop 中，Hadoop 在 append 模式中打开一个文件被禁用，因为它不可靠。因此，目前只能创建新文件。Camel HDFS 端点尝试以这种方式解决此问题：如果定义了 split 策略选项，则 hdfs 路径将用作目录，使用配置的 UuidGenerator 创建文件。每次满足分割条件时，都会创建一个新文件。splitStrategy 选项被定义为一个字符串，语法如下：splitStrategy=ST:value,ST:value,... where ST can be be: BYTES a new file is created file, and the old is closed when the written bytes is more than value MESSAGES a new file is created, old is closed when the written messages is more than a new file is more than a new file is in a new file, in last milliseconds		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int

Name	描述	默认值	类型
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

145.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.hdfs2.enabled	启用 hdfs2 组件	true	布尔值
camel.component.hdfs2.j-a-a-s-configuration	将给定的配置用于 JAAS 的安全性。选项是 javax.security.auth.login.Configuration 类型。		字符串
camel.component.hdfs2.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

145.3.1. keyType 和 ValueType

- ***NULL 表示缺少键或值***
- ***用于编写字节的 BYTE，java Byte 类映射到 BYTE***
- ***用于编写一系列字节的 BYTES。它映射 java ByteBuffer 类***
- ***INT 用于编写 java 整数***
- ***用于编写 java 浮点的 FLOAT***
- ***用于编写 java 长的 LONG***
- ***用于编写 java double 的 DOUBLE***
- ***写入 java 字符串的 TEXT***

BYTES 也与所有其他对象一起使用，例如，在 Camel 中，文件作为 InputStream 发送，在内，这个情况会以一个序列形式编写，或映射文件作为一系列字节数。

145.4. 分割策略

在当前版本的 Hadoop 中，Hadoop 在 append 模式中打开一个文件被禁用，因为它不可靠。因此，目前只能创建新文件。Camel HDFS 端点尝试以这种方式解决这个问题：

- 如果定义了 split 策略选项，hdfs 路径将用作目录，使用配置的 UuidGenerator 创建文件
- 每次满足分割条件时，都会创建一个新文件。
  splitStrategy 选项被定义为一个字符串，语法如下： splitStrategy=<ST>:<value>,<ST>:<value>,*

其中 <ST> 可以是：

- BYTES 创建了一个新文件，当写入字节数超过 <value> 时，旧的将关闭
- MESSAGES 创建了一个新文件，当写入的消息数量超过 <value> 时，旧旧会被关闭
- IDLE 创建了一个新文件，当最后一个 <value> 毫秒中没有写时，旧的会被关闭

请注意，这个策略目前需要设置 IDLE 值，或者将 HdfsConstants.HDFS_CLOSE 标头设置为 false 来使用 BYTES/MESSAGES configuration...otherwise，该文件将关闭每个消息

例如：

```
hdfs2://localhost/tmp/simple-file?splitStrategy=IDLE:1000,BYTES:5
```

这意味着，当新文件闲置超过 1 秒或写入了 5 字节时，就会创建一个新文件。因此，运行 hadoop fs -ls /tmp/simple-file 您会看到已创建了多个文件。

145.5. 消息标头

此组件支持以下标头：

145.5.1. 仅限生成者

标头	描述
CamelFileName	Camel 2.13 ： 指定要写入的文件的名称（相对于端点路径）。名称可以是 **String** 或 Expression 对象。仅在不使用分割策略时相关。

145.6. 控制关闭文件流

当在没有 分割策略时使用 HDFS2 producer 时，文件输出流默认在写入后关闭。但是，您可能希望使流保持打开，以后只明确关闭流。为此，您可以使用标头 HdfsConstants.HDFS_CLOSE (value = "CamelHdfsClose")来控制它。将此值设置为布尔值后，您可以明确控制流是否应关闭。

请注意，如果您使用分割策略，则这不适用，因为有各种策略可以控制流何时关闭。

145.7. 在 OSGI 中使用此组件

在与 Hadoop 2.x 机制相关的 OSGi 环境中运行此组件时有一些要求，以发现不同的 org.apache.hadoop.fs.FileSystem 实施。Hadoop 2.x 使用 java.util.ServiceLoader，它查找定义可用文件系统类型和实施的 /META-INF/services/org.apache.hadoop.fs.FileSystem 文件。在 OSGi 中运行时，这些资源不可用。

与 camel-hdfs 组件一样，需要从捆绑包类加载程序中查看默认配置文件。处理它的一种典型方法是在您的捆绑包 root 中保留 core-default.xml （以及 hdfs-default.xml）的副本。

145.7.1. 将此组件与手动定义的路由搭配使用

有两个选项：

1. 使用定义路由的捆绑包，软件包 /META-INF/services/org.apache.hadoop.fs.FileSystem 资源。此资源应列出所有所需的 Hadoop 2.x 文件系统实施。

2. 提供 boilerplate 初始化代码，它会在 org.apache.hadoop.fs.FileSystem 类中填充内部静态缓存：

```
org.apache.hadoop.conf.Configuration conf = new org.apache.hadoop.conf.Configuration();
conf.setClass("fs.file.impl", org.apache.hadoop.fs.LocalFileSystem.class, FileSystem.class);
```

```
conf.setClass("fs.hdfs.impl", org.apache.hadoop.hdfs.DistributedFileSystem.class,
FileSystem.class);
...
FileSystem.get("file:///", conf);
FileSystem.get("hdfs://localhost:9000/", conf);
...
```

145.7.2. 将这个组件与 Blueprint 容器一起使用

两个选项：

1. 带有包含蓝图定义的捆绑包的 /META-INF/services/org.apache.hadoop.fs.FileSystem 资源。

2. 将以下内容添加到蓝图定义文件中：

```
<bean id="hdfsOsgiHelper" class="org.apache.camel.component.hdfs2.HdfsOsgiHelper">
  <argument>
    <map>
      <entry key="file:///" value="org.apache.hadoop.fs.LocalFileSystem"  />
      <entry key="hdfs://localhost:9000/"
value="org.apache.hadoop.hdfs.DistributedFileSystem" />
      ...
    </map>
  </argument>
</bean>

<bean id="hdfs2" class="org.apache.camel.component.hdfs2.HdfsComponent" depends-
on="hdfsOsgiHelper" />
```

这样，Hadoop 2.x 将正确映射 URI 方案到文件系统实施。

第 146 章 HEADERSMAP

从 Camel 2.20 开始提供

camel-headersmap 是一个快速实现情况映射的实现，它可以在运行时插入并供 Camel Message 标头使用。

146.1. 从 CLASSPATH 中自动检测

要使用此实现，您只需要将 camel-headersmap 依赖项添加到 classpath 中，Camel 应该在启动时自动检测它，并记录如下：

```
Detected and using custom HeadersMapFactory:
org.apache.camel.component.headersmap.FastHeadersMapFactory@71e9ebae
```

对于 spring-boot，您可以使用 camel-headersmap-starter 依赖项。

146.2. 手动启用

如果您使用 OSGi 或实施没有添加到 classpath 中，则需要启用这个预期：

```
CamelContext camel = ...

camel.setHeadersMapFactory(new FastHeadersMapFactory());
```

或者，在 XML DSL (spring 或 blueprint XML 文件)中，您可以将工厂声明为 < bean> ：

```
<bean id="fastMapFactory"
class="org.apache.camel.component.headersmap.FastHeadersMapFactory"/>
```

然后，Camel 应该检测 bean 并使用工厂。

第 147 章 HESSIAN DATAFORMAT （已弃用）

从 Camel 版本 2.17 开始提供

Hessian 是 Data Format for marshalling and unmarshalling 消息，使用 Caucho 的 Hessian 格式。

如果要使用 Maven 中的 Hessian Data Format，请将以下内容添加到 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-hessian</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

147.1. 选项

Hessian dataformat 支持 4 个选项，如下所列。

Name	默认值	Java 类型	描述
whitelistEnabled	true	布尔值	定义是否启用了 Whitelist 功能
allowedUnmarshallObjects		字符串	定义允许的对象为 unmarshalled
deniedUnmarshallObjects		字符串	将被拒绝的对象定义为 unmarshalled
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

147.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.hessian.allowed-unmarshall-objects	定义允许的对象为 unmarshalled		字符串
camel.dataformat.hessian.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.hessian.denied-unmarshall-objects	将被拒绝的对象定义为 unmarshalled		字符串
camel.dataformat.hessian.enabled	启用 hessian dataformat	true	布尔值
camel.dataformat.hessian.whitelist-enabled	定义是否启用了 Whitelist 功能	true	布尔值

ND

147.3. 在 JAVA DSL 中使用 HESSIAN 数据格式

```
from("direct:in")
    .marshal().hessian();
```

147.4. 在 SPRING DSL 中使用 HESSIAN 数据格式

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
    <route>
        <from uri="direct:in"/>
        <marshal ref="hessian"/>
    </route>
</camelContext>
```

第 148 章 HIPCHAT 组件

从 Camel 版本 2.15 开始提供

Hipchat 组件支持从/到 Hipchat 服务生成和使用消息。

先决条件

您必须有一个有效的 Hipchat 用户帐户并获得可用于生成/恢复消息的个人访问令牌。https://www.hipchat.com/account/api

148.1. URI 格式

```
hipchat://[host][:port]?options
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

148.2. URI 选项

Hipchat 组件没有选项。

Hipchat 端点使用 URI 语法进行配置：

```
hipchat:protocol:host:port
```

使用以下路径和查询参数：

148.2.1. 路径参数(3 参数)：

Name	描述	默认值	类型
protocol	需要 hipchat 服务器的协议，如 http。		字符串
主机	需要 hipchat 服务器的主机，如 api.hipchat.com		字符串

Name	描述	默认值	类型
port	hipchat 服务器的端口。默认为 80。	80	整数

148.2.2. 查询参数(22 参数)：

Name	描述	默认值	类型
authToken (common)	OAuth 2 身份验证令牌		字符串
consumeUsers (common)	从 hiptchat 服务器消耗消息时的用户名。可以使用逗号分隔多个用户名。		字符串
httpclient (common)	从 registry 的 CloseableHttpClient 引用，以便在 API HTTP 请求期间使用。	来自 HttpClient 库的 CloseableHttpClient 默认	CloseableHttpClient
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

148.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.hipchat.enabled	启用 hipchat 组件	true	布尔值
camel.component.hipchat.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

148.4. SCHEDULED POLL CONSUMER

此组件实现 ScheduledPollConsumer。只有提供的"consumeUsers"中的最后一个消息才会检索并作为 Exchange body 发送。如果您不希望在下一次轮询上没有新消息时再次检索相同的消息，则可以添加幂等消费者，如下所示。ScheduledPollConsumer 中的所有选项也可用于对消费者进行更多控制。

```
@Override
public void configure() throws Exception {
 String hipchatEndpointUri = "hipchat://?authToken=XXXX&consumeUsers=@Joe,@John";
 from(hipchatEndpointUri)
  .idempotentConsumer(
    simple("${in.header.HipchatMessageDate} ${in.header.HipchatFromUser}"),
    MemoryIdempotentRepository.memoryIdempotentRepository(200)
  )
  .to("mock:result");
}
```

148.4.1. Hipchat consumer 设置的消息标头

标头	常数	类型	描述
HipchatFromUser	HipchatConstants.FROM_USER	*字符串*	正文包含从此用户发送到 authToken 的所有者的消息

标头	常数	类型	描述
HipchatMessageDate	HipchatConstants.MESSAGE_DATE	*字符串*	发送的日期消息。格式为 ISO-8601，如 Hipchat 响应 中所示。

148.5. HIPCHAT PRODUCER

生产者可以同时将消息发送到 Room 的 和 User。交换的正文作为邮件发送。示例用法如下所示。需要设置适当的标头。

```
@Override
 public void configure() throws Exception {
  String hipchatEndpointUri = "hipchat:///?authToken=XXXX";
  from("direct:start")
   .to(hipchatEndpointUri)
   .to("mock:result");
 }
```

148.5.1. Hipchat producer 评估的消息标头

标头	常数	类型	描述
HipchatToUser	HipchatConstants.TO_USER	*字符串*	需要发送消息的 Hipchat 用户。
HipchatToRoom	HipchatConstants.TO_ROOM	*字符串*	需要向其发送消息的 Hipchat 房间。
HipchatMessageFormat	HipchatConstants.MESSAGE_FORMAT	字符串	有效格式为 'text' 或 'html'。**默认： 'text'**

标头	常数	类型	描述
HipchatMessageBackgroundColor	HipchatConstants.MESSAGE_BACKGROUND_COLOR	*字符串*	有效的颜色值为 'yellow', 'green', 'red', 'purple', 'gray', 'random'。**默认： 'yellow'(Room Only)**
HipchatTriggerNotification	HipchatConstants.TRIGGER_NOTIFY	*字符串*	有效值为 'true' 或 'false'。此消息是否应触发用户通知（更改选项卡颜色、播放声音、通知移动电话等）。**默认： 'false'(Room Only)**

148.5.2. Hipchat producer 设置的消息标头

标头	常数	类型	描述
HipchatToUserResponseStatus	HipchatConstants.TO_USER_RESPONSE_STATUS	*StatusLine* 发送到用户的消息时收到的 API 响应的状态。	HipchatFromUserResponseStatus

148.5.3. 配置 Http Client

HipChat 组件允许您自己的 HttpClient 配置。这可以通过在 registry （如 Spring Context）中定义 CloseableHttpClient 的引用，然后在 Endpoint 定义中设置参数，例如： hipchat:http://api.hipchat.com?httpClient=#myHttpClient。

```
CloseableHttpClient httpclient = HttpClients.custom()
    .setConnectionManager(connManager)
    .setDefaultCookieStore(cookieStore)
    .setDefaultCredentialsProvider(credentialsProvider)
    .setProxy(new HttpHost("myproxy", 8080))
    .setDefaultRequestConfig(defaultRequestConfig)
    .build();
```

要查看有关 Http 客户端配置的更多信息，请参阅 官方文档。

148.5.4. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-hipchat</artifactId>
    <version>${camel-version}</version>
</dependency>
```

其中 ${camel-version} 必须替换为 Camel 的实际版本(2.15.0 或更高版本)

第 149 章 HL7 DATAFORMAT

从 Camel 版本 2.0 开始提供

HL7 组件用于使用 HL7 MLLP 协议和 HL7 v2 消息 （利用 HAPI 库）。

这个组件支持以下内容：

- Mina的 HL7 MLLP codec
- 用于 Camel 2.15 以后 Netty4 的 HL7 MLLP codec
- 键入 Converter from/to HAPI 和 String
- 使用 HAPI 库的 HL7 DataFormat
- 更加易于使用，因为它与 camel-mina2 组件集成在一起。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-hl7</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

149.1. HL7 MLLP 协议

HL7 通常与 HL7 MLLP 协议一起使用，该协议是基于文本的 TCP 套接字协议。此组件附带一个 Mina 和 Netty4 Codec，它符合 MLLP 协议，以便您可以轻松地公开接受 TCP 传输层上的 HL7 请求的 HL7 侦听器。要公开 HL7 侦听器服务，camel-mina2 或 camel-netty4 组件与 HL7MLLPCodec (mina2)或 HL7MLLPNettyDecoder/HL7MLLPNettyEncoder (Netty4)一起使用。

HL7 MLLP codec 可以配置如下：

Name	默认值	描述
startByte	**0x0b**	跨越 HL7 有效负载的开始字节。
endByte1	**0x1c**	跨越 HL7 有效负载的第一个结束字节。
endByte2	**0x0d**	跨越 HL7 有效负载的第 2 个结束字节。
charset	JVM 默认	用于 codec 的编码(charset 名称)。如果没有提供，Camel 将使用 JVM 默认 Charset。
produceString	**true**	**（自 Camel 2.14.1）** 如果为 true，codec 会使用定义的 charset 创建一个字符串。如果为 false，则 codec 会将一个普通字节数组发送到路由中，以便 HL7 Data Format 可以确定 HL7 消息内容的实际 charset。
convertLFtoCR	**false**	将 **\n** 转换为 **\r** (**0x0d**, 13 十进制)作为 HL7 stipulates**\r** 作为网段术语。HAPI 库需要使用 **\r**。

149.1.1. 使用 Mina 公开 HL7 侦听器

在 Spring XML 文件中，我们将 mina2 端点配置为使用 TCP 在端口 8888 上侦听 HL7 请求：

```
<endpoint id="hl7MinaListener" uri="mina2:tcp://localhost:8888?sync=true&amp;codec=#hl7codec"/>
```

sync=true 表示此监听程序是同步的，因此会向调用者返回 HL7 响应。HL7 codec 使用 codec=problemhl7codec 设置。请注意，hl7codec 只是 Spring bean ID，因此可以命名为 mygreatcodecforhl7 或任何。codec 还在 Spring XML 文件中设置：

```
<bean id="hl7codec" class="org.apache.camel.component.hl7.HL7MLLPCodec">
    <property name="charset" value="iso-8859-1"/>
</bean>
```

然后，端点 hl7MinaListener 可以在作为消费者的路由中使用，因为此 Java DSL 示例：

```
from("hl7MinaListener")
  .bean("patientLookupService");
```

这是一个非常简单的路由，它将侦听 HL7，并将其路由到名为 patient LookupService 的服务。这也是 Spring XML 中配置的 Spring bean ID：

```
<bean id="patientLookupService"
class="com.mycompany.healthcare.service.PatientLookupService"/>
```

业务逻辑可以在不依赖于 Camel 的 POJO 类中实施，如下所示：

```
import ca.uhn.hl7v2.HL7Exception;
import ca.uhn.hl7v2.model.Message;
import ca.uhn.hl7v2.model.v24.segment.QRD;

public class PatientLookupService {
    public Message lookupPatient(Message input) throws HL7Exception {
        QRD qrd = (QRD)input.get("QRD");
        String patientId = qrd.getWhoSubjectFilter(0).getIDNumber().getValue();

        // find patient data based on the patient id and create a HL7 model object with the
response
        Message response = ... create and set response data
        return response
    }
```

149.1.2. 使用 Netty 公开 HL7 侦听器（自 Camel 2.15 起可用）

在 Spring XML 文件中，我们将 netty4 端点配置为使用 TCP 在端口 8888 上侦听 HL7 请求：

```
<endpoint id="hl7NettyListener" uri="netty4:tcp://localhost:8888?
sync=true&amp;encoder=#hl7encoder&amp;decoder=#hl7decoder"/>
```

sync=true 表示此监听程序是同步的，因此会向调用者返回 HL7 响应。HL7 codec 使用 encoder="hl7encoder*and*decoder="hl7decoder 进行设置。请注意，hl7encoder 和 hl7decoder 只是 bean ID，因此它们可以以不同的方式命名。Bean 可以在 Spring XML 文件中设置：

```
<bean id="hl7decoder" class="org.apache.camel.component.hl7.HL7MLLPNettyDecoderFactory"/>
<bean id="hl7encoder" class="org.apache.camel.component.hl7.HL7MLLPNettyEncoderFactory"/>
```

然后，端点 hl7NettyListener 可以在作为消费者的路由中使用，因为此 Java DSL 示例：

```
from("hl7NettyListener")
  .bean("patientLookupService");
```

149.2. 使用 JAVA.LANG.STRING 或 BYTE[] 的 HL7 MODEL

HL7 MLLP codec 使用普通字符串作为其数据格式。Camel 使用 Type Converter 将字符串转换为 HAPI HL7 模型对象，但如果您想自己解析数据，您可以使用普通的 String 对象。

从 Camel 2.14.1 开始，您还可以让 Mina 和 Netty codecs 使用 plain byte[] 作为其数据格式，方法是将 produceString 属性设置为 false。Type Converter 也可以将 byte[] 转换为 HAPI HL7 模型对象。

149.3. 使用 HAPI 的 HL7V2 MODEL

HL7v2 模型使用 HAPI 库中的 Java 对象。使用这个库，您可以对大多数用于 HL7v2 的 EDI 格式 (ER7)进行编码和解码。

以下示例是查找具有病人 ID 0101701234 的病人的请求。

```
MSH|^~\\&|MYSENDER|MYRECEIVER|MYAPPLICATION||200612211200||QRY^A19|1234|P|2.4
QRD|200612211200|R|I|GetPatient|||1^RD|0101701234|DEM||
```

使用 HL7 模型，您可以使用 ca.uhn.hl7v2.model.Message 对象，例如检索一个病人 ID：

```
Message msg = exchange.getIn().getBody(Message.class);
QRD qrd = (QRD)msg.get("QRD");
String patientId = qrd.getWhoSubjectFilter(0).getIDNumber().getValue(); // 0101701234
```

这在与 HL7 侦听器结合使用时非常强大，因为您不必使用 byte[]、String 或任何其他简单对象格式。您只能使用 HAPI HL7v2 模型对象。如果您事先知道消息类型，您可以输入更多类型：

```
QRY_A19 msg = exchange.getIn().getBody(QRY_A19.class);
String patientId = msg.getQRD().getWhoSubjectFilter(0).getIDNumber().getValue();
```

149.4. HL7 DATAFORMAT

HL7 组件附带了 HL7 数据格式，可用于 marshal 或 unmarshal HL7 模型对象。

HL7 dataformat 支持 2 个选项，如下所列。

Name	默认值	Java 类型	描述
validate	**true**	布尔值	是否默认验证 HL7 消息是否默认为 true。
contentTypeHeader	**false**	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

149.5. SPRING BOOT AUTO-CONFIGURATION

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.hl7.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.hl7.enabled	启用 hl7 dataformat	true	布尔值
camel.dataformat.hl7.validate	是否默认验证 HL7 消息是否默认为 true。	true	布尔值
camel.language.terser.enabled	启用 terser 语言	true	布尔值
camel.language.terser.trim	是否修剪值以移除前导和结尾的空格和换行符	true	布尔值

ND

- ***marshal = 从 Message 到字节流（在使用 HL7 MLLP codec 时可以使用）***
- ***unmarshal = 从字节流到消息（可在从 HL7 MLLP 接收流数据时使用***

要使用数据格式，只需实例化实例并在路由构建器中调用 marshal 或 unmarshal 操作：

```
DataFormat hl7 = new HL7DataFormat();

from("direct:hl7in")
  .marshal(hl7)
  .to("jms:queue:hl7out");
```

在上例中，HL7 从 HAPI Message 对象放入字节流并放入 JMS 队列。
下一个示例是相反：

```
DataFormat hl7 = new HL7DataFormat();

from("jms:queue:hl7out")
  .unmarshal(hl7)
  .to("patientLookupService");
```

在这里，我们将字节流入 HAPI 消息对象，该对象传递到我们的病人查找服务。

149.5.1. 序列化消息

从 HAPI 2.0 开始（由 Camel 2.11使用），HL7v2 模型类可以完全序列化。因此，您可以将 HL7v2 消息直接放入 JMS 队列（即不调用 marshal （） ），然后直接从队列读取它们（例如，无需调用 unmarshal （） ）。

149.5.2. 片段分隔符

从 Camel 2.11 开始，unmarshal 不再通过将 \n 转换为 \r 来自动修复网段分隔符。如果您需要这个转换，org.apache.camel.component.hl7.HL7 failingconvertLFToCR 为这个目的提供了一个方便的 Expression。

149.5.3. charset

自 Camel 2.14.1 起，marshal 和 unmarshal 都评估字段 MSH-18 中提供的 charset。如果此字段为空，则默认假定对应的 Camel charset property/header 中包含的 charset。在继承 HL7DataFormat 类时，您也可以通过覆盖 guessCharsetName 方法来更改此默认行为。

Camel 中有一个简写的语法，用于常用数据格式。然后，您不需要创建 HL7DataFormat 对象的实例：

```
from("direct:hl7in")
  .marshal().hl7()
  .to("jms:queue:hl7out");
```

```
from("jms:queue:hl7out")
  .unmarshal().hl7()
  .to("patientLookupService");
```

149.6. 消息标头

unmarshal 操作将 MSH 片段中的这些字段作为 Camel 信息上的标头添加：

键	MSH 字段	示例
CamelHL7SendingApplication	MSH-3	MYSERVER
CamelHL7SendingFacility	MSH-4	MYSERVERAPP
CamelHL7ReceivingApplication	MSH-5	MYCLIENT
CamelHL7ReceivingFacility	MSH-6	MYCLIENTAPP
CamelHL7Timestamp	MSH-7	20071231235900
CamelHL7Security	MSH-8	null
CamelHL7MessageType	MSH-9-1	ADT

键	MSH 字段	示例
CamelHL7TriggerEvent	**MSH-9-2**	**A01**
CamelHL7MessageControl	**MSH-10**	**1234**
CamelHL7ProcessingId	**MSH-11**	**P**
CamelHL7VersionId	**MSH-12**	**2.4**
`CamelHL7Context	``	' **(Camel 2.14)** 包含用于解析消息的 HapiContext
CamelHL7Charset	**MSH-18**	(Camel 2.14.1) **UNICODE UTF-8**

除 CamelHL7Context 'are 'String types 之外的所有标头。如果缺少标头值，则其值为 null。

149.7. 选项

HL7 数据格式支持以下选项：

选项	默认	描述
validate	true	HAPI Parser 是否应该使用默认验证规则验证消息。建议您使用 **parser** 或 **hapiContext** 选项，并使用所需的 HAPI **ValidationContext**初始化它

选项	默认	描述
解析器	**ca.uhn.hl7v2.parser.GenericParser**	要使用的自定义解析器。必须是 **ca.uhn.hl7v2.parser.Parser**。请注意，**GenericParser** 还允许解析 XML 编码的 HL7v2 信息
hapiContext	**ca.uhn.hl7v2.DefaultHapiContext**	Camel 2.14： 可定义自定义解析器、自定义 ValidationContext 等自定义 HAPI 上下文。这可让您完全控制 HL7 解析和渲染过程。

149.8. 依赖项

要在 Camel 路由中使用 HL7，您需要添加对上面列出的 camel-hl7 的依赖，它会实施此数据格式。

HAPI 库被分成一个基础库和几个结构 库，每个 HL7v2 消息版本对应一个：

- ***v2.1 结构库***
- ***v2.2 结构库***
- ***v2.3 结构库***
- ***v2.3.1 结构库***
- ***v2.4 结构库***
- ***v2.5 结构库***
- ***v2.5.1 结构库***

- v2.6 结构库

默认情况下，camel-hl7 仅引用 HAPI 基础库。应用程序负责包括结构库本身。例如，如果应用程序与 HL7v2 消息版本 2.4 和 2.5 一起工作，则必须添加以下依赖项：

```
<dependency>
  <groupId>ca.uhn.hapi</groupId>
  <artifactId>hapi-structures-v24</artifactId>
  <version>2.2</version>
  <!-- use the same version as your hapi-base version -->
</dependency>
<dependency>
  <groupId>ca.uhn.hapi</groupId>
  <artifactId>hapi-structures-v25</artifactId>
  <version>2.2</version>
  <!-- use the same version as your hapi-base version -->
</dependency>
```

或者，包含基本库的 OSGi 捆绑包，可以从 中央 Maven 存储库下载 所有结构库和所需依赖项（在捆绑包类路径上）。

```
<dependency>
  <groupId>ca.uhn.hapi</groupId>
  <artifactId>hapi-osgi-base</artifactId>
  <version>2.2</version>
</dependency>
```

149.9. TERSER 语言

HAPI 提供了一个 Terser 类，它通过常用的 terse 位置规格语法提供对字段的访问。Terser 语言允许使用此语法从消息中提取值，并将它们用作表达式和 predicates 进行过滤、基于内容的路由等。

示例：

```
import static org.apache.camel.component.hl7.HL7.terser;

  // extract patient ID from field QRD-8 in the QRY_A19 message above and put into message
header
  from("direct:test1")
    .setHeader("PATIENT_ID",terser("QRD-8(0)-1"))
    .to("mock:test1");

  // continue processing if extracted field equals a message header
```

```
from("direct:test2")
  .filter(terser("QRD-8(0)-1").isEqualTo(header("PATIENT_ID"))
  .to("mock:test2");
```

149.10. HL7 VALIDATION PREDICATE

通常最好首先解析 HL7v2 消息，并在单独的步骤中根据 HAPI ValidationContext 验证它。

示例：

```
import static org.apache.camel.component.hl7.HL7.messageConformsTo;
import ca.uhn.hl7v2.validation.impl.DefaultValidation;

  // Use standard or define your own validation rules
  ValidationContext defaultContext = new DefaultValidation();

  // Throws PredicateValidationException if message does not validate
  from("direct:test1")
    .validate(messageConformsTo(defaultContext))
    .to("mock:test1");
```

149.11. HL7 VALIDATION PREDICATE 使用 HAPICONTEXT (CAMEL 2.14)

HAPI 上下文始终配置有 ValidationContext （或 ValidationRuleBuilder），以便您可以间接访问验证规则。另外，当 unmarshalling the HL7DataFormat 在 CamelHL7Context 标头中转发配置的 HAPI 上下文时，可以轻松地重复使用此上下文的验证规则：

```
import static org.apache.camel.component.hl7.HL7.messageConformsTo;
import static org.apache.camel.component.hl7.HL7.messageConforms

  HapiContext hapiContext = new DefaultHapiContext();
  hapiContext.getParserConfiguration().setValidating(false); // don't validate during parsing

  // customize HapiContext some more ... e.g. enforce that PID-8 in ADT_A01 messages of
version 2.4 is not empty
  ValidationRuleBuilder builder = new ValidationRuleBuilder() {
    @Override
    protected void configure() {
      forVersion(Version.V24)
          .message("ADT", "A01")
          .terser("PID-8", not(empty()));
      }
    };
  hapiContext.setValidationRuleBuilder(builder);

  HL7DataFormat hl7 = new HL7DataFormat();
  hl7.setHapiContext(hapiContext);
```

```
from("direct:test1")
  .unmarshal(hl7)              // uses the GenericParser returned from the HapiContext
  .validate(messageConforms())  // uses the validation rules returned from the HapiContext
                          // equivalent with .validate(messageConformsTo(hapiContext))
  // route continues from here
```

149.12. HL7 ACKNOWLEDGEMENT 表达式

HL7v2 处理中的常见任务是生成确认消息，作为响应传入的 HL7v2 消息，如基于验证结果。ack 表达式让我们可以完美地完成这一目的：

```
import static org.apache.camel.component.hl7.HL7.messageConformsTo;
import static org.apache.camel.component.hl7.HL7.ack;
import ca.uhn.hl7v2.validation.impl.DefaultValidation;

// Use standard or define your own validation rules
ValidationContext defaultContext = new DefaultValidation();

from("direct:test1")
  .onException(Exception.class)
    .handled(true)
    .transform(ack()) // auto-generates negative ack because of exception in Exchange
    .end()
  .validate(messageConformsTo(defaultContext))
  // do something meaningful here

  // acknowledgement
  .transform(ack())
```

第 150 章 HTTP 组件（已弃用）

从 Camel 版本 1.0 开始提供

http: 组件为消耗外部 HTTP 资源提供基于 HTTP 的端点（作为客户端使用 HTTP 调用外部服务器）。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-http</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

150.1. URI 格式

```
http:hostname[:port][/resourceUri][?param1=value1][&param2=value2]
```

默认情况下，HTTP 和 443 将端口 80 用于 HTTPS。

camel-http vs camel-jetty

您只能生成到 HTTP 组件生成的端点。因此，它不应该被用作您的 camel Routes 中的输入。要绑定/通过 HTTP 服务器作为对 camel 路由的输入，您可以使用 Jetty 组件 或 Servlet 组件

150.2. 例子

使用 POST 使用正文调用 url，并返回响应作为 out 消息。如果正文为 null 调用 URL，使用 GET 并返回响应作为 out 信息

Java DSL

Spring DSL

```
from("direct:start")
  .to("http://myhost/mypath");
```

```
<from uri="direct:start"/>
<to uri="http://oldhost"/>
```

您可以通过添加一个标头来覆盖 HTTP 端点 URI。Camel 将调用 http://newhost。例如 REST urls 是非常方便的。

Java DSL

```
from("direct:start")
  .setHeader(Exchange.HTTP_URI, simple("http://myserver/orders/${header.orderId}"))
  .to("http://dummyhost");
```

URI 参数可以直接在端点 URI 或标头上设置

Java DSL

```
from("direct:start")
  .to("http://oldhost?order=123&detail=short");
from("direct:start")
  .setHeader(Exchange.HTTP_QUERY, constant("order=123&detail=short"))
  .to("http://oldhost");
```

将 HTTP 请求方法设置为 POST

Java DSL

Spring DSL

```
from("direct:start")
  .setHeader(Exchange.HTTP_METHOD, constant("POST"))
  .to("http://www.google.com");
```

```
<from uri="direct:start"/>
<setHeader headerName="CamelHttpMethod">
  <constant>POST</constant>
```

```
</setHeader>
<to uri="http://www.google.com"/>
<to uri="mock:results"/>
```

150.3. HTTP 选项

HTTP 组件支持 8 个选项，如下所列。

Name	描述	默认值	类型
httpClientConfigurer (advanced)	若要使用自定义 HttpClientConfigurer 来执行要使用的 HttpClient 配置。		HttpClientConfigurer
httpConnectionManager (advanced)	使用自定义 HttpConnectionManager 管理连接		HttpConnectionManager
httpBinding (producer)	使用自定义 HttpBinding 来控制 Camel 消息和 HttpClient 之间的映射。		HttpBinding
httpConfiguration (producer)	将共享 HttpConfiguration 用作基础配置。		HttpConfiguration
allowJavaSerialized Object (producer)	当请求使用 context-type=application/x-java-serialized-object This 默认为 off 时允许 java serialization。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

HTTP 端点使用 URI 语法进行配置：

```
http:httpUri
```

使用以下路径和查询参数：

150.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
httpUri	**必需** 调用的 HTTP 端点的 url。		URI

150.3.2. 查询参数(38 参数)：

Name	描述	默认值	类型
disableStreamCache (common)	确定 Servlet 中的原始输入流是否被缓存(Camel 将会在 memory/overflow 中读取流到文件，流缓存)缓存。默认情况下，Camel 将缓存 Servlet 输入流，以支持多次读取，以确保其 Camel 可以从流检索所有数据。但是，当您需要访问原始流时，您可以将此选项设置为 true，例如将其直接流传输到文件或其他持久性存储。DefaultHttpBinding 将请求输入流复制到流缓存中，如果此选项为 false，则将它放入消息正文，以支持多次读取流。如果您使用 Servlet 到 bridge/proxy 端点，则请考虑启用此选项来提高性能，如果您不需要多次读取消息有效负载。http/http4 producer 默认缓存响应正文流。如果将此选项设置为 true，则生成者不会缓存响应正文流，而是使用响应流作为消息正文。	false	布尔值
headerFilterStrategy (common)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
httpBinding (common)	使用自定义 HttpBinding 来控制 Camel 消息和 HttpClient 之间的映射。		HttpBinding
bridgeEndpoint (producer)	如果选项为 true，则 HttpProducer 将忽略 Exchange.HTTP_URI 标头，并使用端点的 URI 进行请求。您也可以将 选项 throwExceptionOnFailure 设置为 false，让 HttpProducer 发送所有错误响应。	false	布尔值
chunked (producer)	如果此选项为 false，则 Servlet 将禁用 HTTP 流，并在响应上设置 content-length 标头	true	布尔值
connectionClose (producer)	指定是否应将 Connection Close 标头添加到 HTTP Request 中。默认情况下，connectionClose 为 false。	false	布尔值

Name	描述	默认值	类型
copyHeaders (producer)	如果此选项为 true，则 IN Exchange 标头将根据复制策略复制到 OUT 交换标头。把它设置为 false，仅允许包含来自 HTTP 响应的标头（而不是传播 IN 标头）。	true	布尔值
httpMethod (producer)	配置要使用的 HTTP 方法。如果设置，HttpMethod 标头无法覆盖此选项。		HttpMethods
ignoreResponseBody (producer)	如果此选项为 true，http producer 不会读取响应正文，并缓存输入流	false	布尔值
preserveHostHeader (producer)	如果选项为 true，HttpProducer 会将 Host 标头设置为当前交换主机标头中包含的值，这适用于您希望下游服务器收到的 Host 标头来反映上游客户端调用的 URL 的 URL，这将允许使用 Host 标头为代理服务生成准确的 URL 的应用	false	布尔值
throwExceptionOnFailure (producer)	如果来自远程服务器的失败响应，用于禁用抛出 HttpOperationFailedException。这样，您可以获取所有响应，而不考虑 HTTP 状态代码。	true	布尔值
transferException (producer)	如果在消费者端启用并交换失败处理，如果原因 Exception 在响应中作为 application/x-java-serialized-object 内容类型发送序列化，则结果为 application/x-java-serialized-object 内容类型。在生产者侧，异常将反序列化并丢弃为原样，而不是 HttpOperationFailedException。原因异常需要按顺序处理。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
cookieHandler (producer)	配置 Cookie 处理程序，以维护 HTTP 会话		CookieHandler
okStatusCodeRange (producer)	被视为成功响应的状态代码。值包含。可以定义多个范围，用逗号分开，例如 200-204,209,301-304。每个范围必须是单个数字，或从 到，其中包含短划线。	200-299	字符串
urlRewrite (producer)	**弃用** 引用自定义 org.apache.camel.component.http.UrlRewrite，它允许您在 bridge/proxy 端点时重写 url。更多信息，请访问 http://camel.apache.org/urlrewrite.html		UrlRewrite
httpClientConfigurer (advanced)	为生成者或消费者创建的新 HttpClient 实例注册自定义配置策略，如配置身份验证机制等		HttpClientConfigurer
httpClientOptions (advanced)	使用映射中的键/值配置 HttpClient：		Map

Name	描述	默认值	类型
httpConnectionManager (advanced)	使用自定义 HttpConnectionManager 管理连接		HttpConnectionManager
httpConnectionManager 选项 (advanced)	使用映射中的键/值来配置 HttpConnectionManager。		Map
mapHttpMessageBody (advanced)	如果此选项为 true，则交换的 IN Exchange Body 将映射到 HTTP 正文。把它设置为 false 可以避免 HTTP 映射。	true	布尔值
mapHttpMessageFormUrlEncodedBody (advanced)	如果此选项为 true，则交换的 IN Exchange Form Encoded body 将映射到 HTTP。把它设置为 false 可以避免 HTTP Form Encoded body 映射。	true	布尔值
mapHttpMessageHeaders (advanced)	如果此选项为 true，则交换的 IN Exchange Headers 将映射到 HTTP 标头。把它设置为 false 将避免 HTTP 标头映射。	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
proxyAuthDomain (proxy)	与 NTML 搭配使用的代理身份验证域		字符串
proxyAuthHost (proxy)	代理身份验证主机		字符串
proxyAuthMethod (proxy)	要使用的代理验证方法		字符串
proxyAuthPassword (proxy)	代理身份验证密码		字符串
proxyAuthPort (proxy)	代理身份验证端口		int
proxyAuthScheme (proxy)	要使用的代理身份验证方案		字符串
proxyAuthUsername (proxy)	代理身份验证用户名		字符串
proxyHost (proxy)	要使用的代理主机名		字符串
proxyPort (proxy)	要使用的代理端口		int

Name	描述	默认值	类型
authDomain (security)	与 NTML 搭配使用的身份验证域		字符串
authHost (security)	与 NTML 搭配使用的身份验证主机		字符串
authmethod (security)	允许将 身份验证方法用作以逗号分开的值列表，即 Basic、Digest 或 NTLM。		字符串
authMethodPriority (security)	要优先使用哪个验证方法，可以是 Basic、Digest 或 NTLM。		字符串
authPassword (security)	身份验证密码		字符串
authUsername (security)	身份验证用户名		字符串

150.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
camel.component .http.allow-java-serialized-object	当请求使用 context-type=application/x-java-serialized-object This 默认为 off 时允许 java serialization。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
camel.component .http.enabled	启用 http 组件	true	布尔值
camel.component .http.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		字符串
camel.component .http.http-binding	使用自定义 HttpBinding 来控制 Camel 消息和 HttpClient 之间的映射。选项是 org.apache.camel.http.common.HttpBinding 类型。		字符串

Name	描述	默认值	类型
camel.component.http.http-client-configurer	若要使用自定义 HttpClientConfigurer 来执行要使用的 HttpClient 配置。选项是 org.apache.camel.component.http.HttpClientConfigurer 类型。		字符串
camel.component.http.http-configuration	将共享 HttpConfiguration 用作基础配置。选项是 org.apache.camel.http.common.HttpConfiguration 类型。		字符串
camel.component.http.http-connection-manager	使用自定义 HttpConnectionManager 来管理连接。选项是 org.apache.commons.httpclient.HttpConnectionManager 类型。		字符串
camel.component.http.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.http.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

150.5. 消息标头

Name	类型	描述
Exchange.HTTP_URI	**字符串**	要调用的 URI。将覆盖直接在端点上设置的现有 URI。此 uri 是要调用的 http 服务器的 uri。它与 Camel 端点 uri 不同，您可以在其中配置端点选项，如安全等。此标头不支持，它只是 http 服务器的 uri。
Exchange.HTTP_METHOD	**字符串**	HTTP 方法 / Verb 使用(GET/POST/PUT/DELETE/HEAD/OPTIONS/TRACE)
Exchange.HTTP_PATH	**字符串**	请求 URI 的路径，标头将使用 HTTP_URI 构建请求 URI。**Camel 2.3.0：** 如果路径以 "/" 开头，http producer 将尝试根据 Exchange.HTTP_BASE_URI 标头或 exchange.getFromEndpointUri （）；

Name	类型	描述
Exchange.HTTP_QUERY	**字符串**	URI 参数.将覆盖直接在端点上设置的现有 URI 参数。
Exchange.HTTP_RESPONSE_CODE	**int**	来自外部服务器的 HTTP 响应代码。确定为 200。
Exchange.HTTP_CHARACTER_ENCODING	**字符串**	字符编码.
Exchange.CONTENT_TYPE	**字符串**	HTTP 内容类型。在 IN 和 OUT 消息上设置，以提供内容类型，如 **text/html**。
Exchange.CONTENT_ENCODING	**字符串**	HTTP 内容编码。在 IN 和 OUT 消息上设置，以提供内容编码，如 **gzip**。
Exchange.HTTP_SERVLET_REQUEST	**HttpServletRequest**	**HttpServletRequest** 对象。

Name	类型	描述
Exchange.HTTP_SERVLET_RESPONSE	**HttpServletResponse**	**HttpServletResponse** 对象。
Exchange.HTTP_PROTOCOL_VERSION	**字符串**	Camel 2.5： 您可以使用此标头设置 http 协议版本，例如。"HTTP/1.0".如果您没有指定标头，HttpProducer 将使用默认值 "HTTP/1.1"

上面的标头名称是常量。对于 spring DSL，您必须使用常量的值而不是名称。

150.6. 消息正文

Camel 会将来自外部服务器的 HTTP 响应存储在 OUT 正文上。IN 消息中的所有标头都将复制到 OUT 消息，因此在路由过程中会保留标头。另外，Camel 将添加 HTTP 响应标头以及 OUT 消息标头。

150.7. 响应代码

Camel 将根据 HTTP 响应代码进行处理：

- *响应代码位于 100..299 之间，Camel 会将它视为成功的响应。*
- *响应代码位于范围 300..399 中，Camel 会将它视为重定向响应，并将抛出带有信息的 HttpOperationFailedException。*
- *响应代码为 400+，Camel 会将其视为外部服务器故障，并将抛出带有信息的 HttpOperationFailedException。*

throwExceptionOnFailure

可以选项 throwExceptionOnFailure，它可以设置为 false，以防止为失败的响应代码抛出 HttpOperationFailedException。这样，您可以从远程服务器获得任何响应。
以下示例中提供了演示这一点。

150.8. HTTPOPERATIONFAILEDEXCEPTION

这个例外包括以下信息：

- *HTTP 状态代码*
- *HTTP status 行（状态代码的文本）*
- *重定向位置，如果服务器返回重定向*
- *如果服务器提供了正文作为响应，则响应正文作为 java.lang.String*

150.9. 使用哪个 HTTP 方法

以下算法用于确定应使用的 HTTP 方法：
1。使用作为端点配置提供的方法(httpMethod)。
2.使用标头中提供的方法(Exchange.HTTP_METHOD)。
3.如果标头中提供了查询字符串，GET。
4.GET 如果端点配置了查询字符串。
5.POST 如果有要发送的数据（用户不是 null）。
6.GET 否则。

150.10. 如何访问 HTTPSERVLETREQUEST 和 HTTPSERVLETRESPONSE

您可以使用 Camel 类型转换器系统访问这两个系统

```
HttpServletRequest request = exchange.getIn().getBody(HttpServletRequest.class);
HttpServletRequest response = exchange.getIn().getBody(HttpServletResponse.class);
```

150.11. 使用客户端超时 - SO_TIMEOUT

请参阅 Camel Jetty 存储库中的单元测试

150.12. 更多示例

150.12.1. 配置代理

Java DSL

```
from("direct:start")
  .to("http://oldhost?proxyHost=www.myproxy.com&proxyPort=80");
```

还支持通过 proxyUsername 和 proxyPassword 选项进行代理身份验证。

150.12.2. 使用 URI 外部的代理设置

Java DSL

Spring DSL

```
context.getProperties().put("http.proxyHost", "172.168.18.9");
context.getProperties().put("http.proxyPort" "8080");
```

```
<camelContext>
    <properties>
        <property key="http.proxyHost" value="172.168.18.9"/>
        <property key="http.proxyPort" value="8080"/>
    </properties>
</camelContext>
```

Endpoint 上的选项将覆盖上下文上的选项。

150.13. 配置 CHARSET

如果您使用 POST 来发送数据，您可以配置 charset

```
setProperty(Exchange.CHARSET_NAME, "iso-8859-1");
```

150.14. 带有调度的轮询的示例

示例每 10 秒轮询 Google 主页，并将页面写入文件 message.html ：

```
from("timer://foo?fixedRate=true&delay=0&period=10000")
    .to("http://www.google.com")
    .setHeader(FileComponent.HEADER_FILE_NAME, "message.html").to("file:target/google");
```

150.15. 获取响应代码

您可以通过带有 Exchange.HTTP_RESPONSE_CODE 的 Out message 标头获取 HTTP 响应代码。

```
    Exchange exchange = template.send("http://www.google.com/search", new Processor() {
            public void process(Exchange exchange) throws Exception {
                exchange.getIn().setHeader(Exchange.HTTP_QUERY,
constant("hl=en&q=activemq"));
            }
    });
    Message out = exchange.getOut();
    int responseCode = out.getHeader(Exchange.HTTP_RESPONSE_CODE, Integer.class);
```

150.16. 使用 THROWEXCEPTIONONFAILURE=FALSE 获取任何响应

在以下路由中，我们希望路由一条消息，使我们丰富的数据从远程 HTTP 调用返回。我们希望远程服务器的任何响应，我们将 throwExceptionOnFailure 选项设置为 false，以便在 AggregationStrategy 中获得任何响应。由于代码基于单元测试，模拟 HTTP 状态代码 404，因此有一些断言代码等。

150.17. 禁用 COOKIES

要禁用 Cookie，您可以通过添加此 URI 选项：
httpClient.cookiePolicy=ignoreCookies将 HTTP 客户端设置为忽略 Cookie

150.18. 高级用法

如果您需要对 HTTP 生成者进行更多控制，则应使用 HttpComponent，您可以在其中设置各种类来为您提供自定义行为。

150.18.1. 设置 MaxConnectionsPerHost

HTTP 组件有一个 org.apache.commons.httpclient.HttpConnectionManager，您可以在其中为给定组件配置各种全局配置。
通过全局配置，意味着组件创建的任何端点具有相同的共享 HttpConnectionManager。因此，如果要为每个主机的最大连接设置不同的值，则需要在 HTTP 组件中定义它，而不是在 常用的端点 URI 中定义。这里有：

首先，我们在 Spring XML 中定义 http 组件。是的，我们使用相同的方案名称 http，因为否则 Camel 将自动发现并使用默认设置创建组件。我们需要覆盖这一点，以便我们可以设置我们的选项。在以下示例中，我们将 max 连接设置为 5，而不是默认的 2。

然后我们可以像在我们的路由中一样使用它：

150.18.2. 使用抢占身份验证

最终用户表示他在使用 HTTPS 进行身份验证时遇到问题。当发现 HTTPS 服务器没有返回所需的 HTTP 代码 401 授权时，这个问题最终会被解决。解决方案是设置以下 URI 选项：
httpClient.authenticationPreemptive=true

150.18.3. 接受来自远程服务器的自签名证书

请参阅有关一些代码的邮件列表中的 这个链接，以概述了如何使用 Apache Commons HTTP API 执行此操作。

150.18.4. 为 HTTP 客户端设置 SSL

使用 JSSE 配置实用程序

自 Camel 2.8 起，HTTP4 组件通过 Camel JSSE 配置实用程序支持 SSL/TLS 配置。这个实用程序可大大减少您需要写入的组件特定代码量，并在端点和组件级别进行配置。以下示例演示了如何将 实用程序与 HTTP4 组件一起使用。

此组件中使用的 Apache HTTP 客户端版本从全局"协议" registry 中解析 SSL/TLS 信息。此组件提供 HTTP 客户端的协议套接字工厂的一个实现 org.apache.camel.component.http.SSLContextParametersSecureProtocolSocketFactory，以支持使用 Camel JSSE 配置实用程序。以下示例演示了如何配置协议 registry 并使用路由中的注册协议信息。

```
KeyStoreParameters ksp = new KeyStoreParameters();
ksp.setResource("/users/home/server/keystore.jks");
ksp.setPassword("keystorePassword");
```

```
KeyManagersParameters kmp = new KeyManagersParameters();
kmp.setKeyStore(ksp);
kmp.setKeyPassword("keyPassword");

SSLContextParameters scp = new SSLContextParameters();
scp.setKeyManagers(kmp);

ProtocolSocketFactory factory =
   new SSLContextParametersSecureProtocolSocketFactory(scp);

Protocol.registerProtocol("https",
      new Protocol(
      "https",
      factory,
      443));

from("direct:start")
      .to("https://mail.google.com/mail/").to("mock:results");
```

直接配置 Apache HTTP 客户端

基本上 camel-http 组件基于 Apache HTTP 客户端构建，您可以实施自定义 org.apache.camel.component.http.HttpClientConfigurer 在 http 客户端上进行一些配置，如果您需要完全控制它。

但是，如果您只想 指定密钥存储和信任存储，您可以使用 Apache HTTP HttpClientConfigurer 来执行此操作，例如：

```
Protocol authhttps = new Protocol("https", new AuthSSLProtocolSocketFactory(
  new URL("file:my.keystore"), "mypassword",
  new URL("file:my.truststore"), "mypassword"), 443);

Protocol.registerProtocol("https", authhttps);
```

然后，您需要创建一个实施 HttpClientConfigurer 的类，并注册上述示例提供密钥存储或信任存储的 https 协议。然后，在 camel route builder 类中，您可以按类似以下内容进行 hook：

```
HttpComponent httpComponent = getContext().getComponent("http", HttpComponent.class);
httpComponent.setHttpClientConfigurer(new MyHttpClientConfigurer());
```

如果使用 Spring DSL 执行此操作，您可以使用 URI 指定 HttpClientConfigurer。例如：

```
<bean id="myHttpClientConfigurer"
 class="my.https.HttpClientConfigurer">
```

```
</bean>

<to uri="https://myhostname.com:443/myURL?
httpClientConfigurerRef=myHttpClientConfigurer"/>
```

只要您实施 HttpClientConfigurer 并配置密钥存储和信任存储，它就可以正常工作。

150.19. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- Jetty

第 151 章 HTTP4 组件

从 Camel 版本 2.3 开始提供

http4: 组件提供基于 HTTP 的端点来调用外部 HTTP 资源（作为客户端使用 HTTP 调用外部服务器）。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-http4</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

camel-http4 与 camel-http

camel-http4 使用 Apache HttpClient 4.x，而 camel-http 使用 Apache HttpClient 3.x。

151.1. URI 格式

对于 HTTP

```
http4:hostname[:port][/resourceUri][?options]
```

对于 HTTPS

```
https4:hostname[:port][/resourceUri][?options]
```

默认情况下，HTTP 和 443 将端口 80 用于 HTTPS。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

camel-http4 与 camel-jetty

您只能生成 HTTP4 组件生成的端点。因此，它不应该被用作 Camel 路由中的输入。要通过 HTTP 服务器绑定/公开 HTTP 端点作为 Camel 路由的输入，请使用 Jetty 组件。

151.2. HTTP4 组件选项

HTTP4 组件支持 18 个选项，如下所列。

Name	描述	默认值	类型
httpClientConfigurer (advanced)	若要使用自定义 HttpClientConfigurer 来执行要使用的 HttpClient 配置。		HttpClientConfigurer
clientConnectionManager (advanced)	使用自定义和共享 HttpClientConnectionManager 来管理连接。如果进行了配置，则始终将此端点用于此组件创建的所有端点。		HttpClientConnection Manager
httpContext (advanced)	在执行请求时使用自定义 org.apache.http.protocol.HttpContext。		HttpContext
sslContextParameters (security)	使用 SSLContextParameters 配置安全性。重要： 每个 HttpComponent 支持 org.apache.camel.util.jsse.SSLContextParameters 的一个实例。如果您需要使用 2 个或更多不同的实例，您需要为每个实例定义一个新的 HttpComponent。		SSLContextParameters
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
x509HostnameVerifier (security)	使用自定义 X509HostnameVerifier，如 DefaultHostnameVerifier 或 org.apache.http.conn.ssl.NoopHostnameVerifier。		HostnameVerifier
maxTotalConnections (advanced)	连接的最大数量。	200	int
connectionsPerRoute (advanced)	每个路由的最大连接数。	20	int
connectionTimeToLive (advanced)	连接到 live 的时间，时间单位为 millisecond，默认值为始终保持活动状态。		long

Name	描述	默认值	类型
cookieStore (producer)	使用自定义 org.apache.http.client.CookieStore。默认情况下，使用 org.apache.http.impl.client.BasicCookieStore，它是一个仅限内存的 Cookie 存储。请注意，如果 bridgeEndpoint=true，那么 Cookie 存储被强制成为 noop cookie 存储，因为 Cookie 不应像我们只是桥接一样存储（例如作为代理）。		CookieStore
connectionRequest Timeout (timeout)	从连接管理器请求连接时使用的超时时间（毫秒）。超时值为零解释为无限超时。超时值为零解释为无限超时。负值被解释为 undefined （系统默认值）。默认： code -1	-1	int
connectTimeout (timeout)	决定在建立连接前的超时时间（以毫秒为单位）。超时值为零解释为无限超时。超时值为零解释为无限超时。负值被解释为 undefined （系统默认值）。默认： code -1	-1	int
socketTimeout (timeout)	以毫秒为单位定义套接字超时(SO_TIMEOUT)，这是等待数据的超时时间，或者以不同方式设置，一个最长期限在两个连续数据数据包之间不活跃。超时值为零解释为无限超时。负值被解释为 undefined （系统默认值）。默认： code -1	-1	int
httpBinding (advanced)	使用自定义 HttpBinding 来控制 Camel 消息和 HttpClient 之间的映射。		HttpBinding
httpConfiguration (advanced)	将共享 HttpConfiguration 用作基础配置。		HttpConfiguration
允许 JavaSerialized Object (advanced)	当请求使用 context-type=application/x-java-serialized-object 时，是否允许 java serialization。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

HTTP4 端点使用 URI 语法进行配置：

```
http4:httpUri
```

使用以下路径和查询参数：

151.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
httpUri	**必需** 调用的 HTTP 端点的 url。		URI

151.2.2. 查询参数(49 参数)：

Name	描述	默认值	类型
disableStreamCache (common)	确定 Servlet 中的原始输入流是否被缓存(Camel 将会在 memory/overflow 中读取流到文件，流缓存)缓存。默认情况下，Camel 将缓存 Servlet 输入流，以支持多次读取，以确保其 Camel 可以从流检索所有数据。但是，当您需要访问原始流时，您可以将此选项设置为 true，例如将其直接流传输到文件或其他持久性存储。DefaultHttpBinding 将请求输入流复制到流缓存中，如果此选项为 false，则将它放入消息正文，以支持多次读取流。如果您使用 Servlet 到 bridge/proxy 端点，则请考虑启用此选项来提高性能，如果您不需要多次读取消息有效负载。http/http4 producer 默认缓存响应正文流。如果将此选项设置为 true，则生成者不会缓存响应正文流，而是使用响应流作为消息正文。	false	布尔值
headerFilterStrategy (common)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
httpBinding (common)	使用自定义 HttpBinding 来控制 Camel 消息和 HttpClient 之间的映射。		HttpBinding
authenticationPreemptive (producer)	如果此选项为 true，camel-http4 会将抢占基本身份验证发送到服务器。	false	布尔值
bridgeEndpoint (producer)	如果选项为 true，则 HttpProducer 将忽略 Exchange.HTTP_URI 标头，并使用端点的 URI 进行请求。您也可以将 选项 throwExceptionOnFailure 设置为 false，让 HttpProducer 发送所有错误响应。	false	布尔值
chunked (producer)	如果此选项为 false，则 Servlet 将禁用 HTTP 流，并在响应上设置 content-length 标头	true	布尔值
clearExpiredCookies (producer)	在发送 HTTP 请求前是否清除过期的 Cookie。这样可确保 Cookie 存储不会通过添加新 Cookie （当它们过期时删除）来保持增长。	true	布尔值

Name	描述	默认值	类型
connectionClose (producer)	指定是否应将 Connection Close 标头添加到 HTTP Request 中。默认情况下，connectionClose 为 false。	false	布尔值
cookieStore (producer)	使用自定义 CookieStore。默认情况下，使用 BasicCookieStore，它是一个仅内存的 cookie 存储。请注意，如果 bridgeEndpoint=true，那么 Cookie 存储被强制成为 noop cookie 存储，因为 Cookie 不应像我们只是桥接一样存储（例如作为代理）。如果设置了 CookieHandler，则 Cookie 存储也强制成为 noop cookie 存储，因为 Cookie 处理随后由 cookieHandler 执行。		CookieStore
copyHeaders (producer)	如果此选项为 true，则 IN Exchange 标头将根据复制策略复制到 OUT 交换标头。把它设置为 false，仅允许包含来自 HTTP 响应的标头（而不是传播 IN 标头）。	true	布尔值
deleteWithBody (producer)	HTTP DELETE 是否应该包含消息正文。默认情况下，HTTP DELETE 不包含任何 HTTP 信息。然而，在某些情况下，用户可能需要包含消息正文。	false	布尔值
httpMethod (producer)	配置要使用的 HTTP 方法。如果设置，HttpMethod 标头无法覆盖此选项。		HttpMethods
ignoreResponseBody (producer)	如果此选项为 true，http producer 不会读取响应正文，并缓存输入流	false	布尔值
preserveHostHeader (producer)	如果选项为 true，HttpProducer 会将 Host 标头设置为当前交换主机标头中包含的值，这适用于您希望下游服务器收到的 Host 标头来反映上游客户端调用的 URL 的 URL，这将允许使用 Host 标头为代理服务生成准确的 URL 的应用	false	布尔值
throwExceptionOnFailure (producer)	如果来自远程服务器的失败响应，用于禁用抛出 HttpOperationFailedException。这样，您可以获取所有响应，而不考虑 HTTP 状态代码。	true	布尔值

Name	描述	默认值	类型
transferException (producer)	如果在消费者端启用并交换失败处理，如果原因 Exception 在响应中作为 application/x-java-serialized-object 内容类型发送序列化，则结果为 application/x-java-serialized-object 内容类型。在生产者侧，异常将反序列化并丢弃为原样，而不是 HttpOperationFailedException。原因异常需要按顺序处理。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
cookieHandler (producer)	配置 Cookie 处理程序，以维护 HTTP 会话		CookieHandler
okStatusCodeRange (producer)	被视为成功响应的状态代码。值包含。可以定义多个范围，用逗号分开，例如 200-204,209,301-304。每个范围必须是单个数字，或从 到，其中包含短划线。	200-299	字符串
urlRewrite (producer)	**弃用** 引用自定义 org.apache.camel.component.http.UrlRewrite，它允许您在 bridge/proxy 端点时重写 url。更多信息，请访问 http://camel.apache.org/urlrewrite.html		UrlRewrite
ClientBuilder (advanced)	提供对此端点的生产者或消费者使用的新 RequestConfig 实例上使用的 http 客户端请求参数的访问权限。		HttpClientBuilder
clientConnectionManager (advanced)	使用自定义 HttpClientConnectionManager 管理连接		HttpClientConnection Manager
connectionsPerRoute (advanced)	每个路由的最大连接数。	20	int
httpclient (advanced)	设置一个由制作者使用的自定义 HttpClient		HttpClient
httpClientConfigurer (advanced)	为生成者或消费者创建的新 HttpClient 实例注册自定义配置策略，如配置身份验证机制等		HttpClientConfigurer
httpClientOptions (advanced)	使用映射中的键/值配置 HttpClient：		Map
httpContext (advanced)	使用自定义 HttpContext 实例		HttpContext

Name	描述	默认值	类型
mapHttpMessageBody (advanced)	如果此选项为 true，则交换的 IN Exchange Body 将映射到 HTTP 正文。把它设置为 false 可以避免 HTTP 映射。	true	布尔值
mapHttpMessageFormUrlEncodedBody (advanced)	如果此选项为 true，则交换的 IN Exchange Form Encoded body 将映射到 HTTP。把它设置为 false 可以避免 HTTP Form Encoded body 映射。	true	布尔值
mapHttpMessageHeaders (advanced)	如果此选项为 true，则交换的 IN Exchange Headers 将映射到 HTTP 标头。把它设置为 false 将避免 HTTP 标头映射。	true	布尔值
maxTotalConnections (advanced)	连接的最大数量。	200	int
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
useSystemProperties (advanced)	使用系统属性作为配置的回退	false	布尔值
proxyAuthDomain (proxy)	与 NTML 搭配使用的代理身份验证域		字符串
proxyAuthHost (proxy)	代理身份验证主机		字符串
proxyAuthMethod (proxy)	要使用的代理验证方法		字符串
proxyAuthPassword (proxy)	代理身份验证密码		字符串
proxyAuthPort (proxy)	代理身份验证端口		int
proxyAuthScheme (proxy)	要使用的代理身份验证方案		字符串
proxyAuthUsername (proxy)	代理身份验证用户名		字符串
proxyHost (proxy)	要使用的代理主机名		字符串
proxyPort (proxy)	要使用的代理端口		int

Name	描述	默认值	类型
authDomain (security)	与 NTML 搭配使用的身份验证域		字符串
authHost (security)	与 NTML 搭配使用的身份验证主机		字符串
authmethod (security)	允许将 身份验证方法用作以逗号分开的值列表，即 Basic、Digest 或 NTLM。		字符串
authMethodPriority (security)	要优先使用哪个验证方法，可以是 Basic、Digest 或 NTLM。		字符串
authPassword (security)	身份验证密码		字符串
authUsername (security)	身份验证用户名		字符串
sslContextParameters (security)	使用 SSLContextParameters 配置安全性。重要： 每个 HttpComponent 支持 org.apache.camel.util.jsse.SSLContextParameters 的一个实例。如果您需要使用 2 个或更多不同的实例，您需要为每个实例定义一个新的 HttpComponent。		SSLContextParameters
x509HostnameVerifier (security)	使用自定义 X509HostnameVerifier，如 DefaultHostnameVerifier 或 org.apache.http.conn.ssl.NoopHostnameVerifier。		HostnameVerifier

151.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 19 个选项，如下所列。

Name	描述	默认值	类型
camel.component.http4.allow-java-serialized-object	当请求使用 context-type=application/x-java-serialized-object 时，是否允许 java serialization。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
camel.component.http4.client-connection-manager	使用自定义和共享 HttpClientConnectionManager 来管理连接。如果进行了配置，则始终将此端点用于此组件创建的所有端点。选项是 org.apache.http.conn.HttpClientConnectionManager 类型。		字符串

Name	描述	默认值	类型
camel.component.http4.connect-timeout	决定在建立连接前的超时时间（以毫秒为单位）。超时值为零解释为无限超时。超时值为零解释为无限超时。负值被解释为 undefined （系统默认值）。默认： code -1	-1	整数
camel.component.http4.connection-request-timeout	从连接管理器请求连接时使用的超时时间（毫秒）。超时值为零解释为无限超时。超时值为零解释为无限超时。负值被解释为 undefined （系统默认值）。默认： code -1	-1	整数
camel.component.http4.connection-time-to-live	连接到 live 的时间，时间单位为 millisecond，默认值为始终保持活动状态。		Long
camel.component.http4.connections-per-route	每个路由的最大连接数。	20	整数
camel.component.http4.cookie-store	使用自定义 org.apache.http.client.CookieStore。默认情况下，使用 org.apache.http.impl.client.BasicCookieStore，它是一个仅限内存的 Cookie 存储。请注意，如果 bridgeEndpoint=true，那么 Cookie 存储被强制成为 noop cookie 存储，因为 Cookie 不应像我们只是桥接一样存储（例如作为代理）。选项是 org.apache.http.client.CookieStore 类型。		字符串
camel.component.http4.enabled	启用 http4 组件	true	布尔值
camel.component.http4.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		字符串
camel.component.http4.http-binding	使用自定义 HttpBinding 来控制 Camel 消息和 HttpClient 之间的映射。选项是 org.apache.camel.http.common.HttpBinding 类型。		字符串
camel.component.http4.http-client-configurer	若要使用自定义 HttpClientConfigurer 来执行要使用的 HttpClient 配置。选项是 org.apache.camel.component.http4.HttpClientConfigurer 类型。		字符串
camel.component.http4.http-configuration	将共享 HttpConfiguration 用作基础配置。选项是 org.apache.camel.http.common.HttpConfiguration 类型。		字符串

Name	描述	默认值	类型
camel.component.http4.http-context	在执行请求时使用自定义 org.apache.http.protocol.HttpContext。选项是 org.apache.http.protocol.HttpContext 类型。		字符串
camel.component.http4.max-total-connections	连接的最大数量。	200	整数
camel.component.http4.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.http4.socket-timeout	以毫秒为单位定义套接字超时(SO_TIMEOUT)，这是等待数据的超时时间，或者以不同方式设置，一个最长期限在两个连续数据数据包之间不活跃。超时值为零解释为无限超时。负值被解释为 undefined （系统默认值）。默认： code -1	-1	整数
camel.component.http4.ssl-context-parameters	使用 SSLContextParameters 配置安全性。重要： 每个 HttpComponent 支持 org.apache.camel.util.jsse.SSLContextParameters 的一个实例。如果您需要使用 2 个或更多不同的实例，您需要为每个实例定义一个新的 HttpComponent。选项是 org.apache.camel.util.jsse.SSLContextParameters 类型。		字符串
camel.component.http4.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值
camel.component.http4.x509-hostname-verifier	使用自定义 X509HostnameVerifier，如 DefaultHostnameVerifier 或 org.apache.http.conn.ssl.NoopHostnameVerifier。选项是一个 javax.net.ssl.HostnameVerifier 类型。		字符串

151.4. 消息标头

Name	类型	描述
Exchange.HTTP_URI	**字符串**	要调用的 URI。将覆盖直接在端点上设置的现有 URI。此 uri 是要调用的 http 服务器的 uri。它与 Camel 端点 uri 不同，您可以在其中配置端点选项，如安全等。此标头不支持，它只是 http 服务器的 uri。

Name	类型	描述
Exchange.HTTP_PATH	**字符串**	请求 URI 的路径，标头将使用 HTTP_URI 构建请求 URI。
Exchange.HTTP_QUERY	**字符串**	URI 参数.将覆盖直接在端点上设置的现有 URI 参数。
Exchange.HTTP_RESPONSE_CODE	**int**	来自外部服务器的 HTTP 响应代码。确定为 200。
Exchange.HTTP_RESPONSE_TEXT	**字符串**	来自外部服务器的 HTTP 响应文本。
Exchange.HTTP_CHARACTER_ENCODING	**字符串**	字符编码.
Exchange.CONTENT_TYPE	**字符串**	HTTP 内容类型。在 IN 和 OUT 消息上设置，以提供内容类型，如 **text/html**。
Exchange.CONTENT_ENCODING	**字符串**	HTTP 内容编码。在 IN 和 OUT 消息上设置，以提供内容编码，如 **gzip**。

151.5. 消息正文

Camel 会将来自外部服务器的 HTTP 响应存储在 OUT 正文上。IN 消息中的所有标头都将复制到 OUT 消息，因此在路由过程中会保留标头。另外，Camel 将添加 HTTP 响应标头以及 OUT 消息标头。

151.6. 使用系统属性

当将 useSystemProperties 设置为 true 时，HTTP 客户端将查找以下系统属性，它将使用它：

- ssl.TrustManagerFactory.algorithm
- javax.net.ssl.trustStoreType
- javax.net.ssl.trustStore
- javax.net.ssl.trustStoreProvider
- javax.net.ssl.trustStorePassword
- java.home
- ssl.KeyManagerFactory.algorithm
- javax.net.ssl.keyStoreType
- javax.net.ssl.keyStore
- javax.net.ssl.keyStoreProvider
- javax.net.ssl.keyStorePassword

- *http.proxyHost*
- *http.proxyPort*
- *http.nonProxyHosts*
- *http.keepAlive*
- *http.maxConnections*

151.7. 响应代码

Camel 将根据 HTTP 响应代码进行处理：

- *响应代码位于 100..299 之间，Camel 会将它视为成功的响应。*
- *响应代码位于范围 300..399 中，Camel 会将它视为重定向响应，并将抛出带有信息的 HttpOperationFailedException。*
- *响应代码为 400+，Camel 会将其视为外部服务器故障，并将抛出带有信息的 HttpOperationFailedException。*

throwExceptionOnFailure 选项 throwExceptionOnFailure 可以设置为 false，以防止为失败的响应代码抛出 HttpOperationFailedException。这样，您可以从远程服务器获得任何响应。
以下示例中提供了演示这一点。

151.8. HTTPOPERATIONFAILEDEXCEPTION

这个例外包括以下信息：

- *HTTP 状态代码*

- HTTP status 行（状态代码的文本）
- 重定向位置，如果服务器返回重定向
- 如果服务器提供了正文作为响应，则响应正文作为 java.lang.String

151.9. 使用哪个 HTTP 方法

以下算法用于确定应使用的 HTTP 方法：
1。使用作为端点配置提供的方法(httpMethod)。
2.使用标头中提供的方法(Exchange.HTTP_METHOD)。
3.如果标头中提供了查询字符串，GET。
4.GET 如果端点配置了查询字符串。
5.POST 如果有要发送的数据（用户不是 null）。
6.GET 否则。

151.10. 如何访问 HTTPSERVLETREQUEST 和 HTTPSERVLETRESPONSE

您可以使用 Camel 类型转换器系统访问这两个系统
注意，您只能在 camel-jetty 或 camel-cxf 端点后从处理器中获取请求和响应。

```
HttpServletRequest request = exchange.getIn().getBody(HttpServletRequest.class);
HttpServletRequest response = exchange.getIn().getBody(HttpServletResponse.class);
```

151.11. 将 URI 配置为调用

您可以直接设置 HTTP 制作者的 URI，形成端点 URI。在以下路由中，Camel 将使用 HTTP 调用外部服务器 oldhost。

```
from("direct:start")
        .to("http4://oldhost");
```

以及等同的 Spring 示例：

```
<camelContext xmlns="http://activemq.apache.org/camel/schema/spring">
  <route>
    <from uri="direct:start"/>
```

```
    <to uri="http4://oldhost"/>
  </route>
</camelContext>
```

您可以通过在消息中添加带有密钥 Exchange.HTTP_URI 的标头来覆盖 HTTP 端点 URI。

```
from("direct:start")
  .setHeader(Exchange.HTTP_URI, constant("http://newhost"))
  .to("http4://oldhost");
```

在上例中，尽管端点配置了 http4://oldhost，但以上 Camel 会调用 http://newhost。
如果 http4 端点在网桥模式下工作，它将忽略 Exchange.HTTP_URI 的消息标头。

151.12. 配置 URI 参数

http producer 支持将 URI 参数发送到 HTTP 服务器。URI 参数可以直接在端点 URI 上设置，也可以作为消息上带有密钥 Exchange.HTTP_QUERY 的标头。

```
from("direct:start")
  .to("http4://oldhost?order=123&detail=short");
```

或在标头中提供的选项：

```
from("direct:start")
  .setHeader(Exchange.HTTP_QUERY, constant("order=123&detail=short"))
  .to("http4://oldhost");
```

151.13. 如何将 HTTP 方法(GET/PATCH/POST/PUT/DELETE/HEAD/OPTIONS/TRACE)设置为 HTTP 生成者

使用 http PATCH 方法

从 Camel 2.11.3 / 2.12.1 开始支持 http PATCH 方法。

HTTP4 组件通过设置消息标头来提供设置 HTTP 请求方法的方法。下面是一个示例：

```
from("direct:start")
  .setHeader(Exchange.HTTP_METHOD,
```

```
constant(org.apache.camel.component.http4.HttpMethods.POST))
  .to("http4://www.google.com")
  .to("mock:results");
```

可以使用字符串常量编写方法较短：

```
.setHeader("CamelHttpMethod", constant("POST"))
```

以及等同的 Spring 示例：

```
<camelContext xmlns="http://activemq.apache.org/camel/schema/spring">
  <route>
    <from uri="direct:start"/>
    <setHeader headerName="CamelHttpMethod">
        <constant>POST</constant>
    </setHeader>
    <to uri="http4://www.google.com"/>
    <to uri="mock:results"/>
  </route>
</camelContext>
```

151.14. 使用客户端超时 - SO_TIMEOUT

请参阅 HttpSOTimeoutTest 单元测试。

自 Camel 2.13.0 起：查看更新的 HttpSOTimeoutTest 单元测试。

151.15. 配置代理

HTTP4 组件提供了一种配置代理的方法。

```
from("direct:start")
  .to("http4://oldhost?proxyAuthHost=www.myproxy.com&proxyAuthPort=80");
```

还支持通过 proxyAuthUsername 和 proxyAuthPassword 选项进行代理身份验证。

151.15.1. 使用 URI 外部的代理设置

为了避免系统属性冲突，您只能从 CamelContext 或 URI 设置代理配置。
Java DSL：

```
context.getProperties().put("http.proxyHost", "172.168.18.9");
context.getProperties().put("http.proxyPort" "8080");
```

Spring XML

```
<camelContext>
  <properties>
    <property key="http.proxyHost" value="172.168.18.9"/>
    <property key="http.proxyPort" value="8080"/>
  </properties>
</camelContext>
```

Camel 首先将设置 Java 系统或 CamelContext Properties 中的设置，然后设置端点代理选项（如果提供）。
因此，您可以使用端点选项覆盖系统属性。

请注意，在 Camel 2.8 中，还有一个 http.proxyScheme 属性，您可以将 设置为显式配置要使用的方案。

151.16. 配置 CHARSET

如果您使用 POST 发送数据，您可以使用 Exchange 属性配置 charset ：

```
exchange.setProperty(Exchange.CHARSET_NAME, "ISO-8859-1");
```

151.16.1. 带有调度的轮询的示例

这个示例每 10 秒轮询 Google 主页，并将页面写入文件 message.html ：

```
from("timer://foo?fixedRate=true&delay=0&period=10000")
  .to("http4://www.google.com")
  .setHeader(FileComponent.HEADER_FILE_NAME, "message.html")
  .to("file:target/google");
```

151.16.2. 端点 URI 中的 URI 参数

在本例中，我们有一个完整的 URI 端点，即您在 Web 浏览器中键入的内容。可以使用 & 作为分隔符设置多个 URI 参数，就像您在 Web 浏览器中一样。Camel 这里没有复杂的操作。

```
// we query for Camel at the Google page
template.sendBody("http4://www.google.com/search?q=Camel", null);
```

151.16.3. 消息中的 URI 参数

```
Map headers = new HashMap();
headers.put(Exchange.HTTP_QUERY, "q=Camel&lr=lang_en");
// we query for Camel and English language at Google
template.sendBody("http4://www.google.com/search", null, headers);
```

在上面的标头值中，它不应该带有前缀 ?，您可以使用 & amp; char 分隔参数。

151.16.4. 获取响应代码

您可以通过带有 Exchange.HTTP_RESPONSE_CODE 的 Out message 标头获取 HTTP4 组件的 HTTP 响应代码。

```
Exchange exchange = template.send("http4://www.google.com/search", new Processor() {
  public void process(Exchange exchange) throws Exception {
    exchange.getIn().setHeader(Exchange.HTTP_QUERY, constant("hl=en&q=activemq"));
  }
});
Message out = exchange.getOut();
int responseCode = out.getHeader(Exchange.HTTP_RESPONSE_CODE, Integer.class);
```

151.17. 禁用 COOKIES

要禁用 Cookie，您可以通过添加此 URI 选项： httpClient.cookiePolicy=ignoreCookies将 HTTP 客户端设置为忽略 Cookie

151.18. 高级用法

如果您需要对 HTTP 生成者进行更多控制，则应使用 HttpComponent，您可以在其中设置各种类来为您提供自定义行为。

151.18.1. 为 HTTP 客户端设置 SSL

使用 JSSE 配置实用程序

自 Camel 2.8 起，HTTP4 组件通过 Camel JSSE 配置实用程序支持 SSL/TLS 配置。这个实用程序可

大大减少您需要写入的组件特定代码量，并在端点和组件级别进行配置。以下示例演示了如何将 实用程序与 HTTP4 组件一起使用。

组件的编程配置

```
KeyStoreParameters ksp = new KeyStoreParameters();
ksp.setResource("/users/home/server/keystore.jks");
ksp.setPassword("keystorePassword");

KeyManagersParameters kmp = new KeyManagersParameters();
kmp.setKeyStore(ksp);
kmp.setKeyPassword("keyPassword");

SSLContextParameters scp = new SSLContextParameters();
scp.setKeyManagers(kmp);

HttpComponent httpComponent = getContext().getComponent("https4",
HttpComponent.class);
httpComponent.setSslContextParameters(scp);
```

基于 Spring DSL 的端点配置

```
...
  <camel:sslContextParameters
     id="sslContextParameters">
    <camel:keyManagers
       keyPassword="keyPassword">
      <camel:keyStore
         resource="/users/home/server/keystore.jks"
         password="keystorePassword"/>
    </camel:keyManagers>
  </camel:sslContextParameters>...
...
  <to uri="https4://127.0.0.1/mail/?sslContextParameters=#sslContextParameters"/>...
```

直接配置 Apache HTTP 客户端

基本上 camel-http4 组件基于 Apache HttpClient 构建。有关详细信息，请参阅 SSL/TLS 自定义，或查看 org.apache.camel.component.http4.HttpsServerTestSupport 单元测试基础类。
您还可以实施自定义 org.apache.camel.component.http4.HttpClientConfigurer，以便在 http 客户端上执行一些配置（如果需要完全控制）。

但是，如果您只想 指定密钥存储和信任存储，您可以使用 Apache HTTP HttpClientConfigurer 来执行此操作，例如：

-

```
KeyStore keystore = ...;
KeyStore truststore = ...;

SchemeRegistry registry = new SchemeRegistry();
registry.register(new Scheme("https", 443, new SSLSocketFactory(keystore, "mypassword",
truststore)));
```

然后，您需要创建一个实施 HttpClientConfigurer 的类，并注册上述示例提供密钥存储或信任存储的 https 协议。然后，在 camel route builder 类中，您可以按类似以下内容进行 hook：

```
HttpComponent httpComponent = getContext().getComponent("http4", HttpComponent.class);
httpComponent.setHttpClientConfigurer(new MyHttpClientConfigurer());
```

如果使用 Spring DSL 执行此操作，您可以使用 URI 指定 HttpClientConfigurer。例如：

```
<bean id="myHttpClientConfigurer"
 class="my.https.HttpClientConfigurer">
</bean>

<to uri="https4://myhostname.com:443/myURL?httpClientConfigurer=myHttpClientConfigurer"/>
```

只要您实施 HttpClientConfigurer 并配置密钥存储和信任存储，它就可以正常工作。

使用 HTTPS 验证 gotchas

最终用户表示他在使用 HTTPS 进行身份验证时遇到问题。这个问题最终通过提供自定义配置的 org.apache.http.protocol.HttpContext 来解决此问题：

- 1.为 HttpContexts 创建(Spring)工厂：

```
public class HttpContextFactory {

  private String httpHost = "localhost";
  private String httpPort = 9001;

  private BasicHttpContext httpContext = new BasicHttpContext();
  private BasicAuthCache authCache = new BasicAuthCache();
  private BasicScheme basicAuth = new BasicScheme();

  public HttpContext getObject() {
    authCache.put(new HttpHost(httpHost, httpPort), basicAuth);

    httpContext.setAttribute(ClientContext.AUTH_CACHE, authCache);
```

```
    return httpContext;
  }

  // getter and setter
}
```

- 2.在 Spring 应用上下文文件中声明一个 HttpContext：

```
<bean id="myHttpContext" factory-bean="httpContextFactory" factory-method="getObject"/>
```

- 3.在 http4 URL 中引用上下文：

```
<to uri="https4://myhostname.com:443/myURL?httpContext=myHttpContext"/>
```

使用不同的 SSLContextParameters

HTTP4 组件只支持每个组件的 org.apache.camel.util.jsse.SSLContextParameters 的一个实例。如果您需要使用 2 个或更多不同的实例，则需要设置多个 HTTP4 组件，如下所示。在我们有 2 个组件时，每个组件都使用它们自己的 sslContextParameters 属性实例。

```
<bean id="http4-foo" class="org.apache.camel.component.http4.HttpComponent">
  <property name="sslContextParameters" ref="sslContextParams1"/>
  <property name="x509HostnameVerifier" ref="hostnameVerifier"/>
</bean>

<bean id="http4-bar" class="org.apache.camel.component.http4.HttpComponent">
  <property name="sslContextParameters" ref="sslContextParams2"/>
  <property name="x509HostnameVerifier" ref="hostnameVerifier"/>
</bean>
```

第 152 章 HYSTRY 组件

从 Camel 版本 2.18 开始提供

hystrix 组件将 Netflix Hystrix 断路器集成到 Camel 路由中。

详情请查看 Hystrix EIP 文档。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-hystrix</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

152.1. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.hystrix.mapping.enabled	启用 hystrics 指标 servlet 的自动映射到 Spring Web 上下文。	true	布尔值
camel.component.hystrix.mapping.path	Hystrix 指标 servlet 的端点。	/hystrix.stream	字符串
camel.component.hystrix.mapping.servlet-name	Hystrix 指标 servlet 的名称。	HystrixEventStreamServlet	字符串

第 153 章 ICAL DATAFORMAT

从 Camel 版本 2.12 开始提供

ICal dataformat 用于处理 iCalendar 消息。

典型的 iCalendar 信息如下：

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Events Calendar//iCal4j 1.0//EN
CALSCALE:GREGORIAN
BEGIN:VEVENT
DTSTAMP:20130324T180000Z
DTSTART:20130401T170000
DTEND:20130401T210000
SUMMARY:Progress Meeting
TZID:America/New_York
UID:00000000
ATTENDEE;ROLE=REQ-PARTICIPANT;CN=Developer 1:mailto:dev1@mycompany.com
ATTENDEE;ROLE=OPT-PARTICIPANT;CN=Developer 2:mailto:dev2@mycompany.com
END:VEVENT
END:VCALENDAR
```

153.1. 选项

iCal dataformat 支持 2 个选项，如下所列。

Name	默认值	Java 类型	描述
验证	**false**	布尔值	是否要验证。
contentTypeHeader	**false**	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

153.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.ical.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.ical.enabled	启用 ical dataformat	true	布尔值
camel.dataformat.ical.validating	是否要验证。	false	布尔值

ND

153.3. 基本用法

要对上述消息进行 unmarshal 和 marshal，您的路由将类似如下：

```
from("direct:ical-unmarshal")
  .unmarshal("ical")
  .to("mock:unmarshaled")
  .marshal("ical")
  .to("mock:marshaled");
```

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
   <groupId>org.apache.camel</groupId>
   <artifactId>camel-ical</artifactId>
   <version>x.x.x</version>
   <!-- use the same version as your Camel core version -->
</dependency>
```

153.4. 另请参阅

- *配置 Camel*
- *组件*

- *端点*
- *开始使用*

第 154 章 IEC 60870 CLIENT 组件

从 Camel 版本 2.20 开始提供

IEC 60870-5-104 客户端 组件利用 Eclipse NeoSCADA™ 实施，提供对 IEC 60870 服务器的访问。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-iec60870</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

IEC 60870 客户端组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
defaultConnection Options (common)	默认连接选项		ClientOptions
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

154.1. URI 格式

端点的 URI 语法为：

```
iec60870-client:host:port/00-01-02-03-04
```

信息对象地址用上述语法中的路径进行编码。请注意，始终使用完整的 5 个八位字节地址格式。未使用的八位字节必须填写零。

154.2. URI 选项

IEC 60870 客户端端点使用 URI 语法进行配置：

```
iec60870-client:uriPath
```

使用以下路径和查询参数：

154.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
uriPath	**必需的** 对象信息地址		ObjectAddress

154.2.2. 查询参数(19 参数)：

Name	描述	默认值	类型
dataModuleOptions (common)	数据模块选项		DataModuleOptions
protocolOptions (common)	协议选项		ProtocolOptions
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
acknowledgeWindow (connection)	参数 W - 致用窗口.	10	short

Name	描述	默认值	类型
adsuAddressType (connection)	常见的 ASDU 地址大小。可以是 SIZE_1 或 SIZE_2。		ASDUAddressType
causeOfTransmissionType (connection)	传输类型的原因。可以是 SIZE_1 或 SIZE_2。		CauseOfTransmission Type
informationObjectAddress Type (connection)	信息地址大小。可以是 SIZE_1、SIZE_2 或 SIZE_3。		InformationObjectAddressType
maxUnacknowledged (connection)	参数 K - 最大未确认消息数。	15	short
timeout1 （连接）	以毫秒为单位超时 T1。	15000	int
timeout2 （连接）	以毫秒为单位超时 T2。	10000	int
timeout3 （连接）	以毫秒为单位超时 T3。	20000	int
causeSourceAddress (data)	是否包括源地址	true	布尔值
ignoreBackgroundScan (data)	应忽略后台扫描传输。	true	布尔值
ignoreDaylightSavingTime (data)	是否忽略或遵守 DST	false	布尔值
timezone (data)	要使用的时区。可以是任何 Java 时区字符串	UTC	TimeZone
connectionId (id)	标识符分组连接实例		字符串

154.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.iec60870-client.default-connection-options.ignore-background-scan	应忽略后台扫描传输。	true	布尔值
camel.component.iec60870-client.enabled	是否启用 iec60870-client 组件的自动配置。这默认是启用的。		布尔值
camel.component.iec60870-client.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

如果由 URI 的主机和端口部分标识，则连接实例，外加"id"组中的所有参数。如果遇到连接 id，则会评估连接选项，并使用这些选项创建连接实例。

注意

如果两个 URI 指定相同的连接（主机，端口 ...）但不同的连接选项，则未定义这些连接选项。

最终的连接选项将按以下顺序评估：

- *如果存在，将使用 connectionOptions 参数*
- *否则，在以下步骤中复制和自定义 defaultConnectionOptions 实例*
- *如果存在，应用 protocolOptions*
- *如果存在，应用 dataModuleOptions*

- *应用所有显式连接参数（如 timeZone）*

第 155 章 IEC 60870 服务器组件

从 Camel 版本 2.20 开始提供

IEC 60870-5-104 服务器组件 利用 Eclipse NeoSCADA™ 实施，提供对 IEC 60870 服务器的访问。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-iec60870</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

IEC 60870 服务器组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
defaultConnection Options (common)	默认连接选项		ServerOptions
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

155.1. URI 格式

端点的 URI 语法为：

```
iec60870-server:host:port/00-01-02-03-04
```

信息对象地址用上述语法中的路径进行编码。请注意，始终使用完整的 5 个八位字节地址格式。未使用的八位字节必须填写零。

155.2. URI 选项

IEC 60870 服务器端点使用 URI 语法进行配置：

```
iec60870-server:uriPath
```

使用以下路径和查询参数：

155.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
uriPath	**必需的** 对象信息地址		ObjectAddress

155.2.2. 查询参数(20 参数)：

Name	描述	默认值	类型
dataModuleOptions (common)	数据模块选项		DataModuleOptions
filterNonExecute (common)	过滤掉未设置 execute 位的所有请求	true	布尔值
protocolOptions (common)	协议选项		ProtocolOptions
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

Name	描述	默认值	类型
acknowledgeWindow (connection)	参数 W - 致用窗口.	10	short
adsuAddressType (connection)	常见的 ASDU 地址大小。可以是 SIZE_1 或 SIZE_2。		ASDUAddressType
causeOfTransmissionType (connection)	传输类型的原因。可以是 SIZE_1 或 SIZE_2。		CauseOfTransmission Type
informationObjectAddress Type (connection)	信息地址大小。可以是 SIZE_1、SIZE_2 或 SIZE_3。		InformationObjectAddressType
maxUnacknowledged (connection)	参数 K - 最大未确认消息数。	15	short
timeout1 （连接）	以毫秒为单位超时 T1。	15000	int
timeout2 （连接）	以毫秒为单位超时 T2。	10000	int
timeout3 （连接）	以毫秒为单位超时 T3。	20000	int
causeSourceAddress (data)	是否包括源地址	true	布尔值
ignoreBackgroundScan (data)	应忽略后台扫描传输。	true	布尔值
ignoreDaylightSavingTime (data)	是否忽略或遵守 DST	false	布尔值
timezone (data)	要使用的时区。可以是任何 Java 时区字符串	UTC	TimeZone
connectionId (id)	标识符分组连接实例		字符串

155.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 7 个选项，如下所列。

Name	描述	默认值	类型
camel.component.iec60870-server.default-connection-options.background-scan-period	后台传输周期之间的"ms"中的句点。 如果将其设置为零或更少，则会禁用后台传输。		整数
camel.component.iec60870-server.default-connection-options.booleans-with-timestamp	使用时间戳发送布尔值		布尔值
camel.component.iec60870-server.default-connection-options.buffering-period	协议层中的时间周期将缓冲更改事件，以发送聚合更改消息		整数
camel.component.iec60870-server.default-connection-options.floats-with-timestamp	使用时间戳发送浮点数		布尔值
camel.component.iec60870-server.default-connection-options.spontaneous-duplicates	要保留在缓冲区中的 spontaneous 事件数量。 当缓冲区中的事件数量超过这个数量时，事件将被丢弃，以保持缓冲区大小。		整数
camel.component.iec60870-server.enabled	是否启用 iec60870-server 组件的自动配置。这默认是启用的。		布尔值
camel.component.iec60870-server.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 156 章 IGNITE CACHE 组件

从 Camel 版本 2.17 开始提供

Ignite Cache 端点是 camel-ignite 端点之一，允许您与 Ignite Cache 交互。这同时提供了 Producer（对 Ignite 缓存调用缓存操作），以及一个 Consumer （消耗来自持续查询的更改）。

缓存值始终是消息的正文，而 cache 键始终存储在 IgniteConstants.IGNITE_CACHE_KEY 消息标头中。

即使您在端点 URI 中配置固定操作，您也可以通过设置 IgniteConstants.IGNITE_CACHE_OPERATION 消息标头而有所不同。

156.1. 选项

Ignite Cache 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
Ignite (common)	设置 Ignite 实例。		Ignite
configurationResource (common)	设置从中加载配置的资源。它可以是：URI、String (URI)或 InputStream。		对象
igniteConfiguration (common)	允许用户设置编程 IgniteConfiguration。		IgniteConfiguration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Ignite Cache 端点使用 URI 语法进行配置：

```
ignite-cache:cacheName
```

使用以下路径和查询参数：

156.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	**所需的** 缓存名称。		字符串

156.1.2. 查询参数(16 参数)：

Name	描述	默认值	类型
propagateIncomingBodyIfNoReturnValue (common)	如果底层 Ignite 操作的返回类型是 void，则设置是否传播传入的正文。	true	布尔值
treatCollectionsAsCache Objects (common)	设置是否将集合视为缓存对象或项目集合，以插入/更新/计算等。	false	布尔值
autoUnsubscribe (consumer)	是否在 Continuous Query Consumer 中启用自动取消订阅。	true	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
fireExistingQueryResults (consumer)	是否处理与查询匹配的现有结果。用于初始化持续 Query Consumer。	false	布尔值
oneExchangePerUpdate (consumer)	是否在单个 Exchange 中打包每个更新，即使一个批处理中收到多个更新。仅供cont Query Consumer 使用。	true	布尔值
pageSize (consumer)	页面大小。仅供cont Query Consumer 使用。	1	int
query (consumer)	Query to execute，只适用于需要它的操作，以及 continuous Query Consumer。		查询
remoteFilter (consumer)	远程过滤器，仅由 Continuous Query Consumer 使用。		CacheEntryEvent SerializableFilter
timeInterval (consumer)	持续 Query Consumer 的时间间隔。	0	long

Name	描述	默认值	类型
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
cachePeekMode (producer)	CachePeekMode，只适用于需要它的操作 (IgniteCacheOperation 7.3SIZE)。	ALL	CachePeekMode
failIfInexistentCache (producer)	如果缓存不存在，是否初始化失败。	false	布尔值
operation (producer)	要调用的缓存操作。可能的值有： GET、PUT、REMOVE、SIZE、REBALANCE、QUERY、CLEAR。		IgniteCacheOperation
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

156.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.component.ignite-cache.configuration-resource	设置从中加载配置的资源。它可以是：URI、String (URI)或 InputStream。选项是一个 java.lang.Object 类型。		字符串
camel.component.ignite-cache.enabled	启用 ignite-cache 组件	true	布尔值
camel.component.ignite-cache.ignite	设置 Ignite 实例。选项是 org.apache.ignite.Ignite 类型。		字符串
camel.component.ignite-cache.ignite-configuration	允许用户设置编程 IgniteConfiguration。选项是 org.apache.ignite.configuration.IgniteConfiguration 类型。		字符串

Name	描述	默认值	类型
camel.component.ignite-cache.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

156.2.1. 使用的标头

此端点使用以下标头：

标头名称	常数	预期类型	描述
CamelIgniteCacheKey	IgniteConstants.IGNITE_CACHE_KEY	字符串	消息正文中条目值的缓存键。
CamelIgniteCacheQuery	IgniteConstants.IGNITE_CACHE_QUERY	查询	调用 QUERY 操作时要运行的查询(producer)。
CamelIgniteCacheOperation	IgniteConstants.IGNITE_CACHE_OPERATION	IgniteCacheOperation enum	允许您动态更改要执行(producer)的缓存操作。
CamelIgniteCachePeekMode	IgniteConstants.IGNITE_CACHE_PEEK_MODE	CachePeekMode enum	允许您在运行 SIZE 操作时动态更改缓存 peek 模式。
CamelIgniteCacheEventType	IgniteConstants.IGNITE_CACHE_EVENT_TYPE	int (EventType constants)	在使用持续查询消费者时，此标头会传输收到的事件类型。
CamelIgniteCacheName	IgniteConstants.IGNITE_CACHE_NAME	字符串	此标头传输接收持续查询事件的缓存名称(consumer)。它不允许您动态更改执行生成者操作的缓存。对此使用 EIP （如接收者列表、动态路由器）。

标头名称	常数	预期类型	描述
CamelIgniteCacheOldValue	IgniteConstants.IGNITE_CACHE_OLD_VALUE	对象	在传入的缓存事件(consumer)中传递时，此标头会传输旧的缓存值。

第 157 章 IGNITE COMPUTE 组件

从 Camel 版本 2.17 开始提供

Ignite Compute 端点是 camel-ignite 端点之一，它允许您通过传递 IgniteCallable、IgniteRunnable、IgniteRunnable 或它们的集合来在集群中运行 计算操作。

此端点仅支持生成者。

端点 URI 的主机部分是一个符号链接端点 ID，它不用于任何目的。

端点尝试运行作为计算作业在 IN 消息正文中传递的对象。它需要不同的有效负载类型，具体取决于执行类型。

157.1. 选项

Ignite Compute 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
Ignite (producer)	设置 Ignite 实例。		Ignite
configurationResource (producer)	设置从中加载配置的资源。它可以是：URI、String (URI)或 InputStream。		对象
igniteConfiguration (producer)	允许用户设置编程 IgniteConfiguration。		IgniteConfiguration
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Ignite Compute 端点使用 URI 语法进行配置：

```
ignite-compute:endpointId
```

使用以下路径和查询参数：

157.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
endpointId	**必需** 端点 ID （未使用）。		字符串

157.1.2. 查询参数(8 参数)：

Name	描述	默认值	类型
clusterGroupExpression (producer)	返回 IgniteCompute 实例的 Cluster Group 的表达式。		ClusterGroupExpression
computeName (producer)	计算作业的名称，它将通过 IgniteCompute#withName (String)进行设置。		字符串
executionType (producer)	**需要** 执行计算操作。可能的值有：CALL, BROADCAST, APPLY, EXECUTE, RUN, AFFINITY_CALL, AFFINITY_RUN.组件需要不同的有效负载类型，具体取决于操作。		IgniteComputeExecution Type
propagateIncomingBodyIfNoreturnValue (producer)	如果底层 Ignite 操作的返回类型是 void，则设置是否传播传入的正文。	true	布尔值
taskName (producer)	任务名称，仅在使用 IgniteComputeExecutionType#EXECUTE 执行类型时才适用。		字符串
timeoutMillis (producer)	触发的作业超时间隔（以毫秒为单位），该间隔将通过 IgniteComputeAllwithTimeout (long)设置。		Long
treatCollectionsAsCache Objects (producer)	设置是否将集合视为缓存对象或项目集合，以插入/更新/计算等。	false	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

157.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.component.ignite-compute.configuration-resource	设置从中加载配置的资源。它可以是：URI、String (URI)或 InputStream。选项是一个 java.lang.Object 类型。		字符串
camel.component.ignite-compute.enabled	启用 ignite-compute 组件	true	布尔值
camel.component.ignite-compute.ignite	设置 Ignite 实例。选项是 org.apache.ignite.Ignite 类型。		字符串
camel.component.ignite-compute.ignite-configuration	允许用户设置编程 IgniteConfiguration。选项是 org.apache.ignite.configuration.IgniteConfiguration 类型。		字符串
camel.component.ignite-compute.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

157.2.1. 预期的有效负载类型

每个操作都需要指定的类型：

操作	预期有效负载
调用	IgniteCallable 或单个 IgniteCallable 的集合。
BROADCAST	IgniteCallable, IgniteRunnable, IgniteClosure.
应用	IgniteClosure.
执行	如果 taskName 选项不是 null，则 ComputeTask、class<? extends ComputeTask> 或代表参数的对象。
运行	A Collection of IgniteRunnables, 或一个 IgniteRunnable。

操作	预期有效负载
AFFINITY_CALL	IgniteCallable.
AFFINITY_RUN	IgniteRunnable。

157.2.2. 使用的标头

此端点使用以下标头：

标头名称	常数	预期类型	描述
CamelIgniteComputeExecutionType	IgniteConstants.IGNITE_COMPUTE_EXECUTION_TYPE	IgniteComputeExecutionType enum	允许您动态更改计算操作。
CamelIgniteComputeParameters	IgniteConstants.IGNITE_COMPUTE_PARAMS	对象或集合的任何对象或集合。	APPLY、BROADCAST 和 EXECUTE 操作的参数.
CamelIgniteComputeReducer	IgniteConstants.IGNITE_COMPUTE_REDUCER	IgniteReducer	减少 APPLY 和 CALL 操作。
CamelIgniteComputeAffinityCacheName	IgniteConstants.IGNITE_COMPUTE_AFFINITY_CACHE_NAME	字符串	AFFINITY_CALL 和 AFFINITY_RUN 操作的关联性缓存名称。
CamelIgniteComputeAffinityKey	IgniteConstants.IGNITE_COMPUTE_AFFINITY_KEY	对象	AFFINITY_CALL 和 AFFINITY_RUN 操作的关联性密钥。

第 158 章 IGNITE EVENTS 组件

从 Camel 版本 2.17 开始提供

Ignite Events 端点是 camel-ignite 端点之一，允许您通过创建本地事件监听程序从 Ignite 集群接收事件。

此端点仅支持使用者。此使用者创建的 Exchange 会将接收的事件对象放入 IN 消息的正文中。

158.1. 选项

Ignite Events 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
Ignite (consumer)	设置 Ignite 实例。		Ignite
configurationResource (consumer)	设置从中加载配置的资源。它可以是：URI、String (URI)或 InputStream。		对象
igniteConfiguration (consumer)	允许用户设置编程 IgniteConfiguration。		IgniteConfiguration
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Ignite Events 端点使用 URI 语法进行配置：

```
ignite-events:endpointId
```

使用以下路径和查询参数：

158.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
endpointId	端点 ID （未使用）。		字符串

158.1.2. 查询参数(8 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
clusterGroupExpression (consumer)	集群组表达式。		ClusterGroupExpression
Events (consumer)	要作为 Set 订阅的事件 ID，其中 ID 是 org.apache.ignite.events.EventType 中的不同常量。	EventType.EVTS_ALL	SetOrString
propagateIncomingBodyIfNoreturnValue (consumer)	如果底层 Ignite 操作的返回类型是 void，则设置是否传播传入的正文。	true	布尔值
treatCollectionsAsCache Objects (consumer)	设置是否将集合视为缓存对象或项目集合，以插入/更新/计算等。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

158.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.component.ignite-events.configuration-resource	设置从中加载配置的资源。它可以是：URI、String (URI)或 InputStream。选项是一个 java.lang.Object 类型。		字符串
camel.component.ignite-events.enabled	启用 ignite-events 组件	true	布尔值
camel.component.ignite-events.ignite	设置 Ignite 实例。选项是 org.apache.ignite.Ignite 类型。		字符串
camel.component.ignite-events.ignite-configuration	允许用户设置编程 IgniteConfiguration。选项是 org.apache.ignite.configuration.IgniteConfiguration 类型。		字符串
camel.component.ignite-events.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 159 章 IGNITE ID GENERATOR 组件

从 Camel 版本 2.17 开始提供

Ignite ID Generator 端点是 camel-ignite 端点之一，它允许您与 Ignite Atomic Sequences 和 ID Generators 交互。

此端点仅支持生成者。

159.1. 选项

Ignite ID Generator 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
Ignite (producer)	设置 Ignite 实例。		Ignite
configurationResource (producer)	设置从中加载配置的资源。它可以是：URI、String (URI)或 InputStream。		对象
igniteConfiguration (producer)	允许用户设置编程 IgniteConfiguration。		IgniteConfiguration
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Ignite ID Generator 端点使用 URI 语法进行配置：

```
ignite-idgen:name
```

使用以下路径和查询参数：

159.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	**必需的** 序列名称。		字符串

159.1.2. 查询参数(6 参数)：

Name	描述	默认值	类型
batchSize (producer)	批处理大小。		整数
initialValue (producer)	初始值。	0	Long
operation (producer)	对 Ignite ID 生成器调用的操作。被 IN 消息中的 IgniteConstants.IGNITE_IDGEN_OPERATION 标头取代。可能的值有： ADD_AND_GET、GET_AND_ADD、GET_AND_INCREMENT、INCREMENT_AND_GET。		IgniteIdGenOperation
propagateIncomingBodyIfNoreturnValue (producer)	如果底层 Ignite 操作的返回类型是 void，则设置是否传播传入的正文。	true	布尔值
treatCollectionsAsCache Objects (producer)	设置是否将集合视为缓存对象或项目集合，以插入/更新/计算等。	false	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

159.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.component.ignite-idgen.configuration-resource	设置从中加载配置的资源。它可以是：URI、String (URI)或 InputStream。选项是一个 java.lang.Object 类型。		字符串
camel.component.ignite-idgen.enabled	启用 ignite-idgen 组件	true	布尔值

Name	描述	默认值	类型
camel.component.ignite-idgen.ignite	设置 Ignite 实例。选项是 org.apache.ignite.Ignite 类型。		字符串
camel.component.ignite-idgen.ignite-configuration	允许用户设置编程 IgniteConfiguration。选项是 org.apache.ignite.configuration.IgniteConfiguration 类型。		字符串
camel.component.ignite-idgen.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 160 章 IGNITE MESSAGING 组件

从 Camel 版本 2.17 开始提供

Ignite Messaging 端点是 camel-ignite 端点之一，它允许您发送和使用来自 Ignite 主题的消息。

此端点支持生成者（发送消息）和消费者（以接收消息）。

160.1. 选项

Ignite Messaging 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
Ignite (common)	设置 Ignite 实例。		Ignite
configurationResource (common)	设置从中加载配置的资源。它可以是：URI、String (URI)或 InputStream。		对象
igniteConfiguration (common)	允许用户设置编程 IgniteConfiguration。		IgniteConfiguration
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Ignite Messaging 端点使用 URI 语法进行配置：

```
ignite-messaging:topic
```

使用以下路径和查询参数：

160.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
topic	**必需** The topic name。		字符串

160.1.2. 查询参数(9 参数)：

Name	描述	默认值	类型
propagateIncomingBodyIfNoReturnValue (common)	如果底层 Ignite 操作的返回类型是 void，则设置是否传播传入的正文。	true	布尔值
treatCollectionsAsCache Objects (common)	设置是否将集合视为缓存对象或项目集合，以插入/更新/计算等。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
clusterGroupExpression (producer)	集群组表达式。		ClusterGroupExpression
sendMode (producer)	要使用的 send 模式。可能的值：UNORDERED、ORDERED.	UNORDERED	IgniteMessagingSend 模式
timeout (producer)	使用排序消息时发送操作的超时时间。		Long
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

160.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.component.ignite-messaging.configuration-resource	设置从中加载配置的资源。它可以是：URI、String (URI)或 InputStream。选项是一个 java.lang.Object 类型。		字符串
camel.component.ignite-messaging.enabled	启用 ignite-messaging 组件	true	布尔值
camel.component.ignite-messaging.ignite	设置 Ignite 实例。选项是 org.apache.ignite.Ignite 类型。		字符串
camel.component.ignite-messaging.ignite-configuration	允许用户设置编程 IgniteConfiguration。选项是 org.apache.ignite.configuration.IgniteConfiguration 类型。		字符串
camel.component.ignite-messaging.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

160.2.1. 使用的标头

此端点使用以下标头：

标头名称	常数	预期类型	描述
CamelIgniteMessagingTopic	IgniteConstants.IGNITE_MESSAGING_TOPIC	字符串	允许您动态更改主题，以将消息发送到(producer)。它还执行接收消息的主题(consumer)。
CamelIgniteMessagingUUID	IgniteConstants.IGNITE_MESSAGING_UUID	UUID	当消息到达(consumer)时，此标头会填写订阅的 UUID。

第 161 章 IGNITE QUEUES 组件

从 Camel 版本 2.17 开始提供

Ignite Queue 端点是 camel-ignite 端点之一，允许您与 Ignite Queue 数据结构 交互。

此端点仅支持生成者。

161.1. 选项

Ignite Queues 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
Ignite (producer)	设置 Ignite 实例。		Ignite
configurationResource (producer)	设置从中加载配置的资源。它可以是：URI、String (URI)或 InputStream。		对象
igniteConfiguration (producer)	允许用户设置编程 IgniteConfiguration。		IgniteConfiguration
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Ignite Queues 端点使用 URI 语法进行配置：

```
ignite-queue:name
```

使用以下路径和查询参数：

161.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	**必需** 队列名称。		字符串

161.1.2. 查询参数(7 参数)：

Name	描述	默认值	类型
capacity (producer)	队列容量。默认 ：非绑定。		int
配置 （生成者）	集合配置。默认：空配置。您还可以使用 configuration.xyz=123 选项方便地设置内部属性。		CollectionConfiguration
operation (producer)	在 Ignite Queue 上调用的操作。被 IN 消息中的 IgniteConstants.IGNITE_QUEUE_OPERATION 标头取代。可能的值有： CONTAINS、ADD、SIZE、REMOVE、ITERATOR、CLEAR、RETAIN_ALL、ARRAY、DRAIN、ELEMENT、PEEK、OFFER、POLL、TAKE、PUT.		IgniteQueueOperation
propagateIncomingBodyIfNoreturnValue (producer)	如果底层 Ignite 操作的返回类型是 void，则设置是否传播传入的正文。	true	布尔值
timeoutMillis (producer)	队列超时（以毫秒为单位）。默认：无超时。		Long
treatCollectionsAsCache Objects (producer)	设置是否将集合视为缓存对象或项目集合，以插入/更新/计算等。	false	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

161.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.component.ignite-queue.configuration-resource	设置从中加载配置的资源。它可以是：URI、String (URI)或 InputStream。选项是一个 java.lang.Object 类型。		字符串
camel.component.ignite-queue.enabled	启用 ignite-queue 组件	true	布尔值

Name	描述	默认值	类型
camel.component.ignite-queue.ignite	设置 Ignite 实例。选项是 org.apache.ignite.Ignite 类型。		字符串
camel.component.ignite-queue.ignite-configuration	允许用户设置编程 IgniteConfiguration。选项是 org.apache.ignite.configuration.IgniteConfiguration 类型。		字符串
camel.component.ignite-queue.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

161.2.1. 使用的标头

此端点使用以下标头：

标头名称	常数	预期类型	描述
CamelIgniteQueueOperation	IgniteConstants.IGNITE_QUEUE_OPERATION	IgniteQueueOperation enum	允许您动态更改队列操作。
CamelIgniteQueueMaxElements	IgniteConstants.IGNITE_QUEUE_MAX_ELEMENTS	整数或 int	在调用 DRAIN 操作时，要排空的项目数量。
CamelIgniteQueueTransferredCount	IgniteConstants.IGNITE_QUEUE_TRANSFERRED_COUNT	整数或 int	DRAIN 操作后传输的项目数量。
CamelIgniteQueueTimeoutMillis	IgniteConstants.IGNITE_QUEUE_TIMEOUT_MILLIS	long 或 long	在调用 OFFER 或 POLL 操作时，动态设置要使用的超时（毫秒）。

第 162 章 IGNITE SETS 组件

从 Camel 版本 2.17 开始提供

Ignite Sets 端点是 camel-ignite 端点之一，它允许您与 Ignite Set 数据结构交互。

此端点仅支持生成者。

162.1. 选项

Ignite Sets 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
Ignite (producer)	设置 Ignite 实例。		Ignite
configurationResource (producer)	设置从中加载配置的资源。它可以是：URI、String (URI)或 InputStream。		对象
igniteConfiguration (producer)	允许用户设置编程 IgniteConfiguration。		IgniteConfiguration
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Ignite Sets 端点使用 URI 语法进行配置：

```
ignite-set:name
```

使用以下路径和查询参数：

162.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	**必需** 设定名称。		字符串

162.1.2. 查询参数(5 参数)：

Name	描述	默认值	类型
配置 （生成者）	集合配置。默认：空配置。您还可以使用 configuration.xyz=123 选项方便地设置内部属性。		CollectionConfiguration
operation (producer)	在 Ignite Set 中调用的操作。被 IN 消息中的 IgniteConstants.IGNITE_SETS_OPERATION 标头取代。可能的值有： CONTAINS、ADD、SIZE、REMOVE、ITERATOR、CLEAR、RETAIN_ALL、ARRAY。		IgniteSetOperation
propagateIncomingBodyIfNoreturnValue (producer)	如果底层 Ignite 操作的返回类型是 void，则设置是否传播传入的正文。	true	布尔值
treatCollectionsAsCache Objects (producer)	设置是否将集合视为缓存对象或项目集合，以插入/更新/计算等。	false	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

162.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.component.ignite-set.configuration-resource	设置从中加载配置的资源。它可以是：URI、String (URI)或 InputStream。选项是一个 java.lang.Object 类型。		字符串
camel.component.ignite-set.enabled	启用 ignite-set 组件	true	布尔值
camel.component.ignite-set.ignite	设置 Ignite 实例。选项是 org.apache.ignite.Ignite 类型。		字符串
camel.component.ignite-set.ignite-configuration	允许用户设置编程 IgniteConfiguration。选项是 org.apache.ignite.configuration.IgniteConfiguration 类型。		字符串

Name	描述	默认值	类型
camel.component.ignite-set.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

162.2.1. 使用的标头

此端点使用以下标头：

标头名称	常数	预期类型	描述
CamelIgniteSetsOperation	IgniteConstants.IGNITE_SETS_OPERATION	IgniteSetOperation enum	允许您动态更改 set 操作。

第 163 章 INFINISPAN 组件

从 Camel 版本 2.13 开始提供

此组件允许您与 Infinispan 分布式数据网格/缓存进行交互。Infinispan 是一个高度可扩展的、高度可用的键/值数据存储和使用 Java 编写的数据网格平台。

从 Camel 2.17 开始，Infinispan 需要 Java 8。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-infinispan</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

163.1. URI 格式

```
infinispan://cacheName?[options]
```

163.2. URI 选项

生产者允许使用 HotRod 协议将消息发送到 registry 中配置的本地 infinispan 缓存。使用者允许侦听可从 registry 访问的本地 infinispan 缓存中的事件。

Infinispan 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	端点之间共享的默认配置。		InfinispanConfiguration
cacheContainer (common)	默认缓存容器。		BasicCacheContainer

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Infinispan 端点使用 URI 语法进行配置：

```
infinispan:cacheName
```

使用以下路径和查询参数：

163.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	**要使用的** 缓存		字符串

163.2.2. 查询参数(18 参数)：

Name	描述	默认值	类型
hosts (common)	指定 Infinispan 实例上的缓存主机		字符串
queryBuilder (common)	指定查询构建器。		InfinispanQueryBuilder
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
clusteredListener (consumer)	如果为 true，则会为整个集群安装监听程序	false	布尔值
command (consumer)	**弃用了** 要执行的操作。	PUT	字符串
customListener (consumer)	如果提供，返回自定义监听程序（如果提供）		InfinispanCustomListener

Name	描述	默认值	类型
eventTypes (consumer)	指定要由消费者注册的事件类型集。可以使用逗号分隔多个事件。可能的事件类型是：CACHE_ENTRY_ACTIVATED, CACHE_ENTRY_PASSIVATED, CACHE_ENTRY_VISITED, CACHE_ENTRY_LOADED, CACHE_ENTRY_EVICTED, CACHE_ENTRY_CREATED, CACHE_ENTRY_REMOVED, CACHE_ENTRY_MODIFIED, TRANSACTION_COMPLETED, TRANSACTION_REGISTERED, CACHE_ENTRY_INVALIDATED, DATA_REHASHED, TOPOLOGY_CHANGED, PARTITION_STATUS_CHANGED		字符串
sync (consumer)	如果为 true，则消费者将以同步方式接收通知	true	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
operation (producer)	要执行的操作。	PUT	InfinispanOperation
cacheContainer (advanced)	指定要连接的缓存容器		BasicCacheContainer
cacheContainerConfiguration (advanced)	CacheContainer 配置。如果没有定义 cacheContainer，则使用。必须是以下类型：org.infinispan.client.hotrod.configuration.Configuration - 用于远程缓存交互配置；org.infinispan.configuration.cache.Configuration - 用于嵌入式缓存交互配置；		对象
configurationProperties (advanced)	为 CacheManager 实现特定属性		Map
configurationUri (advanced)	CacheManager 的实现特定 URI		字符串
标记 (advanced)	每个缓存调用中默认应用逗号分隔的标记列表，不适用于远程缓存。		字符串

Name	描述	默认值	类型
resultHeader (advanced)	将操作结果存储在标头中，而不是消息正文。默认情况下，resultHeader == null，查询结果存储在消息正文中，消息正文中的任何现有内容都会被丢弃。如果设置了 resultHeader，则值将用作标头名称来存储查询结果，并保留原始消息正文。这个值可以被名为 CamelInfinispanOperationResultHeader 的消息标头中覆盖		对象
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

163.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 21 个选项，如下所列。

Name	描述	默认值	类型
camel.component.infinispan.cache-container	默认缓存容器。选项是一个 org.infinispan.commons.api.BasicCacheContainer 类型。		字符串
camel.component.infinispan.configuration.cache-container	指定要连接的缓存容器		BasicCacheContainer
camel.component.infinispan.configuration.cache-container-configuration	CacheContainer 配置。如果没有定义 cacheContainer，则使用。必须是以下类型：org.infinispan.client.hotrod.configuration.Configuration - 用于远程缓存交互配置；org.infinispan.configuration.cache.Configuration - 用于嵌入式缓存交互配置；		对象
camel.component.infinispan.configuration.clustered-listener	如果为 true，则会为整个集群安装监听程序	false	布尔值
camel.component.infinispan.configuration.configuration-properties	为 CacheManager 实现特定属性		Map

Name	描述	默认值	类型
camel.component.infinispan.configuration.configuration-uri	CacheManager 的实现特定 URI		字符串
camel.component.infinispan.configuration.custom-listener	如果提供，返回自定义监听程序（如果提供）		InfinispanCustomListener
camel.component.infinispan.configuration.event-types	指定要由消费者注册的事件类型集。可以使用逗号分隔多个事件。可能的事件类型是：CACHE_ENTRY_ACTIVATED, CACHE_ENTRY_PASSIVATED, CACHE_ENTRY_VISITED, CACHE_ENTRY_LOADED, CACHE_ENTRY_EVICTED, CACHE_ENTRY_CREATED, CACHE_ENTRY_REMOVED, CACHE_ENTRY_MODIFIED, TRANSACTION_COMPLETED, TRANSACTION_REGISTERED, CACHE_ENTRY_INVALIDATED, DATA_REHASHED, TOPOLOGY_CHANGED, PARTITION_STATUS_CHANGED		Set
camel.component.infinispan.configuration.flags	每个缓存调用中默认应用逗号分隔的标记列表，不适用于远程缓存。		flag[]
camel.component.infinispan.configuration.hosts	指定 Infinispan 实例上的缓存主机		字符串
camel.component.infinispan.configuration.operation	要执行的操作。		InfinispanOperation
camel.component.infinispan.configuration.query-builder	指定查询构建器。		InfinispanQueryBuilder

Name	描述	默认值	类型
camel.component.infinispan.configuration.result-header	将操作结果存储在标头中，而不是消息正文。默认情况下，resultHeader == null，查询结果存储在消息正文中，消息正文中的任何现有内容都会被丢弃。如果设置了 resultHeader，则值将用作标头名称来存储查询结果，并保留原始消息正文。这个值可以被名为 CamelInfinispanOperationResultHeader 的消息标头中覆盖		对象
camel.component.infinispan.configuration.sync	如果为 true，则消费者将以同步方式接收通知	true	布尔值
camel.component.infinispan.customizer.embedded-cache-manager.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component.infinispan.customizer.embedded-cache-manager.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component.infinispan.customizer.remote-cache-manager.enabled	启用或禁用 cache-manager customizer。	true	布尔值
camel.component.infinispan.customizer.remote-cache-manager.override	配置组件上最终设置的缓存管理器是否应被自定义器覆盖。	false	布尔值
camel.component.infinispan.enabled	启用 infinispan 组件	true	布尔值
camel.component.infinispan.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.infinispan.configuration.command	要执行的操作。	PUT	字符串

Name	描述	默认值	类型

163.4. 消息标头

Name	默认值	类型	Context	描述
CamelInfinispanCacheName	**null**	字符串	共享	参与操作或事件的缓存。
CamelInfinispanOperation	**PUT**	InfinispanOperation	制作者	要执行的操作。
CamelInfinispanMap	**null**	Map	制作者	CamelInfinispanOperationPutAll 操作时要使用的映射
CamelInfinispanKey	**null**	对象	共享	对生成事件的密钥或密钥。
CamelInfinispanValue	**null**	对象	制作者	用于操作的值。
CamelInfinispanEventType	**null**	字符串	消费者	接收的事件的类型。此处 org.infinispan.notifications.cachelistener.event.Event.Type 中定义的可能值
CamelInfinispanIsPre	**null**	布尔值	消费者	Infinispan 会为每个操作触发两个事件：在操作前和一个事件。
CamelInfinispanLifespanTime	**null**	long	制作者	缓存内值的 Lifespan 时间。负值被解释为 infinity。

Name	默认值	类型	Context	描述
CamelInfinispanTimeUnit	**null**	字符串	制作者	条目 Lifespan Time 的时间单位.
CamelInfinispanMaxIdleTime	**null**	long	制作者	在条目被视为过期前允许为 闲置的最长时间。
CamelInfinispanMaxIdleTimeUnit	**null**	字符串	制作者	条目的最大空闲时间的时间单位.
CamelInfinispanQueryBuilder	null	InfinispanQueryBuilder	制作者	**从 Camel 2.17：** 如果命令默认为 InifinispanConfiguration，则用于 QUERY 命令的 QueryBuilde
CamelInfinispanIgnoreReturnValues	null	布尔值	制作者	**从 Camel 2.17：** 如果设置了此标头，则客户端应用程序忽略缓存操作的返回值
CamelInfinispanOperationResultHeader	null	字符串	制作者	**在 Camel 2.20 中：** 将操作结果保存在标头中，而不是消息正文

163.5. 例子

- ***使用自定义缓存容器从默认缓存检索特定密钥：***

```
from("direct:start")
   .setHeader(InfinispanConstants.OPERATION).constant(InfinispanOperation.GET)
   .setHeader(InfinispanConstants.KEY).constant("123")
   .to("infinispan?cacheContainer=#cacheContainer");
```

- 从命名的缓存中检索特定密钥：

```
from("direct:start")
   .setHeader(InfinispanConstants.OPERATION).constant(InfinispanOperation.PUT)
   .setHeader(InfinispanConstants.KEY).constant("123")
   .to("infinispan:myCacheName");
```

- 使用 lifespan 替换值

```
from("direct:start")
   .setHeader(InfinispanConstants.OPERATION).constant(InfinispanOperation.GET)
   .setHeader(InfinispanConstants.KEY).constant("123")
   .setHeader(InfinispanConstants.LIFESPAN_TIME).constant(100L)

.setHeader(InfinispanConstants.LIFESPAN_TIME_UNIT.constant(TimeUnit.MILLISECO
NDS.toString())
   .to("infinispan:myCacheName");
```

- 使用带有额外参数（主机、端口和协议版本）的缓存容器配置从远程缓存检索特定密钥：

```
org.infinispan.client.hotrod.configuration.Configuration cacheContainerConfiguration
= new org.infinispan.client.hotrod.configuration.ConfigurationBuilder()
   .addServer()
      .host("localhost")
      .port(9999)
      .version(org.infinispan.client.hotrod.ProtocolVersion.PROTOCOL_VERSION_25)
   .build();
...

from("direct:start")
   .setHeader(InfinispanConstants.OPERATION).constant(InfinispanOperation.GET)
   .setHeader(InfinispanConstants.KEY).constant("123")
   .to("infinispan?cacheContainerConfiguration=#cacheContainerConfiguration");
```

163.6. 使用基于 INFINISPAN 的幂等存储库

在本节中，我们将使用基于 Infinispan 的幂等存储库。

首先，我们需要创建一个 cacheManager，然后配置我们的

```
org.apache.camel.component.infinispan.processor.idempotent.InfinispanIdempotentReposito
ry:
```

```
<!-- set up the cache manager -->
<bean id="cacheManager"
      class="org.infinispan.manager.DefaultCacheManager"
      init-method="start"
      destroy-method="stop"/>

<!-- set up the repository -->
<bean id="infinispanRepo"

class="org.apache.camel.component.infinispan.processor.idempotent.InfinispanIdempotentRepository"

      factory-method="infinispanIdempotentRepository">
    <argument ref="cacheManager"/>
    <argument value="idempotent"/>
</bean>
```

然后，我们可以在 spring XML 文件中创建 Infinispan idempotent 存储库：

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
    <route id="JpaMessageIdRepositoryTest">
        <from uri="direct:start" />
        <idempotentConsumer messageIdRepositoryRef="infinispanStore">
            <header>messageId</header>
            <to uri="mock:result" />
        </idempotentConsumer>
    </route>
</camelContext>
```

163.7. 使用基于 INFINISPAN 的路由策略

163.8. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 164 章 INFLUXDB COMPONENT

从 Camel 版本 2.18 开始提供

这个组件允许您与 InfluxDB https://influxdata.com/time-series-platform/influxdb/ 一个时间序列数据库进行交互。此组件的原生正文类型是 Point （原生 influxdb 类），但它也可以接受 Map<String, Object> 作为消息正文，它将被转换为 Point.class，请注意映射必须包含带有 InfluxbConstants.MEASUREMENT_NAME 的元素。

平均而言，您可以将自己的转换器注册到数据类型中，或使用 camel 提供的(un) marshalling 工具。

从 Camel 2.18 开始，Influxdb 需要 Java 8。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-influxdb</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

164.1. URI 格式

```
influxdb://beanName?[options]
```

164.2. URI 选项

生产者允许使用原生 java 驱动程序将消息发送到 registry 中配置的 influxdb。

InfluxDB 组件没有选项。

InfluxDB 端点使用 URI 语法进行配置：

```
influxdb:connectionBean
```

使用以下路径和查询参数：

164.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
connectionBean	**需要连接到** influx 数据库，类 InfluxDB.class		字符串

164.2.2. 查询参数(6 参数)：

Name	描述	默认值	类型
batch (producer)	定义此操作是否为批处理操作	false	布尔值
databaseName (producer)	存储时间序列的数据库名称		字符串
operation (producer)	定义此操作是否为插入或查询	insert	字符串
query (producer)	在操作查询时定义查询		字符串
retentionPolicy (producer)	将保留策略定义为端点创建的数据的字符串	default	字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

164.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.influxdb.enabled	启用 influxdb 组件	true	布尔值
camel.component.influxdb.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

164.4. 消息标头

Name	默认值	类型	Context	描述

164.5. EXAMPLE

以下是将点存储到 db （使用 URI 中 db 名称）特定键的示例路由：

```
from("direct:start")
        .setHeader(InfluxDbConstants.DBNAME_HEADER, constant("myTimeSeriesDB"))
        .to("influxdb://connectionBean);
```

```
from("direct:start")
        .to("influxdb://connectionBean?databaseName=myTimeSeriesDB");
```

如需更多信息，请参阅这些资源...

164.6. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 165 章 IPFS COMPONENT

从 Camel 版本 2.23 开始提供

ipfs: 组件提供对 Interplanetary 文件系统 (IPFS) 的访问。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-ipfs</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

165.1. URI 格式

```
ipfs://cmd?options
```

165.2. 选项

IPFS 组件没有选项。

IPFS 端点使用 URI 语法进行配置：

```
ipfs:host:port/cmd
```

使用以下路径和查询参数：

165.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
ipfsCmd	ipfs 命令		字符串

165.2.2. 查询参数(2 参数)：

Name	描述	默认值	类型
outdir (producer)	ipfs 输出目录		路径
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

165.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.ipfs.enabled	是否启用 ipfs 组件的自动配置。这默认是启用的。		布尔值
camel.component.ipfs.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

165.4. KARAF 支持

actually 该组件在 Karaf 中不受支持

165.5. 消息标头

<TODO><title>Samples</title>

在本例中，我们向 IPFS 添加一个文件，从 IPFS 获取文件，最后访问 IPFS 文件的内容。

```
from("direct:start").to("ipfs:add")
from("direct:start").to("ipfs:get?outdir=target")
from("direct:start").to("ipfs:cat");
```

</TODO>

第 166 章 IRC 组件

从 Camel 版本 1.1 开始提供

irc 组件实现了 IRC (Internet Relay Chat)传输。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-irc</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

166.1. URI 格式

```
irc:nick@host[:port]/#room[?options]
irc:nick@host[:port]?channels=#channel1,#channel2,#channel3[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

166.2. 选项

IRC 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

IRC 端点使用 URI 语法进行配置：

```
irc:hostname:port
```

使用以下路径和查询参数：

166.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
hostname	IRC chat 服务器 **所需的** 主机名		字符串
port	IRC chat 服务器的端口号。如果没有配置端口，则使用 6667、6668 或 6669 的默认端口。		int

166.2.2. 查询参数(25 参数)：

Name	描述	默认值	类型
autoRejoin (common)	启动时是否自动加入	true	布尔值
commandTimeout (common)	连接建立后发送命令前的延迟（毫秒）。	5000	long
namesOnJoin (common)	在加入后将 NAMES 命令发送到频道。onReply 需要为 true，以便处理结果，其将具有标头值 irc.num = '353'。	false	布尔值
nickname (common)	chat 中使用的别名。		字符串
persistent (common)	**弃用的** 使用持久性消息。	true	布尔值
realName (common)	IRC 用户的实际名称。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler

Name	描述	默认值	类型
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
颜色 （高级）	服务器是否支持颜色代码。	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
onJoin (filter)	处理用户加入事件。	true	布尔值
onKick (filter)	处理启动事件。	true	布尔值
onMode (filter)	处理模式更改事件。	true	布尔值
onNick (filter)	处理别名更改事件。	true	布尔值
onpart (filter)	处理用户部分事件。	true	布尔值
onPrivmsg (filter)	处理私有消息事件。	true	布尔值
onQuit (filter)	处理用户退出事件。	true	布尔值
onReply (filter)	是否处理命令的常规响应或信息信息。	false	布尔值
onTopic (filter)	处理主题更改事件。	true	布尔值
nickPassword (security)	您的 IRC 服务器别名密码。		字符串
密码 (security)	IRC 服务器密码。		字符串
sslContextParameters (security)	用于通过 SSL 配置安全性。对 Registry 中的 org.apache.camel.util.jsse.SSLContextParameters 的引用。此引用覆盖组件级别上配置的 SSLContextParameters。请注意，此设置会覆盖 trustManager 选项。		SSLContextParameters
trustmanager (security)	用于验证 SSL 服务器的证书的信任管理器。		SSLTrustManager
用户名 (security)	IRC 服务器用户名。		字符串

166.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.irc.enabled	启用 irc 组件	true	布尔值
camel.component.irc.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.irc.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

166.4. SSL 支持

166.4.1. 使用 JSSE 配置实用程序

自 Camel 2.9 起，IRC 组件通过 Camel JSSE 配置实用程序支持 SSL/TLS 配置。这个实用程序可大大减少您需要写入的组件特定代码量，并在端点和组件级别进行配置。以下示例演示了如何将 实用程序与 IRC 组件一起使用。

端点的编程配置

```
KeyStoreParameters ksp = new KeyStoreParameters();
ksp.setResource("/users/home/server/truststore.jks");
ksp.setPassword("keystorePassword");

TrustManagersParameters tmp = new TrustManagersParameters();
tmp.setKeyStore(ksp);

SSLContextParameters scp = new SSLContextParameters();
scp.setTrustManagers(tmp);

Registry registry = ...
registry.bind("sslContextParameters", scp);

...

from(...)
    .to("ircs://camel-prd-user@server:6669/#camel-test?nickname=camel-prd&password=password&sslContextParameters=#sslContextParameters");
```

基于 Spring DSL 的端点配置

```
...
  <camel:sslContextParameters
      id="sslContextParameters">
    <camel:trustManagers>
      <camel:keyStore
          resource="/users/home/server/truststore.jks"
          password="keystorePassword"/>
    </camel:keyManagers>
  </camel:sslContextParameters>...
...
  <to uri="ircs://camel-prd-user@server:6669/#camel-test?nickname=camel-
prd&password=password&sslContextParameters=#sslContextParameters"/>...
```

166.4.2. 使用旧的基本配置选项

您还可以连接到启用 SSL 的 IRC 服务器，如下所示：

```
ircs:host[:port]/#room?username=user&password=pass
```

默认情况下，IRC 传输使用 SSLDefaultTrustManager。如果您需要提供自己的自定义信任管理器，请使用 trustManager 参数，如下所示：

```
ircs:host[:port]/#room?
username=user&password=pass&trustManager=#referenceToMyTrustManagerBean
```

166.5. 使用密钥

从 Camel 2.2 开始提供

有些 irc 房需要您提供一个密钥来加入该频道。键只是一个 secret 单词。

例如，我们加入 3 个频道，其中只有频道 1 和 3 使用密钥。

```
irc:nick@irc.server.org?channels=#chan1,#chan2,#chan3&keys=chan1Key,,chan3key
```

166.6. 获取频道用户列表

使用 nameOnJoin 选项，可以在组件加入频道后调用 IRC-NAMES 命令。服务器将使用 irc.num = 353 回复。因此，为了处理结果，Reply 必须为 true。另外，还需要过滤 onReply Exchanges 才能获取名称。

例如，我们希望获取包含频道用户名的所有交换：

```
from("ircs:nick@myserver:1234/#mychannelname?namesOnJoin=true&onReply=true")
 .choice()
  .when(header("irc.messageType").isEqualToIgnoreCase("REPLY"))
   .filter(header("irc.num").isEqualTo("353"))
   .to("mock:result").stop();
```

166.7. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 167 章 JACKSONXML DATAFORMAT

从 Camel 版本 2.16 开始提供

Jackson XML 是一个数据格式，它使用带有 XMLMapper 扩展的 Jackson 库来 将 XML 有效负载 unmarshal 到 Java 对象，或将 Java 对象 marshal Java 对象放入 XML 有效负载中。

INFO：如果您熟悉 Jackson，这个 XML 数据格式的行为与其 JSON 对应部分的行为相同，因此可与为 JSON 序列化/序列化注解的类一起使用。

此扩展也模仿 JAXB 的"代码先代码"方法。

此数据格式依赖于 Woodstox （特别是对于用户打印等功能），这是一个快速高效的 XML 处理器。

```
from("activemq:My.Queue").
  unmarshal().jacksonxml().
  to("mqseries:Another.Queue");
```

167.1. JACKSONXML OPTIONS

JacksonXML dataformat 支持 15 个选项，如下所列。

Name	默认值	Java 类型	描述
xmlMapper		字符串	查找并使用具有给定 ID 的现有 XmlMapper。
prettyPrint	false	布尔值	要启用用户的打印输出，请执行以下操作：默认为 false。
unmarshalTypeName		字符串	取消警报时要使用的 java 类型的类名称
jsonView		类	当将 POJO 聚合到 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。此选项是引用具有 JsonView 注释的类
Include		字符串	如果您要将 pojo 放入 JSON，并且 pojo 有一些带有 null 值的字段。如果您想要跳过这些 null 值，您可以将这个选项设置为 NON_NULL

Name	默认值	Java 类型	描述
allowJmsType	false	布尔值	用于 JMS 用户，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。
collectionTypeName		字符串	引用要在要使用的 registry 中的自定义集合类型。这个选项很少被使用，但允许使用不同的集合类型，而不是基于 java.util.Collection 作为默认值。
useList	false	布尔值	要取消警报到映射列表或 Pojo 列表。
enableJaxbAnnotationModule	false	布尔值	使用 jackson 时是否启用 JAXB 注释模块。启用后，Jackson 可以使用 JAXB 注释。
moduleClassNames		字符串	要使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String 带有 FQN 类名称。可以使用逗号分隔多个类。
moduleRefs		字符串	使用 Camel registry 引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。
enableFeatures		字符串	要在 Jackson com.fasterxml.jackson.databind.ObjectMapper 中启用的功能集。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称
disableFeatures		字符串	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的一组功能。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称
allowUnmarshallType	false	布尔值	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。只有在需要使用时，才会启用。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

167.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 16 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.jacksonxml.allow-jms-type	用于 JMS 用户，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。	false	布尔值
camel.dataformat.jacksonxml.allow-unmarshall-type	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。只有在需要使用时，才会启用。	false	布尔值
camel.dataformat.jacksonxml.collection-type-name	引用要在要使用的 registry 中的自定义集合类型。这个选项很少被使用，但允许使用不同的集合类型，而不是基于 java.util.Collection 作为默认值。		字符串
camel.dataformat.jacksonxml.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.jacksonxml.disable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的一组功能。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称		字符串
camel.dataformat.jacksonxml.enable-features	要在 Jackson com.fasterxml.jackson.databind.ObjectMapper 中启用的功能集。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称		字符串
camel.dataformat.jacksonxml.enable-jaxb-annotation-module	使用 jackson 时是否启用 JAXB 注释模块。启用后，Jackson 可以使用 JAXB 注释。	false	布尔值
camel.dataformat.jacksonxml.enabled	启用 jacksonxml dataformat	true	布尔值
camel.dataformat.jacksonxml.include	如果您要将 pojo 放入 JSON，并且 pojo 有一些带有 null 值的字段。如果您想要跳过这些 null 值，您可以将这个选项设置为 NON_NULL		字符串

Name	描述	默认值	类型
camel.dataformat.jacksonxml.json-view	当将 POJO 聚合到 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。此选项是引用具有 JsonView 注释的类		类
camel.dataformat.jacksonxml.module-class-names	要使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String 带有 FQN 类名称。可以使用逗号分隔多个类。		字符串
camel.dataformat.jacksonxml.module-refs	使用 Camel registry 引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。		字符串
camel.dataformat.jacksonxml.pretty-print	要启用用户的打印输出，请执行以下操作：默认为 false。	false	布尔值
camel.dataformat.jacksonxml.unmarshal-type-name	取消警报时要使用的 java 类型的类名称		字符串
camel.dataformat.jacksonxml.use-list	要取消警报到映射列表或 Pojo 列表。	false	布尔值
camel.dataformat.jacksonxml.xml-mapper	查找并使用具有给定 ID 的现有 XmlMapper。		字符串

ND

167.2.1. 在 Spring DSL 中使用 Jackson XML

在 Spring DSL 中使用 Data Format 时，您需要首先声明数据格式。这是在 DataFormats XML 标签中完成的。

```
    <dataFormats>
      <!-- here we define a Xml data format with the id jack and that it should use the TestPojo as the class type when
           doing unmarshal. The unmarshalTypeName is optional, if not provided Camel will use a Map as the type -->
      <jacksonxml id="jack" unmarshalTypeName="org.apache.camel.component.jacksonxml.TestPojo"/>
    </dataFormats>
```

然后您可以在路由中引用此 id：

```
    <route>
        <from uri="direct:back"/>
        <unmarshal ref="jack"/>
        <to uri="mock:reverse"/>
     </route>
```

167.3. 从 MARSHALLING 排除 POJO 字段

当将 POJO 聚合到 XML 时，您可能希望从 XML 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。首先创建一个或多个标记类。

将标记类与 @JsonView 注释一起使用，以包含/排除某些字段。该注释也适用于 getters。

最后，使用 Camel JacksonXMLDataFormat marshall POJO 到 XML。

请注意，生成的 XML 中缺少 weight 字段：

```
<pojo age="30" weight="70"/>
```

167.4. 使用 JSONVIEW 属性和 'JACKSONXML'DATAFORMAT 的 INCLUDE/EXCLUDE 字段

例如，您可以使用此属性而不是：

```
JacksonXMLDataFormat ageViewFormat = new JacksonXMLDataFormat(TestPojoView.class,
Views.Age.class);
from("direct:inPojoAgeView").
  marshal(ageViewFormat);
```

直接指定 Java DSL 中的 JSON 视图，如下所示：

```
from("direct:inPojoAgeView").
  marshal().jacksonxml(TestPojoView.class, Views.Age.class);
```

在 XML DSL 中相同：

```
<from uri="direct:inPojoAgeView"/>
  <marshal>
    <jacksonxml unmarshalTypeName="org.apache.camel.component.jacksonxml.TestPojoView"
jsonView="org.apache.camel.component.jacksonxml.Views$Age"/>
  </marshal>
```

167.5. 设置序列化 INCLUDE 选项

如果您要将 pojo 放入 XML，并且 pojo 有一些带有 null 值的字段。并且您要跳过这些 null 值，然后您需要在 pojo 上设置注解，

```
@JsonInclude(Include.NON_NULL)
public class MyPojo {
   ...
}
```

但这需要您在 pojo 源代码中包含该注解。您还可以配置 Camel JacksonXMLDataFormat 来设置 include 选项，如下所示：

```
JacksonXMLDataFormat format = new JacksonXMLDataFormat();
format.setInclude("NON_NULL");
```

或者从 XML DSL 中将其配置为

```
  <dataFormats>
    <jacksonxml id="jacksonxml" include="NON_NULL"/>
  </dataFormats>
```

167.6. 使用动态类名称从 XML 迁移到 POJO

如果您使用 jackson unmarshal XML 到 POJO，那么您现在可以在消息中指定一个标头，以指示要 unmarshal 的类名称。
如果消息中存在该标头，则标头键为 CamelJacksonUnmarshalType，则 Jackson 将使用该键作为 POJO 类的 FQN，以便按以下方式处理 XML 有效负载。

```
 For JMS end users there is the JMSType header from the JMS spec that
indicates that also. To enable support for JMSType you would need to
turn that on, on the jackson data format as shown:
```

```
JacksonDataFormat format = new JacksonDataFormat();
format.setAllowJmsType(true);
```

或者从 XML DSL 中将其配置为

```
<dataFormats>
  <jacksonxml id="jacksonxml" allowJmsType="true"/>
</dataFormats>
```

167.7. 从 XML 到 LIST<MAP> 或 LIST<POJO>

如果您使用 Jackson 将 Jackson 用于映射/pojo 列表，您现在可以通过设置 useList="true" 或使用 org.apache.camel.component.jacksonxml.ListJacksonXMLDataFormat 来指定。例如，您可以使用 Java，如下所示：

```
JacksonXMLDataFormat format = new ListJacksonXMLDataFormat();
// or
JacksonXMLDataFormat format = new JacksonXMLDataFormat();
format.useList();
// and you can specify the pojo class type also
format.setUnmarshalType(MyPojo.class);
```

如果使用 XML DSL，您可以使用 useList 属性将列表配置为使用 list，如下所示：

```
<dataFormats>
  <jacksonxml id="jack" useList="true"/>
</dataFormats>
```

此外，您还可以指定 pojo 类型

```
<dataFormats>
  <jacksonxml id="jack" useList="true" unmarshalTypeName="com.foo.MyPojo"/>
</dataFormats>
```

167.8. 使用自定义 JACKSON 模块

您可以使用 moduleClassNames 选项指定它们的类名称，以使用自定义 Jackson 模块，如下所示。

```
<dataFormats>
  <jacksonxml id="jack" useList="true" unmarshalTypeName="com.foo.MyPojo"
moduleClassNames="com.foo.MyModule,com.foo.MyOtherModule"/>
</dataFormats>
```

当使用 moduleClassNames 时，不会配置自定义 jackson 模块，方法是使用默认构造器创建并使用

as-is。如果自定义模块需要任何自定义配置，则可以创建和配置模块实例，然后使用 moduleRefs 引用该模块，如下所示：

```
<bean id="myJacksonModule" class="com.foo.MyModule">
  ... // configure the module as you want
</bean>

<dataFormats>
  <jacksonxml id="jacksonxml" useList="true" unmarshalTypeName="com.foo.MyPojo"
moduleRefs="myJacksonModule"/>
</dataFormats>
```

可以使用逗号分隔多个模块，如 moduleRefs="myJacksonModule,myOtherModule"

167.9. 使用 JACKSON 启用或禁用功能

Jackson 具有很多功能，您可以启用或禁用其 ObjectMapper 使用的功能。例如，要在 marshalling 时禁用未知属性失败，您可以使用 disableFeatures 进行配置：

```
<dataFormats>
  <jacksonxml id="jacksonxml" unmarshalTypeName="com.foo.MyPojo"
disableFeatures="FAIL_ON_UNKNOWN_PROPERTIES"/>
</dataFormats>
```

您可以通过使用逗号分隔值来禁用多个功能。功能的值必须是来自以下 enum 类的 Jackson 的 enums 的名称

- com.fasterxml.jackson.databind.SerializationFeature
- com.fasterxml.jackson.databind.DeserializationFeature
- com.fasterxml.jackson.databind.MapperFeature

要启用功能，请使用 enableFeatures 选项。

在 Java 代码中，您可以使用 camel-jackson 模块中的类型安全方法：

```
JacksonDataFormat df = new JacksonDataFormat(MyPojo.class);
df.disableFeature(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES);
df.disableFeature(DeserializationFeature.FAIL_ON_NULL_FOR_PRIMITIVES);
```

167.10. 使用 JACKSON 将映射转换为 POJO

Jackson ObjectMapper 可用于将映射转换为 POJO 对象。Jackson 组件附带数据转换器，可用于将 java.util.Map 实例转换为非字符串、非原语和非数字对象。

```
Map<String, Object> invoiceData = new HashMap<String, Object>();
invoiceData.put("netValue", 500);
producerTemplate.sendBody("direct:mapToInvoice", invoiceData);
...
// Later in the processor
Invoice invoice = exchange.getIn().getBody(Invoice.class);
```

如果 Camel registry 中只有一个 ObjectMapper 实例，它将供转换器用于执行转换。否则将使用默认映射程序。

167.11. 格式化的 XML MARSHALLING (PRETTY-PRINTING)

使用 prettyPrint 选项，您可以在 marshalling 时输出良好格式化的 XML：

```
<dataFormats>
    <jacksonxml id="jack" prettyPrint="true"/>
</dataFormats>
```

Java DSL 中：

```
from("direct:inPretty").marshal().jacksonxml(true);
```

请注意，有 5 个不同的过载 jacksonxml （） DSL 方法，它支持 prettyPrint 选项以及 unmarshalType、 jsonView 等其他设置。

167.12. 依赖项

要在 camel 路由中使用 Jackson XML，您需要对实现此数据格式的 camel-jacksonxml 添加依赖项。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅

最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jacksonxml</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

第 168 章 JASYPT 组件

从 Camel 2.5 开始提供

Jasypt 是一个简化的加密库，使加密和解密容易。Camel 与 Jasypt 集成，以允许 加密 属性文件中的敏感信息。通过在类路径上丢弃 camel-jasypt，这些加密值将自动由 Camel 解密。这样可保证人为人关注的不能轻松发现敏感信息，如用户名和密码。

如果使用 Maven，则需要将以下依赖项添加到此组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jasypt</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

如果使用 Apache Karaf 容器，则需要将以下依赖项添加到此组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.karaf.jaas</groupId>
  <artifactId>org.apache.karaf.jaas.jasypt</artifactId>
  <version>x.x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

168.1. 工具

Jasypt 组件提供一些命令行工具来加密或解密值。

控制台输出语法及其提供的选项：

```
Apache Camel Jasypt takes the following options

  -h or -help = Displays the help screen
  -c or -command <command> = Command either encrypt or decrypt
  -p or -password <password> = Password to use
  -i or -input <input> = Text to encrypt or decrypt
  -a or -algorithm <algorithm> = Optional algorithm to use
```

例如，使用以下参数来 加密您运行的值：在 apache camel kit 中，您 cd 进入 lib 文件夹并运行以下

java cmd，其中 & lt;CAMEL_HOME > 是您下载并提取 Camel 发行版的位置。

```
$ cd <CAMEL_HOME>/lib
$ java -jar camel-jasypt-2.5.0.jar -c encrypt -p secret -i tiger
```

输出以下结果

```
Encrypted text: qaEEacuW7BUti8LcMgyjKw==
```

这意味着，如果您知道 master 密码是 secret，则加密表示 qaEEacuW7BUti8LcMgyjKw== 可以将其解密回 tiger。
如果您再次运行该工具，则加密值将返回不同的结果。但是解密该值将始终返回正确的原始值。

因此，您可以使用以下参数运行工具来测试它：

```
$ cd <CAMEL_HOME>/lib
$ java -jar camel-jasypt-2.5.0.jar -c decrypt -p secret -i qaEEacuW7BUti8LcMgyjKw==
```

输出以下结果：

```
Decrypted text: tiger
```

然后，在属性文件中使用这些 加密值。注意密码值如何加密，值具有与 ENC （这里的值）相关的令牌。

提示

在运行 jasypt 工具时，如果您出现 java.lang.NoClassDefFoundError: org/jasypt/encryption/pbe/StandardPBEStringEncryptor，这意味着您必须在 classpath 中包含 jasypt7.13.jar。如果您要以 java -jar … 运行，将 jar 添加到 classpath 的示例可将 jasypt7.13.jar 复制到 $JAVA_HOME\jre\lib\ext。后者可能将 jasypt7.13.jar 添加到 classpath using -cp。在这种情况下，您应该提供主类来作为 eg: java -cp jasypt-1.9.2.jar:camel-jasypt-2.18.2.jar org.apache.camel.component.jasypt.Main -c encrypt -p secret -i tiger

168.2. URI 选项

以下选项专用于 Jasypt 组件。

Name	默认值	类型	描述
password	null	字符串	指定用于解密的 master 密码。这个选项是必须的。详情请查看以下。
algorithm	null	字符串	要使用的可选算法的名称。

168.3. 保护 MASTER 密码

必须提供 Jasypt 使用的主密码，以便它可以解密这些值。但是，将此 master 密码处于开放状态可能不是理想的解决方案。因此，例如，您可以将它作为 JVM 系统属性或 OS 环境设置提供。如果您决定这样做，则 password 选项支持指定前缀的前缀。sysenv：使用给定键查找 OS 系统环境。sys ：用于查找 JVM 系统属性。

例如，您可以在启动应用程序前提供密码

```
$ export CAMEL_ENCRYPTION_PASSWORD=secret
```

然后，启动应用，如运行启动脚本。

当应用程序启动并运行时，您可以取消设置环境

```
$ unset CAMEL_ENCRYPTION_PASSWORD
```

password 选项随后定义如下：password=sysenv:CAMEL_ENCRYPTION_PASSWORD。

168.4. 使用 JAVA DSL 的示例

在 Java DSL 中，您需要将 Jasypt 配置为 JasyptPropertiesParser 实例，并在 Properties 组件上设置它，如下所示：

属性文件 myproperties.properties 然后包含加密值，如下所示。注意密码值如何加密，值具有与 ENC （这里的值）相关的令牌。

168.5. SPRING XML 示例

在 Spring XML 中，您需要配置 JasyptPropertiesParser，它如下所示。然后，会告知 Camel Properties 组件使用 jasypt 作为属性解析器，这意味着 Jasypt 有机会解密值在属性中查找。

```
<!-- define the jasypt properties parser with the given password to be used -->
<bean id="jasypt" class="org.apache.camel.component.jasypt.JasyptPropertiesParser">
    <property name="password" value="secret"/>
</bean>

<!-- define the camel properties component -->
<bean id="properties" class="org.apache.camel.component.properties.PropertiesComponent">
    <!-- the properties file is in the classpath -->
    <property name="location"
value="classpath:org/apache/camel/component/jasypt/myproperties.properties"/>
    <!-- and let it leverage the jasypt parser -->
    <property name="propertiesParser" ref="jasypt"/>
</bean>
```

Properties 组件也可以在 < camelContext> 标签中内联，如下所示。注意我们如何使用 propertiesParserRef 属性来引用 Jasypt。

```
<!-- define the jasypt properties parser with the given password to be used -->
<bean id="jasypt" class="org.apache.camel.component.jasypt.JasyptPropertiesParser">
    <!-- password is mandatory, you can prefix it with sysenv: or sys: to indicate it should use
        an OS environment or JVM system property value, so you dont have the master
password defined here -->
    <property name="password" value="secret"/>
</bean>

<camelContext xmlns="http://camel.apache.org/schema/spring">
    <!-- define the camel properties placeholder, and let it leverage jasypt -->
    <propertyPlaceholder id="properties"

location="classpath:org/apache/camel/component/jasypt/myproperties.properties"
                         propertiesParserRef="jasypt"/>
    <route>
        <from uri="direct:start"/>
        <to uri="{{cool.result}}"/>
    </route>
</camelContext>
```

168.6. 带有 BLUEPRINT XML 的示例

在蓝图 XML 中，您需要配置 JasyptPropertiesParser，它如下所示。然后，会告知 Camel Properties 组件使用 jasypt 作为属性解析器，这意味着 Jasypt 有机会解密值在属性中查找。

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:cm="http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.0.0"
```

```
        xsi:schemaLocation="
        http://www.osgi.org/xmlns/blueprint/v1.0.0
http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd">

  <cm:property-placeholder id="myblue" persistent-id="mypersistent">
    <!-- list some properties for this test -->
    <cm:default-properties>
      <cm:property name="cool.result" value="mock:{{cool.password}}"/>
      <cm:property name="cool.password" value="ENC(bsW9uV37gQ0QHFu7KO03Ww==)"/>
    </cm:default-properties>
  </cm:property-placeholder>

  <!-- define the jasypt properties parser with the given password to be used -->
  <bean id="jasypt" class="org.apache.camel.component.jasypt.JasyptPropertiesParser">
    <property name="password" value="secret"/>
  </bean>

  <camelContext xmlns="http://camel.apache.org/schema/blueprint">
    <!-- define the camel properties placeholder, and let it leverage jasypt -->
    <propertyPlaceholder id="properties"
                         location="blueprint:myblue"
                         propertiesParserRef="jasypt"/>
      <route>
        <from uri="direct:start"/>
        <to uri="{{cool.result}}"/>
      </route>
  </camelContext>

</blueprint>
```

Properties *组件也可以在* **< camelContext>** *标签中内联，如下所示。注意我们如何使用* **propertiesParserRef** *属性来引用* **Jasypt**。

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:cm="http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.0.0"
        xsi:schemaLocation="
        http://www.osgi.org/xmlns/blueprint/v1.0.0
http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd">

  <!-- define the jasypt properties parser with the given password to be used -->
  <bean id="jasypt" class="org.apache.camel.component.jasypt.JasyptPropertiesParser">
    <property name="password" value="secret"/>
  </bean>

  <camelContext xmlns="http://camel.apache.org/schema/blueprint">
    <!-- define the camel properties placeholder, and let it leverage jasypt -->
    <propertyPlaceholder id="properties"
                         location="classpath:org/apache/camel/component/jasypt/myproperties.properties"
                         propertiesParserRef="jasypt"/>
      <route>
        <from uri="direct:start"/>
        <to uri="{{cool.result}}"/>
      </route>
```

```
    </camelContext>

</blueprint>
```

168.7. 另请参阅

- 安全性
- Properties
- ActiveMQ- ActiveMQ 中的加密密码 具有类似于 camel-jasypt 组件的功能

第 169 章 JAXB DATAFORMAT

从 Camel 版本 1.0 开始提供

JAXB 是一种 Data Format，它使用 JAXB2 XML marshalling 标准，包含在 Java 6 中，将 XML 有效负载 unmarshal 到 Java 对象，或将 Java 对象嵌套到 XML 有效负载中。

169.1. 选项

JAXB dataformat 支持 18 个选项，如下所列。

Name	默认值	Java 类型	描述
contextPath		字符串	您的 JAXB 类所在的软件包名称。
schema		字符串	对现有模式进行验证。您可以使用前缀 classpath:、file: 或 http: 指定资源应如何解析。您可以使用 ',' 字符分隔多个架构文件。
schemaSeverityLevel	0	整数	设置在针对 schema 验证时要使用的模式严重性级别。此级别决定了触发 JAXB 停止继续解析的最低严重性错误。默认值为 0（警告）意味着任何错误（警告、错误或严重错误）将触发 JAXB 停止。有三个级别有以下三个级别：0=warning, 1=error, 2=fatal 错误。
prettyPrint	false	布尔值	要启用用户的打印输出，请执行以下操作：默认为 false。
objectFactory	false	布尔值	是否允许使用 ObjectFactory 类在 marshalling 期间创建 POJO 类。这只适用于没有使用 JAXB 注解并提供 jaxb.index 描述符文件的 POJO 类。
ignoreJAXBElement	false	布尔值	是否要在非常特殊用例中忽略 JAXBElement 元素 - 只需要设置为 false。
mustBeJAXBElement	false	布尔值	marhsalling 必须是带有 JAXB 注释的 java 对象。如果不是，则失败。这个选项可以设置为 false 以宽松，例如当数据已采用 XML 格式时。
filterNonXmlChars	false	布尔值	要忽略非 xml characheters，并使用一个空空间替换它们。
编码		字符串	overrule 并使用特定的编码

Name	默认值	Java 类型	描述
片段	false	布尔值	打开 marshalling XML 片段树。默认情况下，JAXB 会查找给定类上的 XmlRootElement 注释，以针对整个 XML 树操作。这很有用，但并不总是 - 有时生成的代码没有 XmlRootElement 注解，有时您需要 unmarshall 仅是树的一部分。在这种情况下，您可以使用部分 unmarshalling。要启用此功能，您需要设置属性 partClass。Camel 会将此类传递给 JAXB 的 unmarshaler。
partClass		字符串	用于片段解析的类名称。请参阅片段选项的详情。
partNamespace		字符串	用于片段解析的 XML 命名空间。请参阅片段选项的详情。
namespacePrefixRef		字符串	使用 JAXB 或 SOAP 进行 marshalling 时，JAXB 实施将自动分配命名空间前缀，如 ns2、ns3、ns4 等。要控制此映射，Camel 允许您引用包含所需映射的映射。
xmlStreamWriterWrapper		字符串	使用自定义 xml 流写入器。
schemaLocation		字符串	定义模式的位置
noNamespaceSchemaLocation		字符串	定义无命名空间模式的位置
jaxbProviderProperties		字符串	指的是包含用于 JAXB marshaller 的自定义 JAXB 提供程序属性的 registry 中的自定义 java.util.Map。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

169.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 19 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.jaxb.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.jaxb.context-path	您的 JAXB 类所在的软件包名称。		字符串
camel.dataformat.jaxb.enabled	启用 jaxb dataformat	true	布尔值
camel.dataformat.jaxb.encoding	overrule 并使用特定的编码		字符串
camel.dataformat.jaxb.filter-non-xml-chars	要忽略非 xml characheters，并使用一个空空间替换它们。	false	布尔值
camel.dataformat.jaxb.fragment	打开 marshalling XML 片段树。默认情况下，JAXB 会查找给定类上的 XmlRootElement 注释，以针对整个 XML 树操作。这很有用，但并不总是 - 有时生成的代码没有 XmlRootElement 注解，有时您需要 unmarshall 仅是树的一部分。在这种情况下，您可以使用部分 unmarshalling。要启用此功能，您需要设置属性 partClass。Camel 会将此类传递给 JAXB 的 unmarshaler。	false	布尔值
camel.dataformat.jaxb.ignore-j-a-x-b-element	是否要在非常特殊用例中忽略 JAXBElement 元素 - 只需要设置为 false。	false	布尔值
camel.dataformat.jaxb.jaxb-provider-properties	指的是包含用于 JAXB marshaller 的自定义 JAXB 提供程序属性的 registry 中的自定义 java.util.Map。		字符串
camel.dataformat.jaxb.must-be-j-a-x-b-element	marhsalling 必须是带有 JAXB 注释的 java 对象。如果不是，则失败。这个选项可以设置为 false 以宽松，例如当数据已采用 XML 格式时。	false	布尔值
camel.dataformat.jaxb.namespace-prefix-ref	使用 JAXB 或 SOAP 进行 marshalling 时，JAXB 实施将自动分配命名空间前缀，如 ns2、ns3、ns4 等。要控制此映射，Camel 允许您引用包含所需映射的映射。		字符串
camel.dataformat.jaxb.no-namespace-schema-location	定义无命名空间模式的位置		字符串

Name	描述	默认值	类型
camel.dataformat.jaxb.object-factory	是否允许使用 ObjectFactory 类在 marshalling 期间创建 POJO 类。这只适用于没有使用 JAXB 注解并提供 jaxb.index 描述符文件的 POJO 类。	false	布尔值
camel.dataformat.jaxb.part-class	用于片段解析的类名称。请参阅片段选项的详情。		字符串
camel.dataformat.jaxb.part-namespace	用于片段解析的 XML 命名空间。请参阅片段选项的详情。		字符串
camel.dataformat.jaxb.pretty-print	要启用用户的打印输出，请执行以下操作：默认为 false。	false	布尔值
camel.dataformat.jaxb.schema	对现有模式进行验证。您可以使用前缀 classpath:、file: 或 http: 指定资源应如何解析。您可以使用 ',' 字符分隔多个架构文件。		字符串
camel.dataformat.jaxb.schema-location	定义模式的位置		字符串
camel.dataformat.jaxb.schema-severity-level	设置在针对 schema 验证时要使用的模式严重性级别。此级别决定了触发 JAXB 停止继续解析的最低严重性错误。默认值为 0（警告）意味着任何错误（警告、错误或严重错误）将触发 JAXB 停止。有三个级别有以下三个级别：0=warning, 1=error, 2=fatal 错误。	0	整数
camel.dataformat.jaxb.xml-stream-writer-wrapper	使用自定义 xml 流写入器。		字符串

ND

169.3. 使用 JAVA DSL

例如，以下命令使用名为 DataFormat 的 jaxb，它配置了多个 Java 软件包名称来初始化 JAXBContext。

```
DataFormat jaxb = new JaxbDataFormat("com.acme.model");

from("activemq:My.Queue").
  unmarshal(jaxb).
  to("mqseries:Another.Queue");
```

如果您希望使用与数据格式的命名引用，然后在 Registry 中定义，如通过您的 Spring XML 文件。

```
from("activemq:My.Queue").
  unmarshal("myJaxbDataType").
  to("mqseries:Another.Queue");
```

169.4. 使用 SPRING XML

以下示例演示了如何使用 JAXB 来使用 Spring 配置 jaxb 数据类型

本例演示了如何只配置数据类型一次，并在多个路由中重复使用它。

多个上下文路径

可以将此数据格式与多个上下文路径一起使用。您可以使用 : 作为分隔符指定上下文路径，如 com.mycompany:com.mycompany2。请注意，这由 JAXB 实施处理，如果您使用与 RI 不同的供应商，则可能会改变。

169.5. PARTIAL MARSHALLING/UNMARSHALLING

此功能是 Camel 2.2.0 的新内容。
JAXB 2 支持 marshalling 和 unmarshalling XML 树片段。默认情况下，JAXB 会查找给定类上的 @XmlRootElement 注释，以便对整个 XML 树进行操作。这很有用，但并不总是 - 有时生成的代码没有 @XmlRootElement 注释，有时您需要 unmarshall 仅是树的一部分。
在这种情况下，您可以使用部分 unmarshalling。要启用此功能，您需要设置属性 partClass。Camel 会将此类传递给 JAXB 的 unmarshaler。如果 JaxbConstants.JAXB_PART_CLASS 设置为其中一个标头，（即使在 DataFormat 上设置了 partClass 属性），则使用 DataFormat 的属性，并使用标头中设置的属性。

对于 marshalling，您必须使用目标命名空间的 QName 添加 partNamespace 属性。您可以在上面找到的 Spring DSL 示例。如果 JaxbConstants.JAXB_PART_NAMESPACE 设置为其中一个标头，（即使在 DataFormat 上设置了 partNamespace 属性），则 DataFormat 的属性已超过，并使用标头中设置的属性。在通过 JaxbConstants.JAXB_PART_NAMESPACE 设置 partNamespace 时，您需要指定其值 {[namespaceUri]}[localPart]

```
...
.setHeader(JaxbConstants.JAXB_PART_NAMESPACE, simple("
{http://www.camel.apache.org/jaxb/example/address/1}address"));
...
```

169.6. 片段

此功能是 Camel 2.8.0 中的新功能。
JaxbDataFormat 具有新的属性片段，它们可以在 JAXB Marshaller 上设置 Marshaller.JAXB_FRAGMENT 编码属性。如果您不希望 JAXB Marshaller 生成 XML 声明，您可以将此选项设置为 true。此属性的默认值为 false。

169.7. 忽略 NONXML CHARACTER

此功能是 Camel 2.2.0 的新内容。
JaxbDataFromat 支持忽略 NonXML Character，您只需要将 filterNonXmlChars 属性设置为 true，JaxbDataFormat 会将 NonXML 字符替换为 " "，当它是 marshaling 或 unmarshaling。您还可以通过设置 Exchange 属性 Exchange.FILTER_NON_XML_CHARS 来完成此操作。

	JDK 1.5	JDK 1.6+
使用中的过滤	stax API 和实现	否
不使用过滤	仅限 stax API	否

此功能已使用 Woodstox 3.2.9 和 Sun JDK 1.6 StAX 实现进行了测试。

Camel 2.12.1
JaxbDataFormat 的新信息现在允许您自定义用于向 XML 分割流的 XMLStreamWriter。使用这个配置，您可以添加自己的流写器来完全删除、转义或替换非xml 字符。

```
JaxbDataFormat customWriterFormat = new JaxbDataFormat("org.apache.camel.foo.bar");
customWriterFormat.setXmlStreamWriterWrapper(new TestXmlStreamWriter());
```

以下示例显示了使用 Spring DSL 并启用 Camel 的 NonXML 过滤：

```
<bean id="testXmlStreamWriterWrapper" class="org.apache.camel.jaxb.TestXmlStreamWriter"/>
<jaxb filterNonXmlChars="true"  contextPath="org.apache.camel.foo.bar"
xmlStreamWriterWrapper="#testXmlStreamWriterWrapper" />
```

169.8. 使用 OBJECTFACTORY

如果使用 XJC 从架构创建 java 类，您将获得 JAXB 上下文的 ObjectFactory。由于 ObjectFactory 使用 JAXBElement 来保存 schema 和元素实例值的引用，jaxbDataformat 默认忽略 JAXBElement，您将获取元素实例值而不是 JAXBElement 对象形成 unmarshaled 消息正文。
如果要获取 JAXBElement 对象形成 unmarshaled 消息正文，您需要将 JaxbDataFormat 对象的 ignoreJAXBElement 属性设为 false。

169.9. 设置编码

您可以将 编码 选项设置为在 marshalling 时使用的。其 JAXB Marshaller 上的 Marshaller.JAXB_ENCODING 编码属性。
您可以在声明 JAXB 数据格式时设置要使用的编码。您还可以在 Exchange 属性 Exchange.CHARSET_NAME 中提供编码。此属性覆盖 JAXB 数据格式上设置的编码。

在此 Spring DSL 中，我们定义了 iso-8859-1 作为编码：

169.10. 控制命名空间前缀映射

从 Camel 2.11 开始提供

使用 JAXB 或 SOAP 进行 marshalling 时，JAXB 实施将自动分配命名空间前缀，如 ns2、ns3、ns4 等。要控制此映射，Camel 允许您引用包含所需映射的映射。

请注意，这需要在类路径上具有 JAXB-RI 2.1 或更好（从 SUN），因为映射功能取决于 JAXB 的实施，无论是其受支持。

例如，在 Spring XML 中，我们可以使用映射定义映射。在下面的映射文件中，我们将 SOAP 映射到使用 soap 作为前缀。虽然我们的自定义命名空间 "http://www.mycompany.com/foo/2" 不使用任何前缀。

```
<util:map id="myMap">
  <entry key="http://www.w3.org/2003/05/soap-envelope" value="soap"/>
  <!-- we dont want any prefix for our namespace -->
  <entry key="http://www.mycompany.com/foo/2" value=""/>
</util:map>
```

要在 JAXB 或 SOAP 中使用此功能，您可以使用 namespacePrefixRef 属性来引用此映射，如下所示。然后，Camel 将在带有 id "myMap" 的 Registry 中查找 java.util.Map，这是我们上面定义的。

```
<marshal>
  <soapjaxb version="1.2" contextPath="com.mycompany.foo" namespacePrefixRef="myMap"/>
</marshal>
```

169.11. 模式验证

从 Camel 2.11 开始提供

JAXB 数据格式支持通过从/到 XML 进行 marshalling 和 unmarshalling 的验证。您可以使用前缀 classpath:, file: 或 http: 指定资源应如何解析。您可以使用 ',' 字符分隔多个架构文件。

已知问题

Camel 2.11.0 和 2.11.1 并行验证多个 'Exchange' 存在一个已知问题。请参阅 CAMEL-6630。这可以通过 Camel 2.11.2/2.12.0 解决。

使用 Java DSL，您可以使用以下方法进行配置：

```
JaxbDataFormat jaxbDataFormat = new JaxbDataFormat();
jaxbDataFormat.setContextPath(Person.class.getPackage().getName());
jaxbDataFormat.setSchema("classpath:person.xsd,classpath:address.xsd");
```

您可以使用 XML DSL 执行相同的操作：

```
<marshal>
   <jaxb id="jaxb" schema="classpath:person.xsd,classpath:address.xsd"/>
</marshal>
```

Camel 将实时创建并池ling SchemaFactory 实例，因为 JDK 附带的 SchemaFactory 不安全。但是，如果您有一个 SchemaFactory 实现是安全的线程，您可以将 JAXB 数据格式配置为使用此格式：

```
JaxbDataFormat jaxbDataFormat = new JaxbDataFormat();
jaxbDataFormat.setSchemaFactory(thradSafeSchemaFactory);
```

169.12. 模式位置

从 Camel 2.14 开始提供

JAXB Data Format 支持在编译 XML 时指定 SchemaLocation。

使用 Java DSL，您可以使用以下方法进行配置：

```
JaxbDataFormat jaxbDataFormat = new JaxbDataFormat();
jaxbDataFormat.setContextPath(Person.class.getPackage().getName());
jaxbDataFormat.setSchemaLocation("schema/person.xsd");
```

您可以使用 XML DSL 执行相同的操作：

```
<marshal>
    <jaxb id="jaxb" schemaLocation="schema/person.xsd"/>
</marshal>
```

169.13. 已 XML 的 MARSHAL 数据

从 Camel 2.14.1 开始提供

JAXB marshaller 要求消息正文与 JAXB 兼容，例如其 JAXBElement，例如具有 JAXB 注释的 java 实例或扩展 JAXBElement。在某些情况下，消息正文已在 XML 中，例如从 String 类型。有一个新选项 mustBeJAXBElement，您可以设为 false，以放宽此检查，因此 JAXB marshaller 仅尝试 marshal JAXBElements (javax.xml.bind.JAXBIntrospector"isElement 返回 true)。在这些情况下，marshaller 回退到消息正文原样。

169.14. 依赖项

要在 camel 路由中使用 JAXB，您需要添加对实现此数据格式的 camel-jaxb 的依赖关系。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jaxb</artifactId>
  <version>x.x.x</version>
</dependency>
```

第 170 章 JCACHE 组件

从 Camel 版本 2.17 开始提供

jcache 组件允许您使用 JSR107/JCache 作为缓存实施来执行缓存操作。

170.1. URI 格式

```
jcache:cacheName[?options]
```

170.2. URI 选项

JCache 端点使用 URI 语法进行配置：

```
jcache:cacheName
```

使用以下路径和查询参数：

170.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
cacheName	**所需的** 缓存的名称		字符串

170.2.2. 查询参数(22 参数)：

Name	描述	默认值	类型
cacheConfiguration (common)	缓存的配置		配置
cacheConfigurationProperties (common)	javax.cache.spi.CachingProvider 的属性，以创建 CacheManager		Properties
cachingProvider (common)	javax.cache.spi.CachingProvider 的完全限定类名称		字符串
configurationUri (common)	CacheManager 的实现特定 URI		字符串

Name	描述	默认值	类型
managementEnabled (common)	是否启用管理收集	false	布尔值
readThrough (common)	如果应使用 read-through 缓存	false	布尔值
statisticsEnabled (common)	是否启用统计收集	false	布尔值
storeByValue (common)	如果缓存应使用 store-by-value 或 store-by-reference 语义	true	布尔值
writethrough (common)	如果应使用直写缓存	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
filteredEvents (consumer)	消费者应过滤的事件。如果使用 filteredEvents 选项，则忽略 eventFilters		list
oldValueRequired (consumer)	如果事件需要旧值	false	布尔值
同步 (consumer)	如果事件监听程序应该阻止线程导致事件	false	布尔值
eventFilters (consumer)	CacheEntryEventFilter。如果使用 eventFilters 选项，则 filterEvents 将会被忽略		list
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
action (producer)	默认使用缓存操作配置。如果消息标头中的操作，则标题中的操作具有优先权。		字符串

Name	描述	默认值	类型
cacheLoaderFactory (advanced)	CacheLoader 工厂		factory
cacheWriterFactory (advanced)	CacheWriter 工厂		factory
createCacheIfNotExists (advanced)	如果缓存存在或无法预先配置，则配置是否需要创建缓存。	true	布尔值
expiryPolicyFactory (advanced)	ExpiryPolicy 工厂		factory
lookupProviders (advanced)	配置 camel-cache 是否应该尝试在 OSGi 等运行时查找 jcache api 的实现。	false	布尔值

170.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
camel.component.jcache.cache-configuration	缓存的配置.选项是 javax.cache.configuration.Configuration 类型。		字符串
camel.component.jcache.cache-configuration-properties	javax.cache.spi.CachingProvider 的属性，以创建 CacheManager。选项是一个 java.util.Properties 类型。		字符串
camel.component.jcache.caching-provider	javax.cache.spi.CachingProvider 的完全限定类名称		字符串
camel.component.jcache.configuration-uri	CacheManager 的实现特定 URI		字符串
camel.component.jcache.enabled	启用 jcache 组件	true	布尔值

Name	描述	默认值	类型
camel.component.jcache.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

JCache 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
cachingProvider (common)	javax.cache.spi.CachingProvider 的完全限定类名称		字符串
cacheConfiguration (common)	缓存的配置		配置
cacheConfiguration Properties (common)	javax.cache.spi.CachingProvider 的属性，以创建 CacheManager		Properties
configurationUri (common)	CacheManager 的实现特定 URI		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 171 章 JCLOUDS COMPONENT

从 Camel 版本 2.9 开始提供

此组件允许与云供应商键值引擎(blobstores)和计算服务交互。组件使用 jclouds，即为 blobstores 和 compute 服务提供抽象的库。

ComputeService 简化在云中管理机器的任务。例如，您可以使用 ComputeService 启动 5 机器并在其上安装软件。
BlobStore 简化了处理键值提供程序（如 Amazon S3）的过程。例如，BlobStore 可以为您提供容器的简单映射视图。

camel jclouds 组件允许您同时使用两个抽象，因为它指定了 JcloudsBlobStoreEndpoint 和 JcloudsComputeEndpoint 的两种类型的端点。您可以在 blobstore 端点上同时具有生成者和消费者，但您只能在计算端点上具有制作者。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jclouds</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

171.1. 配置组件

camel jclouds 组件将使用多个 jclouds blobstores 和 compute 服务，只要它们在初始化过程中传递给组件。组件接受列表 blobstores 和 compute 服务。以下是如何配置它。

```
<bean id="jclouds" class="org.apache.camel.component.jclouds.JcloudsComponent">
  <property name="computeServices">
    <list>
      <ref bean="computeService"/>
    </list>
  </property>
  <property name="blobStores">
    <list>
      <ref bean="blobStore"/>
    </list>
  </property>
</bean>
```

```
    <!-- Creating a blobstore from spring / blueprint xml -->
    <bean id="blobStoreContextFactory" class="org.jclouds.blobstore.BlobStoreContextFactory"/>

    <bean id="blobStoreContext" factory-bean="blobStoreContextFactory" factory-
method="createContext">
        <constructor-arg name="provider" value="PROVIDER_NAME"/>
        <constructor-arg name="identity"  value="IDENTITY"/>
        <constructor-arg name="credential" value="CREDENTIAL"/>
    </bean>

    <bean id="blobStore" factory-bean="blobStoreContext" factory-method="getBlobStore"/>

    <!-- Creating a compute service from spring / blueprint xml -->
    <bean id="computeServiceContextFactory"
class="org.jclouds.compute.ComputeServiceContextFactory"/>

    <bean id="computeServiceContext" factory-bean="computeServiceContextFactory" factory-
method="createContext">
        <constructor-arg name="provider" value="PROVIDER_NAME"/>
        <constructor-arg name="identity"  value="IDENTITY"/>
        <constructor-arg name="credential" value="CREDENTIAL"/>
    </bean>

    <bean id="computeService" factory-bean="computeServiceContext" factory-
method="getComputeService"/>
```

您可以看到组件能够处理多个 blobstores 和 compute 服务。每个端点使用的实际实施是通过传递 URI 中的提供程序来指定。

171.2. JCLOUDS 选项

```
jclouds:blobstore:[provider id][?options]
jclouds:compute:[provider id][?options]
```

provider id 是提供目标服务的云供应商的名称(如 aws-s3 或 aws_ec2)。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

171.3. BLOBSTORE URI 选项

JClouds 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
blobStores (common)	要使用给定的 BlobStore，必须在使用 blobstore 时配置。		list
computeServices (common)	要使用给定的 ComputeService，必须在使用 compute 时配置。		list
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

JClouds 端点使用 URI 语法进行配置：

```
jclouds:command:providerId
```

使用以下路径和查询参数：

171.3.1. 路径参数(2 参数)：

Name	描述	默认值	类型
命令	**需要执行** 什么命令，如 blobstore 或 compute。		JcloudsCommand
providerId	**必需** 提供目标服务的云供应商名称（如 aws-s3 或 aws_ec2）。		字符串

171.3.2. 查询参数(15 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler

Name	描述	默认值	类型
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
blobName (blobstore)	blob 的名称。		字符串
容器 (blobstore)	blob 容器的名称。		字符串
目录 (blobstore)	要使用的可选目录名称		字符串
组 (compute)	分配给新创建的节点的组。值取决于实际的云供应商。		字符串
hardwareId (compute)	用于创建节点的硬件。值取决于实际的云供应商。		字符串
imageId (compute)	用于创建节点的 imageId。值取决于实际的云供应商。		字符串
locationId (compute)	用于创建节点的位置。值取决于实际的云供应商。		字符串
nodeId (compute)	将运行脚本或销毁的节点 ID。		字符串
nodeState (compute)	按节点状态过滤为仅选择运行的节点等。		字符串
操作 （计算）	指定将对 blobstore 执行的操作类型。		字符串
用户 （计算）	运行该脚本的目标节点上的用户。		字符串

您可以根据需要拥有任意数量的选项。

```
jclouds:blobstore:aws-s3?
operation=CamelJcloudsGet&container=mycontainer&blobName=someblob
```

对于制作者端点，您可以通过将适当的标头传递给消息来覆盖上述所有 URI 选项。

171.3.3. blobstore 的消息标头

标头	描述
CamelJcloudsOperation	要在 blob 上执行的操作。有效选项为 * PUT * GET
CamelJcloudsContainer	blob 容器的名称。
CamelJcloudsBlobName	blob 的名称。

171.4. BLOBSTORE USAGE SAMPLES

171.4.1. 示例 1：到 blob

本例将演示如何使用 jclouds 组件将任何消息存储在 blob 中。

```
from("direct:start")
   .to("jclouds:blobstore:aws-s3" +
      "?operation=PUT" +
      "&container=mycontainer" +
      "&blobName=myblob");
```

在上例中，您可以使用消息上的标头覆盖任何 URI 参数。以下是上述示例如何使用 xml 定义路由：

```
<route>
   <from uri="direct:start"/>
   <to uri="jclouds:blobstore:aws-s3?operation=PUT&container=mycontainer&blobName=myblob"/>
</route>
```

171.4.2. 示例 2：从 blob 获取/读取

本例将演示如何使用 jclouds 组件读取 blob 的 contnet。

```
from("direct:start")
   .to("jclouds:blobstore:aws-s3" +
      "?operation=GET" +
```

```
        "&container=mycontainer" +
        "&blobName=myblob");
```

在上例中，您可以使用消息上的标头覆盖任何 URI 参数。以下是上述示例如何使用 xml 定义路由：

```
<route>
   <from uri="direct:start"/>
   <to uri="jclouds:blobstore:aws-s3?operation=PUT&container=mycontainer&blobName=myblob"/>
</route>
```

171.4.3. 示例 3：使用 blob

本例将使用指定容器下的所有 Blob。生成的交换将包含 blob 作为正文的载荷。

```
    from("jclouds:blobstore:aws-s3" +
        "?container=mycontainer")
        .to("direct:next");
```

您可以使用 xml 实现相同的目标，如下所示。

```
<route>
   <from uri="jclouds:blobstore:aws-s3?
operation=GET&container=mycontainer&blobName=myblob"/>
   <to uri="direct:next"/>
</route>
```

```
jclouds:compute:aws-ec2?
operation=CamelJcloudsCreateNode&imageId=AMI_XXXXX&locationId=eu-
west-1&group=mygroup
```

171.5. 计算使用示例

以下是在 java dsl 和 spring/blueprint xml 中演示 jclouds compute producer 使用的一些示例。

171.5.1. 示例 1：列出可用的镜像。

```
    from("jclouds:compute:aws-ec2" +
        "&operation=CamelJCloudsListImages")
        .to("direct:next");
```

这将创建一个消息，它将包含其正文内镜像列表。您还可以使用 xml 执行相同的操作。

```
<route>
  <from uri="jclouds:compute:aws-ec2?operation=CamelJCloudsListImages"/>
  <to uri="direct:next"/>
</route>
```

171.5.2. 示例 2：创建新节点。

```
  from("direct:start").
  to("jclouds:compute:aws-ec2" +
    "?operation=CamelJcloudsCreateNode" +
    "&imageId=AMI_XXXXX" +
    "&locationId=XXXXX" +
    "&group=myGroup");
```

这将在云供应商上创建一个新节点。本例中的信息将是一组元数据，其中包含新创建的节点的信息（如 ip、hostname 等）。以下是使用 spring xml 的相同：

```
<route>
  <from uri="direct:start"/>
  <to uri="jclouds:compute:aws-ec2?
operation=CamelJcloudsCreateNode&imageId=AMI_XXXXX&locationId=XXXXX&group=myGroup"/>
</route>
```

171.5.3. 示例 3：在运行的节点上运行 shell 脚本。

```
  from("direct:start").
  to("jclouds:compute:aws-ec2" +
    "?operation=CamelJcloudsRunScript" +
    "?nodeId=10" +
    "&user=ubuntu");
```

上面的示例将检索消息中的正文，该正文应包含要执行的 shell 脚本。检索脚本后，它将发送到节点，以便在指定的用户下执行(按 case case questions)。目标节点使用其 nodeId 指定。在创建节点时，可以检索 nodeId，它将是生成的元数据的一部分，或者执行 LIST_NODES 操作。

请注意，这将要求将传递给组件的计算服务使用相应的 jclouds ssh 功能模块(如 jsch 或 sshj)进行初始化。

以下是使用 spring xml 的相同：

```
<route>
  <from uri="direct:start"/>
  <to uri="jclouds:compute:aws-ec2?operation=CamelJcloudsListNodes&?
nodeId=10&user=ubuntu"/>
</route>
```

171.5.4. 另请参阅

如果您想在此处找到有关 jclouds 的更多信息，则属于值得关注的资源

Jclouds Blobstore wiki

Jclouds Compute wiki

第 172 章 JCR 组件

从 Camel 版本 1.3 开始提供

jcr 组件允许您将/读取节点添加到/读取节点到 JCR 兼容内容存储库（例如，Apache Jackrabbit）及其生成者，或使用消费者注册 EventListener。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jcr</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

172.1. URI 格式

```
jcr://user:password@repository/path/to/node
```

添加消费者

从 Camel 2.10 开始，您可以使用 consumer 作为 JCR 中的 EventListener 或生成者来根据标识符读取节点。

172.2. 使用方法

URI 的 repository 元素用于在 Camel 上下文 registry 中查找 JCR Repository 对象。

172.2.1. JCR 选项

JCR 组件没有选项。

JCR 端点使用 URI 语法进行配置：

```
jcr:host/base
```

使用以下路径和查询参数：

172.2.2. 路径参数(2 参数)：

Name	描述	默认值	类型
主机	**要从** 要使用的 Camel registry 中查询 javax.jcr.Repository 的必要名称。		字符串
base	在访问存储库时获取基本节点		字符串

172.2.3. 查询参数(14 参数)：

Name	描述	默认值	类型
deep (common)	当为Deep 为 true 时，其关联的父节点位于 absPath 或其子部分的事件会被接收。	false	布尔值
eventTypes (common)	eventTypes （一个或多个事件类型的组合编码为位掩码值，如 javax.jcr.observation.Event.NODE_ADDED, javax.jcr.observation.Event.NODE_REMOVED 等）。		int
nodeTypeNames (common)	当设置了逗号分隔的 nodeTypeName 列表字符串时，只有与其关联的父节点有一个节点类型（或其中一个节点类型的子类型）的事件才会接收此列表。		字符串
noLocal (common)	如果 noLocal 为 true，则忽略注册监听程序的会话生成的事件。否则，它们不会被忽略。	false	布尔值
password (common)	用于登录的密码		字符串
sessionLiveCheck Interval (common)	每个会话实时检查默认值 60000 ms 前等待的时间间隔（毫秒）。	60000	long
sessionLiveCheck IntervalOn Start (common)	第一次会话实时检查前等待的时间间隔（毫秒）。默认值为 3000 ms。	3000	long
username (common)	登录的用户名		字符串
UUID (common)	当设置了以逗号分隔的 uuid 列表字符串时，只会接收与其关联的父节点在逗号分隔的 uuid 列表中有一个标识符的事件。		字符串

Name	描述	默认值	类型
workspaceName (common)	用于访问的工作空间。如果未指定，则使用默认值		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

172.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.jcr.enabled	启用 jcr 组件	true	布尔值
camel.component.jcr.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

请注意，JCR Producer 使用消息属性，而不是 Camel 版本早于 2.12.3 的消息标头。详情请查看 https://issues.apache.org/jira/browse/CAMEL-7067。

172.4. EXAMPLE

以下片段在内容存储库的 /home/test 节点下创建名为 node 的节点。另外，还向节点添加了一个附加属性： my.contents.property，它将包含所发送消息的正文。

```
from("direct:a").setHeader(JcrConstants.JCR_NODE_NAME, constant("node"))
    .setHeader("my.contents.property", body())
    .to("jcr://user:pass@repository/home/test");
```

以下代码将在 Event.NODE_ADDED 和 Event.NODE_REMOVED 事件(event 类型 1 和 2)在路径 import-application/inbox 下注册 EventListener，并侦听所有子对象。

```
<route>
    <from uri="jcr://user:pass@repository/import-application/inbox?eventTypes=3&deep=true" />
    <to uri="direct:execute-import-application" />
</route>
```

172.5. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 173 章 JDBC 组件

从 Camel 版本 1.2 开始提供

jdbc 组件允许您通过 JDBC 访问数据库，其中 SQL 查询(SELECT)和操作(INSERT、UPDATE、et c)在消息正文中发送。此组件使用标准的 JDBC API，这与使用 spring-jdbc 的 SQL 组件 不同。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jdbc</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

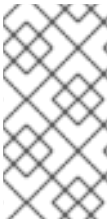
此组件只能用于定义制作者端点，这意味着您无法在 from （） 语句中使用 JDBC 组件。

Transactions

使用 camel-jdbc 选项时，您必须实现并配置事务管理器，并将 Camel 路由定义中的 resetAutoCommit 属性设置为 false ：

使用 camel-jdbc 的事务路由定义

```
from("direct:tx")
  .transacted()
  .to("jdbc:test_db?resetAutoCommit=false")
```



注意

与 camel-jdbc 组件一起使用时，不需要 transacted=true 属性。如果您需要额外的功能，请考虑使用 camel-sql 组件。

173.1. URI 格式

```
jdbc:dataSourceName[?options]
```

此组件仅支持生成者端点。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

173.2. 选项

JDBC 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
DataSource (producer)	使用 DataSource 实例，而不是按名称从 registry 查找数据源。		DataSource
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

JDBC 端点使用 URI 语法进行配置：

```
jdbc:dataSourceName
```

使用以下路径和查询参数：

173.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
dataSourceName	在 Registry 中查找的数据源的名称。**如果名称是 dataSource 或 default，则 Camel 将尝试从 registry 中查找默认 DataSource，这意味着如果只找到一个 DataSource 实例，则将使用此 DataSource。**		字符串

173.2.2. 查询参数(13 参数)：

Name	描述	默认值	类型
allowNamedParameters (producer)	是否允许在查询中使用命名参数。	true	布尔值

Name	描述	默认值	类型
outputClass (producer)	指定在 outputType=SelectOne 或 SelectList 时使用的完整软件包和类名称。		字符串
outputType (producer)	确定生成者应使用的输出。	SelectList	JdbcOutputType
parameters (producer)	java.sql.Statement 的可选参数。例如，要设置 maxRows、fetchSize 等。		Map
readSize (producer)	轮询查询可读取的默认最大行数。默认值为 0。		int
resetAutoCommit (producer)	Camel 会将 JDBC 连接上的 autoCommit 设置为 false，在执行声明后提交更改，并在结尾重置连接的 autoCommit 标志（如果 resetAutoCommit 为 true）。如果 JDBC 连接不支持重置 autoCommit 标志，您可以将 resetAutoCommit 标志设置为 false，并且 Camel 不会尝试重置 autoCommit 标志。与 XA 事务一起使用时，您可能需要将其设置为 false，以便事务管理器负责提交此 tx。	true	布尔值
transacted (producer)	事务是否使用。	false	布尔值
useGetBytesForBlob (producer)	将 BLOB 列读取为字节而不是字符串数据。对于某些数据库（如 Oracle），这可能是必需的，其中您必须将 BLOB 列作为字节读取。	false	布尔值
useHeadersAsParameters (producer)	将这个选项设置为 true 来使用带有命名参数的 prepareStatementStrategy。这允许使用命名占位符定义查询，并将标头与查询占位符的动态值一起使用。	false	布尔值
useJDBC4ColumnNameAndLabelSemantics (producer)	设置在检索列名称时是否使用 JDBC 4 还是 JDBC 3.0 或旧的语义。JDBC 4.0 使用 columnLabel 获取列名称，因为 JDBC 3.0 使用 columnName 或 columnLabel。不幸的是，JDBC 驱动程序的行为不同，如果您遇到这个组件的问题，则可以使用这个选项来排除 JDBC 驱动程序的问题。	true	布尔值
beanRowMapper (advanced)	在使用 outputClass 时，使用自定义 org.apache.camel.component.jdbc.BeanRowMapper。默认实现会降低行名，并跳过下划线和横线。例如，CUST_ID 映射为 custId。		BeanRowMapper
prepareStatementStrategy (advanced)	允许插件使用自定义 org.apache.camel.component.jdbc.JdbcPrepareStatementStrategy 来控制查询和准备语句的准备。		JdbcPrepareStatement Strategy

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

173.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.jdbc.data-source	使用 DataSource 实例，而不是按名称从 registry 查找数据源。选项是 javax.sql.DataSource 类型。		字符串
camel.component.jdbc.enabled	启用 jdbc 组件	true	布尔值
camel.component.jdbc.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

173.4. 结果

默认情况下，结果在 OUT 正文中返回，作为 ArrayList<HashMap<String, Object>>。List 对象包含行列表，Map 对象包含每行，String 键作为列名称。您可以使用选项 outputType 来控制结果。

注：此组件获取 ResultSetMetaData，以便能够将列名称返回为 Map 中的键。

173.4.1. 消息标头

标头	描述
CamelJdbcRowCount	如果查询是 **SELECT**，则查询此 IN 标头中返回行数。

标头	描述
CamelJdbcUpdateCount	如果查询是 **UPDATE**，则查询此 IN 标头中返回更新计数。
CamelGeneratedKeysRows	Camel 2.10： 包含生成的 kets 的 Rows。
CamelGeneratedKeysRowCount	Camel 2.10 ： 包含生成的密钥的标头中的行数。
CamelJdbcColumnNames	Camel 2.11.1 ： 来自 ResultSet 的列名称作为 **java.util.Set** 类型。
CamelJdbcParametes	Camel 2.12： 一个 **java.util.Map**，它在启用了 **useHeadersAsParameters** 时使用标头。

173.5. 生成的密钥

从 Camel 2.10 开始提供

如果您使用 SQL INSERT 插入数据，则 RDBMS 可能会支持自动生成的密钥。您可以指示 JDBC 生成者在标头中返回生成的密钥。
为此，请设置标头 CamelRetrieveGeneratedKeys=true。然后，生成的密钥将作为标头提供，其中包含上表中列出的键。

您可以在此 单元测试 中看到更多详细信息。

使用生成的密钥无法与命名参数一起使用。

173.6. 使用命名参数

从 Camel 2.12 开始提供

在下面的给定路由中，我们想要从 projects 表中获取所有项目。注意 SQL 查询具有 2 个命名参数 :?lic 和 :?min。
然后，Camel 将从消息标头中查找这些参数。请注意，在上面的示例中，我们为命名参数设置两个带有恒定值
的标头：

```
from("direct:projects")
  .setHeader("lic", constant("ASF"))
  .setHeader("min", constant(123))
  .setBody("select * from projects where license = :?lic and id > :?min order by id")
  .to("jdbc:myDataSource?useHeadersAsParameters=true")
```

您也可以将标头值存储在 java.util.Map 中，并将 map 存储在带有键为 CamelJdbcParameters 的标头中。

173.7. SAMPLES

在以下示例中，我们从 customer 表中获取行。

首先，在 Camel registry 中注册数据源为 testdb ：

然后，我们配置路由到 JDBC 组件的路由，以便执行 SQL。请注意，我们如何引用上一步中绑定的 testdb 数据源：

或者您可以在 Spring 中创建数据源，如下所示：

我们创建一个端点，将 SQL 查询添加到 IN 消息的正文，然后发送交换。查询的结果在 OUT 正文中返回：

如果您需要处理一行，而不是使用一个行，而不是整个 ResultSet，您需要使用 Splitter EIP，例如：

```
from("direct:hello")
// here we split the data from the testdb into new messages one by one
```

```
// so the mock endpoint will receive a message per row in the table
// the StreamList option allows to stream the result of the query without creating a List of rows
// and notice we also enable streaming mode on the splitter
.to("jdbc:testdb?outputType=StreamList")
  .split(body()).streaming()
  .to("mock:result");
```

173.8. 示例 - 每分钟轮询数据库

如果我们希望使用 JDBC 组件轮询数据库，我们需要将其与 Timer 或 Quartz 等轮询调度程序合并。在以下示例中，我们每 60 秒从数据库检索数据：

```
from("timer://foo?period=60000")
  .setBody(constant("select * from customer"))
  .to("jdbc:testdb")
  .to("activemq:queue:customers");
```

173.9. 示例 - 在数据源之间移动数据

常见的用例是查询数据，处理数据并将其移动到另一个数据源(ETL 操作)。在以下示例中，我们每小时从源表中检索新的客户记录，过滤/转换它们并将其移到目标表中：

```
from("timer://MoveNewCustomersEveryHour?period=3600000")
    .setBody(constant("select * from customer where create_time > (sysdate-1/24)"))
    .to("jdbc:testdb")
    .split(body())
        .process(new MyCustomerProcessor()) //filter/transform results as needed
        .setBody(simple("insert into processed_customer
values('${body[ID]}','${body[NAME]}')"))
        .to("jdbc:testdb");
```

第 174 章 JETTY 9 组件

从 Camel 版本 1.2 开始提供

警告

producer 已被弃用 - 不使用。我们建议将 jetty 用作消费者（例如 from jetty）

jetty 组件为消耗和生成 HTTP 请求提供基于 HTTP 的端点。也就是说，Jetty 组件的行为是一个简单的 Web 服务器。
jetty 也可以用作 http 客户端，这意味着您也可以将其用作制作者。

Stream

assert 调用会出现在本示例中，因为代码是单元 test.Jetty 基于流的一部分，这意味着它收到的输入作为流提交给 Camel。这意味着，在 后只能读取流的内容。
如果您发现消息正文似乎为空的情况，或者您需要多次访问 Exchange.HTTP_RESPONSE_CODE 数据（例如：执行多播或重新发送错误处理），您应该使用流缓存或将消息正文转换为 字符串，该字符串可以安全地重新读取多次。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jetty</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

174.1. URI 格式

```
jetty:http://hostname[:port][/resourceUri][?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

174.2. 选项

Jetty 9 组件支持 33 选项，如下所列。

Name	描述	默认值	类型
sslKeyPassword (security)	密钥密码，用于访问密钥存储中的证书密钥条目（这与为 keystore 命令的 -keypass 选项提供的密码相同）。		字符串
sslpassword (security)	访问密钥存储文件所需的 ssl 密码（这与为 keystore 命令的 -storepass 选项提供的密码相同）。		字符串
keystore (security)	指定 Java 密钥存储文件的位置，该文件在密钥条目中包含 Jetty 服务器自己的 X.509 证书。		字符串
errorHandler (advanced)	此选项用于设置 Jetty 服务器使用的 ErrorHandler。		ErrorHandler
sslSocketConnectors (security)	包含每个端口号特定的 SSL 连接器的映射。		Map
socketConnectors (security)	包含每个端口号特定 HTTP 连接器的映射。使用与 sslSocketConnectors 相同的原则。		Map
httpClientMinThreads (producer)	要在 HttpClient 线程池中为最少线程数设置值：请注意，必须同时配置 min 和 max 大小。		整数
httpClientMaxThreads (producer)	要在 HttpClient 线程池中为最大线程数设置值：请注意，必须同时配置 min 和 max 大小。		整数
minThreads (consumer)	要为服务器线程池中最少的线程数设置值。请注意，必须同时配置 min 和 max 大小。		整数
maxThreads (consumer)	要为服务器线程池中最大线程数设置值。请注意，必须同时配置 min 和 max 大小。		整数
threadPool (consumer)	为服务器使用自定义线程池。这个选项只在特殊情况下才应使用。		ThreadPool
enableJmx (common)	如果这个选项为 true，则会为这个端点启用 Jetty JMX 支持。	false	布尔值
jettyHttpBinding (advanced)	要使用自定义 org.apache.camel.component.jetty.JettyHttpBinding，它用于自定义为生成者写入响应的方式。		JettyHttpBinding

Name	描述	默认值	类型
httpBinding (advanced)	不使用 - 改为使用 JettyHttpBinding。		HttpBinding
httpConfiguration (advanced)	jetty 组件不使用 HttpConfiguration。		HttpConfiguration
mbContainer (advanced)	要使用现有的已配置的 org.eclipse.jetty.jmx.MBeanContainer，如果启用了 JMX，则 Jetty 用于注册 mbeans。		MBeanContainer
sslSocketConnector Properties (security)	包含常规 SSL 连接器属性的映射。		Map
socketConnector Properties (security)	包含通用 HTTP 连接器属性的映射。使用与 sslSocketConnectorProperties 相同的原则。		Map
continuationTimeout (consumer)	在将 Jetty 用作消费者(server)时，允许在 millis 中设置超时。默认情况下，Jetty 使用 30000。您可以使用 = 0 值永不过期。如果发生超时，则请求将过期，Jetty 会将 http 错误 return 返回给客户端。这个选项仅在将 Jetty 与 Asynchronous Routing Engine 搭配使用时使用。	30000	Long
useContinuation (consumer)	是否对 Jetty 服务器使用 Jetty continuations。	true	布尔值
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数	false	布尔值
responseBufferSize (common)	允许在 Jetty 连接器上配置响应缓冲区大小的自定义值。		整数
requestBufferSize (common)	允许在 Jetty 连接器上配置请求缓冲区大小的自定义值。		整数
requestHeaderSize (common)	允许在 Jetty 连接器上配置请求标头大小的自定义值。		整数
responseHeaderSize (common)	允许在 Jetty 连接器上配置响应标头大小的自定义值。		整数

Name	描述	默认值	类型
proxyHost (proxy)	使用 http 代理配置主机名。		字符串
proxyPort (proxy)	使用 http 代理配置端口号。		整数
useXForwardedFor Header (common)	使用 HttpServletRequest.getRemoteAddr 中的 X-Forwarded-For 标头。	false	布尔值
sendServerVersion (consumer)	如果选项为 true，则 jetty 服务器会将 date 标头发送到发送请求的客户端。请注意，请确保没有任何其他 camel-jetty 端点共享相同的端口，否则此选项可能无法按预期工作。	true	布尔值
允许 JavaSerialized Object (advanced)	当请求使用 context-type=application/x-java-serialized-object 时，是否允许 java serialization。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Jetty 9 端点使用 URI 语法进行配置：

```
jetty:httpUri
```

使用以下路径和查询参数：

174.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
httpUri	**必需** 调用的 HTTP 端点的 url。		URI

174.2.2. 查询参数(54 参数)：

Name	描述	默认值	类型
chunked (common)	如果此选项为 false，则 Servlet 将禁用 HTTP 流，并在响应上设置 content-length 标头	true	布尔值
disableStreamCache (common)	确定 Servlet 中的原始输入流是否被缓存(Camel 将会在 memory/overflow 中读取流到文件，流缓存)缓存。默认情况下，Camel 将缓存 Servlet 输入流，以支持多次读取，以确保其 Camel 可以从流检索所有数据。但是，当您需要访问原始流时，您可以将此选项设置为 true，例如将其直接流传输到文件或其他持久性存储。DefaultHttpBinding 将请求输入流复制到流缓存中，如果此选项为 false，则将它放入消息正文，以支持多次读取流。如果您使用 Servlet 到 bridge/proxy 端点，则请考虑启用此选项来提高性能，如果您不需要多次读取消息有效负载。http/http4 producer 默认缓存响应正文流。如果将此选项设置为 true，则生成者不会缓存响应正文流，而是使用响应流作为消息正文。	false	布尔值
enableMultipartFilter (common)	是否启用 Jetty org.eclipse.jetty.servlets.MultiPartFilter。在桥接端点时，您应该将此值设置为 false，以确保多部分请求被代理/桥接。	false	布尔值
headerFilterStrategy (common)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
transferException (common)	如果在消费者端启用并交换失败处理，如果原因 Exception 在响应中作为 application/x-java-serialized-object 内容类型发送序列化，则结果为 application/x-java-serialized-object 内容类型。在生产者侧，异常将反序列化并丢弃为原样，而不是 HttpOperationFailedException。原因异常需要按顺序处理。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
httpBinding (common)	使用自定义 HttpBinding 来控制 Camel 消息和 HttpClient 之间的映射。		HttpBinding
async (consumer)	将消费者配置为在 async 模式下工作	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值

Name	描述	默认值	类型
continuationTimeout (consumer)	在将 Jetty 用作消费者(server)时，允许在 millis 中设置超时。默认情况下，Jetty 使用 30000。您可以使用 = 0 值永不过期。如果发生超时，则请求将过期，Jetty 会将 http 错误 return 返回给客户端。这个选项仅在将 Jetty 与 Asynchronous Routing Engine 搭配使用时使用。	30000	Long
enableCors (consumer)	如果选项为 true，则 Jetty 服务器将设置 CrossOriginFilter，它支持 out of box。	false	布尔值
enableJmx (consumer)	如果这个选项为 true，则会为这个端点启用 Jetty JMX 支持。如需了解更多详细信息，请参阅 Jetty JMX 支持。	false	布尔值
httpMethodRestrict (consumer)	仅在 HttpMethod 匹配时才允许使用，如 GET/POST/PUT 等。可以使用逗号分隔多个方法。		字符串
matchOnUriPrefix (consumer)	如果找不到完全匹配，消费者是否应该尝试通过匹配 URI 前缀来查找目标消费者。	false	布尔值
responseBufferSize (consumer)	在 javax.servlet.ServletResponse 上使用自定义缓冲区大小。		整数
sendDateHeader (consumer)	如果选项为 true，则 jetty 服务器会将 date 标头发送到发送请求的客户端。请注意，请确保没有任何其他 camel-jetty 端点共享相同的端口，否则此选项可能无法按预期工作。	false	布尔值
sendServerVersion (consumer)	如果选项为 true，则 jetty 会将带有 jetty 版本信息的 server 标头发送到发送请求的客户端。请注意，请确保没有任何其他 camel-jetty 端点共享相同的端口，否则此选项可能无法按预期工作。	true	布尔值
sessionSupport (consumer)	指定是否在 Jetty 的服务器端启用会话管理器。	false	布尔值
useContinuation (consumer)	是否对 Jetty 服务器使用 Jetty continuations。		布尔值
eagerCheckContentAvailable (consumer)	如果 content-length 标头为 0，还是 eager 检查 HTTP 请求是否有内容。如果 HTTP 客户端没有发送流数据，则可以打开此项。	false	布尔值

Name	描述	默认值	类型
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
filterInitParameters (consumer)	配置过滤器 init 参数。在启动 jetty 服务器前，这些参数将应用到过滤器列表。		Map
filtersRef (consumer)	允许使用放入列表中的自定义过滤器，并可以在 Registry 中找到。可以使用逗号分隔多个值。		字符串
handlers (consumer)	指定要在 Registry 中查询的以逗号分隔的处理程序实例集合。这些处理程序添加到 Jetty servlet 上下文（例如，添加安全性）。重要： 您不能使用相同的端口号在不同的 Jetty 端点中使用不同的处理程序。处理程序与端口号关联。如果您需要不同的处理程序，则使用不同的端口号。		字符串
httpBindingRef (consumer)	**弃用** Option 以禁用在来自远程服务器失败响应时抛出 HttpOperationFailedException。这样，您可以获取所有响应，而不考虑 HTTP 状态代码。		字符串
multipartFilter (consumer)	允许使用自定义多部分过滤器。注：将 multipartFilterRef 强制 enableMultipartFilter 的值为 true。		Filter
multipartFilterRef (consumer)	**弃用了** Allows，使用自定义多部分过滤器。注：将 multipartFilterRef 强制 enableMultipartFilter 的值为 true。		字符串
optionsEnabled (consumer)	指定是否为这个 Servlet 消费者启用 HTTP OPTIONS。默认情况下关闭 OPTIONS。	false	布尔值
traceEnabled (consumer)	指定是否为这个 Servlet 使用者启用 HTTP TRACE。默认情况下关闭 TRACE。	false	布尔值
bridgeEndpoint (producer)	如果选项为 true，则 HttpProducer 将忽略 Exchange.HTTP_URI 标头，并使用端点的 URI 进行请求。您也可以将 选项 throwExceptionOnFailure 设置为 false，让 HttpProducer 发送所有错误响应。	false	布尔值
connectionClose (producer)	指定是否应将 Connection Close 标头添加到 HTTP Request 中。默认情况下，connectionClose 为 false。	false	布尔值

Name	描述	默认值	类型
cookieHandler (producer)	配置 Cookie 处理程序，以维护 HTTP 会话		CookieHandler
copyHeaders (producer)	如果此选项为 true，则 IN Exchange 标头将根据复制策略复制到 OUT 交换标头。把它设置为 false，仅允许包含来自 HTTP 响应的标头（而不是传播 IN 标头）。	true	布尔值
httpClientMaxThreads (producer)	要在 HttpClient 线程池中为最大线程数设置值：此设置覆盖在组件级别配置的任何设置。请注意，必须同时配置 min 和 max 大小。如果没有将其设置为 Jettys 线程池中使用的最大 254 线程。	254	整数
httpClientMinThreads (producer)	要在 HttpClient 线程池中为最少线程数设置值：此设置覆盖在组件级别配置的任何设置。请注意，必须同时配置 min 和 max 大小。如果没有将其设置为 Jettys 线程池中使用的 min 8 线程。	8	整数
httpMethod (producer)	配置要使用的 HTTP 方法。如果设置，HttpMethod 标头无法覆盖此选项。		HttpMethods
ignoreResponseBody (producer)	如果此选项为 true，http producer 不会读取响应正文，并缓存输入流	false	布尔值
preserveHostHeader (producer)	如果选项为 true，HttpProducer 会将 Host 标头设置为当前交换主机标头中包含的值，这适用于您希望下游服务器收到的 Host 标头来反映上游客户端调用的 URL 的 URL，这将允许使用 Host 标头为代理服务生成准确的 URL 的应用	false	布尔值
throwExceptionOnFailure (producer)	如果来自远程服务器的失败响应，用于禁用抛出 HttpOperationFailedException。这样，您可以获取所有响应，而不考虑 HTTP 状态代码。	true	布尔值
httpclient (producer)	设置共享 HttpClient，以用于此端点创建的所有制作者。默认情况下，每个制作者都将使用一个新的 http 客户端，而不是共享。重要：确保处理共享客户端的生命周期（如在不再使用客户端时）。Camel 将在客户端上调用 start 方法，以确保在此端点创建制作者时启动它。只有在特殊情况下才应使用此选项。		HttpClient
httpClientParameters (producer)	配置 Jetty 的 HttpClient.例如，设置 httpClient.idleTimeout=30000 将空闲超时设置为 30 秒。和 httpClient.timeout=30000 将请求超时设置为 30 秒，如果您有长时间运行的请求/响应调用，则希望更早地超时。		Map
jettyBinding (producer)	要使用自定义 JettyHttpBinding，它用于自定义应当如何为生成者写入响应。		JettyHttpBinding

Name	描述	默认值	类型
jettyBindingRef (producer)	**弃用了** 自定义 JettyHttpBinding，它用于自定义应当如何为生成者写入响应。		字符串
okStatusCodeRange (producer)	被视为成功响应的状态代码。值包含。可以定义多个范围，用逗号分开，例如 200-204,209,301-304。每个范围必须是单个数字，或从 到，其中包含短划线。	200-299	字符串
urlRewrite (producer)	**弃用** 引用自定义 org.apache.camel.component.http.UrlRewrite，它允许您在 bridge/proxy 端点时重写 url。更多信息，请访问 http://camel.apache.org/urlrewrite.html		UrlRewrite
mapHttpMessageBody (advanced)	如果此选项为 true，则交换的 IN Exchange Body 将映射到 HTTP 正文。把它设置为 false 可以避免 HTTP 映射。	true	布尔值
mapHttpMessageFormUrlEncodedBody (advanced)	如果此选项为 true，则交换的 IN Exchange Form Encoded body 将映射到 HTTP。把它设置为 false 可以避免 HTTP Form Encoded body 映射。	true	布尔值
mapHttpMessageHeaders (advanced)	如果此选项为 true，则交换的 IN Exchange Headers 将映射到 HTTP 标头。把它设置为 false 将避免 HTTP 标头映射。	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
proxyAuthScheme (proxy)	要使用的代理身份验证方案		字符串
proxyHost (proxy)	要使用的代理主机名		字符串
proxyPort (proxy)	要使用的代理端口		int
authHost (security)	与 NTML 搭配使用的身份验证主机		字符串
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters

174.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 34 个选项，如下所列。

Name	描述	默认值	类型
camel.component.jetty.allow-java-serialized-object	当请求使用 context-type=application/x-java-serialized-object 时，是否允许 java serialization。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
camel.component.jetty.continuation-timeout	在将 Jetty 用作消费者(server)时，允许在 millis 中设置超时。默认情况下，Jetty 使用 30000。您可以使用 = 0 值永不过期。如果发生超时，则请求将过期，Jetty 会将 http 错误 return 返回给客户端。这个选项仅在将 Jetty 与 Asynchronous Routing Engine 搭配使用时使用。	30000	Long
camel.component.jetty.enable-jmx	如果这个选项为 true，则会为这个端点启用 Jetty JMX 支持。	false	布尔值
camel.component.jetty.enabled	启用 jetty 组件	true	布尔值
camel.component.jetty.error-handler	此选项用于设置 Jetty 服务器使用的 ErrorHandler。选项是一个 org.eclipse.jetty.server.handler.ErrorHandler 类型。		字符串
camel.component.jetty.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		字符串
camel.component.jetty.http-binding	不使用 - 改为使用 JettyHttpBinding。选项是 org.apache.camel.http.common.HttpBinding 类型。		字符串
camel.component.jetty.http-client-max-threads	要在 HttpClient 线程池中为最大线程数设置值：请注意，必须同时配置 min 和 max 大小。		整数
camel.component.jetty.http-client-min-threads	要在 HttpClient 线程池中为最少线程数设置值：请注意，必须同时配置 min 和 max 大小。		整数
camel.component.jetty.http-configuration	jetty 组件不使用 HttpConfiguration。选项是 org.apache.camel.http.common.HttpConfiguration 类型。		字符串
camel.component.jetty.jetty-http-binding	要使用自定义 org.apache.camel.component.jetty.JettyHttpBinding，它用于自定义为生成者写入响应的方式。选项是 org.apache.camel.component.jetty.JettyHttpBinding 类型。		字符串

Name	描述	默认值	类型
camel.component.jetty.keystore	指定 Java 密钥存储文件的位置，该文件在密钥条目中包含 Jetty 服务器自己的 X.509 证书。		字符串
camel.component.jetty.max-threads	要为服务器线程池中最大线程数设置值。请注意，必须同时配置 min 和 max 大小。		整数
camel.component.jetty.mb-container	要使用现有的已配置的 org.eclipse.jetty.jmx.MBeanContainer，如果启用了 JMX，则 Jetty 用于注册 mbeans。选项是一个 org.eclipse.jetty.jmx.MBeanContainer 类型。		字符串
camel.component.jetty.min-threads	要为服务器线程池中最少的线程数设置值。请注意，必须同时配置 min 和 max 大小。		整数
camel.component.jetty.proxy-host	使用 http 代理配置主机名。		字符串
camel.component.jetty.proxy-port	使用 http 代理配置端口号。		整数
camel.component.jetty.request-buffer-size	允许在 Jetty 连接器上配置请求缓冲区大小的自定义值。		整数
camel.component.jetty.request-header-size	允许在 Jetty 连接器上配置请求标头大小的自定义值。		整数
camel.component.jetty.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.jetty.response-buffer-size	允许在 Jetty 连接器上配置响应缓冲区大小的自定义值。		整数
camel.component.jetty.response-header-size	允许在 Jetty 连接器上配置响应标头大小的自定义值。		整数
camel.component.jetty.send-server-version	如果选项为 true，则 jetty 服务器会将 date 标头发送到发送请求的客户端。请注意，请确保没有任何其他 camel-jetty 端点共享相同的端口，否则此选项可能无法按预期工作。	true	布尔值

Name	描述	默认值	类型
camel.component.jetty.socket-connector-properties	包含通用 HTTP 连接器属性的映射。使用与 sslSocketConnectorProperties 相同的原则。选项是一个 java.util.Map<java.lang.String,java.lang.Object> 类型。		字符串
camel.component.jetty.socket-connectors	包含每个端口号特定 HTTP 连接器的映射。使用与 sslSocketConnectors 相同的原则。选项是一个 java.util.Map<java.lang.Integer,org.eclipse.jetty.server.Connector> 类型。		字符串
camel.component.jetty.ssl-context-parameters	使用 SSLContextParameters 配置安全性。选项是 org.apache.camel.util.jsse.SSLContextParameters 类型。		字符串
camel.component.jetty.ssl-key-password	密钥密码，用于访问密钥存储中的证书密钥条目（这与为 keystore 命令的 -keypass 选项提供的密码相同）。		字符串
camel.component.jetty.ssl-password	访问密钥存储文件所需的 ssl 密码（这与为 keystore 命令的 -storepass 选项提供的密码相同）。		字符串
camel.component.jetty.ssl-socket-connector-properties	包含常规 SSL 连接器属性的映射。选项是一个 java.util.Map<java.lang.String,java.lang.Object> 类型。		字符串
camel.component.jetty.ssl-socket-connectors	包含每个端口号特定的 SSL 连接器的映射。选项是一个 java.util.Map<java.lang.Integer,org.eclipse.jetty.server.Connector> 类型。		字符串
camel.component.jetty.thread-pool	为服务器使用自定义线程池。这个选项只在特殊情况下才应使用。选项是 org.eclipse.jetty.util.thread.ThreadPool 类型。		字符串
camel.component.jetty.use-continuation	是否对 Jetty 服务器使用 Jetty continuations。	true	布尔值
camel.component.jetty.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数	false	布尔值

Name	描述	默认值	类型
camel.component.jetty.use-x-forwarded-for-header	使用 HttpServletRequest.getRemoteAddr 中的 X-Forwarded-For 标头。	false	布尔值

174.4. 消息标头

Camel 使用与 HTTP 组件相同的消息标头。从 Camel 2.2 中，它还使用 (Exchange.HTTP_CHUNKED、CamelHttpChunked)标头来打开或关闭 camel-jetty 消费者上的 chuched 编码。

Camel 还会填充 所有 request.parameter 和 request.headers。例如，给定一个带有 URL 的客户端请求 http://myserver/myserver?orderid=123，交换将包含名为 orderid 的标头，其值为 123。

从 Camel 2.2.0 开始，您可以从消息标头获取 request.parameter，不仅从 Get Method 获取，也可以从其他 HTTP 方法获取。

174.5. 使用方法

Jetty 组件支持使用者和制作者端点。生成其他 HTTP 端点的另一个选项是使用 HTTP 组件

174.6. 生成者示例

警告

producer 已被弃用 - 不使用。我们建议将 jetty 用作消费者（例如 from jetty）

以下是如何将 HTTP 请求发送到现有 HTTP 端点的基本示例。

Java DSL

```
from("direct:start")
  .to("jetty://http://www.google.com");
```

或在 Spring XML 中

```
<route>
    <from uri="direct:start"/>
    <to uri="jetty://http://www.google.com"/>
<route>
```

174.7. 消费者示例

在这个示例中，我们定义在 http://localhost:8080/myapp/myservice 中公开 HTTP 服务的路由：

localhost 的使用

当您在 URL 中指定 localhost 时，Camel 仅在本地 TCP/IP 网络接口上公开端点，因此无法从其操作的计算机外部访问。

如果您需要在特定网络接口上公开 Jetty 端点，此接口的数字 IP 地址应用作主机。如果您需要在所有网络接口上公开 Jetty 端点，则应使用 0.0.0.0 地址。

要侦听整个 URI 前缀，请参阅 如何让 Jetty 匹配通配符。

如果您实际希望通过 HTTP 公开路由，且已经有一个 Servlet，您应该引用 Servlet 传输。

我们的业务逻辑在 MyBookService 类中实施，该类访问 HTTP 请求内容，然后返回响应。
注： assert 调用会出现在本示例中，因为代码是单元测试的一部分。

以下示例显示了基于内容的路由，它将包含 URI 参数的所有请求、一到端点、mock: one 以及所有其他路由到 mock:other。

因此，如果客户端发送 HTTP 请求 http://serverUri?one=hello，Jetty 组件将复制 HTTP 请求参数，一到 交换的 in.header。然后，我们可以使用 简单 语言将包含此标头的交换路由到特定端点，所有

其他端点。如果我们使用比 Simple 更强大的语言（如 OGNL），我们也可以测试参数值，并根据标头值进行路由。

174.8. 会话支持

会话支持选项 sessionSupport 可用于启用 HttpSession 对象，并在处理交换时访问会话对象。例如，以下路由启用会话：

```
<route>
  <from uri="jetty:http://0.0.0.0/myapp/myservice/?sessionSupport=true"/>
  <processRef ref="myCode"/>
<route>
```

myCode 处理器可以通过 Spring bean 元素实例化：

```
<bean id="myCode"class="com.mycompany.MyCodeProcessor"/>
```

其中处理器实施可以访问 HttpSession，如下所示：

```
public void process(Exchange exchange) throws Exception {
  HttpSession session = exchange.getIn(HttpMessage.class).getRequest().getSession();
  ...
}
```

174.9. SSL 支持(HTTPS)

使用 JSSE 配置实用程序

自 Camel 2.8 起，Jetty 组件通过 Camel JSSE 配置实用程序支持 SSL/TLS 配置。这个实用程序可大大减少您需要写入的组件特定代码量，并在端点和组件级别进行配置。以下示例演示了如何将 实用程序与 Jetty 组件一起使用。

组件的编程配置

```
KeyStoreParameters ksp = new KeyStoreParameters();
ksp.setResource("/users/home/server/keystore.jks");
ksp.setPassword("keystorePassword");

KeyManagersParameters kmp = new KeyManagersParameters();
kmp.setKeyStore(ksp);
```

```
kmp.setKeyPassword("keyPassword");

SSLContextParameters scp = new SSLContextParameters();
scp.setKeyManagers(kmp);

JettyComponent jettyComponent = getContext().getComponent("jetty",
JettyComponent.class);
jettyComponent.setSslContextParameters(scp);
```

基于 Spring DSL 的端点配置

```
...
  <camel:sslContextParameters
     id="sslContextParameters">
   <camel:keyManagers
      keyPassword="keyPassword">
     <camel:keyStore
        resource="/users/home/server/keystore.jks"
        password="keystorePassword"/>
    </camel:keyManagers>
  </camel:sslContextParameters>...
...
  <to uri="jetty:https://127.0.0.1/mail/?sslContextParameters=#sslContextParameters"/>
...
```

直接配置 Jetty

jetty 开箱即用提供 SSL 支持。要启用 Jetty 在 SSL 模式下运行，只需使用 https:// 前缀格式化 URI，例如：

```
<from uri="jetty:https://0.0.0.0/myapp/myservice/"/>
```

jetty 还需要知道要从中加载您的密钥存储的位置以及要使用什么密码来加载正确的 SSL 证书。设置以下 JVM 系统属性：

Camel 2.2

- jetty.ssl.keystore 指定 Java 密钥存储文件的位置，该文件在 密钥条目 中包含 Jetty 服务器自己的 X.509 证书。密钥条目存储 X.509 证书（有效，公钥）及其关联的私钥。

- jetty.ssl.password 存储密码，这是访问密钥存储文件所必需的（这与为 keystore 命令的 -storepass 选项提供的密码相同）。

- *jetty.ssl.keypassword 是密钥密码，用于访问密钥存储中的证书密钥条目（这与为 keystore 命令的 -keypass 选项提供的密码相同）。*

从 Camel 2.3 开始

- *org.eclipse.jetty.ssl.keystore 指定 Java 密钥存储文件的位置，该文件在 键条目 中包含 Jetty 服务器自己的 X.509 证书。密钥条目存储 X.509 证书（有效，公钥）及其关联的私钥。*

- *org.eclipse.jetty.ssl.password 访问存储密码，这是访问密钥存储文件所必需的（这与为 keystore 命令的 -storepass 选项提供的密码相同）。*

- *org.eclipse.jetty.ssl.keypassword 是密钥密码，用于访问密钥存储中的证书密钥条目（这与为 keystore 命令的 -keypass 选项提供的密码相同）。*

有关如何在 Jetty 端点上配置 SSL 的详情，请参考 Jetty 站点中的以下文档：
http://docs.codehaus.org/display/JETTY/How+to+configure+SSL

有些 SSL 属性不直接由 Camel 公开，但 Camel 会公开底层 SslSocketConnector，这将允许您为需要客户端证书或 wantClientAuth 用于 mutual 身份验证设置属性，如 needClientAuth 用于 mutual 身份验证，但可以有一个证书。各种 Camel 版本之间存在一些差别：

最多 Camel 2.2

```
<bean id="jetty" class="org.apache.camel.component.jetty.JettyHttpComponent">
  <property name="sslSocketConnectors">
    <map>
      <entry key="8043">
        <bean class="org.mortbay.jetty.security.SslSocketConnector">
          <property name="password"value="..."/>
          <property name="keyPassword"value="..."/>
          <property name="keystore"value="..."/>
          <property name="needClientAuth"value="..."/>
          <property name="truststore"value="..."/>
        </bean>
      </entry>
    </map>
  </property>
</bean>
```

Camel 2.3, 2.4

```
<bean id="jetty" class="org.apache.camel.component.jetty.JettyHttpComponent">
   <property name="sslSocketConnectors">
      <map>
         <entry key="8043">
            <bean class="org.eclipse.jetty.server.ssl.SslSocketConnector">
               <property name="password"value="..."/>
               <property name="keyPassword"value="..."/>
               <property name="keystore"value="..."/>
               <property name="needClientAuth"value="..."/>
               <property name="truststore"value="..."/>
            </bean>
         </entry>
      </map>
   </property>
</bean>
```

**from Camel 2.5 我们切换为使用 SslSelectChannelConnector **

```
<bean id="jetty" class="org.apache.camel.component.jetty.JettyHttpComponent">
   <property name="sslSocketConnectors">
      <map>
         <entry key="8043">
            <bean class="org.eclipse.jetty.server.ssl.SslSelectChannelConnector">
               <property name="password"value="..."/>
               <property name="keyPassword"value="..."/>
               <property name="keystore"value="..."/>
               <property name="needClientAuth"value="..."/>
               <property name="truststore"value="..."/>
            </bean>
         </entry>
      </map>
   </property>
</bean>
```

您在上述映射中用作键的值是将 Jetty 配置为侦听的端口。

174.9.1. 在 IBM Java 中使用 TLS 安全性配置 camel-jetty9

camel-jetty9 组件中的默认 TLS 安全设置与 IBM Java 虚拟机不兼容。IBM Java 中的所有密码都以前缀 SSL fluentd 开头，即使 TLS 协议的密码都以 SSL fluentd 开始。camel-jetty9 仅支持 RFC Cipher Suite 名称，且所有 SSL solutions 密码 都不受保护，且被排除。jetty 排除所有 SSL swig 密码，因此没有可用于 TLS 1.2 和连接的密码。因为无法更改 Jetty 的 ssl 上下文的行为，因此只有临时解决方案来覆盖 Jetty9 组件上的默认 TLS 安全配置。要达到此目的，请在 Application.java 文件的 "sslContextParameters （）" 方法的末尾添加以下代码：

```
FilterParameters fp = new FilterParameters();
    fp.getInclude().add(".*");

    // Exclude weak / insecure ciphers
    fp.getExclude().add("^.*_(MD5|SHA|SHA1)$");
    // Exclude ciphers that don't support forward secrecy
    fp.getExclude().add("^TLS_RSA_.*$");
    // The following exclusions are present to cleanup known bad cipher
    // suites that may be accidentally included via include patterns.
    // The default enabled cipher list in Java will not include these
    // (but they are available in the supported list).
    /* SSL_ ciphers are not excluded
    fp.getExclude().add("^SSL_.*$"); */
    fp.getExclude().add("^.NULL.$");
    fp.getExclude().add("^.anon.$");

    p.setCipherSuitesFilter(fp);
```

此代码通过删除排除所有 SSL solutions 密码来覆盖 Jetty 中定义的排除密码。

174.9.2. 配置常规 SSL 属性

从 Camel 2.5 开始提供

现在，您可以配置适用于所有 SSL 套接字连接器的通用属性，而不是每个端口号特定的 SSL 套接字连接器。

```
<bean id="jetty" class="org.apache.camel.component.jetty.JettyHttpComponent">
    <property name="sslSocketConnectorProperties">
        <map>
            <entry key="password"value="..."/>
            <entry key="keyPassword"value="..."/>
            <entry key="keystore"value="..."/>
            <entry key="needClientAuth"value="..."/>
            <entry key="truststore"value="..."/>
        </map>
    </property>
</bean>
```

174.9.3. 如何获取对 X509Certificate 的引用

jetty 在 HttpServletRequest 中存储对证书的引用，如下所示：

```
HttpServletRequest req = exchange.getIn().getBody(HttpServletRequest.class);
X509Certificate cert = (X509Certificate)
req.getAttribute("javax.servlet.request.X509Certificate")
```

174.9.4. 配置常规 HTTP 属性

从 Camel 2.5 开始提供

现在，您可以配置适用于所有 HTTP 套接字连接器的通用属性，而不是每个端口号特定的 HTTP 套接字连接器（这没有明确配置为条目）。

```
<bean id="jetty" class="org.apache.camel.component.jetty.JettyHttpComponent">
  <property name="socketConnectorProperties">
    <map>
      <entry key="acceptors" value="4"/>
      <entry key="maxIdleTime" value="300000"/>
    </map>
  </property>
</bean>
```

174.9.5. 使用 HttpServletRequest.getRemoteAddr （）获取 X-Forwarded-For 标头

如果 HTTP 请求由 Apache 服务器处理，并通过 mod_proxy 转发到 jetty，原始客户端 IP 地址位于 X-Forwarded-For 标头中，并且 HttpServletRequest.getRemoteAddr （）将返回 Apache 代理的地址。

jetty 有一个 forward 属性，它取 X-Forwarded-For 的值，并将它放在 HttpServletRequest remoteAddr 属性中。此属性不能直接通过端点配置获得，但可以使用 socketConnectors 属性轻松添加：

```
<bean id="jetty" class="org.apache.camel.component.jetty.JettyHttpComponent">
  <property name="socketConnectors">
    <map>
      <entry key="8080">
        <bean class="org.eclipse.jetty.server.nio.SelectChannelConnector">
          <property name="forwarded" value="true"/>
        </bean>
      </entry>
    </map>
  </property>
</bean>
```

当现有 Apache 服务器处理域的 TLS 连接并将其代理到内部应用服务器时，这特别有用。

174.10. 返回 HTTP 状态代码的默认行为

HTTP 状态代码的默认行为由 org.apache.camel.component.http.DefaultHttpBinding 类定义，该类

处理如何编写响应，同时设置 HTTP 状态代码。

如果交换成功处理，则返回 200 HTTP 状态代码。
如果交换失败但异常失败，则返回 500 HTTP 状态代码，并在正文中返回堆栈追踪。如果要指定要返回的 HTTP 状态代码，请在 OUT 消息的 Exchange.HTTP_RESPONSE_CODE 标头中设置代码。

174.11. 自定义 HTTPBINDING

默认情况下，Camel 使用 org.apache.camel.component.http.DefaultHttpBinding 来处理如何编写响应。如果您愿意，您可以通过实施自己的 HttpBinding 类或扩展 DefaultHttpBinding 并覆盖适当的方法来自定义此行为。

以下示例演示了如何自定义 DefaultHttpBinding 以更改异常的返回方式：

然后，我们可以创建绑定实例，并在 Spring registry 中注册，如下所示：

```
<bean id="mybinding"class="com.mycompany.MyHttpBinding"/>
```

然后我们可以在定义路由时引用此绑定：

```
<route>
  <from uri="jetty:http://0.0.0.0:8080/myapp/myservice?httpBindingRef=mybinding"/>
  <to uri="bean:doSomething"/>
</route>
```

174.12. JETTY 处理程序和安全配置

您可以在端点上配置 Jetty 处理程序列表，这对于启用高级 Jetty 安全功能非常有用。这些处理程序在 Spring XML 中配置，如下所示：

```
<-- Jetty Security handling -->
<bean id="userRealm" class="org.mortbay.jetty.plus.jaas.JAASUserRealm">
    <property name="name" value="tracker-users"/>
    <property name="loginModuleName" value="ldaploginmodule"/>
</bean>

<bean id="constraint" class="org.mortbay.jetty.security.Constraint">
    <property name="name" value="BASIC"/>
    <property name="roles" value="tracker-users"/>
    <property name="authenticate" value="true"/>
</bean>
```

```
<bean id="constraintMapping" class="org.mortbay.jetty.security.ConstraintMapping">
    <property name="constraint" ref="constraint"/>
    <property name="pathSpec" value="/*"/>
</bean>

<bean id="securityHandler" class="org.mortbay.jetty.security.SecurityHandler">
    <property name="userRealm" ref="userRealm"/>
    <property name="constraintMappings" ref="constraintMapping"/>
</bean>
```

从 Camel 2.3 开始，您可以配置 Jetty 处理程序列表，如下所示：

```
<-- Jetty Security handling -->
<bean id="constraint" class="org.eclipse.jetty.http.security.Constraint">
    <property name="name" value="BASIC"/>
    <property name="roles" value="tracker-users"/>
    <property name="authenticate" value="true"/>
</bean>

<bean id="constraintMapping" class="org.eclipse.jetty.security.ConstraintMapping">
    <property name="constraint" ref="constraint"/>
    <property name="pathSpec" value="/*"/>
</bean>

<bean id="securityHandler" class="org.eclipse.jetty.security.ConstraintSecurityHandler">
    <property name="authenticator">
        <bean class="org.eclipse.jetty.security.authentication.BasicAuthenticator"/>
    </property>
    <property name="constraintMappings">
        <list>
            <ref bean="constraintMapping"/>
        </list>
    </property>
</bean>
```

然后您可以将端点定义为：

```
from("jetty:http://0.0.0.0:9080/myservice?handlers=securityHandler")
```

如果需要更多处理程序，请将 handlers 选项等于以逗号分隔的 bean ID 列表。

174.13. 如何返回自定义 HTTP 500 回复信息

当出现错误时，您可能希望返回自定义回复消息，而不是默认回复消息 Camel Jetty 回复。
您可以使用自定义 HttpBinding 来控制消息映射，但通常更易于使用 Camel 的 Exception Clause 来构

造自定义回复消息。例如，这里返回 Dude some error with HTTP 错误代码 500 错误 ：

174.14. 多部分表单支持

从 Camel 2.3.0 开始，camel-jetty 支持开箱即用的多部分表单。已提交的表单数据映射到消息标头中。camel-jetty 为每个上传的文件创建一个附件。文件名映射到附加的名称。内容类型被设置为附加文件名的内容类型。您可以在此处找到示例。

注意：getName （）函数（如版本 2.5 及更高版本所示）。在早期版本中，您收到附件的临时文件名

174.15. JETTY JMX 支持

从 Camel 2.3.0 中，camel-jetty 支持在组件中启用 Jetty 的 JMX 功能，以及端点配置具有优先权的端点。请注意，在 Camel 上下文中必须启用 JMX，才能在此组件中启用 JMX 支持，因为组件提供 Jetty 带有对使用 Camel 上下文注册的 MBeanServer 的引用。由于 camel-jetty 组件缓存并重复使用给定协议/主机/端口对的 Jetty 资源，因此仅在创建第一个端点期间评估此配置选项以使用协议/主机/端口对。例如，如果从以下 XML 片段创建的两个路由，JMX 支持将继续对在 "https://0.0.0.0" 上侦听的所有端点启用。

```
<from uri="jetty:https://0.0.0.0/myapp/myservice1/?enableJmx=true"/>
```

```
<from uri="jetty:https://0.0.0.0/myapp/myservice2/?enableJmx=false"/>
```

camel-jetty 组件还提供用于配置 Jetty MBeanContainer。jetty 动态创建 MBean 名称。如果您在 Camel 上下文之外运行另一个 Jetty 实例，并在实例之间共享同一 MBeanServer，您可以为这两个实例提供同一 MBeanContainer 的引用，以避免注册 Jetty MBeans 时名称冲突。

第 175 章 JGROUPS 组件

从 Camel 版本 2.13 开始提供

JGroups 是可靠多播通信的工具包。jgroups： 组件提供 Camel 基础架构和 JGroups 集群之间的消息交换。

Maven 用户需要将以下依赖项添加到此组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache-extras.camel-extra</groupId>
  <artifactId>camel-jgroups</artifactId>
  <!-- use the same version as your Camel core version -->
  <version>x.y.z</version>
</dependency>
```

从 Camel 2.13.0 开始， JGroups 组件已从 Apache Camel umbrella 下的 Camel Extra 移动。如果您使用 Camel 2.13.0 或更高版本， 请改用以下 POM 条目。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jgroups</artifactId>
  <!-- use the same version as your Camel core version -->
  <version>x.y.z</version>
</dependency>
```

175.1. URI 格式

```
jgroups:clusterName[?options]
```

其中 clusterName 代表组件应连接到的 JGroups 集群的名称。

175.2. 选项

JGroups 组件支持 4 个选项， 如下所列。

Name	描述	默认值	类型
channel (common)	要使用的频道		JChannel

Name	描述	默认值	类型
channelProperties (common)	指定端点使用的 JChannel 的配置属性。		字符串
enableViewMessages (consumer)	如果设置为 true，消费者端点也会接收 org.jgroups.View 消息（不仅 org.jgroups.Message 实例）。默认情况下，端点仅消耗常规消息。	false	布尔值
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

JGroups 端点使用 URI 语法进行配置：

```
jgroups:clusterName
```

使用以下路径和查询参数：

175.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
clusterName	**必需** 组件应连接的 JGroups 集群的名称。		字符串

175.2.2. 查询参数(6 参数)：

Name	描述	默认值	类型
channelProperties (common)	指定端点使用的 JChannel 的配置属性。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
enableViewMessages (consumer)	如果设置为 true，消费者端点也会接收 org.jgroups.View 消息（不仅 org.jgroups.Message 实例）。默认情况下，端点仅消耗常规消息。	false	布尔值

Name	描述	默认值	类型
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

175.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
camel.component .jgroups.channel	要使用的频道。选项是一个 org.jgroups.JChannel 类型。		字符串
camel.component .jgroups.channel-properties	指定端点使用的 JChannel 的配置属性。		字符串
camel.component .jgroups.enable-view-messages	如果设置为 true，消费者端点也会接收 org.jgroups.View 消息（不仅 org.jgroups.Message 实例）。默认情况下，端点仅消耗常规消息。	false	布尔值
camel.component .jgroups.enabled	启用 jgroups 组件	true	布尔值
camel.component .jgroups.lock.cluster.service.enabled	设置 jgroups 锁定群集服务是否应启用，则默认为 false。	false	布尔值
camel.component .jgroups.lock.cluster.service.id	集群服务 ID		字符串
camel.component .jgroups.lock.cluster.service.jgroups-cluster-name	JGroups 集群名称		字符串

Name	描述	默认值	类型
camel.component.jgroups.lock.cluster.service.jgroups-config	JGrups 配置文件名称		字符串
camel.component.jgroups.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

175.4. HEADERS

标头	常数	自版本	描述
JGROUPS_ORIGINAL_MESSAGE	**JGroupsEndpoint.HEADER_JGROUPS_ORIGINAL_MESSAGE**	2.13.0	从中提取已提取消息正文的原始 **org.jgroups.Message** 实例。
JGROUPS_SRC	`JGroupsEndpoint.`HEADER_JGROUPS_SRC	2.10.0	**consumer** : 由所消耗的消息的 **org.jgroups.Message**.getSrc （）方法提取的 **org.jgroups.Address** 实例。**producer** ：要发送的消息的自定义源 **org.jgroups.Address**。
JGROUPS_DEST	`JGroupsEndpoint.`HEADER_JGROUPS_DEST	2.10.0	**consumer**: 由所消耗的消息的 **org.jgroups.Message**.getDest （）方法提取的 **org.jgroups.Address** 实例。**producer** ：要发送的消息的自定义目的地 **org.jgroups.Address**。

标头	常数	自版本	描述
JGROUPS_CHANNEL_ADDRESS	`JGroupsEndpoint.`HEADER_JGROUPS_CHANNEL_ADDRESS	2.13.0	与端点关联的频道的地址(**org.jgroups.Address**)。

175.5. 使用方法

在路由的使用者端使用 jgroups 组件将捕获与端点关联的 JChannel 收到的消息，并将它们转发到 Camel 路由。JGroups 使用者 异步 处理传入的消息。

```
// Capture messages from cluster named
// 'clusterName' and send them to Camel route.
from("jgroups:clusterName").to("seda:queue");
```

在路由的制作者上使用 jgroups 组件，会将 Camel 交换的正文转发到端点管理的 JChannel 实例。

```
// Send message to the cluster named 'clusterName'
from("direct:start").to("jgroups:clusterName");
```

175.6. 预定义的过滤器

从 Camel 版本 2.13.0 开始，JGroups 组件附带了名为 JGroupsFilters 的预定义过滤器工厂类。

如果您要仅使用发送到集群的协调器的更改通知（并忽略发送到"从节点"节点），请使用 JGroupsFilters.dropNonCoordinatorViews （） 过滤器。当您希望单个 Camel 节点成为集群中的 master 时，此过滤器特别有用，因为当给定节点成为集群的协调者时，传递此过滤器的消息会通知您。以下片段演示了如何仅收集 master 节点收到的消息。

```
import static
org.apache.camel.component.jgroups.JGroupsFilters.dropNonCoordinatorViews;
...
from("jgroups:clusterName?enableViewMessages=true").
  filter(dropNonCoordinatorViews()).
  to("seda:masterNodeEventsQueue");
```

175.7. 预定义的表达式

从 Camel 版本 2.13.0 开始，JGroups 组件附带了名为 JGroupsExpressions 的预定义表达式工厂类。

如果您要创建仅影响 Camel 上下文时路由的延迟器，请使用 JGroupsExpressions.delayIfContextNotStarted (long delay) factory 方法。此工厂方法创建的表达式只有在 Camel 上下文处于与启动 状态不同时，才会返回给定延迟值。如果您要使用 JGroups 组件在集群中保留单例(master)路由时，此表达式特别有用。如果 Camel 上下文尚未启动，则 控制 Bus start 命令不会初始化单例路由。因此，您需要延迟 master 路由的启动，以确保它在 Camel 上下文启动后初始化。由于这样的场景只能在集群初始化期间发生，因此我们不希望延迟从节点启动成为新的主节点 - 这为什么我们需要有条件延迟表达式。

以下片段演示了如何将条件延迟与 JGroups 组件一起使用，以延迟集群中主节点的初始启动。

```
import static java.util.concurrent.TimeUnit.SECONDS;
import static
org.apache.camel.component.jgroups.JGroupsExpressions.delayIfContextNotStarted;
import static
org.apache.camel.component.jgroups.JGroupsFilters.dropNonCoordinatorViews;
...
from("jgroups:clusterName?enableViewMessages=true").
  filter(dropNonCoordinatorViews()).
  threads().delay(delayIfContextNotStarted(SECONDS.toMillis(5))). // run in separated and
delayed thread. Delay only if the context hasn't been started already.
  to("controlbus:route?routeId=masterRoute&action=start&async=true");

from("timer://master?
repeatCount=1").routeId("masterRoute").autoStartup(false).to(masterMockUri);
```

175.8. 例子

175.8.1. 将消息（接收）消息发送到 JGroups 集群（来自）

若要向 JGroups 集群使用制作者端点发送消息，如以下代码片段中所示：

```
from("direct:start").to("jgroups:myCluster");
...
producerTemplate.sendBody("direct:start", "msg")
```

要接收来自上述代码片段（在相同或其它物理计算机上）的消息，请侦听来自给定集群的消息，就像以下代码片段中演示一样。

```
mockEndpoint.setExpectedMessageCount(1);
mockEndpoint.message(0).body().isEqualTo("msg");
...
from("jgroups:myCluster").to("mock:messagesFromTheCluster");
...
mockEndpoint.assertIsSatisfied();
```

175.8.2. 接收集群视图更改通知

以下片段演示了如何创建侦听集群成员资格更改通知的消费者端点。默认情况下，端点仅消耗常规消息。

```
mockEndpoint.setExpectedMessageCount(1);
mockEndpoint.message(0).body().isInstanceOf(org.jgroups.View.class);
...
from("jgroups:clusterName?enableViewMessages=true").to(mockEndpoint);
...
mockEndpoint.assertIsSatisfied();
```

175.8.3. 在集群中保留单例路由

以下片段演示了如何将单例消费者路由保留在 Camel 上下文的集群中。一旦主节点结束，其中一个从节点将被选为一个新的 master 并启动。在这个特定示例中，我们希望保持单例 jetty 实例侦听地址的 http://localhost:8080/orders` 上的请求。

```
JGroupsLockClusterService service = new JGroupsLockClusterService();
service.setId("uniqueNodeId");
...
context.addService(service);

from("master:mycluster:jetty:http://localhost:8080/orders").to("jms:orders");
```

第 176 章 JIBX DATAFORMAT

从 Camel 版本 2.6 开始提供

jibx 是一个 Data Format，它使用 JiBX 库来整理和从 XML 中对 marshal 和 unmarshal Java 对象进行 marshal 和 unmarshal Java 对象。

```
// lets turn Object messages into XML then send to MQSeries
from("activemq:My.Queue").
  marshal().jibx().
  to("mqseries:Another.Queue");
```

请注意，marshaling 进程可以在运行时识别消息类型。但是，虽然来自 XML 的 unmarshaling 消息需要明确指定目标类。

```
// lets turn XML into PurchaseOrder message
from("mqseries:Another.Queue").
  unmarshal().jibx(PurchaseOrder.class).
  to("activemq:My.Queue");
```

176.1. 选项

JiBX dataformat 支持 3 个选项，如下所列。

Name	默认值	Java 类型	描述
unmarshallClass		字符串	从 XML 到 Java 时使用的类名称。
bindingName		字符串	使用自定义绑定工厂
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

176.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.jibx.binding-name	使用自定义绑定工厂		字符串
camel.dataformat.jibx.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.jibx.enabled	启用 jibx dataformat	true	布尔值
camel.dataformat.jibx.unmarshall-class	从 XML 到 Java 时使用的类名称。		字符串

ND

176.3. JIBX SPRING DSL

Camel Spring DSL 还支持 jibx 数据格式。

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">

  <!-- Define data formats -->
  <dataFormats>
    <jibx id="jibx" unmarshallClass="org.apache.camel.dataformat.jibx.PurchaseOrder"/>
  </dataFormats>

  <!-- Marshal message to XML -->
  <route>
    <from uri="direct:marshal"/>
    <marshal ref="jibx"/>
    <to uri="mock:result"/>
  </route>

  <!-- Unmarshal message from XML -->
  <route>
    <from uri="direct:unmarshal"/>
    <unmarshal ref="jibx"/>
    <to uri="mock:result"/>
  </route>

</camelContext>
```

176.4. 依赖项

要在 camel 路由中使用 JiBX，您需要添加对实现此数据格式的 camel-jibx 的依赖。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jibx</artifactId>
  <version>2.6.0</version>
</dependency>
```

第 177 章 JING COMPONENT

从 Camel 版本 1.1 开始提供

Jing 组件使用 Jing 库 来利用其中之一执行消息正文的 XML 验证

- RelaxNG XML Syntax
- RelaxNG Compact Syntax

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jing</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

请注意，MSV 组件也可以支持 RelaxNG XML 语法。

177.1. URI 格式 CAMEL 2.16

```
jing:someLocalOrRemoteResource
```

在 Camel 2.16 中，组件使用 jing 作为名称，您可以使用选项紧凑语法打开 RNG 或 RNC 模式。

177.2. 选项

Jing 组件没有选项。

Jing 端点使用 URI 语法进行配置：

```
jing:resourceUri
```

使用以下路径和查询参数：

177.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceUri	需要在类路径上本地资源或指向远程资源或资源的完整 URL （包含要进行验证的 schema）的 URL。		字符串

177.2.2. 查询参数(2 参数)：

Name	描述	默认值	类型
compactSyntax (producer)	是否使用 RelaxNG 紧凑语法进行验证。默认情况下，使用 RelaxNG XML 语法(rng)和 true 是使用 RelaxNG Compact Syntax (rnc)的 false	false	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

177.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.jing.enabled	启用 jing 组件	true	布尔值
camel.component.jing.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

177.4. EXAMPLE

以下示例演示了如何 配置来自端点 direct:start 的路由，该路由会被分为两个端点之一，根据 XML 是否与给定的 RelaxNG Compact Syntax 模式（在 classpath 上提供） mock:valid 或 mock:invalid。

177.5. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 178 章 JIRA COMPONENT （已弃用）

从 Camel 版本 2.15 开始提供

JIRA 组件通过封装 Atlassian 的 REST Java 客户端 JIRA 来与 JIRA API 交互。它目前为新问题和新注释提供轮询。它还能够创建新问题。

此端点依赖于简单的轮询，而不是 Webhook。原因包括：

- 可靠性/状态的问题
- 我们正在轮询的有效负载类型通常不大（以及分页在 API 中提供）
- 需要支持在有些地方无法公开运行的应用程序，其中 Webhook 会失败

请注意，JIRA API 相当大。因此，此组件可轻松扩展，以提供额外的交互。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jira</artifactId>
  <version>${camel-version}</version>
</dependency>
```

178.1. URI 格式

```
jira://endpoint[?options]
```

178.2. JIRA 选项

JIRA 组件没有选项。

JIRA 端点使用 URI 语法进行配置：

```
jira:type
```

使用以下路径和查询参数：

178.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
type	**执行** 所需的操作，如创建新问题或新注释		JIRAType

178.2.2. 查询参数(9 参数)：

Name	描述	默认值	类型
password (common)	用于登录的密码		字符串
serverUrl (common)	JIRA 服务器 **所需的** URL		字符串
username (common)	登录的用户名		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
delay (consumer)	使用消费者查询 JIRA 时的延迟（以秒为单位）。	6000	int
jql (consumer)	JQL 是 JIRA 中的查询语言，允许您检索您想要的数据。例如，jql=project=MyProject where MyProject 是 JIRA 中的产品密钥。		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler

Name	描述	默认值	类型
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

178.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.jira.enabled	启用 jira 组件	true	布尔值
camel.component.jira.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

178.4. JQL：

两个消费者端点都使用 JQL URI 选项。在理论上，"项目密钥"等项目可以是 URI 选项本身。然而，由于要求使用 JQL，用户变得更灵活、更强大。

用户至少需要以下内容：

```
jira://[endpoint]?[required options]&jql=project=[project key]
```

需要注意的是，newIssue 消费者将自动将"ORDER BY 键 desc"附加到您的 JQL 中。这是为了优化启动处理，不必对项目中的每个问题进行索引。

另一个备注是，newComment 使用者必须索引项目中每个问题和注释。因此，对于大型项目，优化 JQL 表达式尽可能 至关重要。例如，JIRA Toolkit Plugin 在查询中包含 "Number of comments" custom field the '"Number of comments" > 0'。另外，尝试根据状态(status=Open)最小化，增加轮询延迟等。Example:

> ***jira://[endpoint]?[required options]&jql=RAW(project=[project key] AND status in (Open, \"Coding In Progress\") AND \"Number of comments\">0)"***

第 179 章 JMS 组件

179.1. JMS 组件

提示

使用 ActiveMQ

如果您使用的是 Apache ActiveMQ，您应当首选 ActiveMQ 组件，因为它已针对 ActiveMQ 进行优化。此页面上的所有选项和示例也对 ActiveMQ 组件有效。

注意

转换和缓存

如果您使用 JMS 的事务，请参阅以下的 Transactions 和 Cache Levels，因为它可能会影响性能。

注意

通过 JMS 请求/恢复

确保在此页面上 通过 JMS 进一步阅读第 Request-reply 部分，以获得有关请求/回复的重要备注，因为 Camel 提供了很多选项来提高性能和集群环境。

此组件允许将消息发送到（或从中使用） JMS Queue 或 Topic。它使用 Spring 的 JMS 支持进行声明事务，包括 Spring 的 JmsTemplate 来发送和使用 MessageListenerContainer。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jms</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

179.2. URI 格式

```
jms:[queue:|topic:]destinationName[?options]
```

其中 destinationName 是 JMS 队列或主题名称。默认情况下，destinationName 解释为队列名称。例如，要连接到队列，使用 FOO.BAR ：

```
jms:FOO.BAR
```

如果需要，您可以包含可选的 queue: 前缀：

```
jms:queue:FOO.BAR
```

要连接到主题，您必须 包含 topic: 前缀。例如，连接到主题 Stocks.Prices，请使用：

```
jms:topic:Stocks.Prices
```

您可以使用以下格式将查询选项附加到 URI 中，？option=value&amp ;option=value&...

179.3. 注

179.3.1. 使用 ActiveMQ

JMS 组件重复使用 Spring 2 的 JmsTemplate 来发送消息。这不是在非J2EE 容器中使用的理想选择，通常需要 JMS 提供程序中的一些缓存以避免 性能不佳。

如果要将 Apache ActiveMQ 用作消息代理，建议执行以下操作之一：

- 使用 ActiveMQ 组件，该组件经过优化以高效地使用 ActiveMQ
- 在 ActiveMQ 中使用 PoolingConnectionFactory。

179.3.2. 事务和缓存级别

如果您使用消息并使用事务(transacted=true)，缓存级别的默认设置可能会影响性能。

如果您使用 XA 事务，则无法缓存，因为它可能会导致 XA 事务无法正常工作。

如果您不使用 XA，则应考虑缓存速度，如设置 cacheLevelName=CACHE_CONSUMER。

通过 Camel 2.7.x，cacheLevelName 的默认设置是 CACHE_CONSUMER。您需要显式设置 cacheLevelName=CACHE_NONE。

在 Camel 2.8 以后，cacheLevelName 的默认设置是 CACHE_AUTO。这个默认会自动检测模式，并相应地将缓存级别设置为：

- 如果 transacted=false
- 如果 transacted=true

因此，您可以说默认设置比较保守。如果您使用非 XA 事务，请考虑使用 cacheLevelName=CACHE_CONSUMER。

179.3.3. 持久化订阅

如果要使用持久主题订阅，则需要同时指定 clientId 和 durableSubscriptionName。clientId 的值必须是唯一的，且只能由整个网络中的单个 JMS 连接实例使用。您可能更喜欢使用 虚拟主题 以避免这种限制。有关持久化消息传递的更多背景，此处.

179.3.4. Message Header Mapping

在使用消息标头时，JMS 规范指出标头名称必须是有效的 Java 标识符。因此，请尝试将您的标头命名为有效的 Java 标识符。执行此操作的一个优点是，您可以在 JMS Selector 中使用您的标头（其 SQL92 语法强制使用标头的 Java 标识符语法）。

默认使用映射标头名称的简单策略。策略是替换标头名称中的任何点和连字符，并在从线上发送的 JMS 消息恢复标头名称时反转替换。这意味着什么？不会再丢失在 bean 组件上调用的方法名称，不再丢失 File 组件的文件名标头，以此类推。

在 Camel 中接受标头名称的当前标头名称策略如下：

- *当 Camel 使用消息时，点被 DOT 替换，替换会被反向替换*
- *hyphen 被 HYPHEN 替换，当 Camel 使用消息时，替换会被反向替换*

179.4. 选项

您可以在 JMS 端点上配置许多不同的属性，该端点映射到 JMSConfiguration POJO 的属性。

警告

映射到 Spring JMS

许多这些属性映射到 Spring JMS 上的属性，Camel 用于发送和接收消息。因此，您可以通过咨询相关的 Spring 文档来获取有关这些属性的更多信息。

179.4.1. 组件选项

JMS 组件支持下列列出的 81 选项：

Name	描述	默认值	类型
配置（高级）	使用共享的 JMS 配置		JmsConfiguration
acceptMessages While Stopping (consumer)	指定消费者在停止时是否接受消息。如果您在运行时启动和停止 JMS 路由，同时队列中仍然有消息排队，您可以考虑启用此选项。如果此选项为 false，并且您停止 JMS 路由，则消息可能会被拒绝，并且 JMS 代理必须尝试恢复，并且再次可能被拒绝，最后该消息可能在 JMS 代理的死信队列中移动。为避免这种情况，建议启用这个选项。	false	布尔值

Name	描述	默认值	类型
allowReplyManagerQuick Stop (consumer)	在 request-reply 消息传递的回复管理器中使用的 DefaultMessageListenerContainer 是否允许 DefaultMessageListenerContainer.runningAllowed 标志快速停止，以防 JmsConfigurationBuildisAcceptMessagesWhileStopping 已启用，并且 org.apache.camel.CamelContext 当前已停止。默认情况下，在常规 JMS 用户中启用此快速停止功能，但要启用回复管理器，您必须启用此标志。	false	布尔值
acknowledgementMode (consumer)	JMS 确认模式定义为整数。允许您将特定于供应商的扩展设置为确认模式。对于常规模式，最好改用 confirmmentModeName。		int
eagerLoadingOfProperties (consumer)	加载消息时，启用强制加载 JMS 属性，这通常效率低效，因为 JMS 属性可能并不需要，但有时可以提前捕获与底层 JMS 提供程序的任何问题，以及使用 JMS 属性	false	布尔值
acknowledgementModeName (consumer)	JMS 确认名称，即 SESSION_TRANSACTED、CLIENT_ACKNOWLEDGE、AUTO_ACKNOWLEDGE, DUPS_OK_ACKNOWLEDGE	AUTO_确认	字符串
autoStartup (consumer)	指定消费者容器是否应自动启动。	true	布尔值
cacheLevel (consumer)	根据底层 JMS 资源的 ID 设置缓存级别。如需了解更多详细信息，请参阅 cacheLevelName 选项。		int
cacheLevelName (consumer)	按名称为底层 JMS 资源设置缓存级别。可能的值有：CACHE_AUTO、CACHE_CONNECTION、CACHE_CONSUMER、CACHE_NONE 和 CACHE_SESSION。默认设置为 CACHE_AUTO。如需更多信息，请参阅 Spring 文档和事务缓存级别。	CACHE_AUTO	字符串
replyToCacheLevelName (producer)	在通过 JMS 进行请求/回复时，按名称设置缓存级别。这个选项只适用于使用固定回复队列（而不是临时）。Camel 默认将使用：CACHE_CONSUMER 用于 exclusive 或 shared w/ replyToSelectorName。和 CACHE_SESSION 用于在没有 replyToSelectorName 的情况下共享。有些 JMS 代理（如 IBM WebSphere）可能需要设置 replyToCacheLevelName=CACHE_NONE 才能工作。注意：如果不使用临时队列，则不允许 CACHE_NONE，且您必须使用更高的值，如 CACHE_CONSUMER 或 CACHE_SESSION。		字符串

Name	描述	默认值	类型
clientId (common)	设置要使用的 JMS 客户端 ID。请注意，如果指定，这个值必须是唯一的，且只能由单个 JMS 连接实例使用。它通常只需要 durable 主题订阅。如果使用 Apache ActiveMQ，您可能更喜欢使用虚拟主题。		字符串
concurrentConsumers (consumer)	指定从 JMS 消耗时的默认并发消费者数（而不是通过 JMS 进行请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。当通过 JMS 进行请求/回复时，会使用选项 replyToConcurrentConsumers 来控制回复消息监听器上的并发用户数量。	1	int
replyToConcurrent Consumers (producer)	指定通过 JMS 进行请求/回复时的默认并发消费者数量。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。	1	int
ConnectionFactory (common)	要使用的连接工厂。必须在组件或端点上配置连接工厂。		ConnectionFactory
用户名 (security)	与 ConnectionFactory 搭配使用的用户名。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
密码 (security)	与 ConnectionFactory 搭配使用的密码。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
deliveryPersistent (producer)	指定是否默认使用持久性交付。	true	布尔值
deliveryMode (producer)	指定要使用的交付模式。可能的值有 Possibles 值，这些值由 javax.jms.DeliveryMode 定义。NON_PERSISTENT = 1 和 PERSISTENT = 2.		整数
durableSubscriptionName (common)	用于指定持久主题订阅的持久化订阅者名称。还必须配置 clientId 选项。		字符串
exceptionListener (advanced)	指定正在获得任何底层 JMS 异常通知的 JMS Exception Listener。		ExceptionListener
errorHandler (advanced)	指定在处理消息时引发任何未捕获的异常时调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置 errorHandler，这些例外将在 WARN 级别记录。您可以配置日志记录级别，并使用 errorHandlerLoggingLevel 和 errorHandlerLogStackTrace 选项记录堆栈 trace。这样可以更轻松地配置，而不是对自定义 errorHandler 进行编码。		ErrorHandler

Name	描述	默认值	类型
errorHandlerLoggingLevel (logging)	允许为日志记录未捕获的异常配置默认的 errorHandler 日志记录级别。	WARN	LoggingLevel
errorHandlerLogStackTrace (logging)	允许由默认的 errorHandler 控制是否应记录堆栈追踪。	true	布尔值
explicitQosEnabled (producer)	设置在发送消息时应使用 deliveryMode、priority 或 timeToLive qualities。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用到当前端点。这与 preserveMessageQos 选项相反，它以消息粒度运行，从 Camel In 消息标头读取 QoS 属性。	false	布尔值
exposeListenerSession (consumer)	指定侦听器会话是否应该在消耗消息时公开。	false	布尔值
idleTaskExecutionLimit (advanced)	指定闲置执行接收任务的限值，没有在其执行中收到任何消息。如果达到这个限制，任务将关闭并保持接收其他执行任务（如果是动态调度，请参阅 maxConcurrentConsumers 设置）。Spring 中提供了额外的文档。	1	int
idleConsumerLimit (advanced)	指定在任意给定时间允许闲置的用户数量的限制。	1	int
maxConcurrentConsumers (consumer)	指定从 JMS 消耗时的最大并发消费者数（而不是通过 JMS 进行请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。当通过 JMS 进行请求/回复时，选项 replyToMaxConcurrentConsumers 用于控制回复消息监听器上的并发用户数量。		int
replyToMaxConcurrentConsumers (producer)	指定通过 JMS 使用请求/回复时的最大并发消费者数。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。		int
replyOnTimeoutToMaxConcurrentConsumers (producer)	指定在通过 JMS 使用请求/回复时超时时继续路由的最大并发消费者数。	1	int
maxMessagesPerTask (advanced)	每个任务的消息数量。-1 代表没有限制。如果您将范围用于并发消费者（如 min max），则可以使用此选项将值设置为 eg 100，以控制消费者在需要较少的工作时缩小的速度。	-1	int

Name	描述	默认值	类型
messageConverter (advanced)	要使用自定义 Spring org.springframework.jms.support.converter.MessageConverter，以便您可以控制如何映射到 javax.jms.Message。		MessageConverter
mapJmsMessage (advanced)	指定 Camel 是否应该自动将收到的 JMS 消息映射到合适的有效负载类型，如 javax.jms.TextMessage 到 String 等。如需了解更多详细信息，请参阅映射如何工作。	true	布尔值
messageIdEnabled (advanced)	发送时，指定是否应添加消息 ID。这只是对 JMS Broker 的提示。如果 JMS 提供程序接受此提示，则这些消息必须将消息 ID 设为 null；如果提供程序忽略提示，则必须将消息 ID 设置为其普通唯一值	true	布尔值
messageTimestampEnabled (advanced)	指定在发送消息时是否默认启用时间戳。	true	布尔值
alwaysCopyMessage (producer)	如果为 true，Camel 会在传递给制作者以进行发送时始终生成消息的 JMS 消息副本。在某些情况下需要复制消息，例如当设置了 replyToDestinationSelectorName 时（如果设置了 replyToDestinationSelectorName，Camel 会将 alwaysCopyMessage 选项设置为 true，如果设置了 replyToDestinationSelectorName ）	false	布尔值
useMessageIDAsCorrelationID (advanced)	指定 JMSMessageID 是否应该始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值
priority (producer)	大于 1 的值在发送时指定消息优先级（其中 0 是最低优先级，9 是最高）。必须启用 explicitQosEnabled 选项，才能使此选项有任何效果。	4	int
pubSubNoLocal (advanced)	指定是否禁止发送由其自身连接发布的消息。	false	布尔值
receiveTimeout (advanced)	接收消息的超时时间（以毫秒为单位）。	1000	long
recoveryInterval (advanced)	指定恢复尝试之间的间隔，例如当连接被刷新时，以毫秒为单位。默认值为 5000 ms，即 5 秒。	5000	long
taskExecutor (consumer)	允许您指定用于消耗消息的自定义任务 executor。		TaskExecutor

Name	描述	默认值	类型
deliveryDelay (producer)	设置用于 JMS 发送调用的交付延迟。此选项要求 JMS 2.0 兼容代理。	-1	long
timeToLive (producer)	发送消息时，指定消息的生存时间（以毫秒为单位）。	-1	long
转换 （事务）	指定是否使用转换模式	false	布尔值
lazyCreateTransaction Manager (transaction)	如果为 true，如果选项 transacted=true 时没有注入的 transactionManager，则 Camel 将创建一个 JmsTransactionManager。	true	布尔值
transactionManager (transaction)	要使用的 Spring 事务管理器。		平台交易管理器
transactionName (transaction)	要使用的事务的名称。		字符串
transactionTimeout (transaction)	如果使用转换模式，事务的超时值（以秒为单位）。	-1	int
testConnectionOn Startup (common)	指定是否在启动时测试连接。这样可确保当 Camel 启动所有 JMS 用户与 JMS 代理的有效连接时。如果无法授予连接，则 Camel 会在启动时抛出异常。这样可确保 Camel 不使用失败的连接启动。JMS producers 也经过测试。	false	布尔值
asyncStartListener (advanced)	在启动路由时，是否异步启动 JmsConsumer 消息监听程序。例如，如果 JmsConsumer 无法连接到远程 JMS 代理，则在重试和/或故障转移时可能会阻止。这将导致 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 以异步模式使用专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果无法建立连接，那么会记录异常在 WARN 级别，消费者将无法接收消息；然后您可以重启路由来重试。	false	布尔值
asyncStopListener (advanced)	在停止路由时，是否异步停止 JmsConsumer 消息监听程序。	false	布尔值
forceSendOriginal Message (producer)	使用 mapJmsMessage=false Camel 时，如果您涉及路由期间的标头(get 或 set)，则会创建一个新的 JMS 消息来发送到新的 JMS 目的地。将此选项设置为 true，以强制 Camel 发送收到的原始 JMS 消息。	false	布尔值

Name	描述	默认值	类型
requestTimeout (producer)	使用 InOut Exchange Pattern 时等待回复的超时（以毫秒为单位）。默认值为 20 秒。您可以包含标头 CamelJmsRequestTimeout 来覆盖此端点配置的超时值，因此每个消息都有单独的超时值。另请参阅 requestTimeoutCheckerInterval 选项。	20000	long
requestTimeoutChecker Interval (advanced)	配置 Camel 在通过 JMS 进行请求/回复时应检查超时的频率。默认情况下，Camel 检查每秒一次。但是，如果发生超时时必须更快地响应，则可以降低这个间隔，以便更频繁地检查。超时由 option requestTimeout 决定。	1000	long
transferExchange (advanced)	您可以通过线线传输交换，而不只是正文和标头。以下字段会被传输： 在 body, Out body, Fault body, In headers, Out headers, Fault 标头, Exchange properties, Exchange exception。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。您必须在生成者和消费者端启用这个选项，因此 Camel 知道有效负载是交换而不是常规有效负载。	false	布尔值
transferException (advanced)	如果启用并且您使用 Request Reply messaging (InOut)和在消费者端的 Exchange 失败，则原因 Exception 将发回为 javax.jms.ObjectMessage 的响应。如果客户端是 Camel，则返回的 Exception 为 rerown。这允许您使用 Camel JMS 作为路由中的网桥，例如，使用持久性队列来启用强大的路由。请注意，如果您也启用了 transferExchange，这个选项会优先使用。需要被捕获的例外是串行的。在返回到生成者时，消费者侧的原始 Exception 可以嵌套在外部异常中，如 org.apache.camel.RuntimeCamelException。	false	布尔值
transferFault (advanced)	如果启用且您使用 Request Reply messaging (InOut)，且交换失败，在消费者端使用 SOAP 错误（而不是异常），则 org.apache.camel.MessageAllisFault （）上的错误标志将作为 JMS 标头发送，其键为 JmsConstants#JMS_TRANSFER_FAULT。如果客户端是 Camel，则返回的 fault 标志将在 org.apache.camel.MessageAllsetFault (boolean)上设置。在使用支持错误（如 SOAP）（如 cxf 或 spring-ws）的 Camel 组件时，您可能需要启用此功能。	false	布尔值
jmsOperations (advanced)	允许您使用您自己的 org.springframework.jms.core.JmsOperations 接口实施。Camel 使用 JmsTemplate 作为默认值。可用于测试目的，但不按 spring API 文档中所述使用。		JmsOperations

Name	描述	默认值	类型
destinationResolver (advanced)	可插拔 org.springframework.jms.support.destination.DestinationResolver，允许您使用您自己的解析器（例如，在 JNDI 注册表中查找实际目的地）。		DestinationResolver
replyToType (producer)	允许明确指定在通过 JMS 进行请求/回复时，用于 replyTo 队列的策略。可能的值有： Temporary、Shared 或 Exclusive。默认情况下，Camel 将使用临时队列。但是，如果配置了 replyTo，则默认使用 Shared。此选项允许您使用专用队列而不是共享队列。如需了解更多详细信息，请参阅 Camel JMS 文档，特别是有关在集群环境中运行时所影响的备注，而共享回复队列的性能比其 alternatives Temporary 和 Exclusive 性能较低。		ReplyToType
preserveMessageQos (producer)	如果要使用消息上指定的 QoS 设置发送消息，而不是 JMS 端点上的 QoS 设置，则设置为 true。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供所有或只提供其中一些。如果没有提供，Camel 将回退到使用端点中的值。因此，在使用此选项时，标头会覆盖来自端点的值。相反，explicitQosEnabled 选项只会使用端点上设置的选项，而不使用消息标头中设置的值。	false	布尔值
asyncConsumer (consumer)	JmsConsumer 是否异步处理 Exchange。如果启用，JmsConsumer 可能会从 JMS 队列中选择下一个消息，而前面的消息异步处理（通过 Asynchronous Routing Engine）。这意味着消息可能没有完全严格按照顺序进行处理。如果禁用（作为默认），则在 JmsConsumer 将从 JMS 队列获取下一个消息之前完全处理 Exchange。请注意，如果启用了 transacted，则 asyncConsumer=true 不会异步运行，因为事务必须同步执行(Camel 3.0 可能支持 async 事务)。	false	布尔值
allowNullBody (producer)	是否允许发送没有正文的消息。如果此选项为 false，并且消息正文为 null，则抛出 JMSException。	true	布尔值
includeSentJMSMessageID (producer)	仅在使用 InOnly （如触发和忘记）发送到 JMS 目的地时才适用。启用此选项将使用实际的 JMSMessageID 增强 Camel Exchange，当消息发送到 JMS 目的地时，由 JMS 客户端使用。	false	布尔值
includeAllJMSXProperties (advanced)	从 JMS 映射到 Camel 消息时，是否要包含所有 JMSXxxx 属性。把它设置为 true 将包括 JMSXAppID 和 JMSXUserID 等属性。注： 如果您使用自定义 headerFilterStrategy，则不会应用这个选项。	false	布尔值

Name	描述	默认值	类型
defaultTaskExecutor Type (consumer)	指定在 DefaultMessageListenerContainer 中使用的默认 TaskExecutor 类型，用于消费者端点和生成者端点的 ReplyTo consumer。可能的值： SimpleAsync（使用 Spring 的 SimpleAsyncTaskExecutor）或 ThreadPool （使用 Spring 的 ThreadPoolTaskExecutor 带有最佳值 - 缓存线程池）。如果没有设置，则默认为之前的行为，它使用缓存的线程池作为消费者端点，而 SimpleAsync 用于回复消费者。建议使用 ThreadPool，以动态增加和减少并发消费者在弹性配置中减少线程回收。		DefaultTaskExecutor Type
jmsKeyFormatStrategy (advanced)	可插拔编码和解码 JMS 密钥策略，以便它们能够与 JMS 规范兼容。Camel 提供了两个开箱即用的实现：default 和 passthrough。默认策略可以安全地放入点和连字符(. 和 -)。passthrough 策略将密钥保留原样。可用于 JMS 代理，其不小心是 JMS 标头键是否包含非法字符。您可以提供自己的 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 # 表示法引用它。		JmsKeyFormatStrategy
allowAdditionalHeaders (producer)	此选项用于允许其他标头，这些标头可能具有根据 JMS 规格无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 包含带有字节数组或其他无效类型的值的标头名称。您可以指定用逗号分开的多个标头名称，并使用 作为后缀进行通配符匹配。		字符串
queueBrowseStrategy (advanced)	在浏览队列时使用自定义 QueueBrowseStrategy		QueueBrowseStrategy
messageCreatedStrategy (advanced)	要使用给定的 MessageCreatedStrategy，后者在 Camel 在 Camel 发送 JMS 消息时创建 javax.jms.Message 对象的新实例。		MessageCreatedStrategy
waitForProvisionCorrelationToBeUpdated Counter (advanced)	在通过 JMS 进行请求/回复时，以及启用选项 useMessageIDAsCorrelationID 时，等待 provisional correlation id 的次数。	50	int
waitForProvisionCorrelationToBeUpdated ThreadSleepingTime (advanced)	millis 中的间隔，在每次等待配置关联 ID 时处于睡眠状态。	100	long

Name	描述	默认值	类型
correlationProperty (producer)	使用此 JMS 属性来关联 InOut Exchange pattern (request-reply)中的消息，而不是 JMSCorrelationID 属性。这样，您可以将消息与不使用 JMSCorrelationID JMS 属性关联消息的系统交换。如果使用的 JMSCorrelationID 不会被 Camel 使用或设置。如果没有在相同名称下的消息标头中提供，则会生成此处 named 属性的值。		字符串
subscriptionDurable (consumer)	设置是否使订阅持久。要使用的持久订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册持久化订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。仅在侦听主题(pub-sub domain)时具有意义，因此此方法也会切换 pubSubDomain 标志。	false	布尔值
subscriptionShared (consumer)	设置是否使订阅共享。要使用的共享订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。请注意，共享订阅也可能是持久的，因此此标志也可以与 subscriptionDurable 结合使用。仅在侦听主题(pub-sub domain)时具有意义，因此此方法也会切换 pubSubDomain 标志。需要兼容 JMS 2.0 的消息代理。	false	布尔值
subscriptionName (consumer)	设置要创建的订阅的名称。当具有共享或持久化订阅的主题(pub-sub domain)时应用。订阅名称需要在此客户端的 JMS 客户端 id 中唯一。default 是指定消息监听程序的类名称。注意：除了共享订阅（需要 JMS 2.0）外，每个订阅只允许 1 个并发消费者（此消息监听程序容器的默认值）。		字符串
streamMessageType Enabled (producer)	设置是否启用 StreamMessage 类型。文件、InputStream 等消息有效负载（如文件、InputStream 等）将通过 BytesMessage 或 StreamMessage 发送。此选项控制将使用哪种类型。默认情况下，使用 BytesMessage 来强制将整个消息有效负载读取到内存中。通过启用此选项，消息有效负载以块的形式读取内存，然后每个块写入 StreamMessage，直到没有更多数据。	false	布尔值
formatDateHeadersTo Iso8601 (producer)	设置日期标头是否应根据 ISO 8601 标准进行格式化。	false	布尔值
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

179.4.2. 端点选项

JMS 端点使用 URI 语法进行配置：

```
jms:destinationType:destinationName
```

使用以下路径和查询参数：

179.4.3. 路径参数(2 参数)：

Name	描述	默认值	类型
destinationType	要使用的目的地类型	queue	字符串
destinationName	用作目的地的队列或主题的 **必需** 名称		字符串

179.4.4. 查询参数(93 参数)：

Name	描述	默认值	类型
clientId (common)	设置要使用的 JMS 客户端 ID。请注意，如果指定，这个值必须是唯一的，且只能由单个 JMS 连接实例使用。它通常只需要 durable 主题订阅。如果使用 Apache ActiveMQ，您可能更喜欢使用虚拟主题。		字符串
ConnectionFactory (common)	如果没有为 setTemplateConnectionFactory (ConnectionFactory)或 setListenerConnectionFactory (ConnectionFactory) 指定连接工厂，则设置默认的连接工厂。		ConnectionFactory
disableReplyTo (common)	指定 Camel 是否忽略消息中的 JMSReplyTo 标头。如果为 true，Camel 不会发送回复回 JMSReplyTo 标头中指定的目的地。如果您希望 Camel 从路由消耗且您不希望 Camel 自动发送回复消息，因为代码中的另一个组件处理回复消息，则可以使用这个选项。如果要使用 Camel 作为不同消息代理之间的代理，并希望将消息从一个系统路由到另一个系统，也可以使用这个选项。	false	布尔值

Name	描述	默认值	类型
durableSubscriptionName (common)	用于指定持久主题订阅的持久化订阅者名称。还必须配置 clientId 选项。		字符串
jmsMessageType (common)	允许您强制使用特定的 javax.jms.Message 实施来发送 JMS 消息。可能的值有：Bytes, Map, Object, Stream, Text。默认情况下，Camel 将决定要从 In 正文类型使用哪个 JMS 消息类型。这个选项允许您指定它。		JmsMessageType
testConnectionOnStartup (common)	指定是否在启动时测试连接。这样可确保当 Camel 启动所有 JMS 用户与 JMS 代理的有效连接时。如果无法授予连接，则 Camel 会在启动时抛出异常。这样可确保 Camel 不使用失败的连接启动。JMS producers 也经过测试。	false	布尔值
acknowledgementModeName (consumer)	JMS 确认名称，即 SESSION_TRANSACTED、CLIENT_ACKNOWLEDGE、AUTO_ACKNOWLEDGE, DUPS_OK_ACKNOWLEDGE	AUTO_确认	字符串
asyncConsumer (consumer)	JmsConsumer 是否异步处理 Exchange。如果启用，JmsConsumer 可能会从 JMS 队列中选择下一个消息，而前面的消息异步处理（通过 Asynchronous Routing Engine）。这意味着消息可能没有完全严格按照顺序进行处理。如果禁用（作为默认），则在 JmsConsumer 将从 JMS 队列获取下一个消息之前完全处理 Exchange。请注意，如果启用了 transacted，则 asyncConsumer=true 不会异步运行，因为事务必须同步执行(Camel 3.0 可能支持 async 事务)。	false	布尔值
autoStartup (consumer)	指定消费者容器是否应自动启动。	true	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
cacheLevel (consumer)	根据底层 JMS 资源的 ID 设置缓存级别。如需了解更多详细信息，请参阅 cacheLevelName 选项。		int
cacheLevelName (consumer)	按名称为底层 JMS 资源设置缓存级别。可能的值有：CACHE_AUTO、CACHE_CONNECTION、CACHE_CONSUMER、CACHE_NONE 和 CACHE_SESSION。默认设置为 CACHE_AUTO。如需更多信息，请参阅 Spring 文档和事务缓存级别。	CACHE_AUTO	字符串

Name	描述	默认值	类型
concurrentConsumers (consumer)	指定从 JMS 消耗时的默认并发消费者数（而不是通过 JMS 进行请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。当通过 JMS 进行请求/回复时，会使用选项 replyToConcurrentConsumers 来控制回复消息监听器上的并发用户数量。	1	int
artemisConsumerPriority (consumer)	通过消费者优先级，您可以确保高优先级使用者在激活时收到消息。通常，连接到队列的活跃用户以轮循方式接收来自其的消息。使用消费者优先级时，如果多个活动消费者具有相同高的优先级，则会对消息发送循环。只有在高优先级消费者没有可用的信用时，消息才会进入较低优先级消费者，或者那些高优先级消费者已拒绝接受消息（例如，因为它不符合与消费者关联的任何选择器的条件）。		int
maxConcurrentConsumers (consumer)	指定从 JMS 消耗时的最大并发消费者数（而不是通过 JMS 进行请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。当通过 JMS 进行请求/回复时，选项 replyToMaxConcurrentConsumers 用于控制回复消息监听器上的并发用户数量。		int
replyTo (consumer)	提供一个显式 ReplyTo 目的地，它会覆盖任何 Message.getJMSReplyTo （）的任何传入值。		字符串
replyToDeliveryPersistent (consumer)	指定默认情况下是否使用持久发送回复。	true	布尔值
selector (consumer)	设置要使用的 JMS 选择器		字符串
subscriptionDurable (consumer)	设置是否使订阅持久。要使用的持久订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册持久化订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。仅在侦听主题(pub-sub domain)时具有意义，因此此方法也会切换 pubSubDomain 标志。	false	布尔值
subscriptionName (consumer)	设置要创建的订阅的名称。当具有共享或持久化订阅的主题(pub-sub domain)时应用。订阅名称需要在此客户端的 JMS 客户端 id 中唯一。default 是指定消息监听程序的类名称。注意：除了共享订阅（需要 JMS 2.0）外，每个订阅只允许 1 个并发消费者（此消息监听程序容器的默认值）。		字符串

Name	描述	默认值	类型
subscriptionShared (consumer)	设置是否使订阅共享。要使用的共享订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。请注意，共享订阅也可能是持久的，因此此标志也可以与 subscriptionDurable 结合使用。仅在侦听主题(pub-sub domain)时具有意义，因此此方法也会切换 pubSubDomain 标志。需要兼容 JMS 2.0 的消息代理。	false	布尔值
acceptMessagesWhileStopping (consumer)	指定消费者在停止时是否接受消息。如果您在运行时启动和停止 JMS 路由，同时队列中仍然有消息排队，您可以考虑启用此选项。如果此选项为 false，并且您停止 JMS 路由，则消息可能会被拒绝，并且 JMS 代理必须尝试恢复，并且再次可能被拒绝，最后该消息可能在 JMS 代理的死信队列中移动。为避免这种情况，建议启用这个选项。	false	布尔值
allowReplyManagerQuickStop (consumer)	在 request-reply 消息传递的回复管理器中使用的 DefaultMessageListenerContainer 是否允许 DefaultMessageListenerContainer SerialrunningAllowed （）标志在 JmsConfiguration SerialisAcceptMessagesWhileStopping （）被启用，并且 org.apache.camel.CamelContext 当前被停止。默认情况下，在常规 JMS 用户中启用此快速停止功能，但要启用回复管理器，您必须启用此标志。	false	布尔值
consumerType (consumer)	要使用的消费者类型，可以是： Simple、Default 或 Custom 之一。consumer 类型决定要使用的 Spring JMS 侦听器。default 将使用 org.springframework.jms.listener.DefaultMessageListenerContainer，Simple 将使用 org.springframework.jms.listener.SimpleMessageListenerContainer。当指定 Custom 时，messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 将决定要使用的 org.springframework.jms.listener.AbstractMessageListenerContainer。	default	ConsumerType

Name	描述	默认值	类型
defaultTaskExecutorType (consumer)	指定在 DefaultMessageListenerContainer 中使用的默认 TaskExecutor 类型，用于消费者端点和生成者端点的 ReplyTo consumer。可能的值： SimpleAsync（使用 Spring 的 SimpleAsyncTaskExecutor）或 ThreadPool （使用 Spring 的 ThreadPoolTaskExecutor 带有最佳值 - 缓存线程池）。如果没有设置，则默认为之前的行为，它使用缓存的线程池作为消费者端点，而 SimpleAsync 用于回复消费者。建议使用 ThreadPool，以动态增加和减少并发消费者在弹性配置中减少线程回收。		DefaultTaskExecutor Type
eagerLoadingOfProperties (consumer)	在加载消息时，启用 JMS 属性和有效负载的强制加载，这通常效率低效，因为 JMS 属性可能并不是必需的，但有时可以在底层 JMS 提供程序和 JMS 属性中使用任何问题时捕获早期出现的问题	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
exposeListenerSession (consumer)	指定侦听器会话是否应该在消耗消息时公开。	false	布尔值
replyToSameDestination Allowed (consumer)	是否允许 JMS 使用者向消费者使用的同一目的地发送回复消息。这可防止通过消耗并向其自身发送相同的消息，从而防止无限循环。	false	布尔值
taskExecutor (consumer)	允许您指定用于消耗消息的自定义任务 executor。		TaskExecutor
deliveryDelay (producer)	设置用于 JMS 发送调用的交付延迟。此选项要求 JMS 2.0 兼容代理。	-1	long
deliveryMode (producer)	指定要使用的交付模式。Possibles 值是由 javax.jms.DeliveryMode 定义的值。NON_PERSISTENT = 1 和 PERSISTENT = 2.		整数
deliveryPersistent (producer)	指定是否默认使用持久性交付。	true	布尔值

Name	描述	默认值	类型
explicitQosEnabled (producer)	设置在发送消息时应使用 deliveryMode、priority 或 timeToLive qualities。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用到当前端点。这与 preserveMessageQos 选项相反，它以消息粒度运行，从 Camel In 消息标头读取 QoS 属性。	false	布尔值
formatDateHeadersToIso8601 (producer)	设置日期标头是否应根据 ISO 8601 标准进行格式化。	false	布尔值
preserveMessageQos (producer)	如果要使用消息上指定的 QoS 设置发送消息，而不是 JMS 端点上的 QoS 设置，则设置为 true。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供所有或只提供其中一些。如果没有提供，Camel 将回退到使用端点中的值。因此，在使用此选项时，标头会覆盖来自端点的值。相反，explicitQosEnabled 选项只会使用端点上设置的选项，而不使用消息标头中设置的值。	false	布尔值
priority (producer)	大于 1 的值在发送时指定消息优先级（其中 0 是最低优先级，9 是最高）。必须启用 explicitQosEnabled 选项，才能使此选项有任何效果。	4	int
replyToConcurrentConsumers (producer)	指定通过 JMS 进行请求/回复时的默认并发消费者数量。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。	1	int
replyToMaxConcurrent Consumers (producer)	指定通过 JMS 使用请求/回复时的最大并发消费者数。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。		int
replyToOnTimeoutMax ConcurrentConsumers (producer)	指定在通过 JMS 使用请求/回复时超时时继续路由的最大并发消费者数。	1	int
replyToOverride (producer)	在 JMS 消息中提供一个显式 ReplyTo 目的地，它将覆盖 replyTo 的设置。如果您要将消息转发到远程 Queue，并从 ReplyTo 目的地接收回复消息，这非常有用。		字符串

Name	描述	默认值	类型
replyToType (producer)	允许明确指定在通过 JMS 进行请求/回复时，用于 replyTo 队列的策略。可能的值有： Temporary、Shared 或 Exclusive。默认情况下，Camel 将使用临时队列。但是，如果配置了 replyTo，则默认使用 Shared。此选项允许您使用专用队列而不是共享队列。如需了解更多详细信息，请参阅 Camel JMS 文档，特别是有关在集群环境中运行时所影响的备注，而共享回复队列的性能比其 alternatives Temporary 和 Exclusive 性能较低。		ReplyToType
requestTimeout (producer)	使用 InOut Exchange Pattern 时等待回复的超时（以毫秒为单位）。默认值为 20 秒。您可以包含标头 CamelJmsRequestTimeout 来覆盖此端点配置的超时值，因此每个消息都有单独的超时值。另请参阅 requestTimeoutCheckerInterval 选项。	20000	long
timeToLive (producer)	发送消息时，指定消息的生存时间（以毫秒为单位）。	-1	long
allowAdditionalHeaders (producer)	此选项用于允许其他标头，这些标头可能具有根据 JMS 规格无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 包含带有字节数组或其他无效类型的值的标头名称。您可以指定用逗号分开的多个标头名称，并使用 作为后缀进行通配符匹配。		字符串
allowNullBody (producer)	是否允许发送没有正文的消息。如果此选项为 false，并且消息正文为 null，则抛出 JMSException。	true	布尔值
alwaysCopyMessage (producer)	如果为 true，Camel 会在传递给制作者以进行发送时始终生成消息的 JMS 消息副本。在某些情况下需要复制消息，例如当设置了 replyToDestinationSelectorName 时（如果设置了 replyToDestinationSelectorName，Camel 会将 alwaysCopyMessage 选项设置为 true，如果设置了 replyToDestinationSelectorName ）	false	布尔值
correlationProperty (producer)	使用此 JMS 属性来关联 InOut Exchange pattern (request-reply)中的消息，而不是 JMSCorrelationID 属性。这样，您可以将消息与不使用 JMSCorrelationID JMS 属性关联消息的系统交换。如果使用的 JMSCorrelationID 不会被 Camel 使用或设置。如果没有在相同名称下的消息标头中提供，则会生成此处 named 属性的值。		字符串

Name	描述	默认值	类型
disableTimeToLive (producer)	使用这个选项强制禁用生存时间。例如，当您通过 JMS 进行请求/回复时，Camel 默认将使用 requestTimeout 值作为所发送消息的时间。问题是发送者和接收器系统必须同步其时钟，因此它们处于同步状态。这并非总是容易进行归档。因此，您可以使用 disableTimeToLive=true 设置在发送的消息上生存时间值。然后，消息不会在接收方系统中过期。如需了解更多详细信息，请参见关于生存时间 一节中的。	false	布尔值
forceSendOriginalMessage (producer)	使用 mapJmsMessage=false Camel 时，如果您涉及路由期间的标头(get 或 set)，则会创建一个新的 JMS 消息来发送到新的 JMS 目的地。将此选项设置为 true，以强制 Camel 发送收到的原始 JMS 消息。	false	布尔值
includeSentJMSMessageID (producer)	仅在使用 InOnly （如触发和忘记）发送到 JMS 目的地时才适用。启用此选项将使用实际的 JMSMessageID 增强 Camel Exchange，当消息发送到 JMS 目的地时，由 JMS 客户端使用。	false	布尔值
replyToCacheLevelName (producer)	在通过 JMS 进行请求/回复时，按名称设置缓存级别。这个选项只适用于使用固定回复队列（而不是临时）。Camel 默认将使用：CACHE_CONSUMER 用于 exclusive 或 shared w/ replyToSelectorName。和 CACHE_SESSION 用于在没有 replyToSelectorName 的情况下共享。有些 JMS 代理（如 IBM WebSphere）可能需要设置 replyToCacheLevelName=CACHE_NONE 才能工作。注意：如果不使用临时队列，则不允许 CACHE_NONE，且您必须使用更高的值，如 CACHE_CONSUMER 或 CACHE_SESSION。		字符串
replyToDestinationSelectorName (producer)	使用固定名称设置 JMS Selector，以便在使用共享队列时过滤您自己的回复信息（也就是说，如果您不使用临时回复队列）。		字符串
streamMessageTypeEnabled (producer)	设置是否启用 StreamMessage 类型。文件、InputStream 等消息有效负载（如文件、InputStream 等）将通过 BytesMessage 或 StreamMessage 发送。此选项控制将使用哪种类型。默认情况下，使用 BytesMessage 来强制将整个消息有效负载读取到内存中。通过启用此选项，消息有效负载以块的形式读取内存，然后每个块写入 StreamMessage，直到没有更多数据。	false	布尔值
allowSerializedHeaders (advanced)	控制是否包含序列化标头。仅在 isTransferExchange （）为 true 时才适用。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。	false	布尔值

Name	描述	默认值	类型
asyncStartListener (advanced)	在启动路由时，是否异步启动 JmsConsumer 消息监听程序。例如，如果 JmsConsumer 无法连接到远程 JMS 代理，则在重试和/或故障转移时可能会阻止。这将导致 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 以异步模式使用专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果无法建立连接，那么会记录异常在 WARN 级别，消费者将无法接收消息；然后您可以重启路由来重试。	false	布尔值
asyncStopListener (advanced)	在停止路由时，是否异步停止 JmsConsumer 消息监听程序。	false	布尔值
destinationResolver (advanced)	可插拔 org.springframework.jms.support.destination.DestinationResolver，允许您使用您自己的解析器（例如，在 JNDI 注册表中查找实际目的地）。		DestinationResolver
errorHandler (advanced)	指定在处理消息时引发任何未捕获的异常时调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置 errorHandler，这些例外将在 WARN 级别记录。您可以配置日志记录级别，并使用 errorHandlerLoggingLevel 和 errorHandlerLogStackTrace 选项记录堆栈 trace。这样可以更轻松地配置，而不是对自定义 errorHandler 进行编码。		ErrorHandler
exceptionListener (advanced)	指定正在获得任何底层 JMS 异常通知的 JMS Exception Listener。		ExceptionListener
headerFilterStrategy (advanced)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
idleConsumerLimit (advanced)	指定在任意给定时间允许闲置的用户数量的限制。	1	int
idleTaskExecutionLimit (advanced)	指定闲置执行接收任务的限值，没有在其执行中收到任何消息。如果达到这个限制，任务将关闭并保持接收其他执行任务（如果是动态调度，请参阅 maxConcurrentConsumers 设置）。Spring 中提供了额外的文档。	1	int
includeAllJMSXProperties (advanced)	从 JMS 映射到 Camel 消息时，是否要包含所有 JMSXxxx 属性。把它设置为 true 将包括 JMSXAppID 和 JMSXUserID 等属性。注： 如果您使用自定义 headerFilterStrategy，则不会应用这个选项。	false	布尔值

Name	描述	默认值	类型
jmsKeyFormatStrategy (advanced)	可插拔编码和解码 JMS 密钥策略，以便它们能够与 JMS 规范兼容。Camel 提供了两个开箱即用的实现：default 和 passthrough。默认策略可以安全地放入点和连字符(. 和 -)。passthrough 策略将密钥保留原样。可用于 JMS 代理，其不小心是 JMS 标头键是否包含非法字符。您可以提供自己的 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 # 表示法引用它。		字符串
mapJmsMessage (advanced)	指定 Camel 是否应该自动将收到的 JMS 消息映射到合适的有效负载类型，如 javax.jms.TextMessage 到 String 等。	true	布尔值
maxMessagesPerTask (advanced)	每个任务的消息数量。-1 代表没有限制。如果您将范围用于并发消费者（如 min max），则可以使用此选项将值设置为 eg 100，以控制消费者在需要较少的工作时缩小的速度。	-1	int
messageConverter (advanced)	要使用自定义 Spring org.springframework.jms.support.converter.MessageConverter，以便您可以控制如何映射到 javax.jms.Message。		MessageConverter
messageCreatedStrategy (advanced)	要使用给定的 MessageCreatedStrategy，后者在 Camel 在 Camel 发送 JMS 消息时创建 javax.jms.Message 对象的新实例。		MessageCreatedStrategy
messageIdEnabled (advanced)	发送时，指定是否应添加消息 ID。这只是对 JMS Broker 的提示。如果 JMS 提供程序接受此提示，则这些消息必须将消息 ID 设为 null；如果提供程序忽略提示，则必须将消息 ID 设置为其普通唯一值	true	布尔值
messageListenerContainerFactory (advanced)	用于确定用于使用消息的 org.springframework.jms.listener.AbstractMessageListenerContainer 的 MessageListenerContainer 的 registry ID。设置此选项将自动将 consumerType 设置为 Custom。		MessageListenerContainerFactory
messageTimestampEnabled (advanced)	指定在发送消息时是否默认启用时间戳。这只是对 JMS Broker 的提示。如果 JMS 提供程序接受此提示，则这些消息必须设置时间戳为零；如果提供程序忽略 hint，则必须将时间戳设置为正常值	true	布尔值
pubSubNoLocal (advanced)	指定是否禁止发送由其自身连接发布的消息。	false	布尔值
receiveTimeout (advanced)	接收消息的超时时间（以毫秒为单位）。	1000	long

Name	描述	默认值	类型
recoveryInterval (advanced)	指定恢复尝试之间的间隔，例如当连接被刷新时，以毫秒为单位。默认值为 5000 ms，即 5 秒。	5000	long
requestTimeoutChecker Interval (advanced)	配置 Camel 在通过 JMS 进行请求/回复时应检查超时的频率。默认情况下，Camel 检查每秒一次。但是，如果发生超时时必须更快地响应，则可以降低这个间隔，以便更频繁地检查。超时由 option requestTimeout 决定。	1000	long
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
transferException (advanced)	如果启用并且您使用 Request Reply messaging (InOut)和在消费者端的 Exchange 失败，则原因 Exception 将发回为 javax.jms.ObjectMessage 的响应。如果客户端是 Camel，则返回的 Exception 为 rerown。这允许您使用 Camel JMS 作为路由中的网桥，例如，使用持久性队列来启用强大的路由。请注意，如果您也启用了 transferExchange，这个选项会优先使用。需要被捕获的例外是串行的。在返回到生成者时，消费者侧的原始 Exception 可以嵌套在外部异常中，如 org.apache.camel.RuntimeCamelException。	false	布尔值
transferExchange (advanced)	您可以通过线线传输交换，而不只是正文和标头。以下字段会被传输： 在 body, Out body, Fault body, In headers, Out headers, Fault 标头, Exchange properties, Exchange exception。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。您必须在生成者和消费者端启用这个选项，因此 Camel 知道有效负载是交换而不是常规有效负载。	false	布尔值
transferFault (advanced)	如果启用且您使用 Request Reply messaging (InOut)，且交换失败，在消费者端使用 SOAP 错误（而不是异常），则 org.apache.camel.MessageAllisFault （）上的错误标志将作为 JMS 标头发送，其键为 JmsConstants#JMS_TRANSFER_FAULT。如果客户端是 Camel，则返回的 fault 标志将在 org.apache.camel.MessageAllsetFault (boolean)上设置。在使用支持错误（如 SOAP）（如 cxf 或 spring-ws）的 Camel 组件时，您可能需要启用此功能。	false	布尔值
useMessageIDAs CorrelationID (advanced)	指定 JMSMessageID 是否应该始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值

Name	描述	默认值	类型
waitForProvisionCorrelationToBeUpdatedCounter (advanced)	在通过 JMS 进行请求/回复时，以及启用选项 useMessageIDAsCorrelationID 时，等待 provisional correlation id 的次数。	50	int
waitForProvisionCorrelationToBeUpdatedThreadSleepingTime (advanced)	millis 中的间隔，在每次等待配置关联 ID 时处于睡眠状态。	100	long
errorHandlerLoggingLevel (logging)	允许为日志记录未捕获的异常配置默认的 errorHandler 日志记录级别。	WARN	LoggingLevel
errorHandlerLogStackTrace (logging)	允许由默认的 errorHandler 控制是否应记录堆栈追踪。	true	布尔值
密码 (security)	与 ConnectionFactory 搭配使用的密码。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
用户名 (security)	与 ConnectionFactory 搭配使用的用户名。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
转换 （事务）	指定是否使用转换模式	false	布尔值
lazyCreateTransactionManager (transaction)	如果为 true，如果选项 transacted=true 时没有注入的 transactionManager，则 Camel 将创建一个 JmsTransactionManager。	true	布尔值
transactionManager (transaction)	要使用的 Spring 事务管理器。		平台交易管理器
transactionName (transaction)	要使用的事务的名称。		字符串
transactionTimeout (transaction)	如果使用转换模式，事务的超时值（以秒为单位）。	-1	int

179.5. SPRING BOOT AUTO-CONFIGURATION

组件支持 172 选项，如下所列。

Name	描述	默认值	类型
camel.component.jms.accept-messages-while-stopping	指定消费者在停止时是否接受消息。如果您在运行时启动和停止 JMS 路由，同时队列中仍然有消息排队，您可以考虑启用此选项。如果此选项为 false，并且您停止 JMS 路由，则消息可能会被拒绝，并且 JMS 代理必须尝试恢复，并且再次可能被拒绝，最后该消息可能在 JMS 代理的死信队列中移动。为避免这种情况，建议启用这个选项。	false	布尔值
camel.component.jms.acknowledgement-mode	JMS 确认模式定义为整数。允许您将特定于供应商的扩展设置为确认模式。对于常规模式，最好改用 confirmmentModeName。		整数
camel.component.jms.acknowledgement-mode-name	JMS 确认名称，即 SESSION_TRANSACTED、CLIENT_ACKNOWLEDGE、AUTO_ACKNOWLEDGE, DUPS_OK_ACKNOWLEDGE	AUTO_确认	字符串
camel.component.jms.allow-additional-headers	此选项用于允许其他标头，这些标头可能具有根据 JMS 规格无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 包含带有字节数组或其他无效类型的值的标头名称。您可以指定用逗号分开的多个标头名称，并使用 作为后缀进行通配符匹配。		字符串
camel.component.jms.allow-null-body	是否允许发送没有正文的消息。如果此选项为 false，并且消息正文为 null，则抛出 JMSException。	true	布尔值
camel.component.jms.allow-reply-manager-quick-stop	在 request-reply 消息传递的回复管理器中使用的 DefaultMessageListenerContainer 是否允许 DefaultMessageListenerContainer.runningAllowed 标志快速停止，以防 JmsConfigurationBuildisAcceptMessagesWhileStopping 已启用，并且 org.apache.camel.CamelContext 当前已停止。默认情况下，在常规 JMS 用户中启用此快速停止功能，但要启用回复管理器，您必须启用此标志。	false	布尔值
camel.component.jms.always-copy-message	如果为 true，Camel 会在传递给制作者以进行发送时始终生成消息的 JMS 消息副本。在某些情况下需要复制消息，例如当设置了 replyToDestinationSelectorName 时（如果设置了 replyToDestinationSelectorName，Camel 会将 alwaysCopyMessage 选项设置为 true，如果设置了 replyToDestinationSelectorName ）	false	布尔值

Name	描述	默认值	类型
camel.component.jms.async-consumer	JmsConsumer 是否异步处理 Exchange。如果启用，JmsConsumer 可能会从 JMS 队列中选择下一个消息，而前面的消息异步处理（通过 Asynchronous Routing Engine）。这意味着消息可能没有完全严格按照顺序进行处理。如果禁用（作为默认），则在 JmsConsumer 将从 JMS 队列获取下一个消息之前完全处理 Exchange。请注意，如果启用了 transacted，则 asyncConsumer=true 不会异步运行，因为事务必须同步执行(Camel 3.0 可能支持 async 事务)。	false	布尔值
camel.component.jms.async-start-listener	在启动路由时，是否异步启动 JmsConsumer 消息监听程序。例如，如果 JmsConsumer 无法连接到远程 JMS 代理，则在重试和/或故障转移时可能会阻止。这将导致 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 以异步模式使用专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果无法建立连接，那么会记录异常在 WARN 级别，消费者将无法接收消息；然后您可以重启路由来重试。	false	布尔值
camel.component.jms.async-stop-listener	在停止路由时，是否异步停止 JmsConsumer 消息监听程序。	false	布尔值
camel.component.jms.auto-startup	指定消费者容器是否应自动启动。	true	布尔值
camel.component.jms.cache-level	根据底层 JMS 资源的 ID 设置缓存级别。如需了解更多详细信息，请参阅 cacheLevelName 选项。		整数
camel.component.jms.cache-level-name	按名称为底层 JMS 资源设置缓存级别。可能的值有：CACHE_AUTO、CACHE_CONNECTION、CACHE_CONSUMER、CACHE_NONE 和 CACHE_SESSION。默认设置为 CACHE_AUTO。如需更多信息，请参阅 Spring 文档和事务缓存级别。	CACHE_AUTO	字符串
camel.component.jms.client-id	设置要使用的 JMS 客户端 ID。请注意，如果指定，这个值必须是唯一的，且只能由单个 JMS 连接实例使用。它通常只需要 durable 主题订阅。如果使用 Apache ActiveMQ，您可能更喜欢使用虚拟主题。		字符串
camel.component.jms.concurrent-consumers	指定从 JMS 消耗时的默认并发消费者数（而不是通过 JMS 进行请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。当通过 JMS 进行请求/回复时，会使用选项 replyToConcurrentConsumers 来控制回复消息监听器上的并发用户数量。	1	整数

Name	描述	默认值	类型
camel.component.jms.configuration.accept-messages-while-stopping	指定消费者在停止时是否接受消息。如果您在运行时启动和停止 JMS 路由，同时队列中仍然有消息排队，您可以考虑启用此选项。如果此选项为 false，并且您停止 JMS 路由，则消息可能会被拒绝，并且 JMS 代理必须尝试恢复，并且再次可能被拒绝，最后该消息可能在 JMS 代理的死信队列中移动。为避免这种情况，建议启用这个选项。	false	布尔值
camel.component.jms.configuration.acknowledgement-mode	JMS 确认模式定义为整数。允许您将特定于供应商的扩展设置为确认模式。对于常规模式，最好改用 confirmmentModeName。		整数
camel.component.jms.configuration.acknowledgement-mode-name	JMS 确认名称，即 SESSION_TRANSACTED、CLIENT_ACKNOWLEDGE、AUTO_ACKNOWLEDGE, DUPS_OK_ACKNOWLEDGE	AUTO_确认	字符串
camel.component.jms.configuration.allow-additional-headers	此选项用于允许其他标头，这些标头可能具有根据 JMS 规格无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 包含带有字节数组或其他无效类型的值的标头名称。您可以指定用逗号分开的多个标头名称，并使用 作为后缀进行通配符匹配。		字符串
camel.component.jms.configuration.allow-null-body	是否允许发送没有正文的消息。如果此选项为 false，并且消息正文为 null，则抛出 JMSException。	true	布尔值
camel.component.jms.configuration.allow-reply-manager-quick-stop	在 request-reply 消息传递的回复管理器中使用的 DefaultMessageListenerContainer 是否允许 DefaultMessageListenerContainer SerialrunningAllowed （）标志在 JmsConfiguration SerialisAcceptMessagesWhileStopping （）被启用，并且 org.apache.camel.CamelContext 当前被停止。默认情况下，在常规 JMS 用户中启用此快速停止功能，但要启用回复管理器，您必须启用此标志。	false	布尔值
camel.component.jms.configuration.allow-serialized-headers	控制是否包含序列化标头。仅在 isTransferExchange （）为 true 时才适用。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。	false	布尔值

Name	描述	默认值	类型
camel.component.jms.configuration.always-copy-message	如果为 true，Camel 会在传递给制作者以进行发送时始终生成消息的 JMS 消息副本。在某些情况下需要复制消息，例如当设置了 replyToDestinationSelectorName 时（如果设置了 replyToDestinationSelectorName，Camel 会将 alwaysCopyMessage 选项设置为 true，如果设置了 replyToDestinationSelectorName ）	false	布尔值
camel.component.jms.configuration.async-consumer	JmsConsumer 是否异步处理 Exchange。如果启用，JmsConsumer 可能会从 JMS 队列中选择下一个消息，而前面的消息异步处理（通过 Asynchronous Routing Engine）。这意味着消息可能没有完全严格按照顺序进行处理。如果禁用（作为默认），则在 JmsConsumer 将从 JMS 队列获取下一个消息之前完全处理 Exchange。请注意，如果启用了 transacted，则 asyncConsumer=true 不会异步运行，因为事务必须同步执行(Camel 3.0 可能支持 async 事务)。	false	布尔值
camel.component.jms.configuration.async-start-listener	在启动路由时，是否异步启动 JmsConsumer 消息监听程序。例如，如果 JmsConsumer 无法连接到远程 JMS 代理，则在重试和/或故障转移时可能会阻止。这将导致 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 以异步模式使用专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果无法建立连接，那么会记录异常在 WARN 级别，消费者将无法接收消息；然后您可以重启路由来重试。	false	布尔值
camel.component.jms.configuration.async-stop-listener	在停止路由时，是否异步停止 JmsConsumer 消息监听程序。	false	布尔值
camel.component.jms.configuration.auto-startup	指定消费者容器是否应自动启动。	true	布尔值
camel.component.jms.configuration.cache-level	根据底层 JMS 资源的 ID 设置缓存级别。如需了解更多详细信息，请参阅 cacheLevelName 选项。		整数
camel.component.jms.configuration.cache-level-name	按名称为底层 JMS 资源设置缓存级别。可能的值有：CACHE_AUTO、CACHE_CONNECTION、CACHE_CONSUMER、CACHE_NONE 和 CACHE_SESSION。默认设置为 CACHE_AUTO。如需更多信息，请参阅 Spring 文档和事务缓存级别。	CACHE_AUTO	字符串

Name	描述	默认值	类型
camel.component.jms.configuration.client-id	设置要使用的 JMS 客户端 ID。请注意，如果指定，这个值必须是唯一的，且只能由单个 JMS 连接实例使用。它通常只需要 durable 主题订阅。如果使用 Apache ActiveMQ，您可能更喜欢使用虚拟主题。		字符串
camel.component.jms.configuration.concurrent-consumers	指定从 JMS 消耗时的默认并发消费者数（而不是通过 JMS 进行请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。当通过 JMS 进行请求/回复时，会使用选项 replyToConcurrentConsumers 来控制回复消息监听器上的并发用户数量。	1	整数
camel.component.jms.configuration.connection-factory	如果没有为 setTemplateConnectionFactory (ConnectionFactory)或 setListenerConnectionFactory (ConnectionFactory) 指定连接工厂，则设置默认的连接工厂。		ConnectionFactory
camel.component.jms.configuration.consumer-type	要使用的消费者类型，可以是： Simple、Default 或 Custom 之一。consumer 类型决定要使用的 Spring JMS 侦听器。default 将使用 org.springframework.jms.listener.DefaultMessageListenerContainer，Simple 将使用 org.springframework.jms.listener.SimpleMessageListenerContainer。当指定 Custom 时，messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 将决定要使用的 org.springframework.jms.listener.AbstractMessageListenerContainer。		ConsumerType
camel.component.jms.configuration.correlation-property	使用此 JMS 属性来关联 InOut Exchange pattern (request-reply)中的消息，而不是 JMSCorrelationID 属性。这样，您可以将消息与不使用 JMSCorrelationID JMS 属性关联消息的系统交换。如果使用的 JMSCorrelationID 不会被 Camel 使用或设置。如果没有在相同名称下的消息标头中提供，则会生成此处 named 属性的值。		字符串
camel.component.jms.configuration.default-task-executor-type	指定在 DefaultMessageListenerContainer 中使用的默认 TaskExecutor 类型，用于消费者端点和生成者端点的 ReplyTo consumer。可能的值： SimpleAsync （使用 Spring 的 SimpleAsyncTaskExecutor）或 ThreadPool （使用 Spring 的 ThreadPoolTaskExecutor 带有最佳值 - 缓存线程池）。如果没有设置，则默认为之前的行为，它使用缓存的线程池作为消费者端点，而 SimpleAsync 用于回复消费者。建议使用 ThreadPool，以动态增加和减少并发消费者在弹性配置中减少线程回收。		DefaultTaskExecutor Type

Name	描述	默认值	类型
camel.component.jms.configuration.delivery-mode	指定要使用的交付模式。Possibles 值是由 javax.jms.DeliveryMode 定义的值。NON_PERSISTENT = 1 和 PERSISTENT = 2.		整数
camel.component.jms.configuration.delivery-persistent	指定是否默认使用持久性交付。	true	布尔值
camel.component.jms.configuration.destination-resolver	可插拔 org.springframework.jms.support.destination.DestinationResolver，允许您使用您自己的解析器（例如，在 JNDI 注册表中查找实际目的地）。		DestinationResolver
camel.component.jms.configuration.disable-reply-to	指定 Camel 是否忽略消息中的 JMSReplyTo 标头。如果为 true，Camel 不会发送回复回 JMSReplyTo 标头中指定的目的地。如果您希望 Camel 从路由消耗且您不希望 Camel 自动发送回复消息，因为代码中的另一个组件处理回复消息，则可以使用这个选项。如果要使用 Camel 作为不同消息代理之间的代理，并希望将消息从一个系统路由到另一个系统，也可以使用这个选项。	false	布尔值
camel.component.jms.configuration.disable-time-to-live	使用这个选项强制禁用生存时间。例如，当您通过 JMS 进行请求/回复时，Camel 默认将使用 requestTimeout 值作为所发送消息的时间。问题是发送者和接收器系统必须同步其时钟，因此它们处于同步状态。这并非总是容易进行归档。因此，您可以使用 disableTimeToLive=true 设置在发送的消息上生存时间值。然后，消息不会在接收方系统中过期。如需了解更多详细信息，请参见关于生存时间 一节中的。	false	布尔值
camel.component.jms.configuration.durable-subscription-name	用于指定持久主题订阅的持久化订阅者名称。还必须配置 clientId 选项。		字符串
camel.component.jms.configuration.eager-loading-of-properties	在加载消息时，启用 JMS 属性和有效负载的强制加载，这通常效率低效，因为 JMS 属性可能并不是必需的，但有时可以在底层 JMS 提供程序和 JMS 属性中使用任何问题时捕获早期出现的问题	false	布尔值

Name	描述	默认值	类型
camel.component.jms.configuration.error-handler	指定在处理消息时引发任何未捕获的异常时调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置 errorHandler，这些例外将在 WARN 级别记录。您可以配置日志记录级别，并使用 errorHandlerLoggingLevel 和 errorHandlerLogStackTrace 选项记录堆栈 trace。这样可以更轻松地配置，而不是对自定义 errorHandler 进行编码。		ErrorHandler
camel.component.jms.configuration.error-handler-log-stack-trace	允许由默认的 errorHandler 控制是否应记录堆栈追踪。	true	布尔值
camel.component.jms.configuration.error-handler-logging-level	允许为日志记录未捕获的异常配置默认的 errorHandler 日志记录级别。		LoggingLevel
camel.component.jms.configuration.exception-listener	指定正在获得任何底层 JMS 异常通知的 JMS Exception Listener。		ExceptionListener
camel.component.jms.configuration.expose-listener-session	指定侦听器会话是否应该在消耗消息时公开。	false	布尔值
camel.component.jms.configuration.force-send-original-message	使用 mapJmsMessage=false Camel 时，如果您涉及路由期间的标头(get 或 set)，则会创建一个新的 JMS 消息来发送到新的 JMS 目的地。将此选项设置为 true，以强制 Camel 发送收到的原始 JMS 消息。	false	布尔值
camel.component.jms.configuration.format-date-headers-to-iso8601	设置日期标头是否应根据 ISO 8601 标准进行格式化。	false	布尔值
camel.component.jms.configuration.idle-consumer-limit	指定在任意给定时间允许闲置的用户数量的限制。	1	整数

Name	描述	默认值	类型
camel.component.jms.configuration.idle-task-execution-limit	指定闲置执行接收任务的限值，没有在其执行中收到任何消息。如果达到这个限制，任务将关闭并保持接收其他执行任务（如果是动态调度，请参阅 maxConcurrentConsumers 设置）。Spring 中提供了额外的文档。	1	整数
camel.component.jms.configuration.include-all-j-m-s-x-properties	从 JMS 映射到 Camel 消息时，是否要包含所有 JMSXxxx 属性。把它设置为 true 将包括 JMSXAppID 和 JMSXUserID 等属性。注： 如果您使用自定义 headerFilterStrategy，则不会应用这个选项。	false	布尔值
camel.component.jms.configuration.include-sent-j-m-s-message-i-d	仅在使用 InOnly （如触发和忘记）发送到 JMS 目的地时才适用。启用此选项将使用实际的 JMSMessageID 增强 Camel Exchange，当消息发送到 JMS 目的地时，由 JMS 客户端使用。	false	布尔值
camel.component.jms.configuration.jms-key-format-strategy	可插拔编码和解码 JMS 密钥策略，以便它们能够与 JMS 规范兼容。Camel 提供了两个开箱即用的实现：default 和 passthrough。默认策略可以安全地放入点和连字符(. 和 -)。passthrough 策略将密钥保留原样。可用于 JMS 代理，其不小心是 JMS 标头键是否包含非法字符。您可以提供自己的 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 # 表示法引用它。		JmsKeyFormatStrategy
camel.component.jms.configuration.jms-message-type	允许您强制使用特定的 javax.jms.Message 实施来发送 JMS 消息。可能的值有：Bytes, Map, Object, Stream, Text。默认情况下，Camel 将决定要从 In 正文类型使用哪个 JMS 消息类型。这个选项允许您指定它。		JmsMessageType
camel.component.jms.configuration.jms-operations	允许您使用您自己的 org.springframework.jms.core.JmsOperations 接口实施。Camel 使用 JmsTemplate 作为默认值。可用于测试目的，但不按 spring API 文档中所述使用。		JmsOperations
camel.component.jms.configuration.lazy-create-transaction-manager	如果为 true，如果选项 transacted=true 时没有注入的 transactionManager，则 Camel 将创建一个 JmsTransactionManager。	true	布尔值
camel.component.jms.configuration.listener-connection-factory	设置用于使用信息的连接工厂		ConnectionFactory

Name	描述	默认值	类型
camel.component.jms.configuration.map-jms-message	指定 Camel 是否应该自动将收到的 JMS 消息映射到合适的有效负载类型，如 javax.jms.TextMessage 到 String 等。	true	布尔值
camel.component.jms.configuration.max-concurrent-consumers	指定从 JMS 消耗时的最大并发消费者数（而不是通过 JMS 进行请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。当通过 JMS 进行请求/回复时，选项 replyToMaxConcurrentConsumers 用于控制回复消息监听器上的并发用户数量。		整数
camel.component.jms.configuration.max-messages-per-task	每个任务的消息数量。-1 代表没有限制。如果您将范围用于并发消费者（如 min max），则可以使用此选项将值设置为 eg 100，以控制消费者在需要较少的工作时缩小的速度。	-1	整数
camel.component.jms.configuration.message-converter	要使用自定义 Spring org.springframework.jms.support.converter.MessageConverter，以便您可以控制如何映射到 javax.jms.Message。		MessageConverter
camel.component.jms.configuration.message-created-strategy	要使用给定的 MessageCreatedStrategy，后者在 Camel 在 Camel 发送 JMS 消息时创建 javax.jms.Message 对象的新实例。		MessageCreatedStrategy
camel.component.jms.configuration.message-id-enabled	发送时，指定是否应添加消息 ID。这只是对 JMS Broker 的提示。如果 JMS 提供程序接受此提示，则这些消息必须将消息 ID 设为 null；如果提供程序忽略提示，则必须将消息 ID 设置为其普通唯一值	true	布尔值
camel.component.jms.configuration.message-listener-container-factory	用于确定用于使用消息的 org.springframework.jms.listener.AbstractMessageListenerContainer 的 MessageListenerContainer 的 registry ID。设置此选项将自动将 consumerType 设置为 Custom。		MessageListenerContainerFactory
camel.component.jms.configuration.message-timestamp-enabled	指定在发送消息时是否默认启用时间戳。这只是对 JMS Broker 的提示。如果 JMS 提供程序接受此提示，则这些消息必须设置时间戳为零；如果提供程序忽略 hint，则必须将时间戳设置为正常值	true	布尔值
camel.component.jms.configuration.metadata-jms-operations	设置用于减少 {@link JmsProviderMetadata} 详细信息的 {@link JmsOperations} 详情，如果 none 在需求上是 lazily 创建的。		JmsOperations

Name	描述	默认值	类型
camel.component.jms.configuration.password	与 ConnectionFactory 搭配使用的密码。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
camel.component.jms.configuration.preserve-message-qos	如果要使用消息上指定的 QoS 设置发送消息，而不是 JMS 端点上的 QoS 设置，则设置为 true。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供所有或只提供其中一些。如果没有提供，Camel 将回退到使用端点中的值。因此，在使用此选项时，标头会覆盖来自端点的值。相反，explicitQosEnabled 选项只会使用端点上设置的选项，而不使用消息标头中设置的值。	false	布尔值
camel.component.jms.configuration.priority	大于 1 的值在发送时指定消息优先级（其中 0 是最低优先级，9 是最高）。必须启用 explicitQosEnabled 选项，才能使此选项有任何效果。	4	整数
camel.component.jms.configuration.provider-metadata	允许显式配置供应商元数据。通常这不是强制要求，Camel 将自动检测底层供应商的供应商元数据。		JmsProviderMetadata
camel.component.jms.configuration.pub-sub-no-local	指定是否禁止发送由其自身连接发布的消息。	false	布尔值
camel.component.jms.configuration.receive-timeout	接收消息的超时时间（以毫秒为单位）。	1000	Long
camel.component.jms.configuration.recovery-interval	指定恢复尝试之间的间隔，例如当连接被刷新时，以毫秒为单位。默认值为 5000 ms，即 5 秒。	5000	Long
camel.component.jms.configuration.reply-to	提供一个显式 ReplyTo 目的地，它会覆盖任何 Message.getJMSReplyTo （）的任何传入值。		字符串
camel.component.jms.configuration.reply-to-cache-level-name	在通过 JMS 进行请求/回复时，按名称设置缓存级别。这个选项只适用于使用固定回复队列（而不是临时）。Camel 默认将使用：CACHE_CONSUMER 用于 exclusive 或 shared w/ replyToSelectorName。和 CACHE_SESSION 用于在没有 replyToSelectorName 的情况下共享。有些 JMS 代理（如 IBM WebSphere）可能需要设置 replyToCacheLevelName=CACHE_NONE 才能工作。注意：如果不使用临时队列，则不允许 CACHE_NONE，且您必须使用更高的值，如 CACHE_CONSUMER 或 CACHE_SESSION。		字符串

Name	描述	默认值	类型
camel.component.jms.configuration.reply-to-concurrent-consumers	指定通过 JMS 进行请求/回复时的默认并发消费者数量。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。	1	整数
camel.component.jms.configuration.reply-to-delivery-persistent	指定默认情况下是否使用持久发送回复。	true	布尔值
camel.component.jms.configuration.reply-to-destination-selector-name	使用固定名称设置 JMS Selector，以便在使用共享队列时过滤您自己的回复信息（也就是说，如果您不使用临时回复队列）。		字符串
camel.component.jms.configuration.reply-to-max-concurrent-consumers	指定通过 JMS 使用请求/回复时的最大并发消费者数。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。		整数
camel.component.jms.configuration.reply-to-on-timeout-max-concurrent-consumers	指定在通过 JMS 使用请求/回复时超时时继续路由的最大并发消费者数。	1	整数
camel.component.jms.configuration.reply-to-override	在 JMS 消息中提供一个显式 ReplyTo 目的地，它将覆盖 replyTo 的设置。如果您要将消息转发到远程 Queue，并从 ReplyTo 目的地接收回复消息，这非常有用。		字符串
camel.component.jms.configuration.reply-to-same-destination-allowed	是否允许 JMS 使用者向消费者使用的同一目的地发送回复消息。这可防止通过消耗并向其自身发送相同的消息，从而防止无限循环。	false	布尔值

Name	描述	默认值	类型
camel.component.jms.configuration.reply-to-type	允许明确指定在通过 JMS 进行请求/回复时，用于 replyTo 队列的策略。可能的值有： Temporary、Shared 或 Exclusive。默认情况下，Camel 将使用临时队列。但是，如果配置了 replyTo，则默认使用 Shared。此选项允许您使用专用队列而不是共享队列。如需了解更多详细信息，请参阅 Camel JMS 文档，特别是有关在集群环境中运行时所影响的备注，而共享回复队列的性能比其 alternatives Temporary 和 Exclusive 性能较低。		ReplyToType
camel.component.jms.configuration.request-timeout	使用 InOut Exchange Pattern 时等待回复的超时（以毫秒为单位）。默认值为 20 秒。您可以包含标头 CamelJmsRequestTimeout 来覆盖此端点配置的超时值，因此每个消息都有单独的超时值。另请参阅 requestTimeoutCheckerInterval 选项。	20000	Long
camel.component.jms.configuration.request-timeout-checker-interval	配置 Camel 在通过 JMS 进行请求/回复时应检查超时的频率。默认情况下，Camel 检查每秒一次。但是，如果发生超时时必须更快地响应，则可以降低这个间隔，以便更频繁地检查。超时由 option requestTimeout 决定。	1000	Long
camel.component.jms.configuration.selector	设置要使用的 JMS 选择器		字符串
camel.component.jms.configuration.stream-message-type-enabled	设置是否启用 StreamMessage 类型。文件、InputStream 等消息有效负载（如文件、InputStream 等）将通过 BytesMessage 或 StreamMessage 发送。此选项控制将使用哪种类型。默认情况下，使用 BytesMessage 来强制将整个消息有效负载读取到内存中。通过启用此选项，消息有效负载以块的形式读取内存，然后每个块写入 StreamMessage，直到没有更多数据。	false	布尔值
camel.component.jms.configuration.subscription-durable	设置是否使订阅持久。要使用的持久订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册持久化订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。仅在侦听主题(pub-sub domain)时具有意义，因此此方法也会切换 pubSubDomain 标志。	false	布尔值

Name	描述	默认值	类型
camel.component.jms.configuration.subscription-name	设置要创建的订阅的名称。当具有共享或持久化订阅的主题(pub-sub domain)时应用。订阅名称需要在此客户端的 JMS 客户端 id 中唯一。default 是指定消息监听程序的类名称。注意：除了共享订阅（需要 JMS 2.0）外，每个订阅只允许 1 个并发消费者（此消息监听程序容器的默认值）。		字符串
camel.component.jms.configuration.subscription-shared	设置是否使订阅共享。要使用的共享订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。请注意，共享订阅也可能是持久的，因此此标志也可以与 subscriptionDurable 结合使用。仅在侦听主题(pub-sub domain)时具有意义，因此此方法也会切换 pubSubDomain 标志。需要兼容 JMS 2.0 的消息代理。	false	布尔值
camel.component.jms.configuration.task-executor	允许您指定用于消耗消息的自定义任务 executor。		TaskExecutor
camel.component.jms.configuration.template-connection-factory	将连接工厂设置为用于通过 {@link JmsTemplate} 通过 {@link #createInOnlyTemplate (JmsEndpoint,boolean,String)} 发送信息		ConnectionFactory
camel.component.jms.configuration.test-connection-on-startup	指定是否在启动时测试连接。这样可确保当 Camel 启动所有 JMS 用户与 JMS 代理的有效连接时。如果无法授予连接，则 Camel 会在启动时抛出异常。这样可确保 Camel 不使用失败的连接启动。JMS producers 也经过测试。	false	布尔值
camel.component.jms.configuration.time-to-live	发送消息时，指定消息的生存时间（以毫秒为单位）。	-1	Long
camel.component.jms.configuration.transacted	指定是否使用转换模式	false	布尔值
camel.component.jms.configuration.transaction-manager	要使用的 Spring 事务管理器。		平台交易管理器

Name	描述	默认值	类型
camel.component.jms.configuration.transaction-name	要使用的事务的名称。		字符串
camel.component.jms.configuration.transaction-timeout	如果使用转换模式，事务的超时值（以秒为单位）。	-1	整数
camel.component.jms.configuration.transfer-exception	如果启用并且您使用 Request Reply messaging (InOut)和在消费者端的 Exchange 失败，则原因 Exception 将发回为 javax.jms.ObjectMessage 的响应。如果客户端是 Camel，则返回的 Exception 为 rerown。这允许您使用 Camel JMS 作为路由中的网桥，例如，使用持久性队列来启用强大的路由。请注意，如果您也启用了 transferExchange，这个选项会优先使用。需要被捕获的例外是串行的。在返回到生成者时，消费者侧的原始 Exception 可以嵌套在外部异常中，如 org.apache.camel.RuntimeCamelException。	false	布尔值
camel.component.jms.configuration.transfer-exchange	您可以通过线线传输交换，而不只是正文和标头。以下字段会被传输： 在 body, Out body, Fault body, In headers, Out headers, Fault 标头, Exchange properties, Exchange exception。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。您必须在生成者和消费者端启用这个选项，因此 Camel 知道有效负载是交换而不是常规有效负载。	false	布尔值
camel.component.jms.configuration.transfer-fault	如果启用且您使用 Request Reply messaging (InOut)，且交换失败，在消费者端使用 SOAP 错误（而不是异常），则 org.apache.camel.MessageAllisFault （）上的错误标志将作为 JMS 标头发送，其键为 JmsConstants#JMS_TRANSFER_FAULT。如果客户端是 Camel，则返回的 fault 标志将在 org.apache.camel.MessageAllsetFault (boolean)上设置。在使用支持错误（如 SOAP）（如 cxf 或 spring-ws）的 Camel 组件时，您可能需要启用此功能。	false	布尔值
camel.component.jms.configuration.use-message-id-as-correlation-id	指定 JMSMessageID 是否应该始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值

Name	描述	默认值	类型
camel.component.jms.configuration.username	与 ConnectionFactory 搭配使用的用户名。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
camel.component.jms.configuration.wait-for-provision-correlation-to-be-updated-counter	在通过 JMS 进行请求/回复时，以及启用选项 useMessageIDAsCorrelationID 时，等待 provisional correlation id 的次数。	50	整数
camel.component.jms.configuration.wait-for-provision-correlation-to-be-updated-thread-sleeping-time	millis 中的间隔，在每次等待配置关联 ID 时处于睡眠状态。	100	Long
camel.component.jms.connection-factory	要使用的连接工厂。必须在组件或端点上配置连接工厂。选项是 javax.jms.ConnectionFactory 类型。		字符串
camel.component.jms.correlation-property	使用此 JMS 属性来关联 InOut Exchange pattern (request-reply)中的消息，而不是 JMSCorrelationID 属性。这样，您可以将消息与不使用 JMSCorrelationID JMS 属性关联消息的系统交换。如果使用的 JMSCorrelationID 不会被 Camel 使用或设置。如果没有在相同名称下的消息标头中提供，则会生成此处 named 属性的值。		字符串
camel.component.jms.default-task-executor-type	指定在 DefaultMessageListenerContainer 中使用的默认 TaskExecutor 类型，用于消费者端点和生成者端点的 ReplyTo consumer。可能的值： SimpleAsync （使用 Spring 的 SimpleAsyncTaskExecutor）或 ThreadPool （使用 Spring 的 ThreadPoolTaskExecutor 带有最佳值 - 缓存线程池）。如果没有设置，则默认为之前的行为，它使用缓存的线程池作为消费者端点，而 SimpleAsync 用于回复消费者。建议使用 ThreadPool，以动态增加和减少并发消费者在弹性配置中减少线程回收。		DefaultTaskExecutor Type
camel.component.jms.delivery-mode	指定要使用的交付模式。可能的值有 Possibles 值，这些值由 javax.jms.DeliveryMode 定义。NON_PERSISTENT = 1 和 PERSISTENT = 2.		整数

Name	描述	默认值	类型
camel.component.jms.delivery-persistent	指定是否默认使用持久性交付。	true	布尔值
camel.component.jms.destination-resolver	可插拔 org.springframework.jms.support.destination.DestinationResolver，允许您使用您自己的解析器（例如，在 JNDI 注册表中查找实际目的地）。选项是一个 org.springframework.jms.support.destination.DestinationResolver 类型。		字符串
camel.component.jms.durable-subscription-name	用于指定持久主题订阅的持久化订阅者名称。还必须配置 clientId 选项。		字符串
camel.component.jms.eager-loading-of-properties	加载消息时，启用强制加载 JMS 属性，这通常效率低效，因为 JMS 属性可能并不需要，但有时可以提前捕获与底层 JMS 提供程序的任何问题，以及使用 JMS 属性	false	布尔值
camel.component.jms.enabled	启用 jms 组件	true	布尔值
camel.component.jms.error-handler	指定在处理消息时引发任何未捕获的异常时调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置 errorHandler，这些例外将在 WARN 级别记录。您可以配置日志记录级别，并使用 errorHandlerLoggingLevel 和 errorHandlerLogStackTrace 选项记录堆栈 trace。这样可以更轻松地配置，而不是对自定义 errorHandler 进行编码。选项是一个 org.springframework.util.ErrorHandler 类型。		字符串
camel.component.jms.error-handler-log-stack-trace	允许由默认的 errorHandler 控制是否应记录堆栈追踪。	true	布尔值
camel.component.jms.error-handler-logging-level	允许为日志记录未捕获的异常配置默认的 errorHandler 日志记录级别。		LoggingLevel
camel.component.jms.exception-listener	指定正在获得任何底层 JMS 异常通知的 JMS Exception Listener。选项是 javax.jms.ExceptionListener 类型。		字符串

Name	描述	默认值	类型
camel.component.jms.explicit-qos-enabled	设置在发送消息时应使用 deliveryMode、priority 或 timeToLive qualities。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用到当前端点。这与 preserveMessageQos 选项相反，它以消息粒度运行，从 Camel In 消息标头读取 QoS 属性。	false	布尔值
camel.component.jms.expose-listener-session	指定侦听器会话是否应该在消耗消息时公开。	false	布尔值
camel.component.jms.force-send-original-message	使用 mapJmsMessage=false Camel 时，如果您涉及路由期间的标头(get 或 set)，则会创建一个新的 JMS 消息来发送到新的 JMS 目的地。将此选项设置为 true，以强制 Camel 发送收到的原始 JMS 消息。	false	布尔值
camel.component.jms.format-date-headers-to-iso8601	设置日期标头是否应根据 ISO 8601 标准进行格式化。	false	布尔值
camel.component.jms.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		字符串
camel.component.jms.idle-consumer-limit	指定在任意给定时间允许闲置的用户数量的限制。	1	整数
camel.component.jms.idle-task-execution-limit	指定闲置执行接收任务的限值，没有在其执行中收到任何消息。如果达到这个限制，任务将关闭并保持接收其他执行任务（如果是动态调度，请参阅 maxConcurrentConsumers 设置）。Spring 中提供了额外的文档。	1	整数
camel.component.jms.include-all-j-m-s-x-properties	从 JMS 映射到 Camel 消息时，是否要包含所有 JMSXxxx 属性。把它设置为 true 将包括 JMSXAppID 和 JMSXUserID 等属性。注： 如果您使用自定义 headerFilterStrategy，则不会应用这个选项。	false	布尔值
camel.component.jms.include-sent-j-m-s-message-i-d	仅在使用 InOnly （如触发和忘记）发送到 JMS 目的地时才适用。启用此选项将使用实际的 JMSMessageID 增强 Camel Exchange，当消息发送到 JMS 目的地时，由 JMS 客户端使用。	false	布尔值

Name	描述	默认值	类型
camel.component.jms.jms-key-format-strategy	可插拔编码和解码 JMS 密钥策略，以便它们能够与 JMS 规范兼容。Camel 提供了两个开箱即用的实现：default 和 passthrough。默认策略可以安全地放入点和连字符(. 和 -)。passthrough 策略将密钥保留原样。可用于 JMS 代理，其不小心是 JMS 标头键是否包含非法字符。您可以提供自己的 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 # 表示法引用它。选项是 org.apache.camel.component.jms.JmsKeyFormatStrategy 类型。		字符串
camel.component.jms.jms-operations	允许您使用您自己的 org.springframework.jms.core.JmsOperations 接口实施。Camel 使用 JmsTemplate 作为默认值。可用于测试目的，但不按 spring API 文档中所述使用。选项是一个 org.springframework.jms.core.JmsOperations 类型。		字符串
camel.component.jms.lazy-create-transaction-manager	如果为 true，如果选项 transacted=true 时没有注入的 transactionManager，则 Camel 将创建一个 JmsTransactionManager。	true	布尔值
camel.component.jms.map-jms-message	指定 Camel 是否应该自动将收到的 JMS 消息映射到合适的有效负载类型，如 javax.jms.TextMessage 到 String 等。如需了解更多详细信息，请参阅映射如何工作。	true	布尔值
camel.component.jms.max-concurrent-consumers	指定从 JMS 消耗时的最大并发消费者数（而不是通过 JMS 进行请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。当通过 JMS 进行请求/回复时，选项 replyToMaxConcurrentConsumers 用于控制回复消息监听器上的并发用户数量。		整数
camel.component.jms.max-messages-per-task	每个任务的消息数量。-1 代表没有限制。如果您将范围用于并发消费者（如 min max），则可以使用此选项将值设置为 eg 100，以控制消费者在需要较少的工作时缩小的速度。	-1	整数
camel.component.jms.message-converter	要使用自定义 Spring org.springframework.jms.support.converter.MessageConverter，以便您可以控制如何映射到 javax.jms.Message。选项是 org.springframework.jms.support.converter.MessageConverter 类型。		字符串

Name	描述	默认值	类型
camel.component.jms.message-created-strategy	要使用给定的 MessageCreatedStrategy，后者在 Camel 在 Camel 发送 JMS 消息时创建 javax.jms.Message 对象的新实例。选项是 org.apache.camel.component.jms.MessageCreatedStrategy 类型。		字符串
camel.component.jms.message-id-enabled	发送时，指定是否应添加消息 ID。这只是对 JMS Broker 的提示。如果 JMS 提供程序接受此提示，则这些消息必须将消息 ID 设为 null；如果提供程序忽略提示，则必须将消息 ID 设置为其普通唯一值	true	布尔值
camel.component.jms.message-timestamp-enabled	指定在发送消息时是否默认启用时间戳。	true	布尔值
camel.component.jms.password	与 ConnectionFactory 搭配使用的密码。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
camel.component.jms.preserve-message-qos	如果要使用消息上指定的 QoS 设置发送消息，而不是 JMS 端点上的 QoS 设置，则设置为 true。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供所有或只提供其中一些。如果没有提供，Camel 将回退到使用端点中的值。因此，在使用此选项时，标头会覆盖来自端点的值。相反，explicitQosEnabled 选项只会使用端点上设置的选项，而不使用消息标头中设置的值。	false	布尔值
camel.component.jms.priority	大于 1 的值在发送时指定消息优先级（其中 0 是最低优先级，9 是最高）。必须启用 explicitQosEnabled 选项，才能使此选项有任何效果。	4	整数
camel.component.jms.pub-sub-no-local	指定是否禁止发送由其自身连接发布的消息。	false	布尔值
camel.component.jms.queue-browse-strategy	在浏览队列时使用自定义 QueueBrowseStrategy。选项是 org.apache.camel.component.jms.QueueBrowseStrategy 类型。		字符串
camel.component.jms.receive-timeout	接收消息的超时时间（以毫秒为单位）。	1000	Long
camel.component.jms.recovery-interval	指定恢复尝试之间的间隔，例如当连接被刷新时，以毫秒为单位。默认值为 5000 ms，即 5 秒。	5000	Long

Name	描述	默认值	类型
camel.component.jms.reply-on-timeout-to-max-concurrent-consumers	指定在通过 JMS 使用请求/回复时超时时继续路由的最大并发消费者数。	1	整数
camel.component.jms.reply-to-cache-level-name	在通过 JMS 进行请求/回复时，按名称设置缓存级别。这个选项只适用于使用固定回复队列（而不是临时）。Camel 默认将使用：CACHE_CONSUMER 用于 exclusive 或 shared w/ replyToSelectorName。和 CACHE_SESSION 用于在没有 replyToSelectorName 的情况下共享。有些 JMS 代理（如 IBM WebSphere）可能需要设置 replyToCacheLevelName=CACHE_NONE 才能工作。注意：如果不使用临时队列，则不允许 CACHE_NONE，且您必须使用更高的值，如 CACHE_CONSUMER 或 CACHE_SESSION。		字符串
camel.component.jms.reply-to-concurrent-consumers	指定通过 JMS 进行请求/回复时的默认并发消费者数量。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。	1	整数
camel.component.jms.reply-to-max-concurrent-consumers	指定通过 JMS 使用请求/回复时的最大并发消费者数。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/关闭。		整数
camel.component.jms.reply-to-type	允许明确指定在通过 JMS 进行请求/回复时，用于 replyTo 队列的策略。可能的值有： Temporary、Shared 或 Exclusive。默认情况下，Camel 将使用临时队列。但是，如果配置了 replyTo，则默认使用 Shared。此选项允许您使用专用队列而不是共享队列。如需了解更多详细信息，请参阅 Camel JMS 文档，特别是有关在集群环境中运行时所影响的备注，而共享回复队列的性能比其 alternatives Temporary 和 Exclusive 性能较低。		ReplyToType
camel.component.jms.request-timeout	使用 InOut Exchange Pattern 时等待回复的超时（以毫秒为单位）。默认值为 20 秒。您可以包含标头 CamelJmsRequestTimeout 来覆盖此端点配置的超时值，因此每个消息都有单独的超时值。另请参阅 requestTimeoutCheckerInterval 选项。	20000	Long
camel.component.jms.request-timeout-checker-interval	配置 Camel 在通过 JMS 进行请求/回复时应检查超时的频率。默认情况下，Camel 检查每秒一次。但是，如果发生超时时必须更快地响应，则可以降低这个间隔，以便更频繁地检查。超时由 option requestTimeout 决定。	1000	Long

Name	描述	默认值	类型
camel.component.jms.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.jms.stream-message-type-enabled	设置是否启用 StreamMessage 类型。文件、InputStream 等消息有效负载（如文件、InputStream 等）将通过 BytesMessage 或 StreamMessage 发送。此选项控制将使用哪种类型。默认情况下，使用 BytesMessage 来强制将整个消息有效负载读取到内存中。通过启用此选项，消息有效负载以块的形式读取内存，然后每个块写入 StreamMessage，直到没有更多数据。	false	布尔值
camel.component.jms.subscription-durable	设置是否使订阅持久。要使用的持久订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册持久化订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。仅在侦听主题(pub-sub domain)时具有意义，因此此方法也会切换 pubSubDomain 标志。	false	布尔值
camel.component.jms.subscription-name	设置要创建的订阅的名称。当具有共享或持久化订阅的主题(pub-sub domain)时应用。订阅名称需要在此客户端的 JMS 客户端 id 中唯一。default 是指定消息监听程序的类名称。注意：除了共享订阅（需要 JMS 2.0）外，每个订阅只允许 1 个并发消费者（此消息监听程序容器的默认值）。		字符串
camel.component.jms.subscription-shared	设置是否使订阅共享。要使用的共享订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。请注意，共享订阅也可能是持久的，因此此标志也可以与 subscriptionDurable 结合使用。仅在侦听主题(pub-sub domain)时具有意义，因此此方法也会切换 pubSubDomain 标志。需要兼容 JMS 2.0 的消息代理。	false	布尔值
camel.component.jms.task-executor	允许您指定用于消耗消息的自定义任务 executor。选项是一个 org.springframework.core.task.TaskExecutor 类型。		字符串

Name	描述	默认值	类型
camel.component.jms.test-connection-on-startup	指定是否在启动时测试连接。这样可确保当 Camel 启动所有 JMS 用户与 JMS 代理的有效连接时。如果无法授予连接，则 Camel 会在启动时抛出异常。这样可确保 Camel 不使用失败的连接启动。JMS producers 也经过测试。	false	布尔值
camel.component.jms.time-to-live	发送消息时，指定消息的生存时间（以毫秒为单位）。	-1	Long
camel.component.jms.transacted	指定是否使用转换模式	false	布尔值
camel.component.jms.transaction-manager	要使用的 Spring 事务管理器。选项是一个 org.springframework.transaction.PlatformTransactionManager 类型。		字符串
camel.component.jms.transaction-name	要使用的事务的名称。		字符串
camel.component.jms.transaction-timeout	如果使用转换模式，事务的超时值（以秒为单位）。	-1	整数
camel.component.jms.transfer-exception	如果启用并且您使用 Request Reply messaging (InOut)和在消费者端的 Exchange 失败，则原因 Exception 将发回为 javax.jms.ObjectMessage 的响应。如果客户端是 Camel，则返回的 Exception 为 rerown。这允许您使用 Camel JMS 作为路由中的网桥，例如，使用持久性队列来启用强大的路由。请注意，如果您也启用了 transferExchange，这个选项会优先使用。需要被捕获的例外是串行的。在返回到生成者时，消费者侧的原始 Exception 可以嵌套在外部异常中，如 org.apache.camel.RuntimeCamelException。	false	布尔值
camel.component.jms.transfer-exchange	您可以通过线线传输交换，而不只是正文和标头。以下字段会被传输： 在 body, Out body, Fault body, In headers, Out headers, Fault 标头, Exchange properties, Exchange exception。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。您必须在生成者和消费者端启用这个选项，因此 Camel 知道有效负载是交换而不是常规有效负载。	false	布尔值

Name	描述	默认值	类型
camel.component.jms.transfer-fault	如果启用且您使用 Request Reply messaging (InOut)，且交换失败，在消费者端使用 SOAP 错误（而不是异常），则 org.apache.camel.MessageAllisFault （）上的错误标志将作为 JMS 标头发送，其键为 JmsConstants#JMS_TRANSFER_FAULT。如果客户端是 Camel，则返回的 fault 标志将在 org.apache.camel.MessageAllsetFault (boolean)上设置。在使用支持错误（如 SOAP）（如 cxf 或 spring-ws）的 Camel 组件时，您可能需要启用此功能。	false	布尔值
camel.component.jms.use-message-i-d-as-correlation-i-d	指定 JMSMessageID 是否应该始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值
camel.component.jms.username	与 ConnectionFactory 搭配使用的用户名。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
camel.component.jms.wait-for-provision-correlation-to-be-updated-counter	在通过 JMS 进行请求/回复时，以及启用选项 useMessageIDAsCorrelationID 时，等待 provisional correlation id 的次数。	50	整数
camel.component.jms.wait-for-provision-correlation-to-be-updated-thread-sleeping-time	millis 中的间隔，在每次等待配置关联 ID 时处于睡眠状态。	100	Long
camel.component.jms.configuration.transacted-in-out			布尔值

179.6. SAMPLES

JMS 也用于其他组件。但我们提供了几个示例来开始。

179.6.1. 从 JMS 接收

在以下示例中，我们配置一个接收 JMS 消息的路由，并将消息路由到 POJO：

```
from("jms:queue:foo").
  to("bean:myBusinessLogic");
```

当然，您可以使用任何 EIP 模式，以便路由可以基于上下文。例如，以下为大型开支过滤订单主题：

```
from("jms:topic:OrdersTopic").
  filter().method("myBean", "isGoldCustomer").
  to("jms:queue:BigSpendersQueue");
```

179.6.2. 发送到 JMS

在以下示例中，我们轮询文件文件夹，并将文件内容发送到 JMS 主题。我们希望文件的内容作为 TextMessage 而不是 BytesMessage，因此我们需要将正文转换为 String ：

```
from("file://orders").
  convertBodyTo(String.class).
  to("jms:topic:OrdersTopic");
```

179.6.3. 使用注解

Camel 还具有注解，因此您可以使用 POJO 消耗和 POJO Producing。

179.6.4. Spring DSL 示例

前面的示例使用了 Java DSL。Camel 还支持 Spring XML DSL。以下是使用 Spring DSL 的大型开销示例：

```
<route>
  <from uri="jms:topic:OrdersTopic"/>
  <filter>
    <method bean="myBean" method="isGoldCustomer"/>
    <to uri="jms:queue:BigSpendersQueue"/>
  </filter>
</route>
```

179.6.5. 其他示例

JMS 出现在其他组件和 EIP 模式的许多示例中，以及 Camel 文档。因此请随时浏览文档。

179.6.6. 使用 JMS 作为死信队列存储交换

通常，当将 JMS 用作传输时，它只传输正文和标头作为载荷。如果要将 JMS 与死信 频道 搭配使用，请将 JMS 队列用作 Dead Letter Queue，通常原因的 Exception 不在 JMS 消息中。但是，您可以对 JMS 死信队列使用 transferExchange 选项，以指示 Camel 将整个 Exchange 存储在队列中，作为 javax.jms.ObjectMessage，其中包含 org.apache.camel.impl.DefaultExchangeHolder。这样，您可以从死信队列使用，并使用密钥 Exchange.EXCEPTION_CAUGHT 从 Exchange 属性检索导致的异常。以下演示说明了这一点：

```
// setup error handler to use JMS as queue and store the entire Exchange
errorHandler(deadLetterChannel("jms:queue:dead?transferExchange=true"));
```

然后，您可以从 JMS 队列中消耗并分析问题：

```
from("jms:queue:dead").to("bean:myErrorAnalyzer");

// and in our bean
String body = exchange.getIn().getBody();
Exception cause = exchange.getProperty(Exchange.EXCEPTION_CAUGHT, Exception.class);
// the cause message is
String problem = cause.getMessage();
```

179.6.7. 仅将 JMS 用作死信频道存储错误

您可以使用 JMS 来存储原因错误消息或存储自定义正文，供您自行初始化。以下示例使用 Message Translator EIP 在进入 JMS 死信队列前对失败的交换进行转换：

```
// we sent it to a seda dead queue first
errorHandler(deadLetterChannel("seda:dead"));

// and on the seda dead queue we can do the custom transformation before its sent to the
JMS queue
from("seda:dead").transform(exceptionMessage()).to("jms:queue:dead");
```

在这里，我们在转换中仅存储原始原因错误消息。但是，您可以使用任何 Expression 来发送您喜欢的任何对象。例如，您可以在 Bean 上调用方法或使用自定义处理器。

179.6.8. 使用 JMS 指定消费者优先级

有些代理（特别是 Artemis）可让您为要发送的消息指定消费者优先级。在以下示例中，我们使用 artemisConsumerPriority 属性根据优先级转换消息。

```
<camelContext id="jms-basic-consumer-priority" xmlns="http://camel.apache.org/schema/spring"
```

```
autoStartup="true">
  <endpoint id="highPriorityQueue" uri="jms:queue:inputQueue?artemisConsumerPriority=1000"/>
  <endpoint id="mediumPriorityQueue" uri="jms:queue:inputQueue?artemisConsumerPriority=500"/>
  <endpoint id="lowPriorityQueue" uri="jms:queue:inputQueue?artemisConsumerPriority=1"/>

  <endpoint id="resultQueue" uri="jms:queue:outputQueue"/>

  <route>
   <from uri="ref:highPriorityQueue"/>
   <transform>
    <simple>High</simple>
   </transform>
   <to uri="ref:resultQueue"/>
  </route>
  <route>
   <from uri="ref:mediumPriorityQueue"/>
   <transform>
    <simple>Medium</simple>
   </transform>
   <to uri="ref:resultQueue"/>
  </route>
  <route>
   <from uri="ref:lowPriorityQueue"/>
   <transform>
    <simple>Low</simple>
   </transform>
   <to uri="ref:resultQueue"/>
  </route>
 </camelContext>
```

179.7. JMS 和 CAMEL 之间的消息映射

Camel 会自动映射 javax.jms.Message 和 org.apache.camel.Message 之间的消息。

在发送 JMS 消息时，Camel 会将消息正文转换为以下 JMS 消息类型：

正文类型	JMS 消息	注释
字符串	javax.jms.TextMessage	
org.w3c.dom.Node	javax.jms.TextMessage	DOM 将转换为 String。

正文类型	JMS 消息	注释
Map	javax.jms.MapMessage	
java.io.Serializable	javax.jms.ObjectMessage	
byte[]	javax.jms.BytesMessage	
java.io.File	javax.jms.BytesMessage	
java.io.Reader	javax.jms.BytesMessage	
java.io.InputStream	javax.jms.BytesMessage	
java.nio.ByteBuffer	javax.jms.BytesMessage	

在收到 JMS 消息时，Camel 会将 JMS 消息转换为以下正文类型：

JMS 消息	正文类型
javax.jms.TextMessage	字符串
javax.jms.BytesMessage	byte[]

JMS 消息	正文类型
javax.jms.MapMessage	Map<String, Object>
javax.jms.ObjectMessage	对象

179.7.1. 禁用 JMS 消息自动映射

您可以使用 mapJmsMessage 选项禁用上面的自动映射。如果禁用，Camel 将不会尝试映射收到的 JMS 消息，而是直接将其用作有效负载。这样，您可以避免映射的开销，并让 Camel 仅通过 JMS 消息。例如，它甚至允许您路由带有使用了没有包括在 classpath 中的类的 javax.jms.ObjectMessage JMS 消息。

179.7.2. 使用自定义 MessageConverter

您可以使用 messageConverter 选项在 Spring org.springframework.jms.support.converter.MessageConverter 类中自行进行映射。

例如，在下面的路由中，在向 JMS 顺序队列发送消息时，我们使用自定义消息转换器：

```
from("file://inbox/order").to("jms:queue:order?messageConverter=#myMessageConverter");
```

从 JMS 目的地消耗时，也可以使用自定义消息转换器。

179.7.3. 控制所选的映射策略

您可以在端点 URL 上使用 jmsMessageType 选项，以强制对所有消息进行特定的消息类型。

在以下路由中，我们轮询文件夹中的文件，并将其作为 javax.jms.TextMessage 发送，因为我们强制 JMS producer 端点使用文本消息：

```
from("file://inbox/order").to("jms:queue:order?jmsMessageType=Text");
```

您还可以通过使用键 CamelJmsMessageType 设置标头来指定要用于每个消息的消息类型。例如：

```
from("file://inbox/order").setHeader("CamelJmsMessageType",
JmsMessageType.Text).to("jms:queue:order");
```

可能的值在 enum 类 org.apache.camel.jms.JmsMessageType 中定义。

179.8. 发送时消息格式

通过 JMS 线发送的交换必须符合 JMS 消息规范。

对于 exchange.in.header，以下规则适用于标头 键 ：

- 从 JMS 或 JMSX 开始的密钥被保留。
- exchange.in.headers 键必须是 literals，且所有是有效的 Java 标识符（不要在键名称中使用点）。
- 在消耗 JMS 消息时，Camel 替换点 & hyphens，在 Camel 使用消息时由 DOT 替换点和反向替换。
  - 在 Camel 使用消息时被 HYPHEN 替换为反向替换。
- 另请参阅 jmsKeyFormatStrategy 选项，它允许使用您自己的自定义策略格式化密钥。

对于 exchange.in.header，以下规则适用于 标头值 ：

- 值必须是原语或其计数器对象（如 Integer、Long、Character）。type, String,CharSequence,Date,BigDecimal 和 BigInteger 都转换为它们的 toString （） 表示。所有其他类型都将被丢弃。

如果 Camel 丢弃给定标头值，则 Camel 将在 DEBUG 级别上记录类别 org.apache.camel.component.jms.JmsBinding。例如：

```
2008-07-09 06:43:04,046 [main           ] DEBUG JmsBinding
  - Ignoring non primitive header: order of class:
org.apache.camel.component.jms.issues.DummyOrder with value: DummyOrder{orderId=333,
itemId=4444, quantity=2}
```

179.9. 接收时的消息格式

Camel 在接收消息时为 Exchange 添加以下属性：

属性	类型	描述
org.apache.camel.jms.replyDestination	**javax.jms.Destination**	回复目的地。

Camel 在收到 JMS 消息消息时，将以下 JMS 属性添加到 In 消息标头：

标头	类型	描述
JMSCorrelationID	**字符串**	JMS 关联 ID。
JMSDeliveryMode	**int**	JMS 交付模式。
JMSDestination	**javax.jms.Destination**	JMS 目的地。
JMSExpiration	**long**	JMS 过期.
JMSMessageID	**字符串**	JMS 唯一消息 ID。
JMSPriority	**int**	JMS 优先级（优先级最低，9 为最高）。
JMSRedelivered	**布尔值**	是 JMS 消息 redelivered。

标头	类型	描述
JMSReplyTo	**javax.jms.Destination**	JMS 回复目的地。
JMSTimestamp	**long**	JMS 时间戳。
JMSType	**字符串**	JMS 类型。
JMSXGroupID	**字符串**	JMS 组 ID。

由于以上所有信息都是标准 JMS，您可以检查 JMS 文档以了解 更多详细信息。

179.10. 关于使用 CAMEL 发送和接收消息和 JMSREPLYTO

JMS 组件比较复杂，您必须仔细注意它在一些情况下的工作方式。这是要查找的一些区域的简短总结。

当 Camel 使用其 JMSProducer 发送消息时，它会检查以下条件：

- *消息交换模式，*
- *在端点或消息标头中设置了 JMSReplyTo*
- *是否在 JMS 端点上设置了以下任何选项：disableReplyTo,preserveMessageQos,explicitQosEnabled。*

所有这些都很复杂，以了解和配置来支持您的用例。

179.10.1. JmsProducer

JmsProducer 的行为如下，具体取决于配置：

交换模式	其他选项	描述
InOut	-	Camel 将期望一个回复，设置一个临时 **JMSReplyTo**，并在发送消息后，它将开始侦听临时队列上的回复消息。
InOut	**JMSReplyTo** is set	Camel 将期望回复，在发送邮件后，它将开始侦听指定 **JMSReplyTo** 队列上的回复消息。
InOnly	-	Camel 将发送消息 **且不会** 期望回复。
InOnly	**JMSReplyTo** is set	默认情况下，Camel 会丢弃 **JMSReplyTo** 目的地，并在发送消息前清除 **JMSReplyTo** 标头。然后 Camel 会发送消息，**且不会** 期望回复。Camel 会在 **WARN** 级别登录时记录此项（从 **Camel 2.6** 开始更改为 **DEBUG** 级别）。您可以使用 **preserveMessageQuo=true** 来指示 Camel 保持 **JMSReplyTo**。在所有情况下，**JmsProducer 不会** 期望任何回复，因此在发送消息后继续。

179.10.2. JmsConsumer

JmsConsumer 的行为如下，具体取决于配置：

交换模式	其他选项	描述
InOut	-	Camel 会将回复发送回 **JMSReplyTo** 队列。
InOnly	-	Camel 将不会发送回复，因为模式是 *InOnly*。
-	**disableReplyTo=true**	这个选项阻止回复。

因此，请注意交换上设置的消息交换模式。

如果您在路由中间向 JMS 目的地发送消息，您可以指定要使用的交换模式，请参阅 Request Reply。如果您要向 JMS 主题发送 InOnly 消息，这很有用：

```
from("activemq:queue:in")
```

```
    .to("bean:validateOrder")
    .to(ExchangePattern.InOnly, "activemq:topic:order")
    .to("bean:handleOrder");
```

179.11. 重复使用端点并发送到在运行时计算的不同目的地

如果您需要发送消息到许多不同的 JMS 目的地，则重复利用 JMS 端点并在消息标头中指定实际目的地是明智的。这允许 Camel 重复使用同一端点，但发送到不同的目的地。这大大减少了在内存和线程资源上创建和经济的端点数量。

您可以在以下标头中指定目的地：

标头	类型	描述
CamelJmsDestination	javax.jms.Destination	目标对象。
CamelJmsDestinationName	字符串	目标名称。

例如，以下路由演示了如何在运行时计算目的地，并使用它来覆盖 JMS URL 中显示的目的地：

```
from("file://inbox")
  .to("bean:computeDestination")
  .to("activemq:queue:dummy");
```

队列名称 dummy 仅是一个占位符。它必须作为 JMS 端点 URL 的一部分提供，但本例中将被忽略。

在 computeDestination bean 中，通过设置 CamelJmsDestinationName 标头来指定实际目的地，如下所示：

```
public void setJmsHeader(Exchange exchange) {
   String id = ....
   exchange.getIn().setHeader("CamelJmsDestinationName", "order:" + id");
}
```

然后，Camel 将读取此标头并将其用作目的地，而不是端点上配置的。因此，在本示例中，Camel 会将消息发送到 activemq:queue:order:2，假定 id 值是 2。

如果设置了 CamelJmsDestination 和 CamelJmsDestinationName 标头，则 CamelJmsDestination 将具有优先权。请记住，JMS 生成者会从交换中删除 CamelJmsDestination and CamelJmsDestinationName 标头，且不会将它们传播到所创建的 JMS 消息，以避免路由中的意外循环（当消息将转发到另一个 JMS 端点时）。

179.12. 配置不同的 JMS 提供程序

您可以在 Spring XML 中配置 JMS 供应商，如下所示：

基本上，您可以根据需要配置多个 JMS 组件实例，您需要使用 id 属性 为它们指定唯一的名称。前面的示例配置一个 activemq 组件。您可以执行相同的操作来配置 MQ 系列、TibCo、BEA、Snic 等。

具有命名的 JMS 组件后，您可以使用 URI 引用该组件中的端点。例如，对于组件名称 activemq，您可以使用 URI 格式引用目的地 ：[queue:|topic:]destinationName.您可以将相同的方法用于所有其他 JMS 提供程序。

这由 SpringCamelContext lazily 从您用于 Endpoint URI 的 spring 上下文获取组件，并让组件解析端点 URI。

179.12.1. 使用 JNDI 查找 ConnectionFactory

如果您使用的是 J2EE 容器，您可能需要查找 JNDI 来查找 JMS ConnectionFactory，而不必在 Spring 中使用常见的 < bean> 机制。您可以使用 Spring 的 factory bean 或新的 Spring XML 命名空间进行此操作。例如：

```
<bean id="weblogic" class="org.apache.camel.component.jms.JmsComponent">
  <property name="connectionFactory" ref="myConnectionFactory"/>
</bean>

<jee:jndi-lookup id="myConnectionFactory" jndi-name="jms/connectionFactory"/>
```

有关 JNDI 查找的详情，请参阅 Spring 参考文档中的 jee 模式。

179.13. 并发消耗

与 JMS 的常见要求是在多个线程中同时消耗消息，以便应用更快响应。您可以设置 concurrentConsumers 选项来指定为 JMS 端点提供服务的线程数量，如下所示：

```
from("jms:SomeQueue?concurrentConsumers=20").
  bean(MyClass.class);
```

您可以使用以下方法之一配置这个选项：

- *在 JmsComponent 上，*
- *在端点 URI 或.*
- *通过在 JmsEndpoint 上直接调用 setConcurrentConsumers （）。*

179.13.1. 使用 async 消费者并发消耗

请注意，当当前消息被完全处理时，每个并发消费者只会从 JMS 代理获取下一个可用消息。您可以设置 asyncConsumer=true 选项，以便使用者从 JMS 队列获取下一个消息，而之前的消息则异步处理（通过异步路由引擎）。请参阅有关 asyncConsumer 选项页面顶部的表中的更多详细信息。

```
from("jms:SomeQueue?concurrentConsumers=20&asyncConsumer=true").
  bean(MyClass.class);
```

179.14. 通过 JMS 的 REQUEST-REPLY

Camel 支持通过 JMS 进行请求。当您向 JMS 队列发送消息时，交换的 MEP 应该为 InOut。

Camel 提供了多个选项，用于通过 JMS 配置请求/回复，影响性能和集群环境。下表总结了选项。

选项	性能	集群	描述
临时	速度快	是	临时队列用作回复队列，并由 Camel 自动创建。要使用它，**不要指定** 一个 replyTo 队列名称。您可选择配置 **replyToType=Temporary**，以代替该临时队列正在使用。

选项	性能	集群	描述
共享	slow	是	共享持久队列用作回复队列。必须事先创建队列，但有些代理可以立即创建它们，如 Apache ActiveMQ。要使用它，您必须指定 replyTo 队列名称。另外，您还可以配置 **replyToType=Shared** 来代替该共享队列正在使用。可以在集群环境中使用共享队列，同时运行此 Camel 应用程序的多个节点。所有都使用相同的共享回复队列。这是因为 JMS 消息选择器用于关联预期的回复消息；这影响了性能。JMS 消息选择器速度较慢，因此不像 **临时** 或排除队列一样快速。**请参阅下面如何调整它以提高性能。**
exclusive	速度快	否(*是)	独占持久队列用作回复队列。必须事先创建队列，但有些代理可以立即创建它们，如 Apache ActiveMQ。要使用它，您必须指定 replyTo 队列名称。如果配置了 **replyTo** 队列名称，并且 **您必须配置 replyToType=Exclusive**，以指示 Camel 使用专用队列队列。**使用专用回复队列时，JMS 消息选择器 不会被使用，因此其他应用不得使用此队列。在集群环境中不能同时使用具有同时运行此 Camel 应用程序的多个节点；因为如果回复队列回到发送请求消息的相同节点，我们就没有控制该队列；因此，共享队列使用 JMS 消息选择器来确保这一点。虽然 如果您将每个 Exclusive 回复队列配置为每个节点的唯一名称，那么您可以在集群环境中运行。然后，回复消息将发回到给定节点的该队列，等待回复消息。**
concurrentConsumers	速度快	是	Camel 2.10. 3 ：允许使用中的并发消息监听程序同时处理回复信息。您可以使用 **concurrentConsumers** 和 **maxConcurrentConsumers** 选项指定范围。**注意： 使用共享** 回复队列可能无法与并发监听器一起工作，因此请谨慎使用这个选项。
maxConcurrentConsumers	速度快	是	Camel 2.10. 3 ：允许使用中的并发消息监听程序同时处理回复信息。您可以使用 **concurrentConsumers** 和 **maxConcurrentConsumers** 选项指定范围。**注意： 使用共享** 回复队列可能无法与并发监听器一起工作，因此请谨慎使用这个选项。

JmsProducer 检测到 InOut，并提供带有要使用的回复目的地的 JMSReplyTo 标头。默认情况下，Camel 使用临时队列，但您可以使用端点上的 replyTo 选项指定固定的回复队列（请参阅以下有关固定回复队列）。

Camel 将自动设置侦听回复队列的消费者，因此 您不应该 执行任何操作。这个消费者是一个 Spring DefaultMessageListenerContainer，它侦听回复。但是，它被固定到 1 个并发消费者。

这意味着，将按顺序处理回复，因为只有 1 个线程来处理回复。如果您希望更快地处理回复，则需要使用并发。但不使用 concurrentConsumer 选项。我们应使用 Camel DSL 中的 线程，如下所示：

如果使用 Camel 2.10.3 或更高版本，则使用 concurrentConsumers 选项而不是使用线程。请参阅以下。

```
from(xxx)
.inOut().to("activemq:queue:foo")
.threads(5)
.to(yyy)
.to(zzz);
```

在此路由中，我们指示 Camel 使用具有 5 个线程的线程池异步路由回复。

现在，从 Camel 2.10.3 开始，您可以使用 concurrentConsumers 和 maxConcurrentConsumers 选项将监听程序配置为使用并发线程。这可让您在 Camel 中更轻松地配置它，如下所示：

```
from(xxx)
.inOut().to("activemq:queue:foo?concurrentConsumers=5")
.to(yyy)
.to(zzz);
```

179.14.1. 通过 JMS 进行请求回复并使用共享的固定回复队列

如果您在执行 Request Reply over JMS 时使用固定的回复队列，如以下示例所示，请注意。

```
from(xxx)
.inOut().to("activemq:queue:foo?replyTo=bar")
.to(yyy)
```

本例中使用了名为"bar"的固定回复队列。默认情况下，Camel 假定队列在使用固定回复队列时共享，因此它使用 JMSSelector 仅选择预期的回复消息（如 JMSCorrelationID）。有关专用固定回复队列，请参见下一小节。这意味着它没有作为临时队列的速度。您可以使用 receiveTimeout 选项加快 Camel 将拉取回复消息的频率。默认情况下，其 1000 millis。因此，您可以更快地将其设置为 250 millis 以每秒拉取 4 次，如下所示：

```
from(xxx)
.inOut().to("activemq:queue:foo?replyTo=bar&receiveTimeout=250")
.to(yyy)
```

请注意，这会导致 Camel 更频繁地向消息代理发送拉取请求，因此需要更多网络流量。通常，建议您尽可能使用临时队列。

179.14.2. 通过 JMS 的 request-reply 并使用专用固定回复队列

从 Camel 2.9 开始可用

在上例中，Camel 会预期名为"bar"的固定回复队列已共享，因此它使用 JMSSelector 只消耗期望的回复消息。但是，这样做的一个缺陷是因为 JMS selectos 较慢。此外，回复队列上的使用者使用新的 JMS 选择器 ID 更新速度较慢。实际上，它只在 receiveTimeout 选项超时时才更新，默认为 1 秒。因此，在回复信息中，会花费大约 1 秒的时间被检测。另一方面，如果固定回复队列专用于 Camel 回复消费者，我们可以避免使用 JMS 选择器，因此更高性能。实际上，使用临时队列就快。因此，在 Camel 2.9 中，我们引入了 ReplyToType 选项，您可以将它配置为 Exclusive
，以告知 Camel 回复队列是独占的，如下例所示：

```
from(xxx)
.inOut().to("activemq:queue:foo?replyTo=bar&replyToType=Exclusive")
.to(yyy)
```

请注意，队列必须专用于每个端点和每个端点。因此，如果您有两个路由，它们各自需要一个唯一的回复队列，如下例所示：

```
from(xxx)
.inOut().to("activemq:queue:foo?replyTo=bar&replyToType=Exclusive")
.to(yyy)

from(aaa)
.inOut().to("activemq:queue:order?replyTo=order.reply&replyToType=Exclusive")
.to(bbb)
```

如果您在集群环境中运行，也是如此。然后，集群中的每个节点都必须使用唯一的回复队列名称。否则，集群中的每个节点可能会选择信息，该消息旨在作为另一节点上的回复。在集群环境中，建议使用共享回复队列。

179.15. 在发送者和接收器间同步时钟

在系统间消息传递时，需要系统同步时钟。例如，在发送 JMS 消息时，您可以将一个生存时间设置为消息上的实时值。然后，接收器可以检查这个值，并确定消息是否已过期，从而丢弃消息，而不是消耗并处理它。但是，这需要发送者和接收器都有同步时钟。如果使用 ActiveMQ，您可以使用 时间戳插件来同步时钟。

179.16. 关于生存时间

阅读以上关于同步时钟的先读。

当您使用 Camel 通过 Camel 对 JMS 进行请求/回复(InOut)时，Camel 会使用发送端的超时，这是 requestTimeout 选项的默认 20 秒。您可以通过设置更高的/lower 值来控制它。但是，在要发送的 JMS 消息上仍设置了生存时间值。因此，这需要在系统间同步时钟。如果没有，您可能想禁用设置生存时间值。现在，可以使用 Camel 2.8 onwards 中的 disableTimeToLive 选项。因此，如果您将此选项设置为 disableTimeToLive=true，则 Camel 在发送 JMS 消息时 不会将 任何时间设置为 live 值。但是 请求超时仍处于活动状态。例如，如果您通过 JMS 进行请求/回复，并且禁用了生存时间，则 Camel 仍将使用 20 秒(requestTimeout 选项)的超时。也可自行配置该选项。因此，两个选项 requestTimeout 和 disableTimeToLive 可让您在进行请求/回复时进行精细的控制。

从 Camel 2.13/2.12.3 开始，您可以在消息中提供标头来覆盖并使用 作为请求超时值，而不是端点配置的值。例如：

```
from("direct:someWhere")
  .to("jms:queue:foo?replyTo=bar&requestTimeout=30s")
  .to("bean:processReply");
```

在上面的路由中，我们有一个端点将 requestTimeout 配置为 30 秒。因此，Camel 将等待 30 秒，以便该回复消息回到栏队列中。如果没有收到回复消息，则在 Exchange 中设置 org.apache.camel.ExchangeTimedOutException，Camel 会继续路由消息，然后因为例外和 Camel 的错误处理程序响应而失败。

如果要使用每个消息超时值，您可以使用键 org.apache.camel.component.jms.JmsConstants SerialJMS_REQUEST_TIMEOUT 设置标头，其常量值为 "CamelJmsRequestTimeout"，超时值为长类型。

例如，我们可以使用 bean 来计算每个消息的超时值，如调用服务 bean 上的 "whatIsTheTimeout" 方法，如下所示：

```
from("direct:someWhere")
  .setHeader("CamelJmsRequestTimeout", method(ServiceBean.class, "whatIsTheTimeout"))
  .to("jms:queue:foo?replyTo=bar&requestTimeout=30s")
  .to("bean:processReply");
```

当您使用 Camel 对 JMS 进行触发和忘记(InOut)时，默认情况下 Camel 不会 随时将消息的值设置为 live 值。您可以使用 timeToLive 选项配置值。例如，要指明 5 sec，您可以设置 timeToLive=5000。选项 disableTimeToLive 可用于强制将时间禁用为 live，也用于 InOnly messaging。requestTimeout 选项不用于 InOnly messaging。

179.17. 启用 TRANSACTED CONSUMPTION

常见的要求是从事务中的队列消耗，然后使用 Camel 路由处理消息。要做到这一点，只确保您在组件/端点上设置以下属性：

- transacted = true
- transactionManager = Transsaction Manager - 通常 JmsTransactionManager

详情请查看 Transactional Client EIP 模式。

事务和 [Request Reply] over JMS

使用 Request Reply over JMS 时，无法使用单个事务；JMS 不会在执行提交前发送任何消息，因此服务器端不会接收任何消息，直到事务提交为止。因此，要使用 Request Reply，您必须在发送请求后提交事务，然后使用单独的事务来接收响应。

要解决这个问题，JMS 组件使用不同的属性来指定事务用于单向消息传递和请求回复消息传递：

transacted 属性只适用于 InOnly message Exchange Pattern (MEP)。

transactedInOut 属性应用到 InOut (Request Reply)消息交换模式(MEP)。

如果要将事务用于 Request Reply(InOut MEP)，您必须设置 transactedInOut=true。

从 Camel 2.10 开始提供

您可以使用组件/端点中的以下属性来利用 DMLC 转换会话 API ：

- transacted = true

- *lazyCreateTransactionManager = false*

这样做的好处是，在使用没有配置的 TransactionManager 的情况下，cacheLevel 设置将会被遵守。配置 TransactionManager 时，在 DMLC 级别不会发生缓存，需要依赖池的连接工厂。有关此类设置的详情，请查看 此处 和 此处。

179.18. 使用 JMSREPLYTO 进行回复

将 Camel 用作 JMS 侦听器时，它会使用 ReplyTo javax.jms.Destination 对象的值设置 Exchange 属性，其键为 ReplyTo。您可以按照以下方法 获取 此目标：

```
Destination replyDestination =
exchange.getIn().getHeader(JmsConstants.JMS_REPLY_DESTINATION, Destination.class);
```

然后稍后使用它来通过常规 JMS 或 Camel 发送回复。

```
// we need to pass in the JMS component, and in this sample we use ActiveMQ
JmsEndpoint endpoint = JmsEndpoint.newInstance(replyDestination, activeMQComponent);
// now we have the endpoint we can use regular Camel API to send a message to it
template.sendBody(endpoint, "Here is the late reply.");
```

发送回复的不同解决方案是在发送时在同一 Exchange 属性中提供 replyDestination 对象。然后，Camel 会获取此属性并将其用于实际目的地。但是，端点 URI 必须包含 dummy 目标。例如：

```
// we pretend to send it to some non existing dummy queue
template.send("activemq:queue:dummy, new Processor() {
  public void process(Exchange exchange) throws Exception {
    // and here we override the destination with the ReplyTo destination object so the message
is sent to there instead of dummy
    exchange.getIn().setHeader(JmsConstants.JMS_DESTINATION, replyDestination);
    exchange.getIn().setBody("Here is the late reply.");
  }
}
```

179.19. 使用请求超时

在以下示例中，我们将 Request Reply 风格的消息交换（我们使用 requestBody method = InOut）发送给 slow 队列以在 Camel 中进一步处理，我们等待返回回复：

179.20. SAMPLES

JMS 也用于其他组件。但我们提供了几个示例来开始。

179.20.1. 从 JMS 接收

在以下示例中，我们配置一个接收 JMS 消息的路由，并将消息路由到 POJO：

```
from("jms:queue:foo").
  to("bean:myBusinessLogic");
```

当然，您可以使用任何 EIP 模式，以便路由可以基于上下文。例如，以下为大型开支过滤订单主题：

```
from("jms:topic:OrdersTopic").
  filter().method("myBean", "isGoldCustomer").
    to("jms:queue:BigSpendersQueue");
```

179.20.2. 发送到 JMS

在以下示例中，我们轮询文件文件夹，并将文件内容发送到 JMS 主题。我们希望文件的内容作为 TextMessage 而不是 BytesMessage，因此我们需要将正文转换为 String ：

```
from("file://orders").
  convertBodyTo(String.class).
  to("jms:topic:OrdersTopic");
```

179.20.3. 使用注解

Camel 还具有注解，因此您可以使用 POJO 消耗和 POJO Producing。

179.20.4. Spring DSL 示例

前面的示例使用了 Java DSL。Camel 还支持 Spring XML DSL。以下是使用 Spring DSL 的大型开销示例：

```
<route>
  <from uri="jms:topic:OrdersTopic"/>
  <filter>
    <method bean="myBean" method="isGoldCustomer"/>
    <to uri="jms:queue:BigSpendersQueue"/>
  </filter>
</route>
```

179.20.5. 其他示例

JMS 出现在其他组件和 EIP 模式的许多示例中，以及 Camel 文档。因此请随时浏览文档。如果您有时间，请查看本教程，该教程使用 JMS，但侧重于 Spring Remoting 和 Camel 如何一起工作 Tutorial-JmsRemoting。

179.20.6. 使用 JMS 作为死信队列存储交换

通常，当将 JMS 用作传输时，它只传输正文和标头作为载荷。如果要将 JMS 与死信 频道 搭配使用，请将 JMS 队列用作 Dead Letter Queue，通常原因的 Exception 不在 JMS 消息中。但是，您可以对 JMS 死信队列使用 transferExchange 选项，以指示 Camel 将整个 Exchange 存储在队列中，作为 javax.jms.ObjectMessage，其中包含 org.apache.camel.impl.DefaultExchangeHolder。这样，您可以从死信队列使用，并使用密钥 Exchange.EXCEPTION_CAUGHT 从 Exchange 属性检索导致的异常。以下演示说明了这一点：

```
// setup error handler to use JMS as queue and store the entire Exchange
errorHandler(deadLetterChannel("jms:queue:dead?transferExchange=true"));
```

然后，您可以从 JMS 队列中消耗并分析问题：

```
from("jms:queue:dead").to("bean:myErrorAnalyzer");

// and in our bean
String body = exchange.getIn().getBody();
Exception cause = exchange.getProperty(Exchange.EXCEPTION_CAUGHT, Exception.class);
// the cause message is
String problem = cause.getMessage();
```

179.20.7. 仅将 JMS 用作死信频道存储错误

您可以使用 JMS 来存储原因错误消息或存储自定义正文，供您自行初始化。以下示例使用 Message Translator EIP 在进入 JMS 死信队列前对失败的交换进行转换：

```
// we sent it to a seda dead queue first
errorHandler(deadLetterChannel("seda:dead"));

// and on the seda dead queue we can do the custom transformation before its sent to the
JMS queue
from("seda:dead").transform(exceptionMessage()).to("jms:queue:dead");
```

在这里，我们在转换中仅存储原始原因错误消息。但是，您可以使用任何 Expression 来发送您喜欢的任何对象。例如，您可以在 Bean 上调用方法或使用自定义处理器。

179.21. 发送 INONLY 消息并保持 JMSREPLYTO 标头

使用 camel-jms 发送到 JMS 目的地时，生成者将使用 MEP 来检测其 InOnly 或 InOut 消息传递。然而，您可能希望发送 InOnly 消息，但保留 JMSReplyTo 标头。为此，您必须指示 Camel 保留它，否则将丢弃 JMSReplyTo 标头。

例如，要将 InOnly 消息发送到 foo 队列，但使用 JMSReplyTo 及 bar 队列，您可以执行以下操作：

```
template.send("activemq:queue:foo?preserveMessageQos=true", new Processor() {
  public void process(Exchange exchange) throws Exception {
    exchange.getIn().setBody("World");
    exchange.getIn().setHeader("JMSReplyTo", "bar");
  }
});
```

请注意，我们使用 preserveMessageQos=true 来指示 Camel 保留 JMSReplyTo 标头。

179.22. 在目的地上设置 JMS 提供程序选项

某些 JMS 提供程序，如 IBM 的 WebSphere MQ 需要在 JMS 目的地上设置选项。例如，您可能需要指定 targetClient 选项。由于 targetClient 是 WebSphere MQ 选项，而不是 Camel URI 选项，您需要在 JMS 目标名称上设置，如下所示：

```
// ...
.setHeader("CamelJmsDestinationName", constant("queue:///MY_QUEUE?targetClient=1"))
.to("wmq:queue:MY_QUEUE?useMessageIDAsCorrelationID=true");
```

某些版本的 WMQ 不接受目的地名称上的此选项，您会遇到如下例外：

```
com.ibm.msg.client.jms.DetailedJMSException: JMSCC0005: The specified
value 'MY_QUEUE?targetClient=1' is not allowed for
'XMSC_DESTINATION_NAME'
```

临时解决方案是使用自定义 DestinationResolver：

```
JmsComponent wmq = new JmsComponent(connectionFactory);

wmq.setDestinationResolver(new DestinationResolver() {
   public Destination resolveDestinationName(Session session, String destinationName,
boolean pubSubDomain) throws JMSException {
       MQQueueSession wmqSession = (MQQueueSession) session;
```

```
        return wmqSession.createQueue("queue:///" + destinationName + "?targetClient=1");
    }
});
```

179.23. 另请参阅

- JMSTemplate gotchas

第 180 章 JMX 组件

180.1. CAMEL JMX

Apache Camel 对 JMX 有广泛的支持，允许您使用 JMX 客户端监控和管理 Camel 管理对象。

Camel 还提供了一个 JMX 组件，允许您订阅 MBean 通知。本页介绍如何使用 JMX 管理和监控 Camel。

180.2. 选项

JMX 组件没有选项。

JMX 端点使用 URI 语法进行配置：

```
jmx:serverURL
```

使用以下路径和查询参数：

180.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
serverURL	服务器 url 来自剩余的端点。使用平台连接到本地 JVM。		字符串

180.2.2. 查询参数(30 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值

Name	描述	默认值	类型
格式 (consumer)	消息正文格式。xml 或 raw。如果 xml，则通知被序列化为 xml。如果 raw，则原始 java 对象被设置为正文。	xml	字符串
granularityPeriod (consumer)	轮询 bean 以检查 monitor 的频率（仅监控类型）。	10000	long
monitorType (consumer)	要创建的 monitor 类型。一个字符串、量表、计数器（仅限监控类型）。		字符串
objectDomain (consumer)	要连接的 mbean **所需的** 域		字符串
objectName (consumer)	要连接的 mbean 的名称键。这个值与传递的对象属性相互排斥。		字符串
observedAttribute (consumer)	要观察 monitor bean 或 consumer 的属性。		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
executorService (advanced)	将自定义共享线程池用于消费者。默认情况下，每个消耗都有自己的 thread-pool 来处理和路由通知。		ExecutorService
handback (advanced)	收到通知时，返回到侦听器的值。这个值将放置在带有键 jmx.handback 的消息标头中		对象
notificationFilter (advanced)	对实现 NotificationFilter 的 bean 的引用。		NotificationFilter
objectProperties (advanced)	对象名称的属性。如果没有设置 objectName param，则使用这些值		Map
reconnectDelay (advanced)	尝试重试初始连接建立前等待的秒数，或者尝试重新连接丢失的连接	10	int
reconnectOnConnection Failure (advanced)	如果为 true，则消费者将在发生任何连接失败时尝试重新连接到 JMX 服务器。消费者将尝试每 'x' 秒重新建立 JMX 连接，直到连接被建立 - 其中 'x' 是配置的 reconnectionDelay	false	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

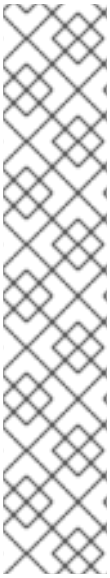
Name	描述	默认值	类型
testConnectionOnStartup (advanced)	如果为 true，如果在启动时无法建立 JMX 连接，消费者将抛出异常。如果为 false，则消费者将尝试每 'x' 秒建立 JMX 连接，直到连接被 makewagon-wagonwhere 'x' 是配置的 reconnectionDelay	true	布尔值
notifyDiffer (string)	如果为 true，当字符串属性与要比较的字符串不同时（字符串 monitor 或 consumer）将触发通知。默认情况下，如果配置了要比较的观察属性和字符串，使用者将通知匹配项。	false	布尔值
notifyMatch (string)	如果为 true，当 string 属性与要比较的字符串匹配时（字符串 monitor 或 consumer）将触发通知。默认情况下，如果配置了要比较的观察属性和字符串，使用者将通知匹配项。	false	布尔值
stringToCompare (string)	要比较的属性的值（字符串 monitor 或 consumer）。默认情况下，如果配置了要比较的观察属性和字符串，使用者将通知匹配项。		字符串
initThreshold (counter)	monitor 的初始阈值。该值必须在触发通知前超过这个值（仅限计数）。		int
modulus (counter)	计数器重置为零（仅监控数）的值。		int
offset (counter)	超过阈值的数量（只限于监控）。		int
differenceMode (gauge)	如果为 true，则通知中报告的值与阈值不同，而不是值本身（仅限计数和量表监视器）。	false	布尔值
notifyHigh (gauge)	如果为 true，则量表将在超过高阈值时触发通知（仅计算 monitor）。	false	布尔值
notifyLow (gauge)	如果为 true，则量表将在超过低阈值时触发通知（仅计算 monitor）。	false	布尔值
thresholdHigh (gauge)	量表高阈值（仅gauge monitor）的值。		å�Œ
thresholdLow (gauge)	量表低阈值的值（仅限 gauge monitor）。		å�Œ
密码 (security)	用于进行远程连接的凭证		字符串
用户 （安全）	用于进行远程连接的凭证		字符串

180.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.jmx.enabled	启用 jmx 组件	true	布尔值
camel.component.jmx.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

180.4. 在 CAMEL 中激活 JMX



注意

Camel 2.8 或更早的需要 Spring JAR 依赖项

spring-context.jar,spring-aop.jar,spring-beans.jar, 和 spring-core.jar 以便在 Camel 的 classpath 上需要使用 JMX 检测。如果这些 .jar 不在 classpath 上，则 Camel 将回退到非 JMX 模式。此情况使用日志记录器名称 org.apache.camel.impl.DefaultCamelContext 记录在 WARN 级别。

从 Camel 2.9 开始，Spring JAR 不再需要 以 JMX 模式运行 Camel。

180.4.1. 使用 JMX 管理 Apache Camel

默认情况下，在 Camel 中启用了 JMX 检测代理，这意味着 Camel 运行时会在虚拟机中使用 MBeanServer 实例创建和注册 MBean 管理对象。这允许 Camel 用户立即获取对 Camel 路由如何对各个处理器级别执行的见解。

支持的管理对象类型有 端点、路由、服务 和 处理器。除了性能计数器属性外，其中一些管理对象还公开生命周期操作。

DefaultManagementNamingStrategy 是默认的命名策略，它构建用于 MBean 注册的对象名称。默认情况下，org.apache.camel 是 CamelNamingStrategy 创建的所有对象名称。MBean 对象的域名可

由 Java VM 系统属性配置：

```
-Dorg.apache.camel.jmx.mbeanObjectDomainName=your.domain.name
```

或者，通过在 Spring 配置中的 camelContext 元素中添加 jmxAgent 元素：

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <jmxAgent id="agent" mbeanObjectDomainName="your.domain.name"/>
    ...
</camelContext>
```

当这两个配置都存在时，Spring 配置始终优先于系统属性。所有 JMX 相关配置都为 true。

180.4.2. 在 Camel 中禁用 JMX 检测代理

您可以通过设置 Java VM 系统属性来禁用 JMX 检测代理，如下所示：

```
-Dorg.apache.camel.jmx.disabled=true
```

属性值被视为 布尔值。

或者，通过在 Spring 配置中的 camelContext 元素中添加 jmxAgent 元素：

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <jmxAgent id="agent" disabled="true"/>
    ...
</camelContext>
```

或在 Camel 2.1 中，使用纯 Java 时（无需使用 JVM 系统属性），您可以禁用它，如下所示：

```
CamelContext camel = new DefaultCamelContext();
camel.disableJMX();
```

180.4.3. 在 Java 虚拟机中查找 MBeanServer

每个 CamelContext 都可以有一个 InstrumentationAgent 实例嵌套在 InstrumentationLifecycleStrategy 中。InstrumentationAgent 是与 MBeanServer 进行注册 / unregister Camel MBeans 的接口的对象。多个 CamelContexts / InstrumentationAgents can / should share a MBeanServer.默认情况下，Camel 运行时会选择 MBeanServer

Factory.findMBeanServer 方法返回的第一个 MBeanServer，它与 org.apache.camel 的默认域名匹配。

您可能需要更改默认域名，以匹配已在应用中使用的 MBeanServer 实例。特别是，如果您的 MBeanServer 附加到 JMX 连接器服务器，则不需要在 Camel 中创建连接器服务器。

您可以通过系统属性配置匹配的默认域名。

```
-Dorg.apache.camel.jmx.mbeanServerDefaultDomain=<your.domain.name>
```

或者，通过在 Spring 配置中的 camelContext 元素中添加 jmxAgent 元素：

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <jmxAgent id="agent" mbeanServerDefaultDomain="your.domain.name"/>
    ...
</camelContext>
```

如果没有找到匹配的 MBeanServer，则会创建一个新名称，并根据上述默认配置设置新的 'MBeanServer's 默认域名。

当需要通过设置系统属性来管理 JVM MBeans 时，也可以使用 PlatformMBeanServer。MBeanServer 默认域名配置会被忽略，因为它不适用。

小心

从下一个版本(1.5)开始，usePlatformMBeanServer 的默认值将更改为 true。您可以使用 platform MBeanServer 将属性设置为 false 以禁用。

```
-Dorg.apache.camel.jmx.usePlatformMBeanServer=True
```

或者，通过在 Spring 配置中的 camelContext 元素中添加 jmxAgent 元素：

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <jmxAgent id="agent" usePlatformMBeanServer="true"/>
    ...
</camelContext>
```

180.4.4. Camel JMX 的系统属性支持

属性名称	value	描述
org.apache.camel.jmx	**true** 或 **false**	如果为 **true**，它将在 Camel 中启用 jmx 功能

请参见本节中的更多系统属性： jmxAgent Properties Reference。

180.4.5. 如何将身份验证与 JMX 搭配使用

JDK 中的 JMX 具有用于身份验证的功能，以及通过 SSL 使用安全连接。您必须参考 SUN 文档如何使用它：

- *http://java.sun.com/j2se/1.5.0/docs/guide/management/agent.html*
- *http://java.sun.com/javase/6/docs/technotes/guides/management/agent.html*

180.4.6. 应用服务器内的 JMX

180.4.6.1. Tomcat 6

有关在 Tomcat 中启用 JMX 的详情，请查看 此页面。

简而言之，修改您的 catalina.sh （或 Windows 中的 catalina.bat），以设置以下选项...

```
set CATALINA_OPTS=-Dcom.sun.management.jmxremote \
  -Dcom.sun.management.jmxremote.port=1099 \
  -Dcom.sun.management.jmxremote.ssl=false \
  -Dcom.sun.management.jmxremote.authenticate=false
```

180.4.6.2. JBoss AS 4

默认情况下，JBoss 创建自己的 MBeanServer。要允许 Camel 向同一服务器公开，请按照以下步骤操作：

1. *告诉 Camel 使用 Platform MBeanServer （这在 Camel 1.5 中默认为 true）*

```
<camel:camelContext id="camelContext">
  <camel:jmxAgent id="jmxAgent" mbeanObjectDomainName="org.yourname"
usePlatformMBeanServer="true"  />
</camel:camelContext>
```

1. 更改您的 JBoss 实例以使用 Platform MBeanServer。
通过编辑 run.sh 或 run.conf -Djboss.platform.mbeanserver，将以下属性添加到 JAVA_OPTS。See http://wiki.jboss.org/wiki/JBossMBeansInJConsole

180.4.6.3. WebSphere

将 mbeanServerDefaultDomain 更改为 WebSphere ：

```
<camel:jmxAgent id="agent" createConnector="true" mbeanObjectDomainName="org.yourname"
usePlatformMBeanServer="false" mbeanServerDefaultDomain="WebSphere"/>
```

180.4.6.4. Oracle OC4j

Oracle OC4J J2EE 应用服务器将不允许 Camel 访问平台 MBeanServer。您可以在日志中识别这一点，因为 Camel 将记录 WARNING。

```
xxx xx, xxxx xx:xx:xx xx org.apache.camel.management.InstrumentationLifecycleStrategy
onContextStart
WARNING: Could not register CamelContext MBean
java.lang.SecurityException: Unauthorized access from application: xx to MBean:
java.lang:type=ClassLoading
        at
oracle.oc4j.admin.jmx.shared.UserMBeanServer.checkRegisterAccess(UserMBeanServer.java:873)
```

要解决这个问题，您应该在 Camel 中禁用 JMX 代理，请参阅在 Camel 中禁用 JMX 检测代理。

180.4.7. 高级 JMX 配置

Spring 配置文件允许您配置如何将 Camel 公开给 JMX 进行管理。在某些情况下，您可以在此处指定更多信息，如连接器的端口或路径名称。

180.4.8. Example:

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <jmxAgent id="agent" createConnector="true" registryPort="2000"
mbeanServerDefaultDomain="org.apache.camel.test"/>
    <route>
      <from uri="seda:start"/>
```

```
    <to uri="mock:result"/>
  </route>
</camelContext>
```

如果要更改 Java 5 JMX 设置，您可以使用各种 JMX 系统属性

例如，您可以通过设置以下环境变量（根据您的平台使用 设置 或导出 ）启用到 Sun JMX 连接器的远程 JMX 连接。这些设置仅在 Java 1.5+ 中配置 Sun JMX 连接器，而不是 Camel 默认创建的 JMX 连接器。

```
SUNJMX=-Dcom.sun.management.jmxremote=true -Dcom.sun.management.jmxremote.port=1616 \
-Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false
```

（SUNJMX 环境变量是 Camel 的启动脚本简单，作为 JVM 的额外启动参数。如果您直接启动 Camel，则必须自行传递这些参数。）

180.4.9. jmxAgent Properties Reference

Spring 属性	系统属性	默认值	描述
id			JMX 代理名称，它并不是可选的
usePlatformMBeanServer	**org.apache.camel.jmx.usePlatformMBeanServer**	**false**,**true** - Release 1.5 或更高版本	如果为 **true**，它将使用 JVM 中的 **MBeanServer**
mbeanServerDefaultDomain	**org.apache.camel.jmx.mbeanServerDefaultDomain**	**org.apache.camel**	**MBeanServer**的默认 JMX 域
mbeanObjectDomainName	**org.apache.camel.jmx.mbeanObjectDomainName**	**org.apache.camel**	所有对象名称都使用的 JMX 域
createConnector	**org.apache.camel.jmx.createRmiConnect**	**false**	如果我们应该为 **MBeanServer**创建 JMX 连接器（以允许远程管理）
registryPort	**org.apache.camel.jmx.rmiConnector.registryPort**	**1099**	JMX RMI registry 将使用的端口

Spring 属性	系统属性	默认值	描述
connectorPort	**org.apache.camel.jmx.rmiConnector.connectorPort**	-1（动态）	JMX RMI 服务器要使用的端口
serviceUrlPath	**org.apache.camel.jmx.serviceUrlPath**	**/jmxrmi/camel**	JMX 连接器将在其下注册的路径
onlyRegisterProcessorWithCustomId	**org.apache.camel.jmx.onlyRegisterProcessorWithCustomId**	**false**	Camel 2.0： 如果启用这个选项，则只有带有自定义 id 集合的处理器才会注册。这可让您在 JMX 控制台中处理不需要的处理器。
statisticsLevel		**所有/默认**	Camel 2.1： 配置是否为 MBean 启用性能统计的级别。如需了解更多详细信息，*请参阅为性能统计配置粒度级别*。从 **Camel 2.16** 开始，All 选项被重命名为 Default，并引入了一个新的扩展选项，允许收集额外的运行时 JMX 指标。
includeHostName	**org.apache.camel.jmx.includeHostName**		Camel 2.13 ： 是否要在 MBean 命名中包含主机名。从 Camel 2.13 开始，这是 **false**，在旧版本中，默认为 **true**。如果真正需要，您可以使用这个选项恢复旧行为。
useHostIPAddress	**org.apache.camel.jmx.useHostIPAddress**	**false**	Camel 2.16 ： 在创建远程连接器时，是否在服务 url 中使用主机名或 IP 地址。默认情况下将使用主机名。
loadStatisticsEnabled	**org.apache.camel.jmx.loadStatisticsEnabled**	**false**	Camel 2.16 ：是否启用负载统计（使用每个 CamelContext 的后台线程收集负载统计）。

Spring 属性	系统属性	默认值	描述
endpointRuntimeStatisticsEnabled	**org.apache.camel.jmx.endpointRuntimeStatisticsEnabled**	**true**	Camel 2.16： 是否启用端点运行时统计信息（每个传入和传出端点的收集运行时使用情况）。

180.4.10. 配置是否始终注册 MBeans always、新路由或默认注册

从 Camel 2.7 开始提供

Camel 现在提供 2 个设置来控制是否要注册 mbeans

选项	默认	描述
registerAlways	**false**	如果启用，则始终注册 MBeans。
registerNewRoutes	**true**	如果启用，在 CamelContext 启动后添加新路由也会从那个给定路由注册 MBeans。

默认情况下，Camel 为配置的所有路由注册 MBeans。如果稍后添加新路由，registerNewRoutes 选项控制 MBeans 是否应该被注册。例如，您可以禁用此功能，例如，在不需要管理的情况下添加和删除临时路由。

在使用动态 EIP 模式（如 Recipient List 具有唯一端点）时，请小心使用 registerAlways 选项。如果这样，每个唯一的端点及其关联的服务/producer 也会被注册。这可能会导致因为 registry 中大量 mbeans 导致系统分离。MBean 不是轻量级对象，因此消耗内存。

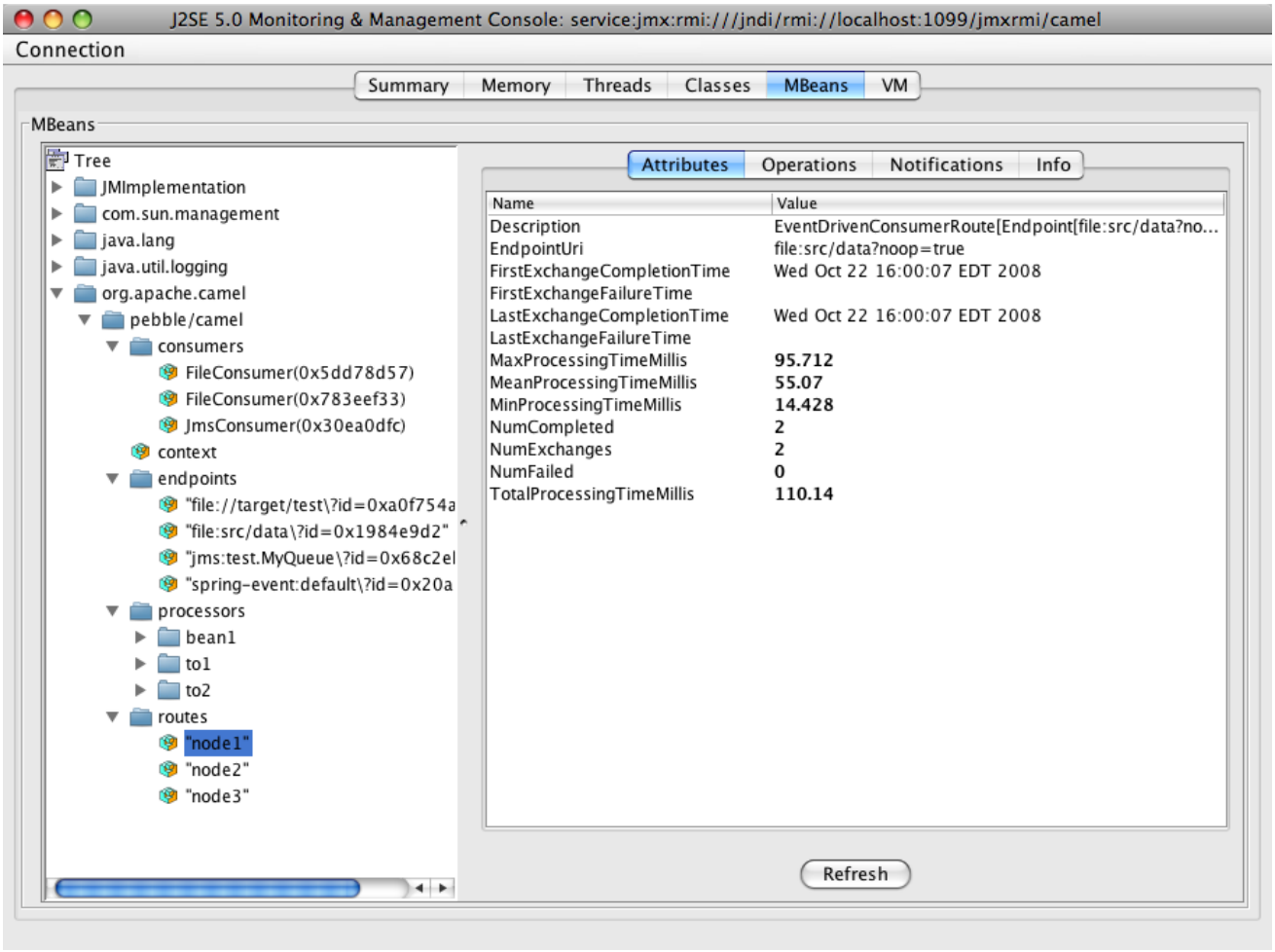
180.5. 使用 JMX 监控 CAMEL

180.5.1. 使用 JConsole 监控 Camel

如果您在与 Camel 相同的主机上运行 JConsole，则 CamelContext 应该出现在本地连接列表中。

要连接到远程 Camel 实例，或者本地进程未显示，请使用 Remote Process 选项，并输入 URL。以下是 localhost URL:service:jmx:rmi:///jndi/rmi://localhost:1099/jmxrmi/camel 示例。

使用带有 JConsole 的 Apache Camel：



180.5.2. 注册哪些端点

在 Camel 2.1 中，只有 单例 端点注册为非单例的开销将在使用数千或数百万端点的情况下大量注册。当使用 Recipient List EIP 或发送大量消息的 ProducerTemplate 时，会出现这种情况。

180.5.3. 哪些处理器被注册

请查看此常见问题。

180.5.4. 如何使用 JMX NotificationListener 侦听 camel 事件？

Camel 通知事件给出粗略概述发生的情况。您可以查看上下文和端点的生命周期事件，您可以看到由端点接收的交换并发送到端点。

在 Camel 2.4 中，您可以使用自定义 JMX NotificationListener 侦听 camel 事件。

首先，您需要在启动 CamelContext 前设置 JmxNotificationEventNotifier ：

```
// Set up the JmxNotificationEventNotifier
notifier = new JmxNotificationEventNotifier();
notifier.setSource("MyCamel");
notifier.setIgnoreCamelContextEvents(true);
notifier.setIgnoreRouteEvents(true);
notifier.setIgnoreServiceEvents(true);

CamelContext context = new DefaultCamelContext(createRegistry());
context.getManagementStrategy().addEventNotifier(notifier);
```

其次，您可以注册侦听事件的监听程序：

```
// register the NotificationListener
ObjectName on = ObjectName.getInstance("org.apache.camel:context=camel-
1,type=eventnotifiers,name=JmxEventNotifier");
MyNotificationListener listener = new MyNotificationListener();
context.getManagementStrategy().getManagementAgent().getMBeanServer().addNotificationLi
stener(on,
   listener,
   new NotificationFilter() {
      private static final long serialVersionUID = 1L;

      public boolean isNotificationEnabled(Notification notification) {
         return notification.getSource().equals("MyCamel");
      }
   }, null);
```

180.5.5. 使用 Tracer MBean 获取精细的追踪

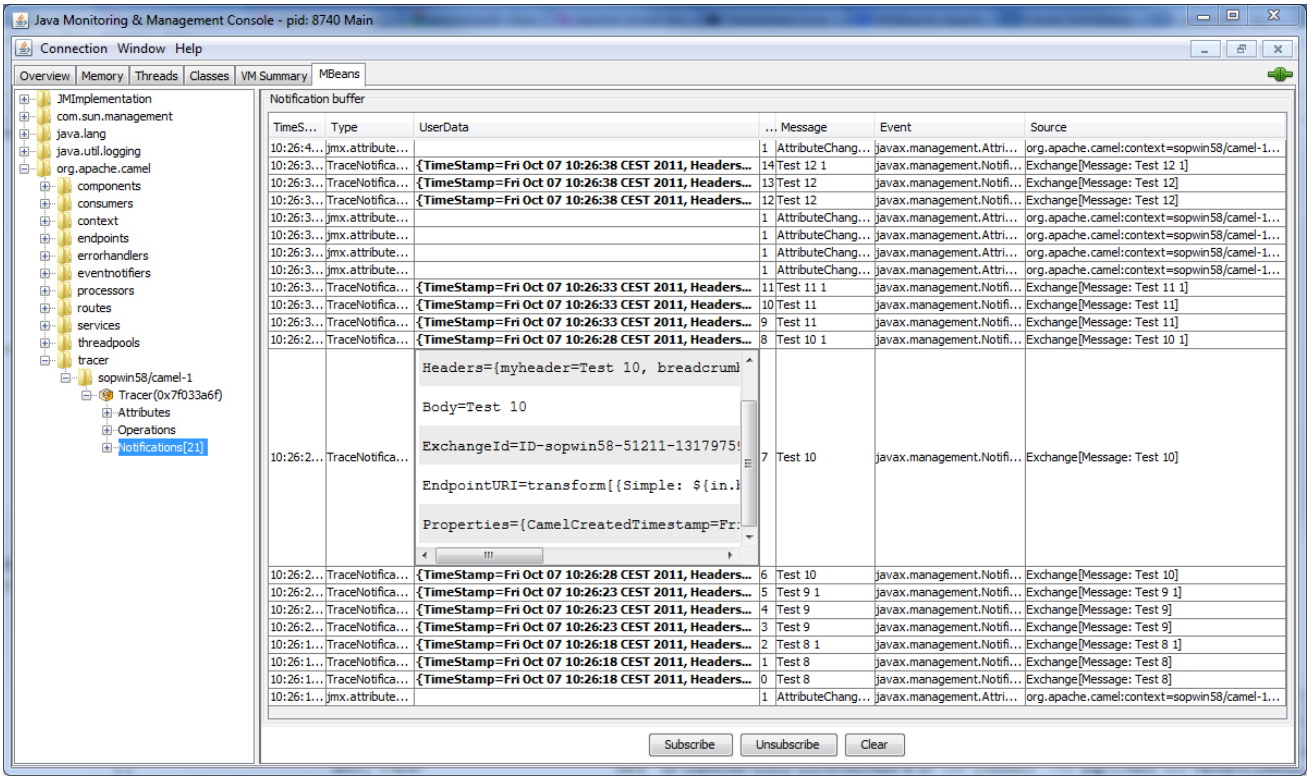
除了 Camel 2.9.0 以上的粗粒度通知外，还支持 JMX 通知以进行细粒度跟踪事件。

它们可以在 Tracer MBean 中找到。要激活精细的追踪，首先需要在上下文或路由上激活追踪。

这可以在配置上下文或上下文/路由 MBeans 时完成。

作为第二步，您必须在 tracer 上将 jmxTraceNotifications 属性设置为 true。在配置上下文或 tracer MBean 的运行时，可以再次完成此操作。

现在，您可以使用 JConsole 在 Tracer MBean 上注册 TraceEvent 通知。路由上带有所有交换和消息详情的每个步骤都会有一个通知：



180.6. 将 JMX 用于您自己的 CAMEL 代码

180.6.1. 注册您自己的受管端点

从 Camel 2.0 开始，您可以使用 Spring managed annotations @ManagedResource 分离自己的端点，以允许在 Camel MBeanServer 中注册它们，从而使用 JMX 访问您的自定义 MBeans。

注意

在 Camel 2.1 中，我们改为应用端点以外的其他内容，但还需要实施接口 org.apache.camel.spi.ManagementAware。更多关于此内容的信息。

例如，我们有以下自定义端点，我们定义要管理的一些选项：

```
@ManagedResource(description = "Our custom managed endpoint")
```

```
public class CustomEndpoint extends MockEndpoint implements
ManagementAware<CustomEndpoint> {

    public CustomEndpoint(final String endpointUri, final Component component) {
        super(endpointUri, component);
    }

    public Object getManagedObject(CustomEndpoint object) {
        return this;
    }

    public boolean isSingleton() {
        return true;
    }

    protected String createEndpointUri() {
        return "custom";
    }

    @ManagedAttribute
    public String getFoo() {
        return "bar";
    }

    @ManagedAttribute
    public String getEndpointUri() {
        return super.getEndpointUri();
    }
}
```

鼓励从 Camel 2.9 开始，使用 org.apache.camel.api.management 软件包中的 @ManagedResource、@ManagedAttribute 和 @ManagedOperation。这可让您的自定义代码不依赖于 Spring JAR。

180.6.2. 编程您自己的托管服务

从 Camel 2.1 开始提供

Camel 现在提供在注册用于管理服务时使用您自己的 MBeans。例如，您可以开发自定义 Camel 组件，使其公开 MBeans 用于端点、消费者和制作者等。您只需要实现接口 org.apache.camel.spi.ManagementAware，并返回受管对象 Camel 应使用。

现在，在你认为 JMX API 确实非常困难、不错，你就没错了。Eucky for us Spring though 还创建了一系列注释，您可以使用它们在现有 Bean 上导出管理。这意味着，您通常使用它，且只从 ManagementAware 接口在 getManagedObject 中返回它。例如，请参阅上面的带有 CustomEndpoint 的代码示例。

现在，在 Camel 2.1 中，您可以为 Camel 注册管理的所有对象执行此操作，而这是个生动，但不是全部。

对于没有实现此 ManagementAware 接口的服务，Camel 将使用下表中定义的默认打包程序：

类型	MBean wrapper
CamelContext	**ManagedCamelContext**
组件	**ManagedComponent**
端点	**ManagedEndpoint**
消费者	**ManagedConsumer**
制作者	**ManagedProducer**
Route	**ManagedRoute**
处理器	**ManagedProcessor**
tracer	**ManagedTracer**
Service	**ManagedService**

除了一些用于特殊类型的扩展打包程序外，例如：

类型	MBean wrapper
ScheduledPollConsumer	**ManagedScheduledPollConsumer**
BrowsableEndpoint	**ManagedBrowseableEndpoint**
Throttler	**ManagedThrottler**
Delayer	**ManagedDelayer**
SendProcessor	**ManagedSendProcessor**

以后，我们将为更多 EIP 模式添加额外的打包程序。

180.6.3. ManagementNamingStrategy

从 Camel 2.1 开始提供

Camel 通过 org.apache.camel.spi.ManagementNamingStrategy 为命名策略提供可插拔 API。默认实现用于计算所有 MBeans 注册的 MBean 名称。

180.6.4. 管理命名模式

从 Camel 2.10 开始提供

从 Camel 2.10 开始，我们更容易为 MBeans 配置命名模式。模式用作 ObjectName 的一部分，作为域名后面的键。

默认情况下，Camel 将对 ManagedCamelContextMBean 使用 MBean 名称，如下所示：

```
org.apache.camel:context=localhost/camel-1,type=context,name=camel-1
```

从 Camel 2.13 开始，主机名不包括在 MBean 名称中，因此上面的示例如下：

```
org.apache.camel:context=camel-1,type=context,name=camel-1
```

如果您在 CamelContext 上配置一个名称，则该名称也是 ObjectName 的一部分。例如，如果我们有

```
<camelContext id="myCamel" ...>
```

然后 MBean 名称将如下：

```
org.apache.camel:context=localhost/myCamel,type=context,name=myCamel
```

现在，如果 JVM 中存在一个命名冲突，例如，在上面具有给定名称的 MBean 已存在，则 Camel 将默认尝试通过使用计数器在 JMXMBeanServer 中查找新的空闲名称来自动更正。如计数器所示，现在附加 myCamel-1 作为 ObjectName 的一部分：

```
org.apache.camel:context=localhost/myCamel-1,type=context,name=myCamel
```

这是因为 Camel 默认使用命名模式来支持以下令牌：

- *camelId = CamelContext id （例如，名称）*
- *name - 与 camelId相同*
- *计数器 - 递增计数器 * bundleId - OSGi 捆绑包 id （仅适用于 OSGi 环境）*
- *symbolicName - OSGi 符号链接名称（仅适用于 OSGi 环境）*
- *版本 - OSGi 捆绑包版本（仅适用于 OSGi 环境）*

默认命名模式在 OSGi 和非OSGi 之间区分，如下所示：

- *非 OSGI： 名称*
- *OSGi: bundleId-name*
- *OSGi Camel 2.13: symbolicName*

但是，如果 JMXMBeanServer 中存在命名 clash，则 Camel 将自动回退，并在模式中使用 计数器 进行补救。因此，将使用以下模式：

- *非 OSGI: 名称计数器*
- *OSGi: bundleId-name-counter*
- *OSGi Camel 2.13: symbolicName-counter*

如果您设置了显式命名模式，则始终使用该模式，并且不使用 上述默认模式。

这样，我们可以完全控制在 Registry 中的 CamelContext id 以及 JMXMBeanRegistry 中的 JMX MBeans。

从 Camel 2.15 开始，您可以使用 JVM 系统属性配置默认管理名称模式，以便为 JVM 全局进行配置。请注意，您可以通过明确配置它来覆盖此模式，如下例所示。

设置 JVM 系统属性，以使用以 cool 为名称前缀的默认管理名称模式。

```
System.setProperty(JmxSystemPropertyKeys.MANAGEMENT_NAME_PATTERN, "cool-#name#");
```

因此，如果我们希望显式命名 CamelContext 并使用固定 MBean 名称，该名称不会改变（例如没有计数器），那么我们可以使用新的 managementNamePattern 属性：

```
<camelContext id="myCamel" managementNamePattern="#name#">
```

然后 MBean 名称将始终如下：

```
org.apache.camel:context=localhost/myCamel,type=context,name=myCamel
```

在 Java 中，您可以配置 managementNamePattern，如下所示：

```
context.getManagementNameStrategy().setNamePattern("#name#");
```

您还可以在与 id 不同的 managementNamePattern 中使用名称，例如，我们可以这样做：

```
<camelContext id="myCamel" managementNamePattern="coolCamel">
```

如果您不希望 OSGi 捆绑包 id 作为 MBean 名称的一部分，您可能希望在 OSGi 环境中执行此操作。如果您重新启动服务器，或者卸载并安装同一应用程序，则 OSGi 捆绑包 id 可能会改变。然后您可以执行以下操作，不要将 OSGi 捆绑包 id 用作名称的一部分：

```
<camelContext id="myCamel" managementNamePattern="#name#">
```

请注意，这需要 myCamel 在整个 JVM 中是唯一的。如果您安装了具有相同 CamelContext id 和 managementNamePattern 的 2nd Camel 应用程序，则 Camel 将在启动时失败，并报告 MBean 已存在异常。

180.6.5. ManagementStrategy

从 Camel 2.1 开始提供

Camel 现在提供了一个完全可插拔的管理策略，允许您控制管理。这是一个丰富的界面，具有多种用于管理方法。不仅用于从 MBeanServer 添加和删除受管对象，也提供了事件通知，也使用 org.apache.camel.spi.EventNotifier API。例如，它允许更轻松地为其他管理产品提供适配器。此外，它还允许您提供 Apache 中开箱即用的更多详细信息和功能。

180.6.6. 为性能统计配置粒度级别

从 Camel 2.1 开始提供

现在，您可以设置一个预设置级别，无论是启用性能统计数据，还是在 Camel 启动时启用。级别是

- 作为默认 扩展，但运行时会收集其他统计信息，如对端点的细粒度使用情况等。这个选项需要 Camel 2.16
- 所有 / Default - Camel 将为路由和处理器（细粒度）启用统计信息。在 All 选项后的 Camel 2.16 中被重命名为 Default。
- RoutesOnly - Camel 将仅为路由启用统计信息（粗粒度）
- off - Camel 不会为任何启用统计信息。

从 Camel 2.9 开始，性能统计还包括每个 CamelContext 和 Route MBeans 的平均负载统计。统计数据是基于动态交换数量的平均负载，每 1、5 和 15 分钟率。这和 Unix 系统上的负载统计信息类似。Camel 2.11 以后允许您通过在 < jmxAgent> 中设置 loadStatisticsEnabled=false 来显式 禁用负载性能统计信息。请注意，如果静态级别被配置为 off，它将处于 off 状态。从 Camel 2.13 开始的负载性能统计会被默认禁用。您可以通过在 < jmxAgent> 中设置 loadStatisticsEnabled=true 来启用它。

在运行时，您可以始终使用管理控制台（如 JConsole）来更改给定路由或处理器，无论其统计数据是启用的。

注意

启用统计数据的含义是什么？

启用统计数据意味着 Camel 将为特定 MBean 进行精细的性能统计信息。您可以看到的统计数据有很多，例如： Exchanges completed/failed、last/total/mina/max/mean 处理时间、第一/最后失败时间等。

使用 Java DSL，您可以通过以下方法设置这个级别：

```
// only enable routes when Camel starts
context.getManagementStrategy().setStatisticsLevel(ManagementStatisticsLevel.RoutesOnly);
```

在 Spring DSL 中：

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
    <jmxAgent id="agent" statisticsLevel="RoutesOnly"/>
 ...
</camelContext>
```

180.7. 隐藏敏感信息

从 Camel 2.12 开始提供

默认情况下，Camel enlists MBeans 在 JMX 中，如使用 URI 配置的端点。在此配置中，可能有一些敏感信息，如密码。

通过启用 mask 选项可以隐藏这些信息，如下所示：

使用 Java DSL，您可以通过以下方法打开：

```
// only enable routes when Camel starts
context.getManagementStrategy().getManagementAgent().setMask(true);
```

在 Spring DSL 中：

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
    <jmxAgent id="agent" mask="true"/>
 ...
</camelContext>
```

这将屏蔽带有密码和密码短语等选项的 URI，并使用 xxxxxx 作为替换值。

180.7.1. 声明哪些 JMX 属性和操作到掩码

在 org.apache.camel.api.management.ManagedAttribute 和 org.apache.camel.api.management.ManagedOperation 中，属性 掩码 可以设置为 true，以指示此 JMX 属性/操作的结果应该被屏蔽（如果在 JMX 代理上启用）。

例如，在 camel-core org.apache.camel.api.management.mbean.ManagedEndpointMBean 中的默认管理端点上，我们声明了 EndpointUri JMX 属性被屏蔽：

```
@ManagedAttribute(description = "Endpoint URI", mask = true)
String getEndpointUri();
```

180.8. 另请参阅

- 管理示例
- 为什么我的处理器没有在 JConsole 中显示

第 181 章 JOLT 组件

从 Camel 版本 2.16 开始提供

jolt: 组件允许您使用 JOLT 规格处理 JSON 消息。在进行 JSON 转换为 JSON 转换时，这可能是理想的选择。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jolt</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

181.1. URI 格式

```
jolt:specName[?options]
```

其中 specName 是要调用的规格的 classpath-local URI，或者远程规格的完整 URL （例如：file://folder/myfile.json）。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

181.2. 选项

JOLT 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
allowContextMapAll (producer)	设置上下文映射是否应该允许访问所有详情。默认情况下，只能访问消息正文和标头。可以启用此选项以完全访问当前 Exchange 和 CamelContext。这样做会带来潜在的安全风险，因为这会打开对 CamelContext API 的完整功能的访问。	false	布尔值

Name	描述	默认值	类型
allowTemplateFromHeader (producer)	是否允许使用来自标头的资源模板（默认 false）。**启用此选项具有安全原因。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。**	false	布尔值
转换 (advanced)	明确设置要使用的 Transform。如果没有设置由 transformDsl 指定的 Transform		转换
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

JOLT 端点使用 URI 语法进行配置：

```
jolt:resourceUri
```

使用以下路径和查询参数：

181.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceUri	资源 **所需的** 路径。您可以使用这些协议(classpath 为 default)使用： classpath、file、http、ref 或 bean. classpath、file 和 http 加载资源的前缀。ref 将查找 registry 中的资源。bean 将调用 bean 以供用作资源的方法。对于 bean，您可以在点后指定方法名称，如 bean:myBean.myMethod。		字符串

181.2.2. 查询参数(7 参数)：

Name	描述	默认值	类型
allowContextMapAll (producer)	设置上下文映射是否应该允许访问所有详情。默认情况下，只能访问消息正文和标头。可以启用此选项以完全访问当前 Exchange 和 CamelContext。这样做会带来潜在的安全风险，因为这会打开对 CamelContext API 的完整功能的访问。	false	布尔值

Name	描述	默认值	类型
allowTemplateFromHeader (producer)	是否允许使用来自标头的资源模板（默认 false）。**启用此选项具有安全原因。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。**	false	布尔值
contentCache (producer)	设置是否使用资源内容缓存	false	布尔值
inputType (producer)	指定输入是否假设 JSON 或 JSON 字符串。	Hydrated	JoltInputOutputType
outputType (producer)	指定输出是否应该被假设为 JSON 或 JSON 字符串。	Hydrated	JoltInputOutputType
transformDsl (producer)	指定端点资源的 Transform DSL。如果未指定 Chainr，则将使用 Chainr。	链器	JoltTransformType
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

181.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.jolt.enabled	启用 jolt 组件	true	布尔值
camel.component.jolt.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.jolt.transform	明确设置要使用的 Transform。如果没有设置由 transformDsl 指定的 Transform。选项是一个 com.bazaarvoice.jolt.Transform 类型。		字符串

181.4. SAMPLES

例如，您可以使用类似如下的内容

```
from("activemq:My.Queue").
  to("jolt:com/acme/MyResponse.json");
```

和基于文件的资源：

```
from("activemq:My.Queue").
  to("jolt:file://myfolder/MyResponse.json?contentCache=true").
  to("activemq:Another.Queue");
```

您还可以指定组件应该通过标头动态使用哪些规格，例如：

```
from("direct:in").
  setHeader("CamelJoltResourceUri").constant("path/to/my/spec.json").
  to("jolt:dummy?allowTemplateFromHeader=true");
```

警告

启用 allowTemplateFromHeader 选项具有安全等级。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。

181.5. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 182 章 JPA COMPONENT

从 Camel 版本 1.0 开始提供

jpa 组件可让您使用 EJB 3 的 Java Persistence 架构(langpack)从持久性存储存储和检索 Java 对象，它是一个标准接口层，用于嵌套对象/关系映射(ORM)产品，如 Openrhgs、Hibernate、topLink 等。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jpa</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

182.1. 发送到端点

您可以通过将 Java 实体 Bean 发送到 JPA producer 端点来将其存储在数据库中。In 消息的正文被假定为实体 Bean （即具有 @Entity 注释的 POJO），或实体 Bean 的集合或数组。

如果正文是实体列表，请确保使用 entityType=java.util.ArrayList 作为传递给制作者端点的配置。

如果正文不包含之前列出的类型之一，请在端点前面放置 Message Translator，首先执行必要的转换。

从 Camel 2.19 开始，您还可以将 查询、namedQuery 或 nativeQuery 用于生成者。另外，在 参数中的值 中，您可以使用简单表达式，允许您从消息正文、标头等检索参数值。这些查询可用于使用 SELECT JPQL/SQL 语句检索一组数据，并使用 UPDATE/DELETE JPQL/SQL 语句来执行批量更新/删除。请注意，如果您使用 namedQuery 执行 UPDATE/DELETE，则需要指定 useExecuteUpdate to true，因为 camel don't look look the named 查询与查询和 nativeQuery 不同。

182.2. 从端点消耗

使用 JPA consumer 端点的消息会删除（或更新）数据库中的实体 Bean。这可让您将数据库表用作逻辑队列：消费者从队列中获取消息，然后删除/更新它们，以将其从队列中删除。

如果您不想在处理实体 bean 时删除它（并在路由完成后），您可以在 URI 中指定 consumeDelete=false。这会导致每个轮询处理实体。

如果您改为对实体执行一些更新以将其标记为已处理（例如，将其从将来的查询中排除），然后您可以使用 @Consumed 注解方法，当实体 bean 在被处理时（以及完成路由时）将在您的实体 bean 上调用它。

从 Camel 2.13 开始，您可以使用 @PreConsumed，在被处理前，将在实体 bean 上调用它（之前路由）。

如果您消耗了很多行(100K+)，且遇到 OutOfMemory 问题，您应该将 maximumResults 设置为 sensible 值。

182.3. URI 格式

```
jpa:entityClassName[?options]
```

对于发送到端点，entityClassName 是可选的。如果指定，它可帮助 Type Converter 确保正文正确类型。

为消耗，entityClassName 是必需的。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

182.4. 选项

JPA 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
entityManagerFactory (common)	使用 EntityManagerFactory。强烈建议您配置。		EntityManagerFactory
transactionManager (common)	使用 PlatformTransactionManager 管理事务。		平台交易管理器

Name	描述	默认值	类型
joinTransaction (common)	camel-jpa 组件将默认加入事务。您可以使用此选项关闭此选项，例如，如果您使用 LOCAL_RESOURCE 和 join 事务无法与 JPA 提供程序一起工作。这个选项也可以在 JpaComponent 上全局设置，而不必在所有端点上设置它。	true	布尔值
sharedEntityManager (common)	是否将 Spring 的 SharedEntityManager 用于 consumer/producer。在大多数情况下，joinTransaction 应该被设置为 false，因为这不是 EXTENDED EntityManager。	false	布尔值
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

JPA 端点使用 URI 语法进行配置：

```
jpa:entityType
```

使用以下路径和查询参数：

182.4.1. 路径参数(1 参数)：

Name	描述	默认值	类型
entityType	**必需的** JPA 注解的类以用作实体。		类

182.4.2. 查询参数(42 参数)：

Name	描述	默认值	类型
joinTransaction (common)	camel-jpa 组件将默认加入事务。您可以使用此选项关闭此选项，例如，如果您使用 LOCAL_RESOURCE 和 join 事务无法与 JPA 提供程序一起工作。这个选项也可以在 JpaComponent 上全局设置，而不必在所有端点上设置它。	true	布尔值
maximumResults (common)	设置在 Query 上检索的最大结果数。	-1	int

Name	描述	默认值	类型
namedQuery (common)	使用命名的查询。		字符串
nativeQuery (common)	使用自定义原生查询。在使用原生查询时，您可能还想使用 option resultClass。		字符串
参数 (common)	此键/值映射用于构建查询参数。应该是通用类型 java.util.Map，其中键是给定 JPA 查询的命名参数，并且值是您要为其选择的相应有效值。当它用于生成者时，可以将简单表达式用作参数值。它允许您从消息正文、标头等检索参数值。		Map
PersistenceUnit (common)	**必需** 默认使用的 JPA Persistence 单元。	Camel	字符串
query (common)	使用自定义查询。		字符串
resultClass (common)	定义返回的有效负载（我们将调用 entityManager.createNativeQuery (nativeQuery, resultClass)而不是 entityManager.createNativeQuery (nativeQuery)）。如果没有这个选项，我们将返回一个对象数组。只有在使用数据时，只有与原生查询一起使用时具有影响。		类
sharedEntityManager (common)	是否将 Spring 的 SharedEntityManager 用于 consumer/producer。在大多数情况下，joinTransaction 应该被设置为 false，因为这不是 EXTENDED EntityManager。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
consumeDelete (consumer)	如果为 true，则实体会在被使用后删除；如果为 false，则实体不会被删除。	true	布尔值
consumeLockEntity (consumer)	指定在处理从轮询结果时，是否在每个实体 bean 上设置专用锁定。	true	布尔值
deleteHandler (consumer)	使用自定义 DeleteHandler 在消费者处理交换后删除行		DeleteHandler

Name	描述	默认值	类型
lockModeType (consumer)	要在消费者上配置锁定模式。	PESSIMISTIC_WRITE	LockModeType
maxMessagesPerPoll (consumer)	整数值，定义每个轮询要收集的最大消息数。默认情况下，不设置最大值。可以用来避免在启动服务器时轮询数以千计的消息。设置要禁用的 0 或 negative 值。		int
preDeleteHandler (consumer)	使用自定义 Pre-DeleteHandler 在使用者读取实体后删除行。		DeleteHandler
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
skipLockedEntity (consumer)	要配置是否在锁定时使用 NOWAIT，并静默跳过该实体。	false	布尔值
transacted (consumer)	是否在转换模式下运行消费者，在处理整个批处理时，所有消息都会提交或回滚。默认行为(false)是提交所有之前成功处理的消息，仅回滚最后的失败消息。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
flushOnSend (producer)	在实体 bean 持久化后清除 EntityManager。	true	布尔值
remove (producer)	指明使用 entityManager.remove (entity)。	false	布尔值
useExecuteUpdate (producer)	要配置是否在生成者执行查询时使用 executeUpdate（）。当您使用 INSERT、UPDATE 或 DELETE 语句作为命名查询时，您需要将此选项指定为 'true'。		布尔值

Name	描述	默认值	类型
usePassedInEntityManager (producer)	如果设置为 true，则 Camel 将使用来自标头 JpaConstants.ENTITY_MANAGER 中的 EntityManager，而不是在组件/端点上配置的实体管理器。这允许最终用户控制使用哪个实体管理器。	false	布尔值
usePersist (producer)	指明使用 entityManager.persist (entity)而不是 entityManager.merge (entity)。注意：entityManager.persist (entity)不适用于分离实体（实体管理器必须执行 UPDATE 而不是 INSERT 查询）！	false	布尔值
entityManagerProperties (advanced)	要使用的实体管理器的其他属性。		Map
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service

Name	描述	默认值	类型
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

182.5. SPRING BOOT AUTO-CONFIGURATION

组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
camel.component.jpa.enabled	启用 jpa 组件	true	布尔值
camel.component.jpa.entity-manager-factory	使用 EntityManagerFactory。强烈建议您配置。选项是 javax.persistence.EntityManagerFactory 类型。		字符串
camel.component.jpa.join-transaction	camel-jpa 组件将默认加入事务。您可以使用此选项关闭此选项，例如，如果您使用 LOCAL_RESOURCE 和 join 事务无法与 JPA 提供程序一起工作。这个选项也可以在 JpaComponent 上全局设置，而不必在所有端点上设置它。	true	布尔值
camel.component.jpa.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.jpa.shared-entity-manager	是否将 Spring 的 SharedEntityManager 用于 consumer/producer。在大多数情况下，joinTransaction 应该被设置为 false，因为这不是 EXTENDED EntityManager。	false	布尔值

Name	描述	默认值	类型
camel.component.jpa.transaction-manager	使用 PlatformTransactionManager 管理事务。选项是一个 org.springframework.transaction.PlatformTransactionManager 类型。		字符串

182.6. 消息标头

Camel 在交换中添加以下消息标头：

标头	类型	描述
CamelJpaTemplate	**JpaTemplate**	**从 Camel 2.12 开始，不再支持它：** 用于访问实体 Bean 的**JpaTemplate** 对象。在某些情况下，您需要此对象，例如在类型转换器或执行一些自定义处理时。有关此标头支持的原因，请参阅 CAMEL-5932。
CamelEntityManager	**EntityManager**	Camel 2.12: JPA consumer / Camel 2.12.2: JPA producer:**JpaConsumer** 或 **JpaProducer** 使用的 JPA **EntityManager** 对象。
CamelJpaParameters	**Map<String, Object>**	Camel 2.23: JPA producer： 将查询参数作为 Exchange 标头传递的替代方法。

182.7. 配置 ENTITYMANAGERFACTORY

强烈建议您将 JPA 组件配置为使用特定的 EntityManagerFactory 实例。如果无法这样做，则每个 JpaEndpoint 都会自动创建自己的 EntityManagerFactory 实例，这通常不是您想要的。

例如，您可以实例化一个引用 myEMFactory 实体管理器工厂的 JPA 组件，如下所示：

```
<bean id="jpa" class="org.apache.camel.component.jpa.JpaComponent">
  <property name="entityManagerFactory" ref="myEMFactory"/>
</bean>
```

在 Camel 2.3 中，JpaComponent 将从 Registry 中自动查找 EntityManagerFactory，这意味着您不需要在 JpaComponent 上配置它，如上所示。您只需要这样做，在这种情况下，Camel 会记录

WARN。

182.8. 配置 TRANSACTIONMANAGER

自 Camel 2.3 起，JpaComponent 将会自动从 Registry 中查找 TransactionManager。如果 Camel 找不到任何注册 TransactionManager 实例，它将查找 TransactionTemplate，并尝试从其中提取 TransactionManager。

如果注册表中没有 TransactionTemplate，则 JpaEndpoint 将自动创建自己的 TransactionManager 实例，而这通常不是您想要的。

如果找到了 TransactionManager 的多个实例，Camel 将记录 WARN。在这种情况下，您可能想要实例化和明确配置引用 myTransactionManager 事务管理器的 JPA 组件，如下所示：

```
<bean id="jpa" class="org.apache.camel.component.jpa.JpaComponent">
  <property name="entityManagerFactory" ref="myEMFactory"/>
  <property name="transactionManager" ref="myTransactionManager"/>
</bean>
```

182.9. 使用带有命名查询的消费者

对于只消耗所选实体，您可以使用 consumer.namedQuery URI 查询选项。首先，您必须在 JPA Entity 类中定义命名查询：

```
@Entity
@NamedQuery(name = "step1", query = "select x from MultiSteps x where x.step = 1")
public class MultiSteps {
  ...
}
```

在定义消费者 uri 后，如下所示：

```
from("jpa://org.apache.camel.examples.MultiSteps?consumer.namedQuery=step1")
.to("bean:myBusinessLogic");
```

182.10. 使用带有查询的消费者

对于只消耗所选实体，您可以使用 consumer.query URI 查询选项。您只需要定义查询选项：

```
from("jpa://org.apache.camel.examples.MultiSteps?consumer.query=select o from
org.apache.camel.examples.MultiSteps o where o.step = 1")
.to("bean:myBusinessLogic");
```

182.11. 使用带有原生查询的消费者

对于只消耗所选实体，您可以使用 consumer.nativeQuery URI 查询选项。您只需要定义原生查询选项：

```
from("jpa://org.apache.camel.examples.MultiSteps?consumer.nativeQuery=select * from
MultiSteps where step = 1")
.to("bean:myBusinessLogic");
```

如果使用原生查询选项，您将在消息正文中收到对象数组。

182.12. 使用带有命名查询的制作者

要检索所选实体或执行批量更新/删除，您可以使用 namedQuery URI 查询选项。首先，您必须在 JPA Entity 类中定义命名查询：

```
@Entity
@NamedQuery(name = "step1", query = "select x from MultiSteps x where x.step = 1")
public class MultiSteps {
   ...
}
```

在定义制作者 uri 后，如下所示：

```
from("direct:namedQuery")
.to("jpa://org.apache.camel.examples.MultiSteps?namedQuery=step1");
```

请注意，您需要将 useExecuteUpdate 选项指定为 true，才能将 UPDATE/DELETE 语句指定为命名查询。

182.13. 使用带有查询的制作者

要检索所选实体或执行批量更新/删除，您可以使用 查询 URI 查询选项。您只需要定义查询选项：

```
from("direct:query")
.to("jpa://org.apache.camel.examples.MultiSteps?query=select o from
org.apache.camel.examples.MultiSteps o where o.step = 1");
```

182.14. 使用带有原生查询的制作者

要检索所选实体或执行批量更新/删除，您可以使用 nativeQuery URI 查询选项。您只需要定义原生查询选项：

```
from("direct:nativeQuery")
.to("jpa://org.apache.camel.examples.MultiSteps?
resultClass=org.apache.camel.examples.MultiSteps&nativeQuery=select * from MultiSteps
where step = 1");
```

如果您在没有指定 resultClass 的情况下使用原生查询选项，您将在消息正文中接收对象数组。

182.15. EXAMPLE

有关使用 JPA 将追踪消息存储到数据库中的示例，请参阅 Tracer 示例。

182.16. 使用基于 JPA 的 IDEMPOTENT 存储库

EIP 模式中的 Idempotent Consumer 用于过滤重复的消息。提供了基于 JPA 的幂等存储库。

使用基于 JPA 的幂等存储库：

流程

1. 在 persistence.xml 文件中设置 persistence-unit ：
2. 设置 org.springframework.orm.jpa.JpaTemplate，它由 org.apache.camel.processor.idempotent.jpa.JpaMessageIdRepository 使用：
3. 配置错误格式化宏： snippet: java.lang.IndexOutOfBoundsException: Index: 20, Size: 20
4.

配置幂等存储库：
org.apache.camel.processor.idempotent.jpa.JpaMessageIdRepository ：

5.
在 Spring XML 文件中创建 JPA idempotent 存储库：

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <route id="JpaMessageIdRepositoryTest">
    <from uri="direct:start" />
    <idempotentConsumer messageIdRepositoryRef="jpaStore">
      <header>messageId</header>
      <to uri="mock:result" />
    </idempotentConsumer>
  </route>
</camelContext>
```

在 IDE 中运行此 Camel 组件测试时

如果您在 IDE 中直接运行此组件 的测试，而不是通过 Maven 运行，则您可能会看到类似以下的例外：

```
org.springframework.transaction.CannotCreateTransactionException: Could not open JPA
EntityManager for transaction; nested exception is
<openjpa-2.2.1-r422266:1396819 nonfatal user error>
org.apache.openjpa.persistence.ArgumentException: This configuration disallows runtime
optimization,
but the following listed types were not enhanced at build time or at class load time with a
javaagent: "org.apache.camel.examples.SendEmail".
  at
org.springframework.orm.jpa.JpaTransactionManager.doBegin(JpaTransactionManager.java:4
27)
  at
org.springframework.transaction.support.AbstractPlatformTransactionManager.getTransactio
n(AbstractPlatformTransactionManager.java:371)
  at
org.springframework.transaction.support.TransactionTemplate.execute(TransactionTemplate.
java:127)
  at org.apache.camel.processor.jpa.JpaRouteTest.cleanupRepository(JpaRouteTest.java:96)
  at
org.apache.camel.processor.jpa.JpaRouteTest.createCamelContext(JpaRouteTest.java:67)
  at org.apache.camel.test.junit4.CamelTestSupport.doSetUp(CamelTestSupport.java:238)
  at org.apache.camel.test.junit4.CamelTestSupport.setUp(CamelTestSupport.java:208)
```

这里的问题在于，源是通过 IDE 编译或重新编译，而不是通过 Maven 编译，这会在构建时增强字节代码。要克服这一问题，您需要启用 Open JPA 的动态字节代码增强。例如，假设 Camel 中使用的当前 Opengfortran 版本为 2.2.1，以便在 IDE 中运行测试，您需要将以下参数传递给 JVM：

```
-javaagent:
<path_to_your_local_m2_cache>/org/apache/openjpa/openjpa/2.2.1/openjpa-2.2.1.jar
```

182.17. 在事务上下文中使用连接池

182.17.1. 设置数据源连接池大小

当使用 JpaTransactionManager 时，它需要一个单独的连接来控制事务。因此，您必须配置 JDBC 连接池，以实现至少两个 JDBC 连接的容量限制。

当 camel-jpa 与 transacted （） 和 split （）, multicast （）或 receiptList （）结合使用时，这适用。

- 将数据源连接池大小设置为 至少 2。

182.17.2. 为 Content Enricher 添加方法

当将 EIP 模式中的 Content Enricher 与自定义聚合策略搭配使用时，您必须将 JpaConstants.ENTITY_MANAGE 属性从 newExchange 复制到 oldExchange。

- 向 JpaHelper.copyEntityManagers 添加一个方法来执行复制操作：

```
from("direct:enrich")
   .transacted().enrich("jpa://" + Example.class.getName(), new AggregationStrategy() {
      @Override
      public Exchange aggregate(Exchange oldExchange, Exchange newExchange) {
            JpaHelper.copyEntityManagers(oldExchange, newExchange);
            return newExchange;
         }
         })
      .to("jpa://" + Example.class.getName());
```

182.18. 另请参阅

- 配置 Camel
- 组件

- *端点*
- *开始使用*
- *tracer 示例*

第 183 章 JSON FASTJSON DATAFORMAT

从 Camel 版本 2.20 开始提供

Fastjson 是一个数据格式，它使用 Fastjson 库

```
from("activemq:My.Queue").
  marshal().json(JsonLibrary.Fastjson).
  to("mqseries:Another.Queue");
```

183.1. FASTJSON 选项

JSon Fastjson dataformat 支持 19 个选项，如下所列。

Name	默认值	Java 类型	描述
objectMapper		字符串	在使用 Jackson 时，查找并使用给定 id 的现有 ObjectMapper。
useDefaultObjectMapper	true	布尔值	是否从 registry 中查找和使用默认的 Jackson ObjectMapper。
prettyPrint	false	布尔值	要启用用户的打印输出，请执行以下操作：默认为 false。
程序库	xstream	JsonLibrary	要使用哪个 json 库。
unmarshalTypeName		字符串	取消警报时要使用的 java 类型的类名称
jsonView		类	当将 POJO 聚合到 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。此选项是引用具有 JsonView 注释的类
Include		字符串	如果您要将 pojo 放入 JSON，并且 pojo 有一些带有 null 值的字段。如果您想要跳过这些 null 值，您可以将这个选项设置为 NON_NULL
allowJmsType	false	布尔值	用于 JMS 用户，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。
collectionTypeName		字符串	引用要在要使用的 registry 中的自定义集合类型。这个选项很少被使用，但允许使用不同的集合类型，而不是基于 java.util.Collection 作为默认值。

Name	默认值	Java 类型	描述
useList	false	布尔值	要取消警报到映射列表或 Pojo 列表。
enableJaxbAnnotationModule	false	布尔值	使用 jackson 时是否启用 JAXB 注释模块。启用后，Jackson 可以使用 JAXB 注释。
moduleClassNames		字符串	要使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String 带有 FQN 类名称。可以使用逗号分隔多个类。
moduleRefs		字符串	使用 Camel registry 引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。
enableFeatures		字符串	要在 Jackson com.fasterxml.jackson.databind.ObjectMapper 中启用的功能集。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称
disableFeatures		字符串	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的一组功能。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称
权限		字符串	添加控制在 unmarshal from xml/json 到 Java Bean 期间允许使用哪些 Java 软件包和类 XStream 的权限。必须在此处或全局使用 JVM 系统属性配置权限。权限可以使用加号允许的语法指定，减号是 deny。使用 . 作为前缀来支持通配符。例如，允许 com.foo 和所有子软件包，然后 specfy com.foo.。可以用逗号分隔多个权限，如 com.foo.,-com.foo.bar.MySecretBean。以下默认权限始终被包括： -,java.lang.,java.util.，除非其使用键 org.apache.camel.xstream.permissions 指定 JVM 系统属性来覆盖它。
allowUnmarshallType	false	布尔值	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。只有在需要使用时，才会启用。
timezone		字符串	如果设置，则 Jackson 会在 marshalling/unmarshalling 时使用 Timezone。此选项对其他 Json DataFormat （如 gson、fastjson 和 xstream）没有影响。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

183.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 20 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.json-fastjson.allow-jms-type	用于 JMS 用户，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。	false	布尔值
camel.dataformat.json-fastjson.allow-unmarshall-type	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。只有在需要使用时，才会启用。	false	布尔值
camel.dataformat.json-fastjson.collection-type-name	引用要在要使用的 registry 中的自定义集合类型。这个选项很少被使用，但允许使用不同的集合类型，而不是基于 java.util.Collection 作为默认值。		字符串
camel.dataformat.json-fastjson.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.json-fastjson.disable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的一组功能。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称		字符串
camel.dataformat.json-fastjson.enable-features	要在 Jackson com.fasterxml.jackson.databind.ObjectMapper 中启用的功能集。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称		字符串
camel.dataformat.json-fastjson.enable-jaxb-annotation-module	使用 jackson 时是否启用 JAXB 注释模块。启用后，Jackson 可以使用 JAXB 注释。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.json-fastjson.enabled	是否启用 json-fastjson 数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.json-fastjson.include	如果您要将 pojo 放入 JSON，并且 pojo 有一些带有 null 值的字段。如果您想要跳过这些 null 值，您可以将这个选项设置为 NON_NULL		字符串
camel.dataformat.json-fastjson.json-view	当将 POJO 聚合到 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。此选项是引用具有 JsonView 注释的类		类
camel.dataformat.json-fastjson.library	要使用哪个 json 库。		JsonLibrary
camel.dataformat.json-fastjson.module-class-names	要使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String 带有 FQN 类名称。可以使用逗号分隔多个类。		字符串
camel.dataformat.json-fastjson.module-refs	使用 Camel registry 引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。		字符串
camel.dataformat.json-fastjson.object-mapper	在使用 Jackson 时，查找并使用给定 id 的现有 ObjectMapper。		字符串
camel.dataformat.json-fastjson.permissions	添加控制在 unmarshal from xml/json 到 Java Bean 期间允许使用哪些 Java 软件包和类 XStream 的权限。必须在此处或全局使用 JVM 系统属性配置权限。权限可以使用加号允许的语法指定，减号是 deny。使用 . 作为前缀来支持通配符。例如，允许 com.foo 和所有子软件包，然后 specfy com.foo.。可以用逗号分隔多个权限，如 com.foo.,-com.foo.bar.MySecretBean。以下默认权限始终被包括： -,java.lang.,java.util.，除非其使用键 org.apache.camel.xstream.permissions 指定 JVM 系统属性来覆盖它。		字符串
camel.dataformat.json-fastjson.pretty-print	要启用用户的打印输出，请执行以下操作：默认为 false。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.json-fastjson.timezone	如果设置，则 Jackson 会在 marshalling/unmarshalling 时使用 Timezone。此选项对其他 Json DataFormat （如 gson、fastjson 和 xstream）没有影响。		字符串
camel.dataformat.json-fastjson.unmarshal-type-name	取消警报时要使用的 java 类型的类名称		字符串
camel.dataformat.json-fastjson.use-default-object-mapper	是否从 registry 中查找和使用默认的 Jackson ObjectMapper。	true	布尔值
camel.dataformat.json-fastjson.use-list	要取消警报到映射列表或 Pojo 列表。	false	布尔值

183.3. 依赖项

要在 camel 路由中使用 Fastjson，您需要添加对实现此数据格式的 camel-fastjson 的依赖关系。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-fastjson</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

第 184 章 JSON GSON DATAFORMAT

从 Camel 版本 2.10 开始提供

gson 是一个数据格式，它使用 Gson Library

```
from("activemq:My.Queue").
  marshal().json(JsonLibrary.Gson).
  to("mqseries:Another.Queue");
```

184.1. GSON 选项

JSon GSon dataformat支持 19 个选项，如下所列。

Name	默认值	Java 类型	描述
objectMapper		字符串	在使用 Jackson 时，查找并使用给定 id 的现有 ObjectMapper。
useDefaultObjectMapper	true	布尔值	是否从 registry 中查找和使用默认的 Jackson ObjectMapper。
prettyPrint	false	布尔值	要启用用户的打印输出，请执行以下操作：默认为 false。
程序库	xstream	JsonLibrary	要使用哪个 json 库。
unmarshalTypeName		字符串	取消警报时要使用的 java 类型的类名称
jsonView		类	当将 POJO 聚合到 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。此选项是引用具有 JsonView 注释的类
Include		字符串	如果您要将 pojo 放入 JSON，并且 pojo 有一些带有 null 值的字段。如果您想要跳过这些 null 值，您可以将这个选项设置为 NON_NULL
allowJmsType	false	布尔值	用于 JMS 用户，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。
collectionTypeName		字符串	引用要在要使用的 registry 中的自定义集合类型。这个选项很少被使用，但允许使用不同的集合类型，而不是基于 java.util.Collection 作为默认值。

Name	默认值	Java 类型	描述
useList	false	布尔值	要取消警报到映射列表或 Pojo 列表。
enableJaxbAnnotationModule	false	布尔值	使用 jackson 时是否启用 JAXB 注释模块。启用后，Jackson 可以使用 JAXB 注释。
moduleClassNames		字符串	要使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String 带有 FQN 类名称。可以使用逗号分隔多个类。
moduleRefs		字符串	使用 Camel registry 引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。
enableFeatures		字符串	要在 Jackson com.fasterxml.jackson.databind.ObjectMapper 中启用的功能集。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称
disableFeatures		字符串	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的一组功能。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称
权限		字符串	添加控制在 unmarshal from xml/json 到 Java Bean 期间允许使用哪些 Java 软件包和类 XStream 的权限。必须在此处或全局使用 JVM 系统属性配置权限。权限可以使用加号允许的语法指定，减号是 deny。使用 . 作为前缀来支持通配符。例如，允许 com.foo 和所有子软件包，然后 specfy com.foo.。可以用逗号分隔多个权限，如 com.foo.,-com.foo.bar.MySecretBean。以下默认权限始终被包括： -,java.lang.,java.util.，除非其使用键 org.apache.camel.xstream.permissions 指定 JVM 系统属性来覆盖它。
allowUnmarshallType	false	布尔值	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。只有在需要使用时，才会启用。
timezone		字符串	如果设置，则 Jackson 会在 marshalling/unmarshalling 时使用 Timezone。此选项对其他 Json DataFormat （如 gson、fastjson 和 xstream）没有影响。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

184.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 20 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.json-gson.allow-jms-type	用于 JMS 用户，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。	false	布尔值
camel.dataformat.json-gson.allow-unmarshall-type	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。只有在需要使用时，才会启用。	false	布尔值
camel.dataformat.json-gson.collection-type-name	引用要在要使用的 registry 中的自定义集合类型。这个选项很少被使用，但允许使用不同的集合类型，而不是基于 java.util.Collection 作为默认值。		字符串
camel.dataformat.json-gson.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.json-gson.disable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的一组功能。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称		字符串
camel.dataformat.json-gson.enable-features	要在 Jackson com.fasterxml.jackson.databind.ObjectMapper 中启用的功能集。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称		字符串
camel.dataformat.json-gson.enable-jaxb-annotation-module	使用 jackson 时是否启用 JAXB 注释模块。启用后，Jackson 可以使用 JAXB 注释。	false	布尔值
camel.dataformat.json-gson.enabled	启用 json-gson dataformat	true	布尔值

Name	描述	默认值	类型
camel.dataformat.json-gson.include	如果您要将 pojo 放入 JSON，并且 pojo 有一些带有 null 值的字段。如果您想要跳过这些 null 值，您可以将这个选项设置为 NON_NULL		字符串
camel.dataformat.json-gson.json-view	当将 POJO 聚合到 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。此选项是引用具有 JsonView 注释的类		类
camel.dataformat.json-gson.library	要使用哪个 json 库。		JsonLibrary
camel.dataformat.json-gson.module-class-names	要使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String 带有 FQN 类名称。可以使用逗号分隔多个类。		字符串
camel.dataformat.json-gson.module-refs	使用 Camel registry 引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。		字符串
camel.dataformat.json-gson.object-mapper	在使用 Jackson 时，查找并使用给定 id 的现有 ObjectMapper。		字符串
camel.dataformat.json-gson.permissions	添加控制在 unmarshal from xml/json 到 Java Bean 期间允许使用哪些 Java 软件包和类 XStream 的权限。必须在此处或全局使用 JVM 系统属性配置权限。权限可以使用加号允许的语法指定，减号是 deny。使用 . 作为前缀来支持通配符。例如，允许 com.foo 和所有子软件包，然后 specfy com.foo.。可以用逗号分隔多个权限，如 com.foo.,-com.foo.bar.MySecretBean。以下默认权限始终被包括： -,java.lang.,java.util.，除非其使用键 org.apache.camel.xstream.permissions 指定 JVM 系统属性来覆盖它。		字符串
camel.dataformat.json-gson.pretty-print	要启用用户的打印输出，请执行以下操作：默认为 false。	false	布尔值
camel.dataformat.json-gson.timezone	如果设置，则 Jackson 会在 marshalling/unmarshalling 时使用 Timezone。此选项对其他 Json DataFormat （如 gson、fastjson 和 xstream）没有影响。		字符串
camel.dataformat.json-gson.unmarshal-type-name	取消警报时要使用的 java 类型的类名称		字符串

Name	描述	默认值	类型
camel.dataformat.json-gson.use-default-object-mapper	是否从 registry 中查找和使用默认的 Jackson ObjectMapper。	true	布尔值
camel.dataformat.json-gson.use-list	要取消警报到映射列表或 Pojo 列表。	false	布尔值

184.3. 依赖项

要在 camel 路由中使用 Gson，您需要添加实现此数据格式的 camel-gson 的依赖项。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-gson</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

第 185 章 JSON JACKSON DATAFORMAT

从 Camel 版本 2.0 开始提供

Jackson 是一个数据格式，它使用 Jackson 库

```
from("activemq:My.Queue").
  marshal().json(JsonLibrary.Jackson).
  to("mqseries:Another.Queue");
```

185.1. JACKSON 选项

JSon Jackson dataformat支持 19 个选项，如下所列。

Name	默认值	Java 类型	描述
objectMapper		字符串	在使用 Jackson 时，查找并使用给定 id 的现有 ObjectMapper。
useDefaultObjectMapper	true	布尔值	是否从 registry 中查找和使用默认的 Jackson ObjectMapper。
prettyPrint	false	布尔值	要启用用户的打印输出，请执行以下操作：默认为 false。
程序库	xstream	JsonLibrary	要使用哪个 json 库。
unmarshalTypeName		字符串	取消警报时要使用的 java 类型的类名称
jsonView		类	当将 POJO 聚合到 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。此选项是引用具有 JsonView 注释的类
Include		字符串	如果您要将 pojo 放入 JSON，并且 pojo 有一些带有 null 值的字段。如果您想要跳过这些 null 值，您可以将这个选项设置为 NON_NULL
allowJmsType	false	布尔值	用于 JMS 用户，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。
collectionTypeName		字符串	引用要在要使用的 registry 中的自定义集合类型。这个选项很少被使用，但允许使用不同的集合类型，而不是基于 java.util.Collection 作为默认值。

Name	默认值	Java 类型	描述
useList	false	布尔值	要取消警报到映射列表或 Pojo 列表。
enableJaxbAnnotationModule	false	布尔值	使用 jackson 时是否启用 JAXB 注释模块。启用后，Jackson 可以使用 JAXB 注释。
moduleClassNames		字符串	要使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String 带有 FQN 类名称。可以使用逗号分隔多个类。
moduleRefs		字符串	使用 Camel registry 引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。
enableFeatures		字符串	要在 Jackson com.fasterxml.jackson.databind.ObjectMapper 中启用的功能集。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称
disableFeatures		字符串	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的一组功能。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称
权限		字符串	添加控制在 unmarshal from xml/json 到 Java Bean 期间允许使用哪些 Java 软件包和类 XStream 的权限。必须在此处或全局使用 JVM 系统属性配置权限。权限可以使用加号允许的语法指定，减号是 deny。使用 . 作为前缀来支持通配符。例如，允许 com.foo 和所有子软件包，然后 specfy com.foo.。可以用逗号分隔多个权限，如 com.foo.,-com.foo.bar.MySecretBean。以下默认权限始终被包括： -,java.lang.,java.util.，除非其使用键 org.apache.camel.xstream.permissions 指定 JVM 系统属性来覆盖它。
allowUnmarshallType	false	布尔值	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。只有在需要使用时，才会启用。
timezone		字符串	如果设置，则 Jackson 会在 marshalling/unmarshalling 时使用 Timezone。此选项对其他 Json DataFormat （如 gson、fastjson 和 xstream）没有影响。
contentTypeHeader	true	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

185.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 20 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.json-jackson.allow-jms-type	用于 JMS 用户，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。	false	布尔值
camel.dataformat.json-jackson.allow-unmarshall-type	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。只有在需要使用时，才会启用。	false	布尔值
camel.dataformat.json-jackson.collection-type-name	引用要在要使用的 registry 中的自定义集合类型。这个选项很少被使用，但允许使用不同的集合类型，而不是基于 java.util.Collection 作为默认值。		字符串
camel.dataformat.json-jackson.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.json-jackson.disable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的一组功能。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称		字符串
camel.dataformat.json-jackson.enable-features	要在 Jackson com.fasterxml.jackson.databind.ObjectMapper 中启用的功能集。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称		字符串
camel.dataformat.json-jackson.enable-jaxb-annotation-module	使用 jackson 时是否启用 JAXB 注释模块。启用后，Jackson 可以使用 JAXB 注释。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.json-jackson.enabled	启用 json-jackson dataformat	true	布尔值
camel.dataformat.json-jackson.include	如果您要将 pojo 放入 JSON，并且 pojo 有一些带有 null 值的字段。如果您想要跳过这些 null 值，您可以将这个选项设置为 NON_NULL		字符串
camel.dataformat.json-jackson.json-view	当将 POJO 聚合到 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。此选项是引用具有 JsonView 注释的类		类
camel.dataformat.json-jackson.library	要使用哪个 json 库。		JsonLibrary
camel.dataformat.json-jackson.module-class-names	要使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String 带有 FQN 类名称。可以使用逗号分隔多个类。		字符串
camel.dataformat.json-jackson.module-refs	使用 Camel registry 引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。		字符串
camel.dataformat.json-jackson.object-mapper	在使用 Jackson 时，查找并使用给定 id 的现有 ObjectMapper。		字符串
camel.dataformat.json-jackson.permissions	添加控制在 unmarshal from xml/json 到 Java Bean 期间允许使用哪些 Java 软件包和类 XStream 的权限。必须在此处或全局使用 JVM 系统属性配置权限。权限可以使用加号允许的语法指定，减号是 deny。使用 . 作为前缀来支持通配符。例如，允许 com.foo 和所有子软件包，然后 specfy com.foo.。可以用逗号分隔多个权限，如 com.foo.,-com.foo.bar.MySecretBean。以下默认权限始终被包括： -,java.lang.,java.util.，除非其使用键 org.apache.camel.xstream.permissions 指定 JVM 系统属性来覆盖它。		字符串
camel.dataformat.json-jackson.pretty-print	要启用用户的打印输出，请执行以下操作：默认为 false。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.json-jackson.timezone	如果设置，则 Jackson 会在 marshalling/unmarshalling 时使用 Timezone。此选项对其他 Json DataFormat （如 gson、fastjson 和 xstream）没有影响。		字符串
camel.dataformat.json-jackson.unmarshal-type-name	取消警报时要使用的 java 类型的类名称		字符串
camel.dataformat.json-jackson.use-default-object-mapper	是否从 registry 中查找和使用默认的 Jackson ObjectMapper。	true	布尔值
camel.dataformat.json-jackson.use-list	要取消警报到映射列表或 Pojo 列表。	false	布尔值

185.3. 使用自定义 OBJECTMAPPER

如果需要更多控制映射配置，您可以将 JacksonDataFormat 配置为使用自定义 ObjectMapper。

如果您在 registry 中设置 单个对象 映射程序，则 Camel 将自动查找并使用此对象映射 程序。例如，如果您使用 Spring Boot，如果启用了 Spring MVC，则 Spring Boot 可为您提供默认的 ObjectMapper。这将允许 Camel 检测 Spring Boot bean registry 中是否有 ObjectMapper 类类型的 bean，然后使用它。当发生这种情况时，您应该从 Camel 设置 INFO 日志记录。

185.4. 依赖项

要在 camel 路由中使用 Jackson，您需要添加对实现此数据格式的 camel-jackson 的依赖关系。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jackson</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

185.5. JACKSON OBJECTMAPPER

185.5.1. 什么是对象映射？

jackson 提供了使用 com.fasterxml.jackson.databind.ObjectMapper 类序列化 Java 对象的机制。例如，您可以使用 ObjectMapper 序列化 MyClass java 对象，如下所示：

```
ObjectMapper objectMapper = new ObjectMapper();
MyClass myobject = new MyClass("foo", "bar");
objectMapper.writeValue(new File("myobject.json"), myobject);
```

对象 myobject 将被序列化为 JSON 格式并写入文件 myobject.json (Jackson 也支持转换为 XML 和 YAML 格式)。

要反序列化文件的 JSON 内容 myobject.json，您可以调用 ObjectMapper，如下所示：

```
ObjectMapper objectMapper = new ObjectMapper();
MyClass myobject = objectMapper.readValue(new File("myobject.json"), MyClass.class);
```

请注意，接收器需要提前知道类的类型，并且必须指定 type MyClass.class，作为 readValue （） 的第二个参数。

185.5.2. 什么是 polymorphic 对象映射？

在某些情况下，序列化对象的接收器无法预先知道对象类型。例如，这适用于 polymorphic 对象数组的情况。考虑抽象类型、Shape、子类型、Triangle、Square （等等）：

```
package com.example;
...
public abstract class Shape {
}

public class Triangle extends Shape {
  ...
}

public class Square extends Shape {
  ...
}

public class ListOfShape {
  public List<Shape> shapes;
  ...
}
```

-

您可以实例化并序列化一个表单列表(ListOfShape)，如下所示：

```
ObjectMapper objectMapper = new ObjectMapper();

ListOfShape shapeList = new ListOfShape();
shapeList.shapes = new ArrayList<Shape>();
shapeList.shapes.add(new Triangle());
shapeList.shapes.add(new Square());

String serialized = objectMapper.writeValueAsString(shapeList);
```

但是，在接收器一侧出现问题。您可以将这个类型指定为 **readValue （） 的第二个参数来告知接收器预期 ListOfShape** 对象：

```
MyClass myobject = objectMapper.readValue(serialized, ListOfShape.class);
ObjectMapper objectMapper = new ObjectMapper();
```

但是，接收器无法知道列表的第一个元素是 **Triangle，第二个元素是 Square。要临时解决这个问题，您需要启用 polymorphic** 对象映射， 如下一节所述。

185.5.3. 如何启用 polymorphic 对象映射

polymorphic 对象映射是一种机制，可通过在序列化数组中提供额外的元数据来识别阵列中对象的类型，从而序列化和反序列化数组。

重要

polymorphic 对象映射导致了固有的安全风险，因为该机制允许 发送者 选择哪个类实例化，从而形成发送方的攻击基础。红帽对 **FasterXML Jackson** 库的分发具有白名单机制，它提供额外的保护级别。您必须确保使用红帽的 **jackson-databind** 库发布（由 **Fuse** 版本 **7.7** 及更新版本提供）以获取这个额外的保护层。如需了解更多详细信息，请参阅 第 185.5.5 节 “polymorphic deserialization 的安全风险”。

要使接收器可以反序列化阵列中的对象，需要在序列化数据中提供类型元数据。默认情况下，**Jackson** 不会对序列化对象的任何类型元数据进行编码，因此您需要编写一些额外的代码来启用此功能。

要启用 **polymorphic** 对象映射，请执行以下步骤（使用 **ListOfShape** 作为示例）：

1. 对于可以列表元素的每个类(**Shape的子类**)，给类标上 **@JsonTypeInfo**，如下所示：

```
@JsonTypeInfo(use=JsonTypeInfo.Id.CLASS, include=JsonTypeInfo.As.PROPERTY)
public class Triangle extends Shape {
  ...
}
```

2. 当 **Triangle** 类被序列化为 **JSON** 格式时，它的格式如下：

```
{"@class":"com.example.Triangle", "property1":"value1", "property2":"value2", ...}
```

3. 接收器必须配置为允许对 **Triangle**、**Square** 和其他格式类进行反序列化，方法是将这些类添加到反序列化白名单中。要配置白名单，请将 **jackson.deserialization.whitelist.packages** 系统属性设置为以逗号分隔的类和软件包列表。例如，要允许对 **Triangle**、**Square** 类进行反序列化，请按如下所示设置系统属性：

```
-Djackson.deserialization.whitelist.packages=com.example.Triangle,com.example.Square
```

另外，您可以设置系统属性来允许整个 **com.example** 软件包：

```
-Djackson.deserialization.whitelist.packages=com.example
```

注意

此白名单机制仅适用于红帽对 **jackson-databind** 库的分发。标准 **jackson-databind** 库使用黑名单机制，每次发现潜在的新 **gadget** 类时都需要更新该机制。

185.5.4. polymorphic deserialization 的默认映射

如果给定的 **Java** 类 **com.example.MyClass** 未列入白名单，则仍有可能对类序列化实例，但在接收端，将使用通用默认映射来反序列化实例。

当在 **Jackson** 中启用 **polymorphic** 对象映射时，有几个替代方法对对象进行编码：

- 使用 **@JsonTypeInfo (use=JsonTypeInfo.Id.CLASS, include=JsonTypeInfo.As.PROPERTY)**：

```
{"@class":"com.example.MyClass", "property1":"value1", "property2":"value2", ...}
```

在这种情况下，实例将反序列化到带有 属性的对象。

- 使用 @JsonTypeInfo (use=JsonTypeInfo.Id.CLASS, include=JsonTypeInfo.As.WRAPPER_ARRAY):

```
["com.example.MyClass", {"property1":"value1", "property2":"value2", ...}]
```

在这种情况下，实例将反序列化到一个 JSON 数组，其中包含两个字段：

  - 带有值 com.example.MyClass的字符串
  - 具有两个 （或更多）属性的对象

- 使用 @JsonTypeInfo (use=JsonTypeInfo.Id.CLASS, include=JsonTypeInfo.As.WRAPPER_OBJECT):

```
{"com.example.MyClass":{"property1":"value1", "property2":"value2", ...}}
```

在这种情况下，实例将反序列化为带有单个字段 com.example.MyClass 的 JSON 映射，值是 Object with two （或更多）属性。

185.5.5. polymorphic deserialization 的安全风险

使用 FasterXML jackson-databind 库通过反序列化 JSON 内容来实例化 Java 对象的应用程序可能会受到 远程代码执行攻击的影响。此漏洞不是自动的，但如果您采取适当的缓解方案步骤，可以避免此漏洞。

在攻击成为可能前，至少需要满足以下先决条件：

1. 您已启用了 polymorphic 类型处理，以便在 jackson-databind 中反序列化 JSON 内容。在 Jackson JSON 中启用多语言类型处理有两种替代方法：
   a. 结合使用 @JsonTypeInfo 和 @JsonSubTypes 注释。

b.

通过调用 ObjectMapper.enableDefaultTyping （） 方法。这个选项特别危险，因为它可以在全局范围内启用多形键入。

2.

Java 类路径中 有一个或多个 gadget 类。gadget 类被定义为执行敏感（可能被利用）操作的任何类，作为执行构造器或集合方法（这是在反序列化期间可调用的方法）的副作用。

3.

Java 类路径中的一个或多个 gadget 类尚未由当前版本的 jackson-databind 列入黑名单。如果您使用 jackson-databind 库的标准发行版，则 Jackson JSON 库维护的 gadget 黑名单是针对远程代码执行漏洞的最后一行。

4.

（红帽仅发布 jackson-databind 库） 您可以将其中一个 gadget 类明确添加到接收方的反序列化白名单中（通过设置 jackson.deserialization.whitelist.packages 系统属性）。因为您不太可能这样做，因此白名单机制默认对所有 gadget 类提供有效的保护。

第 186 章 JSON JOHNZON DATAFORMAT

从 Camel 版本 2.18 开始提供

Johnzon 是一个数据格式，它使用 Johnzon Library

```
from("activemq:My.Queue").
  marshal().json(JsonLibrary.Johnzon).
  to("mqseries:Another.Queue");
```

186.1. JOHNZON 选项

JSon Johnzon dataformat支持 19 个选项，如下所列。

Name	默认值	Java 类型	描述
objectMapper		字符串	在使用 Jackson 时，查找并使用给定 id 的现有 ObjectMapper。
useDefaultObjectMapper	true	布尔值	是否从 registry 中查找和使用默认的 Jackson ObjectMapper。
prettyPrint	false	布尔值	要启用用户的打印输出，请执行以下操作：默认为 false。
程序库	xstream	JsonLibrary	要使用哪个 json 库。
unmarshalTypeName		字符串	取消警报时要使用的 java 类型的类名称
jsonView		类	当将 POJO 聚合到 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。此选项是引用具有 JsonView 注释的类
Include		字符串	如果您要将 pojo 放入 JSON，并且 pojo 有一些带有 null 值的字段。如果您想要跳过这些 null 值，您可以将这个选项设置为 NON_NULL
allowJmsType	false	布尔值	用于 JMS 用户，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。
collectionTypeName		字符串	引用要在要使用的 registry 中的自定义集合类型。这个选项很少被使用，但允许使用不同的集合类型，而不是基于 java.util.Collection 作为默认值。

Name	默认值	Java 类型	描述
useList	false	布尔值	要取消警报到映射列表或 Pojo 列表。
enableJaxbAnnotationModule	false	布尔值	使用 jackson 时是否启用 JAXB 注释模块。启用后，Jackson 可以使用 JAXB 注释。
moduleClassNames		字符串	要使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String 带有 FQN 类名称。可以使用逗号分隔多个类。
moduleRefs		字符串	使用 Camel registry 引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。
enableFeatures		字符串	要在 Jackson com.fasterxml.jackson.databind.ObjectMapper 中启用的功能集。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称
disableFeatures		字符串	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的一组功能。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称
权限		字符串	添加控制在 unmarshal from xml/json 到 Java Bean 期间允许使用哪些 Java 软件包和类 XStream 的权限。必须在此处或全局使用 JVM 系统属性配置权限。权限可以使用加号允许的语法指定，减号是 deny。使用 . 作为前缀来支持通配符。例如，允许 com.foo 和所有子软件包，然后 specfy com.foo.。可以用逗号分隔多个权限，如 com.foo.,-com.foo.bar.MySecretBean。以下默认权限始终被包括： -,java.lang.,java.util.，除非其使用键 org.apache.camel.xstream.permissions 指定 JVM 系统属性来覆盖它。
allowUnmarshallType	false	布尔值	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。只有在需要使用时，才会启用。
timezone		字符串	如果设置，则 Jackson 会在 marshalling/unmarshalling 时使用 Timezone。此选项对其他 Json DataFormat （如 gson、fastjson 和 xstream）没有影响。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

186.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 20 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.json-johnzon.allow-jms-type	用于 JMS 用户，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。	false	布尔值
camel.dataformat.json-johnzon.allow-unmarshall-type	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。只有在需要使用时，才会启用。	false	布尔值
camel.dataformat.json-johnzon.collection-type-name	引用要在要使用的 registry 中的自定义集合类型。这个选项很少被使用，但允许使用不同的集合类型，而不是基于 java.util.Collection 作为默认值。		字符串
camel.dataformat.json-johnzon.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.json-johnzon.disable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的一组功能。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称		字符串
camel.dataformat.json-johnzon.enable-features	要在 Jackson com.fasterxml.jackson.databind.ObjectMapper 中启用的功能集。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称		字符串
camel.dataformat.json-johnzon.enable-jaxb-annotation-module	使用 jackson 时是否启用 JAXB 注释模块。启用后，Jackson 可以使用 JAXB 注释。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.json-johnzon.enabled	启用 json-johnzon dataformat	true	布尔值
camel.dataformat.json-johnzon.include	如果您要将 pojo 放入 JSON，并且 pojo 有一些带有 null 值的字段。如果您想要跳过这些 null 值，您可以将这个选项设置为 NON_NULL		字符串
camel.dataformat.json-johnzon.json-view	当将 POJO 聚合到 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。此选项是引用具有 JsonView 注释的类		类
camel.dataformat.json-johnzon.library	要使用哪个 json 库。		JsonLibrary
camel.dataformat.json-johnzon.module-class-names	要使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String 带有 FQN 类名称。可以使用逗号分隔多个类。		字符串
camel.dataformat.json-johnzon.module-refs	使用 Camel registry 引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。		字符串
camel.dataformat.json-johnzon.object-mapper	在使用 Jackson 时，查找并使用给定 id 的现有 ObjectMapper。		字符串
camel.dataformat.json-johnzon.permissions	添加控制在 unmarshal from xml/json 到 Java Bean 期间允许使用哪些 Java 软件包和类 XStream 的权限。必须在此处或全局使用 JVM 系统属性配置权限。权限可以使用加号允许的语法指定，减号是 deny。使用 . 作为前缀来支持通配符。例如，允许 com.foo 和所有子软件包，然后 specfy com.foo.。可以用逗号分隔多个权限，如 com.foo.,-com.foo.bar.MySecretBean。以下默认权限始终被包括： -,java.lang.,java.util.，除非其使用键 org.apache.camel.xstream.permissions 指定 JVM 系统属性来覆盖它。		字符串
camel.dataformat.json-johnzon.pretty-print	要启用用户的打印输出，请执行以下操作：默认为 false。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.json-johnzon.timezone	如果设置，则 Jackson 会在 marshalling/unmarshalling 时使用 Timezone。此选项对其他 Json DataFormat （如 gson、fastjson 和 xstream）没有影响。		字符串
camel.dataformat.json-johnzon.unmarshal-type-name	取消警报时要使用的 java 类型的类名称		字符串
camel.dataformat.json-johnzon.use-default-object-mapper	是否从 registry 中查找和使用默认的 Jackson ObjectMapper。	true	布尔值
camel.dataformat.json-johnzon.use-list	要取消警报到映射列表或 Pojo 列表。	false	布尔值

186.3. 依赖项

要在 camel 路由中使用 Johnzon，您需要添加对实现此数据格式的 camel-johnzon 的依赖。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-johnzon</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

第 187 章 JSON 架构验证器组件

从 Camel 版本 2.20 开始提供

JSON Schema Validator 组件使用 NetworkNT JSON Schema 库 (https://github.com/networknt/json-schema-validator)对 JSON 架构 v4 草案执行消息正文的 bean 验证。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-json-validator</artifactId>
  <version>x.y.z</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

187.1. URI 格式

```
json-validator:resourceUri[?options]
```

其中 resourceUri 是 classpath 上本地资源或指向远程资源或资源的完整 URL，其中包含要进行验证的 JSON 架构。

187.2. URI 选项

JSON Schema Validator 组件没有选项。

JSON Schema Validator 端点使用 URI 语法进行配置：

```
json-validator:resourceUri
```

使用以下路径和查询参数：

187.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceUri	资源 **所需的** 路径。您可以使用这些协议(classpath 为 default)使用： classpath、file、http、ref 或 bean. classpath、file 和 http 加载资源的前缀。ref 将查找 registry 中的资源。bean 将调用 bean 以供用作资源的方法。对于 bean，您可以在点后指定方法名称，如 bean:myBean.myMethod。		字符串

187.2.2. 查询参数(7 参数)：

Name	描述	默认值	类型
contentCache (producer)	设置是否使用资源内容缓存	false	布尔值
failOnNullBody (producer)	如果没有正文，是否失败。	true	布尔值
failOnNullHeader (producer)	在针对标头验证时，是否没有标头。	true	布尔值
headerName (producer)	以针对标头而不是邮件正文进行验证。		字符串
errorHandler (advanced)	使用自定义 ValidatorErrorHandler。默认错误处理程序捕获错误并抛出异常。		JsonValidatorErro r Handler
schemaLoader (advanced)	使用自定义模式加载程序来添加自定义格式验证。默认实施将创建一个模式加载程序，它带有 v4 支持草案。		JsonSchemaLoad er
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

187.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component .json-validator.enabled	是否启用 json-validator 组件的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.component.json-validator.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

187.4. EXAMPLE

假设我们有以下 JSON 架构

myschema.json

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "my-schema",
  "properties": {
    "id": {
      "default": 1,
      "description": "An explanation about the purpose of this instance.",
      "id": "/properties/id",
      "title": "The id schema",
      "type": "integer"
    },
    "name": {
      "default": "A green door",
      "description": "An explanation about the purpose of this instance.",
      "id": "/properties/name",
      "title": "The name schema",
      "type": "string"
    },
    "price": {
      "default": 12.5,
      "description": "An explanation about the purpose of this instance.",
      "id": "/properties/price",
      "title": "The price schema",
      "type": "number"
    }
  },
  "required": [
    "name",
    "id",
    "price"
  ],
  "type": "object"
}
```

我们可以使用以下 Camel 路由来验证传入的 JSON，其中 myschema.json 从 classpath 加载。

```
from("direct:start")
  .to("json-validator:myschema.json")
  .to("mock:end")
```

第 188 章 JSON XSTREAM DATAFORMAT

从 Camel 版本 2.0 开始提供

Xstream 是一个 Data Format，它使用 XStream 库来聚合和来自 XML 的 Java 对象。

要在 camel 路由中使用 XStream，您需要添加对实现此数据格式的 camel-xstream 的依赖关系。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-xstream</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

188.1. 选项

JSon XStream dataformat 支持 19 个选项，如下所列。

Name	默认值	Java 类型	描述
objectMapper		字符串	在使用 Jackson 时，查找并使用给定 id 的现有 ObjectMapper。
useDefaultObjectMapper	true	布尔值	是否从 registry 中查找和使用默认的 Jackson ObjectMapper。
prettyPrint	false	布尔值	要启用用户的打印输出，请执行以下操作：默认为 false。
程序库	xstream	JsonLibrary	要使用哪个 json 库。
unmarshalTypeName		字符串	取消警报时要使用的 java 类型的类名称
jsonView		类	当将 POJO 聚合到 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。此选项是引用具有 JsonView 注释的类

Name	默认值	Java 类型	描述
Include		字符串	如果您要将 pojo 放入 JSON，并且 pojo 有一些带有 null 值的字段。如果您想要跳过这些 null 值，您可以将这个选项设置为 NON_NULL
allowJmsType	false	布尔值	用于 JMS 用户，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。
collectionTypeName		字符串	引用要在要使用的 registry 中的自定义集合类型。这个选项很少被使用，但允许使用不同的集合类型，而不是基于 java.util.Collection 作为默认值。
useList	false	布尔值	要取消警报到映射列表或 Pojo 列表。
enableJaxbAnnotationModule	false	布尔值	使用 jackson 时是否启用 JAXB 注释模块。启用后，Jackson 可以使用 JAXB 注释。
moduleClassNames		字符串	要使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String 带有 FQN 类名称。可以使用逗号分隔多个类。
moduleRefs		字符串	使用 Camel registry 引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。
enableFeatures		字符串	要在 Jackson com.fasterxml.jackson.databind.ObjectMapper 中启用的功能集。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称
disableFeatures		字符串	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的一组功能。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称
权限		字符串	添加控制在 unmarshal from xml/json 到 Java Bean 期间允许使用哪些 Java 软件包和类 XStream 的权限。必须在此处或全局使用 JVM 系统属性配置权限。权限可以使用加号允许的语法指定，减号是 deny。使用 . 作为前缀来支持通配符。例如，允许 com.foo 和所有子软件包，然后 specfy com.foo.。可以用逗号分隔多个权限，如 com.foo.,-com.foo.bar.MySecretBean。以下默认权限始终被包括： -,java.lang.,java.util.，除非其使用键 org.apache.camel.xstream.permissions 指定 JVM 系统属性来覆盖它。

Name	默认值	Java 类型	描述
allowUnmarshallType	false	布尔值	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。只有在需要使用时，才会启用。
timezone		字符串	如果设置，则 Jackson 会在 marshalling/unmarshalling 时使用 Timezone。此选项对其他 Json DataFormat （如 gson、fastjson 和 xstream）没有影响。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

188.2. 使用 JAVA DSL

```
// lets turn Object messages into XML then send to MQSeries
from("activemq:My.Queue").
  marshal().xstream().
  to("mqseries:Another.Queue");
```

如果要配置 Camel 用于消息转换的 XStream 实例，您只需在 DSL 级别传递对该实例的引用。

```
XStream xStream = new XStream();
xStream.aliasField("money", PurchaseOrder.class, "cash");
// new Added setModel option since Camel 2.14
xStream.setModel("NO_REFERENCES");
...

from("direct:marshal").
  marshal(new XStreamDataFormat(xStream)).
  to("mock:marshaled");
```

188.3. XMLINPUTFACTORY 和 XMLOUTPUTFACTORY

XStream 库使用 javax.xml.stream.XMLInputFactory 和 javax.xml.stream.XMLOutputFactory，您可以控制应该使用哪个工厂实施。

使用这个算法发现 Factory: 1。使用 javax.xml.stream.XMLInputFactory , javax.xml.stream.XMLOutputFactory 系统属性。2.使用 JRE_HOME 目录中的 lib/xml.stream.properties 文件。3.使用 Services API （如果有）来通过查看 META-INF/services/javax.xml.stream.XMLInputFactory,META-

INF/services/javax.xml.stream.XMLOutputFactory 文件来决定 classname。4.使用平台默认 XMLInputFactory、XMLOutputFactory 实例。

188.4. 如何在 XSTREAM DATAFORMAT 中设置 XML 编码？

在 Camel 2.2.0 中，您可以使用密钥 Exchange.CHARSET_NAME 设置 Exchange 的属性，或者在 DSL 或 Spring 配置上设置编码属性，在 Xstream DataFormat 中设置 XML 编码。

```
from("activemq:My.Queue").
  marshal().xstream("UTF-8").
  to("mqseries:Another.Queue");
```

188.5. 设置 XSTREAM DATAFORMAT 的类型权限

在 Camel 中，始终使用路由中的自己的处理步骤来过滤和阻止某些 XML 文档路由到 XStream 的 unmarhall 步骤。从 Camel 2.16.1、2.15.5，您可以设置 XStream 的类型权限，以自动允许或拒绝某些类型的实例化。

Camel 使用的默认类型权限设置拒绝所有类型，但 java.lang 和 java.util 软件包中除外。此设置可以通过设置系统属性 org.apache.camel.xstream.permissions 来更改。其值是一个以逗号分隔的权限术语字符串，每个术语代表允许或拒绝的类型，具体取决于术语是前缀为 " （请注意 " 可能会被省略）还是分别带有 '-'。

每个术语都可以包含一个通配符字符 "。例如，值 "-,java.lang.,java.util." 表示拒绝除 java.lang unset 和 java.util targeted 类以外的所有类型。将此值设置为空字符串 ""，会恢复到默认的 XStream 类型权限处理，该处理拒绝某些黑名单类并允许其他用户。

通过设置 type permissions 属性，可以在单独的 XStream DataFormat 实例上扩展 type 权限设置。

```
<dataFormats>
  <xstream id="xstream-default"
        permissions="org.apache.camel.samples.xstream.*"/>
  ...
```

第 189 章 JSONPATH LANGUAGE

从 Camel 版本 2.13 开始提供

Camel 支持 JSonPath，以允许对 json 消息使用 Expression 或 Predicate。

```
from("queue:books.new")
  .choice()
    .when().jsonpath("$.store.book[?(@.price < 10)]")
      .to("jms:queue:book.cheap")
    .when().jsonpath("$.store.book[?(@.price < 30)]")
      .to("jms:queue:book.average")
    .otherwise()
      .to("jms:queue:book.expensive")
```

189.1. JSONPATH 选项

JsonPath 语言支持 7 个选项，如下所列。

Name	默认值	Java 类型	描述
resultType		字符串	设置结果类型的类名称（输出中的类型）
suppressExceptions	false	布尔值	是否阻止例外，如 PathNotFoundException。
allowSimple	true	布尔值	JsonPath 表达式中是否允许内联简单的异常
allowEasyPredicate	true	布尔值	允许使用 easy predicate 解析器来预解析 predicates。
writeAsString	false	布尔值	是否将每个行/元素的输出写为 JSON String 值，而不是映射/POJO 值。
headerName		字符串	用作输入的标头名称，而不是消息正文
trim	true	布尔值	是否修剪值以移除前导和结尾的空格和换行符

189.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 7 个选项，如下所列。

Name	描述	默认值	类型
camel.language.jsonpath.allow-easy-predicate	允许使用 easy predicate 解析器来预解析 predicates。	true	布尔值
camel.language.jsonpath.allow-simple	JsonPath 表达式中是否允许内联简单的异常	true	布尔值
camel.language.jsonpath.enabled	启用 jsonpath 语言	true	布尔值
camel.language.jsonpath.header-name	用作输入的标头名称，而不是消息正文		字符串
camel.language.jsonpath.suppress-exceptions	是否阻止例外，如 PathNotFoundException。	false	布尔值
camel.language.jsonpath.trim	是否修剪值以移除前导和结尾的空格和换行符	true	布尔值
camel.language.jsonpath.write-as-string	是否将每个行/元素的输出写为 JSON String 值，而不是映射/POJO 值。	false	布尔值

189.3. 使用 XML 配置

如果要在 Spring XML 文件中配置路由，您可以使用 JSonPath 表达式，如下所示

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="direct:start"/>
    <choice>
      <when>
        <jsonpath>$.store.book[?(@.price &lt; 10)]</jsonpath>
        <to uri="mock:cheap"/>
      </when>
      <when>
        <jsonpath>$.store.book[?(@.price &lt; 30)]</jsonpath>
        <to uri="mock:average"/>
      </when>
      <otherwise>
        <to uri="mock:expensive"/>
      </otherwise>
```

```
    </choice>
  </route>
</camelContext>
```

189.4. 语法

如需了解更多示例，请参阅 JSonPath 项目页面。

189.5. 轻松语法

从 Camel 2.19 开始提供

当您只想使用 jsonpath 语法定义基本 predicate 时，很难记住语法。例如，要找出您需要做的所有价格手册

```
$.store.book[?(@.price < 20)]
```

但是，如果您只可以写入它，会怎么做

```
store.book.price < 20
```

如果您只想查看具有价格键的节点，可以省略该路径

```
price < 20
```

为了支持此功能，有一个 EasyPredicateParser，如果您已使用基本风格定义 predicate，则启动它。这意味着 predicate 不得以 $ 符号开头，且仅包含一个 Operator。

简单语法是：

```
left OP right
```

您可以在正确的运算符中使用 Camel 简单语言，例如

```
store.book.price < ${header.limit}
```

189.6. 支持的消息正文类型

Camel JSonPath 支持使用以下类型的消息正文：

类型	注释
File	从文件中读取
字符串	普通字符串
Map	消息 bodies 作为 **java.util.Map** 类型
list	消息正文作为 **java.util.List** 类型
POJO	**可选的** If Jackson 位于 classpath 上，则 camel-jsonpath 可以使用 Jackson 来读取消息正文为 POJO，并转换为 **java.util.Map**，它由 JSonPath 支持。例如，您可以添加 **camel-jackson** 作为依赖项，使其包含 Jackson。
InputStream	如果以上类型都不匹配，则 Camel 会尝试将消息正文读取为 **java.io.InputStream**。

如果消息正文是不支持的类型，则默认抛出异常，但您可以将 JSonPath 配置为阻止异常（请参阅以下）

189.7. 抑制异常

从 Camel 2.16 开始提供

如果 json 有效负载没有到配置的 jsonpath 表达式的有效路径，则 jsonpath 会抛出异常。在一些用例中，如果 json 有效负载包含可选数据，您可能需要忽略它。因此，您可以将选项 suppressExceptions 设置为 true 以忽略它，如下所示：

```
from("direct:start")
   .choice()
      // use true to suppress exceptions
      .when().jsonpath("person.middlename", true)
         .to("mock:middle")
      .otherwise()
         .to("mock:other");
```

在 XML DSL 中：

```
<route>
  <from uri="direct:start"/>
  <choice>
    <when>
      <jsonpath suppressExceptions="true">person.middlename</jsonpath>
      <to uri="mock:middle"/>
    </when>
    <otherwise>
      <to uri="mock:other"/>
    </otherwise>
  </choice>
</route>
```

此选项也适用于 @JsonPath 注释。

189.8. 内联简单例外

从 Camel 2.18 开始提供

现在，可以使用简单语法 ${xxx}，在 JSonPath 表达式中内联 Simple 语言表达式。下面是一个示例：

```
from("direct:start")
  .choice()
    .when().jsonpath("$.store.book[?(@.price < ${header.cheap})]")
      .to("mock:cheap")
    .when().jsonpath("$.store.book[?(@.price < ${header.average})]")
      .to("mock:average")
    .otherwise()
      .to("mock:expensive");
```

在 XML DSL 中：

```
<route>
  <from uri="direct:start"/>
  <choice>
    <when>
      <jsonpath>$.store.book[?(@.price < ${header.cheap})]</jsonpath>
      <to uri="mock:cheap"/>
    </when>
    <when>
      <jsonpath>$.store.book[?(@.price < ${header.average})]</jsonpath>
      <to uri="mock:average"/>
    </when>
    <otherwise>
      <to uri="mock:expensive"/>
```

```
    </otherwise>
  </choice>
</route>
```

您可以通过将选项 allowSimple 设置为 false 来关闭对内联简单表达式的支持，如下所示：

```
.when().jsonpath("$.store.book[?(@.price < 10)]", false, false)
```

在 XML DSL 中：

```
<jsonpath allowSimple="false">$.store.book[?(@.price < 10)]</jsonpath>
```

189.9. JSONPATH 注入

您可以使用 Bean Integration 对 bean 调用方法，并使用 JSonPath 等各种语言从消息中提取值并将其绑定到方法参数。

例如：

```
public class Foo {

    @Consume(uri = "activemq:queue:books.new")
    public void doSomething(@JsonPath("$.store.book[*].author") String author, @Body String
json) {
      // process the inbound message here
    }
}
```

189.10. 编码检测

自 Camel 版本 2.16 起，会自动检测到 JSON 文档的编码，如果文档以 unicode (UTF-8, UTF-16LE, UTF-16BE, UTF-16BE, UTF-32LE, UTF-32BE)中编码，如 RFC-4627 中指定的。如果编码是非unicode 编码，您可以确保以 String 格式向 JSONPath 组件输入文档，或者您可以在标题 "CamelJsonPathJsonEncoding" (JsonpathConstants.HEADER_JSON_ENCODING)中指定编码。

189.11. 将 JSON 数据分成子行，作为 JSON

您可以使用 jsonpath 来分割 JSon 文档，例如：

```
from("direct:start")
    .split().jsonpath("$.store.book[*]")
    .to("log:book");
```

然后，记录每个图书，但消息正文是一个 映射 实例。有时，您可能需要输出它作为普通 String JSon 值，它可以从 Camel 2.20 开始使用 writeAsString 选项完成，如下所示：

```
from("direct:start")
    .split().jsonpathWriteAsString("$.store.book[*]")
    .to("log:book");
```

然后，每个图书都会作为 String JSon 值进行记录。对于早期版本的 Camel，您需要使用 camel-jackson dataformat 和 marshal 消息正文，使其将消息正文从 Map 转换为 String 类型。

189.12. 使用标头作为输入

从 Camel 2.20 开始提供

默认情况下，jsonpath 使用消息正文作为输入源。但是，您还可以通过指定 headerName 选项来使用标头作为输入。

例如，要计算存储在名为 books 的标题中的 json 文档中的 图书 数量，您可以执行以下操作：

```
from("direct:start")
    .setHeader("numberOfBooks")
        .jsonpath("$..store.book.length()", false, int.class, "books")
    .to("mock:result");
```

在上面的 jsonpath 表达式中，我们将标头名称指定为 书本，并且我们被告知我们想要将结果转换为 int.class 的整数。

XML DSL 中的相同例子是：

```
<route>
  <from uri="direct:start"/>
  <setHeader headerName="numberOfBooks">
    <jsonpath headerName="books" resultType="int">$..store.book.length()</jsonpath>
  </transform>
  <to uri="mock:result"/>
</route>
```

189.13. 依赖项

要在 camel 路由中使用 JSonPath，您需要添加对实现 JSonPath 语言的 camel-jsonpath 的依赖。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jsonpath</artifactId>
  <version>x.x.x</version>
</dependency>
```

第 190 章 JT400 组件

从 Camel 版本 1.5 开始提供

jt400 组件允许您使用数据队列与 AS/400 系统交换消息。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jt400</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

190.1. URI 格式

```
jt400://user:password@system/QSYS.LIB/LIBRARY.LIB/QUEUE.DTAQ[?options]
```

调用远程程序(Camel 2.7)

```
jt400://user:password@system/QSYS.LIB/LIBRARY.LIB/program.PGM[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

190.2. JT400 选项

JT400 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
connectionPool (advanced)	返回此组件使用的默认连接池。		AS400ConnectionPool
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

JT400 端点使用 URI 语法进行配置：

```
jt400:userID:password/systemName/objectPath.type
```

使用以下路径和查询参数：

190.2.1. 路径参数(5 参数)：

Name	描述	默认值	类型
userID	**必需** 返回 AS/400 用户的 ID。		字符串
password	**需要** 返回 AS/400 用户的密码。		字符串
systemName	**必需** 返回 AS/400 系统的名称。		字符串
objectPath	**必需** 返回此端点的目标对象的完全限定集成文件系统路径名称。		字符串
type	**必需** 是否使用数据队列或远程程序调用		Jt400Type

190.2.2. 查询参数(30 参数)：

Name	描述	默认值	类型
ccsid (common)	设置 CCSID 用于与 AS/400 系统的连接。		int
格式 (common)	设置用于发送消息的数据格式。	text	格式
guiAvailable (common)	设置在运行 Camel 的环境中是否启用了 AS/400 提示。	false	布尔值
keyed (common)	使用密钥或非密钥的数据队列。	false	布尔值
outputFieldsIdxArray (common)	指定哪些字段(program 参数)是输出参数。		Integer[]
outputFieldsLengthArray (common)	指定字段(program 参数)长度，如 AS/400 程序定义中。		Integer[]
searchKey (common)	搜索关键数据队列的关键。		字符串

Name	描述	默认值	类型
searchType (common)	搜索类型，如 EQ 等于等。	EQ	SearchType
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
readTimeout (consumer)	在尝试读取数据队列的新消息时，消费者将等待超时。	30000	int
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int

Name	描述	默认值	类型
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
procedureName (procedureName)	从服务程序调用的步骤名称		字符串
安全 （安全）	到 AS/400 的连接是否通过 SSL 保护。	false	布尔值

190.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.jt400.connection-pool	返回此组件使用的默认连接池。选项是一个 com.ibm.as400.access.AS400ConnectionPool 类型。		字符串
camel.component.jt400.enabled	启用 jt400 组件	true	布尔值
camel.component.jt400.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

190.4. 使用方法

当配置为消费者端点时，端点将在远程系统上轮询数据队列。对于数据队列上的每个条目，使用 In 消息正文中的条目数据发送一个新的 Exchange，其格式为 String 或 byte[]，具体取决于格式。对于提供程序端点，In 消息正文内容将作为原始字节或文本放在数据队列中。

190.5. 连接池

从 Camel 2.10 开始提供

连接池从 Camel 2.10 开始使用。您可以在 Jt400Component 上明确配置连接池，或者作为端点上的 uri 选项。

190.5.1. 远程程序调用(Camel 2.7)

此端点预期输入是 String 数组或 byte[] 数组（取决于格式），并通过原生 jt400 库机制处理所有 CCSID 处理。通过将 null 传递为位置中的值（远程程序必须支持它），可以省略 参数。在程序执行端点后，端点会返回一个 String 数组或 byte[] 数组，其值与程序返回的值（只有输入的参数将包含与调用开始相同的数据）。此端点不实施供应商端点！

190.6. EXAMPLE

在以下代码片段中，发送到 direct:george 端点的交换数据将放置在名为 LIVERPOOL 的系统上的数据队列 PENNYLANE 中。
另一个用户连接到同一数据队列，以接收来自数据队列的信息并将其转发到 mock:ringo 端点。

```
public class Jt400RouteBuilder extends RouteBuilder {
```

```
    @Override
    public void configure() throws Exception {

from("direct:george").to("jt400://GEORGE:EGROEG@LIVERPOOL/QSYS.LIB/BEATLES.LIB/PE
NNYLANE.DTAQ");

from("jt400://RINGO:OGNIR@LIVERPOOL/QSYS.LIB/BEATLES.LIB/PENNYLANE.DTAQ").to("
mock:ringo");
    }
}
```

190.6.1. 远程程序调用示例(Camel 2.7)

在以下代码片段中，发送到 direct:work 端点的数据交换将包含三个字符串，该字符串将用作库"计算"中程序"计算"的参数。该程序将在第二和第 3 个参数中写入输出值。所有参数都将发送到 direct:play 端点。

```
public class Jt400RouteBuilder extends RouteBuilder {
    @Override
    public void configure() throws Exception {

from("direct:work").to("jt400://GRUPO:ATWORK@server/QSYS.LIB/assets.LIB/compute.PGM?
fieldsLength=10,10,512&ouputFieldsIdx=2,3").to(“direct:play”);
    }
}
```

190.6.2. 写入密钥的数据队列

```
from("jms:queue:input")
.to("jt400://username:password@system/lib.lib/MSGINDQ.DTAQ?keyed=true");
```

190.6.3. 从密钥化数据队列读取

```
from("jt400://username:password@system/lib.lib/MSGOUTDQ.DTAQ?
keyed=true&searchKey=MYKEY&searchType=GE")
.to("jms:queue:output");
```

190.7. 另请参阅

- 配置 Camel
- 组件
- 端点

- *开始使用*

第 191 章 KAFKA 组件

从 Camel 版本 2.13 开始提供

kafka: 组件用于与 Apache Kafka 消息代理通信。

Maven 用户需要将以下依赖项添加到此组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-kafka</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

191.1. URI 格式

```
kafka:topic[?options]
```

191.2. 选项

Kafka 组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	允许预先配置带有端点重复使用的通用选项的 Kafka 组件。		KafkaConfiguration
代理 (common)	要使用的 Kafka 代理的 URL。格式为 host1:port1,host2:port2，列表可以是代理的子集，也可以是指向代理子集的 VIP。这个选项在 Kafka 文档中被称为 bootstrap.servers。		字符串
workerpool (advanced)	要在 kafka 服务器确认通过异步非阻塞处理从 KafkaProducer 发送到它的消息后，使用共享自定义 worker 池继续路由 Exchange。如果使用这个选项，则必须处理线程池的生命周期，以便在不再需要时关闭池。		ExecutorService
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值

Name	描述	默认值	类型
breakOnFirstError (consumer)	这个选项控制消费者处理交换时会发生什么，它会失败。如果选项为 false，则使用者将继续进入下一个消息并处理它。如果选项为 true，则消费者将中断，并且将看到导致失败的消息偏移，然后重新尝试处理此消息。但是，如果绑定到每次都失败，则可能会导致意外处理同一消息，例如 poison 信息。因此，建议您处理这个情况，例如使用 Camel 的错误处理程序。	false	布尔值
allowManualCommit (consumer)	是否允许通过 KafkaManualCommit 进行手动提交。如果启用了这个选项，则 KafkaManualCommit 实例存储在 Exchange message 标头中，它允许最终用户访问此 API 并通过 Kafka 使用者执行手动偏移提交。	false	布尔值
kafkaManualCommit Factory (consumer)	用于创建 KafkaManualCommit 实例的工厂。这允许在进行手动提交时，需要自定义 KafkaManualCommit 实例来插件自定义 KafkaManualCommit 实例。		KafkaManualCommit Factory
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
shutdownTimeout (common)	超时时间（以毫秒为单位）会安全等待消费者或制作者关闭并终止其 worker 线程。	30000	int

Kafka 端点使用 URI 语法进行配置：

```
kafka:topic
```

使用以下路径和查询参数：

191.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
topic	要使用的主题 **必需** 名称。在消费者中，您可以使用逗号分隔多个主题。生产者只能发送消息到单个主题。		字符串

191.2.2. 查询参数(94 参数)：

Name	描述	默认值	类型
代理 (common)	要使用的 Kafka 代理的 URL。格式为 host1:port1,host2:port2，列表可以是代理的子集，也可以是指向代理子集的 VIP。这个选项在 Kafka 文档中被称为 bootstrap.servers。		字符串
clientId (common)	客户端 ID 是在每个请求中发送的用户指定的字符串，以帮助跟踪调用。它应在逻辑上标识发出请求的应用程序。		字符串
headerFilterStrategy (common)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
reconnectBackoffMaxMs (common)	重新连接到重复连接失败的代理时等待的最大时间（以毫秒为单位）。如果提供，每个主机的 backoff 将为每个连续的连接失败指数增加，直到最高值。在计算 backoff 增长后，添加了 20% 的随机 jitter，以避免连接状况。	1000	整数
allowManualCommit (consumer)	是否允许通过 KafkaManualCommit 进行手动提交。如果启用了这个选项，则 KafkaManualCommit 实例存储在 Exchange message 标头中，它允许最终用户访问此 API 并通过 Kafka 使用者执行手动偏移提交。	false	布尔值
autoCommitEnable (consumer)	如果为 true，请定期提交到 ZooKeeper，以偏移已由消费者获取的信息。当进程作为新消费者开始的位置失败时，将使用此提交偏移。	true	布尔值
autoCommitIntervalMs (consumer)	ms 中消费者偏移用于 zookeeper 的频率。	5000	整数
autoCommitOnStop (consumer)	当消费者停止以确保代理具有最近使用的消息提交时，是否执行显式自动提交。这要求打开 autoCommitEnable 选项。可能的值有： sync、sync 或 none。sync 是默认值。	同步	字符串
autoOffsetReset (consumer)	当 ZooKeeper 中没有初始偏移时，或者偏移没有范围： earliest : 自动将偏移重置为最早的偏移量： 自动将偏移重置为最新的偏移失败： 向消费者抛出异常	latest	字符串
breakOnFirstError (consumer)	这个选项控制消费者处理交换时会发生什么，它会失败。如果选项为 false，则使用者将继续进入下一个消息并处理它。如果选项为 true，则消费者将中断，并且将看到导致失败的消息偏移，然后重新尝试处理此消息。但是，如果绑定到每次都失败，则可能会导致意外处理同一消息，例如 poison 信息。因此，建议您处理这个情况，例如使用 Camel 的错误处理程序。	false	布尔值

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
checkCrcs (consumer)	自动检查消耗的记录的 CRC32。这样可确保不会发生对消息进行 on-wire 或 on-disk 崩溃。此检查增加了一些开销，因此在寻求极佳性能的情况下可能会禁用它。	true	布尔值
consumerRequestTimeoutMs (consumer)	配置控制客户端等待请求响应的最长时间。如果在超时之前没有收到响应，客户端将在需要时重新发送请求（如果重试用时失败）。	40000	整数
consumersCount (consumer)	连接到 kafka 服务器的消费者数量	1	int
consumerStreams (consumer)	消费者上的并发用户数	10	int
fetchMaxBytes (consumer)	如果 fetch 请求的第一个非空分区中的第一个非空分区大于这个值，则服务器应返回的最大数据量。代理接受的最大消息大小通过 message.max.bytes (broker config)或 max.message.bytes (topic config)定义。请注意，消费者并行执行多个获取。	52428800	整数
fetchMinBytes (consumer)	服务器应返回的最小数据量，用于获取请求。如果数据不足，请求将等待这么多的数据在回答请求前累积。	1	整数
fetchWaitMaxMs (consumer)	如果没有足够的数据来立即满足 fetch.min.bytes，则在回答获取请求前，服务器将阻止的最大时间量。	500	整数
GroupId (consumer)	唯一标识此消费者所属的消费者进程组的字符串。通过设置同一组 id 多个进程，表示它们都是同一消费者组的一部分。消费者需要这个选项。		字符串
heartbeatIntervalMs (consumer)	使用 Kafka 的组管理功能时，与消费者协调器之间预期的时间。心跳用于确保消费者保持活跃状态，并在新用户加入或离开该组时促进重新平衡。该值必须小于 session.timeout.ms，但设置的值通常不应超过该值的 1/3。它可以调整甚至较低，以控制正常重新平衡的预期时间。	3000	整数

Name	描述	默认值	类型
kafkaHeaderDeserializer (consumer)	设置自定义 KafkaHeaderDeserializer，将 deserialization kafka 标头值设置为 camel 标头值。		KafkaHeaderDeserializer
keyDeserializer (consumer)	实现 Deserializer 接口的密钥的反序列化器类。	org.apache.kafka.common.serialization.StringDeserializer	字符串
maxPartitionFetchBytes (consumer)	服务器将返回的每个分区的最大数据量。用于请求的最大内存总量为 #partitions max.partition.fetch.bytes。此大小必须至少与服务器允许的最大消息大小相同，或者生产者可以发送消息大于消费者是否可以获取。如果发生这种情况，消费者会卡住在某个分区上获取大型消息。	1048576	整数
maxPollIntervalMs (consumer)	使用消费者组管理时，poll （）调用之间的最大延迟。这会在获取更多记录前将上限放在消费者可以闲置的时间长度。如果在这个超时时间前没有调用 poll （），则消费者被视为失败，并且组将重新平衡，以便将分区重新分配给另一个成员。		Long
maxPollRecords (consumer)	在单个调用中返回的最大记录数（）	500	整数
offsetRepository (consumer)	使用偏移存储库来本地存储主题的每个分区的偏移量。定义一个将禁用自动提交。		StateRepository
partitionAssignor (consumer)	使用组管理时，客户端将在消费者实例之间分发分区所有权的策略的类名称。	org.apache.kafka.clients.consumer.RangeAssignor	字符串
pollTimeoutMs (consumer)	轮询 KafkaConsumer 时使用的超时。	5000	Long
seekTo (consumer)	set if KafkaConsumer will read from start or end on start: beginning : read from beginning end : read from end this is replace the earlier property seekToBeginning		字符串

Name	描述	默认值	类型
sessionTimeoutMs (consumer)	使用 Kafka 组管理工具时用于检测故障的超时。	10000	整数
shutdownTimeout (common)	超时时间（以毫秒为单位）会安全等待消费者或制作者关闭并终止其 worker 线程。	30000	int
topicIsPattern (consumer)	主题是否为模式（正则表达式）。这可用于订阅与模式匹配的主题数量。	false	布尔值
valueDeserializer (consumer)	deserializer 类，用于实现 Deserializer 接口的值。	org.apache.kafka.common.serialization.StringDeserializer	字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
bridgeEndpoint (producer)	如果选项为 true，则 KafkaProducer 将忽略入站消息的 KafkaConstants.TOPIC 标头设置。	false	布尔值
bufferMemorySize (producer)	制作者可用于缓冲记录等待发送到服务器的内存总量。如果记录比将记录发送到服务器的速度快，生产者将根据 block.on.buffer.full.full 指定的首选项阻止或抛出异常。此设置应大致对应于制作者将使用的总内存，但不是硬绑定，因为生产者使用的所有内存都不用于缓冲。一些额外的内存将用于压缩（如果启用了压缩），以及维护动态请求。	33554432	整数
circularTopicDetection (producer)	如果选项为 true，则 KafkaProducer 将检测信息是否尝试发回到它可能来自的同一主题，如果消息是从 kafka 使用者原始的。如果 KafkaConstants.TOPIC 标头与原始 kafka 消费者主题相同，则忽略标头设置，并使用制作者端点的主题。换句话说，这样可避免将同一消息发送回它的位置。如果选项 bridgeEndpoint 设为 true，则不使用这个选项。	true	布尔值
compressionCodec (producer)	这个参数允许您为此制作者生成的所有数据指定压缩 codec。有效值为 none、gzip 和 snappy。	none	字符串

Name	描述	默认值	类型
connectionMaxIdleMs (producer)	在这个配置指定的毫秒数后关闭闲置连接。	540000	整数
enableIdempotence (producer)	如果设置为 'true'，则生成者将确保每个消息的一个副本用流写入。如果 'false'，则生成者可能会重试流中重试消息的重复。如果设置为 true，这个选项需要 max.in.flight.requests.per.connection 设置为 1，并且重试不能为零，另外将 acks 设置为 'all'。	false	布尔值
kafkaHeaderSerializer (producer)	设置自定义 KafkaHeaderDeserializer，将序列化 camel 标头值设置为 kafka 标头值。		KafkaHeaderSerializer
key (producer)	记录键（如果未指定密钥，则为 null）。如果配置了这个选项，则它优先于标头 KafkaConstants#KEY		字符串
keySerializerClass (producer)	键的 serializer 类（如果没有提供，则默认为与消息相同）。	org.apache.kafka.common.serialization.StringSerializer	字符串
lingerMs (producer)	生产者将请求传输之间到达单个批处理请求的任何记录组合在一起。通常，这只在记录到达速度快于发送时发生。然而，在某些情况下，客户端可能希望减少请求的数量，即使负载低下也是如此。此设置通过添加少量的 artificial 延迟来实现这一目的，而是不会立即发送记录，而是会等待到给定延迟，以允许发送其他记录一起批处理。这可能被视为与 TCP 中的 Nagle 算法类似。此设置会使批处理延迟有上限：一旦我们获得 batch.size，对于一个分区而言，无论此设置如何立即发送，则无论此设置如何，我们会立即发送的记录数超过这个分区所累计的字节数，我们将"闲置"，等待更多记录显示。此设置默认为 0 （例如，无延迟）。例如，设置 linger.ms=5 会影响减少发送的请求数，但会对负载发送的记录最多增加 5ms 的延迟。	0	整数
maxBlockMs (producer)	配置控制发送到 kafka 的时长。这些方法可能会因为多种原因阻止。例如：缓冲区已满，元数据不可用。此配置对获取元数据、键和值序列化和值、分区和分配缓冲区内存的总时间施加最大限制，在执行 send （）时对缓冲区内存进行分区和分配。如果是 partitionsFor （），此配置会在等待元数据时产生最大时间阈值	60000	整数

Name	描述	默认值	类型
maxInFlightRequest (producer)	客户端在阻止前在单个连接上发送的最大未确认请求数。请注意，如果此设置大于 1，且发送失败，则因为重试而出现消息重新排序的风险（例如，如果启用了重试）。	5	整数
maxRequestSize (producer)	请求的最大大小。这实际上也是最大记录大小的上限。请注意，服务器对记录大小有自己的上限，可能与此不同。此设置将限制制作者将在单个请求中发送的记录批处理数量，以避免发送大量请求。	1048576	整数
metadataMaxAgeMs (producer)	即使我们未看到任何分区领导力更改来主动发现任何新的代理或分区，我们才会强制刷新元数据的时间（以毫秒为单位）。	300000	整数
metricReporters (producer)	用作指标报告器的类列表。实施 MetricReporter 接口允许插入新指标创建通知的类。总是包括 JmxReporter 来注册 JMX 统计信息。		字符串
metricsSampleWindowMs (producer)	为计算指标维护的示例数量。	30000	整数
noOfMetricsSample (producer)	为计算指标维护的示例数量。	2	整数
partitioner (producer)	在子主题间分区消息的分区类。默认分区器基于密钥的哈希。	org.apache.kafka.clients.producer.internals.DefaultPartitioner	字符串
PartitionKey (producer)	将记录发送到的分区（如果没有指定分区，则为 null）。如果配置了这个选项，则它优先于标头 KafkaConstants#PARTITION_KEY		整数

Name	描述	默认值	类型
producerBatchSize (producer)	每当将多个记录发送到同一分区时，生成者将尝试将记录一起批处理到较少的请求。这有助于客户端和服务器的性能。此配置控制默认批处理大小（以字节为单位）。对于大于此 size.Requests 的批处理记录不会进行任何尝试。发送到代理的请求将包含多个批处理，每个分区一个可用数据。小批处理大小将较少的批处理，并且可能会降低吞吐量（批量大小为零将完全禁用批处理）。非常大的批处理大小可能会更严重地使用内存，因为我们将始终在附加记录下分配指定批处理大小的缓冲。	16384	整数
queueBufferingMaxMessages (producer)	在使用 async 模式时，可以在生成者必须被阻止或数据丢弃前，可以排队生成者的最大未发送消息数。	10000	整数
receiveBufferBytes (producer)	读取数据时要使用的 TCP 接收缓冲区(SO_RCVBUF)的大小。	65536	整数
reconnectBackoffMs (producer)	尝试重新连接给定主机前需要等待的时间。这可避免在紧密循环中重复连接到主机。此 backoff 适用于消费者向代理发送的所有请求。	50	整数
RecordMetadata (producer)	制作者是否应该存储发送到 Kafka 的 RecordMetadata 结果。结果存储在包含 RecordMetadata 元数据的列表中。该列表存储在带有键 KafkaConstantsKAFKA_RECORDMETA 的标头中	true	布尔值
requestRequiredAcks (producer)	在考虑请求完成前，生成者需要收到的确认数量。这控制发送的记录的持久性。以下设置是常见的：acks=0 如果设为零，则生成者不会等待服务器中的任何确认。记录将立即添加到套接字缓冲区中并被视为发送。在这种情况下，不能保证服务器收到记录，重试配置不会生效（因为客户端通常不知道任何故障）。每个记录返回的偏移量始终设置为 -1 acks=1，这意味着领导机会将记录写入到其本地日志，但不会等待所有后续者的完全确认。在这种情况下，领导机在确认记录后马上失败，但在后续者复制记录之前，该记录将会丢失。acks=all 意味着领导机将等待全部同步副本确认记录。这样可保证记录在至少一个同步副本仍然处于活动状态时不会丢失。这是最强的可用保证。	1	字符串
requestTimeoutMs (producer)	代理在向客户端发送错误前尝试满足 request.required.acks 要求的时间长度。	305000	整数

Name	描述	默认值	类型
retries (producer)	设置大于零的值将导致客户端重新发送发送失败的记录，并显示潜在的临时错误。请注意，这个重试与在收到错误时重新记录不同的是不同的。允许重试可能会更改记录顺序，因为如果将两个记录发送到单个分区，第一次失败并重试，但会成功，但第二个记录可能会首先出现。	0	整数
retryBackoffMs (producer)	在每个重试前，生成者会刷新相关主题的元数据，以查看是否选择了新的领导。由于领导选举需要一些时间，此属性指定制作者在刷新元数据前等待的时间。	100	整数
sendBufferBytes (producer)	套接字写入缓冲区大小	131072	整数
serializerClass (producer)	用于消息的 serializer 类。	org.apache.kafka.common.serialization.StringSerializer	字符串
workerpool (producer)	要在 kafka 服务器确认使用异步非阻塞处理从 KafkaProducer 发送到它的消息后，使用自定义 worker 池来继续路由 Exchange。		ExecutorService
workerPoolCoreSize (producer)	kafka 服务器确认从 KafkaProducer 发送到 KafkaProducer 的消息后，用于继续路由 Exchange 的 worker 池的核心线程数量，使用异步非阻塞处理。	10	整数
workerPoolMaxSize (producer)	kafka 服务器确认从 KafkaProducer 发送到 KafkaProducer 的消息后，用于继续路由 Exchange 的 worker 池的最大线程数量。	20	整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
interceptorClasses (monitoring)	为生成者或消费者设置拦截器。生产者拦截器必须是实施 org.apache.kafka.clients.producer.ProducerInterceptor Consumer 拦截器的类，必须是实现 org.apache.kafka.clients.consumer.ConsumerInterceptor.consumer.ConsumerInterceptor 注意，它会在运行时抛出类广播异常		字符串
kerberosBeforeReloginMin Time (security)	刷新尝试之间的登录线程睡眠时间。	60000	整数

Name	描述	默认值	类型
kerberosInitCmd (security)	Kerberos kinit 命令路径。默认为 /usr/bin/kinit	/usr/bin/kinit	字符串
kerberosPrincipalToLocalRules (security)	从主体名称映射到短名称（通常是操作系统用户名）的规则列表。规则按顺序评估，第一个与主体名称匹配的规则用于将其映射到短名称。稍后列表中的任何规则都会被忽略。默认情况下，用户名/主机名 REALM的主体名称映射到 username。有关格式的详情，请参阅安全授权和 acls。可以使用逗号分隔多个值	DEFAULT	字符串
kerberosRenewJitter (security)	添加到续订时间的随机 jitter 的百分比。	0.05	å�Œ
kerberosRenewWindowFactor (security)	登录线程将处于睡眠状态，直到达到最后一次刷新到票据到期的时间因素前处于睡眠状态，此时将尝试续订票据。	0.8	å�Œ
saslJaasConfig (security)	公开 kafka sasl.jaas.config 参数示例：org.apache.kafka.common.security.plain.PlainLoginModule required username=USERNAME password=PASSWORD;		字符串
saslKerberosServiceName (security)	Kafka 运行的 Kerberos 主体名称。这可以在 Kafka 的 JAAS 配置或 Kafka 配置中定义。		字符串
saslMechanism (security)	使用简单身份验证和安全层(SASL)机制。如需有效值，请参阅 http://www.iana.org/assignments/sasl-mechanisms/sasl-mechanisms.xhtml	GSSAPI	字符串
securityProtocol (security)	用于与代理通信的协议。目前只支持 PLAINTEXT 和 SSL。	PLAINTEXT	字符串
sslCipherSuites (security)	密码套件列表。这是用来使用 TLS 或 SSL 网络协议协商网络连接的安全设置的身份验证、加密、MAC 和密钥交换算法的命名组合。只要默认支持所有可用的密码套件。		字符串
sslContextParameters (security)	使用 Camel SSLContextParameters 对象的 SSL 配置。如果配置在其他 SSL 端点参数之前应用。		SSLContextParameters
sslEnabledProtocols (security)	为 SSL 连接启用的协议列表。TLSv1.2、TLSv1.1 和 TLSv1 会被默认启用。	TLSv1.2,TLSv1.1,TLSv1	字符串
sslEndpointAlgorithm (security)	使用服务器证书验证服务器主机名的端点标识算法。		字符串

Name	描述	默认值	类型
sslKeymanagerAlgorithm (security)	密钥管理器工厂用于 SSL 连接的算法。默认值是为 Java 虚拟机配置的密钥管理器工厂算法。	SunX509	字符串
sslKeyPassword (security)	密钥存储文件中私钥的密码。这对客户端是可选的。		字符串
sslKeystoreLocation (security)	密钥存储文件的位置。这对客户端是可选的，可用于客户端进行双向身份验证。		字符串
sslKeystorePassword (security)	密钥存储文件的存储密码。这对客户端是可选的，只有在配置了 ssl.keystore.location 时才需要。		字符串
sslKeystoreType (security)	密钥存储文件的文件格式。这对客户端是可选的。默认值为 JKS	JKS	字符串
SSLProtocol (security)	用于生成 SSLContext 的 SSL 协议。默认设置为 TLS，对于大多数情况来说是正常的。最近 JVM 中允许的值是 TLS、TLSv1.1 和 TLSv1.2。较旧的 JVM 中可能会支持 SSL、SSLv2 和 SSLv3，但由于已知的安全漏洞，不建议使用它们。	TLS	字符串
sslProvider (security)	用于 SSL 连接的安全供应商的名称。默认值是 JVM 的默认安全提供程序。		字符串
sslTrustmanagerAlgorithm (security)	信任管理器工厂用于 SSL 连接的算法。默认值是为 Java 虚拟机配置的信任管理器工厂算法。	PKIX	字符串
sslTruststoreLocation (security)	信任存储文件的位置。		字符串
sslTruststorePassword (security)	信任存储文件的密码。		字符串
sslTruststoreType (security)	信任存储文件的文件格式。默认值为 JKS。	JKS	字符串

191.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 99 个选项，如下所列。

Name	描述	默认值	类型
camel.component.kafka.allow-manual-commit	是否允许通过 KafkaManualCommit 进行手动提交。如果启用了这个选项，则 KafkaManualCommit 实例存储在 Exchange message 标头中，它允许最终用户访问此 API 并通过 Kafka 使用者执行手动偏移提交。	false	布尔值
camel.component.kafka.break-on-first-error	这个选项控制消费者处理交换时会发生什么，它会失败。如果选项为 false，则使用者将继续进入下一个消息并处理它。如果选项为 true，则消费者将中断，并且将看到导致失败的消息偏移，然后重新尝试处理此消息。但是，如果绑定到每次都失败，则可能会导致意外处理同一消息，例如 poison 信息。因此，建议您处理这个情况，例如使用 Camel 的错误处理程序。	false	布尔值
camel.component.kafka.brokers	要使用的 Kafka 代理的 URL。格式为 host1:port1,host2:port2，列表可以是代理的子集，也可以是指向代理子集的 VIP。这个选项在 Kafka 文档中被称为 bootstrap.servers。		字符串
camel.component.kafka.configuration.allow-manual-commit	是否允许通过 KafkaManualCommit 进行手动提交。如果启用了这个选项，则 KafkaManualCommit 实例存储在 Exchange message 标头中，它允许最终用户访问此 API 并通过 Kafka 使用者执行手动偏移提交。	false	布尔值
camel.component.kafka.configuration.auto-commit-enable	如果为 true，请定期提交到 ZooKeeper，以偏移已由消费者获取的信息。当进程作为新消费者开始的位置失败时，将使用此提交偏移。	true	布尔值
camel.component.kafka.configuration.auto-commit-interval-ms	ms 中消费者偏移用于 zookeeper 的频率。	5000	整数
camel.component.kafka.configuration.auto-commit-on-stop	当消费者停止以确保代理具有最近使用的消息提交时，是否执行显式自动提交。这要求打开 autoCommitEnable 选项。可能的值有： sync、sync 或 none。sync 是默认值。	同步	字符串
camel.component.kafka.configuration.auto-offset-reset	当 ZooKeeper 中没有初始偏移时，或者偏移没有范围： earliest : 自动将偏移重置为最早的偏移量： 自动将偏移重置为最新的偏移失败： 向消费者抛出异常	latest	字符串

Name	描述	默认值	类型
camel.component.kafka.configuration.break-on-first-error	这个选项控制消费者处理交换时会发生什么，它会失败。如果选项为 false，则使用者将继续进入下一个消息并处理它。如果选项为 true，则消费者将中断，并且将看到导致失败的消息偏移，然后重新尝试处理此消息。但是，如果绑定到每次都失败，则可能会导致意外处理同一消息，例如 poison 信息。因此，建议您处理这个情况，例如使用 Camel 的错误处理程序。	false	布尔值
camel.component.kafka.configuration.bridge-endpoint	如果选项为 true，则 KafkaProducer 将忽略入站消息的 KafkaConstants.TOPIC 标头设置。	false	布尔值
camel.component.kafka.configuration.brokers	要使用的 Kafka 代理的 URL。格式为 host1:port1,host2:port2，列表可以是代理的子集，也可以是指向代理子集的 VIP。这个选项在 Kafka 文档中被称为 bootstrap.servers。		字符串
camel.component.kafka.configuration.buffer-memory-size	制作者可用于缓冲记录等待发送到服务器的内存总量。如果记录比将记录发送到服务器的速度快，生产者将根据 block.on.buffer.full.full 指定的首选项阻止或抛出异常。此设置应大致对应于制作者将使用的总内存，但不是硬绑定，因为生产者使用的所有内存都不用于缓冲。一些额外的内存将用于压缩（如果启用了压缩），以及维护动态请求。	33554432	整数
camel.component.kafka.configuration.check-crcs	自动检查消耗的记录的 CRC32。这样可确保不会发生对消息进行 on-wire 或 on-disk 崩溃。此检查增加了一些开销，因此在寻求极佳性能的情况下可能会禁用它。	true	布尔值
camel.component.kafka.configuration.circular-topic-detection	如果选项为 true，则 KafkaProducer 将检测信息是否尝试发回到它可能来自的同一主题，如果消息是从 kafka 使用者原始的。如果 KafkaConstants.TOPIC 标头与原始 kafka 消费者主题相同，则忽略标头设置，并使用制作者端点的主题。换句话说，这样可避免将同一消息发送回它的位置。如果选项 bridgeEndpoint 设为 true，则不使用这个选项。	true	布尔值
camel.component.kafka.configuration.client-id	客户端 ID 是在每个请求中发送的用户指定的字符串，以帮助跟踪调用。它应在逻辑上标识发出请求的应用程序。		字符串
camel.component.kafka.configuration.compression-codec	这个参数允许您为此制作者生成的所有数据指定压缩 codec。有效值为 none、gzip 和 snappy。	none	字符串

Name	描述	默认值	类型
camel.component.kafka.configuration.connection-max-idle-ms	在这个配置指定的毫秒数后关闭闲置连接。	540000	整数
camel.component.kafka.configuration.consumer-request-timeout-ms	配置控制客户端等待请求响应的最长时间。如果在超时之前没有收到响应，客户端将在需要时重新发送请求（如果重试用时失败）。	40000	整数
camel.component.kafka.configuration.consumer-streams	消费者上的并发用户数	10	整数
camel.component.kafka.configuration.consumers-count	连接到 kafka 服务器的消费者数量	1	整数
camel.component.kafka.configuration.enable-idempotence	如果设置为 'true'，则生成者将确保每个消息的一个副本用流写入。如果 'false'，则生成者可能会重试流中重试消息的重复。如果设置为 true，这个选项需要 max.in.flight.requests.per.connection 设置为 1，并且重试不能为零，另外将 acks 设置为 'all'。	false	布尔值
camel.component.kafka.configuration.fetch-max-bytes	如果 fetch 请求的第一个非空分区中的第一个非空分区大于这个值，则服务器应返回的最大数据量。代理接受的最大消息大小通过 message.max.bytes (broker config)或 max.message.bytes (topic config)定义。请注意，消费者并行执行多个获取。	52428800	整数
camel.component.kafka.configuration.fetch-min-bytes	服务器应返回的最小数据量，用于获取请求。如果数据不足，请求将等待这么多的数据在回答请求前累积。	1	整数
camel.component.kafka.configuration.fetch-wait-max-ms	如果没有足够的数据来立即满足 fetch.min.bytes，则在回答获取请求前，服务器将阻止的最大时间量。	500	整数
camel.component.kafka.configuration.group-id	唯一标识此消费者所属的消费者进程组的字符串。通过设置同一组 id 多个进程，表示它们都是同一消费者组的一部分。消费者需要这个选项。		字符串

Name	描述	默认值	类型
camel.component.kafka.configuration.header-filter-strategy	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
camel.component.kafka.configuration.heartbeat-interval-ms	使用 Kafka 的组管理功能时，与消费者协调器之间预期的时间。心跳用于确保消费者保持活跃状态，并在新用户加入或离开该组时促进重新平衡。该值必须小于 session.timeout.ms，但设置的值通常不应超过该值的 1/3。它可以调整甚至较低，以控制正常重新平衡的预期时间。	3000	整数
camel.component.kafka.configuration.interceptor-classes	为生成者或消费者设置拦截器。生产者拦截器必须是实施 org.apache.kafka.clients.producer.ProducerInterceptor Consumer 拦截器的类，必须是实现 org.apache.kafka.clients.consumer.ConsumerInterceptor.consumer.ConsumerInterceptor 注意，它会在运行时抛出类广播异常		字符串
camel.component.kafka.configuration.kafka-header-deserializer	设置自定义 KafkaHeaderDeserializer，将 deserialization kafka 标头值设置为 camel 标头值。		KafkaHeaderDeserializer
camel.component.kafka.configuration.kafka-header-serializer	设置自定义 KafkaHeaderDeserializer，将序列化 camel 标头值设置为 kafka 标头值。		KafkaHeaderSerializer
camel.component.kafka.configuration.kerberos-before-relogin-min-time	刷新尝试之间的登录线程睡眠时间。	60000	整数
camel.component.kafka.configuration.kerberos-init-cmd	Kerberos kinit 命令路径。默认为 /usr/bin/kinit	/usr/bin/kinit	字符串
camel.component.kafka.configuration.kerberos-principal-to-local-rules	从主体名称映射到短名称（通常是操作系统用户名）的规则列表。规则按顺序评估，第一个与主体名称匹配的规则用于将其映射到短名称。稍后列表中的任何规则都会被忽略。默认情况下，用户名/主机名 REALM的主体名称映射到 username。有关格式的详情，请参阅安全授权和 acls。可以使用逗号分隔多个值	DEFAULT	字符串

Name	描述	默认值	类型
camel.component.kafka.configuration.kerberos-renew-jitter	添加到续订时间的随机 jitter 的百分比。		å�Œ
camel.component.kafka.configuration.kerberos-renew-window-factor	登录线程将处于睡眠状态，直到达到最后一次刷新到票据到期的时间因素前处于睡眠状态，此时将尝试续订票据。		å�Œ
camel.component.kafka.configuration.key	记录键（如果未指定密钥，则为 null）。如果配置了这个选项，则它优先于标头 KafkaConstants#KEY		字符串
camel.component.kafka.configuration.key-deserializer	实现 Deserializer 接口的密钥的反序列化器类。	org.apache.kafka.common.serialization.StringDeserializer	字符串
camel.component.kafka.configuration.key-serializer-class	键的 serializer 类（如果没有提供，则默认为与消息相同）。	org.apache.kafka.common.serialization.StringSerializer	字符串
camel.component.kafka.configuration.linger-ms	生产者将请求传输之间到达单个批处理请求的任何记录组合在一起。通常，这只在记录到达速度快于发送时发生。然而，在某些情况下，客户端可能希望减少请求的数量，即使负载低下也是如此。此设置通过添加少量的 artificial 延迟来实现这一目的，而是不会立即发送记录，而是会等待到给定延迟，以允许发送其他记录一起批处理。这可能被视为与 TCP 中的 Nagle 算法类似。此设置会使批处理延迟有上限：一旦我们获得 batch.size，对于一个分区而言，无论此设置如何立即发送，则无论此设置如何，我们会立即发送的记录数超过这个分区所累计的字节数，我们将"闲置"，等待更多记录显示。此设置默认为 0 （例如，无延迟）。例如，设置 linger.ms=5 会影响减少发送的请求数，但会对负载发送的记录最多增加 5ms 的延迟。	0	整数

Name	描述	默认值	类型
camel.component.kafka.configuration.max-block-ms	配置控制发送到 kafka 的时长。这些方法可能会因为多种原因阻止。例如：缓冲区已满，元数据不可用。此配置对获取元数据、键和值序列化和值、分区和分配缓冲区内存的总时间施加最大限制，在执行 send （）时对缓冲区内存进行分区和分配。如果是 partitionsFor （），此配置会在等待元数据时产生最大时间阈值	60000	整数
camel.component.kafka.configuration.max-in-flight-request	客户端在阻止前在单个连接上发送的最大未确认请求数。请注意，如果此设置大于 1，且发送失败，则因为重试而出现消息重新排序的风险（例如，如果启用了重试）。	5	整数
camel.component.kafka.configuration.max-partition-fetch-bytes	服务器将返回的每个分区的最大数据量。用于请求的最大内存总量为 #partitions max.partition.fetch.bytes。此大小必须至少与服务器允许的最大消息大小相同，或者生产者可以发送消息大于消费者是否可以获取。如果发生这种情况，消费者会卡住在某个分区上获取大型消息。	1048576	整数
camel.component.kafka.configuration.max-poll-interval-ms	使用消费者组管理时，poll （）调用之间的最大延迟。这会在获取更多记录前将上限放在消费者可以闲置的时间长度。如果在这个超时时间前没有调用 poll （），则消费者被视为失败，并且组将重新平衡，以便将分区重新分配给另一个成员。		Long
camel.component.kafka.configuration.max-poll-records	在单个调用中返回的最大记录数（）	500	整数
camel.component.kafka.configuration.max-request-size	请求的最大大小。这实际上也是最大记录大小的上限。请注意，服务器对记录大小有自己的上限，可能与此不同。此设置将限制制作者将在单个请求中发送的记录批处理数量，以避免发送大量请求。	1048576	整数
camel.component.kafka.configuration.metadata-max-age-ms	即使我们未看到任何分区领导力更改来主动发现任何新的代理或分区，我们才会强制刷新元数据的时间（以毫秒为单位）。	300000	整数
camel.component.kafka.configuration.metric-reporters	用作指标报告器的类列表。实施 MetricReporter 接口允许插入新指标创建通知的类。总是包括 JmxReporter 来注册 JMX 统计信息。		字符串

Name	描述	默认值	类型
camel.component.kafka.configuration.metrics-sample-window-ms	为计算指标维护的示例数量。	30000	整数
camel.component.kafka.configuration.no-of-metrics-sample	为计算指标维护的示例数量。	2	整数
camel.component.kafka.configuration.offset-repository	使用偏移存储库来本地存储主题的每个分区的偏移量。定义一个将禁用自动提交。		StateRepository
camel.component.kafka.configuration.partition-assignor	使用组管理时，客户端将在消费者实例之间分发分区所有权的策略的类名称。	org.apache.kafka.clients.consumer.RangeAssignor	字符串
camel.component.kafka.configuration.partition-key	将记录发送到的分区（如果没有指定分区，则为 null）。如果配置了这个选项，则它优先于标头 KafkaConstants#PARTITION_KEY		整数
camel.component.kafka.configuration.partitioner	在子主题间分区消息的分区类。默认分区器基于密钥的哈希。	org.apache.kafka.clients.producer.internals.DefaultPartitioner	字符串
camel.component.kafka.configuration.poll-timeout-ms	轮询 KafkaConsumer 时使用的超时。	5000	Long

Name	描述	默认值	类型
camel.component.kafka.configuration.producer-batch-size	每当将多个记录发送到同一分区时，生成者将尝试将记录一起批处理到较少的请求。这有助于客户端和服务器的性能。此配置控制默认批处理大小（以字节为单位）。对于大于此 size.Requests 的批处理记录不会进行任何尝试。发送到代理的请求将包含多个批处理，每个分区一个可用数据。小批处理大小将较少的批处理，并且可能会降低吞吐量（批量大小为零将完全禁用批处理）。非常大的批处理大小可能会更严重地使用内存，因为我们将始终在附加记录下分配指定批处理大小的缓冲。	16384	整数
camel.component.kafka.configuration.queue-buffering-max-messages	在使用 async 模式时，可以在生成者必须被阻止或数据丢弃前，可以排队生成者的最大未发送消息数。	10000	整数
camel.component.kafka.configuration.receive-buffer-bytes	读取数据时要使用的 TCP 接收缓冲区(SO_RCVBUF)的大小。	65536	整数
camel.component.kafka.configuration.reconnect-backoff-max-ms	重新连接到重复连接失败的代理时等待的最大时间（以毫秒为单位）。如果提供，每个主机的 backoff 将为每个连续的连接失败指数增加，直到最高值。在计算 backoff 增长后，添加了 20% 的随机 jitter，以避免连接状况。	1000	整数
camel.component.kafka.configuration.reconnect-backoff-ms	尝试重新连接给定主机前需要等待的时间。这可避免在紧密循环中重复连接到主机。此 backoff 适用于消费者向代理发送的所有请求。	50	整数
camel.component.kafka.configuration.record-metadata	制作者是否应该存储发送到 Kafka 的 RecordMetadata 结果。结果存储在包含 RecordMetadata 元数据的列表中。该列表存储在带有键 KafkaConstantsKAFKA_RECORDMETA 的标头中	true	布尔值

Name	描述	默认值	类型
camel.component.kafka.configuration.request-required-acks	在考虑请求完成前，生成者需要收到的确认数量。这控制发送的记录的持久性。以下设置是常见的：acks=0 如果设为零，则生成者不会等待服务器中的任何确认。记录将立即添加到套接字缓冲区中并被视为发送。在这种情况下，不能保证服务器收到记录，重试配置不会生效（因为客户端通常不知道任何故障）。每个记录返回的偏移量始终设置为 -1 acks=1，这意味着领导机会将记录写入到其本地日志，但不会等待所有后续者的完全确认。在这种情况下，领导机在确认记录后马上失败，但在后续者复制记录之前，该记录将会丢失。acks=all 意味着领导机将等待全部同步副本确认记录。这样可保证记录在至少一个同步副本仍然处于活动状态时不会丢失。这是最强的可用保证。	1	字符串
camel.component.kafka.configuration.request-timeout-ms	代理在向客户端发送错误前尝试满足 request.required.acks 要求的时间长度。	305000	整数
camel.component.kafka.configuration.retries	设置大于零的值将导致客户端重新发送发送失败的记录，并显示潜在的临时错误。请注意，这个重试与在收到错误时重新记录不同的是不同的。允许重试可能会更改记录顺序，因为如果将两个记录发送到单个分区，第一次失败并重试，但会成功，但第二个记录可能会首先出现。	0	整数
camel.component.kafka.configuration.retry-backoff-ms	在每个重试前，生成者会刷新相关主题的元数据，以查看是否选择了新的领导。由于领导选举需要一些时间，此属性指定制作者在刷新元数据前等待的时间。	100	整数
camel.component.kafka.configuration.sasl-jaas-config	公开 kafka sasl.jaas.config 参数示例：org.apache.kafka.common.security.plain.PlainLoginModule required username=USERNAME password=PASSWORD;		字符串
camel.component.kafka.configuration.sasl-kerberos-service-name	Kafka 运行的 Kerberos 主体名称。这可以在 Kafka 的 JAAS 配置或 Kafka 配置中定义。		字符串
camel.component.kafka.configuration.sasl-mechanism	使用简单身份验证和安全层(SASL)机制。如需有效值，请参阅 http://www.iana.org/assignments/sasl-mechanisms/sasl-mechanisms.xhtml	GSSAPI	字符串

Name	描述	默认值	类型
camel.component.kafka.configuration.security-protocol	用于与代理通信的协议。目前只支持 PLAINTEXT 和 SSL。	PLAINTEXT	字符串
camel.component.kafka.configuration.seek-to	set if KafkaConsumer will read from start or end on start: beginning : read from beginning end : read from end this is replace the earlier property seekToBeginning		字符串
camel.component.kafka.configuration.send-buffer-bytes	套接字写入缓冲区大小	131072	整数
camel.component.kafka.configuration.serializer-class	用于消息的 serializer 类。	org.apache.kafka.common.serialization.StringSerializer	字符串
camel.component.kafka.configuration.session-timeout-ms	使用 Kafka 组管理工具时用于检测故障的超时。	10000	整数
camel.component.kafka.configuration.ssl-cipher-suites	密码套件列表。这是用来使用 TLS 或 SSL 网络协议协商网络连接的安全设置的身份验证、加密、MAC 和密钥交换算法的命名组合。只要默认支持所有可用的密码套件。		字符串
camel.component.kafka.configuration.ssl-context-parameters	使用 Camel SSLContextParameters 对象的 SSL 配置。如果配置在其他 SSL 端点参数之前应用。		SSLContextParameters
camel.component.kafka.configuration.ssl-enabled-protocols	为 SSL 连接启用的协议列表。TLSv1.2、TLSv1.1 和 TLSv1 会被默认启用。	TLSv1.2,TLSv1.1,TLSv1	字符串
camel.component.kafka.configuration.ssl-endpoint-algorithm	使用服务器证书验证服务器主机名的端点标识算法。		字符串

Name	描述	默认值	类型
camel.component.kafka.configuration.ssl-key-password	密钥存储文件中私钥的密码。这对客户端是可选的。		字符串
camel.component.kafka.configuration.ssl-keymanager-algorithm	密钥管理器工厂用于 SSL 连接的算法。默认值是为 Java 虚拟机配置的密钥管理器工厂算法。	SunX509	字符串
camel.component.kafka.configuration.ssl-keystore-location	密钥存储文件的位置。这对客户端是可选的，可用于客户端进行双向身份验证。		字符串
camel.component.kafka.configuration.ssl-keystore-password	密钥存储文件的存储密码。这对客户端是可选的，只有在配置了 ssl.keystore.location 时才需要。		字符串
camel.component.kafka.configuration.ssl-keystore-type	密钥存储文件的文件格式。这对客户端是可选的。默认值为 JKS	JKS	字符串
camel.component.kafka.configuration.ssl-protocol	用于生成 SSLContext 的 SSL 协议。默认设置为 TLS，对于大多数情况来说是正常的。最近 JVM 中允许的值是 TLS、TLSv1.1 和 TLSv1.2。较旧的 JVM 中可能会支持 SSL、SSLv2 和 SSLv3，但由于已知的安全漏洞，不建议使用它们。	TLS	字符串
camel.component.kafka.configuration.ssl-provider	用于 SSL 连接的安全供应商的名称。默认值是 JVM 的默认安全提供程序。		字符串
camel.component.kafka.configuration.ssl-trustmanager-algorithm	信任管理器工厂用于 SSL 连接的算法。默认值是为 Java 虚拟机配置的信任管理器工厂算法。	PKIX	字符串
camel.component.kafka.configuration.ssl-truststore-location	信任存储文件的位置。		字符串

Name	描述	默认值	类型
camel.component.kafka.configuration.ssl-truststore-password	信任存储文件的密码。		字符串
camel.component.kafka.configuration.ssl-truststore-type	信任存储文件的文件格式。默认值为 JKS。	JKS	字符串
camel.component.kafka.configuration.topic	要使用的主题名称。在消费者中，您可以使用逗号分隔多个主题。生产者只能发送消息到单个主题。		字符串
camel.component.kafka.configuration.topic-is-pattern	主题是否为模式（正则表达式）。这可用于订阅与模式匹配的主题数量。	false	布尔值
camel.component.kafka.configuration.value-deserializer	deserializer 类，用于实现 Deserializer 接口的值。	org.apache.kafka.common.serialization.StringDeserializer	字符串
camel.component.kafka.configuration.worker-pool	要在 kafka 服务器确认使用异步非阻塞处理从 KafkaProducer 发送到它的消息后，使用自定义 worker 池来继续路由 Exchange。		ExecutorService
camel.component.kafka.configuration.worker-pool-core-size	kafka 服务器确认从 KafkaProducer 发送到 KafkaProducer 的消息后，用于继续路由 Exchange 的 worker 池的核心线程数量，使用异步非阻塞处理。	10	整数
camel.component.kafka.configuration.worker-pool-max-size	kafka 服务器确认从 KafkaProducer 发送到 KafkaProducer 的消息后，用于继续路由 Exchange 的 worker 池的最大线程数量。	20	整数
camel.component.kafka.enabled	启用 kafka 组件	true	布尔值

Name	描述	默认值	类型
camel.component.kafka.kafka-manual-commit-factory	用于创建 KafkaManualCommit 实例的工厂。这允许在进行手动提交时，需要自定义 KafkaManualCommit 实例来插件自定义 KafkaManualCommit 实例。选项是 org.apache.camel.component.kafka.KafkaManualCommitFactory 类型。		字符串
camel.component.kafka.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.kafka.shutdown-timeout	超时时间（以毫秒为单位）会安全等待消费者或制作者关闭并终止其 worker 线程。	30000	整数
camel.component.kafka.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值
camel.component.kafka.worker-pool	要在 kafka 服务器确认通过异步非阻塞处理从 KafkaProducer 发送到它的消息后，使用共享自定义 worker 池继续路由 Exchange。如果使用这个选项，则必须处理线程池的生命周期，以便在不再需要时关闭池。选项是一个 java.util.concurrent.ExecutorService 类型。		字符串

有关 Producer/Consumer 配置的更多信息：

http://kafka.apache.org/documentation.html#newconsumerconfigshttp://kafka.apache.org/documentation.html#producerconfigs

191.4. 消息标头

191.4.1. 消费者标头

在消耗 Kafka 的消息时，以下标头可用。

标头常数	标头值	类型	描述
KafkaConstants .TOPIC	**"kafka.TOPIC"**	**字符串**	消息源自的主题
KafkaConstants .PARTITION	**"kafka.PARTITION"**	**整数**	保存消息的分区
KafkaConstants .OFFSET	**"kafka.OFFSET"**	**Long**	消息的偏移
KafkaConstants .KEY	**"kafka.KEY"**	**对象**	如果配置，消息的键
KafkaConstants .HEADERS	**"kafka.HEADERS"**	**org.apache.kafka.common.header.Headers**	记录标头
KafkaConstants .LAST_RECORD_BEFORE_COMMIT	**"kafka.LAST_RECORD_BEFORE_COMMIT"**	**布尔值**	提交前是否是最后一个记录（仅在 **autoCommitEnable** endpoint 参数为 **false**时才可用）
KafkaConstants .MANUAL_COMMIT	**"CamelKafkaManualCommit"**	**KafkaManualCommit**	在使用 Kafka 使用者时，可用于强制手动偏移提交。

191.4.2. 制作者标头

在向 Kafka 发送消息前，您可以配置以下标头。

标头常数	标头值	类型	描述
KafkaConstants .KEY	**"kafka.KEY"**	**对象**	**必需** 消息的密钥，以确保所有相关消息都在同一分区中
KafkaConstants .TOPIC	**"kafka.TOPIC"**	**字符串**	发送消息的主题（仅当 **bridgeEndpoint** 端点参数为 **true**时读取）

标头常数	标头值	类型	描述
KafkaConstants.PARTITION_KEY	**"kafka.PARTITION_KEY"**	**整数**	明确指定分区（仅在定义了 **KafkaConstants.KEY** 标头时使用）

将消息发送到 Kafka 后，以下标头可用

标头常数	标头值	类型	描述
KafkaConstants.KAFKA_RECORDMETA	**"org.apache.kafka.clients.producer.RecordMetadata"**	**List<RecordMetadata>**	元数据（仅在 **recordMetadata** endpoint 参数为 **true**时配置

191.5. SAMPLES

191.5.1. 消耗 Kafka 的消息

以下是从 Kafka 读取信息所需的最小路由。

```
from("kafka:test?brokers=localhost:9092")
    .log("Message received from Kafka : ${body}")
    .log("    on the topic ${headers[kafka.TOPIC]}")
    .log("    on the partition ${headers[kafka.PARTITION]}")
    .log("    with the offset ${headers[kafka.OFFSET]}")
    .log("    with the key ${headers[kafka.KEY]}")
```

如果您需要消耗来自多个主题的消息，您可以使用以逗号分隔的主题名称列表

```
from("kafka:test,test1,test2?brokers=localhost:9092")
    .log("Message received from Kafka : ${body}")
    .log("    on the topic ${headers[kafka.TOPIC]}")
    .log("    on the partition ${headers[kafka.PARTITION]}")
    .log("    with the offset ${headers[kafka.OFFSET]}")
    .log("    with the key ${headers[kafka.KEY]}")
```

当消耗来自 Kafka 的消息时，您可以使用自己的偏移管理，而不将这个管理委派给 Kafka。为了保持偏移量，组件需要 StateRepository 实施，如 FileStateRepository。此 bean 应在 registry 中提供。在这里如何使用它：

```
// Create the repository in which the Kafka offsets will be persisted
```

```
FileStateRepository repository = FileStateRepository.fileStateRepository(new
File("/path/to/repo.dat"));

// Bind this repository into the Camel registry
JndiRegistry registry = new JndiRegistry();
registry.bind("offsetRepo", repository);

// Configure the camel context
DefaultCamelContext camelContext = new DefaultCamelContext(registry);
camelContext.addRoutes(new RouteBuilder() {
    @Override
    public void configure() throws Exception {
        from("kafka:" + TOPIC + "?brokers=localhost:{{kafkaPort}}" +
                // Setup the topic and broker address
                "&groupId=A" +
                // The consumer processor group ID
                "&autoOffsetReset=earliest" +
                // Ask to start from the beginning if we have unknown offset
                "&offsetRepository=#offsetRepo")
                // Keep the offsets in the previously configured repository
            .to("mock:result");
    }
});
```

191.5.2. 将消息生成到 Kafka

以下是将信息写入 Kafka 所需的最小路由。

```
from("direct:start")
    .setBody(constant("Message from Camel"))         // Message to send
    .setHeader(KafkaConstants.KEY, constant("Camel")) // Key of the message
    .to("kafka:test?brokers=localhost:9092");
```

191.6. SSL 配置

您可以使用两种不同的方法在 Kafka 组件中配置 SSL 通信。

第一种方法是通过多个 SSL 端点参数

```
from("kafka:" + TOPIC + "?brokers=localhost:{{kafkaPort}}" +
            "&groupId=A" +
            "&sslKeystoreLocation=/path/to/keystore.jks" +
            "&sslKeystorePassword=changeit" +
            "&sslKeyPassword=changeit" +
            "&securityProtocol=SSL")
        .to("mock:result");
```

第二种方法是使用 sslContextParameters 端点参数。

```
// Configure the SSLContextParameters object
KeyStoreParameters ksp = new KeyStoreParameters();
ksp.setResource("/path/to/keystore.jks");
ksp.setPassword("changeit");
KeyManagersParameters kmp = new KeyManagersParameters();
kmp.setKeyStore(ksp);
kmp.setKeyPassword("changeit");
SSLContextParameters scp = new SSLContextParameters();
scp.setKeyManagers(kmp);

// Bind this SSLContextParameters into the Camel registry
JndiRegistry registry = new JndiRegistry();
registry.bind("ssl", scp);

// Configure the camel context
DefaultCamelContext camelContext = new DefaultCamelContext(registry);
camelContext.addRoutes(new RouteBuilder() {
    @Override
    public void configure() throws Exception {
        from("kafka:" + TOPIC + "?brokers=localhost:{{kafkaPort}}" +
                // Setup the topic and broker address
                "&groupId=A" +
                // The consumer processor group ID
                "&sslContextParameters=#ssl" +
                // The security protocol
                "&securityProtocol=SSL)
                // Reference the SSL configuration
             .to("mock:result");
    }
});
```

191.7. 使用 KAFKA IDEMPOTENT 软件仓库

Camel 2.19 中可用

camel-kafka 库提供基于 Kafka 主题的幂等存储库。此存储库将广播到 Kafka 主题中的幂等状态（添加/删除）的所有更改，并通过事件源为每个存储库的进程实例填充本地内存缓存。

使用的主题每个幂等存储库实例必须是唯一的。机制对主题分区的数量没有任何要求，因为存储库同时消耗所有分区。它还对主题的复制因素没有任何要求。

使用主题的每个存储库实例（例如，通常在并行运行的不同机器上）控制其自己的消费者组，因此使用相同主题的 10 个 Camel 进程集群中都会控制自己的偏移量。

在启动时，实例订阅了主题，并将偏移回开始，将缓存重建到最新的状态。缓存不会被视为温设置，直到一个轮询 pollDurationMs （以长度为 0 条）返回 0 记录。在缓存已温启动或 30 秒结束前，启动不会完成；如果后者发生幂等存储库可能处于不一致状态，直到消费者捕获到主题的末尾为止。

KafkaIdempotentRepository 具有以下属性：

属性	描述
topic	用于广播更改的 Kafka 主题的名称（必需）
bootstrapServers	内部 Kafka producer 和消费者上的 **bootstrap.servers** 属性。如果没有设置 **consumerConfig** 和 **producerConfig**，则使用此选项作为简写。如果使用，此组件将应用生产者和消费者的默认配置。
producerConfig	设置用于广播更改的 Kafka producer 使用的属性。覆盖 **bootstrapServers**，因此必须定义 Kafka **bootstrap.servers** 属性本身
consumerConfig	设置 Kafka 使用者将使用的属性，从主题填充缓存。覆盖 **bootstrapServers**，因此必须定义 Kafka **bootstrap.servers** 属性本身
maxCacheSize	最近使用多少密钥应存储在内存中（默认值 1000）。
pollDurationMs	Kafka 使用者的轮询持续时间。本地缓存会立即更新。这个值将影响从主题更新其缓存的其他对等点的时长，相对于发送缓存操作消息的幂等消费者实例。默认值为 100 ms。 如果明确设置这个值，请注意，远程缓存存活度和此存储库的消费者和 Kafka 代理之间的网络流量卷之间存在权衡。缓存温进程还取决于有一个无法获取的轮询 - 这表示流已消耗至当前点。如果轮询持续时间太长，对于在主题上发送消息的速率过长，则存在缓存可能无法被温的，并且将在其对等点相对处于不一致的状态进行操作。

可以通过定义 主题和 bootstrapServers、或 producerConfig 和 consumerConfig 属性集来实例化存储库，以启用 SSL/SASL 等功能。

要使用，此存储库必须放在 Camel registry 中，也可以手动或通过注册作为 Spring/Blueprint 中的 bean，因为它是 CamelContext aware。

示例用法如下：

```
KafkaIdempotentRepository kafkaIdempotentRepository = new
KafkaIdempotentRepository("idempotent-db-inserts", "localhost:9091");
```

```
SimpleRegistry registry = new SimpleRegistry();
registry.put("insertDbIdemRepo", kafkaIdempotentRepository); // must be registered in the
registry, to enable access to the CamelContext
CamelContext context = new CamelContext(registry);

// later in RouteBuilder...
from("direct:performInsert")
    .idempotentConsumer(header("id")).messageIdRepositoryRef("insertDbIdemRepo")
        // once-only insert into database
    .end()
```

在 XML 中：

```
<!-- simple -->
<bean id="insertDbIdemRepo"
  class="org.apache.camel.processor.idempotent.kafka.KafkaIdempotentRepository">
  <property name="topic" value="idempotent-db-inserts"/>
  <property name="bootstrapServers" value="localhost:9091"/>
</bean>

<!-- complex -->
<bean id="insertDbIdemRepo"
  class="org.apache.camel.processor.idempotent.kafka.KafkaIdempotentRepository">
  <property name="topic" value="idempotent-db-inserts"/>
  <property name="maxCacheSize" value="10000"/>
  <property name="consumerConfig">
    <props>
      <prop key="bootstrap.servers">localhost:9091</prop>
    </props>
  </property>
  <property name="producerConfig">
    <props>
      <prop key="bootstrap.servers">localhost:9091</prop>
    </props>
  </property>
</bean>
```

191.8. 在 KAFKA 使用者中使用手动提交

从 Camel 2.21 开始提供

默认情况下，Kafka 使用者将使用自动提交，其中偏移将使用给定间隔在后台自动提交。

如果要强制手动提交，您可以使用 Camel Exchange 中的 KafkaManualCommit API，存储在消息标头中。这要求通过将 KafkaComponent 或端点上的 allowManualCommit 设置为 true 来打开手动提交，例如：

```
KafkaComponent kafka = new KafkaComponent();
kafka.setAllowManualCommit(true);
...
camelContext.addComponent("kafka", kafka);
```

然后，您可以使用 Java 代码（如 Camel 处理器）的 KafkaManualCommit ：

```
public void process(Exchange exchange) {
    KafkaManualCommit manual =
        exchange.getIn().getHeader(KafkaConstants.MANUAL_COMMIT,
KafkaManualCommit.class);
    manual.commitSync();
}
```

这将强制进行同步提交，它将阻止到提交在 Kafka 上确认，或者抛出异常。

如果要使用 KafkaManualCommit 的自定义实现，您可以在 KafkaComponent 中配置自定义 KafkaManualCommitFactory，用于创建自定义实现实例。

191.9. KAFKA HEADERS 传播

从 Camel 2.22 开始提供

当消耗 Kafka 的消息时，标头会自动传播到 camel Exchange 标头。由同一行为支持的生成流 - 特定交换的 camel 标头将传播到 kafka 消息标头。

因为 kafka 标头只允许 byte[] 值，以便 camel exchnage 标头传播其值应该被序列化为 bytes[]，否则标头会被跳过。支持以下标头值类型： 字符串,Integer,Long, Double, 布尔值,byte[]。注： 所有标头生成的 从 kafka 到 camel 的交换默认都会包括值 byte[]。要覆盖默认功能 uri 参数，可以为 route 和 kafkaHeaderSerializer 设置为 route: kafkaHeaderDeserializer。 Example:

```
from("kafka:my_topic?kafkaHeaderDeserializer=#myDeserializer")
...
.to("kafka:my_topic?kafkaHeaderSerializer=#mySerializer")
```

默认情况下，所有标头都由 KafkaHeaderFilterStrategy 过滤。策略过滤出以 Camel 或 org.apache.camel 前缀开头的标头。默认的策略可以通过在 to 和 from 路由中使用 headerFilterStrategy uri 参数进行覆盖：

```
from("kafka:my_topic?headerFilterStrategy=#myStrategy")
...
.to("kafka:my_topic?headerFilterStrategy=#myStrategy")
```

myStrategy 对象应该是 HeaderFilterStrategy 的子类，且必须手动放置到 Camel registry 中，或者通过注册作为 Spring/Blueprint 中的 bean，因为它是 CamelContext aware。

第 192 章 KESTREL 组件（已弃用）

从 Camel 版本 2.6 开始提供

Kestrel 组件允许消息发送到 Kestrel 队列，或从 Kestrel 队列使用消息。此组件使用 spymemcached 客户端与 Kestrel 服务器进行 memcached 协议通信。

警告

kestrel 项目不活跃，Camel 团队将此组件视为已弃用。

192.1. URI 格式

```
kestrel://[addresslist/]queuename[?options]
```

其中 queuename 是 Kestrel 上队列的名称。URI 的 addresslist 部分可能包含一个或多个 host:port 对。例如，要连接到 kserver01:22133 上的队列 foo，请使用：

```
kestrel://kserver01:22133/foo
```

如果省略了 addresslist，则假定为 localhost:22133，例如：

```
kestrel://foo
```

同样，如果在 addresslist 中的 host:port 对中省略了端口，则假设默认端口 22133，即：

```
kestrel://kserver01/foo
```

以下是用于生成集群队列的 Kestrel 端点 URI 示例：

```
kestrel://kserver01:22133,kserver02:22133,kserver03:22133/massive
```

以下是用于从队列同时消耗的 Kestrel 端点 URI 示例：

```
kestrel://kserver03:22133/massive?concurrentConsumers=25&waitTimeMs=500
```

192.2. 选项

Kestrel 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	使用共享配置作为基础来创建新端点。		KestrelConfiguration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Kestrel 端点使用 URI 语法进行配置：

```
kestrel:addresses/queue
```

使用以下路径和查询参数：

192.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
addresses	运行 kestrel 的地址	localhost:22133	string[]
queue	**必需的** 队列，我们正在轮询		字符串

192.2.2. 查询参数(6 参数)：

Name	描述	默认值	类型
concurrentConsumers (common)	为线程池调度的并发监听程序数量	1	int

Name	描述	默认值	类型
waitTimeMs (common)	给定等待应阻塞（服务器端）的时间（以毫秒为单位）	100	int
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

192.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.component.kestrel.configuration.addresses	运行 kestrel 的地址		string[]
camel.component.kestrel.configuration.concurrent-consumers	为线程池调度的并发监听程序数量	1	整数
camel.component.kestrel.configuration.wait-time-ms	给定等待应阻塞（服务器端）的时间（以毫秒为单位）	100	整数
camel.component.kestrel.enabled	启用 kestrel 组件	true	布尔值

Name	描述	默认值	类型
camel.component.kestrel.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

192.4. 使用 SPRING XML 配置 KESTREL 组件

最简单的显式配置形式如下：

```
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="
      http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
      http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd">

  <bean id="kestrel" class="org.apache.camel.component.kestrel.KestrelComponent"/>

  <camelContext xmlns="http://camel.apache.org/schema/spring">
  </camelContext>

</beans>
```

这将启用带有所有默认设置的 Kestrel 组件，即它将使用 localhost:22133、100ms 等待时间和单个非并发消费者。

要使用基本配置中的特定选项（向未指定 ?properties 的端点提供配置），您可以设置一个 KestrelConfiguration POJO，如下所示：

```
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="
      http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
      http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd">

  <bean id="kestrelConfiguration" class="org.apache.camel.component.kestrel.KestrelConfiguration">
    <property name="addresses" value="kestrel01:22133"/>
    <property name="waitTimeMs" value="100"/>
    <property name="concurrentConsumers" value="1"/>
  </bean>
```

```
  <bean id="kestrel" class="org.apache.camel.component.kestrel.KestrelComponent">
    <property name="configuration" ref="kestrelConfiguration"/>
  </bean>

  <camelContext xmlns="http://camel.apache.org/schema/spring">
  </camelContext>

</beans>
```

192.5. 使用示例

192.5.1. 示例 1：消耗

```
from("kestrel://kserver02:22133/massive?concurrentConsumers=10&waitTimeMs=500")
  .bean("myConsumer", "onMessage");
```

```
public class MyConsumer {
    public void onMessage(String message) {
        ...
    }
}
```

192.5.2. 示例 2: Producing

```
public class MyProducer {
    @EndpointInject(uri = "kestrel://kserver01:22133,kserver02:22133/myqueue")
    ProducerTemplate producerTemplate;

    public void produceSomething() {
        producerTemplate.sendBody("Hello, world.");
    }
}
```

192.5.3. 示例 3：Spring XML 配置

```
  <camelContext xmlns="http://camel.apache.org/schema/spring">
    <route>
      <from uri="kestrel://ks01:22133/sequential?concurrentConsumers=1&waitTimeMs=500"/>
      <bean ref="myBean" method="onMessage"/>
    </route>
    <route>
      <from uri="direct:start"/>
      <to uri="kestrel://ks02:22133/stuff"/>
    </route>
  </camelContext>
```

```
public class MyBean {
    public void onMessage(String message) {
        ...
```

```
    }
  }
```

192.6. 依赖项

Kestrel 组件有以下依赖项：

- spymemcached 2.5 （或更多）

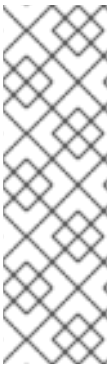
192.6.1. spymemcached

您必须在 classpath 上具有 spymemcached jar。以下是您可以在 pom.xml 中使用的一个代码片段：

```
<dependency>
  <groupId>spy</groupId>
  <artifactId>memcached</artifactId>
  <version>2.5</version>
</dependency>
```

或者，您可以直接 下载 jar。

警告：限制



注意

当启用 JVM 断言时，spymemcached 客户端库 无法正常工作。当启用断言时，spymemcached 有一个已知问题，一个请求的密钥包含 /t=... 扩展（例如，如果您在端点 URI 中使用 waitTimeMs 选项，则强烈建议您这样做）。除非明确 启用了它们，否则默认禁用 JVM 断言，因此在正常情况下，这应该不会出现问题。需要注意的是，Maven 的 Surefire 测试插件 启用了 断言。如果您在 Maven 测试环境中使用此组件，您可能需要将 enableAssertions 设置为 false。有关详细信息，请参阅 surefire:test 参考。

192.7. 另请参阅

- 配置 Camel

- *组件*
- *端点*
- *开始使用*

第 193 章 KIE-CAMEL

193.1. 概述

kie-camel 组件是由红帽 Fuse 提供的 Apache Camel 端点，它将 Fuse 与 Red Hat Process Automation Manager 集成。它允许您使用 Maven 组 ID、工件 ID 和版本(GAV)标识符来指定 Red Hat Process Automation Manager 模块，您可以拉取到路由并执行。它还允许您将消息正文的部分指定为事实。您可以将 kie-camel 组件与嵌入式引擎或进程服务器一起使用。

有关 kie-camel 组件的详情，请参阅将 Red Hat Fuse 与 Red Hat Process Automation Manager 集成。

第 194 章 KRATI 组件（已弃用）

从 Camel 版本 2.9 开始提供

此组件允许在 Camel 中使用 krati 数据存储和数据集。Krati 是一个简单的持久数据存储，延迟非常低，高吞吐量。它旨在与读写密集型应用程序轻松集成，在调优配置、性能和 JVM 垃圾回收方面很少的工作。

Camel 为 krati datastore_(key/value engine)_ 提供制作者和消费者。它还提供用于过滤重复的消息的幂等存储库。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-krati</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

194.1. URI 格式

```
krati:[the path of the datastore][?options]
```

数据存储的路径是 krati 将用于数据存储的文件夹的相对路径。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

194.2. KRATI 选项

Krati 组件没有选项。

Krati 端点使用 URI 语法进行配置：

```
krati:path
```

使用以下路径和查询参数：

194.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
path	数据存储 **所需的** 路径是 krati 将用于数据存储的文件夹的相对路径。		字符串

194.2.2. 查询参数(29 参数)：

Name	描述	默认值	类型
hashFunction (common)	要使用的 hash 功能。		HashFunction
initialCapacity (common)	存储的 inital capcity。	100	int
keySerializer (common)	用于序列化密钥的序列化器。		serializer
segmentFactory (common)	设置目标存储的网段工厂。		SegmentFactory
segmentFileSize (common)	数据存储片段大小（以 MB 为单位）。	64	int
valueSerializer (common)	用于序列化值的序列化器。		serializer
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
maxMessagesPerPoll (consumer)	在一个轮询中接收的最大消息数。这可用于避免读取太多数据并占用太多内存。		int
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值

Name	描述	默认值	类型
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
key (producer)	密钥。		字符串
operation (producer)	指定将对数据存储执行的操作类型。		字符串
value (producer)	价值.		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long

Name	描述	默认值	类型
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

194.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.krati.enabled	启用 krati 组件	true	布尔值
camel.component.krati.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

```
krati:/tmp/krati?
operation=CamelKratiGet&initialCapacity=10000&keySerializer=#myCustomSerializer
```

对于制作者端点，您可以通过将适当的标头传递给消息来覆盖上述所有 URI 选项。

194.3.1. 用于数据存储的消息标头

标头	描述
CamelKratiOperation	要在数据存储上执行的操作。有效选项包括 CamelKratiAdd, CamelKratiGet, CamelKratiDelete, CamelKratiDeleteAll
CamelKratiKey	密钥。
CamelKratiValue	该值。

194.4. 使用示例

194.4.1. 示例 1：计算数据存储.

本例将演示如何将任何消息存储在数据存储中。

```
from("direct:put").to("krati:target/test/producertest");
```

在上例中，您可以使用消息上的标头覆盖任何 URI 参数。
以下是上述示例如何使用 xml 定义路由：

```
<route>
    <from uri="direct:put"/>
    <to uri="krati:target/test/producerspringtest"/>
</route>
```

194.4.2. 示例 2：从数据存储中获取/读取

本例将演示如何读取数据存储的 contnet。

```
from("direct:get")
    .setHeader(KratiConstants.KRATI_OPERATION,
constant(KratiConstants.KRATI_OPERATION_GET))
    .to("krati:target/test/producertest");
```

在上例中，您可以使用消息上的标头覆盖任何 URI 参数。

以下是上述示例如何使用 xml 定义路由：

```
<route>
    <from uri="direct:get"/>
    <to uri="krati:target/test/producerspringtest?operation=CamelKratiGet"/>
</route>
```

194.4.3. 示例 3：从数据存储中消耗

本例将使用指定数据存储下的所有项目。

```
    from("krati:target/test/consumertest")
        .to("direct:next");
```

您可以使用 xml 实现相同的目标，如下所示。

```
<route>
    <from uri="krati:target/test/consumerspringtest"/>
    <to uri="mock:results"/>
</route>
```

194.5. 幂等存储库

如前文所述，该组件还提供了 和 idemptonet 存储库，可用于过滤重复的消息。

```
from("direct://in").idempotentConsumer(header("messageId"), new
KratiIdempotentRepositroy("/tmp/idempotent").to("log://out");
```

194.5.1. 另请参阅

Krati 网站

第 195 章 KUBERNETES 组件

从 Camel 版本 2.17 开始提供

Kubernetes 组件将应用程序与 Kubernetes 独立或 Openshift 集成。

camel-kubernetes 由 13 个组件组成：

- Kubernetes ConfigMap
- Kubernetes 命名空间
- Kubernetes 节点
- Kubernetes 持久性卷
- Kubernetes 持久性卷声明
- Kubernetes Pod
- Kubernetes 复制控制器
- Kubernetes 资源配额
- Kubernetes Secret
- Kubernetes 服务帐户

- *Kubernetes Service*

在 OpenShift 中，还要：

- *Kubernetes 构建配置*
- *Kubernetes 构建*

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-kubernetes</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

195.1. HEADERS

Name	类型	描述
CamelKubernetesOperation	字符串	Producer 操作
CamelKubernetesNamespaceName	字符串	命名空间名称
CamelKubernetesNamespaceLabels	Map	命名空间标签

Name	类型	描述
CamelKubernetesServiceLabels	Map	Service 标签
CamelKubernetesServiceName	字符串	服务名称
CamelKubernetesServiceSpec	io.fabric8.kubernetes.api.model.ServiceSpec	服务的 Spec
CamelKubernetesReplicationControllersLabels	Map	复制控制器标签
CamelKubernetesReplicationControllerName	字符串	复制控制器名称
CamelKubernetesReplicationControllerSpec	io.fabric8.kubernetes.api.model.ReplicationControllerSpec	复制控制器的 Spec

Name	类型	描述
CamelKubernetesReplicationControllerReplicas	整数	在 Scale 操作过程中的 Replication Controller 的副本数
CamelKubernetesPodsLabels	Map	Pod 标签
CamelKubernetesPodName	字符串	Pod 名称
CamelKubernetesPodSpec	io.fabric8.kubernetes.api.model.PodSpec	Pod 的 Spec
CamelKubernetesPersistentVolumesLabels	Map	持久性卷标签
CamelKubernetesPersistentVolumesName	字符串	持久性卷名称
CamelKubernetesPersistentVolumesClaimsLabels	Map	持久性卷声明标签

Name	类型	描述
CamelKubernetesPersistentVolumesClaimsName	字符串	持久性卷声明名称
CamelKubernetesPersistentVolumesClaimsSpec	io.fabric8.kubernetes.api.model.PersistentVolumeClaimSpec	持久性卷声明的 Spec
CamelKubernetesSecretsLabels	Map	Secret 标签
CamelKubernetesSecretsName	字符串	Secret 名称
CamelKubernetesSecret	io.fabric8.kubernetes.api.model.Secret	Secret 对象
CamelKubernetesResourcesQuotaLabels	Map	资源配额标签

Name	类型	描述
CamelKubernetesResourcesQuotaName	字符串	资源配额名称
CamelKubernetesResourceQuotaSpec	io.fabric8.kubernetes.api.model.ResourceQuotaSpec	资源配额的 Spec
CamelKubernetesServiceAccountsLabels	Map	服务帐户标签
CamelKubernetesServiceAccountName	字符串	服务帐户名称
CamelKubernetesServiceAccount	io.fabric8.kubernetes.api.model.ServiceAccount	Service Account 对象
CamelKubernetesNodesLabels	Map	节点标签
CamelKubernetesNodeName	字符串	节点名

Name	类型	描述
CamelKubernetesBuildsLabels	Map	OpenShift 构建标签
CamelKubernetesBuildName	字符串	OpenShift 构建名称
CamelKubernetesBuildConfigsLabels	Map	OpenShift 构建配置标签
CamelKubernetesBuildConfigName	字符串	OpenShift 构建配置名称
CamelKubernetesEventAction	io.fabric8.kubernetes.client.Watcher.Action	消费者监视的操作
CamelKubernetesEventTimestamp	字符串	消费者监视的操作的时间戳
CamelKubernetesConfigMapName	字符串	ConfigMap 名称

Name	类型	描述
CamelKubernetesConfigMapsLabels	Map	ConfigMap 标签
CamelKubernetesConfigData	Map	ConfigMap 数据

195.2. 使用方法

195.2.1. 生成者示例

在这里，我们演示了一些使用 camel-kubernetes 的制作者示例。

195.2.2. 创建 pod

```
from("direct:createPod")
    .toF("kubernetes-pods://%s?oauthToken=%s&operation=createPod", host, authToken);
```

通过使用 KubernetesConstants.KUBERNETES_POD_SPEC 标头，您可以指定 PodSpec 并将其传递给此操作。

195.2.3. 删除 pod

```
from("direct:createPod")
    .toF("kubernetes-pods://%s?oauthToken=%s&operation=deletePod", host, authToken);
```

通过使用 KubernetesConstants.KUBERNETES_POD_NAME 标头，您可以指定 Pod 名称并将其传递给此操作。

第 196 章 KUBERNETES 组件（已弃用）

从 Camel 版本 2.17 开始提供

重要

复合 kubernetes 组件已弃用。按如下所示使用独立组件。

- Kubernetes 组件
  - Kubernetes 构建配置
  - Kubernetes 构建
  - Kubernetes ConfigMap
  - Kubernetes 命名空间
  - Kubernetes 节点
  - Kubernetes 持久性卷
  - Kubernetes 持久性卷声明
  - Kubernetes Pod
  - Kubernetes 复制控制器
  - Kubernetes 资源配额

- *Kubernetes Secret*
- *Kubernetes 服务帐户*
- *Kubernetes Service*

Kubernetes 组件是将应用程序与 Kubernetes 独立或 Openshift 集成的组件。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-kubernetes</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

196.1. URI 格式

```
kubernetes:masterUrl[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

196.2. 选项

Kubernetes 组件没有选项。

Kubernetes 端点使用 URI 语法进行配置：

```
kubernetes:masterUrl
```

使用以下路径和查询参数：

196.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
masterUrl	**所需的** Kubernetes API 服务器 URL		字符串

196.2.2. 查询参数(29 参数)：

Name	描述	默认值	类型
apiVersion (common)	要使用的 Kubernetes API 版本		字符串
category (common)	**所需的** Kubernetes Producer 和 Consumer 类别		字符串
dnsDomain (common)	用于 ServiceCall EIP 的 dns 域		字符串
kubernetesClient (common)	如果提供，则使用默认 KubernetesClient		KubernetesClient
portName (common)	用于 ServiceCall EIP 的端口名称		字符串
portProtocol (common)	端口协议，用于 ServiceCall EIP	tcp	字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
labelKey (consumer)	在监控某些资源时的 Consumer Label 键		字符串
labelValue (consumer)	在监控某些资源时的 Consumer Label 值		字符串
namespace (consumer)	命名空间		字符串
poolSize (consumer)	Consumer 池大小	1	int
resourceName (consumer)	要监视的 Consumer Resource Name		字符串

Name	描述	默认值	类型
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
operation (producer)	在 Kubernetes 上执行的操作		字符串
connectionTimeout (advanced)	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
caCertData (security)	CA 认证数据		字符串
caCertFile (security)	CA 认证文件		字符串
clientCertData (security)	客户端证书数据		字符串
clientCertFile (security)	客户端认证文件		字符串
clientKeyAlgo (security)	客户端使用的密钥算法		字符串
ClientKeyData (security)	客户端密钥数据		字符串
clientKeyFile (security)	客户端密钥文件		字符串
clientKeyPassphrase (security)	客户端密钥密码		字符串
oauthToken (security)	Auth Token		字符串
密码 (security)	连接到 Kubernetes 的密码		字符串

Name	描述	默认值	类型
trustCerts (security)	定义我们使用的证书是否被信任		布尔值
用户名 (security)	连接到 Kubernetes 的用户名		字符串

196.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 16 个选项，如下所列。

Name	描述	默认值	类型
camel.component.kubernetes.cluster.service.attributes	自定义服务属性。		Map
camel.component.kubernetes.cluster.service.cluster-labels	设置用于标识组成集群的 pod 的标签。		Map
camel.component.kubernetes.cluster.service.config-map-name	设置用于进行最佳锁定的 ConfigMap 的名称（默认为 'leaders'）。		字符串
camel.component.kubernetes.cluster.service.connection-timeout-millis	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
camel.component.kubernetes.cluster.service.enabled	如果 Kubernetes 集群服务应该启用或不启用，则默认为 false。	false	布尔值
camel.component.kubernetes.cluster.service.id	集群服务 ID		字符串
camel.component.kubernetes.cluster.service.jitter-factor	要应用 jitter 的因子，以防止所有 pod 在同一即时调用 Kubernetes API。		å�Œ

Name	描述	默认值	类型
camel.component.kubernetes.cluster.service.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测）		字符串
camel.component.kubernetes.cluster.service.lease-duration-millis	当前领导租期的默认持续时间。		Long
camel.component.kubernetes.cluster.service.master-url	设置 Kubernetes API 服务器 URL 的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串
camel.component.kubernetes.cluster.service.order	服务查找顺序/优先级.		整数
camel.component.kubernetes.cluster.service.pod-name	设置当前 pod 的名称（默认为从容器主机名自动探测）。		字符串
camel.component.kubernetes.cluster.service.renew-deadline-millis	领导机必须停止其服务的时间截止时间，因为它可能已经丢失领导力。		Long
camel.component.kubernetes.cluster.service.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它通过 jitter 因子进行随机化。		Long
camel.component.kubernetes.enabled	是否启用 kubernetes 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

196.4. HEADERS

Name	类型	描述
CamelKubernetesOperation	字符串	Producer 操作
CamelKubernetesNamespaceName	字符串	命名空间名称
CamelKubernetesNamespaceLabels	Map	命名空间标签
CamelKubernetesServiceLabels	Map	Service 标签
CamelKubernetesServiceName	字符串	服务名称
CamelKubernetesServiceSpec	io.fabric8.kubernetes.api.model.ServiceSpec	服务的 Spec
CamelKubernetesReplicationControllersLabels	Map	复制控制器标签

Name	类型	描述
CamelKubernetesReplicationControllerName	字符串	复制控制器名称
CamelKubernetesReplicationControllerSpec	io.fabric8.kubernetes.api.model.ReplicationControllerSpec	复制控制器的 Spec
CamelKubernetesReplicationControllerReplicas	整数	在 Scale 操作过程中的 Replication Controller 的副本数
CamelKubernetesPodsLabels	Map	Pod 标签
CamelKubernetesPodName	字符串	Pod 名称
CamelKubernetesPodSpec	io.fabric8.kubernetes.api.model.PodSpec	Pod 的 Spec
CamelKubernetesPersistentVolumesLabels	Map	持久性卷标签

Name	类型	描述
CamelKubernetesPersistentVolumesName	字符串	持久性卷名称
CamelKubernetesPersistentVolumesClaimsLabels	Map	持久性卷声明标签
CamelKubernetesPersistentVolumesClaimsName	字符串	持久性卷声明名称
CamelKubernetesPersistentVolumesClaimsSpec	io.fabric8.kubernetes.api.model.PersistentVolumeClaimSpec	持久性卷声明的 Spec
CamelKubernetesSecretsLabels	Map	Secret 标签
CamelKubernetesSecretsName	字符串	Secret 名称

Name	类型	描述
CamelKubernetesSecret	io.fabric8.kubernetes.api.model.Secret	Secret 对象
CamelKubernetesResourcesQuotaLabels	Map	资源配额标签
CamelKubernetesResourcesQuotaName	字符串	资源配额名称
CamelKubernetesResourceQuotaSpec	io.fabric8.kubernetes.api.model.ResourceQuotaSpec	资源配额的 Spec
CamelKubernetesServiceAccountsLabels	Map	服务帐户标签
CamelKubernetesServiceAccountName	字符串	服务帐户名称

Name	类型	描述
CamelKubernetesServiceAccount	io.fabric8.kubernetes.api.model.ServiceAccount	Service Account 对象
CamelKubernetesNodesLabels	Map	节点标签
CamelKubernetesNodeName	字符串	节点名
CamelKubernetesBuildsLabels	Map	OpenShift 构建标签
CamelKubernetesBuildName	字符串	OpenShift 构建名称
CamelKubernetesBuildConfigsLabels	Map	OpenShift 构建配置标签
CamelKubernetesBuildConfigName	字符串	OpenShift 构建配置名称

Name	类型	描述
CamelKubernetesEventAction	io.fabric8.kubernetes.client.Watcher.Action	消费者监视的操作
CamelKubernetesEventTimestamp	字符串	消费者监视的操作的时间戳
CamelKubernetesConfigMapName	字符串	ConfigMap 名称
CamelKubernetesConfigMapsLabels	Map	ConfigMap 标签
CamelKubernetesConfigData	Map	ConfigMap 数据

196.5. CATEGORIES

*实际上，**camel-kubernetes** 组件支持以下 **Kubernetes** 资源*

- ***命名空间***
- ***Pods***
- ***复制控制器***

- *服务*
- *持久性卷*
- *持久性卷声明*
- *Secrets*
- *资源配额*
- *服务帐户*
- *节点*
- *configmaps*

在 Openshift 中

- *Builds*
- *BuildConfig*

196.6. 使用方法

196.6.1. 生成者示例

在这里，我们演示了一些使用 camel-kubernetes 的制作者示例。

196.6.2. 创建 pod

```
from("direct:createPod")
    .toF("kubernetes://%s?oauthToken=%s&category=pods&operation=createPod", host,
authToken);
```

通过使用 KubernetesConstants.KUBERNETES_POD_SPEC 标头，您可以指定 PodSpec 并将其传递给此操作。

196.6.3. 删除 pod

```
from("direct:createPod")
    .toF("kubernetes://%s?oauthToken=%s&category=pods&operation=deletePod", host,
authToken);
```

通过使用 KubernetesConstants.KUBERNETES_POD_NAME 标头，您可以指定 Pod 名称并将其传递给此操作。

第 197 章 KUBERNETES CONFIGMAP 组件

从 Camel 版本 2.17 开始提供

Kubernetes ConfigMap 组件是 Kubernetes 组件 之一，它为执行 kubernetes ConfigMap 操作提供一个制作者。

197.1. 组件选项

Kubernetes ConfigMap 组件没有选项。

197.2. 端点选项

Kubernetes ConfigMap 端点使用 URI 语法进行配置：

```
kubernetes-config-maps:masterUrl
```

使用以下路径和查询参数：

197.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
masterUrl	**所需的** Kubernetes API 服务器 URL		字符串

197.2.2. 查询参数(20 参数)：

Name	描述	默认值	类型
apiVersion (producer)	要使用的 Kubernetes API 版本		字符串
dnsDomain (producer)	用于 ServiceCall EIP 的 dns 域		字符串
kubernetesClient (producer)	如果提供，则使用默认 KubernetesClient		KubernetesClient

Name	描述	默认值	类型
operation (producer)	在 Kubernetes 上执行的操作		字符串
PORTNAME (producer)	用于 ServiceCall EIP 的端口名称		字符串
portProtocol (producer)	端口协议，用于 ServiceCall EIP	tcp	字符串
connectionTimeout (advanced)	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
caCertData (security)	CA 认证数据		字符串
caCertFile (security)	CA 认证文件		字符串
clientCertData (security)	客户端证书数据		字符串
clientCertFile (security)	客户端认证文件		字符串
clientKeyAlgo (security)	客户端使用的密钥算法		字符串
ClientKeyData (security)	客户端密钥数据		字符串
clientKeyFile (security)	客户端密钥文件		字符串
clientKeyPassphrase (security)	客户端密钥密码		字符串
oauthToken (security)	Auth Token		字符串
密码 (security)	连接到 Kubernetes 的密码		字符串
trustCerts (security)	定义我们使用的证书是否被信任		布尔值

Name	描述	默认值	类型
用户名 (security)	连接到 Kubernetes 的用户名		字符串

197.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.kubernetes-config-maps.enabled	是否启用 kubernetes-config-maps 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-config-maps.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 198 章 KUBERNETES DEPLOYMENTS 组件

从 Camel 版本 2.20 开始提供

Kubernetes Deployments 组件是 Kubernetes 组件 之一，它为执行 kubernetes secret 操作提供一个制作者。

198.1. 组件选项

Kubernetes Deployments 组件没有选项。

198.2. 端点选项

Kubernetes Deployments 端点使用 URI 语法进行配置：

```
kubernetes-deployments:masterUrl
```

使用以下路径和查询参数：

198.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
masterUrl	**所需的** Kubernetes API 服务器 URL		字符串

198.2.2. 查询参数(28 参数)：

Name	描述	默认值	类型
apiVersion (common)	要使用的 Kubernetes API 版本		字符串
dnsDomain (common)	用于 ServiceCall EIP 的 dns 域		字符串
kubernetesClient (common)	如果提供，则使用默认 KubernetesClient		KubernetesClient

Name	描述	默认值	类型
portName (common)	用于 ServiceCall EIP 的端口名称		字符串
portProtocol (common)	端口协议，用于 ServiceCall EIP	tcp	字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
labelKey (consumer)	在监控某些资源时的 Consumer Label 键		字符串
labelValue (consumer)	在监控某些资源时的 Consumer Label 值		字符串
namespace (consumer)	命名空间		字符串
poolSize (consumer)	Consumer 池大小	1	int
resourceName (consumer)	要监视的 Consumer Resource Name		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
operation (producer)	在 Kubernetes 上执行的操作		字符串
connectionTimeout (advanced)	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
caCertData (security)	CA 认证数据		字符串

Name	描述	默认值	类型
caCertFile (security)	CA 认证文件		字符串
clientCertData (security)	客户端证书数据		字符串
clientCertFile (security)	客户端认证文件		字符串
clientKeyAlgo (security)	客户端使用的密钥算法		字符串
ClientKeyData (security)	客户端密钥数据		字符串
clientKeyFile (security)	客户端密钥文件		字符串
clientKeyPassphrase (security)	客户端密钥密码		字符串
oauthToken (security)	Auth Token		字符串
密码 (security)	连接到 Kubernetes 的密码		字符串
trustCerts (security)	定义我们使用的证书是否被信任		布尔值
用户名 (security)	连接到 Kubernetes 的用户名		字符串

198.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.kubernetes-deployments.enabled	是否启用 kubernetes-deployments 组件的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.component.kubernetes-deployments.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 199 章 KUBERNETES HPA 组件

从 Camel 版本 2.23 开始提供

Kubernetes HPA 组件是 Kubernetes 组件 之一，它提供了一个生成者来执行 kubernetes hpa 操作，以及消费者来使用 kubernetes hpa 事件。

199.1. 组件选项

Kubernetes HPA 组件没有选项。

199.2. 端点选项

Kubernetes HPA 端点使用 URI 语法进行配置：

```
kubernetes-hpa:masterUrl
```

使用以下路径和查询参数：

199.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
masterUrl	**所需的** Kubernetes API 服务器 URL		字符串

199.2.2. 查询参数(28 参数)：

Name	描述	默认值	类型
apiVersion (common)	要使用的 Kubernetes API 版本		字符串
dnsDomain (common)	用于 ServiceCall EIP 的 dns 域		字符串
kubernetesClient (common)	如果提供，则使用默认 KubernetesClient		KubernetesClient

Name	描述	默认值	类型
portName (common)	用于 ServiceCall EIP 的端口名称		字符串
portProtocol (common)	端口协议，用于 ServiceCall EIP	tcp	字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
labelKey (consumer)	在监控某些资源时的 Consumer Label 键		字符串
labelValue (consumer)	在监控某些资源时的 Consumer Label 值		字符串
namespace (consumer)	命名空间		字符串
poolSize (consumer)	Consumer 池大小	1	int
resourceName (consumer)	要监视的 Consumer Resource Name		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
operation (producer)	在 Kubernetes 上执行的操作		字符串
connectionTimeout (advanced)	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
caCertData (security)	CA 认证数据		字符串

Name	描述	默认值	类型
caCertFile (security)	CA 认证文件		字符串
clientCertData (security)	客户端证书数据		字符串
clientCertFile (security)	客户端认证文件		字符串
clientKeyAlgo (security)	客户端使用的密钥算法		字符串
ClientKeyData (security)	客户端密钥数据		字符串
clientKeyFile (security)	客户端密钥文件		字符串
clientKeyPassphrase (security)	客户端密钥密码		字符串
oauthToken (security)	Auth Token		字符串
密码 (security)	连接到 Kubernetes 的密码		字符串
trustCerts (security)	定义我们使用的证书是否被信任		布尔值
用户名 (security)	连接到 Kubernetes 的用户名		字符串

199.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.kubernetes-hpa.enabled	是否启用 kubernetes-hpa 组件的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.component.kubernetes-hpa.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 200 章 KUBERNETES 任务组件

从 Camel 版本 2.23 开始提供

Kubernetes 作业 组件是 Kubernetes 组件 之一，它为执行 kubernetes 任务操作提供一个制作者。

200.1. 组件选项

Kubernetes 作业组件没有选项。

200.2. 端点选项

Kubernetes Job 端点使用 URI 语法进行配置：

```
kubernetes-job:masterUrl
```

使用以下路径和查询参数：

200.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
masterUrl	**所需的** Kubernetes API 服务器 URL		字符串

200.2.2. 查询参数(28 参数)：

Name	描述	默认值	类型
apiVersion (common)	要使用的 Kubernetes API 版本		字符串
dnsDomain (common)	用于 ServiceCall EIP 的 dns 域		字符串
kubernetesClient (common)	如果提供，则使用默认 KubernetesClient		KubernetesClient

Name	描述	默认值	类型
portName (common)	用于 ServiceCall EIP 的端口名称		字符串
portProtocol (common)	端口协议，用于 ServiceCall EIP	tcp	字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
labelKey (consumer)	在监控某些资源时的 Consumer Label 键		字符串
labelValue (consumer)	在监控某些资源时的 Consumer Label 值		字符串
namespace (consumer)	命名空间		字符串
poolSize (consumer)	Consumer 池大小	1	int
resourceName (consumer)	要监视的 Consumer Resource Name		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
operation (producer)	在 Kubernetes 上执行的操作		字符串
connectionTimeout (advanced)	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
caCertData (security)	CA 认证数据		字符串

Name	描述	默认值	类型
caCertFile (security)	CA 认证文件		字符串
clientCertData (security)	客户端证书数据		字符串
clientCertFile (security)	客户端认证文件		字符串
clientKeyAlgo (security)	客户端使用的密钥算法		字符串
ClientKeyData (security)	客户端密钥数据		字符串
clientKeyFile (security)	客户端密钥文件		字符串
clientKeyPassphrase (security)	客户端密钥密码		字符串
oauthToken (security)	Auth Token		字符串
密码 (security)	连接到 Kubernetes 的密码		字符串
trustCerts (security)	定义我们使用的证书是否被信任		布尔值
用户名 (security)	连接到 Kubernetes 的用户名		字符串

200.3. 支持的 PRODUCER 操作

Kubernetes 作业组件支持以下制作者操作：

- ***listJob***
- ***listJobByLabels***
- ***getJob***

- createJob
- replaceJob
- deleteJob

200.4. KUBERNETES 任务生成示例

- listJob：此操作列出了 kubernetes 集群上的作业

```
from("direct:list").
   toF("kubernetes-job:///?kubernetesClient=#kubernetesClient&operation=listJob").
   to("mock:result");
```

此操作从集群中返回作业列表

- listJobByLabels：此操作根据 kubernetes 集群上的标签列出作业

```
from("direct:listByLabels").process(new Processor() {
          @Override
          public void process(Exchange exchange) throws Exception {
             Map<String, String> labels = new HashMap<>();
             labels.put("key1", "value1");
             labels.put("key2", "value2");

exchange.getIn().setHeader(KubernetesConstants.KUBERNETES_JOB_LABELS,
labels);
          }
       });
   toF("kubernetes-job:///?
kubernetesClient=#kubernetesClient&operation=listJobByLabels").
   to("mock:result");
```

此操作使用标签选择器（带有 key1 和 key2，值为 value1 和 value2）从集群中返回作业列表。

- CreateJob：此操作在 Kubernetes 集群上创建作业

得益于此 Java 测试中的 Emmerson Miranda，我们对这个操作有很好的例子

```
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import javax.inject.Inject;

import org.apache.camel.Endpoint;
import org.apache.camel.builder.RouteBuilder;
import org.apache.camel.cdi.Uri;
import org.apache.camel.component.kubernetes.KubernetesConstants;
import org.apache.camel.component.kubernetes.KubernetesOperations;

import io.fabric8.kubernetes.api.model.Container;
import io.fabric8.kubernetes.api.model.ObjectMeta;
import io.fabric8.kubernetes.api.model.PodSpec;
import io.fabric8.kubernetes.api.model.PodTemplateSpec;
import io.fabric8.kubernetes.api.model.batch.JobSpec;

public class KubernetesCreateJob extends RouteBuilder {

    @Inject
    @Uri("timer:foo?delay=1000&repeatCount=1")
    private Endpoint inputEndpoint;

    @Inject
    @Uri("log:output")
    private Endpoint resultEndpoint;

    @Override
    public void configure() {
        // you can configure the route rule with Java DSL here

        from(inputEndpoint)
         .routeId("kubernetes-jobcreate-client")
         .process(exchange -> {
          exchange.getIn().setHeader(KubernetesConstants.KUBERNETES_JOB_NAME,
"camel-job"); //DNS-1123 subdomain must consist of lower case alphanumeric
characters, '-' or '.', and must start and end with an alphanumeric character (e.g.
'example.com', regex used for validation is '[a-z0-9]([-a-z0-9]*[a-z0-9])?(\.[a-z0-9]([-a-z0-
9]*[a-z0-9])?)*')

exchange.getIn().setHeader(KubernetesConstants.KUBERNETES_NAMESPACE_NAM
E, "default");

                Map<String, String> joblabels = new HashMap<String, String>();
                joblabels.put("jobLabelKey1", "value1");
                joblabels.put("jobLabelKey2", "value2");
                joblabels.put("app", "jobFromCamelApp");

exchange.getIn().setHeader(KubernetesConstants.KUBERNETES_JOB_LABELS,
```

```
joblabels);


exchange.getIn().setHeader(KubernetesConstants.KUBERNETES_JOB_SPEC,
generateJobSpec());
        })
        .toF("kubernetes-job:///{{kubernetes-master-url}}?oauthToken={{kubernetes-
oauth-token:}}&operation=" + KubernetesOperations.CREATE_JOB_OPERATION)
        .log("Job created:")
        .process(exchange -> {
         System.out.println(exchange.getIn().getBody());
        })
          .to(resultEndpoint);
    }

 private JobSpec generateJobSpec() {
  JobSpec js = new JobSpec();

  PodTemplateSpec pts = new PodTemplateSpec();

  PodSpec ps = new PodSpec();
  ps.setRestartPolicy("Never");
  ps.setContainers(generateContainers());
  pts.setSpec(ps);

  ObjectMeta metadata = new ObjectMeta();
  Map<String, String> annotations = new HashMap<String, String>();
  annotations.put("jobMetadataAnnotation1", "random value");
  metadata.setAnnotations(annotations);

  Map<String, String> podlabels = new HashMap<String, String>();
  podlabels.put("podLabelKey1", "value1");
  podlabels.put("podLabelKey2", "value2");
  podlabels.put("app", "podFromCamelApp");
  metadata.setLabels(podlabels);

  pts.setMetadata(metadata);
  js.setTemplate(pts);
  return js;
 }

 private List<Container> generateContainers() {
  Container container = new Container();
  container.setName("pi");
  container.setImage("perl");
  List<String> command = new ArrayList<String>();
  command.add("echo");
  command.add("Job created from Apache Camel code at " + (new Date()));
  container.setCommand(command);
  List<Container> containers = new ArrayList<Container>();
  containers.add(container);
  return containers;
 }
}
```

200.5. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.kubernetes-job.enabled	是否启用 kubernetes-job 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-job.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 201 章 KUBERNETES 命名空间组件

从 Camel 版本 2.17 开始提供

Kubernetes Namespaces 组件是 Kubernetes 组件 之一，它提供了一个生成者来执行 kubernetes 命名空间操作，以及一个消费者来使用 kubernetes 命名空间事件。

201.1. 组件选项

Kubernetes 命名空间组件没有选项。

201.2. 端点选项

Kubernetes Namespaces 端点使用 URI 语法进行配置：

```
kubernetes-namespaces:masterUrl
```

使用以下路径和查询参数：

201.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
masterUrl	所需的 Kubernetes API 服务器 URL		字符串

201.2.2. 查询参数(28 参数)：

Name	描述	默认值	类型
apiVersion (common)	要使用的 Kubernetes API 版本		字符串
dnsDomain (common)	用于 ServiceCall EIP 的 dns 域		字符串
kubernetesClient (common)	如果提供，则使用默认 KubernetesClient		KubernetesClient

Name	描述	默认值	类型
portName (common)	用于 ServiceCall EIP 的端口名称		字符串
portProtocol (common)	端口协议，用于 ServiceCall EIP	tcp	字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
labelKey (consumer)	在监控某些资源时的 Consumer Label 键		字符串
labelValue (consumer)	在监控某些资源时的 Consumer Label 值		字符串
namespace (consumer)	命名空间		字符串
poolSize (consumer)	Consumer 池大小	1	int
resourceName (consumer)	要监视的 Consumer Resource Name		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
operation (producer)	在 Kubernetes 上执行的操作		字符串
connectionTimeout (advanced)	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
caCertData (security)	CA 认证数据		字符串

Name	描述	默认值	类型
caCertFile (security)	CA 认证文件		字符串
clientCertData (security)	客户端证书数据		字符串
clientCertFile (security)	客户端认证文件		字符串
clientKeyAlgo (security)	客户端使用的密钥算法		字符串
ClientKeyData (security)	客户端密钥数据		字符串
clientKeyFile (security)	客户端密钥文件		字符串
clientKeyPassphrase (security)	客户端密钥密码		字符串
oauthToken (security)	Auth Token		字符串
密码 (security)	连接到 Kubernetes 的密码		字符串
trustCerts (security)	定义我们使用的证书是否被信任		布尔值
用户名 (security)	连接到 Kubernetes 的用户名		字符串

201.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.kubernetes-namespaces.enabled	是否启用 kubernetes-namespaces 组件的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.component.kubernetes-namespaces.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 202 章 KUBERNETES 节点组件

从 Camel 版本 2.17 开始提供

Kubernetes 节点 组件是 Kubernetes 组件 之一， 它提供了一个生成者来执行 kubernetes 节点操作，以及一个消费者来使用 kubernetes 节点事件。

202.1. 组件选项

Kubernetes 节点组件没有选项。

202.2. 端点选项

Kubernetes 节点端点使用 URI 语法进行配置：

```
kubernetes-nodes:masterUrl
```

使用以下路径和查询参数：

202.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
masterUrl	所需的 Kubernetes API 服务器 URL		字符串

202.2.2. 查询参数(28 参数)：

Name	描述	默认值	类型
apiVersion (common)	要使用的 Kubernetes API 版本		字符串
dnsDomain (common)	用于 ServiceCall EIP 的 dns 域		字符串
kubernetesClient (common)	如果提供，则使用默认 KubernetesClient		KubernetesClient

Name	描述	默认值	类型
portName (common)	用于 ServiceCall EIP 的端口名称		字符串
portProtocol (common)	端口协议，用于 ServiceCall EIP	tcp	字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
labelKey (consumer)	在监控某些资源时的 Consumer Label 键		字符串
labelValue (consumer)	在监控某些资源时的 Consumer Label 值		字符串
namespace (consumer)	命名空间		字符串
poolSize (consumer)	Consumer 池大小	1	int
resourceName (consumer)	要监视的 Consumer Resource Name		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
operation (producer)	在 Kubernetes 上执行的操作		字符串
connectionTimeout (advanced)	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
caCertData (security)	CA 认证数据		字符串

Name	描述	默认值	类型
caCertFile (security)	CA 认证文件		字符串
clientCertData (security)	客户端证书数据		字符串
clientCertFile (security)	客户端认证文件		字符串
clientKeyAlgo (security)	客户端使用的密钥算法		字符串
ClientKeyData (security)	客户端密钥数据		字符串
clientKeyFile (security)	客户端密钥文件		字符串
clientKeyPassphrase (security)	客户端密钥密码		字符串
oauthToken (security)	Auth Token		字符串
密码 (security)	连接到 Kubernetes 的密码		字符串
trustCerts (security)	定义我们使用的证书是否被信任		布尔值
用户名 (security)	连接到 Kubernetes 的用户名		字符串

202.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.kubernetes-nodes.enabled	是否启用 kubernetes-nodes 组件的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.component.kubernetes-nodes.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 203 章 KUBERNETES 持久性卷声明组件

从 Camel 版本 2.17 开始提供

Kubernetes 持久性卷声明组件是 Kubernetes 组件 之一，它提供了一个生成者来执行 kubernetes 持久性卷声明操作。

203.1. 组件选项

Kubernetes 持久性卷声明组件没有选项。

203.2. 端点选项

Kubernetes 持久性卷声明端点使用 URI 语法进行配置：

```
kubernetes-persistent-volumes-claims:masterUrl
```

使用以下路径和查询参数：

203.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
masterUrl	**所需的** Kubernetes API 服务器 URL		字符串

203.2.2. 查询参数(20 参数)：

Name	描述	默认值	类型
apiVersion (producer)	要使用的 Kubernetes API 版本		字符串
dnsDomain (producer)	用于 ServiceCall EIP 的 dns 域		字符串
kubernetesClient (producer)	如果提供，则使用默认 KubernetesClient		KubernetesClient

Name	描述	默认值	类型
operation (producer)	在 Kubernetes 上执行的操作		字符串
PORTNAME (producer)	用于 ServiceCall EIP 的端口名称		字符串
portProtocol (producer)	端口协议，用于 ServiceCall EIP	tcp	字符串
connectionTimeout (advanced)	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
caCertData (security)	CA 认证数据		字符串
caCertFile (security)	CA 认证文件		字符串
clientCertData (security)	客户端证书数据		字符串
clientCertFile (security)	客户端认证文件		字符串
clientKeyAlgo (security)	客户端使用的密钥算法		字符串
ClientKeyData (security)	客户端密钥数据		字符串
clientKeyFile (security)	客户端密钥文件		字符串
clientKeyPassphrase (security)	客户端密钥密码		字符串
oauthToken (security)	Auth Token		字符串
密码 (security)	连接到 Kubernetes 的密码		字符串
trustCerts (security)	定义我们使用的证书是否被信任		布尔值

Name	描述	默认值	类型
用户名 (security)	连接到 Kubernetes 的用户名		字符串

203.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.kubernetes-persistent-volumes-claims.enabled	是否启用 kubernetes-persistent-volumes-claims 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-persistent-volumes-claims.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 204 章 KUBERNETES 持久性卷组件

从 Camel 版本 2.17 开始提供

Kubernetes 持久性卷组件是 Kubernetes 组件 之一，它提供了一个生成者来执行 kubernetes 持久性卷操作。

204.1. 组件选项

Kubernetes 持久性卷组件没有选项。

204.2. 端点选项

Kubernetes 持久性卷端点使用 URI 语法进行配置：

```
kubernetes-persistent-volumes:masterUrl
```

使用以下路径和查询参数：

204.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
masterUrl	**所需的** Kubernetes API 服务器 URL		字符串

204.2.2. 查询参数(20 参数)：

Name	描述	默认值	类型
apiVersion (producer)	要使用的 Kubernetes API 版本		字符串
dnsDomain (producer)	用于 ServiceCall EIP 的 dns 域		字符串
kubernetesClient (producer)	如果提供，则使用默认 KubernetesClient		KubernetesClient

Name	描述	默认值	类型
operation (producer)	在 Kubernetes 上执行的操作		字符串
PORTNAME (producer)	用于 ServiceCall EIP 的端口名称		字符串
portProtocol (producer)	端口协议，用于 ServiceCall EIP	tcp	字符串
connectionTimeout (advanced)	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
caCertData (security)	CA 认证数据		字符串
caCertFile (security)	CA 认证文件		字符串
clientCertData (security)	客户端证书数据		字符串
clientCertFile (security)	客户端认证文件		字符串
clientKeyAlgo (security)	客户端使用的密钥算法		字符串
ClientKeyData (security)	客户端密钥数据		字符串
clientKeyFile (security)	客户端密钥文件		字符串
clientKeyPassphrase (security)	客户端密钥密码		字符串
oauthToken (security)	Auth Token		字符串
密码 (security)	连接到 Kubernetes 的密码		字符串
trustCerts (security)	定义我们使用的证书是否被信任		布尔值

Name	描述	默认值	类型
用户名 (security)	连接到 Kubernetes 的用户名		字符串

204.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.kubernetes-persistent-volumes.enabled	是否启用 kubernetes-persistent-volumes 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-persistent-volumes.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 205 章 KUBERNETES POD 组件

从 Camel 版本 2.17 开始提供

Kubernetes Pod 组件是 Kubernetes 组件 之一，它为执行 kubernetes pod 操作提供一个制作者。

205.1. 组件选项

Kubernetes Pod 组件没有选项。

205.2. 端点选项

Kubernetes Pod 端点使用 URI 语法进行配置：

```
kubernetes-pods:masterUrl
```

使用以下路径和查询参数：

205.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
masterUrl	所需的 Kubernetes API 服务器 URL		字符串

205.2.2. 查询参数(28 参数)：

Name	描述	默认值	类型
apiVersion (common)	要使用的 Kubernetes API 版本		字符串
dnsDomain (common)	用于 ServiceCall EIP 的 dns 域		字符串
kubernetesClient (common)	如果提供，则使用默认 KubernetesClient		KubernetesClient

Name	描述	默认值	类型
portName (common)	用于 ServiceCall EIP 的端口名称		字符串
portProtocol (common)	端口协议，用于 ServiceCall EIP	tcp	字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
labelKey (consumer)	在监控某些资源时的 Consumer Label 键		字符串
labelValue (consumer)	在监控某些资源时的 Consumer Label 值		字符串
namespace (consumer)	命名空间		字符串
poolSize (consumer)	Consumer 池大小	1	int
resourceName (consumer)	要监视的 Consumer Resource Name		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
operation (producer)	在 Kubernetes 上执行的操作		字符串
connectionTimeout (advanced)	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
caCertData (security)	CA 认证数据		字符串

Name	描述	默认值	类型
caCertFile (security)	CA 认证文件		字符串
clientCertData (security)	客户端证书数据		字符串
clientCertFile (security)	客户端认证文件		字符串
clientKeyAlgo (security)	客户端使用的密钥算法		字符串
ClientKeyData (security)	客户端密钥数据		字符串
clientKeyFile (security)	客户端密钥文件		字符串
clientKeyPassphrase (security)	客户端密钥密码		字符串
oauthToken (security)	Auth Token		字符串
密码 (security)	连接到 Kubernetes 的密码		字符串
trustCerts (security)	定义我们使用的证书是否被信任		布尔值
用户名 (security)	连接到 Kubernetes 的用户名		字符串

205.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.kubernetes-pods.enabled	是否启用 kubernetes-pods 组件的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.component.kubernetes-pods.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 206 章 KUBERNETES 复制控制器组件

从 Camel 版本 2.17 开始提供

Kubernetes Replication Controller 组件是 Kubernetes 组件 之一，它为执行 kubernetes 复制控制器操作和消费者提供一个使用者来使用 kubernetes 复制控制器事件。

206.1. 组件选项

Kubernetes Replication Controller 组件没有选项。

206.2. 端点选项

Kubernetes Replication Controller 端点使用 URI 语法进行配置：

```
kubernetes-replication-controllers:masterUrl
```

使用以下路径和查询参数：

206.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
masterUrl	所需的 Kubernetes API 服务器 URL		字符串

206.2.2. 查询参数(28 参数)：

Name	描述	默认值	类型
apiVersion (common)	要使用的 Kubernetes API 版本		字符串
dnsDomain (common)	用于 ServiceCall EIP 的 dns 域		字符串
kubernetesClient (common)	如果提供，则使用默认 KubernetesClient		KubernetesClient

Name	描述	默认值	类型
portName (common)	用于 ServiceCall EIP 的端口名称		字符串
portProtocol (common)	端口协议，用于 ServiceCall EIP	tcp	字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
labelKey (consumer)	在监控某些资源时的 Consumer Label 键		字符串
labelValue (consumer)	在监控某些资源时的 Consumer Label 值		字符串
namespace (consumer)	命名空间		字符串
poolSize (consumer)	Consumer 池大小	1	int
resourceName (consumer)	要监视的 Consumer Resource Name		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
operation (producer)	在 Kubernetes 上执行的操作		字符串
connectionTimeout (advanced)	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
caCertData (security)	CA 认证数据		字符串

Name	描述	默认值	类型
caCertFile (security)	CA 认证文件		字符串
clientCertData (security)	客户端证书数据		字符串
clientCertFile (security)	客户端认证文件		字符串
clientKeyAlgo (security)	客户端使用的密钥算法		字符串
ClientKeyData (security)	客户端密钥数据		字符串
clientKeyFile (security)	客户端密钥文件		字符串
clientKeyPassphrase (security)	客户端密钥密码		字符串
oauthToken (security)	Auth Token		字符串
密码 (security)	连接到 Kubernetes 的密码		字符串
trustCerts (security)	定义我们使用的证书是否被信任		布尔值
用户名 (security)	连接到 Kubernetes 的用户名		字符串

206.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.kubernetes-replication-controllers.enabled	是否启用 kubernetes-replication-controllers 组件的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.component.kubernetes-replication-controllers.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 207 章 KUBERNETES 资源配额组件

从 Camel 版本 2.17 开始提供

Kubernetes Resources Quota 组件是 Kubernetes 组件 之一，它为执行 kubernetes 资源配额操作提供一个制作者。

207.1. 组件选项

Kubernetes Resources Quota 组件没有选项。

207.2. 端点选项

Kubernetes Resources Quota 端点使用 URI 语法进行配置：

```
kubernetes-resources-quota:masterUrl
```

使用以下路径和查询参数：

207.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
masterUrl	**所需的** Kubernetes API 服务器 URL		字符串

207.2.2. 查询参数(20 参数)：

Name	描述	默认值	类型
apiVersion (producer)	要使用的 Kubernetes API 版本		字符串
dnsDomain (producer)	用于 ServiceCall EIP 的 dns 域		字符串
kubernetesClient (producer)	如果提供，则使用默认 KubernetesClient		KubernetesClient

Name	描述	默认值	类型
operation (producer)	在 Kubernetes 上执行的操作		字符串
PORTNAME (producer)	用于 ServiceCall EIP 的端口名称		字符串
portProtocol (producer)	端口协议，用于 ServiceCall EIP	tcp	字符串
connectionTimeout (advanced)	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
caCertData (security)	CA 认证数据		字符串
caCertFile (security)	CA 认证文件		字符串
clientCertData (security)	客户端证书数据		字符串
clientCertFile (security)	客户端认证文件		字符串
clientKeyAlgo (security)	客户端使用的密钥算法		字符串
ClientKeyData (security)	客户端密钥数据		字符串
clientKeyFile (security)	客户端密钥文件		字符串
clientKeyPassphrase (security)	客户端密钥密码		字符串
oauthToken (security)	Auth Token		字符串
密码 (security)	连接到 Kubernetes 的密码		字符串
trustCerts (security)	定义我们使用的证书是否被信任		布尔值

Name	描述	默认值	类型
用户名 (security)	连接到 Kubernetes 的用户名		字符串

207.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.kubernetes-resources-quota.enabled	是否启用 kubernetes-resources-quota 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-resources-quota.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 208 章 KUBERNETES SECRET 组件

从 Camel 版本 2.17 开始提供

Kubernetes Secret 组件是 Kubernetes 组件 之一，它为执行 kubernetes secret 操作提供一个制作者。

208.1. 组件选项

Kubernetes Secrets 组件没有选项。

208.2. 端点选项

Kubernetes Secrets 端点使用 URI 语法进行配置：

```
kubernetes-secrets:masterUrl
```

使用以下路径和查询参数：

208.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
masterUrl	**所需的** Kubernetes API 服务器 URL		字符串

208.2.2. 查询参数(20 参数)：

Name	描述	默认值	类型
apiVersion (producer)	要使用的 Kubernetes API 版本		字符串
dnsDomain (producer)	用于 ServiceCall EIP 的 dns 域		字符串
kubernetesClient (producer)	如果提供，则使用默认 KubernetesClient		KubernetesClient

Name	描述	默认值	类型
operation (producer)	在 Kubernetes 上执行的操作		字符串
PORTNAME (producer)	用于 ServiceCall EIP 的端口名称		字符串
portProtocol (producer)	端口协议，用于 ServiceCall EIP	tcp	字符串
connectionTimeout (advanced)	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
caCertData (security)	CA 认证数据		字符串
caCertFile (security)	CA 认证文件		字符串
clientCertData (security)	客户端证书数据		字符串
clientCertFile (security)	客户端认证文件		字符串
clientKeyAlgo (security)	客户端使用的密钥算法		字符串
ClientKeyData (security)	客户端密钥数据		字符串
clientKeyFile (security)	客户端密钥文件		字符串
clientKeyPassphrase (security)	客户端密钥密码		字符串
oauthToken (security)	Auth Token		字符串
密码 (security)	连接到 Kubernetes 的密码		字符串
trustCerts (security)	定义我们使用的证书是否被信任		布尔值

Name	描述	默认值	类型
用户名 (security)	连接到 Kubernetes 的用户名		字符串

208.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.kubernetes-secrets.enabled	是否启用 kubernetes-secrets 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-secrets.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 209 章 KUBERNETES 服务帐户组件

从 Camel 版本 2.17 开始提供

Kubernetes Service Account 组件是 Kubernetes 组件 之一，它为执行 kubernetes Service Account 操作提供一个制作者。

209.1. 组件选项

Kubernetes Service Account 组件没有选项。

209.2. 端点选项

Kubernetes Service Account 端点使用 URI 语法进行配置：

```
kubernetes-service-accounts:masterUrl
```

使用以下路径和查询参数：

209.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
masterUrl	所需的 Kubernetes API 服务器 URL		字符串

209.2.2. 查询参数(20 参数)：

Name	描述	默认值	类型
apiVersion (producer)	要使用的 Kubernetes API 版本		字符串
dnsDomain (producer)	用于 ServiceCall EIP 的 dns 域		字符串
kubernetesClient (producer)	如果提供，则使用默认 KubernetesClient		KubernetesClient

Name	描述	默认值	类型
operation (producer)	在 Kubernetes 上执行的操作		字符串
PORTNAME (producer)	用于 ServiceCall EIP 的端口名称		字符串
portProtocol (producer)	端口协议，用于 ServiceCall EIP	tcp	字符串
connectionTimeout (advanced)	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
caCertData (security)	CA 认证数据		字符串
caCertFile (security)	CA 认证文件		字符串
clientCertData (security)	客户端证书数据		字符串
clientCertFile (security)	客户端认证文件		字符串
clientKeyAlgo (security)	客户端使用的密钥算法		字符串
ClientKeyData (security)	客户端密钥数据		字符串
clientKeyFile (security)	客户端密钥文件		字符串
clientKeyPassphrase (security)	客户端密钥密码		字符串
oauthToken (security)	Auth Token		字符串
密码 (security)	连接到 Kubernetes 的密码		字符串
trustCerts (security)	定义我们使用的证书是否被信任		布尔值

Name	描述	默认值	类型
用户名 (security)	连接到 Kubernetes 的用户名		字符串

209.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.kubernetes-service-accounts.enabled	是否启用 kubernetes-service-accounts 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-service-accounts.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 210 章 KUBERNETES 服务组件

从 Camel 版本 2.17 开始提供

Kubernetes Services 组件是 Kubernetes 组件 之一，它提供了一个生成者来执行 kubernetes 服务操作，以及一个消费者来使用 kubernetes 服务事件。

210.1. 组件选项

Kubernetes Services 组件没有选项。

210.2. 端点选项

Kubernetes Services 端点使用 URI 语法进行配置：

```
kubernetes-services:masterUrl
```

使用以下路径和查询参数：

210.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
masterUrl	**所需的** Kubernetes API 服务器 URL		字符串

210.2.2. 查询参数(28 参数)：

Name	描述	默认值	类型
apiVersion (common)	要使用的 Kubernetes API 版本		字符串
dnsDomain (common)	用于 ServiceCall EIP 的 dns 域		字符串
kubernetesClient (common)	如果提供，则使用默认 KubernetesClient		KubernetesClient

Name	描述	默认值	类型
portName (common)	用于 ServiceCall EIP 的端口名称		字符串
portProtocol (common)	端口协议，用于 ServiceCall EIP	tcp	字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
labelKey (consumer)	在监控某些资源时的 Consumer Label 键		字符串
labelValue (consumer)	在监控某些资源时的 Consumer Label 值		字符串
namespace (consumer)	命名空间		字符串
poolSize (consumer)	Consumer 池大小	1	int
resourceName (consumer)	要监视的 Consumer Resource Name		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
operation (producer)	在 Kubernetes 上执行的操作		字符串
connectionTimeout (advanced)	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
caCertData (security)	CA 认证数据		字符串

Name	描述	默认值	类型
caCertFile (security)	CA 认证文件		字符串
clientCertData (security)	客户端证书数据		字符串
clientCertFile (security)	客户端认证文件		字符串
clientKeyAlgo (security)	客户端使用的密钥算法		字符串
ClientKeyData (security)	客户端密钥数据		字符串
clientKeyFile (security)	客户端密钥文件		字符串
clientKeyPassphrase (security)	客户端密钥密码		字符串
oauthToken (security)	Auth Token		字符串
密码 (security)	连接到 Kubernetes 的密码		字符串
trustCerts (security)	定义我们使用的证书是否被信任		布尔值
用户名 (security)	连接到 Kubernetes 的用户名		字符串

210.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.kubernetes-services.enabled	是否启用 kubernetes-services 组件的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.component.kubernetes-services.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

210.4. ECLIPSE KURA 组件

从 Camel 2.15 开始提供

本文档页面涵盖了使用 Eclipse Kura M2M 网关 Camel 的集成选项。将 Camel 路由部署到 Eclipse Kura 的常见原因是向消息传递 M2M 网关提供企业集成模式和 Camel 组件。例如，您可能想要在 Raspberry PI 上安装 Kura，然后使用 Kura 服务读取与 Raspberry PI 附加到 Raspberry PI 的温度值，最后使用 Camel EIP 和组件将当前的温度值转发到您的数据中心服务。

210.4.1. KuraRouter activator

部署到 Eclipse Kura 的捆绑包通常作为捆绑包激活器进行开发。因此，将 Apache Camel 路由部署到 Kura 的最简单方法是创建一个包含扩展 org.apache.camel.kura.KuraRouter 类的 OSGi 捆绑包：

```
public class MyKuraRouter extends KuraRouter {

  @Override
  public void configure() throws Exception {
    from("timer:trigger").
      to("netty-http:http://app.mydatacenter.com/api");
  }

}
```

请记住，K uraRouter 实现 org.osgi.framework.BundleActivator 接口，因此您需要 在创建 Kura 捆绑包组件类时 注册其 启动和停止 生命周期方法。

Kura 路由器启动其自己的 OSGi 感知型 CamelContext。这意味着，对于每个类扩展 KuraRouter，将有一个专用的 CamelContext 实例。理想情况下，我们建议为每个 OSGi 捆绑部署一个 KuraRouter。

210.4.2. 部署 KuraRouter

包含 Kura 路由器类的捆绑包应在 OSGi 清单中导入以下软件包：

```
Import-Package: org.osgi.framework;version="1.3.0",
 org.slf4j;version="1.6.4",

org.apache.camel,org.apache.camel.impl,org.apache.camel.core.osgi,org.apache.camel.builde
r,org.apache.camel.model,
 org.apache.camel.component.kura
```

请记住，您不必导入要在路由中使用的每个 Camel 组件捆绑包，因为 Camel 组件会作为运行时级别的服务解析。

在部署路由器捆绑包前，请确定您部署了（并启动）以下 Camel 核心捆绑包（使用 Kura GoGo shell）...

```
install file:///home/user/.m2/repository/org/apache/camel/camel-core/2.15.0/camel-core-
2.15.0.jar
start <camel-core-bundle-id>
install file:///home/user/.m2/repository/org/apache/camel/camel-core-osgi/2.15.0/camel-core-
osgi-2.15.0.jar
start <camel-core-osgi-bundle-id>
install file:///home/user/.m2/repository/org/apache/camel/camel-kura/2.15.0/camel-kura-
2.15.0.jar
start <camel-kura-bundle-id>
```

... 以及您要在路由中使用的所有组件：

```
install file:///home/user/.m2/repository/org/apache/camel/camel-stream/2.15.0/camel-stream-
2.15.0.jar
start <camel-stream-bundle-id>
```

最后部署路由器捆绑包：

```
install file:///home/user/.m2/repository/com/example/myrouter/1.0/myrouter-1.0.jar
start <your-bundle-id>
```

210.4.3. KuraRouter 工具

```
 Kura router base class provides many useful utilities. This section
explores each of them.
```

210.4.3.1. SLF4J logger

Kura 使用 SLF4J facade 进行日志记录目的。受保护的成员 日志 返回与给定 Kura 路由器关联的 SLF4J 日志记录器实例。

```
public class MyKuraRouter extends KuraRouter {

    @Override
    public void configure() throws Exception {
        log.info("Configuring Camel routes!");
        ...
    }

}
```

210.4.3.2. BundleContext

受保护的成员 bundleContext 返回与给定 Kura 路由器关联的捆绑包上下文。

```
public class MyKuraRouter extends KuraRouter {

    @Override
    public void configure() throws Exception {
        ServiceReference<MyService> serviceRef =
bundleContext.getServiceReference(LogService.class.getName());
        MyService myService = bundleContext.getService(serviceRef);
        ...
    }

}
```

210.4.3.3. CamelContext

保护的成员 camelContext 是与给定 Kura 路由器关联的 CamelContext。

```
public class MyKuraRouter extends KuraRouter {

    @Override
    public void configure() throws Exception {
        camelContext.getStatus();
        ...
    }

}
```

210.4.3.4. ProducerTemplate

受保护的 member producerTemplate 是与给定 Camel 上下文关联的 ProducerTemplate 实例。

```
public class MyKuraRouter extends KuraRouter {

    @Override
    public void configure() throws Exception {
        producerTemplate.sendBody("jms:temperature", 22.0);
        ...
    }

}
```

210.4.3.5. ConsumerTemplate

受保护的成员 consumerTemplate 是与给定 Camel 上下文关联的 ConsumerTemplate 实例。

```
public class MyKuraRouter extends KuraRouter {

    @Override
    public void configure() throws Exception {
        double currentTemperature = producerTemplate.receiveBody("jms:temperature",
Double.class);
        ...
    }

}
```

210.4.3.6. OSGi 服务解析器

OSGi 服务解析器(Class<T> serviceType)用于通过类型从 OSGi 捆绑包上下文轻松检索服务。

```
public class MyKuraRouter extends KuraRouter {

    @Override
    public void configure() throws Exception {
        MyService myService = service(MyService.class);
        ...
    }

}
```

如果没有找到 service，则返回 null 值。如果您希望应用程序在服务不可用时失败，请使用 requiredService (Class) 方法。如果找不到服务，则 requiredService 会抛出 IllegalStateException。

```
public class MyKuraRouter extends KuraRouter {
```

```
    @Override
    public void configure() throws Exception {
        MyService myService = requiredService(MyService.class);
        ...
    }

}
```

210.4.4. KuraRouter activator 回调

Kura 路由器附带生命周期回调，可用于自定义 Camel 路由器的工作方式。例如，要在前配置与路由器关联的 CamelContext 实例，请覆盖 KuraRouter 类的 beforeStart 方法：

```
public class MyKuraRouter extends KuraRouter {

  ...

  protected void beforeStart(CamelContext camelContext) {
    OsgiDefaultCamelContext osgiContext = (OsgiCamelContext) camelContext;
    osgiContext.setName("NameOfTheRouter");
  }

}
```

210.4.5. 从 ConfigurationAdmin 加载 XML 路由

有时需要从服务器配置中读取路由的 XML 定义。对于参与式重新部署成本的 IoT 网关来说，这种常见方案可能是很大的。为了满足此要求，每个 KuraRouter 使用 OSGi ConfigurationAdmin 从 kura.camel.BUNDLE-SYMBOLIC-SYMBOLIC-NAME.route 属性中查找 kura.camel PID。这个方法允许您为每个部署的 KuraRouter 定义 Camel XML 路由文件。要更新路由，只需编辑适当的配置属性并重启与其关联的捆绑包。kura.camel.BUNDLE-SYMBOLIC-NAME.route 属性的内容应该是 Camel XML 路由文件，例如：

```
<routes xmlns="http://camel.apache.org/schema/spring">
    <route id="loaded">
        <from uri="direct:bar"/>
        <to uri="mock:bar"/>
    </route>
</routes>
```

210.4.6. 将 Kura 路由器部署为声明 OSGi 服务

如果要将 Kura 路由器部署为声明 OSGi 服务，您可以使用 KuraRouter 提供的 激活和停用 方法。

```
<scr:component name="org.eclipse.kura.example.camel.MyKuraRouter" activate="activate"
deactivate="deactivate" enabled="true" immediate="true">
  <implementation class="org.eclipse.kura.example.camel.MyKuraRouter"/>
```

```
</scr:component>
```

210.4.7. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 211 章 语言组件

从 Camel 版本 2.5 开始提供

语言组件允许您将 Exchange 发送到端点，该端点通过 Camel 中任何支持的语言执行脚本。通过有一个组件来执行语言脚本，它允许更多动态路由功能。例如，通过使用 Routing Slip 或 Dynamic Router EIPs，您可以将消息发送到脚本动态定义 的语言 端点。

此组件在 camel-core 中提供，因此不需要额外的 JAR。只有选择的语言需要包括额外的 Camel 组件，比如使用 Groovy 或 JavaScript 语言。

211.1. URI 格式

```
language://languageName[:script][?options]
```

从 Camel 2.11 开始，您可以使用与 Camel 中其他语言支持的相同表示法引用脚本的外部资源 ???

```
language://languageName:resource:scheme:location][?options]
```

211.2. URI 选项

Language 组件没有选项。

Language 端点使用 URI 语法进行配置：

```
language:languageName:resourceUri
```

使用以下路径和查询参数：

211.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
languageName	**必需** 设置要使用的语言名称		字符串

Name	描述	默认值	类型
resourceUri	资源的路径，或查找 Registry 中要用作资源的 bean 的引用		字符串

211.2.2. 查询参数(6 参数)：

Name	描述	默认值	类型
binary (producer)	脚本是二进制内容还是文本内容。默认情况下，该脚本读取为文本内容（如 java.lang.String）	false	布尔值
cacheScript (producer)	是否缓存编译的脚本并重复使用 Notice，重新使用脚本可能会导致脚本处理一个 Camel org.apache.camel.Exchange 到下一个 org.apache.camel.Exchange。	false	布尔值
contentCache (producer)	设置是否使用资源内容缓存。	false	布尔值
script (producer)	设置要执行的脚本		字符串
transform (producer)	脚本的结果是否应用作消息正文。这个选项是默认的 true。	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

211.3. 消息标头

以下消息标头可用于影响组件的行为

标头	描述
CamelLanguageScript	要在标头中提供的脚本。优先于端点上配置的脚本。

211.4. 例子

例如，您可以使用 Simple 语言来消息转换信息：

如果要转换消息正文类型，您也可以执行此操作：

您还可以使用 Groovy 语言，例如，输入消息将乘以 2 ：

您还可以将脚本作为标头提供，如下所示。在这里，我们使用 XPath 语言从 < foo> 标签中提取文本。

```
Object out = producer.requestBodyAndHeader("language:xpath", "<foo>Hello World</foo>",
Exchange.LANGUAGE_SCRIPT, "/foo/text()");
assertEquals("Hello World", out);
```

211.5. 从资源载入脚本

从 Camel 2.9 开始可用

您可以为要在端点 uri 或 Exchange.LANGUAGE_SCRIPT 标头中载入的脚本指定资源 uri。
uri 必须以以下方案之一开始：file:, classpath:, 或 http ：

例如，从 classpath 中载入脚本：

默认情况下，会加载一次并缓存脚本。但是，您可以禁用 contentCache 选项，并在每次评估时载入脚本。
例如，如果在磁盘上更改了 myscript.txt 文件，则使用更新的脚本：

从 Camel 2.11 开始，您可以使用 "resource:" 前缀来引用 Camel 中其他语言资源，如下所示： ???

第 212 章 LDAP 组件

从 Camel 版本 1.5 开始提供

ldap 组件允许您使用过滤器作为消息有效负载在 LDAP 服务器中执行搜索。
此组件使用标准 JNDI (javax.naming package)来访问服务器。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-ldap</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

212.1. URI 格式

```
ldap:ldapServerBean[?options]
```

URI 的 ldapServerBean 部分指的是 registry 中的 DirContext bean。LDAP 组件仅支持生成者端点，这意味着 ldap URI 无法在路由开始时出现在 from 中。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

212.2. 选项

LDAP 组件没有选项。

LDAP 端点使用 URI 语法进行配置：

```
ldap:dirContextName
```

使用以下路径和查询参数：

212.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
dirContextName	**必需的** Name of a javax.naming.directory.DirContext, 或 java.util.Hashtable, 或 Map bean to lookup。如果 bean 是 Hashtable 或 Map，则为每个用途创建新的 javax.naming.directory.DirContext 实例。如果 bean 是 javax.naming.directory.DirContext，则 bean 将用作给定的。在不能共享 javax.naming.directory.DirContext 时，后者可能无法被共享，在这种情况下，最好使用 java.util.Hashtable 或 Map。		字符串

212.2.2. 查询参数(5 参数)：

Name	描述	默认值	类型
base (producer)	用于搜索的基本 DN。	ou=system	字符串
pageSize (producer)	当指定 ldap 模块使用分页来检索所有结果时（大多数 LDAP 服务器在尝试检索一个查询中超过 1000 个条目时会抛出异常）。要能够使用此 LdapContext （子类 DirContext）必须作为 ldapServerBean 传递（否则抛出异常）		整数
returnedAttributes (producer)	在结果的每个条目中应设置的属性的逗号分隔列表		字符串
scope (producer)	指定从基本 DN 开始搜索条目树的方式。	subtree	字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

212.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.ldap.enabled	启用 ldap 组件	true	布尔值

Name	描述	默认值	类型
camel.component.ldap.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

212.4. 结果

结果在 Out 正文中返回，作为 ArrayList<javax.naming.directory.SearchResult> 对象。

212.5. DIRCONTEXT

URI ldap:ldapserver 引用了 ID 为 ldapserver 的 Spring bean。ldapserver bean 可以定义如下：

```
<bean id="ldapserver" class="javax.naming.directory.InitialDirContext" scope="prototype">
  <constructor-arg>
    <props>
      <prop key="java.naming.factory.initial">com.sun.jndi.ldap.LdapCtxFactory</prop>
      <prop key="java.naming.provider.url">ldap://localhost:10389</prop>
      <prop key="java.naming.security.authentication">none</prop>
    </props>
  </constructor-arg>
</bean>
```

前面的示例声明了一个基于 Sun 的常规 LDAP DirContext，它将匿名连接到本地托管的 LDAP 服务器。

注意

不需要 DirContext 对象来支持根据合同的并发。因此，务必要在 bean 定义中使用 set, scope="prototype" 声明目录上下文，或者上下文支持并发。在 Spring 框架中，每次查找时都会对范围对象进行实例化。

212.6. SAMPLES

在上面的 Spring 配置后，以下代码示例发送一个 LDAP 请求，以过滤成员的搜索组。然后，从响应中提取通用名称。

```
ProducerTemplate<Exchange> template = exchange
  .getContext().createProducerTemplate();
```

```
Collection<?> results = (Collection<?>) (template
  .sendBody(
    "ldap:ldapserver?base=ou=mygroup,ou=groups,ou=system",
    "(member=uid=huntc,ou=users,ou=system)"));

if (results.size() > 0) {
  // Extract what we need from the device's profile

  Iterator<?> resultIter = results.iterator();
  SearchResult searchResult = (SearchResult) resultIter
      .next();
  Attributes attributes = searchResult
      .getAttributes();
  Attribute deviceCNAttr = attributes.get("cn");
  String deviceCN = (String) deviceCNAttr.get();

  ...
```

如果不需要特定的过滤器 - 例如，您只需要查找单个条目 - 指定通配符过滤器表达式。例如，如果 LDAP 条目具有通用名称，请使用如下过滤器表达式：

```
(cn=*)
```

212.6.1. 使用凭证绑定

Camel 最终用户授予本示例代码，其用于使用凭据绑定到 ldap 服务器。

```
Properties props = new Properties();
props.setProperty(Context.INITIAL_CONTEXT_FACTORY,
"com.sun.jndi.ldap.LdapCtxFactory");
props.setProperty(Context.PROVIDER_URL, "ldap://localhost:389");
props.setProperty(Context.URL_PKG_PREFIXES, "com.sun.jndi.url");
props.setProperty(Context.REFERRAL, "ignore");
props.setProperty(Context.SECURITY_AUTHENTICATION, "simple");
props.setProperty(Context.SECURITY_PRINCIPAL, "cn=Manager");
props.setProperty(Context.SECURITY_CREDENTIALS, "secret");

SimpleRegistry reg = new SimpleRegistry();
reg.put("myldap", new InitialLdapContext(props, null));

CamelContext context = new DefaultCamelContext(reg);
context.addRoutes(
   new RouteBuilder() {
      public void configure() throws Exception {
         from("direct:start").to("ldap:myldap?base=ou=test");
      }
   }
);
context.start();
```

```
ProducerTemplate template = context.createProducerTemplate();

Endpoint endpoint = context.getEndpoint("direct:start");
Exchange exchange = endpoint.createExchange();
exchange.getIn().setBody("(uid=test)");
Exchange out = template.send(endpoint, exchange);

Collection<SearchResult> data = out.getOut().getBody(Collection.class);
assert data != null;
assert !data.isEmpty();

System.out.println(out.getOut().getBody());

context.stop();
```

212.7. 配置 SSL

所有必要的是创建自定义套接字工厂，并在 InitialDirContext bean 中引用它，如下例所示。

SSL 配置

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.osgi.org/xmlns/blueprint/v1.0.0
http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd
              http://camel.apache.org/schema/blueprint http://camel.apache.org/schema/blueprint/camel-
blueprint.xsd">


   <sslContextParameters xmlns="http://camel.apache.org/schema/blueprint"
                        id="sslContextParameters">
      <keyManagers
            keyPassword="{{keystore.pwd}}">
         <keyStore
               resource="{{keystore.url}}"
               password="{{keystore.pwd}}"/>
      </keyManagers>
   </sslContextParameters>

   <bean id="customSocketFactory" class="zotix.co.util.CustomSocketFactory">
      <argument ref="sslContextParameters" />
   </bean>
   <bean id="ldapserver" class="javax.naming.directory.InitialDirContext" scope="prototype">
      <argument>
         <props>
            <prop key="java.naming.factory.initial" value="com.sun.jndi.ldap.LdapCtxFactory"/>
            <prop key="java.naming.provider.url" value="ldaps://lab.zotix.co:636"/>
            <prop key="java.naming.security.protocol" value="ssl"/>
            <prop key="java.naming.security.authentication" value="simple" />
            <prop key="java.naming.security.principal" value="cn=Manager,dc=example,dc=com"/>
```

```
            <prop key="java.naming.security.credentials" value="passw0rd"/>
            <prop key="java.naming.ldap.factory.socket"
                value="zotix.co.util.CustomSocketFactory"/>
        </props>
    </argument>
  </bean>
</blueprint>
```

自定义插槽工厂

```
import org.apache.camel.util.jsse.SSLContextParameters;

import javax.net.SocketFactory;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.TrustManagerFactory;
import java.io.IOException;
import java.net.InetAddress;
import java.net.Socket;
import java.security.KeyStore;

/**
 * The CustomSocketFactory. Loads the KeyStore and creates an instance of
SSLSocketFactory
 */
public class CustomSocketFactory extends SSLSocketFactory {

    private static SSLSocketFactory socketFactory;

    /**
     * Called by the getDefault() method.
     */
    public CustomSocketFactory() {

    }

    /**
     * Called by Blueprint DI to initialise an instance of SocketFactory
     *
     * @param sslContextParameters
     */
    public CustomSocketFactory(SSLContextParameters sslContextParameters) {
        try {
            KeyStore keyStore =
sslContextParameters.getKeyManagers().getKeyStore().createKeyStore();
            TrustManagerFactory tmf = TrustManagerFactory.getInstance("SunX509");
            tmf.init(keyStore);
            SSLContext ctx = SSLContext.getInstance("TLS");
            ctx.init(null, tmf.getTrustManagers(), null);
            socketFactory = ctx.getSocketFactory();
        } catch (Exception ex) {
            ex.printStackTrace(System.err);  /* handle exception */
        }
    }
```

```
    /**
     * Getter for the SocketFactory
     *
     * @return
     */
    public static SocketFactory getDefault() {
        return new CustomSocketFactory();
    }

    @Override
    public String[] getDefaultCipherSuites() {
        return socketFactory.getDefaultCipherSuites();
    }

    @Override
    public String[] getSupportedCipherSuites() {
        return socketFactory.getSupportedCipherSuites();
    }

    @Override
    public Socket createSocket(Socket socket, String string, int i, boolean bln) throws
IOException {
        return socketFactory.createSocket(socket, string, i, bln);
    }

    @Override
    public Socket createSocket(String string, int i) throws IOException {
        return socketFactory.createSocket(string, i);
    }

    @Override
    public Socket createSocket(String string, int i, InetAddress ia, int i1) throws IOException {
        return socketFactory.createSocket(string, i, ia, i1);
    }

    @Override
    public Socket createSocket(InetAddress ia, int i) throws IOException {
        return socketFactory.createSocket(ia, i);
    }

    @Override
    public Socket createSocket(InetAddress ia, int i, InetAddress ia1, int i1) throws IOException
{
        return socketFactory.createSocket(ia, i, ia1, i1);
    }
}
```

212.8. 另请参阅

- 配置 Camel

- *组件*
- *端点*
- *开始使用*

第 213 章 LDIF 组件

从 Camel 版本 2.20 开始提供

ldif 组件允许您从 LDIF 正文内容在 LDAP 服务器上执行更新。

此组件使用基本的 URL 语法来访问服务器。它使用 Apache DS LDAP 库来处理 LDIF。处理 LDIF 后，响应正文将是每个条目成功/失败的状态列表。

注意

Apache LDAP API 对 LDIF 语法错误非常敏感。如果不确定，请参阅单元测试来查看每种更改类型的示例。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-ldif</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

213.1. URI 格式

```
ldap:ldapServerBean[?options]
```

URI 的 ldapServerBean 部分引用 LdapConnection。这应该从工厂构建，以避免连接超时。LDIF 组件只支持生成者端点，这意味着 ldif URI 不能出现在路由开始时。

对于 SSL 配置，请参考 camel-ldap 组件，其中有一个设置自定义 SocketFactory 实例的示例。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

213.2. 选项

LDIF 组件没有选项。

LDIF 端点使用 URI 语法进行配置：

```
ldif:ldapConnectionName
```

使用以下路径和查询参数：

213.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
ldapConnectionName	**必需** 从 registry 中拉取的 LdapConnection bean 的名称。请注意，这必须是范围原型，以避免它在线程之间共享或使用已超时的连接。		字符串

213.2.2. 查询参数(1 参数)：

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

213.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.ldif.enabled	是否启用 ldif 组件的自动配置。这默认是启用的。		布尔值
camel.component.ldif.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

213.4. 正文类型：

正文可以是 LDIF 文件的 URL 或内联 LDIF 文件。要表示正文类型中的区别，内联 LDIF 必须以以下内容开头：

```
version: 1
```

如果没有，则组件将尝试将正文解析为 URL。

213.5. 结果

结果在 Out 正文中返回，作为 ArrayList<java.lang.String> 对象。这包括每个 LDIF 条目的 "success" 或 Exception 消息。

213.6. LDAPCONNECTION

URI ldif:ldapConnectionName 引用了 ID 为 ldapConnectionName 的 bean。ldapConnection 可以使用 LdapConnectionConfig bean 进行配置。请注意，范围必须具有设计 模型 范围，以避免共享连接或获取过时的连接。

LdapConnection bean 可以在 Spring XML 中定义：

```
<bean id="ldapConnectionOptions"
class="org.apache.directory.ldap.client.api.LdapConnectionConfig">
  <property name="ldapHost" value="${ldap.host}"/>
  <property name="ldapPort" value="${ldap.port}"/>
  <property name="name" value="${ldap.username}"/>
  <property name="credentials" value="${ldap.password}"/>
  <property name="useSsl" value="false"/>
  <property name="useTls" value="false"/>
</bean>

<bean id="ldapConnectionFactory"
class="org.apache.directory.ldap.client.api.DefaultLdapConnectionFactory">
  <constructor-arg index="0" ref="ldapConnectionOptions"/>
</bean>

<bean id="ldapConnection" factory-bean="ldapConnectionFactory" factory-
method="newLdapConnection" scope="prototype"/>
```

或者在 OSGi 蓝图.xml 中：

```
<bean id="ldapConnectionOptions"
class="org.apache.directory.ldap.client.api.LdapConnectionConfig">
  <property name="ldapHost" value="${ldap.host}"/>
```

```
  <property name="ldapPort" value="${ldap.port}"/>
  <property name="name" value="${ldap.username}"/>
  <property name="credentials" value="${ldap.password}"/>
  <property name="useSsl" value="false"/>
  <property name="useTls" value="false"/>
</bean>

<bean id="ldapConnectionFactory"
class="org.apache.directory.ldap.client.api.DefaultLdapConnectionFactory">
  <argument ref="ldapConnectionOptions"/>
</bean>

<bean id="ldapConnection" factory-ref="ldapConnectionFactory" factory-
method="newLdapConnection" scope="prototype"/>
```

213.7. SAMPLES

在上面的 Spring 配置后，以下代码示例发送一个 LDAP 请求，以过滤成员的搜索组。然后，从响应中提取通用名称。

```
ProducerTemplate<Exchange> template = exchange.getContext().createProducerTemplate();

List<?> results = (Collection<?>) template.sendBody("ldap:ldapConnection, "LDiff goes
here");

if (results.size() > 0) {
  // Check for no errors

  for (String result : results) {
    if ("success".equalTo(result)) {
      // LDIF entry success
    } else {
      // LDIF entry failure
    }
  }
}
```

213.8. LEVELDB

从 Camel 2.10 开始提供

leveldb 是一个轻量级且可嵌入的键值数据库。它允许与 Camel 结合使用，为各种 Camel 功能（如聚合器）提供持久支持。

它提供的当前功能：

- *LevelDBAggregationRepository*

213.8.1. 使用 LevelDBAggregationRepository

LevelDBAggregationRepository 是一个 聚合存储库，实时保留聚合的消息。这样可确保您不会松散消息，因为默认聚合器将使用仅在内存中的 AggregationRepository。

它有以下选项：

选项	类型	描述
repositoryName	字符串	必需的仓库名称。允许您将共享级别DB **File** 用于多个存储库。
persistentFileName	字符串	持久性存储的文件名。如果启动文件时不存在任何文件，则创建新文件。
levelDBFile	LevelDBFile	使用现有的 **org.apache.camel.component.leveldb.LevelDBFile** 实例。
sync	布尔值	Camel 2.12： LevelDBFile 是否应在写入时同步。默认值为 false。通过同步写入，可以确保始终等待所有写入假脱机到磁盘，因此不会造成松散更新。有关 async 与同步写入的详情，请参阅 LevelDB 文档。
returnOldExchange	布尔值	get 操作是否应该返回旧的现有 Exchange （如果存在）。默认情况下，此选项为 **false**，以优化，因为在聚合时我们不需要旧的交换。
useRecovery	布尔值	是否启用恢复。此选项默认为 **true**。启用 Camel Aggregator 自动恢复失败的聚合交换，并重新提交。
recoveryInterval	long	如果启用了恢复，则每次 x 次运行后台任务，以扫描失败的交换以恢复并重新提交。默认情况下，这个间隔为 5000 millis。
maximumRedeliveries	int	允许您限制恢复的交换的最大重新发送尝试次数。如果启用，如果所有重新发送尝试都失败，则交换将移到死信频道。默认情况下禁用这个选项。如果使用这个选项，则必须提供 **deadLetterUri** 选项。

选项	类型	描述
deadLetterUri	字符串	一个死信频道的端点 uri，将移动耗尽恢复的交换。如果使用这个选项，则必须提供 **maximumRedeliveries** 选项。

必须提供 repositoryName 选项。然后，必须提供 persistentFileName 或 levelDBFile。

213.8.2. 持久性时保留的内容

LevelDBAggregationRepository 将仅保留任何 Serializable 兼容消息正文数据类型。消息标头必须是原语 / 字符串 / 等。如果数据类型不是这样一个类型，则会丢弃它，并记录 WARN。它只会保留消息正文和 Message 标头。Exchange 属性 不会被保留。

213.8.3. 恢复

LevelDBAggregationRepository 默认恢复任何失败的 Exchange。它通过在持久性存储中扫描失败的交换的后台任务来实现此目的。您可以使用 checkInterval 选项设置此任务运行的频率。恢复充当事务处理，可确保 Camel 会尝试恢复并恢复失败的交换。发现恢复的所有交换都将从持久存储恢复并重新提交并再次发送出。

当交换被恢复/冗余时，会设置以下标头：

标头	类型	描述
Exchange.REDELIVERED	布尔值	设置为 true，表示 Exchange 正在重新设计。
Exchange.REDELIVERY_COUNTER	整数	从 1 开始重新发送尝试。

只有在成功处理交换时，才会将其标记为 complete，只有在 AggregationRepository 上调用 confirm 方法时才会发生这种情况。这意味着，如果同一交换再次失败，它将被重试，直到成功为止。

您可以使用选项 maximumRedeliveries 来限制给定恢复的交换的最大重新发送尝试次数。您还必须设置 deadLetterUri 选项，以便 Camel 知道在 max Redeliveries 正在命中时要发送交换的位置。

您可以在 camel-leveldb 的单元测试中看到一些示例，例如 此测试。

213.8.3.1. 在 Java DSL 中使用 LevelDBAggregationRepository

在本例中，我们希望在 target/data/leveldb.dat 文件中持久保留聚合的消息。

213.8.3.2. 在 Spring XML 中使用 LevelDBAggregationRepository

同一示例，但使用 Spring XML：

213.8.4. 依赖项

要在 camel 路由中使用 LevelDB，您需要添加对 camel-leveldb 的依赖。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-leveldb</artifactId>
  <version>2.10.0</version>
</dependency>
```

213.8.5. 另请参阅

- 配置 Camel
- 组件
- 端点

- *开始使用*
- *聚合器*
- *HawtDB*
- *组件*

第 214 章 日志组件

从 Camel 版本 1.1 开始提供

log: component 将消息交换到底层的日志机制。

Camel 使用 sfl4j，它允许您通过它配置日志记录：

- Log4j
- Logback
- Java Util Logging

214.1. URI 格式

```
log:loggingCategory[?options]
```

其中 loggingCategory 是要使用的日志记录类别的名称。您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

INFO:*使用 Registry* 中的 Logger 实例作为 Camel 2.12.4/2.13.1，如果 Registry 中有单一的 org.slf4j.Logger 实例，则 loggingCategory 不再用于创建日志记录器实例。改为使用注册的实例。另外，也可以使用 ?logger=#myLogger URI 参数来引用特定的 Logger 实例。最后，如果没有注册并且 URI 日志记录器 参数，则使用 loggingCategory 创建 logger 实例。

例如，日志端点通常使用 level 选项指定 日志记录级别，如下所示：

```
log:org.apache.camel.example?level=DEBUG
```

默认日志记录器记录每个交换(常规日志记录)。但是，Camel 也附带了 Throughput logger，它会在指定 groupSize 选项时使用。

*TIP:*Also a log in the DSL* 也可以在 DSL 中直接有一个日志，但它具有不同的目的。它适用于轻量级和人类日志。请参阅 LogEIP 的详情。*

214.2. 选项

日志组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
exchangeFormatter (advanced)	设置自定义 ExchangeFormatter，将 Exchange 转换为适合日志记录的字符串。如果没有指定，则默认为 DefaultExchangeFormatter。		ExchangeFormatter
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

日志端点使用 URI 语法进行配置：

```
log:loggerName
```

使用以下路径和查询参数：

214.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
loggerName	**必需** 要使用的日志记录器名称		字符串

214.2.2. 查询参数(26 参数)：

Name	描述	默认值	类型
groupActiveOnly (producer)	如果为 true，则在没有收到时间间隔的新消息时隐藏统计信息（如果为 false），无论消息流量是什么，都会隐藏统计信息。	true	布尔值
groupDelay (producer)	设置统计数据的初始延迟（在 millis 中）		Long

Name	描述	默认值	类型
groupInterval (producer)	如果指定，则会按这个时间间隔对消息统计进行分组（在 millis 中）		Long
groupSize (producer)	指定用于吞吐量日志的组群大小的整数。		整数
level (producer)	要使用的日志记录级别。默认值为 INFO。	INFO	字符串
logMask (producer)	如果为 true，请屏蔽日志中密码或密码短语等敏感信息。		布尔值
marker (producer)	要使用的可选 Marker 名称。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
maxChars (formatting)	限制每行记录的字符数。	10000	int
多行 （格式）	如果启用，则每个信息会在一行上输出。	false	布尔值
showAll (formatting)	打开打开所有选项的快速选项。（多行，如果要使用 maxChars，则必须手动设置 maxChars）	false	布尔值
showBody (formatting)	显示消息正文。	true	布尔值
showBodyType (formatting)	显示正文 Java 类型。	true	布尔值
showCaughtException (formatting)	f the Exchange has caught exception, show the exception message (no stack trace)。一个发现的异常作为属性存储在交换中（使用键 org.apache.camel.Exchange""EXCEPTION_CAUGHT），对于 doCatch 可以捕获异常的实例。	false	布尔值
showException (formatting)	如果交换存在异常，请显示异常消息（无堆栈追踪）	false	布尔值
showExchangeId (formatting)	显示唯一的交换 ID。	false	布尔值
showExchangePattern (formatting)	显示消息交换模式（或 MEP 表示短）。	true	布尔值
showFiles (formatting)	如果启用的 Camel 将输出文件	false	布尔值

Name	描述	默认值	类型
showFuture (formatting)	如果启用的 Camel 将在 Future 对象上等待其完成以获取有效负载记录。	false	布尔值
showHeaders (formatting)	显示消息标头。	false	布尔值
showOut （信息）	如果交换的消息为 out，则显示 out 消息。	false	布尔值
showProperties (formatting)	显示交换属性。	false	布尔值
showStackTrace （信息）	如果交换存在异常，显示堆栈追踪。仅在启用了其中一个 showAll, showException 或 showCaughtException 时才有效。	false	布尔值
showStreams (formatting)	Camel 是否应该显示流正文（如 java.io.InputStream）。请注意，如果您启用了这个选项，那么以后可能无法访问消息正文，因为此日志记录器已读取流。要补救这一点，您必须使用流缓存。	false	布尔值
skipBodyLineSeparator (formatting)	在记录消息正文时是否跳过行分隔符。这允许在一行中记录消息正文，将此选项设置为 false 将保留来自正文的任何行分隔符，然后将正文记录为。	true	布尔值
样式 （格式）	设置要使用的输出风格。	default	OutputStyle

214.3. 常规日志记录器示例

在以下路由中，我们在处理顺序前将传入的顺序记录在 DEBUG 级别：

```
from("activemq:orders").to("log:com.mycompany.order?
level=DEBUG").to("bean:processOrder");
```

或使用 Spring XML 定义路由：

```
<route>
  <from uri="activemq:orders"/>
  <to uri="log:com.mycompany.order?level=DEBUG"/>
  <to uri="bean:processOrder"/>
</route>
```

214.4. 带有 FORMATTER 示例的常规日志记录器

在以下路由中，我们在处理订购前在 INFO 级别记录传入的订单。

```
from("activemq:orders").
    to("log:com.mycompany.order?showAll=true&multiline=true").to("bean:processOrder");
```

214.5. 吞吐量日志记录器，带有 GROUPSIZE 示例

在以下路由中，我们将以 DEBUG 级别分组的传入订单的吞吐量记录到 10 个消息。

```
from("activemq:orders").
    to("log:com.mycompany.order?level=DEBUG&groupSize=10").to("bean:processOrder");
```

214.6. 吞吐量日志记录器，带有 GROUPINTERVAL 示例

此路由将导致每个 10 的消息统计记录每个 10，即使没有任何消息流量，也会显示初始 60s 延迟和统计。

```
from("activemq:orders").
    to("log:com.mycompany.order?
level=DEBUG&groupInterval=10000&groupDelay=60000&groupActiveOnly=false").to("bean:pr
ocessOrder");
```

将记录以下内容：

```
"Received: 1000 new messages, with total 2000 so far. Last group took: 10000 millis which is: 100
messages per second. average: 100"
```

214.7. 屏蔽敏感信息，如密码

从 Camel 2.19 开始提供

您可以通过将 logMask 标志设置为 true 来为日志记录启用安全掩码。请注意，这个选项也会影响日志 EIP。

在 CamelContext 级别启用 Java DSL 中的掩码：

```
camelContext.setLogMask(true);
```

在 XML 中：

```
<camelContext logMask="true">
```

您还可以在端点级别打开|关闭。要在端点级别的 Java DSL 中启用掩码，请在日志端点的 URI 中添加 logMask=true 选项：

```
from("direct:start").to("log:foo?logMask=true");
```

在 XML 中：

```
<route>
  <from uri="direct:foo"/>
  <to uri="log:foo?logMask=true"/>
</route>
```

org.apache.camel.processor.DefaultMaskingFormatter 默认用于掩码。如果要使用自定义屏蔽格式器，请将其放在带有名称 CamelCustomLogMask 的 registry 中。请注意，掩码格式器必须实施 org.apache.camel.spi.MaskingFormatter。

214.8. 对日志输出进行完全自定义

从 Camel 2.11 开始提供

使用 #Formatting 部分中概述的选项，您可以控制日志记录器的许多输出。但是，日志行始终遵循此结构：

```
Exchange[Id:ID-machine-local-50656-1234567901234-1-2, ExchangePattern:InOut,
Properties:{CamelToEndpoint=log://org.apache.camel.component.log.TEST?showAll=true,
CamelCreatedTimestamp=Thu Mar 28 00:00:00 WET 2013},
Headers:{breadcrumbId=ID-machine-local-50656-1234567901234-1-1}, BodyType:String, Body:Hello
World, Out: null]
```

在某些情况下，这个格式不适合，这可能是因为您需要...

- ... 过滤打印的标头和属性，以便在 Insights 和详细程度之间达到平衡。

- ... 将日志消息调整为您认为最易读的任何消息。
- ... 通过日志 mining 系统（如 Splunk ）为摘要定制日志消息。
- ... 打印特定正文类型的方式不同。
- ... 等。

每当您需要绝对自定义时，您可以创建一个实现 ExchangeFormatter 接口的类。在您有权访问完整交换 的格式(Exchange) 方法中，您可以选择并提取所需的精确信息，以自定义方式对其进行格式化并返回。返回值将变为最终日志消息。

您可以通过两种方式之一获取您的自定义 ExchangeFormatter ：

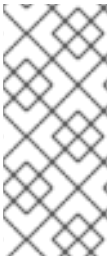
在 Registry 中明确实例化 LogComponent:

```
<bean name="log" class="org.apache.camel.component.log.LogComponent">
  <property name="exchangeFormatter" ref="myCustomFormatter" />
</bean>
```

214.8.1. 配置惯例：*

只需通过将 bean 注册到名称 logFormatter 即可；日志组件足以自动提取它。

```
<bean name="logFormatter" class="com.xyz.MyCustomExchangeFormatter" />
```



注意

ExchangeFormatter 应用到 Camel 上下文中的所有日志端点。如果您需要不同端点的不同 ExchangeFormatter，请根据需要实例化 LogComponent，并使用相关的 bean 名称作为端点前缀。

在使用自定义日志格式器时，从 Camel 2.11.2/2.12 开始，您可以在 log uri 中指定参数，该参数在自定义日志格式ter上配置。虽然当您这样做时，您应该将"logFormatter"定义为特征范围，因此如果您具

有不同的参数，例如：

```
<bean name="logFormatter" class="com.xyz.MyCustomExchangeFormatter" scope="prototype"/>
```

然后，我们可以使用带有不同选项的 log uri 进行 Camel 路由：

```
<to uri="log:foo?param1=foo&amp;param2=100"/>

<to uri="log:bar?param1=bar&amp;param2=200"/>
```

214.9. 在 OSGI 中使用日志组件

从 Camel 2.12.4/2.13.1 开始改进

在 OSGi 中使用日志组件（例如，在 Karaf 中）时，PAX 日志记录提供了底层的日志记录机制。它搜索调用 org.slf4j.LoggerFactory.getLogger （） 方法的捆绑包，并将捆绑包与 logger 实例相关联。在不指定自定义 org.sfl4j.Logger 实例的情况下，由 Log 组件创建的日志记录器与 camel-core 捆绑包关联。

在某些情况下，与 logger 关联的捆绑包应该是包含路由定义的捆绑包。要做到这一点，可以在 Registry 中注册 org.slf4j.Logger 的单个实例，或使用 logger URI 参数引用它。

214.10. 另请参阅

- LogEIP，用于在 DSL 中直接使用 日志。

第 215 章 LUCENE 组件

从 Camel 版本 2.2 开始提供

lucene 组件基于 Apache Lucene 项目。Apache Lucene 是一个功能强大、功能全面的文本搜索引擎库完全使用 Java 编写。有关 Lucene 的详情，请查看以下链接

- http://lucene.apache.org/java/docs/
- http://lucene.apache.org/java/docs/features.html

camel 中的 lucene 组件有助于在企业集成模式和场景中集成和使用 Lucene 端点。lucene 组件执行以下操作

- 当有效负载发送到 Lucene 端点时，构建可搜索的文档索引
- 有助于在 Camel 中执行索引搜索

此组件仅支持生成者端点。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-lucene</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

215.1. URI 格式

```
lucene:searcherName:insert[?options]
lucene:searcherName:query[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

215.2. 插入选项

Lucene 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
config (advanced)	使用共享的 lucene 配置		LuceneConfiguration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Lucene 端点使用 URI 语法进行配置：

```
lucene:host:operation
```

使用以下路径和查询参数：

215.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
主机	**必需的** lucene 服务器的 URL		字符串
operation	**需要执行的操作**，如 insert 或 query。		LuceneOperation

215.2.2. 查询参数(5 参数)：

Name	描述	默认值	类型
analyzer (producer)	分析器构建 TokenStreams，用于分析文本。因此，它代表了从文本中提取索引术语的策略。分析器的值可以是扩展抽象类 org.apache.lucene.analysis.Analyzer 的任何类。Lucene 还开箱即用提供一组丰富的分析器		analyzer
indexDir (producer)	在对指定分析器分析文档时创建索引文件的文件系统目录		File

Name	描述	默认值	类型
maxHits (producer)	限制搜索操作的结果集的整数值		int
srcDir (producer)	可选目录，其中包含要在生成者启动时分析和添加到索引中的文件。		File
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

215.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 11 个选项，如下所列。

Name	描述	默认值	类型
camel.component.lucene.config.analyzer	分析器构建 TokenStreams，用于分析文本。因此，它代表了从文本中提取索引术语的策略。分析器的值可以是扩展抽象类 org.apache.lucene.analysis.Analyzer 的任何类。Lucene 还开箱即用提供一组丰富的分析器		analyzer
camel.component.lucene.config.authority			字符串
camel.component.lucene.config.host	lucene 服务器的 URL		字符串
camel.component.lucene.config.index-directory	在对指定分析器分析文档时创建索引文件的文件系统目录		File
camel.component.lucene.config.lucene-version			Version
camel.component.lucene.config.max-hits	限制搜索操作的结果集的整数值		整数
camel.component.lucene.config.operation	要执行的操作，如 insert 或 query。		LuceneOperation

Name	描述	默认值	类型
camel.component.lucene.config.source-directory	可选目录，其中包含要在生成者启动时分析和添加到索引中的文件。		File
camel.component.lucene.config.uri			URI
camel.component.lucene.enabled	启用 lucene 组件	true	布尔值
camel.component.lucene.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

215.4. 将消息发送到/从缓存中发送/接收消息

215.4.1. 消息标头

标头	描述
查询	要在索引上执行的 Lucene 查询。查询可能包含通配符和短语
RETURN_LUCENE_DOCS	Camel 2.15 ： 将此标头设置为 true，以在返回命中信息时包含实际的 Lucene 文档。

215.4.2. Lucene Producers

此组件支持 2 producer 端点。

insert - insert producer 通过在传入交换中分析正文并将其与令牌("content")关联来构建可搜索的索引。query - query producer 对预先创建的索引执行搜索。查询使用可搜索的索引来执行分数和基于相关性的搜索。通过传入交换发送查询包含名为"QUERY"的标头属性名称。标头属性 'QUERY' 的值是 Lucene Query。有关如何创建 Lucene Queries 的详情，请查看 http://lucene.apache.org/java/3_0_0/queryparsersyntax.html

215.4.3. Lucene Processor

有名为 LuceneQueryProcessor 的处理器可用于对 lucene 执行查询，而无需创建制作者。

215.5. LUCENE 使用示例

215.5.1. 示例 1：创建 Lucene 索引

```
RouteBuilder builder = new RouteBuilder() {
   public void configure() {
      from("direct:start").
         to("lucene:whitespaceQuotesIndex:insert?
            analyzer=#whitespaceAnalyzer&indexDir=#whitespace&srcDir=#load_dir").
         to("mock:result");
   }
};
```

215.5.2. 示例 2：将属性加载到 Camel 上下文中的 JNDI 注册表

```
@Override
protected JndiRegistry createRegistry() throws Exception {
  JndiRegistry registry =
        new JndiRegistry(createJndiContext());
  registry.bind("whitespace", new File("./whitespaceIndexDir"));
  registry.bind("load_dir",
        new File("src/test/resources/sources"));
  registry.bind("whitespaceAnalyzer",
        new WhitespaceAnalyzer());
  return registry;
}
...
CamelContext context = new DefaultCamelContext(createRegistry());
```

215.5.3. 示例 2：使用 Query Producer 执行搜索

```
RouteBuilder builder = new RouteBuilder() {
   public void configure() {
      from("direct:start").
        setHeader("QUERY", constant("Seinfeld")).
        to("lucene:searchIndex:query?
          analyzer=#whitespaceAnalyzer&indexDir=#whitespace&maxHits=20").
        to("direct:next");

      from("direct:next").process(new Processor() {
        public void process(Exchange exchange) throws Exception {
          Hits hits = exchange.getIn().getBody(Hits.class);
          printResults(hits);
        }

        private void printResults(Hits hits) {
          LOG.debug("Number of hits: " + hits.getNumberOfHits());
          for (int i = 0; i < hits.getNumberOfHits(); i++) {
```

```
            LOG.debug("Hit " + i + " Index Location:" + hits.getHit().get(i).getHitLocation());
            LOG.debug("Hit " + i + " Score:" + hits.getHit().get(i).getScore());
            LOG.debug("Hit " + i + " Data:" + hits.getHit().get(i).getData());
          }
        }
      }).to("mock:searchResult");
  }
};
```

215.5.4. 示例 3：使用 Query Processor 执行搜索

```
RouteBuilder builder = new RouteBuilder() {
  public void configure() {
    try {
      from("direct:start").
        setHeader("QUERY", constant("Rodney Dangerfield")).
        process(new LuceneQueryProcessor("target/stdindexDir", analyzer, null, 20)).
        to("direct:next");
    } catch (Exception e) {
      e.printStackTrace();
    }

    from("direct:next").process(new Processor() {
      public void process(Exchange exchange) throws Exception {
        Hits hits = exchange.getIn().getBody(Hits.class);
        printResults(hits);
      }

      private void printResults(Hits hits) {
        LOG.debug("Number of hits: " + hits.getNumberOfHits());
        for (int i = 0; i < hits.getNumberOfHits(); i++) {
          LOG.debug("Hit " + i + " Index Location:" + hits.getHit().get(i).getHitLocation());
          LOG.debug("Hit " + i + " Score:" + hits.getHit().get(i).getScore());
          LOG.debug("Hit " + i + " Data:" + hits.getHit().get(i).getData());
        }
      }
    }).to("mock:searchResult");
  }
};
```

第 216 章 LUMBERJACK 组件

从 Camel 版本 2.18 开始提供

Lumberjack 组件使用 Lumberjack 协议从 Filebeat （实例）检索通过网络发送的日志。网络通信可以通过 SSL 保护。

此组件只支持消费者端点。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-lumberjack</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

216.1. URI 格式

```
lumberjack:host
lumberjack:host:port
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

216.2. 选项

Lumberjack 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
sslContextParameters (security)	设置用于所有端点的默认 SSL 配置。您还可以在端点级别进行直接配置。		SSLContextParameters
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Lumberjack 端点使用 URI 语法进行配置：

```
lumberjack:host:port
```

使用以下路径和查询参数：

216.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
主机	**需要** 侦听 Lumberjack 的网络接口		字符串
port	要侦听 Lumberjack 的网络端口	5044	int

216.2.2. 查询参数(5 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
sslContextParameters (consumer)	SSL 配置		SSLContextParameters
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

216.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.lumberjack.enabled	启用 lumberjack 组件	true	布尔值
camel.component.lumberjack.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.lumberjack.ssl-context-parameters	设置用于所有端点的默认 SSL 配置。您还可以在端点级别进行直接配置。选项是 org.apache.camel.util.jsse.SSLContextParameters 类型。		字符串
camel.component.lumberjack.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

216.4. 结果

结果正文是一个 Map<String, Object> 对象。

216.5. LUMBERJACK 用法示例

216.5.1. 示例 1：流日志消息

```
RouteBuilder builder = new RouteBuilder() {
    public void configure() {
        from("lumberjack:0.0.0.0").              // Listen on all network interfaces using the default
port
```

```
        setBody(simple("${body[message]}")).    // Select only the log message
        to("stream:out");                       // Write it into the output stream
    }
};
```

第 217 章 LZF DEFLATE COMPRESSION DATAFORMAT

从 Camel 版本 2.17 开始提供

LZF Data Format 是消息压缩和解压缩格式。它使用 LZF 保护算法。使用 LZF 压缩的消息可以正常使用 LZF 解压缩，然后再在端点中使用 LZF 解压缩。当您处理大量基于 XML 和文本的有效负载时，或者读取之前使用 LZF algotithm 的消息时，压缩功能非常有用。

217.1. 选项

LZF Deflate Compression dataformat 支持 2 个选项，如下所列。

Name	默认值	Java 类型	描述
usingParallelCompression	false	布尔值	使用多个处理内核启用编码（压缩）。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

217.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.lzf.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.lzf.enabled	启用 lzf dataformat	true	布尔值
camel.dataformat.lzf.using-parallel-compression	使用多个处理内核启用编码（压缩）。	false	布尔值

ND

217.3. MARSHAL

在本例中，我们将常规文本/XML 有效负载放入压缩有效负载使用 LZF 压缩格式，并将其发送到名为 MY_QUEUE 的 ActiveMQ 队列。

```
from("direct:start").marshal().lzf().to("activemq:queue:MY_QUEUE");
```

217.4. UNMARSHAL

在本例中，我们从名为 MY_QUEUE 的 ActiveMQ 队列到其原始格式，并将它转发到 UnGZippedMessageProcessor。

```
from("activemq:queue:MY_QUEUE").unmarshal().lzf().process(new
UnCompressedMessageProcessor());
```

217.5. 依赖项

要在 camel 路由中使用LZF 压缩，您需要添加对实现此数据格式的 camel-lzf 的依赖。

如果使用 Maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-lzf</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

第 218 章 邮件组件

从 Camel 版本 1.0 开始提供

邮件组件通过 Spring 邮件支持和底层 JavaMail 系统提供对电子邮件的访问。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-mail</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

警告

Geronimo mail .jar

我们已发现，geronimo mail .jar (v1.6)在轮询邮件与附件时有一个错误。它无法正确识别 Content-Type。因此，如果您将 .jpeg 文件附加到邮件并轮询了它，则 Content-Type 会解析为 text/plain，而不是 image/jpeg。因此，我们添加了 org.apache.camel.component.ContentTypeResolver SPI 接口，它可让您提供自己的实施，并通过根据文件名返回正确的 Mime 类型来修复这个错误。因此，如果文件名以 jpeg/jpg 结尾，您可以返回 image/jpeg。

您可以在 MailComponent 实例或 MailEndpoint 实例上设置自定义解析器。

提示

POP3 或 IMAP POP3 有一些限制，如果可能，建议最终用户使用 IMAP。

INFO：使用 mock-mail 测试，您可以使用模拟框架进行单元测试，这可让您在不需要真实邮件服务器的情况下进行测试。但是，当您进入生产环境或其他需要向真实邮件服务器发送邮件时，您应该记住不要

包含 mock-mail。仅存在 classpath 上的 mock-javamail.jar 表示它将启动并避免发送邮件。

218.1. URI 格式

邮件端点可以具有以下 URI 格式之一（分别用于协议、SMTP、POP3 或 IMAP）：

```
smtp://[username@]host[:port][?options]
pop3://[username@]host[:port][?options]
imap://[username@]host[:port][?options]
```

邮件组件还支持这些协议的安全变体（通过 SSL 进行层）。您可以通过在方案中添加 s 来启用安全协议：

```
smtps://[username@]host[:port][?options]
pop3s://[username@]host[:port][?options]
imaps://[username@]host[:port][?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

218.2.

邮件组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	设置邮件配置		MailConfiguration
contentTypeResolver (advanced)	解析器决定用于文件附加的 Content-Type。		ContentTypeResolver
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

218.3.

Mail 端点使用 URI 语法进行配置：

```
imap:host:port
```

使用以下路径和查询参数：

218.3.1. 路径参数(2 参数)：

Name	描述	默认值	类型
主机	**必需** 邮件服务器主机名		字符串
port	邮件服务器的端口号		int

218.3.2. 查询参数(62 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
closeFolder (consumer)	消费者是否应该在轮询后关闭文件夹。将此选项设置为 false 并且也具有 disconnect=false，然后消费者在轮询之间保持打开的文件夹。	true	布尔值
copyTo (consumer)	在处理邮件后，可以将其复制到具有指定名称的邮件文件夹中。您可以覆盖此配置值，使用键 copyTo 的标头，允许您将消息复制到运行时配置的文件夹名称。		字符串
delete (consumer)	在消息被处理后删除它们。这可以通过在邮件消息中设置 DELETED 标志来完成。如果为 false，则改为设置 SEEN 标志。从 Camel 2.10 开始，您可以通过设置带有 delete 键的标头来覆盖这个配置选项，以确定是否应删除邮件。	false	布尔值
disconnect (consumer)	消费者是否应该在轮询后断开连接。如果启用此选项，则会强制 Camel 每次轮询上进行连接。	false	布尔值

Name	描述	默认值	类型
handleFailedMessage (consumer)	如果邮件使用者无法检索给定邮件，则此选项允许处理消费者的错误处理程序导致的异常。通过在消费者上启用网桥错误处理程序，则 Camel 路由错误处理程序可以改为处理异常。默认行为是消费者抛出异常，来自批处理中的邮件将无法由 Camel 进行路由。	false	布尔值
maxMessagesPerPoll (consumer)	指定每个轮询要收集的最大消息数。默认情况下，不设置最大值。可用于设置限制，例如 1000，以避免在服务器启动时下载数千个文件。设置 0 或负值以禁用这个选项。		int
mimeDecodeHeaders (consumer)	此选项为邮件标头启用透明 MIME 解码和折叠功能。	false	布尔值
peek (consumer)	将 javax.mail.Message 标记为处理邮件之前。这只适用于 IMAPMessage 消息类型。通过使用在邮件服务器上标记为 SEEN，邮件不会强制标记为 SEEN，如果 Camel 出现错误处理，则我们可以回滚邮件消息。	true	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
skipFailedMessage (consumer)	如果邮件使用者无法检索给定邮件，则此选项允许跳过邮件并继续检索下一个邮件。默认行为是消费者抛出异常，来自批处理中的邮件将无法由 Camel 进行路由。	false	布尔值
unseen (consumer)	是否只被不可编辑的邮件限制。	true	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
fetchSize (consumer)	设置轮询期间要消耗的消息的最大数量。如果邮箱文件夹包含大量邮件，则这可用于避免过载邮件服务器。默认值 -1 表示没有获取大小，将消耗所有信息。将值设为 0 是一个特殊情况，Camel 将根本不使用任何消息。	-1	int
folderName (consumer)	要轮询的文件夹。	INBOX	字符串

Name	描述	默认值	类型
mailUidGenerator (consumer)	可插拔 MailUidGenerator，允许使用自定义逻辑来生成邮件邮件的 UUID。		MailUidGenerator
mapMailMessage (consumer)	指定 Camel 是否应该将接收的邮件映射到 Camel body/headers。如果设置为 true，邮件消息的正文将映射到 Camel IN 消息的正文，并且邮件标头映射到 IN 标头。如果此选项设为 false，则 IN 消息包含原始 javax.mail.Message。您可以通过调用 exchange.getIn（）.getBody (javax.mail.Message.class)来检索这个原始消息。	true	布尔值
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
postProcessAction (consumer)	指的是 MailBoxPostProcessAction，用于在正常处理完后处理邮箱上的任务。		MailBoxPostProcess Action
bcc (producer)	设置 BCC 电子邮件地址。使用逗号分隔多个电子邮件地址。		字符串
CC (producer)	设置 CC 电子邮件地址。使用逗号分隔多个电子邮件地址。		字符串
from (producer)	来自电子邮件地址	camel@localhost	字符串
replyTo (producer)	Reply-To 收件人（响应邮件的接收器）。使用逗号分隔多个电子邮件地址。		字符串
subject (producer)	正在发送的消息主题。注意：在标头中设置主题优先于这个选项。		字符串
to (producer)	设置 To 电子邮件地址。使用逗号分隔多个电子邮件地址。		字符串
javaMailSender (producer)	使用自定义 org.apache.camel.component.mail.JavaMailSender 来发送电子邮件。		JavaMailSender
additionalJavaMail Properties (advanced)	设置其他 java 邮件属性，它将根据所有其他选项附加/覆盖任何默认属性。如果您需要添加一些特殊选项，但希望让其他选项保留原样，这将非常有用。		Properties

Name	描述	默认值	类型
alternativeBodyHeader (advanced)	指定包含替代电子邮件正文的 IN 消息标头的密钥。例如，如果您以文本/html 格式发送电子邮件，并希望为非HTML 电子邮件客户端提供替代邮件正文，请将替代邮件正文设置为标头。	CamelMailAlternativeBody	字符串
attachmentsContentTransferEncodingResolver (advanced)	使用自定义 AttachmentsContentTransferEncodingResolver 来解决要用于附件的 content-type-encoding。		AttachmentsContentTransferEncodingResolver
绑定 (advanced)	设置用于从 Camel 消息转换到邮件消息的绑定		MailBinding
connectionTimeout (advanced)	连接超时（以毫秒为单位）。	30000	int
contentType (advanced)	邮件邮件内容类型。将 text/html 用于 HTML 邮件。	text/plain	字符串
contentTypeResolver (advanced)	解析器决定用于文件附加的 Content-Type。		ContentTypeResolver
debugMode (advanced)	在底层邮件框架上启用调试模式。SUN Mail 框架默认将调试消息记录到 system.out。	false	布尔值
headerFilterStrategy (advanced)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 来过滤标头。		HeaderFilterStrategy
ignoreUnsupportedCharset (advanced)	选项，让 Camel 在发送邮件时忽略本地 JVM 中不支持的 charset。如果不支持 charset=XXX （其中 XXX 代表不受支持的 charset），则从 content-type 中删除 charset，它依赖于平台默认值。	false	布尔值
ignoreUriScheme (advanced)	选项，让 Camel 在发送邮件时忽略本地 JVM 中不支持的 charset。如果不支持 charset=XXX （其中 XXX 代表不受支持的 charset），则从 content-type 中删除 charset，它依赖于平台默认值。	false	布尔值
会话 （高级）	指定 camel 应该用于所有邮件交互的邮件会话。在由某些其他资源（如 modular 容器）创建和管理邮件会话时很有用。如果没有指定，Camel 会自动为您创建邮件会话。		会话
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
useInlineAttachments (advanced)	是否使用 disposition inline 或 attachment。	false	布尔值

Name	描述	默认值	类型
idempotentRepository (filter)	一个可插拔式存储库 org.apache.camel.spi.IdempotentRepository，它允许从同一邮箱使用集群，并让存储库协调邮件是否对消费者有效。默认情况下，没有使用软件仓库。		IdempotentRepository
idempotentRepositoryRemoveOnCommit (filter)	使用幂等存储库时，当邮件被成功处理并提交时，该消息 ID 会从幂等存储库（默认）中删除，或者保存在存储库中。默认情况下，其假定消息 id 是唯一的，且没有在存储库中保留值，因为邮件消息将被标记为 seen/moved 或 delete，以防止它被再次使用。因此，在 idempotent 存储库中存储了消息 id 有较少的值。但是，此选项允许存储消息 id，适用于您可能具有的任何原因。	true	布尔值
searchTerm (filter)	指的是 javax.mail.search.SearchTerm，它允许根据特定日期等搜索条件（如主题、正文等）过滤邮件。		SearchTerm
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。	60000	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler

Name	描述	默认值	类型
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
sortTerm (sort)	排序消息的顺序.仅对 IMAP 进行原生支持。使用 POP3 或 IMAP 服务器没有 SORT 功能时，模拟某些程度。		字符串
dummyTrustManager (security)	使用 dummy 安全设置来信任所有证书。应该只用于开发模式，而不应用于生产环境。	false	布尔值
密码 (security)	登录的密码		字符串
sslContextParameters (security)	使用 SSLContextParameters 配置安全性。		SSLContextParameters
用户名 (security)	登录的用户名		字符串

218.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 48 个选项，如下所列。

Name	描述	默认值	类型
camel.component.mail.configuration.additional-java-mail-properties	设置其他 java 邮件属性，它将根据所有其他选项附加/覆盖任何默认属性。如果您需要添加一些特殊选项，但希望让其他选项保留原样，这将非常有用。		Properties
camel.component.mail.configuration.alternative-body-header	指定包含替代电子邮件正文的 IN 消息标头的密钥。例如，如果您以文本/html 格式发送电子邮件，并希望为非HTML 电子邮件客户端提供替代邮件正文，请将替代邮件正文设置为标头。	CamelMailAlternativeBody	字符串

Name	描述	默认值	类型
camel.component.mail.configuration.attachments-content-transfer-encoding-resolver	使用自定义 AttachmentsContentTransferEncodingResolver 来解决要用于附件的 content-type-encoding。		AttachmentsContentTransferEncodingResolver
camel.component.mail.configuration.bcc	设置 BCC 电子邮件地址。使用逗号分隔多个电子邮件地址。		字符串
camel.component.mail.configuration.cc	设置 CC 电子邮件地址。使用逗号分隔多个电子邮件地址。		字符串
camel.component.mail.configuration.close-folder	消费者是否应该在轮询后关闭文件夹。将此选项设置为 false 并且也具有 disconnect=false，然后消费者在轮询之间保持打开的文件夹。	true	布尔值
camel.component.mail.configuration.connection-timeout	连接超时（以毫秒为单位）。	30000	整数
camel.component.mail.configuration.content-type	邮件邮件内容类型。将 text/html 用于 HTML 邮件。	text/plain	字符串
camel.component.mail.configuration.copy-to	在处理邮件后，可以将其复制到具有指定名称的邮件文件夹中。您可以覆盖此配置值，使用键 copyTo 的标头，允许您将消息复制到运行时配置的文件夹名称。		字符串
camel.component.mail.configuration.debug-mode	在底层邮件框架上启用调试模式。SUN Mail 框架默认将调试消息记录到 system.out。	false	布尔值
camel.component.mail.configuration.delete	在消息被处理后删除它们。这可以通过在邮件消息中设置 DELETED 标志来完成。如果为 false，则改为设置 SEEN 标志。从 Camel 2.10 开始，您可以通过设置带有 delete 键的标头来覆盖这个配置选项，以确定是否应删除邮件。	false	布尔值
camel.component.mail.configuration.disconnect	消费者是否应该在轮询后断开连接。如果启用此选项，则会强制 Camel 每次轮询上进行连接。	false	布尔值

Name	描述	默认值	类型
camel.component.mail.configuration.dummy-trust-manager	使用 dummy 安全设置来信任所有证书。应该只用于开发模式，而不应用于生产环境。	false	布尔值
camel.component.mail.configuration.fetch-size	设置轮询期间要消耗的消息的最大数量。如果邮箱文件夹包含大量邮件，则这可用于避免过载邮件服务器。默认值 -1 表示没有获取大小，将消耗所有信息。将值设为 0 是一个特殊情况，Camel 将根本不使用任何消息。	-1	整数
camel.component.mail.configuration.folder-name	要轮询的文件夹。	INBOX	字符串
camel.component.mail.configuration.from	来自电子邮件地址	camel@localhost	字符串
camel.component.mail.configuration.handle-failed-message	如果邮件使用者无法检索给定邮件，则此选项允许处理消费者的错误处理程序导致的异常。通过在消费者上启用网桥错误处理程序，则 Camel 路由错误处理程序可以改为处理异常。默认行为是消费者抛出异常，来自批处理中的邮件将无法由 Camel 进行路由。	false	布尔值
camel.component.mail.configuration.host	邮件服务器主机名		字符串
camel.component.mail.configuration.ignore-unsupported-charset	选项，让 Camel 在发送邮件时忽略本地 JVM 中不支持的 charset。如果不支持 charset=XXX （其中 XXX 代表不受支持的 charset），则从 content-type 中删除 charset，它依赖于平台默认值。	false	布尔值
camel.component.mail.configuration.ignore-uri-scheme	选项，让 Camel 在发送邮件时忽略本地 JVM 中不支持的 charset。如果不支持 charset=XXX （其中 XXX 代表不受支持的 charset），则从 content-type 中删除 charset，它依赖于平台默认值。	false	布尔值
camel.component.mail.configuration.java-mail-properties	设置 java 邮件选项。将清除任何默认属性，并且仅使用为此方法提供的属性。		Properties
camel.component.mail.configuration.java-mail-sender	使用自定义 org.apache.camel.component.mail.JavaMailSender 来发送电子邮件。		JavaMailSender

Name	描述	默认值	类型
camel.component.mail.configuration.map-mail-message	指定 Camel 是否应该将接收的邮件映射到 Camel body/headers。如果设置为 true，邮件消息的正文将映射到 Camel IN 消息的正文，并且邮件标头映射到 IN 标头。如果此选项设为 false，则 IN 消息包含原始 javax.mail.Message。您可以通过调用 exchange.getIn（）.getBody (javax.mail.Message.class)来检索这个原始消息。	true	布尔值
camel.component.mail.configuration.mime-decode-headers	此选项为邮件标头启用透明 MIME 解码和折叠功能。	false	布尔值
camel.component.mail.configuration.password	登录的密码		字符串
camel.component.mail.configuration.peek	将 javax.mail.Message 标记为处理邮件之前。这只适用于 IMAPMessage 消息类型。通过使用在邮件服务器上标记为 SEEN，邮件不会强制标记为 SEEN，如果 Camel 出现错误处理，则我们可以回滚邮件消息。	true	布尔值
camel.component.mail.configuration.port	邮件服务器的端口号		整数
camel.component.mail.configuration.protocol	用于与邮件服务器通信的协议		字符串
camel.component.mail.configuration.reply-to	Reply-To 收件人（响应邮件的接收器）。使用逗号分隔多个电子邮件地址。		字符串
camel.component.mail.configuration.session	指定 camel 应该用于所有邮件交互的邮件会话。在由某些其他资源（如 modular 容器）创建和管理邮件会话时很有用。如果没有指定，Camel 会自动为您创建邮件会话。		会话
camel.component.mail.configuration.skip-failed-message	如果邮件使用者无法检索给定邮件，则此选项允许跳过邮件并继续检索下一个邮件。默认行为是消费者抛出异常，来自批处理中的邮件将无法由 Camel 进行路由。	false	布尔值
camel.component.mail.configuration.ssl-context-parameters	使用 SSLContextParameters 配置安全性。		SSLContextParameters

Name	描述	默认值	类型
camel.component.mail.configuration.subject	正在发送的消息主题。注意：在标头中设置主题优先于这个选项。		字符串
camel.component.mail.configuration.to	设置 To 电子邮件地址。使用逗号分隔多个电子邮件地址。		字符串
camel.component.mail.configuration.unseen	是否只被不可编辑的邮件限制。	true	布尔值
camel.component.mail.configuration.use-inline-attachments	是否使用 disposition inline 或 attachment。	false	布尔值
camel.component.mail.configuration.username	登录的用户名		字符串
camel.component.mail.content-type-resolver	解析器决定用于文件附加的 Content-Type。选项是 org.apache.camel.component.mail.ContentTypeResolver 类型。		字符串
camel.component.mail.enabled	启用邮件组件	true	布尔值
camel.component.mail.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.mail.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值
camel.dataformat.mime-multipart.binary-content	定义 MIME 多部分的内容是否为二进制(true)或 Base-64 encoded (false) Default 是否为 false。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.mime-multipart.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.mime-multipart.enabled	启用 mime-multipart dataformat	true	布尔值
camel.dataformat.mime-multipart.headers-inline	定义 MIME-Multipart 标头是消息正文(true)的一部分，或设置为 Camel 标头(false)。默认值为 false。	false	布尔值
camel.dataformat.mime-multipart.include-headers	定义将哪些 Camel 标头作为 MIME 标头包含在 MIME 多部分的正则表达式。这只有在 headersInline 设为 true 时才可以正常工作。默认为不包含标头		字符串
camel.dataformat.mime-multipart.multipart-sub-type	指定 MIME 多部分的子类型。默认为 mixed。	mixed	字符串
camel.dataformat.mime-multipart.multipart-without-attachment	定义没有附加的消息是否也会被放入 MIME Multipart 中（只有一个正文部分）。默认值为 false。	false	布尔值

218.4.1. 端点示例

通常，您可以指定带有登录凭证的 URI （将 SMTP 作为示例）：

```
smtp://[username@]host[:port][?password=somepwd]
```

或者，可以将用户名和密码指定为查询选项：

```
smtp://host[:port]?password=somepwd&username=someuser
```

例如：

```
smtp://mycompany.mailserver:30?password=tiger&username=scott
```

218.5. 组件

- *IMAP*
- *IMAPs*
- *POP3s*
- *POP3s*
- *SMTP*
- *SMTPs*

218.5.1. 默认端口

支持默认端口号。如果省略端口号，Camel 会根据协议确定要使用的端口号。

协议	默认端口号
SMTP	25
SMTPS	465
POP3	110
POP3S	995
IMAP	143
IMAPS	993

218.6. SSL 支持

底层邮件框架负责提供 SSL 支持。您可以通过完全指定所需的 Java Mail API 配置选项来配置 SSL/TLS 支持，或者您可以通过组件或端点配置提供配置的 SSLContextParameters。

218.6.1. 使用 JSSE 配置实用程序

从 Camel 2.10 开始，邮件组件通过 Camel JSSE 配置实用程序支持 SSL/TLS 配置。这个实用程序可大大减少您需要写入的组件特定代码量，并在端点和组件级别进行配置。以下示例演示了如何将 实用程序与 mail 组件一起使用。

端点的编程配置

```
KeyStoreParameters ksp = new KeyStoreParameters();
ksp.setResource("/users/home/server/truststore.jks");
ksp.setPassword("keystorePassword");
TrustManagersParameters tmp = new TrustManagersParameters();
tmp.setKeyStore(ksp);
SSLContextParameters scp = new SSLContextParameters();
scp.setTrustManagers(tmp);
Registry registry = ...
registry.bind("sslContextParameters", scp);
...
from(...)
&amp;nbsp; &amp;nbsp; .to("smtps://smtp.google.com?
username=user@gmail.com&password=password&sslContextParameters=#sslContextParam
eters");
```

基于 Spring DSL 的端点配置

```
...
<camel:sslContextParameters id="sslContextParameters">
  <camel:trustManagers>
    <camel:keyStore resource="/users/home/server/truststore.jks" password="keystorePassword"/>
  </camel:trustManagers>
</camel:sslContextParameters>...
...
<to uri="smtps://smtp.google.com?
username=user@gmail.com&password=password&sslContextParameters=#sslContextParameters"/
>...
```

218.6.2. 直接配置 JavaMail

Camel 使用 SUN JavaMail，它仅信任已知证书颁发机构（默认 JVM 信任配置）发布的证书。如果您发出自己的证书，您必须将 CA 证书导入到 JVM 的 Java 信任/密钥存储文件中，请覆盖默认的 JVM 信任/密钥存储文件（请参阅 JavaMail 中的 SSLNOTES.txt ）。

218.7. 邮件消息内容

Camel 使用消息交换的 IN 正文作为 MimeMessage 文本内容。正文转换为 String.class。

Camel 将所有交换的 IN 标头复制到 MimeMessage 标头。

MimeMessage 的主题可以使用 IN 消息上的 header 属性来配置。以下代码演示了这一点：

这同样适用于其他 MimeMessage 标头，如接收者，因此您可以使用作为 To 的标头属性：

由于 Camel 2.11 在使用 MailProducer 将邮件发送到服务器时，您应该可以使用 CamelMailMessageId 键获取 MimeMessage 的消息 ID。

218.8. 标头优先于预先配置的接收者

消息标头中指定的接收者始终优先于端点 URI 中预先配置的接收者。理念是，如果您在消息标头中提供任何接收者，这是您得到什么。端点 URI 中预先配置的接收者被视为回退。

在下面的示例代码中，电子邮件消息发送到 davsclaus@apache.org，因为它优先于预配置的接收者 info@mycompany.com。端点 URI 中的任何 CC 和 BCC 设置都会被忽略，这些接收者不会接收任何邮件。标头和预配置设置之间的选择是完全相互排除的：邮件组件需要完全从标头中接受发送者，或完全从预配置设置中接受发送者。不能混合和匹配标头和预配置设置。

```
Map<String, Object> headers = new HashMap<String, Object>();
headers.put("to", "davsclaus@apache.org");

template.sendBodyAndHeaders("smtp://admin@localhost?to=info@mycompany.com",
"Hello World", headers);
```

218.9. 多个接收者以便更轻松地配置

可以使用逗号分隔或分号分隔列表来设置多个接收者。这适用于标头设置，并应用到端点 URI 中的设置。例如：

```
Map<String, Object> headers = new HashMap<String, Object>();
headers.put("to", "davsclaus@apache.org ; jstrachan@apache.org ;
ningjiang@apache.org");
```

前面的示例使用分号 ; 作为分隔符字符。

218.10. 设置发件人名称和电子邮件

您可以使用名为 < email> 的 格式指定接收者，使其包含接收者的名称和电子邮件地址。

例如，您可以在 Message 中定义以下标头：

```
Map headers = new HashMap();
map.put("To", "Claus Ibsen <davsclaus@apache.org>");
map.put("From", "James Strachan <jstrachan@apache.org>");
map.put("Subject", "Camel is cool");
```

218.11. JAVAMAIL API (SUN JAVAMAIL)

javamail API 用于消耗和生成邮件。
我们鼓励最终用户在使用 POP3 或 IMAP 协议时参考这些参考。请注意，POP3 比 IMAP 有更多有限的功能集合。

- javamail POP3 API
- javamail IMAP API
- 通常有关 MAIL 标记

218.12. SAMPLES

我们从一个简单的路由开始，它将以电子邮件形式发送从 JMS 队列接收的消息。电子邮件帐户是 mymailserver.com 上的 admin 帐户。

```
from("jms://queue:subscription").to("smtp://admin@mymailserver.com?password=secret");
```

在下一个示例中，我们每分钟轮询一次新电子邮件的邮箱。请注意，我们使用特殊 消费者 选项将轮询间隔 consumer.delay 设为 60000 毫秒 = 60 秒。

```
from("imap://admin@mymailserver.com
    password=secret&unseen=true&consumer.delay=60000")
  .to("seda://mails");
```

在这个示例中，我们希望向多个接收者发送邮件：

218.13. 使用附加示例发送邮件

警告

附加功能不受所有 Camel 组件的支持。附件 API 基于 Java 激活框架，通常仅由 Mail API 使用。由于许多其它 Camel 组件不支持附件，因此附加可能会丢失，因为它们与路由传播。因此，thumb 的规则是在向邮件端点发送消息之前添加附件。

邮件组件支持附件。在以下示例中，我们发送一条包含带徽标文件附件的纯文本消息的邮件。

218.14. SSL 示例

在本例中，我们想要为邮件轮询 Google 邮件。要将邮件下载到本地邮件客户端上，Google 邮件要求您启用和配置 SSL。这可以通过登录到 Google 邮件帐户并更改您的设置以允许 IMAP 访问来完成。Google 包含有关如何执行此操作的广泛文档。

```
from("imaps://imap.gmail.com?
username=YOUR_USERNAME@gmail.com&password=YOUR_PASSWORD"
  + "&delete=false&unseen=true&consumer.delay=60000").to("log:newmail");
```

前面的路由会每分钟轮询一次新邮件的 Google 邮件，并将收到的消息记录到 newmail 日志记录器类别。
运行启用了 DEBUG 日志记录的示例，我们可以监控日志中的进度：

```
2008-05-08 06:32:09,640 DEBUG MailConsumer - Connecting to MailStore
imaps//imap.gmail.com:993 (SSL enabled), folder=INBOX
2008-05-08 06:32:11,203 DEBUG MailConsumer - Polling mailfolder:
imaps//imap.gmail.com:993 (SSL enabled), folder=INBOX
2008-05-08 06:32:11,640 DEBUG MailConsumer - Fetching 1 messages. Total 1 messages.
2008-05-08 06:32:12,171 DEBUG MailConsumer - Processing message: messageNumber=
```

```
[332], from=[James Bond <007@mi5.co.uk>], to=YOUR_USERNAME@gmail.com], subject=[...
2008-05-08 06:32:12,187 INFO  newmail - Exchange[MailMessage: messageNumber=[332],
from=[James Bond <007@mi5.co.uk>], to=YOUR_USERNAME@gmail.com], subject=[...
```

218.15. 使用附加示例消耗邮件示例

在本示例中，我们轮询邮箱，并将邮件中的所有附件存储为文件。首先，我们定义用于轮询邮箱的路由。因为这个示例基于 google 邮件，所以它使用与 SSL 示例中所示相同的路由：

```
from("imaps://imap.gmail.com?
username=YOUR_USERNAME@gmail.com&password=YOUR_PASSWORD"
    + "&delete=false&unseen=true&consumer.delay=60000").process(new MyMailProcessor());
```

我们不使用从 java 代码处理邮件的处理器，而不是记录邮件：

```
public void process(Exchange exchange) throws Exception {
    // the API is a bit clunky so we need to loop
    Map<String, DataHandler> attachments = exchange.getIn().getAttachments();
    if (attachments.size() > 0) {
        for (String name : attachments.keySet()) {
            DataHandler dh = attachments.get(name);
            // get the file name
            String filename = dh.getName();

            // get the content and convert it to byte[]
            byte[] data = exchange.getContext().getTypeConverter()
                            .convertTo(byte[].class, dh.getInputStream());

            // write the data to a file
            FileOutputStream out = new FileOutputStream(filename);
            out.write(data);
            out.flush();
            out.close();
        }
    }
}
```

正如您所见，用于处理附件的 API 是一种点冲突，但实际上，您可以获得 javax.activation.DataHandler，以便您可以使用标准 API 处理附件。

218.16. 如何使用附加分割邮件

在这个示例中，我们使用可能有多个附件的邮件。我们需要做的是每个附件使用 Splitter EIP，以单独处理附件。例如，如果邮件邮件有 5 个附件，我们希望 Splitter 处理五个消息，每个消息都有一个附件。要做到这一点，我们需要向 Splitter 提供自定义表达式，其中我们提供一个 List<Message>，其中包含带有单个附件的五个信息。

代码在 camel-mail 组件中的 Camel 2.10 以后提供。代码位于类中：org.apache.camel.component.mail.SplitAttachmentsExpression，您可以在其中找到源代码 https://svn.apache.org/repos/asf/camel/trunk/components/camel-mail/src/main/java/org/apache/camel/component/mail/SplitAttachmentsExpression.java

在 Camel 路由中，您需要在路由中使用此 Expression，如下所示：

如果使用 XML DSL，则需要在 Splitter 中声明一个方法调用表达式，如下所示

```
<split>
  <method beanType="org.apache.camel.component.mail.SplitAttachmentsExpression"/>
  <to uri="mock:split"/>
</split>
```

从 Camel 2.16 开始，您还可以将附件分割为 byte[]，以存储为消息正文。这可以通过创建带有布尔值 true 的表达式来完成。

```
SplitAttachmentsExpression split = SplitAttachmentsExpression(true);
```

然后，使用带有 splitter eip 的表达式。

218.17. 使用自定义 SEARCHTERM

从 Camel 2.11 开始提供

您可以在 MailEndpoint 上配置 searchTerm，它允许您过滤掉不需要的邮件。

例如，要过滤在 Subject 或 Text 中包含 Camel 的邮件，您可以执行以下操作：

```
<route>
  <from uri="imaps://mymailseerver?
username=foo&password=secret&searchTerm.subjectOrBody=Camel"/>
  <to uri="bean:myBean"/>
</route>
```

请注意，我们使用 "searchTerm.subjectOrBody" 作为参数键来指示我们希望搜索邮件主题或正文，来包含单词"Camel"。

类 org.apache.camel.component.mail.SimpleSearchTerm 具有您可以配置的多个选项：

或者为了获得新的不可预见的电子邮件，您可在您这样做时进行 24 小时。请注意"now-24h"语法。详情请查看下表。

```
<route>
  <from uri="imaps://mymailseerver?
username=foo&password=secret&searchTerm.fromSentDate=now-24h"/>
  <to uri="bean:myBean"/>
</route>
```

您可以在 endpoint uri 配置中有多个 searchTerm。然后，它们将使用 AND 运算符合并，例如这两个条件都必须匹配。例如，要得到最后不可见的电子邮件返回 24 小时，该小时在邮件主题中有 Camel，您可以执行以下操作：

```
<route>
  <from uri="imaps://mymailseerver?
username=foo&password=secret&searchTerm.subject=Camel&searchTerm.fromSentDate=now-
24h"/>
  <to uri="bean:myBean"/>
</route>
```

SimpleSearchTerm 旨在可从 POJO 轻松配置，因此您也可以使用 XML 中的 <bean> 样式进行配置。

```
<bean id="mySearchTerm" class="org.apache.camel.component.mail.SimpleSearchTerm">
  <property name="subject" value="Order"/>
  <property name="to" value="acme-order@acme.com"/>
  <property name="fromSentDate" value="now"/>
 </bean>
```

然后，您可以使用 Camel 路由中的 #beanId 来引用这个 bean，如下所示：

```
<route>
  <from uri="imaps://mymailseerver?
username=foo&password=secret&searchTerm=#mySearchTerm"/>
  <to uri="bean:myBean"/>
</route>
```

Java 中有一个构建器类，可使用 org.apache.camel.component.mail.SearchTermBuilder 类构建复合 SearchTerms。这可让您构建复杂的术语，例如：

```
// we just want the unseen mails which is not spam
```

```
SearchTermBuilder builder = new SearchTermBuilder();

builder.unseen().body(Op.not, "Spam").subject(Op.not, "Spam")
  // which was sent from either foo or bar
  .from("foo@somewhere.com").from(Op.or, "bar@somewhere.com");
  // .. and we could continue building the terms

SearchTerm term = builder.build();
```

218.18. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 219 章 MASTER 组件

从 Camel 版本 2.20 开始提供

camel-master: 端点提供了一种方式，可以确保集群中只有一个使用者从给定端点消耗；如果 JVM 分离，则自动故障转移。

如果您需要从一些不支持并发使用的后端使用，或者因为商业或稳定性原因而需要消耗一些旧的后端，则这非常有用。

219.1. 使用 MASTER 端点

只需为任何带有 master:someName: someName: 前缀的 camel 端点，其中 someName 是一个逻辑名称，用于获取 master 锁定。例如：

```
from("master:cheese:jms:foo").to("activemq:wine");
```

以上在 ActiveMQ 中模拟 [Exclusive Consumers](http://activemq.apache.org/exclusive-consumer.html)类型功能；但在任何可能不支持专用用户的第三方 JMS 提供程序上。

219.2. URI 格式

```
master:namespace:endpoint[?options]
```

其中 endpoint 是您要以 master/slave 模式运行的任何 Camel 端点。

219.3. 选项

Master 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
服务 （高级）	注入要使用的服务。		CamelClusterService
serviceSelector (advanced)	注入用于查找要使用的 CamelClusterService 的服务选择器。		选择器

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Master 端点使用 URI 语法进行配置：

```
master:namespace:delegateUri
```

使用以下路径和查询参数：

219.3.1. 路径参数(2 参数)：

Name	描述	默认值	类型
namespace	**必需** 使用的集群命名空间的名称		字符串
delegateUri	**必需的** 在 master/slave 模式中使用的端点 uri		字符串

219.3.2. 查询参数(4 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

219.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.master.enabled	是否启用 master 组件的自动配置。这默认是启用的。		布尔值
camel.component.master.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.master.service	注入要使用的服务。选项是 org.apache.camel.cluster.CamelClusterService 类型。		字符串
camel.component.master.service-selector	注入用于查找要使用的 CamelClusterService 的服务选择器。选项是 org.apache.camel.cluster.CamelClusterService.Selector 类型。		字符串

219.5. EXAMPLE

您可以保护集群的 Camel 应用程序，使其仅消耗来自一个活跃节点的文件。

```
// the file endpoint we want to consume from
String url = "file:target/inbox?delete=true";

// use the camel master component in the clustered group named myGroup
// to run a master/slave mode in the following Camel url
from("master:myGroup:" + url)
    .log(name + " - Received file: ${file:name}")
    .delay(delay)
    .log(name + " - Done file:     ${file:name}")
    .to("file:target/outbox");
```

master 组件利用了您可以使用以下方法配置的 CamelClusterService

- *Java*

```
ZooKeeperClusterService service = new ZooKeeperClusterService();
service.setId("camel-node-1");
```

```
service.setNodes("myzk:2181");
service.setBasePath("/camel/cluster");

context.addService(service)
```

- XML (Spring/Blueprint)

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://camel.apache.org/schema/spring
    http://camel.apache.org/schema/spring/camel-spring.xsd">


  <bean id="cluster"
class="org.apache.camel.component.zookeeper.cluster.ZooKeeperClusterService">
    <property name="id" value="camel-node-1"/>
    <property name="basePath" value="/camel/cluster"/>
    <property name="nodes" value="myzk:2181"/>
  </bean>

  <camelContext xmlns="http://camel.apache.org/schema/spring" autoStartup="false">
    ...
  </camelContext>

</beans>
```

- Spring boot

```
camel.component.zookeeper.cluster.service.enabled   = true
camel.component.zookeeper.cluster.service.id        = camel-node-1
camel.component.zookeeper.cluster.service.base-path = /camel/cluster
camel.component.zookeeper.cluster.service.nodes     = myzk:2181
```

219.6. 实现

Camel 提供以下 ClusterService 实现：

- camel-atomix
- camel-consul

- *camel-file*
- *camel-kubernetes*
- *camel-zookeeper*

219.7. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 220 章 指标组件

220.1. 指标组件

metrics: 组件允许直接从 Camel 路由收集各种指标。支持的指标类型是 计数器、直方图、计量、计时器 和 量表。指标 提供了测量应用程序的简单方法。可配置的报告后端是启用不同的集成选项来收集和视觉化统计信息。组件还提供 MetricsRoutePolicyFactory，它允许使用 Dropwizard Metrics 来公开路由统计信息，详情请参阅页面底部。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-metrics</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

220.2. URI 格式

```
metrics:[ meter | counter | histogram | timer | gauge ]:metricname[?options]
```

220.3. 选项

Metrics 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
MetricRegistry (advanced)	使用自定义配置的 MetricRegistry。		MetricRegistry
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Metrics 端点使用 URI 语法进行配置：

```
metrics:metricsType:metricsName
```

使用以下路径和查询参数：

220.3.1. 路径参数(2 参数)：

Name	描述	默认值	类型
metricsType	**所需的** 指标类型		MetricsType
metricsName	**所需** 指标名称		字符串

220.3.2. 查询参数(7 参数)：

Name	描述	默认值	类型
action (producer)	使用计时器类型时的操作		MetricsTimerAction
decrement (producer)	在使用计数器类型时减少值		Long
increment (producer)	使用计数器类型时递增值		Long
mark (producer)	使用量表类型时的 mark		Long
subject (producer)	使用量表类型时的主题值		对象
value (producer)	使用直方图类型时的值		Long
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

220.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.metrics.enabled	启用指标组件	true	布尔值

Name	描述	默认值	类型
camel.component.metrics.metric-registry	使用自定义配置的 MetricRegistry。选项是一个 com.codahale.metrics.MetricRegistry 类型。		字符串
camel.component.metrics.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

220.5. METRIC REGISTRY

Camel Metrics 组件默认使用一个 MetricRegistry 实例，它带有一个 Slf4jReporter，它有 60 秒报告间隔。通过提供 MetricRegistry bean，可以将此默认 registry 替换为一个自定义 registry。如果应用程序中存在多个 MetricRegistry Bean，则使用一个带有 name metricRegistry 的 MetricRegistry。

例如，使用 Spring Java 配置：

```
@Configuration
public static class MyConfig extends SingleRouteCamelConfiguration {

    @Bean
    @Override
    public RouteBuilder route() {
        return new RouteBuilder() {
            @Override
            public void configure() throws Exception {
                // define Camel routes here
            }
        };
    }

    @Bean(name = MetricsComponent.METRIC_REGISTRY_NAME)
    public MetricRegistry getMetricRegistry() {
        MetricRegistry registry = ...;
        return registry;
    }
}
```

或使用 CDI：

```
class MyBean extends RouteBuilder {
```

```
    @Override
    public void configure() {
      from("...")
          // Register the 'my-meter' meter in the MetricRegistry below
          .to("metrics:meter:my-meter");
    }

    @Produces
    // If multiple MetricRegistry beans
    // @Named(MetricsComponent.METRIC_REGISTRY_NAME)
    MetricRegistry registry() {
        MetricRegistry registry = new MetricRegistry();
        // ...
        return registry;
    }
}
```

220.6. 使用方法

每个指标具有类型和名称。支持的类型是 计数器、直方图、计量、计时器 和 量表。指标名称是简单字符串。如果没有提供指标类型，则默认使用类型 meter。

220.6.1. Headers

URI 中定义的指标名称可以通过使用带有名称 CamelMetricsName 的标头覆盖。

例如：

```
from("direct:in")
    .setHeader(MetricsConstants.HEADER_METRIC_NAME, constant("new.name"))
    .to("metrics:counter:name.not.used")
    .to("direct:out");
```

将使用名称 new.name 而不是 name.not.used 更新计数器。

当 Metrics 端点完成处理交换后，所有特定于 Metrics 的标头都会从消息中删除。在处理交换指标端点时，将使用级别 warn 捕获所有异常和写入日志条目。

220.7. METRICS TYPE COUNTER

```
metrics:counter:metricname[?options]
```

220.7.1. 选项

Name	default	描述
增量	-	要添加到计数器的长值
decrement	-	从计数器中减去的长值

如果没有定义 递增 或减少 值，则计数器值将以 1 递增。如果同时定义了 递增 和减少，则只调用递增操作。

```
// update counter simple.counter by 7
from("direct:in")
    .to("metric:counter:simple.counter?increment=7")
    .to("direct:out");
```

```
// increment counter simple.counter by 1
from("direct:in")
    .to("metric:counter:simple.counter")
    .to("direct:out");
```

```
// decrement counter simple.counter by 3
from("direct:in")
    .to("metrics:counter:simple.counter?decrement=3")
    .to("direct:out");
```

220.7.2. Headers

消息标头可用于覆盖 Metrics 组件 URI 中指定的 递增 和减少值。

Name	描述	预期类型
CamelMetricsCounterIncrement	覆盖 URI 中的递增值	Long
CamelMetricsCounterDecrement	覆盖 URI 中的减少值	Long

```
// update counter simple.counter by 417
from("direct:in")
    .setHeader(MetricsConstants.HEADER_COUNTER_INCREMENT, constant(417L))
    .to("metrics:counter:simple.counter?increment=7")
    .to("direct:out");
```

```
// updates counter using simple language to evaluate body.length
from("direct:in")
    .setHeader(MetricsConstants.HEADER_COUNTER_INCREMENT, simple("${body.length}"))
    .to("metrics:counter:body.length")
    .to("mock:out");
```

220.8. METRIC TYPE HISTOGRAM

```
metrics:histogram:metricname[?options]
```

220.8.1. 选项

Name	default	描述
value	-	在直方图中使用的值

如果没有设置任何 值，则会将任何值添加到直方图，并且会记录警告。

```
// adds value 9923 to simple.histogram
from("direct:in")
    .to("metric:histogram:simple.histogram?value=9923")
    .to("direct:out");
```

```
// nothing is added to simple.histogram; warning is logged
from("direct:in")
    .to("metric:histogram:simple.histogram")
    .to("direct:out");
```

220.8.2. Headers

消息标头可用于覆盖 Metrics 组件 URI 中指定的值。

Name	描述	预期类型
Camel Metrics HistogramValue	覆盖 URI 中的直方图值	Long

```
// adds value 992 to simple.histogram
from("direct:in")
    .setHeader(MetricsConstants.HEADER_HISTOGRAM_VALUE, constant(992L))
    .to("metrics:histogram:simple.histogram?value=700")
    .to("direct:out")
```

220.9. METRIC TYPE METER

```
metrics:meter:metricname[?options]
```

220.9.1. 选项

Name	default	描述
Mark	-	用作标记的长值

如果没有设置 mark，则不使用参数调用 meter.mark （）。

```
// marks simple.meter without value
from("direct:in")
    .to("metric:simple.meter")
    .to("direct:out");
```

```
// marks simple.meter with value 81
from("direct:in")
    .to("metric:meter:simple.meter?mark=81")
    .to("direct:out");
```

220.9.2. Headers

消息标头可用于覆盖 Metrics 组件 URI 中指定的 标记 值。

Name	描述	预期类型
Camel Metrics Meter Mark	覆盖 URI 中的标记值	Long

```
// updates meter simple.meter with value 345
from("direct:in")
    .setHeader(MetricsConstants.HEADER_METER_MARK, constant(345L))
    .to("metrics:meter:simple.meter?mark=123")
    .to("direct:out");
```

220.10. METRICS 类型 TIMER

```
metrics:timer:metricname[?options]
```

220.10.1. 选项

Name	default	描述
action	-	启动或停止

如果没有提供 操作 或无效值，则在没有计时器更新的情况下会记录警告。如果在已在运行的计时器 或停止 上调用操作 start，则不会运行计时器，则不会更新并记录警告。

```
// measure time taken by route "calculate"
from("direct:in")
    .to("metrics:timer:simple.timer?action=start")
    .to("direct:calculate")
    .to("metrics:timer:simple.timer?action=stop");
```

TimerContext 对象存储为不同 Metrics 组件调用之间的 Exchange 属性。

220.10.2. Headers

消息标头可用于覆盖 Metrics 组件 URI 中指定的操作值。

Name	描述	预期类型
Camel Metrics TimerAction	覆盖 URI 中的计时器操作	**org.apache.camel.component.metrics.timer.TimerEndpoint.TimerAction**

```
// sets timer action using header
from("direct:in")
    .setHeader(MetricsConstants.HEADER_TIMER_ACTION, TimerAction.start)
    .to("metrics:timer:simple.timer")
    .to("direct:out");
```

220.11. METRIC TYPE GAUGE

```
metrics:gauge:metricname[?options]
```

220.11.1. 选项

Name	default	描述
subject	-	量表观察到的任何对象

如果没有定义 主题，则直接忽略它，即不会注册量表。

```
// update gauge "simple.gauge" by a bean "mySubjectBean"
from("direct:in")
    .to("metrics:gauge:simple.gauge?subject=#mySubjectBean")
    .to("direct:out");
```

220.11.2. Headers

消息标头可用于覆盖 Metrics 组件 URI 中指定的 主题 值。注： 如果指定了 CamelMetricsName 标头，除了 URI 中指定的默认量外，还会注册新的量表。

Name	描述	预期类型
Camel Metrics Gauge Subject	覆盖 URI 中的主题值	对象

```
// update gauge simple.gauge by a String literal "myUpdatedSubject"
from("direct:in")
    .setHeader(MetricsConstants.HEADER_GAUGE_SUBJECT, constant("myUpdatedSubject"))
    .to("metrics:counter:simple.gauge?subject=#mySubjectBean")
    .to("direct:out");
```

220.12. METRICSROUTEPOLICYFACTORY

此工厂允许为每个路由添加一个 RoutePolicy，该路由使用 Dropwizard 指标公开路由利用率统计。此工厂可在 Java 和 XML 中使用，如下例所示。

注意

如果您只想检测几个所选路由，而不是使用 MetricsRoutePolicyFactory，而是定义您要检测的每个路由。

从 Java 中，您要将工厂添加到 CamelContext 中，如下所示：

```
context.addRoutePolicyFactory(new MetricsRoutePolicyFactory());
```

从 XML DSL 中，您定义一个 <bean>，如下所示：

```
<!-- use camel-metrics route policy to gather metrics for all routes -->
<bean id="metricsRoutePolicyFactory"
class="org.apache.camel.component.metrics.routepolicy.MetricsRoutePolicyFactory"/>
```

MetricsRoutePolicyFactory 和 MetricsRoutePolicy 支持以下选项：

Name	default	描述

Name	default	描述
useJmx	false	使用 **com.codahale.metrics.JmxReporter** 将精细的统计报告给 JMX。 请注意，如果在 CamelContext 上启用了 JMX，则将在 JMX 树中的服务类型下放入 **MetricsRegistryService** mbean。该 mbean 有一个操作来使用 json 输出统计信息。 只有在您希望每个统计类型的精细 mbeans 时才需要将 **useJmx** 设置为 true。
jmxDomain	org.apache.camel.metrics	JMX 域名
prettyPrint	false	在以 json 格式输出统计信息时，是否使用用户用户打印
metricsRegistry		允许使用共享 **com.codahale.metrics.MetricRegistry**。如果没有提供，则 Camel 将创建一个由此 CamelContext 使用的共享实例。
rateUnit	timeunit.SECONDS	指标报告器或以 json 转储统计时使用的单位。
durationUnit	TimeUnit.MILLISECONDS	指标报告器中使用的持续时间单位，或者在将统计信息转储为 json 时。
namePattern	**name.routeId.type**	Camel 2.17： 要使用的名称模式。使用点作为分隔符，但您可以更改。名为 **, routeId**, 和 **type** 的值将被实际值替换。其中 **name** 是 CamelContext 的名称。**routeId** 是路由的名称。和 **type** 是响应的值。

从 Java 代码中，您可以存放来自 org.apache.camel.component.metrics.routepolicy.MetricsRegistry 的 com.codahale.metrics.MetricRegistry Service，如下所示：

```
MetricRegistryService registryService = context.hasService(MetricsRegistryService.class);
if (registryService != null) {
  MetricsRegistry registry = registryService.getMetricsRegistry();
  ...
}
```

220.13. METRICSMESSAGEHISTORYFACTORY

从 Camel 2.17 开始提供

此工厂允许使用指标在路由消息时捕获消息历史性能统计信息。它的工作原理是为所有路由中的每个节点使用指标计时器。此工厂可在 Java 和 XML 中使用，如下例所示。

从 Java 中，您刚刚将工厂设置为 CamelContext，如下所示：

```
context.setMessageHistoryFactory(new MetricsMessageHistoryFactory());
```

从 XML DSL 中，您定义一个 <bean>，如下所示：

```
<!-- use camel-metrics message history to gather metrics for all messages being routed -->
<bean id="metricsMessageHistoryFactory"
class="org.apache.camel.component.metrics.messagehistory.MetricsMessageHistoryFactory"/>
```

工厂支持以下选项：

Name	default	描述
useJmx	false	使用 **com.codahale.metrics.JmxReporter** 将精细的统计报告给 JMX。 请注意，如果在 CamelContext 上启用了 JMX，则将在 JMX 树中的服务类型下放入 **MetricsRegistryService** mbean。该 mbean 有一个操作来使用 json 输出统计信息。 只有在您希望每个统计类型的精细 mbeans 时才需要将 **useJmx** 设置为 true。
jmxDomain	org.apache.camel.metrics	JMX 域名
prettyPrint	false	在以 json 格式输出统计信息时，是否使用用户用户打印
metricsRegistry		允许使用共享 **com.codahale.metrics.MetricRegistry**。如果没有提供，则 Camel 将创建一个由此 CamelContext 使用的共享实例。
rateUnit	timeunit.SECONDS	指标报告器或以 json 转储统计时使用的单位。
durationUnit	TimeUnit.MILLISECONDS	指标报告器中使用的持续时间单位，或者在将统计信息转储为 json 时。

Name	default	描述
namePattern	**name.routeId.id.type**	要使用的名称模式。使用点作为分隔符，但您可以更改。值 **名称**, **routeId**, **type**, 和 **id** 将替换为实际值。其中 **name** 是 CamelContext 的名称。**routeId** 是路由的名称。**id** 模式表示节点 id。和 **type** 是历史记录的值。

在运行时，可以从 Java API 或 JMX 访问指标，允许将数据作为 json 输出收集数据。

在 Java 代码中，您可以从 CamelContext 获取服务，如下所示：

```
MetricsMessageHistoryService service =
context.hasService(MetricsMessageHistoryService.class);
String json = service.dumpStatisticsAsJson();
```

此外，MBean API 使用 name=MetricsMessageHistoryService 在 type=services 树中注册。

220.14. INSTRUMENTEDTHREADPOOLFACTORY

从 Camel 2.18 开始提供

此工厂允许您通过注入一个从 Camel 内部收集信息的 InstrumentedThreadPoolFactory 来收集有关 Camel 线程池的性能信息。请参阅 使用 Spring 的 CamelContext 高级配置中更多详情

220.15. 另请参阅

- *camel-example-cdi-metrics 示例，它演示了 Camel、Metrics 和 CDI 之间的集成。*

第 221 章 MICROMETER 组件

221.1. MICROMETER 组件

micrometer: 组件允许直接从 Camel 路由收集各种指标。支持的指标类型是 计数器、摘要和 计时器。Micrometer 提供简单的方法来测量应用程序的行为。可配置的报告后端（通过 Micrometer registry）启用了不同的集成选项来收集和视觉化统计信息。

组件还提供 MicrometerRoutePolicyFactory，它允许使用 Micrometer 公开路由统计信息，以及计算路由和时间交换从创建到其完成的时间交换。

Maven 用户需要在此组件的 pom.xml 中添加以下依赖项：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-micrometer</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

221.2. URI 格式

```
micrometer:[ counter | summary | timer ]:metricname[?options]
```

221.3. 选项

Micrometer 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
metricsRegistry (advanced)	使用自定义配置的 MetricRegistry。		MeterRegistry
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Micrometer 端点使用 URI 语法进行配置：

micrometer:metricsType:metricsName

使用以下路径和查询参数：

221.3.1. 路径参数(3 参数)：

Name	描述	默认值	类型
metricsType	**所需的** 指标类型		类型
metricsName	**所需** 指标名称		字符串
tags	指标标签		iterable

221.3.2. 查询参数(5 参数)：

Name	描述	默认值	类型
action (producer)	使用计时器类型时的操作表达式		字符串
decrement (producer)	在使用计数器类型时取消值表达式		字符串
increment (producer)	使用计数器类型时递增值表达式		字符串
value (producer)	使用直方类型时的值表达式		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

221.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.micrometer.enabled	是否启用微主题组件的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.component.micrometer.metrics-registry	使用自定义配置的 MetricRegistry。选项是一个 io.micrometer.core.instrument.MeterRegistry 类型。		字符串
camel.component.micrometer.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

221.5. METER REGISTRY

默认情况下，Camel Micrometer 组件会创建一个 SimpleMeterRegistry 实例，主要适合测试。您应该通过提供 MeterRegistry bean 来定义专用 registry。Micrometer registry 主要决定要使用的后端监控系统。CompositeMeterRegistry 可用于处理多个监控目标。

例如，使用 Spring Java 配置：

```
@Configuration
public static class MyConfig extends SingleRouteCamelConfiguration {

    @Bean
    @Override
    public RouteBuilder route() {
        return new RouteBuilder() {
            @Override
            public void configure() throws Exception {
                // define Camel routes here
            }
        };
    }

    @Bean(name = MicrometerComponent.METRICS_REGISTRY_NAME)
    public MeterRegistry getMeterRegistry() {
        CompositeMeterRegistry registry = ...;
        registry.add(...);
        // ...
        return registry;
    }
}
```

或使用 CDI：

```
class MyBean extends RouteBuilder {

    @Override
    public void configure() {
      from("...")
          // Register the 'my-meter' meter in the MetricRegistry below
          .to("metrics:meter:my-meter");
    }

    @Produces
    // If multiple MetricRegistry beans
    // @Named(MicrometerComponent.METRIC_REGISTRY_NAME)
    MetricRegistry registry() {
        CompositeMeterRegistry registry = ...;
        registry.add(...);
        // ...
        return registry;
    }
}
```

221.6. 使用制作者

每个计量具有类型和名称。支持的类型有 计数器、分发摘要和 计时器。如果没有提供类型，则默认使用计数器。

计量名称是一个字符串，被评估为 Simple 表达式。除了使用 CamelMetricsName 标头（请参阅以下），这还允许根据交换数据选择量表。

可选的 tags URI 参数是一个用逗号分开的字符串，由 key=value 表达式组成。 键和值 都是同时被评估为 简单表达式的字符串。例如， URI 参数 tags=X=${header.Y} 会将标头 Y 的当前值分配给键 X。

221.6.1. Headers

URI 中定义的计量名称可以通过填充名为 CamelMetricsName 的标头来覆盖。定义为 URI 参数的计量标签可以通过填充名为 CamelMetricsTags 的标头来增强。

例如：

```
from("direct:in")
    .setHeader(MicrometerConstants.HEADER_METRIC_NAME, constant("new.name"))
    .setHeader(MicrometerConstants.HEADER_METRIC_TAGS, constant(Tags.of("dynamic-key", "dynamic-value")))
    .to("metrics:counter:name.not.used?tags=key=value")
    .to("direct:out");
```

除了值为 的标签键外，还将使用名称 new.name 而不是 name.not.used 更新计数器，并使用标签 dynamic- key 和值 dynamic- value 。

当 Micrometer 端点完成处理交换后，所有特定于 Metrics 的标头都会从消息中删除。在处理交换 Micrometer 端点时，会用级别 warn 捕获所有异常和写入日志条目。

221.7. COUNTER

```
micrometer:counter:name[?options]
```

221.7.1. 选项

Name	default	描述
增量	-	要添加到计数器的双值
decrement	-	从计数器中减去加倍的值

如果没有定义 递增 或减少 值，则计数器值将以 1 递增。如果同时定义了 递增 和减少，则只调用递增操作。

```
// update counter simple.counter by 7
from("direct:in")
    .to("micrometer:counter:simple.counter?increment=7")
    .to("direct:out");
```

```
// increment counter simple.counter by 1
from("direct:in")
    .to("micrometer:counter:simple.counter")
    .to("direct:out");
```

increment 和 decrement 值都被评估为带有 Double 结果的 Simple 表达式，例如，如果标头 X 包含评估为 3.0 的值，则 simple.counter 计数器由 3.0 减少：

```
// decrement counter simple.counter by 3
from("direct:in")
    .to("micrometer:counter:simple.counter?decrement=${header.X}")
    .to("direct:out");
```

221.7.2. Headers

与 camel-metrics 中一样，特定的 Message 标头可用于覆盖 Micrometer 端点 URI 中指定的 递增 和减少值。

Name	描述	预期类型
Camel Metrics CounterIncrement	覆盖 URI 中的递增值	å�Œ
Camel Metrics CounterDecrement	覆盖 URI 中的减少值	å�Œ

```
// update counter simple.counter by 417
from("direct:in")
    .setHeader(MicrometerConstants.HEADER_COUNTER_INCREMENT, constant(417.0D))
    .to("micrometer:counter:simple.counter?increment=7")
    .to("direct:out");
```

```
// updates counter using simple language to evaluate body.length
from("direct:in")
    .setHeader(MicrometerConstants.HEADER_COUNTER_INCREMENT,
simple("${body.length}"))
    .to("micrometer:counter:body.length")
    .to("direct:out");
```

221.8. DISTRIBUTION SUMMARY

```
micrometer:summary:metricname[?options]
```

221.8.1. 选项

Name	default	描述
value	-	在直方图中使用的值

如果没有设置 值，则不会将任何内容添加到直方图，并且会记录警告。

```
// adds value 9923 to simple.histogram
```

```
from("direct:in")
    .to("micrometer:summary:simple.histogram?value=9923")
    .to("direct:out");
```

```
// nothing is added to simple.histogram; warning is logged
from("direct:in")
    .to("micrometer:summary:simple.histogram")
    .to("direct:out");
```

值 被评估为带有 Double 结果的 Simple 表达式，例如，如果标头 X 包含评估为 3.0 的值，则该值会使用 simple.histogram 注册：

```
from("direct:in")
    .to("micrometer:summary:simple.histogram?value=${header.X}")
    .to("direct:out");
```

221.8.2. Headers

与 camel-metrics 中一样，可以使用特定的 Message 标头覆盖 Micrometer 端点 URI 中指定的值。

Name	描述	预期类型
CamelMetricsHistogramValue	覆盖 URI 中的直方图值	Long

```
// adds value 992.0 to simple.histogram
from("direct:in")
    .setHeader(MicrometerConstants.HEADER_HISTOGRAM_VALUE, constant(992.0D))
    .to("micrometer:summary:simple.histogram?value=700")
    .to("direct:out")
```

221.9. TIMER

```
micrometer:timer:metricname[?options]
```

221.9.1. 选项

Name	default	描述
action	-	启动或停止

如果没有提供 操作 或无效值，则在没有计时器更新的情况下会记录警告。如果在已在运行的计时器 或停止 中调用操作 start，在未知计时器中调用操作，则不会更新任何内容并记录警告。

```
// measure time spent in route "direct:calculate"
from("direct:in")
    .to("micrometer:timer:simple.timer?action=start")
    .to("direct:calculate")
    .to("micrometer:timer:simple.timer?action=stop");
```

timer.Sample 对象存储为不同 Metrics 组件调用之间的 Exchange 属性。

操作 被评估为 简单 表达式返回类型为 MicrometerTimerAction 的结果。

221.9.2. Headers

与 camel-metrics 中相似，特定的 Message 标头可用于覆盖 Micrometer 端点 URI 中指定的操作值。

Name	描述	预期类型
Camel Metrics TimerAction	覆盖 URI 中的计时器操作	org.apache.camel.component.micrometer.MicrometerTimerAction

```
// sets timer action using header
from("direct:in")
    .setHeader(MicrometerConstants.HEADER_TIMER_ACTION, MicrometerTimerAction.start)
    .to("micrometer:timer:simple.timer")
    .to("direct:out");
```

221.10. MICROMETERROUTEPOLICYFACTORY

此工厂允许为每个路由添加 RoutePolicy，以便使用 Micrometer 公开路由利用率统计。此工厂可在

Java 和 XML 中使用，如下例所示。

注意

如果您只想检测几个所选路由，而不是使用 MicrometerRoutePolicy Factory，而是定义一个专用 MicrometerRoutePolicy。

从 Java 中，您要将工厂添加到 CamelContext 中，如下所示：

```
context.addRoutePolicyFactory(new MicrometerRoutePolicyFactory());
```

从 XML DSL 中，您定义一个 <bean>，如下所示：

```
<!-- use camel-micrometer route policy to gather metrics for all routes -->
<bean id="metricsRoutePolicyFactory"
class="org.apache.camel.component.micrometer.routepolicy.MicrometerRoutePolicyFactory"/>
```

MicrometerRoutePolicyFactory 和 MicrometerRoutePolicy 支持以下选项：

Name	default	描述
prettyPrint	false	在以 json 格式输出统计信息时，是否使用用户用户打印
meterRegistry		允许使用共享 **MeterRegistry**。如果没有提供，则 Camel 将创建一个由此 CamelContext 使用的共享实例。
durationUnit	TimeUnit.MILLISECONDS	当将统计信息转储为 json 时，用于持续时间的单位。

如果在 CamelContext 中启用了 JMX，则 MBean 会在带有 name=MicrometerRoutePolicy 的 type=services 树中注册。

221.11. MICROMETERMESSAGEHISTORYFACTORY

此工厂允许使用指标在路由消息时捕获消息历史性能统计信息。它的工作原理为所有路由中的每个节点使用 Micrometer Timer。此工厂可在 Java 和 XML 中使用，如下例所示。

从 Java 中，您刚刚将工厂设置为 CamelContext，如下所示：

```
context.setMessageHistoryFactory(new MicrometerMessageHistoryFactory());
```

从 XML DSL 中，您定义一个 <bean>，如下所示：

```
<!-- use camel-micrometer message history to gather metrics for all messages being routed -->
<bean id="metricsMessageHistoryFactory" class="org.apache.camel.component.micrometer.messagehistory.MicrometerMessageHistoryFactory" />
```

工厂支持以下选项：

Name	default	描述
prettyPrint	false	在以 json 格式输出统计信息时，是否使用用户用户打印
meterRegistry		允许使用共享 **MeterRegistry**。如果没有提供，则 Camel 将创建一个由此 CamelContext 使用的共享实例。
durationUnit	TimeUnit.MILLISECONDS	将统计数据转储为 json 时用于持续时间的单位。

在运行时，可以从 Java API 或 JMX 访问指标，允许将数据作为 json 输出收集数据。

从 Java 代码中，您可以从 CamelContext 获取服务，如下所示：

```
MicrometerMessageHistoryService service = context.hasService(MicrometerMessageHistoryService.class);
String json = service.dumpStatisticsAsJson();
```

如果在 CamelContext 中启用了 JMX，则 MBean 会在 type=services 树中注册，并带有 name=MicrometerMessageHistory。

221.12. MICROMETEREVENTNOTIFIERS

有一个 MicrometerRouteEventNotifier （计算添加和运行路由）和一个 MicrometerExchangeEventNotifier （从创建交换到其完成）。

EventNotifiers 可以添加到 CamelContext 中，例如：

```
camelContext.getManagementStrategy().addEventNotifier(new
MicrometerExchangeEventNotifier())
```

在运行时，可以从 Java API 或 JMX 访问指标，允许将数据作为 json 输出收集数据。

在 Java 代码中，您可以从 CamelContext 获取服务，如下所示：

```
MicrometerEventNotifierService service =
context.hasService(MicrometerEventNotifierService.class);
String json = service.dumpStatisticsAsJson();
```

如果在 CamelContext 中启用了 JMX，则 MBean 会在 type=services 树中使用 name=MicrometerEventNotifier 注册。

221.13. INSTRUMENTEDTHREADPOOLFACTORY

此工厂允许您通过注入一个从 Camel 内部收集信息的 InstrumentedThreadPoolFactory 来收集有关 Camel 线程池的性能信息。请参阅 使用 Spring 的 CamelContext 高级配置中更多详情。

221.14. 在 JMX 中公开 MICROMETER 统计信息

Micrometer 使用 MeterRegistry 实现来发布统计信息。虽然在生产环境中建议选择专用的后端，如 Prometheus 或 Graphite，但可能需要测试或本地部署来将统计信息发布到 JMX。

要达到此目的，请添加以下依赖项：

```
<dependency>
  <groupId>io.micrometer</groupId>
  <artifactId>micrometer-registry-jmx</artifactId>
```

```
    <version>${micrometer-version}</version>
  </dependency>
```

并添加 JmxMeterRegistry 实例：

```
@Bean(name = MicrometerComponent.METRICS_REGISTRY_NAME)
public MeterRegistry getMeterRegistry() {
    CompositeMeterRegistry meterRegistry = new CompositeMeterRegistry();
    meterRegistry.add(...);
    meterRegistry.add(new JmxMeterRegistry(
      CamelJmxConfig.DEFAULT,
      Clock.SYSTEM,
      HierarchicalNameMapper.DEFAULT));
    return meterRegistry;
  }
}
```

HierarchicalNameMapper 策略决定如何将量表名称和标签组合成 MBean 名称。

221.15. EXAMPLE

camel-example-micrometer 提供一个使用 Java 配置和 Prometheus 后端通过 Camel 设置 Micrometer 监控的示例。

第 222 章 OPC UA 客户端组件

从 Camel 版本 2.19 开始提供

Milo Client 组件利用 Eclipse Milo™ 实施，提供对 OPC UA 服务器的访问。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-milo</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

OPC UA 客户端组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
defaultConfiguration (common)	客户端的所有默认选项		MiloClientConfiguration
applicationName (common)	默认应用程序名称		字符串
applicationUri (common)	默认应用程序 URI		字符串
productUri (common)	默认产品 URI		字符串
reconnectTimeout (common)	默认重新连接超时		Long
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

222.1. URI 格式

端点的 URI 语法为：

```
milo-client:tcp://[user:password@]host:port/path/to/service?node=RAW(nsu=urn:foo:bar;s=item-1)
```

如果服务器没有使用路径，则可以直接省略它：

```
milo-client:tcp://[user:password@]host:port?node=RAW(nsu=urn:foo:bar;s=item-1)
```

如果没有提供用户凭证，客户端将切换到匿名模式。

222.2. URI 选项

组客户端中的所有配置选项都适用于共享客户端实例。端点将共享每个端点 URI 的客户端实例。因此，第一次发出对该端点 URI 的请求时，将应用客户端组的选项。其他所有实例都将被忽略。

如果您需要同一端点 URI 的替代选项，可以设置 clientId 选项，该选项可以在内部添加到端点 URI 中，以便选择不同的共享连接实例。换句话说，由端点 URI 和客户端 ID 组合共享连接。

OPC UA 客户端端点使用 URI 语法进行配置：

```
milo-client:endpointUri
```

使用以下路径和查询参数：

222.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
endpointUri	**所需的** OPC UA 服务器端点		字符串

222.2.2. 查询参数(27 参数)：

Name	描述	默认值	类型
clientId (common)	用于强制创建新连接实例的虚拟客户端 ID		字符串
defaultAwaitWrites (common)	写入的默认 await 设置	false	布尔值

Name	描述	默认值	类型
discoveryEndpointSuffix (common)	发现时端点 URI 的后缀		字符串
discoveryEndpointUri (common)	备用发现 URI		字符串
方法 (common)	方法定义（请参阅方法 ID）		ExpandedNodeId
node (common)	节点定义（请参阅节点 ID）		ExpandedNodeId
samplingInterval (common)	以毫秒为单位的抽样间隔		å€Œ
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
allowedSecurityPolicies (client)	组允许的安全策略 URI。默认为接受全部并使用最高。		字符串
applicationName (client)	应用程序名称	Eclipse Milo 的 Apache Camel 适配器	字符串

Name	描述	默认值	类型
applicationUri (client)	应用程序 URI	http://camel.apache.org/EclipseMilo/Client	字符串
ChannelLifetime (client)	频道生命周期（以毫秒为单位）		Long
key alias (client)	密钥存储文件中的密钥名称		字符串
keyPassword (client)	密钥密码		字符串
keyStorePassword (client)	密钥存储密码		字符串
keyStoreType (client)	密钥存储类型		字符串
keyStoreUrl (client)	应该从中加载密钥的 URL		URL
maxPendingPublishRequests (client)	等待发布请求的最大数量		Long
maxResponseMessageSize (client)	响应消息可能具有的最大字节数		Long
overrideHost (client)	使用来自端点 URI 的主机覆盖服务器报告的端点主机。	false	布尔值
productUri (client)	产品 URI	http://camel.apache.org/EclipseMilo	字符串
requestTimeout (client)	以毫秒为单位请求超时		Long
sessionName (client)	会话名称		字符串

Name	描述	默认值	类型
SessionTimeout (client)	会话超时（以毫秒为单位）		Long

222.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 24 个选项，如下所列。

Name	描述	默认值	类型
camel.component.milo-client.application-name	默认应用程序名称		字符串
camel.component.milo-client.application-uri	默认应用程序 URI		字符串
camel.component.milo-client.default-configuration.allowed-security-policies	组允许的安全策略 URI。默认为接受全部并使用最高。		Set
camel.component.milo-client.default-configuration.application-name	应用程序名称	Eclipse Milo 的 Apache Camel 适配器	字符串
camel.component.milo-client.default-configuration.application-uri	应用程序 URI	http://camel.apache.org/EclipseMilo/Client	字符串
camel.component.milo-client.default-configuration.channel-lifetime	频道生命周期（以毫秒为单位）		Long

Name	描述	默认值	类型
camel.component.milo-client.default-configuration.client-id	用于强制创建新连接实例的虚拟客户端 ID		字符串
camel.component.milo-client.default-configuration.discovery-endpoint-suffix	发现时端点 URI 的后缀		字符串
camel.component.milo-client.default-configuration.discovery-endpoint-uri	备用发现 URI		字符串
camel.component.milo-client.default-configuration.key-alias	密钥存储文件中的密钥名称		字符串
camel.component.milo-client.default-configuration.key-password	密钥密码		字符串
camel.component.milo-client.default-configuration.key-store-password	密钥存储密码		字符串
camel.component.milo-client.default-configuration.key-store-type	密钥存储类型		字符串
camel.component.milo-client.default-configuration.max-pending-publish-requests	等待发布请求的最大数量		Long

Name	描述	默认值	类型
camel.component.milo-client.default-configuration.max-response-message-size	响应消息可能具有的最大字节数		Long
camel.component.milo-client.default-configuration.override-host	使用来自端点 URI 的主机覆盖服务器报告的端点主机。	false	布尔值
camel.component.milo-client.default-configuration.product-uri	产品 URI	http://camel.apache.org/EclipseMilo	字符串
camel.component.milo-client.default-configuration.request-timeout	以毫秒为单位请求超时		Long
camel.component.milo-client.default-configuration.session-name	会话名称		字符串
camel.component.milo-client.default-configuration.session-timeout	会话超时（以毫秒为单位）		Long
camel.component.milo-client.enabled	启用 milo-client 组件	true	布尔值
camel.component.milo-client.product-uri	默认产品 URI		字符串
camel.component.milo-client.reconnect-timeout	默认重新连接超时		Long

Name	描述	默认值	类型
camel.component.milo-client.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

222.3.1. Discovery（发现）

如果服务器使用专用的发现端点（如 /discovery），它可以支持不同的（无安全）安全策略，那么您可以通过参数 discoveryEndpointSuffix 使用此选项，该端点将附加到 endpointUri 中。或通过使用显式的 discoveryEndpointUri。

222.3.2. 覆盖主机名

客户端使用来自端点信息的主机信息，从服务器查询。然而，在某些情况下，这个端点 URI 可能有所不同，并从连接的客户端的角度来看（如内部主机名）错误。

在这种情况下，可以将参数 overrideHost 设置为 true，这将获取发现的端点信息，但会使用原始 URI 的值覆盖主机信息。

222.3.3. 节点 ID

要定义目标节点，需要一个命名空间和节点 ID。在以前的版本中，这可以通过指定 nodeId 和 namespaceUri 或 namespaceIndex 来实现。但是，这只允许使用基于字符串的节点 ID。虽然这种配置仍然有可能，但较新的配置是首选的。

新方法是以 ns=1;i=1 格式指定完整的命名空间+节点 ID，它也允许使用其他节点 ID 格式（如数字、GUID/UUID 或 opaque）。如果使用了 node 参数，则不得使用旧的参数。此节点格式的语法是 一组键值对，用分号 (;)分隔。

必须使用一个命名空间和一个节点 id 键。有关可能的密钥，请查看下表：

键	类型	描述
ns	namespace	数字命名空间索引
nsu	namespace	命名空间 URI
s	node	字符串节点 ID
i	node	数字节点 ID
g	node	GUID/UUID 节点 ID
b	node	用于不透明节点 ID 的 Base64 编码字符串

因为语法生成的值不能透明地编码为 URI 参数值，因此需要转义它们。但是， Camel 允许在 RAW (...) 内嵌套实际值，这会导致不必要的转义。例如：

```
milo-client:tcp://user:password@localhost:12345?node=RAW(nsu=http://foo.bar;s=foo/bar)
```

222.3.4. 方法 ID

可以在 OPC UA 节点上执行方法调用。如果将参数 方法 设置为方法调用的节点 ID （在这种情况下，节点 ID 必须设为父对象），则将执行方法调用，而不是写入操作。

输入参数从正文获取：

- *如果正文为空，则使用空的 Variant[]*
- *如果正文是 变体[]，它将按原样使用*
- *如果正文是 变体，则它将嵌套在 Variant[] 数组中*
- *否则，正文将转换为 变体，并嵌套在 Variant[]的数组中*

222.3.5. 安全策略

当设置允许安全策略时，可以使用已知的 OPC UA URI （如 http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15）或使用 Milo enum literals （如 None）。指定未知安全策略 URI 或 enum 是一个错误。

此处可以看到已知的安全策略 URI 和 enum 文字： SecurityPolicy.java

注意： 在任何情况下，安全策略被视为区分大小写。

222.4. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 223 章 OPC UA 服务器组件

从 Camel 版本 2.19 开始提供

Milo Server 组件使用 Eclipse Milo™ 实施提供 OPC UA 服务器。

Java 8 ：此组件在运行时需要 Java 8。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-milo</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

从 Camel 发送到端点的消息将从 OPC UA 服务器到 OPC UA 客户端提供。OPC UA 客户端写入请求将触发发送到 Apache Camel 的消息。

OPC UA 服务器组件支持 20 个选项，如下所列。

Name	描述	默认值	类型
namespaceUri (common)	命名空间的 URI，默认为 urn:org:apache:camel		字符串
applicationName (common)	应用程序名称		字符串
applicationUri (common)	应用程序 URI		字符串
productUri (common)	产品 URI		字符串
bindPort (common)	服务器绑定到的 TCP 端口		int

Name	描述	默认值	类型
strictEndpointUrls Enabled (common)	设置是否强制实施严格的端点 URL	false	布尔值
ServerName (common)	服务器名称		字符串
hostname (common)	服务器主机名		字符串
securityPolicies (common)	安全策略		Set
securityPoliciesById (common)	根据 URI 或名称进行安全策略		集合
UserAuthentication Credentials (common)	以 user1:pwd1,user2:pwd2 Usernames 和 password 形式设置用户密码组合将被解码		字符串
enableAnonymous Authentication (common)	启用匿名身份验证，默认禁用	false	布尔值
usernameSecurity Policy Uri (common)	设置在以下情况下使用的 UserTokenPolicy		SecurityPolicy
bindAddresses (common)	设置服务器应绑定到的本地地址的地址		字符串
buildInfo (common)	服务器构建信息		BuildInfo
ServerCertificate (common)	服务器证书		结果
certificateManager (common)	服务器证书管理器		CertificateManager
certificateValidator (common)	客户端证书的验证器		供应商
defaultCertificate Validator (common)	使用基于文件的默认方法的客户端证书验证器		File

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

223.1. URI 格式

> ***milo-server:itemId[?options]***

223.2. URI 选项

OPC UA 服务器端点使用 URI 语法进行配置：

> *milo-server:itemId*

使用以下路径和查询参数：

223.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
itemId	**项目所需的** ID		字符串

223.2.2. 查询参数(4 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

223.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 21 个选项，如下所列。

Name	描述	默认值	类型
camel.component.milo-server.application-name	应用程序名称		字符串
camel.component.milo-server.application-uri	应用程序 URI		字符串
camel.component.milo-server.bind-addresses	设置服务器应绑定到的本地地址的地址		字符串
camel.component.milo-server.bind-port	服务器绑定到的 TCP 端口		整数
camel.component.milo-server.build-info	服务器构建信息.选项是一个 org.eclipse.milo.opcua.stack.core.types.structured.BuildInfo 类型。		字符串
camel.component.milo-server.certificate-manager	服务器证书管理器.选项是一个 org.eclipse.milo.opcua.stack.core.application.CertificateManager 类型。		字符串
camel.component.milo-server.certificate-validator	客户端证书验证器。选项是一个 java.util.function.Supplier <org.eclipse.milo.opcua.stack.core.application.CertificateValidator> 类型。		字符串

Name	描述	默认值	类型
camel.component.milo-server.default-certificate-validator	使用基于文件的默认方法的客户端证书验证器		File
camel.component.milo-server.enable-anonymous-authentication	启用匿名身份验证，默认禁用	false	布尔值
camel.component.milo-server.enabled	启用 milo-server 组件	true	布尔值
camel.component.milo-server.hostname	服务器主机名		字符串
camel.component.milo-server.namespace-uri	命名空间的 URI，默认为 urn:org:apache:camel		字符串
camel.component.milo-server.product-uri	产品 URI		字符串
camel.component.milo-server.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.milo-server.security-policies	安全策略		Set
camel.component.milo-server.security-policies-by-id	根据 URI 或名称进行安全策略		集合

Name	描述	默认值	类型
camel.component.milo-server.server-certificate	服务器证书.选项是 org.apache.camel.component.milo.KeyStoreLoader.Result 类型。		字符串
camel.component.milo-server.server-name	服务器名称		字符串
camel.component.milo-server.strict-endpoint-urls-enabled	设置是否强制实施严格的端点 URL	false	布尔值
camel.component.milo-server.user-authentication-credentials	以 user1:pwd1,user2:pwd2 Usernames 和 password 形式设置用户密码组合将被解码		字符串
camel.component.milo-server.username-security-policy-uri	设置在以下情况下使用的 UserTokenPolicy		SecurityPolicy

223.4. 另请参阅

- ***配置 Camel***
- ***组件***
- ***端点***
- ***开始使用***

第 224 章 MIME MULTIPART DATAFORMAT

从 Camel 版本 2.17 开始提供

此数据格式，可以将带有附件的 Camel 消息转换为 Camel 消息，包含 MIME-Multipart 消息作为消息正文（无附件）。

这样做的用例是允许用户通过不支持附加的端点发送附件（例如，作为特殊协议实现（例如，通过 HTTP 端点发送 MIME 多部分）或作为隧道解决方案（例如，camel-jms 不支持附件，但通过将附件放入 MIME-Multipart）将消息放入 MIME-Multipart。将其发送到 JMS 队列，从 JMS 队列接收消息并再次接收消息（通过附件发送到消息正文）。

mime-multipart 数据格式的 marshal 选项将带有附件的消息转换为 MIME-Multipart 消息。如果参数 "multipartWithoutAttachment" 设置为 true，它将在没有附加信息的情况下进行 marshal 信息，如果该参数被设置为 false，它将只保留消息。

mulitpart 的 MIME 标头作为 "MIME-Version" 和 "Content-Type" 设置为消息。如果参数 "headersInline" 设置为 true，它将在任何情况下还会创建一个 MIME 多部分消息。
另外，多部分的 MIME 标头被编写为消息正文的一部分，而不是作为 camel 标头。

mime-multipart 数据格式的 unmarshal 选项将 MIME-Multipart 消息转换为带有附件的 camel 消息，并只保留其他消息。MIME-Multipart 消息的 MIME-Headers 必须设置为 Camel 标头。只有在将 "Content-Type" 标头设置为"multipart"类型时才会进行 unmarshalling。如果选项 "headersInline" 设为 true，则正文始终解析为 MIME 消息。因此，如果消息正文是流，并且没有启用流缓存，则实际不是消息正文中的 MIME 标头的消息正文将被空消息替换。最多 Camel 版本 2.17.1 将发生所有不包含 MIME 多部分消息的消息正文，无论正文类型和流缓存设置是什么。

224.1. 选项

MIME 多部分 dataformat 支持 6 个选项，如下所列。

Name	默认值	Java 类型	描述
multipartSubType	**mixed**	**字符串**	指定 MIME 多部分的子类型。默认为 mixed。
multipartWithoutAttachment	**false**	**布尔值**	定义没有附加的消息是否也会被放入 MIME Multipart 中（只有一个正文部分）。默认值为 false。

Name	默认值	Java 类型	描述
headersInline	**false**	**布尔值**	定义 MIME-Multipart 标头是消息正文(true)的一部分，或设置为 Camel 标头(false)。默认值为 false。
includeHeaders		**字符串**	定义将哪些 Camel 标头作为 MIME 标头包含在 MIME 多部分的正则表达式。这只有在 headersInline 设为 true 时才可以正常工作。默认为不包含标头
binaryContent	**false**	**布尔值**	定义 MIME 多部分的内容是否为二进制(true)或 Base-64 encoded (false) Default 是否为 false。
contentTypeHeader	**false**	**布尔值**	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

224.2. 消息标头(MARSHAL)

Name	类型	描述
Message-Id	字符串	如果 "headersInline" 参数设置为 false，则 marshal 操作会将此参数设置为生成的 MIME 消息 id。
MIME-Version	字符串	如果 "headersInline" 参数设置为 false，则 marshal 操作会将此参数设置为应用的 MIME 版本(1.0)。
Content-Type	字符串	此标头的内容将用作消息正文部分的内容类型。如果没有设置内容类型，则使用 "application/octet-stream"。在 marshal 操作后，如果 "headersInline" 参数设置为 true，则内容类型被设置为 "multipart/related" 或 empty。
content-Encoding	字符串	如果传入的内容类型是 "text configured"，则内容编码将设置为正文部分的 Content-Type MIME 标头的 encoding 参数。另外，给定 charset 应用于二进制转换的文本。

224.3. 消息标头(UNMARSHAL)

Name	类型	描述
Content-Type	字符串	如果此标头没有设置为 "multipart configured"，则 unmarshal 操作不会进行任何操作。在其他情况下，多部分将解析到带有附件的 camel 消息中，标头被设置为正文部分的 Content-Type 标头，除非这是 application/octet-stream。在后者的情况下，标头会被删除。

Name	类型	描述
content-Encoding	字符串	如果正文部分的内容类型包含一个编码参数，此标头将设置为此编码参数的值（从 MIME 结束描述符到 Java 编码描述符）
MIME-Version	字符串	unmarshal 操作将读取此标头，并使用它来解析 MIME 多部分。标头会在之后删除

224.4. 例子

```
from(...).marshal().mimeMultipart()
```

使用没有设置 Content-Type 标头的消息，将创建一个带有以下消息 Camel 标头的消息：

Camel 消息标头

```
Content-Type=multipart/mixed; \n boundary="----=_Part_0_14180567.1447658227051"
Message-Id=<...>
MIME-Version=1.0
```

```
The message body will be:
```

Camel 消息正文

```
------=_Part_0_14180567.1447658227051
Content-Type: application/octet-stream
Content-Transfer-Encoding: base64
Qm9keSB0ZXh0
------=_Part_0_14180567.1447658227051
Content-Type: application/binary
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="Attachment File Name"
AAECAwQFBgc=
------=_Part_0_14180567.1447658227051--
```

将标头 Content-Type 设置为 "text/plain" 的消息

```
from("...").marshal().mimeMultipart("related", true, true, "(included|x-.*)", true);
```

将创建一个没有特定 MIME 标头设置为 Camel 标头的消息(Content-Type 标头从 Camel 消息中删除)，以及包括原始消息的所有标头，并以 "x-" 开头的标头，以及名称 "included" 的标头。

Camel 消息正文

```
Message-ID: <...>
MIME-Version: 1.0
Content-Type: multipart/related;
    boundary="----=_Part_0_1134128170.1447659361365"
x-bar: also there
included: must be included
x-foo: any value

------=_Part_0_1134128170.1447659361365
Content-Type: text/plain
Content-Transfer-Encoding: 8bit

Body text
------=_Part_0_1134128170.1447659361365
Content-Type: application/binary
Content-Transfer-Encoding: binary
Content-Disposition: attachment; filename="Attachment File Name"

[binary content]
------=_Part_0_1134128170.1447659361365
```

224.5. 依赖项

要在 Camel 路由中使用 MIME-Multipart，您需要添加对实现此数据格式的 camel-mail 的依赖关系。

如果使用 Maven，您只需在 pom.xml 中添加以下内容：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-mail</artifactId>
  <version>x.x.x</version> <!-- use the same version as your Camel core version -->
</dependency>
```

第 225 章 MINA2 组件

从 Camel 版本 2.10 开始提供

mina2: 组件是处理 Apache MINA 2.x的传输

提示

使用 Netty 作为 Netty 是比当前 Apache Mina 更活跃的维护和流行的项目。

INFO ：请小心在消费者端点中使用 sync=false。由于 camel-mina2 所有消费者交换都是 InOut。这与 camel-mina 不同。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-mina2</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

225.1. URI 格式

```
mina2:tcp://hostname[:port][?options]
mina2:udp://hostname[:port][?options]
mina2:vm://hostname[:port][?options]
```

您可以使用 codec 选项指定 Registry 中的 codec。如果您使用 TCP，且没有指定 codec，则使用 textline 标志来确定是否应使用基于文本的 codec 或对象序列化。默认情况下使用对象序列化。

对于 UDP，如果没有指定 codec，则默认使用基于 basic ByteBuffer 的 codec。

VM 协议用作同一 JVM 中的直接转发机制。

Mina producer 的默认超时值为 30 秒，而它会等待远程服务器的响应。

在正常使用中，camel-mina 只支持 marshalling body content-message 标头和交换属性。但是，选项 transferExchange 可让您通过线路传输交换本身。请参见以下的选项。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

225.2. 选项

Mina2 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	使用共享 mina 配置。		Mina2Configuration
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Mina2 端点使用 URI 语法进行配置：

```
mina2:protocol:host:port
```

使用以下路径和查询参数：

225.2.1. 路径参数(3 参数)：

Name	描述	默认值	类型
protocol	要使用所需的协议		字符串

Name	描述	默认值	类型
主机	**要使用** 的主机名。将 localhost 或 0.0.0.0 用于本地服务器作为消费者。对于生成者，请使用远程服务器的主机名或 ip 地址。		字符串
port	**所需的** 端口号		int

225.2.2. 查询参数(27 参数)：

Name	描述	默认值	类型
disconnect (common)	使用后是否从 Mina 会话断开(close)。可用于使用者和制作者。	false	布尔值
minaLogger (common)	您可以启用 Apache MINA 日志记录过滤器。Apache MINA 在 INFO 级别使用 slf4j 日志记录来记录所有输入和输出。	false	布尔值
sync (common)	将端点设置为单向或请求响应。	true	布尔值
timeout (common)	您可以配置超时，指定等待远程服务器响应的时长。超时单元以毫秒为单位，因此 60000 为 60 秒。	30000	long
writeTimeout (common)	将数据发送到 MINA 会话所需的时间上限。默认值为 10000 毫秒。	10000	long
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
clientMode (consumer)	如果 clientMode 为 true，mina consumer 会将地址连接到 TCP 客户端。	false	布尔值
disconnectOnNoReply (consumer)	如果启用了同步，则此选项将指定 MinaConsumer（如果它应该断开连接），没有回复回来。	true	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern

Name	描述	默认值	类型
noReplyLogLevel (consumer)	如果启用了同步，这个选项会指示 MinaConsumer，在记录一个没有回复时要使用的日志记录级别。	WARN	LoggingLevel
cachedAddress (producer)	是否在创建 InetAddress 一次并重复使用。把它设置为 false 允许选择网络中的 DNS 更改。	true	布尔值
lazySessionCreation (producer)	如果远程服务器在 Camel producer 启动时未启动并在运行，则会话可能会被创建来避免异常。	true	布尔值
maximumPoolSize (advanced)	TCP 和 UDP 的 worker 池中 worker 线程数量	16	int
orderedThreadPoolExecutor (advanced)	是否使用排序的线程池，确保同一通道上按顺序处理事件。	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
transferExchange (advanced)	仅用于 TCP。您可以通过线线传输交换，而不是只传输正文。以下字段被传输： 在 body, Out body, fault body, In headers, Out headers, fault headers, Exchange properties, Exchange exception。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。	false	布尔值
allowDefaultCodec (codec)	如果两者都为 null，mina 组件会安装默认的 codec，文本行为 false。将 allowDefaultCodec 设置为 false 可防止 mina 组件将默认 codec 安装为过滤器链中的第一个元素。当另一个过滤器必须是过滤器链中的第一个过滤器（如 SSL 过滤器）时，这非常有用。	true	布尔值
codec (codec)	使用自定义 minda codec 实施。		ProtocolCodecFactory
decoderMaxLineLength (codec)	要设置文本行协议解码器最大行长度。默认情况下，使用 Mina 本身的默认值，即 1024。	1024	int
encoderMaxLineLength (codec)	要设置文本行协议编码器最大行长度。默认情况下，使用 Mina 本身的默认值，即 Integer.MAX_VALUE。	-1	int
编码 (codec)	您可以配置编码（字符名称）以用于 TCP 文本 codec 和 UDP 协议。如果没有提供，Camel 将使用 JVM 默认 Charset		字符串
过滤器 (codec)	您可以设置要使用的 Mina IoFilters 列表。		list

Name	描述	默认值	类型
文本行 (codec)	仅用于 TCP。如果没有指定 codec，您可以使用此标志来指示基于文本的 codec；如果没有指定，或者值为 false，则通过 TCP 假设 Object Serialization。	false	布尔值
textlineDelimiter (codec)	仅用于 TCP，并且使用 textline=true。设置要使用的文本行分隔符。如果没有提供，Camel 将使用 DEFAULT。此分隔符用于标记文本的末尾。		Mina2TextLineDelimiter
autoStartTls (security)	是否自动启动 SSL 握手。	true	布尔值
sslContextParameters (security)	配置 SSL 安全性。		SSLContextParameters

225.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 29 个选项，如下所列。

Name	描述	默认值	类型
camel.component.mina2.configuration.allow-default-codec	如果两者都为 null，mina 组件会安装默认的 codec，文本行为 false。将 allowDefaultCodec 设置为 false 可防止 mina 组件将默认 codec 安装为过滤器链中的第一个元素。当另一个过滤器必须是过滤器链中的第一个过滤器（如 SSL 过滤器）时，这非常有用。	true	布尔值
camel.component.mina2.configuration.auto-start-tls	是否自动启动 SSL 握手。	true	布尔值
camel.component.mina2.configuration.cached-address	是否在创建 InetAddress 一次并重复使用。把它设置为 false 允许选择网络中的 DNS 更改。	true	布尔值
camel.component.mina2.configuration.client-mode	如果 clientMode 为 true，mina consumer 会将地址连接到 TCP 客户端。	false	布尔值
camel.component.mina2.configuration.codec	使用自定义 minda codec 实施。		ProtocolCodecFactory

Name	描述	默认值	类型
camel.component.mina2.configuration.decoder-max-line-length	要设置文本行协议解码器最大行长度。默认情况下，使用 Mina 本身的默认值，即 1024。	1024	整数
camel.component.mina2.configuration.disconnect	使用后是否从 Mina 会话断开(close)。可用于使用者和制作者。	false	布尔值
camel.component.mina2.configuration.disconnect-on-no-reply	如果启用了同步，则此选项将指定 MinaConsumer（如果它应该断开连接），没有回复回来。	true	布尔值
camel.component.mina2.configuration.encoder-max-line-length	要设置文本行协议编码器最大行长度。默认情况下，使用 Mina 本身的默认值，即 Integer.MAX_VALUE。	-1	整数
camel.component.mina2.configuration.encoding	您可以配置编码（字符名称）以用于 TCP 文本 codec 和 UDP 协议。如果没有提供，Camel 将使用 JVM 默认 Charset		字符串
camel.component.mina2.configuration.filters	您可以设置要使用的 Mina IoFilters 列表。		list
camel.component.mina2.configuration.host	要使用的主机名。将 localhost 或 0.0.0.0 用于本地服务器作为消费者。对于生成者，请使用远程服务器的主机名或 ip 地址。		字符串
camel.component.mina2.configuration.lazy-session-creation	如果远程服务器在 Camel producer 启动时未启动并在运行，则会话可能会被创建来避免异常。	true	布尔值
camel.component.mina2.configuration.maximum-pool-size	TCP 和 UDP 的 worker 池中 worker 线程数量	16	整数
camel.component.mina2.configuration.mina-logger	您可以启用 Apache MINA 日志记录过滤器。Apache MINA 在 INFO 级别使用 slf4j 日志记录来记录所有输入和输出。	false	布尔值

Name	描述	默认值	类型
camel.component.mina2.configuration.no-reply-log-level	如果启用了同步，这个选项会指示 MinaConsumer，在记录一个没有回复时要使用的日志记录级别。		LoggingLevel
camel.component.mina2.configuration.ordered-thread-pool-executor	是否使用排序的线程池，确保同一通道上按顺序处理事件。	true	布尔值
camel.component.mina2.configuration.port	端口号		整数
camel.component.mina2.configuration.protocol	要使用的协议		字符串
camel.component.mina2.configuration.ssl-context-parameters	配置 SSL 安全性。		SSLContextParameters
camel.component.mina2.configuration.sync	将端点设置为单向或请求响应。	true	布尔值
camel.component.mina2.configuration.textline	仅用于 TCP。如果没有指定 codec，您可以使用此标志来指示基于文本的 codec；如果没有指定，或者值为 false，则通过 TCP 假设 Object Serialization。	false	布尔值
camel.component.mina2.configuration.textline-delimiter	仅用于 TCP，并且使用 textline=true。设置要使用的文本行分隔符。如果没有提供，Camel 将使用 DEFAULT。此分隔符用于标记文本的末尾。		Mina2TextLineDelimiter
camel.component.mina2.configuration.timeout	您可以配置超时，指定等待远程服务器响应的时长。超时单元以毫秒为单位，因此 60000 为 60 秒。	30000	Long
camel.component.mina2.configuration.transfer-exchange	仅用于 TCP。您可以通过线线传输交换，而不是只传输正文。以下字段被传输： 在 body, Out body, fault body, In headers, Out headers, fault headers, Exchange properties, Exchange exception。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。	false	布尔值

Name	描述	默认值	类型
camel.component.mina2.configuration.write-timeout	将数据发送到 MINA 会话所需的时间上限。默认值为 10000 毫秒。	10000	Long
camel.component.mina2.enabled	启用 mina2 组件	true	布尔值
camel.component.mina2.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.mina2.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

225.4. 使用自定义 CODEC

请参阅 Mina 如何编写您自己的 codec。要将自定义 codec 与 camel-mina 搭配使用，您应该在 Registry 中注册 codec；例如，在 Spring XML 文件中创建一个 bean。然后，使用 codec 选项指定 codec 的 bean ID。请参阅具有自定义 codec 的 HL7。

225.5. 带有 SYNC=FALSE 的示例

在本例中，Camel 会公开侦听端口 6200 上的 TCP 连接的服务。我们使用 文本行 codec。在我们的路由中，我们创建一个侦听端口 6200 的 Mina consumer 端点：

```
from("mina2:tcp://localhost:" + port1 + "?textline=true&sync=false").to("mock:result");
```

因为示例是单元测试的一部分，我们在端口 6200 上向它发送一些数据来测试它。

```
MockEndpoint mock = getMockEndpoint("mock:result");
mock.expectedBodiesReceived("Hello World");

template.sendBody("mina2:tcp://localhost:" + port1 + "?textline=true&sync=false", "Hello World");

assertMockEndpointsSatisfied();
```

225.6. 带有 SYNC=TRUE 的示例

在下一个示例中，在端口 6201 上公开 TCP 服务时，我们有一个更为常见的用例。但是，这次我们想返回响应，因此我们在消费者上将 sync 选项设置为 true。

```
from("mina2:tcp://localhost:" + port2 + "?textline=true&sync=true").process(new Processor() {
    public void process(Exchange exchange) throws Exception {
        String body = exchange.getIn().getBody(String.class);
        exchange.getOut().setBody("Bye " + body);
    }
});
```

然后，我们通过使用 template.requestBody （） 方法发送一些数据并检索响应来测试示例。正如我们知道的响应是字符串，我们会把它转换为 String ，并可以断言响应实际上，我们已在处理器代码逻辑中动态设置。

```
String response = (String)template.requestBody("mina2:tcp://localhost:" + port2 + "?
textline=true&sync=true", "World");
assertEquals("Bye World", response);
```

225.7. 使用 SPRING DSL 的示例

当然，Spring DSL 也可用于 MINA。在以下示例中，我们在端口 5555 上公开 TCP 服务器：

```
<route>
  <from uri="mina2:tcp://localhost:5555?textline=true"/>
  <to uri="bean:myTCPOrderHandler"/>
</route>
```

在上面的路由中，我们使用文本 codec 在端口 5555 上公开 TCP 服务器。我们让 Spring bean 带有 ID myTCPOrderHandler，处理请求并返回回复。例如，handler bean 可以实施，如下所示：

```
public String handleOrder(String payload) {
    ...
    return "Order: OK"
}
```

225.8. 关闭会话完成后

当充当服务器时，有时希望在关闭会话时关闭客户端转换。要指示 Camel 关闭会话，您应该添加一个带有键 CamelMinaCloseSessionWhenComplete 设置为布尔值 true 的标头。

例如，以下示例会在将消息写回客户端后关闭会话：

```
        from("mina2:tcp://localhost:8080?sync=true&textline=true").process(new Processor() {
            public void process(Exchange exchange) throws Exception {
                String body = exchange.getIn().getBody(String.class);
                exchange.getOut().setBody("Bye " + body);

exchange.getOut().setHeader(Mina2Constants.MINA_CLOSE_SESSION_WHEN_COMPLETE,
true);
            }
        });
```

225.9. 获取用于消息的 IOSESSION

您可以从带有此键 Mina2Constants.MINA_IOSESSION 的消息标头中获取 IoSession，并获得带有键 Mina2Constants.MINA_LOCAL_ADDRESS 和远程主机地址为 Mina2Constants.MINA_REMOTE_ADDRESS 的本地主机地址的 IoSession。

225.10. 配置 MINA 过滤器

过滤器允许您使用一些 Mina 过滤器，如 SslFilter。您还可以实施一些自定义过滤器。请注意，codec 和 logger 也作为 Mina 过滤器实现，即 IoFilter。您可以定义的任何过滤器都附加到过滤器链的末尾；即 codec 和 logger 后。

225.11. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- Netty

第 226 章 MLLP 组件

从 Camel 版本 2.17 开始提供

MLLP 组件旨在处理 MLLP 协议的细微，并提供 Healthcare 供应商使用 MLLP 协议与其他系统通信所需的功能。MLLP 组件提供一个简单的配置 URI，自动化 HL7 确认生成和自动确认。

MLLP 协议通常不使用大量并发 TCP 连接 - 单个活跃的 TCP 连接是正常情况。因此，MLLP 组件使用基于标准 Java 套接字的简单线程每个连接模型。这会保持实施简单，并消除了 Camel 本身以外的依赖项。

组件支持以下内容：

- 使用 TCP 服务器的 Camel 使用者
- 使用 TCP 客户端的 Camel producer

MLLP 组件使用 byte[] 有效负载，并依赖于 Camel Type Conversion 将 byte[] 转换为其他类型。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-mllp</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

226.1. MLLP 选项

MLLP 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
logPhi (advanced)	将组件设置为 log PHI 数据。	true	布尔值

Name	描述	默认值	类型
logPhiMaxBytes (advanced)	设置在日志条目中登录的最大 PHI 字节数。	5120	整数
defaultCharset (advanced)	设置默认的字符集，用于字节/来自字符串转换。	ISO-8859-1	字符串
configuration (common)	设置创建 MLLP 端点时要使用的默认配置。		MllpConfiguration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

MLLP 端点使用 URI 语法进行配置：

```
mllp:hostname:port
```

使用以下路径和查询参数：

226.1.1. 路径参数(2 参数)：

Name	描述	默认值	类型
hostname	**必需** TCP 连接的连接的主机名或 IP。默认值为 null，这意味着任何本地 IP 地址		字符串
port	TCP 连接所需的端口号		int

226.1.2. 查询参数(27 参数)：

Name	描述	默认值	类型
autoAck (common)	仅启用/禁用 MLLP Acknowledgement MLLP Consumers 的自动生成	true	布尔值
bufferWrites (common)	在写入套接字前，**弃用了** HL7 有效负载的 Enable/Disable 缓冲。	false	布尔值
hl7Headers (common)	仅启用/禁用 HL7 Message MLLP Consumers 自动生成消息标头	true	布尔值

Name	描述	默认值	类型
requireEndOfData (common)	启用/禁用对 MLLP 标准的严格合规性。MLLP 标准指定 START_OF_BLOCKhl7 payloadEND_OF_BLOCKEND_OF_DATA，但有些系统不会发送最终 END_OF_DATA 字节。此设置控制最终 END_OF_DATA 字节是否需要或可选。	true	布尔值
stringPayload (common)	启用/禁用将有效负载转换为 String。如果启用，从外部系统接收的 HL7 Payloads 将转换为 String。如果设置了 charsetName 属性，则该字符集将用于转换。如果没有设置 charsetName 属性，则会使用 MSH-18 的值来确定适当的字符集。如果没有设置 MSH-18，则将使用默认 ISO-8859-1 字符集。	true	布尔值
validatePayload (common)	启用/禁用 HL7 Payloads 验证（如果已启用），则验证从外部系统接收的 HL7 Payloadloads （有关验证的详情，请参阅 Hl7Util.generateInvalidPayloadExceptionMessage）。如果检测到无效有效负载，则会抛出一个 MllpInvalidMessageException （用于消费者）或 MllpInvalidAcknowledgementException。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由 Error Handler，这意味着当消费者试图接收传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。如果禁用，使用者将使用 org.apache.camel.spi.ExceptionHandler 在 WARN 或 ERROR 级别记录它们来处理例外情况，并忽略它们。	true	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。	InOut	ExchangePattern
同步 （高级）	设置是否应严格使用同步处理（此组件只支持同步操作）。	true	布尔值
积压 (tcp)	传入连接的最大队列长度（要连接的请求）设置为 backlog 参数。如果连接表示队列已满时到达，则拒绝连接。	5	整数
lenientBind (tcp)	仅 TCP 服务器 - 允许端点在绑定 TCP 服务器Socket 之前启动。在某些环境中，可能需要允许端点在绑定 TCP 服务器Socket 之前启动。	false	布尔值
maxConcurrentConsumers (tcp)	允许的最大 MLLP Consumer 连接数。如果收到新连接并且已经建立最大数，新的连接将立即重置。	5	int

Name	描述	默认值	类型
reuseAddress (tcp)	启用/禁用 SO_REUSEADDR 套接字选项。	false	布尔值
acceptTimeout (timeout)	仅在等待 TCP 连接 TCP 服务器时超时（以毫秒为单位）	60000	int
bindRetryInterval (timeout)	仅 TCP 服务器 - 绑定尝试之间等待的毫秒数	5000	int
bindTimeout (timeout)	仅 TCP 服务器 - 重试绑定到服务器端口的毫秒数	30000	int
connectTimeout (timeout)	只为 TCP 连接 TCP 客户端建立超时（以毫秒为单位）	30000	int
idletimeout (timeout)	重置客户端 TCP 连接前允许的大约空闲时间。null 值或小于或等于 0 的值将禁用闲置超时。		整数
maxReceiveTimeouts (timeout)	**弃用** 在重置 TCP 连接前允许的最大超时数（由 receiveTimeout 指定）。		整数
keepalive (tcp)	启用/禁用 SO_KEEPALIVE 套接字选项。	true	布尔值
receiveBufferSize (tcp)	将 SO_RCVBUF 选项设置为指定的值（以字节为单位）	8192	整数
sendBufferSize (tcp)	将 SO_SNDBUF 选项设置为指定的值（以字节为单位）	8192	整数
tcpNoDelay (tcp)	启用/禁用 TCP_NODELAY 套接字选项。	true	布尔值
readTimeout (timeout)	接收 MLLP 帧后使用的 SO_TIMEOUT 值（以毫秒为单位）	5000	int
receiveTimeout (timeout)	等待 MLLP 帧的开始时使用的 SO_TIMEOUT 值（以毫秒为单位）	15000	int
charsetName (codec)	在交换上设置 CamelCharsetName 属性		字符串

226.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 31 个选项，如下所列。

Name	描述	默认值	类型
camel.component.mllp.configuration.accept-timeout	仅在等待 TCP 连接 TCP 服务器时超时（以毫秒为单位）	60000	整数
camel.component.mllp.configuration.auto-ack	仅启用/禁用 MLLP Acknowledgement MLLP Consumers 的自动生成	true	布尔值
camel.component.mllp.configuration.backlog	传入连接的最大队列长度（要连接的请求）设置为 backlog 参数。如果连接表示队列已满时到达，则拒绝连接。	5	整数
camel.component.mllp.configuration.bind-retry-interval	仅 TCP 服务器 - 绑定尝试之间等待的毫秒数	5000	整数
camel.component.mllp.configuration.bind-timeout	仅 TCP 服务器 - 重试绑定到服务器端口的毫秒数	30000	整数
camel.component.mllp.configuration.bridge-error-handler	允许将消费者桥接到 Camel 路由 Error Handler，这意味着当消费者试图接收传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。如果禁用，使用者将使用 org.apache.camel.spi.ExceptionHandler 在 WARN 或 ERROR 级别记录它们来处理例外情况，并忽略它们。	true	布尔值
camel.component.mllp.configuration.charset-name	在交换上设置 CamelCharsetName 属性		字符串
camel.component.mllp.configuration.connect-timeout	只为 TCP 连接 TCP 客户端建立超时（以毫秒为单位）	30000	整数
camel.component.mllp.configuration.exchange-pattern	在消费者创建交换时设置交换模式。		ExchangePattern
camel.component.mllp.configuration.hl7-headers	仅启用/禁用 HL7 Message MLLP Consumers 自动生成消息标头	true	布尔值
camel.component.mllp.configuration.idle-timeout	重置客户端 TCP 连接前允许的大约空闲时间。null 值或小于或等于 0 的值将禁用闲置超时。		整数

Name	描述	默认值	类型
camel.component.mllp.configuration.keep-alive	启用/禁用 SO_KEEPALIVE 套接字选项。	true	布尔值
camel.component.mllp.configuration.lenient-bind	仅 TCP 服务器 - 允许端点在绑定 TCP 服务器Socket 之前启动。在某些环境中，可能需要允许端点在绑定 TCP 服务器Socket 之前启动。	false	布尔值
camel.component.mllp.configuration.max-concurrent-consumers	允许的最大 MLLP Consumer 连接数。如果收到新连接并且已经建立最大数，新的连接将立即重置。	5	整数
camel.component.mllp.configuration.read-timeout	接收 MLLP 帧后使用的 SO_TIMEOUT 值（以毫秒为单位）	5000	整数
camel.component.mllp.configuration.receive-buffer-size	将 SO_RCVBUF 选项设置为指定的值（以字节为单位）	8192	整数
camel.component.mllp.configuration.receive-timeout	等待 MLLP 帧的开始时使用的 SO_TIMEOUT 值（以毫秒为单位）	15000	整数
camel.component.mllp.configuration.require-end-of-data	启用/禁用对 MLLP 标准的严格合规性。MLLP 标准指定 START_OF_BLOCKhl7 payloadEND_OF_BLOCKEND_OF_DATA，但有些系统不会发送最终 END_OF_DATA 字节。此设置控制最终 END_OF_DATA 字节是否需要或可选。	true	布尔值
camel.component.mllp.configuration.reuse-address	启用/禁用 SO_REUSEADDR 套接字选项。	false	布尔值
camel.component.mllp.configuration.send-buffer-size	将 SO_SNDBUF 选项设置为指定的值（以字节为单位）	8192	整数
camel.component.mllp.configuration.string-payload	启用/禁用将有效负载转换为 String。如果启用，从外部系统接收的 HL7 Payloads 将转换为 String。如果设置了 charsetName 属性，则该字符集将用于转换。如果没有设置 charsetName 属性，则会使用 MSH-18 的值来确定适当的字符集。如果没有设置 MSH-18，则将使用默认 ISO-8859-1 字符集。	true	布尔值

Name	描述	默认值	类型
camel.component.mllp.configuration.synchronous	设置是否应严格使用同步处理（此组件只支持同步操作）。	true	布尔值
camel.component.mllp.configuration.tcp-no-delay	启用/禁用 TCP_NODELAY 套接字选项。	true	布尔值
camel.component.mllp.configuration.validate-payload	启用/禁用 HL7 Payloads 验证（如果已启用），则验证从外部系统接收的 HL7 Payloadloads （有关验证的详情，请参阅 Hl7Util.generateInvalidPayloadExceptionMessage）。如果检测到无效有效负载，则会抛出一个 MllpInvalidMessageException （用于消费者）或 MllpInvalidAcknowledgementException。	false	布尔值
camel.component.mllp.default-charset	设置默认的字符集，用于字节/来自字符串转换。	ISO-8859-1	字符串
camel.component.mllp.enabled	启用 mllp 组件	true	布尔值
camel.component.mllp.log-phi	将组件设置为 log PHI 数据。	true	布尔值
camel.component.mllp.log-phi-max-bytes	设置在日志条目中登录的最大 PHI 字节数。	5120	整数
camel.component.mllp.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.mllp.configuration.buffer-writes	在写入套接字前，启用/禁用 HL7 有效负载的缓冲。	false	布尔值
camel.component.mllp.configuration.max-receive-timeouts	在重置 TCP 连接前允许的最大超时数（由 receiveTimeout 指定）。		整数

226.3. MLLP CONSUMER

MLLP Consumer 支持接受 MLLP-framed 消息并发送 HL7 Acknowledgements。MLLP Consumer 可以自动生成 HL7 Acknowledgement （仅限HL7 Application Acknowledgements - AA、AE 和 AR），也可以使用 CamelMllpAcknowledgement 来指定确认。另外，可以通过设置 CamelMllpAcknowledgementType Exchange 属性来控制生成的确认类型。

226.4. 消息标头

MLLP Consumer 在 Camel 消息上添加这些标头：

键	描述	
CamelMllpLocalAddress	套接字的本地 TCP 地址	
CamelMllpRemoteAddress	套接字的本地 TCP 地址	
CamelMllpSendingApplication	MSH-3 值	
CamelMllpSendingFacility	MSH-4 值	
CamelMllpReceivingApplication	MSH-5 值	
CamelMllpReceivingFacility	MSH-6 值	
CamelMllpTimestamp	MSH-7 值	
CamelMllpSecurity	MSH-8 值	
CamelMllpMessageType	MSH-9 值	
CamelMllpEventType	MSH-9-1 值	
CamelMllpTriggerEvent	MSH-9-2 值	
CamelMllpMessageControlId	MSH-10 值	
CamelMllpProcessingId	MSH-11 值	
CamelMllpVersionId	MSH-12 值	
CamelMllpCharset	MSH-18 值	

所有标头都是 String 类型。如果缺少标头值，则其值为 null。

226.5. 交换属性

确认 MLLP Consumer 生成和 TCP 套接字状态的类型可以由 Camel 交换上的这些属性控制：

键	类型	描述
CamelMllpAcknowledgement	byte[]	如果存在，此属性将作为 MLLP Acknowledgement 发送到客户端
CamelMllpAcknowledgementString	字符串	如果不存在和 CamelMllpAcknowledgement，则此属性将作为 MLLP Acknowledgement 发送到客户端
CamelMllpAcknowledgementMsaText	字符串	如果没有 CamelMllpAcknowledgement 或 CamelMllpAcknowledgementString，并且 autoAck 为 true，则此属性可用于在生成的 HL7 确认中指定 MSA-3 的内容
CamelMllpAcknowledgementType	字符串	如果没有 CamelMllpAcknowledgement 或 CamelMllpAcknowledgementString，并且 autoAck 为 true，则此属性可以用来指定 HL7 确认类型（即 AA、AE、AR）
CamelMllpAutoAcknowledge	布尔值	覆盖 autoAck 查询参数
CamelMllpCloseConnectionBeforeSend	布尔值	如果为 true，在发送数据前，套接字将关闭
CamelMllpResetConnectionBeforeSend	布尔值	如果为 true，在发送数据前会重置套接字
CamelMllpCloseConnectionAfterSend	布尔值	如果为 true，在发送数据后套接字会立即关闭
CamelMllpResetConnectionAfterSend	布尔值	如果为 true，在发送任何数据后将立即重置套接字

226.6. MLLP PRODUCER

MLLP Producer 支持发送 MLLP-framed 消息并接收 HL7 Acknowledgements。MLLP Producer 干预 HL7 Acknowledgments，并在收到负确认时引发异常。收到的确认被干扰，在出现负确认时引发异常。

226.7. 消息标头

MLLP Producer 在 Camel 消息上添加这些标头：

键	描述	
CamelMllpLocalAddress	套接字的本地 TCP 地址	
CamelMllpRemoteAddress	套接字的远程 TCP 地址	
CamelMllpAcknowledgement	收到的 HL7 Acknowledgment byte[]	
CamelMllpAcknowledgementString	收到的 HL7 Acknowledgment 转换为字符串	

226.8. 交换属性

TCP 套接字的状态可由 Camel 交换上的这些属性控制：

键	类型	描述
CamelMllpCloseConnectionBeforeSend	布尔值	如果为 true，在发送数据前，套接字将关闭
CamelMllpResetConnectionBeforeSend	布尔值	如果为 true，在发送数据前会重置套接字
CamelMllpCloseConnectionAfterSend	布尔值	如果为 true，在发送数据后套接字会立即关闭
CamelMllpResetConnectionAfterSend	布尔值	如果为 true，在发送任何数据后将立即重置套接字

第 227 章 MOCK 组件

从 Camel 版本 1.0 开始提供

测试分布式和异步处理非常困难。Mock、Test 和 DataSet 端点与 Camel 测试框架协同工作，从而通过使用 企业集成模式 和 Camel 的大量组件以及强大的 Bean 集成来简化您的单元和集成测试。

Mock 组件提供强大的声明测试机制，它类似于 jMock，它允许在测试开始前在任何 Mock 端点上创建声明预期。然后，运行测试，它通常会触发消息到一个或多个端点，最终在测试情况下可以预期确保系统按预期工作。

这可让您测试各种内容：

- 每个端点上都收到正确的消息数，
- 按照正确的顺序接收正确的有效负载，
- 消息按顺序到达端点，使用一些 Expression 来创建顺序测试功能，
- 消息到达某种形式的 Predicate，如该特定标头具有特定值，或者消息的部分与某些 predicate 匹配，例如通过评估 XPath 或 XQuery Expression。

注意

还有一个 Test 端点，它是一个 Mock 端点，但使用第二个端点来提供预期的消息正文列表，并自动设置 Mock 端点断言。换句话说，这是一个 Mock 端点，它会自动从文件或数据库中 的一些示例消息设置断言，例如：

小心

模拟端点会无限期地显示接收的交换。

请记住，Mock 是为测试而设计的。将 Mock 端点添加到路由时，发送到端点的每个交换都将存储在内存中（允许稍后验证），直到显式重置或 JVM 重启为止。如果您要发送大量卷和/或大信息，这可能会导致过量内存使用。如果您的目标是测试内联可部署路由，请考虑在测试中使用 NotifyBuilder 或 AdviceWith，而不是将 Mock 端点直接添加到路由中。

从 Camel 2.10 开始，有两个新选项保留First，并使用 retainLast 来限制 Mock 端点保留的消息数量。

227.1. URI 格式

```
mock:someName[?options]
```

其中 someName 可以是唯一标识端点的任何字符串。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

227.2. 选项

Mock 组件没有选项。

Mock 端点使用 URI 语法进行配置：

```
mock:name
```

使用以下路径和查询参数：

227.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	模拟端点 **必需** 名称		字符串

227.2.2. 查询参数(10 parameters):

Name	描述	默认值	类型
assertPeriod (producer)	设置一个宽限期，之后 mock 端点将重新断言，以确保初始断言仍然有效。例如，这只用于表示多个消息到达的 assert。例如，如果 expectedMessageCount (int)设为 5，则当 5 个或更多消息到达时，会满足断言。为确保准确 5 个消息到达，您需要稍等片刻，以确保进一步的消息到达。这是您可以对此使用 setAssertPeriod （长）方法的内容。默认情况下禁用这个周期。	0	long
expectedCount (producer)	指定此端点应接收的消息交换数量。注意：如果要预期 0 个消息，请在测试启动时做额外的小心，以 0 匹配，因此您需要设置一个 assert 周期时间以便让测试在一段时间内运行，以确保仍然没有消息到达；对于 setAssertPeriod （长）。另一种方法是使用 NotifyBuilder，并在调用 mocks 上的 assertIsSatisfied （）方法前，使用 notifier 知道 Camel 何时进行路由。这可让您不使用固定的 assert 周期来加快测试时间。如果您要断言，其中第 n 个消息到达这个 mock 端点，则还要查看 setAssertPeriod (long)方法了解更多详情。	-1	int
reportGroup (producer)	用于根据大小组打开吞吐量日志记录的数字。		int
resultMinimumWaitTime (producer)	设置 assertIsSatisfied （）的最小预期时间（单位为 millis）将等待 latch，直到它满足为止	0	long
resultWaitTime (producer)	设置 assertIsSatisfied （）将等待的最大时间量，直到它满足为止	0	long
retainFirst (producer)	指定只保留收到的前 n 个接收交换数。这在测试大数据时，通过不存储此模拟端点接收的每一个交换的副本来减少内存消耗。重要： 使用此限制时，getReceivedCounter （）仍然会返回接收的交换的实际数量。例如，如果我们收到 5000 Exchanges，并且配置为仅保留前 10 个交换，那么 getReceivedCounter （） 仍然会返回 5000，但 getExchanges （）中只有前 10 Exchanges （） 和 getReceivedExchanges （）方法。使用此方法时，不支持其他一些预期的方法，例如 expectedBodiesReceived (Object...)在收到的第一个正文数上设置预期。您可以配置 setRetainFirst (int)和 setRetainLast (int)方法，以限制第一个和最后一个收到。	-1	int

Name	描述	默认值	类型
retainLast (producer)	指定只保留最后第 n 个接收交换的数量。这在测试大数据时，通过不存储此模拟端点接收的每一个交换的副本来减少内存消耗。重要： 使用此限制时，getReceivedCounter （）仍然会返回接收的交换的实际数量。例如，如果我们收到 5000 Exchanges，并且配置为仅保留最后 20 个交换，那么 getReceivedCounter （） 仍然会返回 5000，但 getExchanges （） 中只有最后 20 个交换程序）和 getReceivedExchanges （）方法。使用此方法时，不支持其他一些预期的方法，例如 expectedBodiesReceived (Object...)在收到的第一个正文数上设置预期。您可以配置 setRetainFirst (int)和 setRetainLast (int)方法，以限制第一个和最后一个收到。	-1	int
sleepForEmptyTest (producer)	使用零调用 expectedMessageCount (int)时，允许指定 sleep 来检查此端点是否确实为空	0	long
copyOnExchange (producer)	设置在这个模拟端点收到传入 Exchange 时是否进行深度副本。默认为 true。	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

227.3. 简单示例

以下是使用的 Mock 端点的简单示例。首先，端点在上下文中解析。然后，我们在测试运行后设置一个预期，然后在测试运行后对预期得到满足：

```
MockEndpoint resultEndpoint = context.resolveEndpoint("mock:foo", MockEndpoint.class);

// set expectations
resultEndpoint.expectedMessageCount(2);

// send some messages

// now lets assert that the mock:foo endpoint received 2 messages
resultEndpoint.assertIsSatisfied();
```

您通常始终调用 assertIsSatisfied （） 方法，以测试在运行测试后是否满足预期。

当调用 assertIsSatisfied （） 时，Camel 默认会等待 10 秒。这可以通过设置 setResultWaitTime (millis) 方法进行配置。

227.4. 使用 ASSERTPERIOD

从 Camel 2.7 开始提供
当断言满意时，Camel 将停止等待并继续 assertIsSatisfied 方法。这意味着，如果新消息到达模拟端点，稍后仅有点，aurival 不会影响断言的结果。假设您要测试在一段时间后没有新消息到达，您可以通过设置 setAssertPeriod 方法来实现这一目的，例如：

```
MockEndpoint resultEndpoint = context.resolveEndpoint("mock:foo", MockEndpoint.class);
resultEndpoint.setAssertPeriod(5000);
resultEndpoint.expectedMessageCount(2);

// send some messages

// now lets assert that the mock:foo endpoint received 2 messages
resultEndpoint.assertIsSatisfied();
```

227.5. 设置预期

您可以从 MockEndpoint 的 Javadoc 中看到用于设置预期的各种帮助程序方法。主要方法如下：

æ–¹æ³•	描述
expectedMessageCount(int)	在端点上定义预期的消息计数。
expectedMinimumMessageCount (int)	在端点上定义预期消息的最小数量。
expectedBodiesReceived(...)	定义应收到的预期正文（按顺序）。
expectedHeaderReceived(...)	定义应接收的预期标头
expectsAscending (Expression)	要添加预期消息按顺序接收，请使用给定的 Expression 来比较消息。
expectsDescending (Expression)	要添加预期消息按顺序接收，请使用给定的 Expression 来比较消息。
expectsNoDuplicates (Expression)	要添加预期，不会收到重复消息；使用 Expression 来计算每个消息的唯一标识符。如果使用 JMS，这可能类似于 **JMSMessageID**，或者消息中的某些唯一引用号。

下面是另一个示例：

```
resultEndpoint.expectedBodiesReceived("firstMessageBody", "secondMessageBody",
"thirdMessageBody");
```

227.6. 向特定消息添加预期

另外，您可以使用 message (int messageIndex) 方法添加有关收到的特定消息的断言。

例如，要添加第一个消息的标头或正文的预期（使用零索引，如 java.util.List），您可以使用以下代码：

```
resultEndpoint.message(0).header("foo").isEqualTo("bar");
```

camel-core 处理器测试 中使用的 Mock 端点有一些示例。

227.7. 模拟现有端点

从 Camel 2.7 开始提供

Camel 现在允许您自动模拟 Camel 路由中的现有端点。

注意

它如何让 端点仍在操作中。发生了什么情况是，Mock 端点被注入并首先接收消息，然后将消息委派给目标端点。您可以将它视为拦截器类型，以及委派或端点监听程序。

假设您有以下给定路由：

Route

```
@Override
protected RouteBuilder createRouteBuilder() throws Exception {
    return new RouteBuilder() {
        @Override
        public void configure() throws Exception {
            from("direct:start").to("direct:foo").to("log:foo").to("mock:result");
```

```
            from("direct:foo").transform(constant("Bye World"));
        }
    };
}
```

然后，您可以使用 Camel 中的 recommendations With 功能模拟单元测试中给定路由中的所有端点，如下所示：

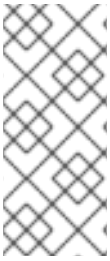
recommendationsWith 模拟所有端点

```
public void testAdvisedMockEndpoints() throws Exception {
    // advice the first route using the inlined AdviceWith route builder
    // which has extended capabilities than the regular route builder
    context.getRouteDefinitions().get(0).adviceWith(context, new AdviceWithRouteBuilder() {
        @Override
        public void configure() throws Exception {
            // mock all endpoints
            mockEndpoints();
        }
    });

    getMockEndpoint("mock:direct:start").expectedBodiesReceived("Hello World");
    getMockEndpoint("mock:direct:foo").expectedBodiesReceived("Hello World");
    getMockEndpoint("mock:log:foo").expectedBodiesReceived("Bye World");
    getMockEndpoint("mock:result").expectedBodiesReceived("Bye World");

    template.sendBody("direct:start", "Hello World");

    assertMockEndpointsSatisfied();

    // additional test to ensure correct endpoints in registry
    assertNotNull(context.hasEndpoint("direct:start"));
    assertNotNull(context.hasEndpoint("direct:foo"));
    assertNotNull(context.hasEndpoint("log:foo"));
    assertNotNull(context.hasEndpoint("mock:result"));
    // all the endpoints was mocked
    assertNotNull(context.hasEndpoint("mock:direct:start"));
    assertNotNull(context.hasEndpoint("mock:direct:foo"));
    assertNotNull(context.hasEndpoint("mock:log:foo"));
}
```

请注意，模拟端点被授予 URI mock:<endpoint>，如 mock:direct:foo。INFO 级别的 Camel 日志

级别是模拟的端点：

```
INFO  Adviced endpoint [direct://foo] with mock endpoint [mock:direct:foo]
```



注意

模拟的端点没有参数
端点，被模拟将具有被模拟的参数被剥离。例如，端点 log:foo?showAll=true 将被模拟到以下端点 mock:log:foo。注意参数已被删除。

也可以仅使用模式模拟某些端点。例如，要模拟您执行 的所有日志 端点，如下所示：

recommendationsWith mocking only log endpoint using a pattern

```
public void testAdvisedMockEndpointsWithPattern() throws Exception {
    // advice the first route using the inlined AdviceWith route builder
    // which has extended capabilities than the regular route builder
    context.getRouteDefinitions().get(0).adviceWith(context, new AdviceWithRouteBuilder() {
        @Override
        public void configure() throws Exception {
            // mock only log endpoints
            mockEndpoints("log*");
        }
    });

    // now we can refer to log:foo as a mock and set our expectations
    getMockEndpoint("mock:log:foo").expectedBodiesReceived("Bye World");

    getMockEndpoint("mock:result").expectedBodiesReceived("Bye World");

    template.sendBody("direct:start", "Hello World");

    assertMockEndpointsSatisfied();

    // additional test to ensure correct endpoints in registry
    assertNotNull(context.hasEndpoint("direct:start"));
    assertNotNull(context.hasEndpoint("direct:foo"));
    assertNotNull(context.hasEndpoint("log:foo"));
    assertNotNull(context.hasEndpoint("mock:result"));
    // only the log:foo endpoint was mocked
    assertNotNull(context.hasEndpoint("mock:log:foo"));
    assertNull(context.hasEndpoint("mock:direct:start"));
    assertNull(context.hasEndpoint("mock:direct:foo"));
}
```

支持的模式可以是通配符或正则表达式。请参阅与 Camel 使用的相同匹配功能进行通信的更多详细信息。

注意

请注意，模拟端点会在到达模拟时复制消息。
这意味着 Camel 将使用更多内存。当您发送大量消息时，这可能不适用。

227.8. 使用 CAMEL-TEST 组件模拟现有端点

您可以使用 camel-test Test Kit 时轻松启用此行为，而不必使用 recommendations With 指示 Camel 进行模拟端点。

同一路由可以按照如下所示进行测试：请注意，我们从 isMockEndpoints 方法返回 "*"，它会告知 Camel 模拟所有端点。

如果您只希望 mock 所有 log 端点，您可以返回 "log*"。

isMockEndpoints 使用 camel-test kit

```
public class IsMockEndpointsJUnit4Test extends CamelTestSupport {

    @Override
    public String isMockEndpoints() {
        // override this method and return the pattern for which endpoints to mock.
        // use * to indicate all
        return "*";
    }

    @Test
    public void testMockAllEndpoints() throws Exception {
        // notice we have automatic mocked all endpoints and the name of the endpoints is
"mock:uri"
        getMockEndpoint("mock:direct:start").expectedBodiesReceived("Hello World");
        getMockEndpoint("mock:direct:foo").expectedBodiesReceived("Hello World");
        getMockEndpoint("mock:log:foo").expectedBodiesReceived("Bye World");
        getMockEndpoint("mock:result").expectedBodiesReceived("Bye World");
```

```
        template.sendBody("direct:start", "Hello World");

        assertMockEndpointsSatisfied();

        // additional test to ensure correct endpoints in registry
        assertNotNull(context.hasEndpoint("direct:start"));
        assertNotNull(context.hasEndpoint("direct:foo"));
        assertNotNull(context.hasEndpoint("log:foo"));
        assertNotNull(context.hasEndpoint("mock:result"));
        // all the endpoints was mocked
        assertNotNull(context.hasEndpoint("mock:direct:start"));
        assertNotNull(context.hasEndpoint("mock:direct:foo"));
        assertNotNull(context.hasEndpoint("mock:log:foo"));
    }

    @Override
    protected RouteBuilder createRouteBuilder() throws Exception {
        return new RouteBuilder() {
            @Override
            public void configure() throws Exception {
                from("direct:start").to("direct:foo").to("log:foo").to("mock:result");

                from("direct:foo").transform(constant("Bye World"));
            }
        };
    }
}
```

227.9. 使用 XML DSL 模拟现有端点

如果您不使用 camel-test 组件进行单元测试（如上所示），您可以在将 XML 文件用于路由时使用不同的方法。
解决方案是创建一个由单元测试使用的新 XML 文件，然后包含包含您要测试的路由的预期 XML 文件。

假设我们在 camel-route.xml 文件中有路由：

camel-route.xml 1

```
<!-- this camel route is in the camel-route.xml file -->
<camelContext xmlns="http://camel.apache.org/schema/spring">

    <route>
        <from uri="direct:start"/>
        <to uri="direct:foo"/>
```

```
        <to uri="log:foo"/>
        <to uri="mock:result"/>
    </route>

    <route>
        <from uri="direct:foo"/>
        <transform>
            <constant>Bye World</constant>
        </transform>
    </route>

</camelContext>
```

然后，我们按如下所示创建新的 XML 文件，其中我们包含 camel-route.xml 文件，并使用类 org.apache.camel.impl.InterceptSendToMockEndpointStrategy 定义 spring bean：

test-camel-route.xml

```
<!-- the Camel route is defined in another XML file -->
 <import resource="camel-route.xml"/>

 <!-- bean which enables mocking all endpoints -->
 <bean id="mockAllEndpoints"
class="org.apache.camel.component.mock.InterceptSendToMockEndpointStrategy"/>
```

然后，在单元测试中，您将加载新的 XML 文件(test-camel-route.xml)而不是 camel-route.xml。

要只 mock 所有 Log 端点，您可以在 bean 的构造器中定义模式：

```
<bean id="mockAllEndpoints"
class="org.apache.camel.impl.InterceptSendToMockEndpointStrategy">
    <constructor-arg index="0" value="log*"/>
</bean>
```

227.10. 模拟端点并跳过发送到原始端点

从 Camel 2.10 开始提供

有时您希望轻松模拟并跳过发送到特定端点。因此，消息会降级，并只发送到模拟端点。现在，在 Camel 2.10 中，您可以使用 AdviceWith 或 Test Kit 的 mockEndpointsAndSkip 方法。以下示例将跳过发送到两个端点 "direct:foo" 和 "direct:bar" 的两个端点。

recommendationsWith mock 并跳过发送到端点

```
public void testAdvisedMockEndpointsWithSkip() throws Exception {
    // advice the first route using the inlined AdviceWith route builder
    // which has extended capabilities than the regular route builder
    context.getRouteDefinitions().get(0).adviceWith(context, new AdviceWithRouteBuilder() {
        @Override
        public void configure() throws Exception {
            // mock sending to direct:foo and direct:bar and skip send to it
            mockEndpointsAndSkip("direct:foo", "direct:bar");
        }
    });

    getMockEndpoint("mock:result").expectedBodiesReceived("Hello World");
    getMockEndpoint("mock:direct:foo").expectedMessageCount(1);
    getMockEndpoint("mock:direct:bar").expectedMessageCount(1);

    template.sendBody("direct:start", "Hello World");

    assertMockEndpointsSatisfied();

    // the message was not send to the direct:foo route and thus not sent to the seda endpoint
    SedaEndpoint seda = context.getEndpoint("seda:foo", SedaEndpoint.class);
    assertEquals(0, seda.getCurrentQueueSize());
}
```

使用 Test Kit 的同一示例

isMockEndpointsAndSkip using camel-test kit

```
public class IsMockEndpointsAndSkipJUnit4Test extends CamelTestSupport {

    @Override
    public String isMockEndpointsAndSkip() {
```

```
        // override this method and return the pattern for which endpoints to mock,
        // and skip sending to the original endpoint.
        return "direct:foo";
    }

    @Test
    public void testMockEndpointAndSkip() throws Exception {
        // notice we have automatic mocked the direct:foo endpoints and the name of the
endpoints is "mock:uri"
        getMockEndpoint("mock:result").expectedBodiesReceived("Hello World");
        getMockEndpoint("mock:direct:foo").expectedMessageCount(1);

        template.sendBody("direct:start", "Hello World");

        assertMockEndpointsSatisfied();

        // the message was not send to the direct:foo route and thus not sent to the seda
endpoint
        SedaEndpoint seda = context.getEndpoint("seda:foo", SedaEndpoint.class);
        assertEquals(0, seda.getCurrentQueueSize());
    }

    @Override
    protected RouteBuilder createRouteBuilder() throws Exception {
        return new RouteBuilder() {
            @Override
            public void configure() throws Exception {
                from("direct:start").to("direct:foo").to("mock:result");

                from("direct:foo").transform(constant("Bye World")).to("seda:foo");
            }
        };
    }
}
```

227.11. 限制要保留的消息数量

从 Camel 2.10 开始提供

Mock 端点默认保留它收到的每个交换的副本。因此，如果您使用大量消息进行测试，它将消耗内存。从 Camel 2.10 开始，我们引入了两个选项 retainFirst 和 retainLast，它们可以用来只保留第 N 个和/或最后一个交换。

例如，在下面的代码中，我们只想保留 mock 接收的前 5 个交换副本。

```
MockEndpoint mock = getMockEndpoint("mock:data");
mock.setRetainFirst(5);
mock.setRetainLast(5);
mock.expectedMessageCount(2000);

mock.assertIsSatisfied();
```

使用这存在一些限制。MockEndpoint 上的 getExchanges （） 和 getReceivedExchanges （） 方法仅返回 Exchange 的保留副本。因此，在上面的示例中，列表中包含 10 个交换；前五，最后五。retainFirst 和 retainLast 选项还对可以使用的方法有限制。例如，适用于消息正文、标头等 的预期XXX 方法将仅操作保留的消息。在上例中，他们只能测试 10 个保留的消息中的预期。

227.12. 使用 ARRIVAL 时间进行测试

从 Camel 2.7 开始提供

Mock 端点将消息的 arrival 时间存储为 Exchange 上的属性。

```
Date time = exchange.getProperty(Exchange.RECEIVED_TIMESTAMP, Date.class);
```

您可以使用这些信息来知道消息何时到达模拟。但它还提供了了解之前和下一个消息到达模拟之间的时间间隔的基础。您可以使用它来设置在 Mock 端点上使用 arrives DSL 的预期。

例如，在下一个操作前，第一个消息应到达 0-2 秒：

```
mock.message(0).arrives().noLaterThan(2).seconds().beforeNext();
```

您还可以将其定义为第 2 个消息（基于 0 个索引）在上一个后不再到达 0-2 秒：

```
mock.message(1).arrives().noLaterThan(2).seconds().afterPrevious();
```

您还可以在 中使用 来设置较低绑定。例如，假设它应该在 1-4 秒之间：

```
mock.message(1).arrives().between(1, 4).seconds().afterPrevious();
```

您还可以对所有信息设置预期的设置，例如，它们之间的差距最多 1 秒：

```
mock.allMessages().arrives().noLaterThan(1).seconds().beforeNext();
```

提示

上面示例中的时间单位是秒，但 Camel 还提供 毫秒、分钟。

227.13. 另请参阅

- *Spring Testing*
- *测试*

第 228 章 MONGODB 组件（已弃用）

从 Camel 版本 2.10 开始提供

维基百科表示："NoSQL 是提升一个松散定义的非关系数据存储类的移动，随着关系数据库的长历史记录和 ACID 保障而破坏"。在过去几年中，NoSQL 解决方案已逐渐普及，主要使用的站点和服务（如 Facebook、LinkedIn、Twanw 等）被广泛使用，以广泛使用它们来实现可扩展性和灵活性。

基本上，NoSQL 解决方案与传统的 RDBMS （关系数据库管理系统）不同，它们不使用 SQL 作为查询语言，通常不提供与类似 ACID 的事务行为或相关数据。相反，它们围绕灵活的数据结构和模式的概念而设计（请注意，具有固定模式的数据库表的传统概念已被丢弃）、商业硬件上的高可扩展性和超快处理。

MongoDB 是一个非常流行的 NoSQL 解决方案，camel-mongodb 组件将 Camel 与 MongoDB 集成，允许您作为生成者（在集合上性能操作）和作为消费者（通过 MongoDB 集合中使用文档）与 MongoDB 集合交互。

MongoDB 围绕文档的概念（而不是办公室文档一样，而是在 JSON/BSON 中定义的分层数据）和集合中定义的数据。此组件页面将假设您熟悉它们。否则，请访问 http://www.mongodb.org/。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-mongodb</artifactId>
  <version>x.y.z</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

228.1. URI 格式

```
mongodb:connectionBean?
database=databaseName&collection=collectionName&operation=operationName[&moreOptio
ns...]
```

228.2. MONGODB 选项

MongoDB 组件没有选项。

MongoDB 端点使用 URI 语法进行配置：

```
mongodb:connectionBean
```

使用以下路径和查询参数：

228.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
connectionBean	要使用的 com.mongodb.Mongo **所需的** 名称。		字符串

228.2.2. 查询参数(23 参数)：

Name	描述	默认值	类型
collection (common)	设置要绑定到此端点的 MongoDB 集合的名称		字符串
collectionIndex (common)	设置集合索引(JSON FORMAT : field1 : order1, field2 : order2)		字符串
createCollection (common)	如果集合不存在，请在初始期间创建集合。默认为 true。	true	布尔值
Database (common)	将 MongoDB 数据库的名称设置为目标		字符串
operation (common)	设置此端点将根据 MongoDB 执行的操作。有关可能的值，请参阅 MongoDbOperation。		MongoDbOperation
outputType (common)	将制作者的输出转换为所选的类型：DBObjectList DBObject 或 DBCursor。DBObjectList 或 DBCursor 适用于 findAll 和 aggregate。DBObject 适用于所有其他操作。		MongoDbOutputType
writeConcern (common)	使用标准的在 MongoDB 上为写入操作设置 WriteConcern。通过调用 WriteConcern#valueOf (String)方法从 WriteConcernBuildvalueOf (String)方法解析。	确认	WriteConcern

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
cursorRegenerationDelay (advanced)	MongoDB tailable cursors 将阻止到新数据到达为止。如果没有插入新数据，则在一些时间后，将由 MongoDB 服务器自动释放和关闭光标。如果需要，客户端会重新生成光标。这个值指定在尝试获取新光标前需要等待的时间，如果尝试失败，则执行下一次尝试前的时长。默认值为 1000ms。	1000	long
动态性(advanced)	设置此端点是否尝试动态解析目标数据库和从传入的 Exchange 属性收集。可用于在运行时覆盖数据库和在其他静态端点 URI 中指定的集合。它默认是禁用的，以提高性能。启用它将需要最小的性能。	false	布尔值
readPreference (advanced)	在 Mongo 连接上设置 MongoDB ReadPreference。直接对连接设置的读取首选项将被此设置覆盖。ReadPreference SlackvalueOf (String)工具方法用于解析传递的 readPreference 值。可能的值的一些示例包括 nearest、primary 或 secondary 等。		ReadPreference
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
writeResultAsHeader (advanced)	在写入操作中，它会决定是否将 WriteResult 返回为 OUT 消息的正文，我们将 IN 消息传送到 OUT，并将 WriteResult 附加为一个标头。	false	布尔值
persistentId (tail)	一个尾部跟踪集合可以托管多个可以被尾部用户的跟踪程序。为了保持它们独立的，每个跟踪器都应该有自己的唯一的 persistentId。		字符串
persistentTailTracking (tail)	启用持久性尾部跟踪，这是一种在系统重启后跟踪最近使用的消息的机制。系统下次启动时，端点会从它最后一次停止的滑动记录中恢复光标。	false	布尔值

Name	描述	默认值	类型
persistRecords (tail)	设置跟踪数据在 MongoDB 中保留的尾部记录数。	-1	int
tailTrackCollection (tail)	在其中保留尾部跟踪信息的集合。如果没有指定，则默认使用 MongoDbTailTrackingConfig#DEFAULT_COLLECTION。		字符串
tailTrackDb (tail)	指明 tail 跟踪机制将保留什么数据库。如果未指定，则默认选择当前数据库。即使启用，动态性也不会考虑，例如，尾部跟踪数据库不会因为端点初始化而不同。		字符串
tailTrackField (tail)	将放置最后一次跟踪值的字段。如果没有指定，则默认使用 MongoDbTailTrackingConfig#DEFAULT_FIELD。		字符串
tailTrackIncreasingField (tail)	传入记录中的关联字段是不断增长的，并且用于在每次生成时都定位尾部光标。光标将(re)创建，其类型为 tailTrackIncreasingField lastValue （可能从持久尾部跟踪中恢复）。可以是 Integer, Date, String 等类型。注意：当前不支持点表示法，因此该字段应处于文档的最高级别。		字符串
tailTrackingStrategy (tail)	设置用于提取 increasing 字段值的策略，并创建查询来定位 tail 光标。	LITERAL	MongoDBTailTracking Enum

228.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.mongodb.enabled	启用 mongodb 组件	true	布尔值
camel.component.mongodb.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

228.4. 在 SPRING XML 中配置数据库

以下 Spring XML 会创建一个定义 MongoDB 实例的连接的 bean。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="mongoBean" class="com.mongodb.Mongo">
        <constructor-arg name="host" value="${mongodb.host}" />
        <constructor-arg name="port" value="${mongodb.port}" />
    </bean>
</beans>
```

228.5. 路由示例

Spring XML 中定义的以下路由对集合执行操作 dbStats。

获取指定集合的 DB stats

```
<route>
  <from uri="direct:start" />
  <!-- using bean 'mongoBean' defined above -->
  <to uri="mongodb:mongoBean?
database=${mongodb.database}&amp;collection=${mongodb.collection}&amp;operation=getDbStats"
/>
  <to uri="direct:result" />
</route>
```

228.6. MONGODB 操作 - 生成者端点

228.6.1. 查询操作

228.6.1.1. findById

此操作仅从其 _id 字段与 IN 消息正文内容匹配的集合中检索一个元素。传入的对象可以是等同于 BSON 类型的任何内容。请参阅 http://bsonspec.org//specification[http://bsonspec.org//specification] 和 http://www.mongodb.org/display/DOCS/Java+Types。

```
from("direct:findById")
    .to("mongodb:myDb?database=flights&collection=tickets&operation=findById")
    .to("mock:resultFindById");
```

提示

支持可选参数。此操作支持指定字段过滤器。请参阅 指定可选参数。

228.6.1.2. findOneByQuery

使用此操作只从与 MongoDB 查询匹配的集合中检索一个元素。查询对象从 IN 消息正文中提取，即它应该类型为 DBObject 或转换为 DBObject。它可以是 JSON 字符串或 Hashmap。如需更多信息，请参阅 #Type conversions。

没有查询的示例（返回集合的任何对象）：

```
from("direct:findOneByQuery")
  .to("mongodb:myDb?database=flights&collection=tickets&operation=findOneByQuery")
  .to("mock:resultFindOneByQuery");
```

带有查询的示例（返回一个匹配结果）：

```
from("direct:findOneByQuery")
  .setBody().constant("{ \"name\": \"Raul Kripalani\" }")
  .to("mongodb:myDb?database=flights&collection=tickets&operation=findOneByQuery")
  .to("mock:resultFindOneByQuery");
```

提示

支持可选参数。此操作支持指定字段 filter 和/或 sort 子句。请参阅 指定可选参数。

228.6.1.3. findAll

findAll 操作会返回与查询匹配的所有文档，或全部都不匹配，在这种情况下，集合中包含的所有文档都会被返回。查询对象从 IN 消息正文中提取，即它应该类型为 DBObject 或转换为 DBObject。它可以是 JSON 字符串或 Hashmap。如需更多信息，请参阅 #Type conversions。

没有查询的示例（返回集合中的所有对象）：

```
from("direct:findAll")
  .to("mongodb:myDb?database=flights&collection=tickets&operation=findAll")
  .to("mock:resultFindAll");
```

带有查询的示例（返回所有匹配结果）：

```
from("direct:findAll")
    .setBody().constant("{ \"name\": \"Raul Kripalani\" }")
    .to("mongodb:myDb?database=flights&collection=tickets&operation=findAll")
    .to("mock:resultFindAll");
```

通过以下标头支持分页和有效的检索：

标头键	快速常数	描述（从 MongoDB API 文档中提取）	预期类型
CamelMongoDbNumToSkip	**MongoDbConstants.NUM_TO_SKIP**	在光标的开始丢弃指定数量的元素。	int/Integer
CamelMongoDbLimit	**MongoDbConstants.LIMIT**	限制返回的元素数量。	int/Integer
CamelMongoDbBatchSize	**MongoDbConstants.BATCH_SIZE**	限制一个批处理中返回的元素数量。光标通常获取一系列结果对象并将其存储在本地。如果 batchSize 为正数，则代表检索的每个对象	int/Integer

标头键	快速常数	描述（从 MongoDB API 文档中提取）	预期类型
		批处理小。可以是已设置性能和限制数据传输。如果 batchSize 为负数，它将限制返回的数量，该对象适合最大批处理大小限制（通常为 4MB），并且光标将关闭。例如，如果 batchSize 是 -10，则服务器将返回最多 10 个文档，且尽可能多的文档可能适合 4MB，然后关闭光标。请注意，这个功能与文件的 limit（）不同，文档必须适合最大大小，它	

标头键	快速常数	描述（从 MongoDB API 文档中提取）	预期类型
		迭代后，批处理大小也可以更改，在这种情况下，设置将应用到下一个批处理检索。	

您还可以通过将 outputType=DBCursor (Camel 2.16+)作为端点选项，从服务器返回的文档作为端点选项，它可能比设置上述标头更为简单。这从 Mongo 驱动程序证明您的 Exchange the DBCursor，就如同您在 Mongo shell 中执行 findAll （）一样，允许您的路由迭代结果。默认情况下，如果没有这个选项，此组件会将文档从驱动程序的光标加载到列表中，并返回到您的路由 - 可能会导致大量内存对象。请记住，使用 DBCursor 不询问与文档匹配的数量 - 详情请查看 MongoDB 文档站点。

带有选项 outputType=DBCursor 和批处理大小的示例：

```
from("direct:findAll")
   .setHeader(MongoDbConstants.BATCH_SIZE).constant(10)
   .setBody().constant("{ \"name\": \"Raul Kripalani\" }")
   .to("mongodb:myDb?
database=flights&collection=tickets&operation=findAll&outputType=DBCursor")
   .to("mock:resultFindAll");
```

findAll 操作也会返回以下 OUT 标头，以便在使用分页时迭代结果页面：

标头键	快速常数	描述（从 MongoDB API 文档中提取）	数据类型
CamelMongoDbResultTotalSize	**MongoDbConstants.RESULT_TOTAL_SIZE**	与查询匹配的对象数量。这不会考虑限制/跳过。	int/Integer
CamelMongoDbResultPageSize	**MongoDbConstants.RESULT_PAGE_SIZE**	与查询匹配的对象数量。这不会考虑限制/跳过。	int/Integer

提示

支持可选参数。此操作支持指定字段 filter 和/或 sort 子句。请参阅 指定可选参数。

228.6.1.4. æ•°é‡

***返回集合中对象总数，返回 Long 作为 OUT 消息正文。
以下示例将计算"dynamicCollectionName"集合中的记录数。请注意启用动态性的方式，因此操作不会针对"notableScientists"集合运行，而是针对"dynamicCollectionName"集合运行。***

```
// from("direct:count").to("mongodb:myDb?
database=tickets&collection=flights&operation=count&dynamicity=true");
Long result = template.requestBodyAndHeader("direct:count", "irrelevantBody",
MongoDbConstants.COLLECTION, "dynamicCollectionName");
assertTrue("Result is not of type Long", result instanceof Long);
```

从 Camel 2.14 开始，您可以在消息正文中提供 com.mongodb.DBObject 对象作为查询，并且操作将返回与此条件匹配的文档量。

```
DBObject query = ...
Long count = template.requestBodyAndHeader("direct:count", query,
MongoDbConstants.COLLECTION, "dynamicCollectionName");
```

228.6.1.5. 指定字段过滤器（项目）

默认情况下，查询操作将返回整个对象（所有字段）。如果您的文档较大，且您只需要检索其字段的子集，您可以在所有查询操作中指定字段过滤器，只需设置相关的 DBObject （或者类型转换为 DBObject），如 CamelMongoDbFieldsFilter 标头上的 JSON String、Map 等。

以下是一个示例，它使用 MongoDB 的 BasicDBObjectBuilder 来简化 DBObjects 的创建。它检索除 _id 和 boringField 以外的所有字段：

```
// route: from("direct:findAll").to("mongodb:myDb?
database=flights&collection=tickets&operation=findAll")
DBObject fieldFilter = BasicDBObjectBuilder.start().add("_id", 0).add("boringField", 0).get();
Object result = template.requestBodyAndHeader("direct:findAll", (Object) null,
MongoDbConstants.FIELDS_FILTER, fieldFilter);
```

228.6.1.6. 指定 sort 子句

根据特定字段的排序，通常需要从集合中获取 min/max 记录。在 Mongo 中，操作使用类似如下的语法：

```
db.collection.find().sort({_id: -1}).limit(1)
// or
db.collection.findOne({$query:{},$orderby:{_id:-1}})
```

在 Camel 路由中，SORT_BY 标头可与 findOneByQuery 操作一起使用，以达到相同的结果。如果还指定了 FIELDS_FILTER 标头，则操作将返回单个字段/值对，可以直接传递给另一个组件（例如，参数化 MyBatis SELECT 查询）。本例演示了从集合中获取最新的文档，并根据 documentTimestamp 字段将结果减少到单个字段：

```
.from("direct:someTriggeringEvent")
.setHeader(MongoDbConstants.SORT_BY).constant("{\"documentTimestamp\": -1}")
.setHeader(MongoDbConstants.FIELDS_FILTER).constant("{\"documentTimestamp\": 1}")
.setBody().constant("{}")
.to("mongodb:myDb?
database=local&collection=myDemoCollection&operation=findOneByQuery")
.to("direct:aMyBatisParameterizedSelect")
;
```

228.6.2. 创建/更新操作

228.6.2.1. insert

将新对象插入到 MongoDB 集合中，从 IN 消息正文获取。类型转换试图将它转换为 DBObject 或 列表。
支持两种模式：单插入和多个插入。对于多个插入，端点将预期任何类型的对象的 List、Array 或 Collections，只要它们是 - 或可以转换为 DBObject。所有对象都会一次性插入。端点将根据输入智能地决定要调用哪个后端操作（单或多个插入）。

Example:

```
from("direct:insert")
  .to("mongodb:myDb?database=flights&collection=tickets&operation=insert");
```

该操作将返回 WriteResult，并根据 WriteConcern 或 invokeGetLastError 选项的值，getLastError （） 将已被调用。如果您希望访问些操作的最终结果，可以通过在 WriteResult 上调用 getLastError() 或 getCachedLastError() 来获取 CommandResult。然后，您可以通过调用 CommandResult.ok （）、CommandResult.getErrorMessage （） 和/或 CommandResult.getException （） 来验证结果。

请注意，新对象的 _id 在集合中必须是唯一的。如果没有指定值，MongoDB 将自动为您生成一个。但是，如果您确实指定它且不是唯一的，插入操作将失败（并且要注意，您需要启用 invokeGetLastError 或设置等待写结果的 WriteConcern）。

这并不是组件的一个限制，但它是在 MongoDB 中用于更高吞吐量的效果。如果您使用自定义 _id，则需要确保在应用程序级别具有唯一性（这也是良好的做法）。

自 Camel 2.15: 插入记录的 OID 存储在 CamelMongoOid 键下的消息标头中 (MongoDbConstants.OID constant)中。存储的值为 org.bson.types.ObjectId 用于单个插入或 java.util.List<org.bson.types.ObjectId> （如果插入了多个记录）。

228.6.2.2. save

save 操作等同于 upsert (UPdate, inSERT)操作，其中将更新记录，如果记录不存在，它将被插入，所有都在一个原子操作中。MongoDB 将基于 _id 字段执行匹配。

请注意，如果有更新，对象会被完全替换，不允许使用 MongoDB 的 $modifiers。因此，如果要操作对象（如果已存在）,有两个选项：

1. 执行查询，首先检索整个对象及其所有字段（可能效率低下），在 Camel 中更改它，然后

保存它。

2. 使用带有 $modifiers 的 update 操作，这将在服务器端执行更新。您可以启用 upsert 标志，在这种情况下，如果需要一个插入，MongoDB 会将 $modifiers 应用到过滤器查询对象并插入结果。

例如：

```
from("direct:insert")
  .to("mongodb:myDb?database=flights&collection=tickets&operation=save");
```

228.6.2.3. update

更新集合中的一个或多个记录。需要一个 List<DBObject> 作为包含完全 2 元素的 IN 消息正文：

- 元素 1 (index 0) navigator filter query determine 确定哪些对象会受到影响，与典型的查询对象相同
- 元素 2 (index 1) package update rules TOKEN 如何更新匹配的对象。支持 MongoDB 中的所有 修饰符操作。

注意

多更新 .默认情况下，即使多个对象与过滤器查询匹配，MongoDB 也会更新 1 对象。要指示 MongoDB 更新 所有匹配 记录，请将 CamelMongoDbMultiUpdate IN 消息标头设置为 true。

将返回一个带有键 CamelMongoDbRecordsAffected 的标头(Affected) (MongoDbConstants.RECORDS_AFFECTED constant)，带有更新的记录数（从 WriteResult.getN（））。

支持以下 IN message 标头：

标头键	快速常数	描述（从 MongoDB API 文档中提取）	预期类型
CamelMongoDbMultiUpdate	**MongoDbConstants.MULTIUPDATE**	如果更新应该应用到匹配的所有对象。请参阅 http://www.mongodb.org/display/DOCS/Atomic+Operations	boolean/Boolean
CamelMongoDbUpsert	**MongoDbConstants.UPSERT**	如果数据库应创建元素（如果它不存在）	boolean/Boolean

例如，以下命令将 "scientist" 字段的值设置为 "Darwin" 字段来更新其 filterField 字段等于 true 的所有记录：

```
// route: from("direct:update").to("mongodb:myDb?
database=science&collection=notableScientists&operation=update");
DBObject filterField = new BasicDBObject("filterField", true);
DBObject updateObj = new BasicDBObject("$set", new BasicDBObject("scientist", "Darwin"));
Object result = template.requestBodyAndHeader("direct:update", new Object[] {filterField,
updateObj}, MongoDbConstants.MULTIUPDATE, true);
```

228.6.3. 删除操作

228.6.3.1. remove

从集合中删除匹配的记录。IN 消息正文将充当删除过滤器查询，并且应该是 DBObject 类型的类型或可转换的类型。

以下示例将删除其字段 'conditionField' 等于 true 的所有对象，在科学数据库中,ableScientists 集合：

```
// route: from("direct:remove").to("mongodb:myDb?
database=science&collection=notableScientists&operation=remove");
DBObject conditionField = new BasicDBObject("conditionField", true);
Object result = template.requestBody("direct:remove", conditionField);
```

返回一个带有键 CamelMongoDbRecordsAffected 的标头 (MongoDbConstants.RECORDS_AFFECTED constant) with type int, 包括删除记录的数量 (从 WriteResult.getN() 复制)。

228.6.4. 批量写操作

228.6.4.1. bulkWrite

从 Camel 2.21 开始提供

批量执行带有控件的写入操作，以便执行顺序。需要一个 List<WriteModel<DBObject >> 作为 IN 消息正文，其中包含用于插入、更新和删除操作的命令。

以下示例将插入一个新的科学家 "Pierre Curie"，更新 id 为 "5" 的记录，将 "scientist" 项的值设置为 "Marie Curie" 并删除 id 为 "3" 的记录：

```
// route: from("direct:bulkWrite").to("mongodb:myDb?
database=science&collection=notableScientists&operation=bulkWrite");
List<WriteModel<DBObject>> bulkOperations = Arrays.asList(
        new InsertOneModel<>(new BasicDBObject("scientist", "Pierre Curie")),
        new UpdateOneModel<>(new BasicDBObject("_id", "5"),
                new BasicDBObject("$set", new BasicDBObject("scientist", "Marie
Curie"))),
        new DeleteOneModel<>(new BasicDBObject("_id", "3")));

BulkWriteResult result = template.requestBody("direct:bulkWrite", bulkOperations,
BulkWriteResult.class);
```

默认情况下，操作会按顺序执行，并在第一个写入错误时中断操作，而不会处理列表中任何剩余的写操作。要指示 MongoDB 继续处理列表中剩余的写操作，请将 CamelMongoDbBulkOrdered IN message 标头设置为 false。未排序的操作并行执行，无法保证此行为。

标头键	快速常数	描述（从 MongoDB API 文档中提取）	预期类型
CamelMongoDbBulkOrdered	**MongoDbConstants.BULK_ORDERED**	执行有序或未排序的操作执行。默认值为 true。	boolean/Boolean

228.6.5. 其他操作

228.6.5.1. 聚合

从 Camel 2.14 开始提供

使用正文中包含的给定管道执行聚合。聚合可能是长时间和重度的操作。请谨慎使用。

```
// route: from("direct:aggregate").to("mongodb:myDb?
database=science&collection=notableScientists&operation=aggregate");
from("direct:aggregate")
    .setBody().constant("[{ $match : {$or : [{\"scientist\" : \"Darwin\"},{\"scientist\" :
\"Einstein\"}]}},{ $group: { _id: \"$scientist\", count: { $sum: 1 }} } ]")
    .to("mongodb:myDb?
database=science&collection=notableScientists&operation=aggregate")
    .to("mock:resultAggregate");
```

支持以下 IN message 标头：

标头键	快速常数	描述（从 MongoDB API 文档中提取）	预期类型
CamelMongoDbBatchSize	**MongoDbConstants.BATCH_SIZE**	设置每个批处理要返回的文档数量。	int/Integer
CamelMongoDbAllowDiskUse	**MongoDbConstants.ALLOW_DISK_USE**	启用聚合管道阶段，将数据写入临时文件。	boolean/Boolean

通过 outputType=DBCursor 支持有效的检索。

您还可以通过将 outputType=DBCursor (Camel 2.21+)作为端点选项，从服务器返回的文档作为端点选项，它可能比设置上述标头更为简单。这通过 Mongo 驱动程序的 Exchange the DBCursor，就如同您在 Mongo shell 中执行 aggregate （）一样，您的路由可以迭代结果。默认情况下，如果没有这个选项，此组件会将文档从驱动程序的光标加载到列表中，并返回到您的路由 - 可能会导致大量内存对象。请记住，使用 DBCursor 不询问与文档匹配的数量 - 详情请查看 MongoDB 文档站点。

带有选项 outputType=DBCursor 和 batch 大小的示例：

```
// route: from("direct:aggregate").to("mongodb:myDb?
database=science&collection=notableScientists&operation=aggregate");
from("direct:aggregate")
    .setHeader(MongoDbConstants.BATCH_SIZE).constant(10)
    .setBody().constant("[{ $match : {$or : [{\"scientist\" : \"Darwin\"},{\"scientist\" :
\"Einstein\"}]}},{ $group: { _id: \"$scientist\", count: { $sum: 1 }} } ]")
    .to("mongodb:myDb?
database=science&collection=notableScientists&operation=aggregate&outputType=DBCursor")
    .to("mock:resultAggregate");
```

228.6.5.2. getDbStats

等同于在 MongoDB shell 中运行 db.stats （） 命令，该命令显示有关数据库的有用统计图。

例如：

```
> db.stats();
{
    "db" : "test",
    "collections" : 7,
    "objects" : 719,
    "avgObjSize" : 59.73296244784423,
    "dataSize" : 42948,
    "storageSize" : 1000058880,
    "numExtents" : 9,
    "indexes" : 4,
    "indexSize" : 32704,
    "fileSize" : 1275068416,
    "nsSizeMB" : 16,
    "ok" : 1
}
```

使用示例：

```
// from("direct:getDbStats").to("mongodb:myDb?
database=flights&collection=tickets&operation=getDbStats");
Object result = template.requestBody("direct:getDbStats", "irrelevantBody");
assertTrue("Result is not of type DBObject", result instanceof DBObject);
```

此操作将返回与 shell 中显示的数据结构类似的数据结构，格式为 OUT 消息正文中的 DBObject。

228.6.5.3. getColStats

等同于在 MongoDB shell 中运行 db.collection.stats （） 命令，该命令显示有关集合的有用统计图。
例如：

```
> db.camelTest.stats();
{
    "ns" : "test.camelTest",
    "count" : 100,
    "size" : 5792,
    "avgObjSize" : 57.92,
    "storageSize" : 20480,
    "numExtents" : 2,
    "nindexes" : 1,
    "lastExtentSize" : 16384,
    "paddingFactor" : 1,
    "flags" : 1,
    "totalIndexSize" : 8176,
    "indexSizes" : {
        "_id_" : 8176
```

```
    },
    "ok" : 1
}
```

使用示例：

```
// from("direct:getColStats").to("mongodb:myDb?
database=flights&collection=tickets&operation=getColStats");
Object result = template.requestBody("direct:getColStats", "irrelevantBody");
assertTrue("Result is not of type DBObject", result instanceof DBObject);
```

此操作将返回与 shell 中显示的数据结构类似的数据结构，格式为 OUT 消息正文中的 DBObject。

228.6.5.4. 命令

从 Camel 2.15 开始提供

在数据库上运行正文作为命令。Usefull 用于 admin 操作作为获取主机信息、复制或分片状态。

此操作不使用集合参数。

```
// route: from("command").to("mongodb:myDb?database=science&operation=command");
DBObject commandBody = new BasicDBObject("hostInfo", "1");
Object result = template.requestBody("direct:command", commandBody);
```

228.6.6. 动态操作

Exchange 可以通过设置 MongoDbConstants.OPERATION_HEADER 常量所定义的 CamelMongoDbOperation 标头来覆盖端点的固定操作。
支持的值由 MongoDbOperation enumeration 决定，并与端点 URI 上的 operation 参数接受的值匹配。

例如：

```
// from("direct:insert").to("mongodb:myDb?
database=flights&collection=tickets&operation=insert");
Object result = template.requestBodyAndHeader("direct:insert", "irrelevantBody",
MongoDbConstants.OPERATION_HEADER, "count");
assertTrue("Result is not of type Long", result instanceof Long);
```

228.7. TAILABLE CURSOR CONSUMER

MongoDB 提供了一种机制，可以立即使用集合中持续使用持续的数据，方法是保持光标的开放方式，类似于 *nix 系统的 tail -f 命令。这种机制比调度的轮询效率更高，因为服务器将新数据推送到客户端，而不是让客户端按计划的间隔重新执行 ping 来获取新数据。它还可减少其他冗余网络流量。

使用可尾随光标的一个先决条件：集合必须是"总结集合"，即它将只保存 N 对象，当达到限制时，MongoDB 会按照最初插入的顺序清除旧对象。如需更多信息，请参阅：http://www.mongodb.org/display/DOCS/Tailable+Cursors。

Camel MongoDB 组件实施可尾部的光标消费者，使此功能可供您用于 Camel 路由。在插入新对象时，MongoDB 将以自然顺序将其推送为 DBObjects，而后者会将其转换为 Exchange，并触发您的路由逻辑。

228.8. 可尾部的光标消费者的工作方式

要将光标转换为可尾部的光标，首先生成光标时，需要向 MongoDB 发出一些特殊标志。创建后，光标将保持打开状态，并在调用 DBCursor.next （） 方法时阻止，直到新数据到达为止。但是，如果新数据在非确定期之后没有显示新数据，MongoDB 服务器保留自己终止您的光标。如果您希望继续使用新数据，则必须重新生成光标。为此，您必须记住关闭的位置，或者您将再次从顶部开始消耗的位置。

Camel MongoDB tailable 光标消费者会为您处理所有这些任务。您只需要向数据中的一些字段提供密钥，该特性将作为每次重新生成时定位光标的标记，例如时间戳、顺序 ID 等。它可以是 MongoDB 支持的任何数据类型。日期、字符串和整数可以正常工作。我们在此组件上下文中调用这种机制"tail 跟踪"。

消费者将记住此字段的最后一个值，每当要重新生成光标时，它将使用类似过滤器（如 increasingField > lastValue ）运行查询，以便只消耗未读取数据。

设置 increasing 字段： 在端点 URI tailTrackingIncreasingField 选项上设置 increasing 字段的密钥。在 Camel 2.10 中，它必须是数据中的顶级字段，因为尚不支持此字段的嵌套导航。也就是说，"timestamp"字段是 okay，但 "nested.timestamp" 将无法正常工作。如果您需要对嵌套增加字段的支持，请在 Camel JIRA 中创建一个问题单。

光标重新生成延迟： 要注意的一个事情是，如果初期没有新数据，MongoDB 将立即终止光标。由于我们不希望在此情形中覆盖服务器，因此增加了一个 cursorRegenerationDelay 选项（默认值为 1000ms）。

例如：

```
from("mongodb:myDb?
database=flights&collection=cancellations&tailTrackIncreasingField=departureTime")
    .id("tailableCursorConsumer1")
    .autoStartup(false)
    .to("mock:test");
```

以上路由会消耗来自"flights.cancellations"的上限集合，使用"departureTime"作为增加字段，默认重新生成光标延迟 1000ms。

228.9. 持久性尾部跟踪

标准尾部跟踪是易失性的，最后一个值仅保存在内存中。但是，在实践中，您需要每次重启 Camel 容器，但最后您的最后一个值将会丢失，且您的尾部的光标消费者将再次开始使用，这很有可能将重复记录发送到您的路由中。

要克服这种情况，您可以启用 持久性尾部跟踪 功能，以跟踪 MongoDB 数据库内特殊集合中持续使用的值。当消费者再次初始时，它将恢复最后跟踪的值，并在不做任何情况一样继续。

最后的读取值会在两个方面保留：每次重新生成光标以及消费者关闭时。如果需求需求，我们可能会考虑定期持续的间隔（每 5 秒清空一次），以便增加稳健性。要请求此功能，请在 Camel JIRA 中创建一个问题单。

228.10. 启用持久性尾部跟踪

要启用这个功能，请在端点 URI 上至少设置以下选项：

- persistentTailTracking 选项为 true
- persistentId 选项是这个消费者的唯一标识符，因此可以在多个用户间重复使用相同的集合

另外，您可以将 tailTrackDb、tailTrackCollection 和 tailTrackField 选项设置为自定义存储运行时信息的位置。有关每个选项的描述，请参阅本页顶部的端点选项表。

例如，以下路由会消耗来自"flights.cancellations"的上限集合，使用 "departureTime" 作为增加的字段，默认的重新生成光标延迟为 1000ms，并且打开持久性尾部跟踪，并在"flights.camelTailTracking" id 下保留。在 "lastTrackingValue" 字段中存储最后处理的值(camelTail Tracking 和 lastTracking Value 是默认值)。

```
from("mongodb:myDb?
database=flights&collection=cancellations&tailTrackIncreasingField=departureTime&persiste
ntTailTracking=true" +
    "&persistentId=cancellationsTracker")
   .id("tailableCursorConsumer2")
   .autoStartup(false)
   .to("mock:test");
```

以下是与上述示例相同的另一个示例，但持久性尾部跟踪运行时信息将存储在"lastProcessedDepartureTime"字段中的"trackers.camelTrackers"集合中：

```
from("mongodb:myDb?
database=flights&collection=cancellations&tailTrackIncreasingField=departureTime&persiste
ntTailTracking=true" +

"&persistentId=cancellationsTracker&tailTrackDb=trackers&tailTrackCollection=camelTracker
s" +
    "&tailTrackField=lastProcessedDepartureTime")
   .id("tailableCursorConsumer3")
   .autoStartup(false)
   .to("mock:test");
```

228.11. OPLOG TAIL TRACKING

oplog collection 跟踪功能允许在 MongoDB 中实施类似于功能的触发器。要激活此集合，需要首先激活副本集。有关此主题的更多信息，请参阅 https://docs.mongodb.com/manual/tutorial/deploy-replica-set/。

您可以在下面找到一个基于 Java DSL 的路由的示例，它演示了如何使用组件来跟踪 oplog 集合。在这个特定情况下，我们过滤影响数据库 optlog_test 中的集合 客户 的事件。请注意，tailTrackIncreasingField 是一个时间戳字段('ts')，这意味着您必须使用带有 TIMESTAMP 值的 tailTrackingStrategy 参数。

```
import com.mongodb.BasicDBObject;
import com.mongodb.MongoClient;
import org.apache.camel.Exchange;
import org.apache.camel.Message;
import org.apache.camel.Processor;
import org.apache.camel.builder.RouteBuilder;
import org.apache.camel.component.mongodb.MongoDBTailTrackingEnum;
import org.apache.camel.main.Main;

import java.io.InputStream;

/**
 * For this to work you need to turn on the replica set
 * <p>
 * Commands to create a replica set:
```

```
 * <p>
 * rs.initiate( {
 * _id : "rs0",
 * members: [ { _id : 0, host : "localhost:27017" } ]
 * })
 */
public class MongoDbTracker {

    private final String database;

    private final String collection;

    private final String increasingField;

    private MongoDBTailTrackingEnum trackingStrategy;

    private int persistRecords = -1;

    private boolean persistenTailTracking;

    public MongoDbTracker(String database, String collection, String increasingField) {
        this.database = database;
        this.collection = collection;
        this.increasingField = increasingField;
    }

    public static void main(String[] args) throws Exception {
        final MongoDbTracker mongoDbTracker = new MongoDbTracker("local", "oplog.rs", "ts");
        mongoDbTracker.setTrackingStrategy(MongoDBTailTrackingEnum.TIMESTAMP);
        mongoDbTracker.setPersistRecords(5);
        mongoDbTracker.setPersistenTailTracking(true);
        mongoDbTracker.startRouter();
        // run until you terminate the JVM
        System.out.println("Starting Camel. Use ctrl + c to terminate the JVM.\n");

    }

    public void setTrackingStrategy(MongoDBTailTrackingEnum trackingStrategy) {
        this.trackingStrategy = trackingStrategy;
    }

    public void setPersistRecords(int persistRecords) {
        this.persistRecords = persistRecords;
    }

    public void setPersistenTailTracking(boolean persistenTailTracking) {
        this.persistenTailTracking = persistenTailTracking;
    }

    void startRouter() throws Exception {
        // create a Main instance
        Main main = new Main();
        main.bind(MongoConstants.CONN_NAME, new MongoClient("localhost", 27017));
        main.addRouteBuilder(new RouteBuilder() {
            @Override
            public void configure() throws Exception {
```

```
            getContext().getTypeConverterRegistry().addTypeConverter(InputStream.class,
BasicDBObject.class,
                    new MongoToInputStreamConverter());
            from("mongodb://" + MongoConstants.CONN_NAME + "?database=" + database
                    + "&collection=" + collection
                    + "&persistentTailTracking=" + persistenTailTracking
                    + "&persistentId=trackerName" + "&tailTrackDb=local"
                    + "&tailTrackCollection=talendTailTracking"
                    + "&tailTrackField=lastTrackingValue"
                    + "&tailTrackIncreasingField=" + increasingField
                    + "&tailTrackingStrategy=" + trackingStrategy.toString()
                    + "&persistRecords=" + persistRecords
                    + "&cursorRegenerationDelay=1000")
                    .filter().jsonpath("$[?(@.ns=='optlog_test.customers')]")
                    .id("logger")
                    .to("log:logger?level=WARN")
                    .process(new Processor() {
                        public void process(Exchange exchange) throws Exception {
                            Message message = exchange.getIn();
                            System.out.println(message.getBody().toString());
                            exchange.getOut().setBody(message.getBody().toString());
                        }
                    });
        }
    });
    main.run();
  }
}
```

228.12. 类型转换

camel-mongodb 组件中包含的 MongoDbBasicConverters 类型转换器提供以下转换：

Name	从类型	要键入	如何？
fromMapToDBObject	**Map**	**DBObject**	通过新的 **BasicDBObject (Map m)构造器构建新的 BasicDBObject**
fromBasicDBObjectToMap	**BasicDBObject**	**Map**	**BasicDBObject** 已实现 **映射**
fromStringToDBObject	**字符串**	**DBObject**	使用 **com.mongodb.util.JSON.parse (String s)**

Name	从类型	要键入	如何？
fromAnyObjectToDBObject	**对象**	**DBObject**	使用 Jackson 库 将对象转换为 **映射**，后者用于初始化新的 **BasicDBObject**

这个类型转换器被自动发现，因此您不需要手动配置任何内容。

228.13. 另请参阅

- *MongoDB 网站*
- *NoSQL Wikipedia 文章*
- *MongoDB Java driver API 文档 - 当前版本 * 单元测试 以了解更多用法示例*

第 229 章 MONGODB GRIDFS COMPONENT

从 Camel 版本 2.18 开始提供

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-mongodb-gridfs</artifactId>
  <version>x.y.z</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

229.1. URI 格式

```
mongodb-gridfs:connectionBean?database=databaseName&bucket=bucketName[&moreOptions...]
```

229.2. MONGODB GRIDFS 选项

MongoDB GridFS 组件没有选项。

MongoDB GridFS 端点使用 URI 语法进行配置：

```
mongodb-gridfs:connectionBean
```

使用以下路径和查询参数：

229.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
connectionBean	要使用的 com.mongodb.Mongo 所需的 名称。		字符串

229.2.2. 查询参数(17 参数)：

Name	描述	默认值	类型
bucket (common)	设置数据库中的 GridFS 存储桶的名称。默认为 fs。	fs	字符串

Name	描述	默认值	类型
Database (common)	将 MongoDB 数据库的名称设置为目标		字符串
readPreference (common)	在 Mongo 连接上设置 MongoDB ReadPreference。直接对连接设置的读取首选项将被此设置覆盖。com.mongodb.ReadPreference SerialvalueOf (String)工具方法用于解析传递的 readPreference 值。可能的值的一些示例包括 nearest、primary 或 secondary 等。		ReadPreference
writeConcern (common)	使用标准的在 MongoDB 上为写入操作设置 WriteConcern。通过调用 WriteConcern#valueOf (String)方法从 WriteConcernBuildvalueOf (String)方法解析。		WriteConcern
writeConcernRef (common)	为 MongoDB 上的写入操作设置 WriteConcern，传递 bean ref 到 Registry 中存在的自定义 WriteConcern。您还可以通过传递其密钥来使用标准 WriteConcerns。请参阅 link #setWriteConcern (String) setWriteConcern 方法。		WriteConcern
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
delay (consumer)	在 Consumer 中设置轮询之间的延迟。默认值为 500ms	500	long
fileAttributeName (consumer)	如果 QueryType 使用 FileAttribute，这会设置使用的属性的名称。默认为 camel-processed。	camel-processed	字符串
initialDelay (consumer)	在消费者开始轮询前设置 initialDelay。默认值为 1000ms	1000	long
persistentTSCollection (consumer)	如果 QueryType 使用持久时间戳，这会设置 DB 中的集合名称来存储时间戳。	camel-timestamps	字符串
persistentTSObject (consumer)	如果 QueryType 使用持久时间戳，这是集合中对象的 ID 来存储时间戳。	camel-timestamp	字符串

Name	描述	默认值	类型
query (consumer)	用于配置用于查找 GridFsConsumer 中文件的查询的其他查询参数（在 JSON 中）		字符串
queryStrategy (consumer)	设置用于轮询新文件的 QueryStrategy。默认为 Timestamp	TimeStamp	QueryStrategy
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
operation (producer)	设置此端点将根据 GridRS 执行的操作。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

229.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.mongodb-gridfs.enabled	启用 mongodb-gridfs 组件	true	布尔值
camel.component.mongodb-gridfs.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

229.4. 在 SPRING XML 中配置数据库

以下 Spring XML 会创建一个定义 MongoDB 实例的连接的 bean。

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
```

```
http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="mongoBean" class="com.mongodb.Mongo">
        <constructor-arg name="host" value="${mongodb.host}" />
        <constructor-arg name="port" value="${mongodb.port}" />
    </bean>
</beans>
```

229.5. 路由示例

Spring XML 中定义的以下路由在集合上执行操作 findOne。

从 GridFS 获取文件

```
<route>
  <from uri="direct:start" />
  <!-- using bean 'mongoBean' defined above -->
  <to uri="mongodb-gridfs:mongoBean?database=${mongodb.database}&amp;operation=findOne" />
  <to uri="direct:result" />
</route>
```

229.6. GRIDFS 操作 - 制作者端点

229.6.1. æ•°é‡

返回集合中文件总数，返回 Integer 作为 OUT 消息正文。

```
// from("direct:count").to("mongodb-gridfs?database=tickets&operation=count");
Integer result = template.requestBodyAndHeader("direct:count", "irrelevantBody");
assertTrue("Result is not of type Long", result instanceof Integer);
```

您可以提供一个文件名标头，以提供与文件名匹配的文件计数。

```
Map<String, Object> headers = new HashMap<String, Object>();
headers.put(Exchange.FILE_NAME, "filename.txt");
Integer count = template.requestBodyAndHeaders("direct:count", query, headers);
```

229.6.2. listAll

返回 Reader，它将在以标签页分开的流中列出所有文件名及其 ID。

```
// from("direct:listAll").to("mongodb-gridfs?database=tickets&operation=listAll");
```

```
Reader result = template.requestBodyAndHeader("direct:listAll", "irrelevantBody");

filename1.txt   1252314321
filename2.txt   2897651254
```

229.6.3. findOne

在 GridFS 系统中查找文件，并将正文设置为内容的 InputStream。还提供元数据的标头。它使用传入标头中的 Exchange.FILE_NAME 来确定要查找的文件。

```
// from("direct:findOne").to("mongodb-gridfs?database=tickets&operation=findOne");
Map<String, Object> headers = new HashMap<String, Object>();
headers.put(Exchange.FILE_NAME, "filename.txt");
InputStream result = template.requestBodyAndHeaders("direct:findOne", "irrelevantBody",
headers);
```

229.6.4. create

在 GridFs 数据库中创建新文件。它使用传入标头中的 Exchange.FILE_NAME 作为名称，正文内容（作为 InputStream）作为内容。

```
// from("direct:create").to("mongodb-gridfs?database=tickets&operation=create");
Map<String, Object> headers = new HashMap<String, Object>();
headers.put(Exchange.FILE_NAME, "filename.txt");
InputStream stream = ... the data for the file ...
template.requestBodyAndHeaders("direct:create", stream, headers);
```

229.6.5. remove

从 GridFS 数据库删除文件。

```
// from("direct:remove").to("mongodb-gridfs?database=tickets&operation=remove");
Map<String, Object> headers = new HashMap<String, Object>();
headers.put(Exchange.FILE_NAME, "filename.txt");
template.requestBodyAndHeaders("direct:remove", "", headers);
```

229.7. GRIDFS CONSUMER

另请参阅

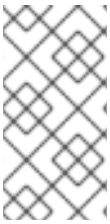
- MongoDB 网站

- *NoSQL Wikipedia 文章*

- *MongoDB Java driver API 文档 - 当前版本* * *单元测试* *以了解更多用法示例*

第 230 章 MONGODB COMPONENT

从 Camel 版本 2.19 开始提供



注意

Camel MongoDB3 组件将 Mongo 驱动程序用于 Java 3.4。如果您使用以前的版本，请使用 Camel MongoDB 组件。

维基百科表示："NoSQL 是提升一个松散定义的非关系数据存储类的移动，随着关系数据库的长历史记录和 ACID 保障而破坏"。在过去几年中，NoSQL 解决方案已逐渐普及，主要使用的站点和服务（如 Facebook、LinkedIn、Twanw 等）被广泛使用，以广泛使用它们来实现可扩展性和灵活性。

基本上，NoSQL 解决方案与传统的 RDBMS （关系数据库管理系统）不同，它们不使用 SQL 作为查询语言，通常不提供与类似 ACID 的事务行为或相关数据。相反，它们围绕灵活的数据结构和模式的概念而设计（请注意，具有固定模式的数据库表的传统概念已被丢弃）、商业硬件上的高可扩展性和超快处理。

MongoDB 是一个非常流行的 NoSQL 解决方案，camel-mongodb 组件将 Camel 与 MongoDB 集成，允许您作为生成者（在集合上性能操作）和作为消费者（通过 MongoDB 集合中使用文档）与 MongoDB 集合交互。

MongoDB 围绕文档的概念（而不是办公室文档一样，而是在 JSON/BSON 中定义的分层数据）和集合中定义的数据。此组件页面将假设您熟悉它们。否则，请访问 http://www.mongodb.org/。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-mongodb3</artifactId>
  <version>x.y.z</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

230.1. URI 格式

```
mongodb3:connectionBean?
database=databaseName&collection=collectionName&operation=operationName[&moreOptio
ns...]
```

230.2. MONGODB 选项

MongoDB 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
mongoConnection (common)	设置用于连接的客户端：从组件生成的所有端点都将共享此连接 bean。		MongoClient
basicPropertyBinding (advanced)	组件是否应该使用基本属性绑定(Camel 2.x)或较新的属性绑定	false	布尔值
lazyStartProducer (producer)	生成者是否应懒惰启动 （在第一个消息中）。通过懒惰启动，您可以使用此选项来允许 CamelContext 和路由在生成者启动期间启动，并导致路由启动失败。通过懒惰启动，启动失败可以在路由信息时通过 Camel 的路由错误处理程序进行处理。请注意，在处理第一个消息时，创建并启动生成者可能需要稍等时间，并延长处理的总处理时间。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值

MongoDB 端点使用 URI 语法进行配置：

```
mongodb3:connectionBean
```

使用以下路径和查询参数：

230.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
connectionBean	要使用的 com.mongodb.Mongo 所需的 名称。		字符串

230.2.2. 查询参数(26 参数)：

Name	描述	默认值	类型
collection (common)	设置要绑定到此端点的 MongoDB 集合的名称		字符串
collectionIndex (common)	设置集合索引(JSON FORMAT : { field1 : order1, field2 : order2})		字符串
createCollection (common)	如果集合不存在，请在初始期间创建集合。默认为 true。	true	布尔值
Database (common)	将 MongoDB 数据库的名称设置为目标		字符串
mongoConnection (common)	设置代表后备连接的 Mongo 实例		MongoClient
operation (common)	设置此端点将根据 MongoDB 执行的操作。		MongoDbOperation
outputType (common)	将制作者的输出转换为所选的类型：DocumentList Document 或 MongoIterable。DocumentList 或 MongoIterable 适用于 findAll 和 aggregate。文档适用于所有其他操作。		MongoDbOutputType
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
consumerType (consumer)	消费者类型。		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
lazyStartProducer (producer)	生成者是否应懒惰启动 （在第一个消息中）。通过懒惰启动，您可以使用此选项来允许 CamelContext 和路由在生成者启动期间启动，并导致路由启动失败。通过懒惰启动，启动失败可以在路由信息时通过 Camel 的路由错误处理程序进行处理。请注意，在处理第一个消息时，创建并启动生成者可能需要稍等时间，并延长处理的总处理时间。	false	布尔值

Name	描述	默认值	类型
basicPropertyBinding (advanced)	端点是否应该使用基本属性绑定(Camel 2.x)或较新的属性绑定	false	布尔值
cursorRegenerationDelay (advanced)	MongoDB tailable cursors 将阻止到新数据到达为止。如果没有插入新数据，则在一些时间后，将由 MongoDB 服务器自动释放和关闭光标。如果需要，客户端会重新生成光标。这个值指定在尝试获取新光标前需要等待的时间，如果尝试失败，则执行下一次尝试前的时长。默认值为 1000ms。	1000	long
动态性(advanced)	设置此端点是否尝试动态解析目标数据库和从传入的 Exchange 属性收集。可用于在运行时覆盖数据库和在其他静态端点 URI 中指定的集合。它默认是禁用的，以提高性能。启用它将需要最小的性能。	false	布尔值
readPreference (advanced)	配置 MongoDB 客户端如何将读取操作路由到副本集的成员。可能的值有 PRIMARY, PRIMARY_PREFERRED, SECONDARY, SECONDARY_PREFERRED 或 NEAREST	主	字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
writeConcern (advanced)	使用 MongoDB 请求的确认级别配置为一个独立的 mongod、replicaset 或 cluster，配置连接 bean。可能的值有 ACKNOWLEDGED, W1, W2, W3, UNACKNOWLEDGED, JOURNALED 或 MAJORITY。	确认	字符串
writeResultAsHeader (advanced)	在写入操作中，它会决定是否将 WriteResult 返回为 OUT 消息的正文，我们将 IN 消息传送到 OUT，并将 WriteResult 附加为一个标头。	false	布尔值
streamFilter (changeStream)	过滤更改流消费者的条件。		字符串
persistentId (tail)	一个尾部跟踪集合可以托管多个可以被尾部用户的跟踪程序。为了保持它们独立的，每个跟踪器都应该有自己的唯一的 persistentId。		字符串
persistentTailTracking (tail)	启用持久性尾部跟踪，这是一种在系统重启后跟踪最近使用的消息的机制。系统下次启动时，端点会从它最后一次停止的滑动记录中恢复光标。	false	布尔值

Name	描述	默认值	类型
tailTrackCollection (tail)	在其中保留尾部跟踪信息的集合。如果没有指定，则默认使用 MongoDbTailTrackingConfig#DEFAULT_COLLECTION。		字符串
tailTrackDb (tail)	指明 tail 跟踪机制将保留什么数据库。如果未指定，则默认选择当前数据库。即使启用，动态性也不会考虑，例如，尾部跟踪数据库不会因为端点初始化而不同。		字符串
tailTrackField (tail)	将放置最后一次跟踪值的字段。如果没有指定，则默认使用 MongoDbTailTrackingConfig#DEFAULT_FIELD。		字符串
tailTrackIncreasingField (tail)	传入记录中的关联字段是不断增长的，并且用于在每次生成时都定位尾部光标。光标将(re)创建，其类型为 tailTrackIncreasingField 大于 lastValue （可能从持久尾部跟踪中恢复）。可以是 Integer, Date, String 等类型。注意：当前不支持点表示法，因此该字段应处于文档的最高级别。		字符串

230.3. SPRING BOOT AUTO-CONFIGURATION

当使用 Spring Boot 时，请确保使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-mongodb3-starter</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.component.mongodb3.basic-property-binding	组件是否应该使用基本属性绑定(Camel 2.x)或较新的属性绑定	false	布尔值

Name	描述	默认值	类型
camel.component.mongodb3.bridge-error-handler	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
camel.component.mongodb3.enabled	是否启用 mongodb 组件的自动配置。这默认是启用的。		布尔值
camel.component.mongodb3.lazy-start-producer	生成者是否应懒惰启动 （在第一个消息中）。通过懒惰启动，您可以使用此选项来允许 CamelContext 和路由在生成者启动期间启动，并导致路由启动失败。通过懒惰启动，启动失败可以在路由信息时通过 Camel 的路由错误处理程序进行处理。请注意，在处理第一个消息时，创建并启动生成者可能需要稍等时间，并延长处理的总处理时间。	false	布尔值
camel.component.mongodb3.mongo-connection	设置用于连接的客户端：从组件生成的所有端点都将共享此连接 bean。选项是一个 com.mongodb.client.MongoClient 类型。		字符串

230.4. 在 SPRING XML 中配置数据库

以下 Spring XML 会创建一个定义 MongoDB 实例的连接的 bean。

从 mongo java 驱动程序 3 开始， WriteConcern 和 readPreference 选项不可动态修改。它们在 mongoClient 对象中定义

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:mongo="http://www.springframework.org/schema/data/mongo"
xsi:schemaLocation="http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/data/mongo
    http://www.springframework.org/schema/data/mongo/spring-mongo.xsd
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd">

  <mongo:mongo-client id="mongoBean" host="${mongo.url}" port="${mongo.port}"
credentials="${mongo.user}:${mongo.pass}@${mongo.dbname}">
```

```
    <mongo:client-options write-concern="NORMAL" />
  </mongo:mongo-client>
</beans>
```

230.5. 路由示例

Spring XML 中定义的以下路由对集合执行操作 dbStats。

获取指定集合的 DB stats

```
<route>
  <from uri="direct:start" />
  <!-- using bean 'mongoBean' defined above -->
  <to uri="mongodb3:mongoBean?
database=${mongodb.database}&amp;collection=${mongodb.collection}&amp;operation=getDbStats"
/>
  <to uri="direct:result" />
</route>
```

230.6. MONGODB 操作 - 生成者端点

230.6.1. 查询操作

230.6.1.1. findById

此操作仅从其 _id 字段与 IN 消息正文内容匹配的集合中检索一个元素。传入的对象可以是与 Bson 类型等效的任何对象。请参阅 http://bsonspec.org//specification[http://bsonspec.org//specification] 和 http://www.mongodb.org/display/DOCS/Java+Types。

```
from("direct:findById")
    .to("mongodb3:myDb?database=flights&collection=tickets&operation=findById")
    .to("mock:resultFindById");
```

提示

支持可选参数。此操作支持指定字段过滤器。请参阅 指定可选参数。

230.6.1.2. findOneByQuery

使用此操作从与 MongoDB 查询匹配的集合中仅检索一个元素（第一个）。查询对象会被提取 CamelMongoDbCriteria 标头。如果 CamelMongoDbCriteria 标头为空，则查询对象将被提取消息正

文，即 Bson 类型，或者转换为 Bson。它可以是 JSON 字符串或 Hashmap。如需更多信息，请参阅 #Type conversions。您可以使用 MongoDB Driver 中的 Filters 类。

没有查询的示例（返回集合的任何对象）：

```
from("direct:findOneByQuery")
   .to("mongodb3:myDb?database=flights&collection=tickets&operation=findOneByQuery")
   .to("mock:resultFindOneByQuery");
```

带有查询的示例（返回一个匹配结果）：

```
from("direct:findOneByQuery")
   .setHeader(MongoDbConstants.CRITERIA, Filters.eq("name", "Raul Kripalani"))
   .to("mongodb3:myDb?database=flights&collection=tickets&operation=findOneByQuery")
   .to("mock:resultFindOneByQuery");
```

提示

支持可选参数。此操作支持指定一个字段 projection 和/或 sort 子句。请参阅 指定可选参数。

230.6.1.3. findAll

findAll 操作会返回与查询匹配的所有文档，或全部都不匹配，在这种情况下，集合中包含的所有文档都会被返回。查询对象会被提取 CamelMongoDbCriteria 标头。如果 CamelMongoDbCriteria 标头为空，则查询对象将被提取消息正文，即 Bson 类型，或者转换为 Bson。它可以是 JSON 字符串或 Hashmap。如需更多信息，请参阅 #Type conversions。

没有查询的示例（返回集合中的所有对象）：

```
from("direct:findAll")
   .to("mongodb3:myDb?database=flights&collection=tickets&operation=findAll")
   .to("mock:resultFindAll");
```

带有查询的示例（返回所有匹配结果）：

```
from("direct:findAll")
   .setHeader(MongoDbConstants.CRITERIA, Filters.eq("name", "Raul Kripalani"))
   .to("mongodb3:myDb?database=flights&collection=tickets&operation=findAll")
   .to("mock:resultFindAll");
```

通过以下标头支持分页和有效的检索：

标头键	快速常数	描述（从 MongoDB API 文档中提取）	预期类型
CamelMongoDbNumToSkip	**MongoDbConstants.NUM_TO_SKIP**	在光标的开始丢弃指定数量的元素。	int/Integer
CamelMongoDbLimit	**MongoDbConstants.LIMIT**	限制返回的元素数量。	int/Integer
CamelMongoDbBatchSize	**MongoDbConstants.BATCH_SIZE**	限制一个批处理中返回的元素数量。光标通常获取一系列结果对象并将其存储在本地。如果 batchSize 为正数，则代表检索的每个对象批处理的大小。可以调整它以优化性能和限制数据传输。如果 batchSize 为负数，它将限制返回的数量，该对象适合最大批处理大小限制（通常为 4MB），并且光标将关闭。例如，如果 batchSize 是 -10，则服务器将返回最多 10 个文档，且尽可能多的文档可能适合 4MB，然后关闭光标。请注意，这个功能与文件的 limit （）不同，文档必须适合最大大小，它会删除发送请求以关闭光标服务器端。即使光标迭代后，批处理大小也可以更改，在这种情况下，设置将应用到下一个批处理检索。	int/Integer

带有选项 outputType=MongoIterable 和 batch 大小的示例：

```
from("direct:findAll")
   .setHeader(MongoDbConstants.BATCH_SIZE).constant(10)
   .setHeader(MongoDbConstants.CRITERIA, Filters.eq("name", "Raul Kripalani"))
   .to("mongodb3:myDb?
database=flights&collection=tickets&operation=findAll&outputType=MongoIterable")
   .to("mock:resultFindAll");
```

findAll 操作也会返回以下 OUT 标头，以便在使用分页时迭代结果页面：

标头键	快速常数	描述（从 MongoDB API 文档中提取）	数据类型
CamelMongoDbResultTotalSize	**MongoDbConstants.RESULT_TOTAL_SIZE**	与查询匹配的对象数量。这不会考虑限制/跳过。	int/Integer
CamelMongoDbResultPageSize	**MongoDbConstants.RESULT_PAGE_SIZE**	与查询匹配的对象数量。这不会考虑限制/跳过。	int/Integer

提示

支持可选参数。此操作支持指定一个字段 projection 和/或 sort 子句。请参阅 指定可选参数。

230.6.1.4. æ•°é‡

返回集合中对象总数，返回 Long 作为 OUT 消息正文。
以下示例将计算"dynamicCollectionName"集合中的记录数。请注意启用动态性的方式，因此操作不会针对"notableScientists"集合运行，而是针对"dynamicCollectionName"集合运行。

```
// from("direct:count").to("mongodb3:myDb?
database=tickets&collection=flights&operation=count&dynamicity=true");
Long result = template.requestBodyAndHeader("direct:count", "irrelevantBody",
MongoDbConstants.COLLECTION, "dynamicCollectionName");
assertTrue("Result is not of type Long", result instanceof Long);
```

您可以提供一个查询。查询 对象会被提取 CamelMongoDbCriteria 标头。如果 CamelMongoDbCriteria 标头为 null，则查询对象会被提取消息正文，即它的类型为 Bson，或者转换为 Bson。操作将返回与此条件匹配的文档数量。

```
Document query = ...
Long count = template.requestBodyAndHeader("direct:count", query,
MongoDbConstants.COLLECTION, "dynamicCollectionName");
```

230.6.1.5. 指定字段过滤器（项目）

默认情况下，查询操作将返回整个对象（所有字段）。如果您的文档较大，且您只需要检索其字段的子集，您可以在所有查询操作中指定字段过滤器，只需设置相关的 Bson （或类型转换为 Bson，如 JSON String、Map 等）。

以下是一个示例，它使用 MongoDB 的 Projections 来简化 Bson 的创建。它检索除 _id 和 boringField 以外的所有字段：

```
// route: from("direct:findAll").to("mongodb3:myDb?
database=flights&collection=tickets&operation=findAll")
Bson fieldProjection = Projection.exclude("_id", "boringField");
Object result = template.requestBodyAndHeader("direct:findAll", ObjectUtils.NULL,
MongoDbConstants.FIELDS_PROJECTION, fieldProjection);
```

以下是一个示例，它使用 MongoDB 的 Projections 来简化 Bson 的创建。它检索除 _id 和 boringField 以外的所有字段：

```
// route: from("direct:findAll").to("mongodb3:myDb?
database=flights&collection=tickets&operation=findAll")
Bson fieldProjection = Projection.exclude("_id", "boringField");
Object result = template.requestBodyAndHeader("direct:findAll", ObjectUtils.NULL,
MongoDbConstants.FIELDS_PROJECTION, fieldProjection);
```

230.6.1.6. 指定 sort 子句

通常，根据特定字段的排序，从集合中获取 min/max 记录（使用 MongoDB 的 排序来简化 Bs on 的创建）。它检索除 _id 和 boringField 以外的所有字段：

```
// route: from("direct:findAll").to("mongodb3:myDb?
database=flights&collection=tickets&operation=findAll")
Bson sorts = Sorts.descending("_id");
Object result = template.requestBodyAndHeader("direct:findAll", ObjectUtils.NULL,
MongoDbConstants.SORT_BY, sorts);
```

在 Camel 路由中，SORT_BY 标头可与 findOneByQuery 操作一起使用，以达到相同的结果。如果还指定了 FIELDS_PROJECTION 标头，则操作将返回可直接传递给另一个组件的单个字段/值对（例如，参数化 MyBatis SELECT 查询）。本例演示了从集合中获取最新的文档，并根据 documentTimestamp 字段将结果减少到单个字段：

```
.from("direct:someTriggeringEvent")
.setHeader(MongoDbConstants.SORT_BY).constant(Sorts.descending("documentTimestamp"
))
.setHeader(MongoDbConstants.FIELDS_PROJECTION).constant(Projection.include("documen
```

```
tTimestamp"))
.setBody().constant("{}")
.to("mongodb3:myDb?
database=local&collection=myDemoCollection&operation=findOneByQuery")
.to("direct:aMyBatisParameterizedSelect")
;
```

230.6.2. 创建/更新操作

230.6.2.1. insert

将新对象插入到 MongoDB 集合中，从 IN 消息正文获取。键入转换尝试将它转换为 文档或 列表。支持两种模式： 单插入和多个插入。对于多个插入，端点将预期任何类型的对象的 List、Array 或 Collections，只要它们是 - 或可以转换为 - 文档。Example:

```
from("direct:insert")
   .to("mongodb3:myDb?database=flights&collection=tickets&operation=insert");
```

该操作将返回 WriteResult，并根据 WriteConcern 或 invokeGetLastError 选项的值，getLastError （） 将已被调用。如果您希望访问些操作的最终结果，可以通过在 WriteResult 上调用 getLastError() 或 getCachedLastError() 来获取 CommandResult。然后，您可以通过调用 CommandResult.ok （）、CommandResult.getErrorMessage （） 和/或 CommandResult.getException （） 来验证结果。

请注意，新对象的 _id 在集合中必须是唯一的。如果没有指定值，MongoDB 将自动为您生成一个。但是，如果您确实指定它且不是唯一的，插入操作将失败（并且要注意，您需要启用 invokeGetLastError 或设置等待写结果的 WriteConcern）。

这并不是组件的一个限制，但它是在 MongoDB 中用于更高吞吐量的效果。如果您使用自定义 _id，则需要确保在应用程序级别具有唯一性（这也是良好的做法）。

插入记录的 OID 存储在 CamelMongoOid 键下的消息标头中(MongoDbConstants.OID constant)。存储的值为 org.bson.types.ObjectId 用于单个插入或 java.util.List<org.bson.types.ObjectId> （如果插入了多个记录）。

在 MongoDB Java Driver 3.x 中，insertOne 和 insertMany 操作返回 void。Camel insert 操作返回插入的 Document 或 Documents 列表。请注意，每个文档都会被新的 OID 更新（如果需要）。

230.6.2.2. save

save 操作等同于 upsert (UPdate, inSERT)操作，其中将更新记录，如果记录不存在，它将被插入，

所有都在一个原子操作中。MongoDB 将基于 _id 字段执行匹配。

请注意，如果有更新，对象会被完全替换，不允许使用 MongoDB 的 $modifiers。因此，如果要操作对象（如果已存在）,有两个选项：

1. 执行查询，首先检索整个对象及其所有字段（可能效率低下），在 Camel 中更改它，然后保存它。

2. 使用带有 $modifiers 的 update 操作，这将在服务器端执行更新。您可以启用 upsert 标志，在这种情况下，如果需要一个插入，MongoDB 会将 $modifiers 应用到过滤器查询对象并插入结果。

如果要保存的文档不包含 _id 属性，则操作将是插入的，则创建新的 _id 将放置在 CamelMongoOid 标头中。

例如：

```
from("direct:insert")
   .to("mongodb3:myDb?database=flights&collection=tickets&operation=save");
```

230.6.2.3. update

更新集合中的一个或多个记录。需要过滤器查询和更新规则。

您可以使用 MongoDBConstants.CRITERIA 标头定义为 Bson，并将更新规则定义为 Body 中的 Bson。

注意

在 功能丰富后更新。虽然通过使用 MongoDBConstants.CRITERIA 标头将 MongoDBConstants.CRITERIA 标头定义为 Bson 在进行更新前查询 mongodb 的过滤器时，您应该会看到在聚合过程中将其从生成的 camel 交换中删除，然后应用 mongodb 更新。如果在聚合和/或将 camel Exchange 发送到 mongodb producer 端点前，在聚合和/或重新定义 MongoDBConstants.CRITERIA 标头的过程中没有删除此标头，则在更新 mongodb 时可能会最终使用无效的 camel Exchange payload。

第二种方法 Require a List<Bson> 作为 IN message body （包括正好 2 个元素）：

- *元素 1 (index 0) navigator filter query determine 确定哪些对象会受到影响，与典型的查询对象相同*
- *元素 2 (index 1) package update rules TOKEN 如何更新匹配的对象。支持 MongoDB 中的所有 修饰符操作。*

注意

多更新 .默认情况下，即使多个对象与过滤器查询匹配，MongoDB 也会更新 1 对象。要指示 MongoDB 更新 所有匹配 记录，请将 CamelMongoDbMultiUpdate IN 消息标头设置为 true。

将返回一个带有键 CamelMongoDbRecordsAffected 的标头(Affected) (MongoDbConstants.RECORDS_AFFECTED constant)，带有更新的记录数（从 WriteResult.getN（））。

支持以下 IN message 标头：

标头键	快速常数	描述（从 MongoDB API 文档中提取）	预期类型

标头键	快速常数	描述（从 MongoDB API 文档中提取）	预期类型
CamelMongoDbMultiUpdate	**MongoDbConstants.MULTIUPDATE**	如果更新应该应用到匹配的所有对象。请参阅 http://www.mongodb.org/display/DOCS/Atomic+Operations	boolean/Boolean
CamelMongoDbUpsert	**MongoDbConstants.UPSERT**	如果数据库应创建元素（如果它不存在）	boolean/Boolean

例如，以下命令将 "scientist" 字段的值设置为 "Darwin" 字段来更新其 filterField 字段等于 true 的所有记录：

```
// route: from("direct:update").to("mongodb3:myDb?
database=science&collection=notableScientists&operation=update");
Bson filterField = Filters.eq("filterField", true);
String updateObj = Updates.set("scientist", "Darwin");
Object result = template.requestBodyAndHeader("direct:update", new Bson[] {filterField,
Document.parse(updateObj)}, MongoDbConstants.MULTIUPDATE, true);
```

```
// route: from("direct:update").to("mongodb3:myDb?
database=science&collection=notableScientists&operation=update");
Maps<String, Object> headers = new HashMap<>(2);
headers.add(MongoDbConstants.MULTIUPDATE, true);
headers.add(MongoDbConstants.FIELDS_FILTER, Filters.eq("filterField", true));
String updateObj = Updates.set("scientist", "Darwin");;
Object result = template.requestBodyAndHeaders("direct:update", updateObj, headers);
```

```
// route: from("direct:update").to("mongodb3:myDb?
database=science&collection=notableScientists&operation=update");
String updateObj = "[{\"filterField\": true}, {\"$set\", {\"scientist\", \"Darwin\"}}]";
Object result = template.requestBodyAndHeader("direct:update", updateObj,
MongoDbConstants.MULTIUPDATE, true);
```

230.6.3. 删除操作

230.6.3.1. remove

从集合中删除匹配的记录。IN 消息正文将充当删除过滤器查询，并且应该是 DBObject 类型的类型或可转换的类型。
以下示例将删除其字段 'conditionField' 等于 true 的所有对象，在科学数据库中,ableScientists 集合：

```
// route: from("direct:remove").to("mongodb3:myDb?
database=science&collection=notableScientists&operation=remove");
Bson conditionField = Filters.eq("conditionField", true);
Object result = template.requestBody("direct:remove", conditionField);
```

返回一个带有键 CamelMongoDbRecordsAffected 的标头
(MongoDbConstants.RECORDS_AFFECTED constant) with type int, 包括删除记录的数量 (从 WriteResult.getN() 复制)。

230.6.4. 批量写操作

230.6.4.1. bulkWrite

从 Camel 2.21 开始提供

批量执行带有控件的写入操作，以便执行顺序。需要一个 List<WriteModel<Document >> 作为 IN 消息正文，其中包含用于插入、更新和删除操作的命令。

以下示例将插入一个新的科学家 "Pierre Curie"，更新 id 为 "5" 的记录，将 "scientist" 项的值设置为 "Marie Curie" 并删除 id 为 "3" 的记录：

```
// route: from("direct:bulkWrite").to("mongodb:myDb?
database=science&collection=notableScientists&operation=bulkWrite");
List<WriteModel<Document>> bulkOperations = Arrays.asList(
        new InsertOneModel<>(new Document("scientist", "Pierre Curie")),
        new UpdateOneModel<>(new Document("_id", "5"),
                new Document("$set", new Document("scientist", "Marie Curie"))),
        new DeleteOneModel<>(new Document("_id", "3")));
```

```
BulkWriteResult result = template.requestBody("direct:bulkWrite", bulkOperations,
BulkWriteResult.class);
```

默认情况下，操作会按顺序执行，并在第一个写入错误时中断操作，而不会处理列表中任何剩余的写操作。要指示 MongoDB 继续处理列表中剩余的写操作，请将 CamelMongoDbBulkOrdered IN message 标头设置为 false。未排序的操作并行执行，无法保证此行为。

标头键	快速常数	描述（从 MongoDB API 文档中提取）	预期类型
CamelMongoDbBulkOrdered	MongoDbConstants.BULK_ORDERED	执行有序或未排序的操作执行。默认值为 true。	boolean/Boolean

230.6.5. 其他操作

230.6.5.1. 聚合

使用正文中包含的给定管道执行聚合。聚合可能是长时间和重度的操作。请谨慎使用。

```
// route: from("direct:aggregate").to("mongodb3:myDb?
database=science&collection=notableScientists&operation=aggregate");
List<Bson> aggregate = Arrays.asList(match(or(eq("scientist", "Darwin"), eq("scientist",
      group("$scientist", sum("count", 1)));
from("direct:aggregate")
   .setBody().constant(aggregate)
   .to("mongodb3:myDb?
database=science&collection=notableScientists&operation=aggregate")
   .to("mock:resultAggregate");
```

支持以下 IN message 标头：

标头键	快速常数	描述（从 MongoDB API 文档中提取）	预期类型
CamelMongoDbBatchSize	**MongoDbConstants.BATCH_SIZE**	设置每个批处理要返回的文档数量。	int/Integer
CamelMongoDbAllowDiskUse	**MongoDbConstants.ALLOW_DISK_USE**	启用聚合管道阶段，将数据写入临时文件。	boolean/Boolean

默认情况下，返回所有结果的列表。根据结果的大小，这可能会对内存进行重量。更安全的替代方案是设置 outputType=MongoIterable。下一个处理器将在消息正文中看到一个不可取的，允许它逐一完成结果。因此，设置批处理大小并返回可使其可以有效地检索和处理结果。

您还可以通过将 outputType=DBCursor (Camel 2.21+)作为端点选项，从服务器返回的文档作为端点选项，它可能比设置上述标头更为简单。这通过 Mongo 驱动程序的 Exchange the DBCursor，就如同您在 Mongo shell 中执行 aggregate （）一样，您的路由可以迭代结果。默认情况下，如果没有这个选项，此组件会将文档从驱动程序的光标加载到列表中，并返回到您的路由 - 可能会导致大量内存对象。请记住，使用 DBCursor 不询问与文档匹配的数量 - 详情请查看 MongoDB 文档站点。

*带有选项 **outputType=MongoIterable 和 batch 大小的示例：***

```
// route: from("direct:aggregate").to("mongodb3:myDb?
database=science&collection=notableScientists&operation=aggregate&outputType=MongoIterable");
List<Bson> aggregate = Arrays.asList(match(or(eq("scientist", "Darwin"), eq("scientist",
      group("$scientist", sum("count", 1))));
from("direct:aggregate")
   .setHeader(MongoDbConstants.BATCH_SIZE).constant(10)
   .setBody().constant(aggregate)
   .to("mongodb3:myDb?
database=science&collection=notableScientists&operation=aggregate&outputType=MongoIterable")
   .split(body())
   .streaming()
   .to("mock:resultAggregate");
```

-

请注意，调用 .split (body ()) 足以逐一发送条目，但仍会首先将所有条目加载到内存中。因此，需要调用 .streaming () 来通过批处理将数据加载到内存中。

230.6.5.2. getDbStats

等同于在 MongoDB shell 中运行 db.stats () 命令，该命令显示有关数据库的有用统计图。
例如：

```
> db.stats();
{
    "db" : "test",
    "collections" : 7,
    "objects" : 719,
    "avgObjSize" : 59.73296244784423,
    "dataSize" : 42948,
    "storageSize" : 1000058880,
    "numExtents" : 9,
    "indexes" : 4,
    "indexSize" : 32704,
    "fileSize" : 1275068416,
    "nsSizeMB" : 16,
    "ok" : 1
}
```

使用示例：

```
// from("direct:getDbStats").to("mongodb3:myDb?
database=flights&collection=tickets&operation=getDbStats");
Object result = template.requestBody("direct:getDbStats", "irrelevantBody");
assertTrue("Result is not of type Document", result instanceof Document);
```

该操作将返回类似于 shell 中显示的数据结构，格式为 OUT 消息正文中的 Document。

230.6.5.3. getColStats

等同于在 MongoDB shell 中运行 db.collection.stats () 命令，该命令显示有关集合的有用统计图。
例如：

```
> db.camelTest.stats();
{
    "ns" : "test.camelTest",
    "count" : 100,
```

```
    "size" : 5792,
    "avgObjSize" : 57.92,
    "storageSize" : 20480,
    "numExtents" : 2,
    "nindexes" : 1,
    "lastExtentSize" : 16384,
    "paddingFactor" : 1,
    "flags" : 1,
    "totalIndexSize" : 8176,
    "indexSizes" : {
        "_id_" : 8176
    },
    "ok" : 1
}
```

使用示例：

```
// from("direct:getColStats").to("mongodb3:myDb?
database=flights&collection=tickets&operation=getColStats");
Object result = template.requestBody("direct:getColStats", "irrelevantBody");
assertTrue("Result is not of type Document", result instanceof Document);
```

该操作将返回类似于 shell 中显示的数据结构，格式为 OUT 消息正文中的 Document。

230.6.5.4. 命令

在数据库上运行正文作为命令。Usefull 用于 admin 操作作为获取主机信息、复制或分片状态。

此操作不使用集合参数。

```
// route: from("command").to("mongodb3:myDb?database=science&operation=command");
DBObject commandBody = new BasicDBObject("hostInfo", "1");
Object result = template.requestBody("direct:command", commandBody);
```

230.6.6. 动态操作

Exchange 可以通过设置 MongoDbConstants.OPERATION_HEADER 常量所定义的 CamelMongoDbOperation 标头来覆盖端点的固定操作。
支持的值由 MongoDbOperation enumeration 决定，并与端点 URI 上的 operation 参数接受的值匹配。

例如：

```
// from("direct:insert").to("mongodb3:myDb?
database=flights&collection=tickets&operation=insert");
Object result = template.requestBodyAndHeader("direct:insert", "irrelevantBody",
MongoDbConstants.OPERATION_HEADER, "count");
assertTrue("Result is not of type Long", result instanceof Long);
```

230.7. 消费者

有几种类型的消费者：

- Tailable Cursor Consumer
- Change Streams Consumer

230.7.1. Tailable Cursor Consumer

MongoDB 提供了一种机制，可以立即使用集合中持续使用持续的数据，方法是保持光标的开放方式，类似于 *nix 系统的 tail -f 命令。这种机制比调度的轮询效率更高，因为服务器将新数据推送到客户端，而不是让客户端按计划的间隔重新执行 ping 来获取新数据。它还可减少其他冗余网络流量。

使用可尾随光标的一个先决条件：集合必须是"总结集合"，即它将只保存 N 对象，当达到限制时，MongoDB 会按照最初插入的顺序清除旧对象。如需更多信息，请参阅：http://www.mongodb.org/display/DOCS/Tailable+Cursors。

Camel MongoDB 组件实施可尾部的光标消费者，使此功能可供您用于 Camel 路由。在插入新对象时，MongoDB 将以自然顺序将其推送为可尾部的光标消费者，后者会将其转换为 Exchange，并触发您的路由逻辑。

230.7.1.1. 可尾部的光标消费者的工作方式

要将光标转换为可尾部的光标，首先生成光标时，需要向 MongoDB 发出一些特殊标志。创建后，光标将保持打开状态，并在调用 MongoCursor.next （） 方法时阻止，直到新数据到达为止。但是，如果新数据在非确定期之后没有显示新数据，MongoDB 服务器保留自己终止您的光标。如果您希望继续使用新数据，则必须重新生成光标。为此，您必须记住关闭的位置，或者您将再次从顶部开始消耗的位置。

Camel MongoDB tailable 光标消费者会为您处理所有这些任务。您只需要向数据中的一些字段提供密钥，该特性将作为每次重新生成时定位光标的标记，例如时间戳、顺序 ID 等。它可以是 MongoDB 支

持的任何数据类型。日期、字符串和整数可以正常工作。我们在此组件上下文中调用这种机制"tail 跟踪"。

消费者将记住此字段的最后一个值，每当要重新生成光标时，它将使用类似过滤器（如 increasingField > lastValue ）运行查询，以便只消耗未读取数据。

设置 increasing 字段： 在端点 URI tailTrackingIncreasingField 选项上设置 increasing 字段的密钥。在 Camel 2.10 中，它必须是数据中的顶级字段，因为尚不支持此字段的嵌套导航。也就是说，"timestamp"字段是 okay，但 "nested.timestamp" 将无法正常工作。如果您需要对嵌套增加字段的支持，请在 Camel JIRA 中创建一个问题单。

光标重新生成延迟： 要注意的一个事情是，如果初期没有新数据，MongoDB 将立即终止光标。由于我们不希望在此情形中覆盖服务器，因此增加了一个 cursorRegenerationDelay 选项（默认值为 1000ms）。

例如：

```
from("mongodb3:myDb?
database=flights&collection=cancellations&tailTrackIncreasingField=departureTime")
    .id("tailableCursorConsumer1")
    .autoStartup(false)
    .to("mock:test");
```

以上路由会消耗来自"flights.cancellations"的上限集合，使用"departureTime"作为增加字段，默认重新生成光标延迟 1000ms。

230.7.1.2. 持久性尾部跟踪

标准尾部跟踪是易失性的，最后一个值仅保存在内存中。但是，在实践中，您需要每次重启 Camel 容器，但最后您的最后一个值将会丢失，且您的尾部的光标消费者将再次开始使用，这很有可能将重复记录发送到您的路由中。

要克服这种情况，您可以启用 持久性尾部跟踪 功能，以跟踪 MongoDB 数据库内特殊集合中持续使用的值。当消费者再次初始时，它将恢复最后跟踪的值，并在不做任何情况一样继续。

最后的读取值会在两个方面保留：每次重新生成光标以及消费者关闭时。如果需求需求，我们可能会考虑定期持续的间隔（每 5 秒清空一次），以便增加稳健性。要请求此功能，请在 Camel JIRA 中创建一个问题单。

230.7.1.3. 启用持久性尾部跟踪

要启用这个功能，请在端点 URI 上至少设置以下选项：

- persistentTailTracking 选项为 true
- persistentId 选项是这个消费者的唯一标识符，因此可以在多个用户间重复使用相同的集合

另外，您可以将 tailTrackDb、tailTrackCollection 和 tailTrackField 选项设置为自定义存储运行时信息的位置。有关每个选项的描述，请参阅本页顶部的端点选项表。

例如，以下路由会消耗来自"flights.cancellations"的上限集合，使用 "departureTime" 作为增加的字段，默认的重新生成光标延迟为 1000ms，并且打开持久性尾部跟踪，并在"flights.camelTailTracking" id 下保留。在 "lastTrackingValue" 字段中存储最后处理的值(camelTail Tracking 和 lastTracking Value 是默认值)。

```
from("mongodb3:myDb?
database=flights&collection=cancellations&tailTrackIncreasingField=departureTime&persistentTailTracking=true" +
    "&persistentId=cancellationsTracker")
  .id("tailableCursorConsumer2")
  .autoStartup(false)
  .to("mock:test");
```

以下是与上述示例相同的另一个示例，但持久性尾部跟踪运行时信息将存储在"lastProcessedDepartureTime"字段中的"trackers.camelTrackers"集合中：

```
from("mongodb3:myDb?
database=flights&collection=cancellations&tailTrackIncreasingField=departureTime&persistentTailTracking=true" +

"&persistentId=cancellationsTracker&tailTrackDb=trackers&tailTrackCollection=camelTrackers" +
    "&tailTrackField=lastProcessedDepartureTime")
  .id("tailableCursorConsumer3")
  .autoStartup(false)
  .to("mock:test");
```

230.7.2. Change Streams Consumer

通过更改流，应用程序可以在不需要复杂性和跟踪 MongoDB oplog 的风险的情况下访问实时数据更改。应用程序可以使用更改流来订阅集合上的所有数据更改，并立即响应它们。由于更改流使用聚合框

架，因此应用程序也可以过滤特定更改或转换通知。

要配置 Change Streams Consumer，您需要指定 consumerType、database、collection 和可选的 JSON 属性 streamFilter 来过滤事件。该 JSON 属性是标准的 MongoDB $match 聚合。它可以通过 XML DSL 配置轻松指定：

```
<route id="filterConsumer">
    <from uri="mongodb3:myDb?
consumerType=changeStreams&amp;database=flights&amp;collection=tickets"/>
    <to uri="mock:test"/>

    <routeProperty key="streamFilter" value="{'$match':{'$or':[{'fullDocument.stringValue':
'specificValue'}]}}"/>
</route>
```

Java 配置：

```
from("mongodb3:myDb?
consumerType=changeStreams&database=flights&collection=tickets")
    .routeProperty("streamFilter", "{'$match':{'$or':[{'fullDocument.stringValue':
'specificValue'}]}}")
    .to("mock:test");
```

230.8. 类型转换

camel-mongodb 组件中包含的 MongoDbBasicConverters 类型转换器提供以下转换：

Name	从类型	要键入	如何？
fromMapToDocument	**Map**	**文档**	创建一个新的 **Document**，通过 **new Document(Map m)** constructor。
fromDocumentToMap	**文档**	**Map**	**文档** 已实施 **映射**。
fromStringToDocument	**字符串**	**文档**	使用 **com.mongodb.Document.parse （字符串 s）**。

Name	从类型	要键入	如何？
fromAnyObjectToDocument	**对象**	**文档**	使用 Jackson 库 将对象转换为 **映射**，后者用于初始新 **文档**。
fromStringToList	**字符串**	**List<Bson>**	使用 **org.bson.codecs.configuration.CodecRegistries** 转换为 BsonArray，然后转换为 List<Bson>。

这个类型转换器被自动发现，因此您不需要手动配置任何内容。

230.9. 另请参阅

- *MongoDB 网站*
- *NoSQL Wikipedia 文章*
- *MongoDB Java driver API 文档 - 当前版本 * 单元测试 以了解更多用法示例*

第 231 章 MQTT 组件

从 Camel 版本 2.10 开始提供

mqtt: 组件用于与 MQTT 兼容消息代理通信，如 Apache ActiveMQ 或 Mosquitto

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-mqtt</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

231.1. URI 格式

```
mqtt://name[?options]
```

其中 name 是您要分配组件的名称。

231.2. 选项

MQTT 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
host (common)	要连接 MQTT 代理的 URI - 此组件也支持 SSL - 例如 ssl://127.0.0.1:8883		字符串
用户名 (security)	用于对 MQTT 代理进行身份验证的用户名		字符串
密码 (security)	用于对 MQTT 代理进行身份验证的密码		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

MQTT 端点使用 URI 语法进行配置：

```
mqtt:name
```

使用以下路径和查询参数：

231.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	**必需** 使用的逻辑名称，而不是主题名称。		字符串

231.2.2. 查询参数(39 参数)：

Name	描述	默认值	类型
blockingExecutor (common)	SSL 连接对内部线程池执行阻止操作，除非您调用 setBlockingExecutor 方法来配置将要使用的 executor。		executor
byDefaultRetain (common)	发送到 MQTT 代理的消息的默认保留策略	false	布尔值
cleanSession (common)	如果您希望 MQTT 服务器跨客户端会话持久保留主题订阅和 ack 位置，则设置为 false。默认值为 true。	false	布尔值
clientId (common)	使用 设置会话的客户端 Id。这是 MQTT 服务器用来识别使用 setCleanSession (false)的会话的内容。id 必须是 23 个字符或更少。默认为自动生成的 id （基于您的套接字地址、端口和时间戳）。		字符串
connectAttemptsMax (common)	在向客户端第一次尝试时将错误报告给客户端之前重新尝试的最大重新连接次数，以连接到服务器。设置为 -1 以使用无限尝试。默认值为 -1。	-1	long
connectWaitInSeconds (common)	组件等待与 MQTT 代理建立连接的延迟（以秒为单位）	10	int
disconnectWaitInSeconds (common)	组件在 stop （）上与 MQTT 代理等待有效的断开连接的秒数	5	int

Name	描述	默认值	类型
assignQueue (common)	HawtDispatch 分配队列用于同步对连接的访问。如果没有通过 setDispatchQueue 方法配置显式队列，则会为连接创建新的队列。如果您希望多个连接共享相同的同步队列，则设置显式队列可能会很方便。		DispatchQueue
host (common)	要连接 MQTT 代理的 URI - 此组件也支持 SSL - 例如 ssl://127.0.0.1:8883	tcp://127.0.0.1:1883	URI
keepalive (common)	以秒为单位配置 Keep Alive 计时器。定义从客户端接收的消息之间的最大时间间隔。它可让服务器检测到客户端的网络连接是否已丢失，而无需等待较长的 TCP/IP 超时。		short
localAddress (common)	要使用的本地 InetAddress 和端口		URI
maxReadRate (common)	设置此传输将接收数据的最大每秒字节数。此设置节流读取，以便不会超过速率。默认为 0，禁用节流。		int
maxWriteRate (common)	设置此传输将发送数据的最大每秒字节数。此设置节流写入，以便不会超过速率。默认为 0，禁用节流。		int
mqttQosPropertyName (common)	要在交换上查找单个已发布消息的属性名称。如果设置了（一个 AtMostOnce、AtLeastOnce 或 ExactlyOnce ）- 那么 QoS 将在发送到 MQTT 消息代理的消息上设置。	MQTT Qos	字符串
mqttRetainPropertyName (common)	要在交换上查找单个已发布消息的属性名称。如果设置了（预期布尔值）- 然后在发送到 MQTT 消息代理的消息上设置 retain 属性。	MQTT Retain	字符串
mqttTopicPropertyName (common)	在 Exchange - 中查找的属性以发布到	MQTT TopicProperty Name	字符串
publishTopicName (common)	发布消息的默认主题	camel/mqtt/test	字符串
qualityOfService (common)	用于主题的服务质量等级。	AtLeastOnce	字符串
receiveBufferSize (common)	设置内部套接字接收缓冲区的大小。默认值为 65536 (64k)	65536	int

Name	描述	默认值	类型
reconnectAttemptsMax (common)	之前建立服务器连接后，将错误报告给客户端的最大重新连接尝试数。设置为 -1 以使用无限尝试。默认值为 -1。	-1	long
reconnectBackOffMultiplier (common)	Exponential backoff 在重新连接尝试之间使用。设置为 1 以禁用 exponential backoff。默认值为 2。	2.0	double
reconnectDelay (common)	第一次重新连接尝试前等待 ms 的时间。默认值为 10。	10	long
reconnectDelayMax (common)	重新连接尝试之间等待的最大时间（以 ms 为单位）。默认值为 30,000。	30000	long
sendBufferSize (common)	设置内部套接字发送缓冲区的大小。默认值为 65536 (64k)	65536	int
sendWaitInSeconds (common)	组件从 MQTT 代理等待收到的最长时间，以确认已发布的消息，然后再抛出异常	5	int
SSLContext (common)	使用 SSLContext 配置配置安全性		SSLContext
subscribeTopicName (common)	**弃用** 它们将在 Endpoint - 中设置，以及从 MQTT 继承的属性		字符串
subscribeTopicNames (common)	为消息订阅以逗号分隔的主题列表。请注意，此列表的每个项目都可以包含 MQTT 通配符（和/或 #），以便订阅与层次结构中特定模式匹配的主题。例如，是层次结构中级别处所有主题的通配符，因此，如果代理有主题/一和主题/两部分，则 topics/ 可用于订阅两者。这里需要考虑的一个注意事项是，如果代理添加了 topics/three，则路由也会开始从该主题接收信息。		字符串
trafficClass (common)	在 IP 标头中为从传输发送的数据包设置流量类或服务 octet。默认值为 8，这意味着应该为吞吐量优化流量。	8	int
version (common)	设置为 3.1.1，以使用 MQTT 版本 3.1。否则，默认为 3.1 协议版本。	3.1	字符串
willMessage (common)	要发送的消息。默认为零长度消息。		字符串
willQos (common)	设置用于 Will 消息的服务质量。默认为 AT_MOST_ONCE。	AtMostOnce	QoS

Name	描述	默认值	类型
willRetain (common)	如果您希望使用 retain 选项发布，则设置为 true。		QoS
willTopic (common)	如果设置服务器，如果客户端出现意外断开连接，则将客户端的消息发布到指定的主题。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
lazySessionCreation (producer)	如果远程服务器在 Camel producer 启动时未启动并在运行，则会话可能会被创建来避免异常。	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

231.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.component.mqtt.enabled	启用 mqtt 组件	true	布尔值
camel.component.mqtt.host	要连接 MQTT 代理的 URI - 此组件也支持 SSL - 例如 ssl://127.0.0.1:8883		字符串
camel.component.mqtt.password	用于对 MQTT 代理进行身份验证的密码		字符串

Name	描述	默认值	类型
camel.component.mqtt.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.mqtt.user-name	用于对 MQTT 代理进行身份验证的用户名		字符串

231.4. SAMPLES

发送消息：

```
from("direct:foo").to("mqtt:cheese?publishTopicName=test.mqtt.topic");
```

使用消息：

```
from("mqtt:bar?
subscribeTopicName=test.mqtt.topic").transform(body().convertToString()).to("mock:result")
```

231.5. ENDPOINTS

Camel 支持使用 Endpoint 接口的消息端点模式。端点通常由组件创建，端点通常通过其 URI 在 DSL 中引用。

从端点中，您可以使用以下方法

- *createProducer （） 将创建一个 Producer 来向端点发送消息交换*
- *createConsumer （） 实现 Event Driven Consumer 模式，以便在创建 Consumer时通过 Processor 从端点消耗消息交换*
- *createPollingConsumer （） 实现 Polling Consumer 模式，用于通过 PollingConsumer从端点消耗消息交换*

231.6. 另请参阅

- 配置 Camel
- 消息端点模式
- URI
- 编写组件

第 232 章 MSV 组件

从 Camel 版本 1.1 开始提供

MSV 组件使用 MSV 库和任何受支持的 XML 模式语言（如 XML Schema 或 RelaxNG XML 语法）执行消息正文的 XML 验证。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-msv</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

请注意，Jing 组件还支持 RelaxNG Compact Syntax

232.1. URI 格式

```
msv:someLocalOrRemoteResource[?options]
```

其中 someLocalOrRemoteResource 是类路径上本地资源的一些 URL，或指向文件系统上的远程资源或资源的完整 URL。例如：

```
msv:org/foo/bar.rng
msv:file:../foo/bar.rng
msv:http://acme.com/cheese.rng
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

232.2. 选项

MSV 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
schemaFactory (advanced)	使用 javax.xml.validation.SchemaFactory。		SchemaFactory
resourceResolverFactory (advanced)	使用自定义 LSResourceResolver，它依赖于动态端点资源 URI		ValidatorResourceResolverFactory
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

MSV 端点使用 URI 语法进行配置：

```
msv:resourceUri
```

使用以下路径和查询参数：

232.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceUri	**所需** URL 到 classpath 上的本地资源，或对 Registry 中查找 bean 的完整 URL，或包含要验证的 XSD 的文件系统上的远程资源或资源的完整 URL。		字符串

232.2.2. 查询参数(11 参数)：

Name	描述	默认值	类型
failOnNullBody (producer)	如果没有正文，是否失败。	true	布尔值
failOnNullHeader (producer)	在针对标头验证时，是否没有标头。	true	布尔值
headerName (producer)	以针对标头而不是邮件正文进行验证。		字符串

Name	描述	默认值	类型
errorHandler (advanced)	使用自定义 org.apache.camel.processor.validation.ValidatorErrorHandler。默认错误处理程序捕获错误并抛出异常。		ValidatorErrorHandler
resourceResolver (advanced)	使用自定义 LSResourceResolver。不要与 resourceResolverFactory 一起使用		LSResourceResolver
resourceResolverFactory (advanced)	使用自定义 LSResourceResolver，它依赖于动态端点资源 URI。默认资源解析器 factory resturns 是一个资源解析器，可以从类路径和文件系统读取文件。不要与 resourceResolver 一起使用。		ValidatorResourceResolverFactory
schemaFactory (advanced)	使用自定义 javax.xml.validation.SchemaFactory		SchemaFactory
schemaLanguage (advanced)	配置 W3C XML Schema 命名空间 URI。	http://www.w3.org/2001/XMLSchema	字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
useDom (advanced)	验证器是否应该使用 DOMSource/DOMResult 或 SaxResult。	false	布尔值
useSharedSchema (advanced)	Schema 实例是否应共享。引进了这个选项来临时解决 JDK 1.6.x 错误。Xerces 不应出现这个问题。	true	布尔值

232.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.msv.enabled	启用 msv 组件	true	布尔值

Name	描述	默认值	类型
camel.component.msv.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.msv.resource-resolver-factory	使用自定义 LSResourceResolver，它依赖于动态端点资源 URI。选项是一个 org.apache.camel.component.validator.ValidatorResourceResolverFactory 类型。		字符串
camel.component.msv.schema-factory	使用 javax.xml.validation.SchemaFactory。选项是 javax.xml.validation.SchemaFactory 类型。		字符串

232.4. EXAMPLE

*以下示例演示了如何 **配置从端点 direct:start 的路由，然后指向两个端点之一，根据 XML 是否与给定的 RelaxNG XML Schema （在 classpath 上提供） mock:valid 或 mock:invalid。***

232.5. 另请参阅

- ***配置 Camel***
- ***组件***
- ***端点***
- ***开始使用***

第 233 章 MUSTACHE 组件

从 Camel 版本 2.12 开始提供

mustache: 组件允许使用 Mustache 模板处理消息。这在使用 Templating 生成请求的响应时是理想的选择。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
<groupId>org.apache.camel</groupId>
<artifactId>camel-mustache</artifactId>
<version>x.x.x</version> <!-- use the same version as your Camel core version -->
</dependency>
```

233.1. URI 格式

```
mustache:templateName[?options]
```

其中 templateName 是要调用的模板的 classpath-local URI，或者远程模板的完整 URL （例如：file://folder/myfile.mustache）。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

233.2. 选项

Mustache 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
allowContextMapAll (producer)	设置上下文映射是否应该允许访问所有详情。默认情况下，只能访问消息正文和标头。可以启用此选项以完全访问当前 Exchange 和 CamelContext。这样做会带来潜在的安全风险，因为这会打开对 CamelContext API 的完整功能的访问。	false	布尔值

Name	描述	默认值	类型
allowTemplateFromHeader (producer)	是否允许使用来自标头的资源模板（默认 false）。**启用此选项具有安全原因。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。**	false	布尔值
mustacheFactory (advanced)	使用自定义 MustacheFactory		MustacheFactory
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Mustache 端点使用 URI 语法进行配置：

```
mustache:resourceUri
```

使用以下路径和查询参数：

233.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceUri	资源 **所需的** 路径。您可以使用这些协议(classpath 为 default)使用： classpath、file、http、ref 或 bean. classpath、file 和 http 加载资源的前缀。ref 将查找 registry 中的资源。bean 将调用 bean 以供用作资源的方法。对于 bean，您可以在点后指定方法名称，如 bean:myBean.myMethod。		字符串

233.2.2. 查询参数(7 参数)：

Name	描述	默认值	类型
allowContextMapAll (producer)	设置上下文映射是否应该允许访问所有详情。默认情况下，只能访问消息正文和标头。可以启用此选项以完全访问当前 Exchange 和 CamelContext。这样做会带来潜在的安全风险，因为这会打开对 CamelContext API 的完整功能的访问。	false	布尔值

Name	描述	默认值	类型
allowTemplateFromHeader (producer)	是否允许使用来自标头的资源模板（默认 false）。**启用此选项具有安全原因。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。**	false	布尔值
contentCache (producer)	设置是否使用资源内容缓存	false	布尔值
encoding (producer)	资源内容的字符编码。		字符串
endDelimiter (producer)	用于标记模板代码结尾的字符。	}}	字符串
startDelimiter (producer)	用于标记模板代码开始的字符。	{{	字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

233.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.mustache.enabled	启用 mustache 组件	true	布尔值
camel.component.mustache.mustache-factory	使用自定义 MustacheFactory。选项是一个 com.github.mustachejava.MustacheFactory 类型。		字符串
camel.component.mustache.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

233.4. MUSTACHE 上下文

Camel 将在 Mustache 上下文中提供交换信息(just a Map)。Exchange 作为以下内容传输：

key	value
交换	**Exchange** 本身。
exchange.properties	**Exchange** 属性.
标头	In 消息的标头。
camelContext	Camel 上下文.
Request（请求）	In 消息。
正文（body）	In 消息正文。
响应	Out 消息（仅适用于 InOut 消息交换模式）。

233.5. 动态模板

Camel 提供了两个标头，您可以为模板或模板内容本身定义不同的资源位置。如果设置了其中任何标头，则 Camel 会通过端点配置的资源使用此标头。这可让您在运行时提供动态模板。

标头	类型	描述	支持版本
MustacheConstants.MUSTACHE_RESOURCE_URI	字符串	要使用的模板资源的 URI，而不是配置的端点。	
MustacheConstants.MUSTACHE_TEMPLATE	字符串	要使用的模板而不是配置的端点。	

233.6. SAMPLES

例如，您可以使用类似如下的内容：

```
from("activemq:My.Queue").
to("mustache:com/acme/MyResponse.mustache");
```

要使用 Mustache 模板模拟对 InOut 消息交换的消息的响应（其中有一个 JMSReplyTo 标头）。

如果要使用 InOnly 并使用消息并将其发送到您可以使用的另一个目的地：

```
from("activemq:My.Queue").
to("mustache:com/acme/MyResponse.mustache").
to("activemq:Another.Queue");
```

可以指定组件应该通过标头动态使用的模板，例如：

```
from("direct:in").
setHeader(MustacheConstants.MUSTACHE_RESOURCE_URI).constant("path/to/my/template.
mustache").
to("mustache:dummy?allowTemplateFromHeader=true");
```

警告

启用 allowTemplateFromHeader 选项具有安全等级。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。

233.7. 电子邮件示例

在本例中，我们希望使用 Mustache 模板进行订单确认电子邮件。电子邮件模板在 Mustache 中布局，如下所示：

```
Dear {{headers.lastName}}}, {{headers.firstName}}

Thanks for the order of {{headers.item}}.
```

> ***Regards Camel Riders Bookstore***
> ***{{body}}***

233.8. 另请参阅

- ***配置 Camel***
- *组件*
- ***端点***
- ***开始使用***

第 234 章 MVEL 组件

从 Camel 版本 2.12 开始提供

mvel: 组件允许您使用 MVEL 模板处理消息。这在使用 Templating 生成请求的响应时是理想的选择。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-mvel</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

234.1. URI 格式

```
mvel:templateName[?options]
```

其中 templateName 是要调用的模板的 classpath-local URI，或者远程模板的完整 URL （例如：file://folder/myfile.mvel）。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

234.2. 选项

MVEL 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
allowContextMapAll (producer)	设置上下文映射是否应该允许访问所有详情。默认情况下，只能访问消息正文和标头。可以启用此选项以完全访问当前 Exchange 和 CamelContext。这样做会带来潜在的安全风险，因为这会打开对 CamelContext API 的完整功能的访问。	false	布尔值

Name	描述	默认值	类型
allowTemplateFromHeader (producer)	是否允许使用来自标头的资源模板（默认 false）。**启用此选项具有安全原因。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。**	false	布尔值

MVEL 端点使用 URI 语法进行配置：

```
mvel:resourceUri
```

使用以下路径和查询参数：

234.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceUri	资源 **所需的** 路径。您可以使用这些协议(classpath 为 default)使用： classpath、file、http、ref 或 bean. classpath、file 和 http 加载资源的前缀。ref 将查找 registry 中的资源。bean 将调用 bean 以供用作资源的方法。对于 bean，您可以在点后指定方法名称，如 bean:myBean.myMethod。		字符串

234.2.2. 查询参数(5 参数)：

Name	描述	默认值	类型
allowContextMapAll (producer)	设置上下文映射是否应该允许访问所有详情。默认情况下，只能访问消息正文和标头。可以启用此选项以完全访问当前 Exchange 和 CamelContext。这样做会带来潜在的安全风险，因为这会打开对 CamelContext API 的完整功能的访问。	false	布尔值
allowTemplateFromHeader (producer)	是否允许使用来自标头的资源模板（默认 false）。**启用此选项具有安全原因。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。**	false	布尔值
contentCache (producer)	设置是否使用资源内容缓存	false	布尔值
encoding (producer)	资源内容的字符编码。		字符串

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

234.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.mvel.enabled	启用 mvel 组件	true	布尔值
camel.component.mvel.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.language.mvel.enabled	启用 mvel 语言	true	布尔值
camel.language.mvel.trim	是否修剪值以移除前导和结尾的空格和换行符	true	布尔值

234.4. 消息标头

mvel 组件在消息上设置几个标头。

标头	描述
CamelMvelResourceUri	templateName 作为 **String** 对象。

234.5. MVEL 上下文

Camel 将在 MVEL 上下文中提供交换信息(just a Map)。 Exchange 被传输为：

key	value
交换	**Exchange** 本身。
exchange.properties	**Exchange** 属性.
标头	In 消息的标头。
camelContext	Camel 上下文安装.
Request（请求）	In 消息。
in	In 消息。
正文（body）	In 消息正文。
out	Out 消息（仅适用于 InOut 消息交换模式）。
响应	Out 消息（仅适用于 InOut 消息交换模式）。

234.6. 热重新载入

mvel 模板资源默认为 file 和 classpath 资源(expanded jar)的热重新加载。如果您设置了 contentCache=true，Camel 将只加载一次资源，因此无法热重新载入。当资源永不更改时，此场景可用于生产环境。

234.7. 动态模板

Camel 提供了两个标头，您可以为模板或模板内容本身定义不同的资源位置。如果设置了其中任何标头，则 Camel 会通过端点配置的资源使用此标头。这可让您在运行时提供动态模板。

标头	类型	描述
CamelMvelResourceUri	字符串	要使用的模板资源的 URI，而不是配置的端点。
CamelMvelTemplate	字符串	要使用的模板而不是配置的端点。

234.8. SAMPLES

例如，您可以使用类似如下的内容

```
from("activemq:My.Queue").
  to("mvel:com/acme/MyResponse.mvel");
```

要使用 MVEL 模板对 InOut 消息交换的消息（其中有一个 JMSReplyTo 标头）发送响应。

要指定组件应该通过标头动态使用的模板，例如：

```
from("direct:in").
  setHeader("CamelMvelResourceUri").constant("path/to/my/template.mvel").
  to("mvel:dummy?allowTemplateFromHeader=true");
```

要将模板直接指定为标头，组件应该通过标头动态使用，例如：

```
from("direct:in").
  setHeader("CamelMvelTemplate").constant("@{\"The result is \" + request.body * 3}\" }").
  to("velocity:dummy?allowTemplateFromHeader=true");
```

警告

启用 allowTemplateFromHeader 选项具有安全等级。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。

234.9. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 235 章 MVEL 语言

从 Camel 版本 2.0 开始提供

Camel 允许 Mvel 用作 Expression 或 Predicate the DSL 或 Xml 配置。

您可以使用 Mvel 在 Message Filter 中创建一个 predicates，或作为 Recipient List 的 Expression

您可以使用 Mvel dot 表示法调用操作。如果您对于实例有一个正文，其中包含 getFamiliyName 方法的 POJO，则您可以按照以下方式构建语法：

```
"request.body.familyName"
  // or
"getRequest().getBody().getFamilyName()"
```

235.1. MVEL 选项

MVEL 语言支持 1 个选项，如下所列。

Name	默认值	Java 类型	描述
trim	**true**	**布尔值**	是否修剪值以移除前导和结尾的空格和换行符

235.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.mvel.enabled	启用 mvel 组件	true	布尔值
camel.component.mvel.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Name	描述	默认值	类型
camel.language.mvel.enabled	启用 mvel 语言	true	布尔值
camel.language.mvel.trim	是否修剪值以移除前导和结尾的空格和换行符	true	布尔值

235.3. 变量

变量	类型	描述
这	Exchange	Exchange 是根对象
交换	Exchange	Exchange 对象
例外	Throwable	Exchange 异常（如果有）
exchangeId	字符串	交换 ID
Fulting	消息	Fault 消息（如果有）
request	消息	exchange.in 消息
response	消息	exchange.out 消息（如果有）
属性	Map	交换属性
property (name)	对象	给定名称的属性
property (name, type)	类型	给定名称中的属性作为给定类型

235.4. SAMPLES

例如，您可以在 XML 中的 Message Filter 中使用 Mvel

-

```
<route>
  <from uri="seda:foo"/>
  <filter>
    <mvel>request.headers.foo == 'bar'</mvel>
    <to uri="seda:bar"/>
  </filter>
</route>
```

和使用 Java DSL 的示例：

```
from("seda:foo").filter().mvel("request.headers.foo == 'bar'").to("seda:bar");
```

235.5. 从外部资源载入脚本

从 Camel 2.11 开始提供

您可以对脚本进行外部化，并让 Camel 从资源（如 "classpath:"、"file:" 或 "http:" ）加载它。这可以通过以下语法完成：" resource:scheme:location"，例如引用您可以进行的类路径上的文件：

```
.setHeader("myHeader").mvel("resource:classpath:script.mvel")
```

235.6. 依赖项

要在 camel 路由中使用 Mvel，您需要添加对实现 Mvel 语言的 camel-mvel 的依赖。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-mvel</artifactId>
  <version>x.x.x</version>
</dependency>
```

第 236 章 MYBATIS 组件

从 Camel 版本 2.7 开始提供

mybatis: 组件允许您使用 MyBatis 查询、轮询、插入、更新和删除关系数据库中的数据。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-mybatis</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

236.1. URI 格式

```
mybatis:statementName[?options]
```

其中 statementName 是 MyBatis XML 映射文件中的声明名称，它映射到您要评估的查询、插入、更新或删除操作。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

默认情况下，此组件将从 classpath 的根目录加载 MyBatis SqlMapConfig 文件，其预期名称为 SqlMapConfig.xml。
如果文件位于另一个位置，则需要在 MyBatisComponent 组件上配置 configurationUri 选项。

236.2. 选项

MyBatis 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
sqlSessionFactory (advanced)	使用 SqlSessionFactory		SqlSessionFactory

Name	描述	默认值	类型
configurationUri (common)	MyBatis xml 配置文件的位置。默认值为：从 classpath 加载的 SqlMapConfig.xml	SqlMapConfig.xml	字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

MyBatis 端点使用 URI 语法进行配置：

```
mybatis:statement
```

使用以下路径和查询参数：

236.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
声明	**必需的** MyBatis XML 映射文件中的声明名称，该文件映射到查询、插入、更新或删除您要评估的操作。		字符串

236.2.2. 查询参数(29 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
maxMessagesPerPoll (consumer)	这个选项旨在将数据库池返回的结果分成批处理，并在多个交换中提供它们。此整数定义了在单个交换中发送的最大消息。默认情况下，不设置最大值。可用于设置限制，例如 1000，以避免在启动有数千个文件的服务器时避免。设置 0 或负值以禁用它。	0	int

Name	描述	默认值	类型
onConsume (consumer)	在路由中处理数据后要运行的声明		字符串
routeEmptyResultSet (consumer)	是否允许空结果集路由到下一跃点	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
transacted (consumer)	启用或禁用事务。如果启用，如果处理交换失败，则消费者无法处理任何进一步的交换，从而导致回滚操作	false	布尔值
useIterator (consumer)	单独或作为列表处理结果	true	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
processingStrategy (consumer)	使用自定义 MyBatisProcessingStrategy		MyBatisProcessing 策略
executorType (producer)	执行语句时要使用的 executor 类型。simple - executor 做任何特殊操作。reuse - executor reuses prepared 语句. batch - executor reuse 语句和批处理更新。	SIMPLE	ExecutorType
inputHeader (producer)	用户输入参数的标头值，而不是消息正文。默认情况下，inputHeader == null 和 input 参数从消息正文获取。如果设置了 outputHeader，则使用值，并从标头中获取查询参数，而不是正文。		字符串

Name	描述	默认值	类型
outputHeader (producer)	将查询结果存储在标头中而不是消息正文。默认情况下，outputHeader == null，查询结果存储在消息正文中，消息正文中的任何现有内容都会被丢弃。如果设置了 outputHeader，则值将用作标头名称来存储查询结果，并保留原始消息正文。设置 outputHeader 也会省略填充默认的 CamelMyBatisResult 标头，因为它与所有时间都相同。		字符串
statementType (producer)	必须为制作者指定控制要调用的操作类型。		StatementType
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler

Name	描述	默认值	类型
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

236.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.mybatis.configuration-uri	MyBatis xml 配置文件的位置。默认值为：从 classpath 加载的 SqlMapConfig.xml	SqlMapConfig.xml	字符串
camel.component.mybatis.enabled	启用 mybatis 组件	true	布尔值
camel.component.mybatis.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.mybatis.sql-session-factory	使用 SqlSessionFactory。选项是 org.apache.ibatis.session.SqlSessionFactory 类型。		字符串

236.4. 消息标头

Camel 将使用语句填充结果消息，可以是 IN 或 OUT ：

标头	类型	描述
CamelMyBatisStatementName	**字符串**	使用的 **statementName** （例如： insertAccount）。
CamelMyBatisResult	**对象**	从 MtBatis 返回的任何操作 **的响应**。例如，**INSERT** 可能会返回自动生成的密钥，或者行数等。

236.5. 消息正文

来自 MyBatis 的响应只有在是 SELECT 语句时，才会设置为正文。例如，对于 INSERT 语句 Camel 不会替换正文。这可让您继续路由，并保留原始的正文。MyBatis 的响应始终存储在标头中，其键为 CamelMyBatisResult。

236.6. SAMPLES

例如，如果要使用 JMS 队列中的 Bean 并将它们插入到数据库中，您可以执行以下操作：

```
from("activemq:queue:newAccount")
  .to("mybatis:insertAccount?statementType=Insert");
```

请注意，我们必须指定 statementType，因为我们需要指示 Camel 要调用的操作类型。

其中 insertAccount 是 SQL 映射文件中的 MyBatis ID：

```
<!-- Insert example, using the Account parameter class -->
<insert id="insertAccount" parameterType="Account">
  insert into ACCOUNT (
    ACC_ID,
    ACC_FIRST_NAME,
    ACC_LAST_NAME,
    ACC_EMAIL
  )
  values (
    #{id}, #{firstName}, #{lastName}, #{emailAddress}
  )
</insert>
```

236.7. 使用 STATEMENTTYPE 来更好地控制 MYBATIS

当路由到 MyBatis 端点时，您需要更为精细的控制，以便您可以控制要执行的 SQL 语句是否为 SELECT、UPDATE、DELETE 或 INSERT 等。例如，如果我们希望路由到一个 MyBatis 端点，则 IN 正文包含参数到 SELECT 语句，我们可以执行以下操作：

在上面的代码中，我们可以调用 MyBatis 语句 selectAccountById，而 IN 正文应包含您要检索的帐户 ID，如 Integer 类型。

对于某些其他操作，我们可以执行相同的操作，如 SelectList ：

与 UPDATE 相同，我们可以将 帐户 对象作为 IN 正文发送到 MyBatis ：

236.7.1. 使用 InsertList StatementType

从 Camel 2.10 开始提供

MyBatis 允许您使用其 for-each batch 驱动程序插入多行。要使用它，您需要在映射器 XML 文件中使用 <foreach>。例如：

然后，您可以通过向使用 InsertList 语句类型的 mybatis 端点发送 Camel 消息来插入多行，如下所示：

236.7.2. 使用 UpdateList StatementType

从 Camel 2.11 开始提供

MyBatis 允许您使用其 for-each batch 驱动程序更新多行。要使用它，您需要在映射器 XML 文件中使用 <foreach>。例如：

```
<update id="batchUpdateAccount" parameterType="java.util.Map">
    update ACCOUNT set
    ACC_EMAIL = #{emailAddress}
    where
    ACC_ID in
    <foreach item="Account" collection="list" open="(" close=")" separator=",">
```

```
        #{Account.id}
    </foreach>
</update>
```

然后，您可以通过向使用 UpdateList 语句类型的 mybatis 端点发送 Camel 消息来更新多行，如下所示：

```
from("direct:start")
    .to("mybatis:batchUpdateAccount?statementType=UpdateList")
    .to("mock:result");
```

236.7.3. 使用 DeleteList StatementType

从 Camel 2.11 开始提供

MyBatis 允许您使用其 for-each batch 驱动程序删除多行。要使用它，您需要在映射器 XML 文件中使用 <foreach>。例如：

```
<delete id="batchDeleteAccountById" parameterType="java.util.List">
    delete from ACCOUNT
    where
    ACC_ID in
    <foreach item="AccountID" collection="list" open="(" close=")" separator=",">
        #{AccountID}
    </foreach>
</delete>
```

然后，您可以通过向使用 DeleteList 语句类型的 mybatis 端点发送 Camel 消息来删除多行，如下所示：

```
from("direct:start")
    .to("mybatis:batchDeleteAccount?statementType=DeleteList")
    .to("mock:result");
```

236.7.4. 注意 InsertList、UpdateList 和 DeleteList StatementTypes

任何类型(List、Map 等)的参数都可以传递给 mybatis，最终用户负责处理它，以帮助 mybatis 动态查询 功能。

236.7.5. 调度的轮询示例

此组件支持调度的轮询，因此可用作 Polling Consumer。例如，每分钟轮询数据库：

```
from("mybatis:selectAllAccounts?delay=60000")
  .to("activemq:queue:allAccounts");
```

如需了解更多选项，请参阅 Polling Consumer 中的 "ScheduledPollConsumer Options"。

或者，您可以使用其他机制触发调度的轮询，如 Timer 或 Quartz 组件。在以下示例中，我们轮询数据库，每 30 秒使用 Timer 组件并将数据发送到 JMS 队列：

```
from("timer://pollTheDatabase?delay=30000")
  .to("mybatis:selectAllAccounts")
  .to("activemq:queue:allAccounts");
```

和使用的 MyBatis SQL 映射文件：

```
<!-- Select with no parameters using the result map for Account class. -->
<select id="selectAllAccounts" resultMap="AccountResult">
  select * from ACCOUNT
</select>
```

236.7.6. 使用 onConsume

此组件支持在 Camel 使用和处理数据 后执行 声明。这可让您在数据库中进行升级后的更新。注意所有语句都必须是 UPDATE 语句。Camel 支持执行多个语句，其名称应该用逗号分开。

以下路由演示了我们执行 consumeAccount 语句数据。这样，我们可以将数据库中行的状态更改为处理，因此我们避免消耗两次。

和 sqlmap 文件中的语句：

236.7.7. 参与事务

在 camel-mybatis 下设置事务管理器可能会很有点，因为它涉及在标准 MyBatis SqlMapConfig.xml 文件外的外部数据库配置。

第一部分要求设置 数据源。这通常是池(DBCP 或 c3p0)，后者需要嵌套在 Spring 代理中。此代理可

让 DataSource 的非 Spring 使用 DataSource 参与 Spring 事务(MyBatis SqlSessionFactory 只是这样)。

```
<bean id="dataSource"
class="org.springframework.jdbc.datasource.TransactionAwareDataSourceProxy">
    <constructor-arg>
        <bean class="com.mchange.v2.c3p0.ComboPooledDataSource">
            <property name="driverClass" value="org.postgresql.Driver"/>
            <property name="jdbcUrl" value="jdbc:postgresql://localhost:5432/myDatabase"/>
            <property name="user" value="myUser"/>
            <property name="password" value="myPassword"/>
        </bean>
    </constructor-arg>
</bean>
```

这具有额外的优点，使数据库配置能够使用属性占位符进行外部化。

然后，事务管理器被配置为管理外部的 DataSource ：

```
<bean id="txManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"/>
</bean>
```

mybatis-springSqlSessionFactoryBean 然后嵌套相同的 DataSource ：

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <!-- standard mybatis config file -->
<property name="configLocation" value="/META-INF/SqlMapConfig.xml"/>
    <!-- externalised mappers -->
<property name="mapperLocations" value="classpath*:META-INF/mappers/**/*.xml"/>
</bean>
```

然后，使用该工厂配置 camel-mybatis 组件：

```
<bean id="mybatis" class="org.apache.camel.component.mybatis.MyBatisComponent">
    <property name="sqlSessionFactory" ref="sqlSessionFactory"/>
</bean>
```

最后，在事务管理器的顶部定义一个事务策略，然后可以正常使用：

```
<bean id="PROPAGATION_REQUIRED"
```

```
class="org.apache.camel.spring.spi.SpringTransactionPolicy">
    <property name="transactionManager" ref="txManager"/>
    <property name="propagationBehaviorName" value="PROPAGATION_REQUIRED"/>
</bean>

<camelContext id="my-model-context" xmlns="http://camel.apache.org/schema/spring">
    <route id="insertModel">
        <from uri="direct:insert"/>
        <transacted ref="PROPAGATION_REQUIRED"/>
        <to uri="mybatis:myModel.insert?statementType=Insert"/>
    </route>
</camelContext>
```

第 237 章 MYBATIS BEAN 组件

从 Camel 版本 2.22 开始提供

mybatis-bean: 组件允许您使用 MyBatis bean 注解查询、插入、更新和删除关系数据库中的数据。

此组件 只能 用作制作者。如果要从 MyBatis 消耗，则使用常规的 mybatis 组件。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-mybatis</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

默认情况下，此组件将从 classpath 的根目录加载 MyBatis SqlMapConfig 文件，其预期名称为 SqlMapConfig.xml。
如果文件位于另一个位置，则需要在 MyBatisComponent 组件上配置 configurationUri 选项。

237.1. 选项

MyBatis Bean 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
sqlSessionFactory (advanced)	使用 SqlSessionFactory		SqlSessionFactory
configurationUri (producer)	MyBatis xml 配置文件的位置。默认值为：从 classpath 加载的 SqlMapConfig.xml	SqlMapConfig.xml	字符串
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

MyBatis Bean 端点使用 URI 语法进行配置：

```
mybatis-bean:beanName:methodName
```

使用以下路径和查询参数：

237.1.1. 路径参数(2 参数)：

Name	描述	默认值	类型
beanName	带有 MyBatis 注解的 bean **必需** 的名称。这可以通过类型别名或 FQN 类名称。		字符串
methodName	Bean 上具有要执行的 SQL 查询方法 **必需** 的名称。		字符串

237.1.2. 查询参数(4 参数)：

Name	描述	默认值	类型
executorType (producer)	执行语句时要使用的 executor 类型。simple - executor 做任何特殊操作。reuse - executor reuses prepared 语句. batch - executor reuse 语句和批处理更新。	SIMPLE	ExecutorType
inputHeader (producer)	用户输入参数的标头值，而不是消息正文。默认情况下，inputHeader == null 和 input 参数从消息正文获取。如果设置了 outputHeader，则使用值，并从标头中获取查询参数，而不是正文。		字符串
outputHeader (producer)	将查询结果存储在标头中而不是消息正文。默认情况下，outputHeader == null，查询结果存储在消息正文中，消息正文中的任何现有内容都会被丢弃。如果设置了 outputHeader，则值将用作标头名称来存储查询结果，并保留原始消息正文。设置 outputHeader 也会省略填充默认的 CamelMyBatisResult 标头，因为它与所有时间都相同。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

237.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.mybatis-bean.configuration-uri	MyBatis xml 配置文件的位置。默认值为：从 classpath 加载的 SqlMapConfig.xml	SqlMapConfig.xml	字符串
camel.component.mybatis-bean.enabled	是否启用 mybatis-bean 组件的自动配置。这默认是启用的。		布尔值
camel.component.mybatis-bean.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.mybatis-bean.sql-session-factory	使用 SqlSessionFactory。选项是 org.apache.ibatis.session.SqlSessionFactory 类型。		字符串

237.3. 消息标头

Camel 将使用语句填充结果消息，可以是 IN 或 OUT ：

标头	类型	描述
CamelMyBatisResult	**对象**	从 MtBatis 返回的任何操作 **的响应**。例如，**INSERT** 可能会返回自动生成的密钥，或者行数等。

237.4. 消息正文

来自 MyBatis 的响应只有在是 SELECT 语句时，才会设置为正文。例如，对于 INSERT 语句 Camel 不会替换正文。这可让您继续路由，并保留原始的正文。MyBatis 的响应始终存储在标头中，其键为 CamelMyBatisResult。

237.5. SAMPLES

例如，如果要使用 JMS 队列中的 Bean 并将它们插入到数据库中，您可以执行以下操作：

```
from("activemq:queue:newAccount")
  .to("mybatis-bean:AccountService:insertBeanAccount");
```

请注意，我们必须指定 bean 名称和方法名称，因为我们需要指示 Camel 要调用的操作类型。

其中 AccountService 是具有 MyBatis bean 注解的 bean 类型别名。您可以在 SqlMapConfig 文件中配置类型别名：

```
    <typeAliases>
        <typeAlias alias="Account" type="org.apache.camel.component.mybatis.Account"/>
        <typeAlias alias="AccountService"
type="org.apache.camel.component.mybatis.bean.AccountService"/>
    </typeAliases>
```

```
On the `AccountService` bean you can declare the MyBatis mappins using annotations as shown:
```

```
public interface AccountService {

    @Select("select ACC_ID as id, ACC_FIRST_NAME as firstName, ACC_LAST_NAME as
lastName"
        + ", ACC_EMAIL as emailAddress from ACCOUNT where ACC_ID = #{id}")
    Account selectBeanAccountById(@Param("id") int no);

    @Select("select * from ACCOUNT order by ACC_ID")
    @ResultMap("Account.AccountResult")
    List<Account> selectBeanAllAccounts();

    @Insert("insert into ACCOUNT
(ACC_ID,ACC_FIRST_NAME,ACC_LAST_NAME,ACC_EMAIL)"
        + " values (#{id}, #{firstName}, #{lastName}, #{emailAddress})")
    void insertBeanAccount(Account account);

}
```

第 238 章 NAGIOS 组件

从 Camel 版本 2.3 开始提供

Nagios 组件允许您将被动检查发送到 Nagios。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-nagios</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

238.1. URI 格式

```
nagios://host[:port][?Options]
```

Camel 通过 Nagios 组件提供两个功能。您可以通过向其端点发送消息来发送被动检查消息。Camel 还提供了一个 EventNotifer，允许您向 Nagios 发送通知。

238.2. 选项

Nagios 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	使用共享 NagiosConfiguration		NagiosConfiguration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Nagios 端点使用 URI 语法进行配置：

```
nagios:host:port
```

使用以下路径和查询参数：

238.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
主机	**必需是** 应发送检查的 Nagios 主机的地址。		字符串
port	**必需** 主机的端口号。		int

238.2.2. 查询参数(7 参数)：

Name	描述	默认值	类型
connectionTimeout (producer)	millis 中的连接超时。	5000	int
sendSync (producer)	在发送被动检查时是否使用同步。将它设置为 false 将允许 Camel 继续路由消息，并且被动检查消息将异步发送。	true	布尔值
timeout (producer)	在 millis 中发送超时。	5000	int
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
加密 （安全）	指定加密方法。		Encryption
encryptionMethod (security)	**弃用** 以指定加密方法。		NagiosEncryptionMethod
密码 (security)	向 Nagios 发送检查时要进行身份验证的密码。		字符串

238.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
camel.component.nagios.configuration.connection-timeout	millis 中的连接超时。	5000	整数
camel.component.nagios.configuration.encryption	指定加密方法。		Encryption
camel.component.nagios.configuration.host	这是应发送检查的 Nagios 主机的地址。		字符串
camel.component.nagios.configuration.nagios-settings			NagiosSettings
camel.component.nagios.configuration.password	向 Nagios 发送检查时要进行身份验证的密码。		字符串
camel.component.nagios.configuration.port	主机的端口号。		整数
camel.component.nagios.configuration.timeout	在 millis 中发送超时。	5000	整数
camel.component.nagios.enabled	启用 nagios 组件	true	布尔值
camel.component.nagios.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.nagios.configuration.encryption-method	指定加密方法。		NagiosEncryptionMethod

238.4. 发送消息示例

您可以向 Nagios 发送消息，其中消息有效负载包含消息。默认情况下，它将是 OK 级别，并使用 CamelContext 名称作为服务名称。您可以使用如上所示的标头来覆盖这些值。

例如，我们将 Hello Nagios 消息发送给 Nagios，如下所示：

```
template.sendBody("direct:start", "Hello Nagios");

from("direct:start").to("nagios:127.0.0.1:5667?password=secret").to("mock:result");
```

要发送 CRITICAL 消息，您可以发送标头，例如：

```
Map headers = new HashMap();
headers.put(NagiosConstants.LEVEL, "CRITICAL");
headers.put(NagiosConstants.HOST_NAME, "myHost");
headers.put(NagiosConstants.SERVICE_NAME, "myService");
template.sendBodyAndHeaders("direct:start", "Hello Nagios", headers);
```

238.5. 使用 NAGIOSEVENTNOTIFER

Nagios 组件还提供 EventNotifer，可用于向 Nagios 发送事件。例如，我们可以从 Java 启用它，如下所示：

```
NagiosEventNotifier notifier = new NagiosEventNotifier();
notifier.getConfiguration().setHost("localhost");
notifier.getConfiguration().setPort(5667);
notifier.getConfiguration().setPassword("password");

CamelContext context = ...
context.getManagementStrategy().addEventNotifier(notifier);
return context;
```

在 Spring XML 中，使用类型 EventNotifier 和 Camel 定义 Spring bean 只会选择它，如此文档：使用 Spring 进行 CamelContext 的高级配置。

238.6. 另请参阅

- 配置 Camel
- 组件

- *端点*
- *开始使用*

第 239 章 NAT 组件

从 Camel 版本 2.17 开始提供

NATS 是一个快速可靠的消息传递平台。

Maven 用户需要将以下依赖项添加到此组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-nats</artifactId>
  <!-- use the same version as your Camel core version -->
  <version>x.y.z</version>
</dependency>
```

239.1. URI 格式

```
nats:servers[?options]
```

其中 servers 代表 NATS 服务器列表。

239.2. 选项

Nats 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Nats 端点使用 URI 语法进行配置：

```
nats:servers
```

使用以下路径和查询参数：

239.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
服务器	一个或多个 NAT 服务器 **所需的** URL。在指定多个服务器时，使用逗号分隔 URL。		字符串

239.2.2. 查询参数(25 参数)：

Name	描述	默认值	类型
connection (common)	引用已经实例化到 Nats 服务器的连接		连接
connectionTimeout (common)	连接尝试超时。（以毫秒为单位）	2000	int
flushConnection (common)	定义是否要刷新连接	false	布尔值
flushTimeout (common)	设置冲刷超时（以毫秒为单位）	1000	int
maxPingsOut (common)	客户端允许的最大 ping 数量	2	int
maxReconnectAttempts (common)	最大重新连接尝试	60	int
noEcho (common)	关闭 echo。如果您连接到此标志的 gnatsd 版本支持可防止服务器在主题上包括订阅时将消息回连接。	false	布尔值
noRandomizeServers (common)	是否为连接尝试随机化服务器顺序	false	布尔值
pedantic (common)	是否以 pedantic 模式运行（这会影响 performace）	false	布尔值
pingInterval (common)	ping 间隔，请注意连接是否仍处于活动状态（以毫秒为单位）	120000	int
reconnect (common)	是否使用 reconnection 功能	true	布尔值

Name	描述	默认值	类型
reconnectTimeWait (common)	尝试重新连接前等待时间（以毫秒为单位）	2000	int
requestCleanupInterval (common)	清理取消/超时请求的间隔。	5000	int
topic (common)	**必需** 要使用的主题名称		字符串
verbose (common)	是否以详细模式运行	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
maxMessages (consumer)	停止从主题接收信息，在 maxMessages 后订阅		字符串
poolSize (consumer)	消费者池大小	10	int
queueName (consumer)	如果我们对队列配置使用 nats，则 Queue 名称		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
replySubject (producer)	订阅者应向其发送响应的主题		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
安全 （安全）	设置 secure 选项表示需要 TLS	false	布尔值
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters

239.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.nats.enabled	启用 nats 组件	true	布尔值
camel.component.nats.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.nats.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

239.4. HEADERS

Name	类型	描述
CamelNatsMessageTimestamp	long	所消耗的消息的时间戳。

生成者示例：

```
from("direct:send").to("nats://localhost:4222?topic=test");
```

如果使用授权，您可以在服务器 URL 中直接指定凭证

```
from("direct:send").to("nats://username:password@localhost:4222?topic=test");
```

或您的令牌

```
from("direct:send").to("nats://token@localhost:4222?topic=test");
```

消费者示例：

```
from("nats://localhost:4222?
topic=test&maxMessages=5&queueName=test").to("mock:result");
```

第 240 章 NETTY 组件（已弃用）

从 Camel 版本 2.3 开始提供

警告

此组件已弃用。您应使用 Netty4。

Camel 中的 netty 组件是一个套接字通信组件，基于 Netty 项目。

Netty 是一个 NIO 客户端服务器框架，能够快速轻松地开发网络应用程序，如协议服务器和客户端。Netty 大大简化了网络编程，如 TCP 和 UDP 套接字服务器。

此 camel 组件支持生成者和消费者端点。

Netty 组件有多个选项，允许对多个 TCP/UDP 通信参数（缓冲大小、keepAlives、tcpNoDelay 等）进行精细控制，并促进 Camel 路由上的 In-Only 和 In-Out 通信。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-netty</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

240.1. URI 格式

netty 组件的 URI 方案如下

```
netty:tcp://0.0.0.0:99999[?options]
netty:udp://remotehost:99999/[?options]
```

此组件支持 TCP 和 UDP 的制作者和消费者端点。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

240.2. 选项

Netty 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	在创建端点时，使用 NettyConfiguration 作为配置。		NettyConfiguration
maximumPoolSize (advanced)	排序线程池的核心池大小（如果使用）。默认值为 16。	16	int
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Netty 端点使用 URI 语法进行配置：

```
netty:protocol:host:port
```

使用以下路径和查询参数：

240.2.1. 路径参数(3 参数)：

Name	描述	默认值	类型
protocol	**必需** 使用的协议，可以是 tcp 或 udp。		字符串
主机	**必需** 的主机名。对于消费者，主机名是 localhost 或 0.0.0.0，对于生成者，主机名是要连接的远程主机		字符串
port	**需要** 主机端口号		int

240.2.2. 查询参数(67 参数)：

Name	描述	默认值	类型
disconnect (common)	使用后是否从 Netty Channel 断开连接（关闭）。可用于使用者和制作者。	false	布尔值
keepalive (common)	设置以确保因为不活跃而不会关闭套接字	true	布尔值
reuseAddress (common)	设置以方便套接字多路	true	布尔值
sync (common)	将端点设置为单向或请求响应	true	布尔值
tcpNoDelay (common)	设置以提高 TCP 协议性能	true	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
broadcast (consumer)	将 设置为选择通过 UDP 的多播	false	布尔值
clientMode (consumer)	如果 clientMode 为 true，netty 使用者会将地址连接到 TCP 客户端。	false	布尔值
backlog (consumer)	允许为 netty consumer (server)配置积压。请注意，积压只是取决于操作系统的最佳努力。将此选项设置为值（如 200、500 或 1000）告知 TCP 堆栈如果未配置此选项，则接受队列的时长，则积压取决于操作系统设置。		int
bossCount (consumer)	当 netty 适用于 nio 模式时，它使用 Netty 中的默认 bossCount 参数，即 1。用户可以使用此操作从 Netty 覆盖默认的 bossCount	1	int
bossPool (consumer)	使用显式 org.jboss.netty.channel.socket.nio.BossPool 作为 boss 线程池。例如，要与多个使用者共享线程池。默认情况下，每个消费者都有自己的带有 1 个核心线程的 boss 池。		BossPool
channelGroup (consumer)	使用显式 ChannelGroup。		ChannelGroup

Name	描述	默认值	类型
disconnectOnNoReply (consumer)	如果启用了同步，则此选项将指定 NettyConsumer（如果它应该断开连接），没有回复回来。	true	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
maxChannelMemorySize (consumer)	使用 orderedThreadPoolExecutor 时每个频道的最大排队事件总量。指定 0 来禁用。	1048576	long
maxTotalMemorySize (consumer)	使用 orderedThreadPoolExecutor 时，此池的最大排队事件总量。指定 0 来禁用。	209715200	long
nettyServerBootstrapFactory (consumer)	使用自定义 NettyServerBootstrapFactory		NettyServerBootstrap Factory
networkInterface (consumer)	在使用 UDP 时，可以使用此选项按名称指定网络接口，如 eth0 以加入多播组。		字符串
noReplyLogLevel (consumer)	如果启用了同步，这个选项会指示 NettyConsumer，在记录一个没有回复时要使用的日志记录级别。	WARN	LoggingLevel
orderedThreadPoolExecutor (consumer)	是否使用排序的线程池，确保同一通道上按顺序处理事件。如需了解更多详细信息，请参阅 org.jboss.netty.handler.execution.OrderedMemoryAwareThreadPoolExecutor 的 netty javadoc。	true	布尔值
serverClosedChannel ExceptionCaught LogLevel (consumer)	如果服务器(NettyConsumer)捕获 java.nio.channels.ClosedChannelException，然后使用此日志级别进行记录。这用于避免记录关闭的频道异常，因为客户端可以立即断开，然后在 Netty 服务器中造成大量关闭的异常。	DEBUG	LoggingLevel
serverExceptionCaughtLog Level (consumer)	如果服务器(NettyConsumer)捕获异常，则使用此日志级别进行日志记录。	WARN	LoggingLevel
serverPipelineFactory (consumer)	使用自定义 ServerPipelineFactory		ServerPipelineFactory

Name	描述	默认值	类型
workerCount (consumer)	当 netty 适用于 nio 模式时，它会使用来自 Netty 的默认 workerCount 参数，即 cpu_core_threads2。用户可以使用此操作从 Netty 覆盖默认的 workerCount		int
workerPool (consumer)	使用显式 org.jboss.netty.channel.socket.nio.WorkerPool 作为 worker 线程池。例如，要与多个使用者共享线程池。默认情况下，每个使用者都有自己的 worker 池，有 2 个 x cpu 数内核线程。		WorkerPool
connectTimeout (producer)	等待套接字连接可用的时间。价值为 millis。	10000	long
requestTimeout (producer)	在调用远程服务器时，允许 Netty producer 使用超时。默认情况下，没有使用超时。该值在 milli 秒内，如 30000 为 30 秒。requestTimeout 使用 Netty 的 ReadTimeoutHandler 来触发超时。		long
clientPipelineFactory (producer)	使用自定义 ClientPipelineFactory		ClientPipelineFactory
lazyChannelCreation (producer)	如果远程服务器在 Camel producer 启动时未启动并运行，则可以创建频道以避免异常。	true	布尔值
producerPoolEnabled (producer)	producer 池是否已启用。重要： 不要关闭此操作，因为处理并发和可靠的请求/回复需要池。	true	布尔值
producerPoolMaxActive (producer)	设置池可分配给客户端或闲置等待签出的对象数量上限。对没有限制，使用负值。	-1	int
producerPoolMaxIdle (producer)	设置池中空闲实例数量上限。	100	int
producerPoolMinEvictable Idle (producer)	设置对象在空闲对象驱除有资格驱除前，对象可能会在池中闲置的最小时间（值为 millis））。	300000	long
producerPoolMinIdle (producer)	设置制作者池中允许的最小实例数量，然后再驱除线程（如果活跃）生成新对象。		int
udpConnectionlessSending (producer)	这个选项支持连接较少的 udp 发送，这是实际触发和忘记的。如果没有端口侦听接收端口，则已连接的 udp 发送 PortUnreachableException。	false	布尔值
useChannelBuffer (producer)	如果 useChannelBuffer 为 true，则netty 生成者会在发送前将消息正文转换为 ChannelBuffer。	false	布尔值

Name	描述	默认值	类型
bootstrapConfiguration (advanced)	使用自定义配置的 NettyServerBootstrapConfiguration 来配置此端点。		NettyServerBootstrap Configuration
选项 (advanced)	允许使用选项配置其他 netty 选项作为前缀。例如：option.child.keepAlive=false 设置 netty 选项 child.keepAlive=false。有关可以使用的选项，请参阅 Netty 文档。		Map
receiveBufferSize (advanced)	在入站通信中使用的 TCP/UDP 缓冲区大小。大小为字节。	65536	long
receiveBufferSizePredictor (advanced)	配置缓冲区大小预测。请参阅 Jetty 文档和此邮件线程的详细信息。		int
sendBufferSize (advanced)	在出站通信中使用的 TCP/UDP 缓冲大小。大小为字节。	65536	long
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
transferExchange (advanced)	仅用于 TCP。您可以通过线线传输交换，而不是只传输正文。以下字段被传输： 在 body, Out body, fault body, In headers, Out headers, fault headers, Exchange properties, Exchange exception。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。	false	布尔值
allowDefaultCodec (codec)	如果两者都是 null，netty 组件会安装默认的 codec，则 encoder/decoder 为 null，文本为 false。将 allowDefaultCodec 设置为 false 可防止 netty 组件将默认 codec 安装为过滤器链中的第一个元素。	true	布尔值
autoAppendDelimiter (codec)	使用文本 codec 发送时是否自动附加缺少的结束分隔符。	true	布尔值
decoder (codec)	**弃用了** 一个自定义 ChannelHandler 类，可用于执行入站有效负载的特殊汇总。必须覆盖 org.jboss.netty.channel.ChannelUpStreamHandler。		ChannelHandler
decoderMaxLineLength (codec)	用于文本行 codec 的最大行长度。	1024	int
decoders (codec)	要使用的解码器列表。您可以使用以逗号分开的值的 String，并在 Registry 中查找值。只需记住使用 # 前缀，以便 Camel 知道它应该查找。		字符串

Name	描述	默认值	类型
delimiter (codec)	用于文本行代码的分隔符。可能的值有 LINE 和 NULL。	行	TextLineDelimiter
encoder (codec)	**弃用了** 一个自定义 ChannelHandler 类，可用于执行出站有效负载的特殊汇总。必须覆盖 org.jboss.netty.channel.ChannelDownStreamHandler 。		ChannelHandler
编码器 (codec)	要使用的编码器列表。您可以使用以逗号分开的值的 String，并在 Registry 中查找值。只需记住使用 # 前缀，以便 Camel 知道它应该查找。		字符串
编码 (codec)	用于文本编码的编码(charset 名称)。如果没有提供，Camel 将使用 JVM 默认 Charset。		字符串
文本行 (codec)	仅用于 TCP。如果没有指定 codec，您可以使用此标志来指示基于文本的 codec；如果没有指定，或者值为 false，则通过 TCP 假设 Object Serialization。	false	布尔值
enabledProtocols (security)	使用 SSL 时要启用的协议	TLSv1, TLSv1.1, TLSv1.2	字符串
keyStoreFile (security)	用于加密的客户端侧证书密钥存储		File
keyStoreFormat (security)	用于有效负载加密的密钥存储格式。如果没有设置，则默认为 JKS	JKS	字符串
keyStoreResource (security)	用于加密的客户端侧证书密钥存储。默认情况下从 classpath 加载，但您可以使用 classpath:、file: 或 http: 前缀来加载来自不同系统的资源。		字符串
needClientAuth (security)	配置服务器在使用 SSL 时是否需要客户端身份验证。	false	布尔值
密码短语 (security)	要使用 SSH 加密/解密有效负载的密码设置		字符串
securityProvider (security)	用于有效负载加密的安全供应商。如果没有设置，则默认为 SunX509。	SunX509	字符串
SSL (security)	设置以指定是否将 SSL 加密应用到此端点	false	布尔值
sslClientCertHeaders (security)	启用和 SSL 模式时，Netty 使用者将增强 Camel 消息，其中包含客户端证书的信息，如主题名称、签发者名称、序列号和有效日期范围。	false	布尔值

Name	描述	默认值	类型
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters
sslHandler (security)	引用可用于返回 SSL 处理程序的类		SslHandler
trustStoreFile (security)	用于加密的服务器端证书密钥存储		File
trustStoreResource (security)	用于加密的服务器端证书密钥存储。默认情况下从 classpath 加载，但您可以使用 classpath:、file: 或 http: 前缀来加载来自不同系统的资源。		字符串

240.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 70 个选项，如下所列。

Name	描述	默认值	类型
camel.component.netty.configuration.allow-default-codec	如果两者都是 null，netty 组件会安装默认的 codec，则 encoder/decoder 为 null，文本为 false。将 allowDefaultCodec 设置为 false 可防止 netty 组件将默认 codec 安装为过滤器链中的第一个元素。	true	布尔值
camel.component.netty.configuration.auto-append-delimiter	使用文本 codec 发送时是否自动附加缺少的结束分隔符。	true	布尔值
camel.component.netty.configuration.backlog	允许为 netty consumer (server)配置积压。请注意，积压只是取决于操作系统的最佳努力。将此选项设置为值（如 200、500 或 1000）告知 TCP 堆栈如果未配置此选项，则接受队列的时长，则积压取决于操作系统设置。		整数
camel.component.netty.configuration.boss-count	当 netty 适用于 nio 模式时，它使用 Netty 中的默认 bossCount 参数，即 1。用户可以使用此操作从 Netty 覆盖默认的 bossCount	1	整数
camel.component.netty.configuration.boss-pool	使用显式 org.jboss.netty.channel.socket.nio.BossPool 作为 boss 线程池。例如，要与多个使用者共享线程池。默认情况下，每个消费者都有自己的带有 1 个核心线程的 boss 池。		BossPool

Name	描述	默认值	类型
camel.component.netty.configuration.broadcast	将 设置为选择通过 UDP 的多播	false	布尔值
camel.component.netty.configuration.channel-group	使用显式 ChannelGroup。		ChannelGroup
camel.component.netty.configuration.client-mode	如果 clientMode 为 true，netty 使用者会将地址连接到 TCP 客户端。	false	布尔值
camel.component.netty.configuration.client-pipeline-factory	使用自定义 ClientPipelineFactory		ClientPipelineFactory
camel.component.netty.configuration.connect-timeout	等待套接字连接可用的时间。价值为 millis。	10000	Long
camel.component.netty.configuration.decoder-max-line-length	用于文本行 codec 的最大行长度。	1024	整数
camel.component.netty.configuration.decoders	要使用的解码器列表。您可以使用以逗号分开的值的 String，并在 Registry 中查找值。只需记住使用 # 前缀，以便 Camel 知道它应该查找。		list
camel.component.netty.configuration.delimiter	用于文本行代码的分隔符。可能的值有 LINE 和 NULL。		TextLineDelimiter
camel.component.netty.configuration.disconnect	使用后是否从 Netty Channel 断开连接（关闭）。可用于使用者和制作者。	false	布尔值
camel.component.netty.configuration.disconnect-on-no-reply	如果启用了同步，则此选项将指定 NettyConsumer（如果它应该断开连接），没有回复回来。	true	布尔值

Name	描述	默认值	类型
camel.component.netty.configuration.enabled-protocols	使用 SSL 时要启用的协议	TLSv1, TLSv1.1, TLSv1.2	字符串
camel.component.netty.configuration.encoders	要使用的编码器列表。您可以使用以逗号分开的值的 String，并在 Registry 中查找值。只需记住使用 # 前缀，以便 Camel 知道它应该查找。		list
camel.component.netty.configuration.encoding	用于文本编码的编码(charset 名称)。如果没有提供，Camel 将使用 JVM 默认 Charset。		字符串
camel.component.netty.configuration.host	主机名。对于消费者，主机名是 localhost 或 0.0.0.0，对于生成者，主机名是要连接的远程主机		字符串
camel.component.netty.configuration.keep-alive	设置以确保因为不活跃而不会关闭套接字	true	布尔值
camel.component.netty.configuration.key-store-format	用于有效负载加密的密钥存储格式。如果没有设置，则默认为 JKS	JKS	字符串
camel.component.netty.configuration.key-store-resource	用于加密的客户端侧证书密钥存储。默认情况下从 classpath 加载，但您可以使用 classpath:、file: 或 http: 前缀来加载来自不同系统的资源。		字符串
camel.component.netty.configuration.lazy-channel-creation	如果远程服务器在 Camel producer 启动时未启动并运行，则可以创建频道以避免异常。	true	布尔值
camel.component.netty.configuration.max-channel-memory-size	使用 orderedThreadPoolExecutor 时每个频道的最大排队事件总量。指定 0 来禁用。	10485760	Long
camel.component.netty.configuration.max-total-memory-size	使用 orderedThreadPoolExecutor 时，此池的最大排队事件总量。指定 0 来禁用。	209715200	Long

Name	描述	默认值	类型
camel.component.netty.configuration.need-client-auth	配置服务器在使用 SSL 时是否需要客户端身份验证。	false	布尔值
camel.component.netty.configuration.netty-server-bootstrap-factory	使用自定义 NettyServerBootstrapFactory		NettyServerBootstrap Factory
camel.component.netty.configuration.network-interface	在使用 UDP 时，可以使用此选项按名称指定网络接口，如 eth0 以加入多播组。		字符串
camel.component.netty.configuration.no-reply-log-level	如果启用了同步，这个选项会指示 NettyConsumer，在记录一个没有回复时要使用的日志记录级别。		LoggingLevel
camel.component.netty.configuration.options	允许使用选项配置其他 netty 选项作为前缀。例如：option.child.keepAlive=false 设置 netty 选项 child.keepAlive=false。有关可以使用的选项，请参阅 Netty 文档。		Map
camel.component.netty.configuration.ordered-thread-pool-executor	是否使用排序的线程池，确保同一通道上按顺序处理事件。如需了解更多详细信息，请参阅 org.jboss.netty.handler.execution.OrderedMemoryAwareThreadPoolExecutor 的 netty javadoc。	true	布尔值
camel.component.netty.configuration.passphrase	要使用 SSH 加密/解密有效负载的密码设置		字符串
camel.component.netty.configuration.port	主机端口号		整数
camel.component.netty.configuration.producer-pool-enabled	producer 池是否已启用。重要： 不要关闭此操作，因为处理并发和可靠的请求/回复需要池。	true	布尔值

Name	描述	默认值	类型
camel.component.netty.configuration.producer-pool-max-active	设置池可分配给客户端或闲置等待签出的对象数量上限。对没有限制，使用负值。	-1	整数
camel.component.netty.configuration.producer-pool-max-idle	设置池中空闲实例数量上限。	100	整数
camel.component.netty.configuration.producer-pool-min-evictable-idle	设置对象在空闲对象驱除有资格驱除前，对象可能会在池中闲置的最小时间（值为 millis））。	300000	Long
camel.component.netty.configuration.producer-pool-min-idle	设置制作者池中允许的最小实例数量，然后再驱除线程（如果活跃）生成新对象。		整数
camel.component.netty.configuration.protocol	要使用的协议，可以是 tcp 或 udp。		字符串
camel.component.netty.configuration.receive-buffer-size	在入站通信中使用的 TCP/UDP 缓冲区大小。大小为字节。	65536	Long
camel.component.netty.configuration.receive-buffer-size-predictor	配置缓冲区大小预测。请参阅 Jetty 文档和此邮件线程的详细信息。		整数
camel.component.netty.configuration.request-timeout	在调用远程服务器时，允许 Netty producer 使用超时。默认情况下，没有使用超时。该值在 milli 秒内，如 30000 为 30 秒。requestTimeout 使用 Netty 的 ReadTimeoutHandler 来触发超时。		Long
camel.component.netty.configuration.reuse-address	设置以方便套接字多路	true	布尔值

Name	描述	默认值	类型
camel.component.netty.configuration.security-provider	用于有效负载加密的安全供应商。如果没有设置，则默认为 SunX509。	SunX509	字符串
camel.component.netty.configuration.send-buffer-size	在出站通信中使用的 TCP/UDP 缓冲大小。大小为字节。	65536	Long
camel.component.netty.configuration.server-closed-channel-exception-caught-log-level	如果服务器(NettyConsumer)捕获 java.nio.channels.ClosedChannelException，然后使用此日志级别进行记录。这用于避免记录关闭的频道异常，因为客户端可以立即断开，然后在 Netty 服务器中造成大量关闭的异常。		LoggingLevel
camel.component.netty.configuration.server-exception-caught-log-level	如果服务器(NettyConsumer)捕获异常，则使用此日志级别进行日志记录。		LoggingLevel
camel.component.netty.configuration.server-pipeline-factory	使用自定义 ServerPipelineFactory		ServerPipelineFactory
camel.component.netty.configuration.ssl	设置以指定是否将 SSL 加密应用到此端点	false	布尔值
camel.component.netty.configuration.ssl-client-cert-headers	启用和 SSL 模式时，Netty 使用者将增强 Camel 消息，其中包含客户端证书的信息，如主题名称、签发者名称、序列号和有效日期范围。	false	布尔值
camel.component.netty.configuration.ssl-context-parameters	使用 SSLContextParameters 配置安全性		SSLContextParameters
camel.component.netty.configuration.ssl-handler	引用可用于返回 SSL 处理程序的类		SslHandler

Name	描述	默认值	类型
camel.component.netty.configuration.sync	将端点设置为单向或请求响应	true	布尔值
camel.component.netty.configuration.tcp-no-delay	设置以提高 TCP 协议性能	true	布尔值
camel.component.netty.configuration.textline	仅用于 TCP。如果没有指定 codec，您可以使用此标志来指示基于文本的 codec；如果没有指定，或者值为 false，则通过 TCP 假设 Object Serialization。	false	布尔值
camel.component.netty.configuration.transfer-exchange	仅用于 TCP。您可以通过线线传输交换，而不是只传输正文。以下字段被传输： 在 body, Out body, fault body, In headers, Out headers, fault headers, Exchange properties, Exchange exception。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。	false	布尔值
camel.component.netty.configuration.trust-store-resource	用于加密的服务器端证书密钥存储。默认情况下从 classpath 加载，但您可以使用 classpath:、file: 或 http: 前缀来加载来自不同系统的资源。		字符串
camel.component.netty.configuration.udp-connectionless-sending	这个选项支持连接较少的 udp 发送，这是实际触发和忘记的。如果没有端口侦听接收端口，则已连接的 udp 发送 PortUnreachableException。	false	布尔值
camel.component.netty.configuration.use-channel-buffer	如果 useChannelBuffer 为 true，则netty 生成者会在发送前将消息正文转换为 ChannelBuffer。	false	布尔值
camel.component.netty.configuration.worker-count	当 netty 适用于 nio 模式时，它会使用来自 Netty 的默认 workerCount 参数，即 cpu_core_threads2。用户可以使用此操作从 Netty 覆盖默认的 workerCount		整数
camel.component.netty.configuration.worker-pool	使用显式 org.jboss.netty.channel.socket.nio.WorkerPool 作为 worker 线程池。例如，要与多个使用者共享线程池。默认情况下，每个使用者都有自己的 worker 池，有 2 个 x cpu 数内核线程。		WorkerPool
camel.component.netty.enabled	启用 netty 组件	true	布尔值

Name	描述	默认值	类型
camel.component.netty.maximum-pool-size	排序线程池的核心池大小（如果使用）。默认值为 16。	16	整数
camel.component.netty.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.netty.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值
camel.component.netty.configuration.decoder	此自定义 ChannelHandler 类可用于执行入站有效负载的特殊汇总。必须覆盖 org.jboss.netty.channel.ChannelUpStreamHandler。		ChannelHandler
camel.component.netty.configuration.encoder	此自定义 ChannelHandler 类可用于执行出站有效负载的特殊汇总。必须覆盖 org.jboss.netty.channel.ChannelDownStreamHandler。		ChannelHandler
camel.component.netty.configuration.key-store-file	用于加密的客户端侧证书密钥存储		File
camel.component.netty.configuration.maximum-pool-size			整数
camel.component.netty.configuration.trust-store-file	用于加密的服务器端证书密钥存储		File

240.4. 基于 REGISTRY 的选项

codec Handlers 和 SSL Keystores 可以在 Registry 中列出，如 Spring XML 文件中。可以传递的值如下：

Name	描述
passphrase	要使用 SSH 加密/解密有效负载的密码设置

Name	描述
keyStoreFormat	用于有效负载加密的密钥存储格式。如果没有设置，则默认为 "JKS"
securityProvider	用于有效负载加密的安全供应商。如果没有设置，则默认为 "SunX509"。
keyStoreFile	**弃用：** 用于加密的客户端侧证书密钥存储
trustStoreFile	**弃用：** 用于加密的服务器侧证书密钥存储
keyStoreResource	Camel 2.11.1： 用于加密的客户端侧证书密钥存储。默认情况下从 classpath 加载，但您可以使用 **"classpath:"**, **"file:"** 或 **"http:"** 前缀来加载来自不同系统的资源。
trustStoreResource	Camel 2.11.1： 用于加密的服务器侧证书密钥存储。默认情况下从 classpath 加载，但您可以使用 **"classpath:"**, **"file:"** 或 **"http:"** 前缀来加载来自不同系统的资源。
sslHandler	引用可用于返回 SSL 处理程序的类
encoder	此自定义 **ChannelHandler** 类可用于执行出站有效负载的特殊汇总。必须覆盖 **org.jboss.netty.channel.ChannelDownStreamHandler**。
encorders	要使用的编码器列表。您可以使用以逗号分开的值的 String，并在 Registry 中查找值。只需记住使用 # 前缀，以便 Camel 知道它应该查找。
decoder	此自定义 **ChannelHandler** 类可用于执行入站有效负载的特殊汇总。必须覆盖 **org.jboss.netty.channel.ChannelUpStreamHandler**。
decoders	要使用的解码器列表。您可以使用以逗号分开的值的 String，并在 Registry 中查找值。只需记住使用 # 前缀，以便 Camel 知道它应该查找。

***重要：** 阅读有关使用不可共享编码器/解码器的信息。*

240.4.1. 使用不可共享编码器或解码器

如果您的编码器或解码器不可共享（例如，它们有 @Shareable 类注解），则您的编码器/解码器必须实施 org.apache.camel.component.netty.ChannelHandlerFactory 接口，并在 newChannelHandler 方法中返回新实例。这是为了确保可以安全地使用 encoder/decoder。否则，Netty 组件会在创建端点时记录 WARN。

Netty 组件提供了一个 org.apache.camel.component.netty.ChannelHandlerFactories 工厂类，它有多个常用的方法。

240.5. 将消息发送到/从 NETTY 端点发送

240.5.1. Netty Producer

在 Producer 模式中，组件提供使用 TCP 或 UDP 协议（具有可选 SSL 支持）将有效负载发送到套接字端点的功能。

producer 模式支持基于单向和请求响应的操作。

240.5.2. Netty Consumer

在 Consumer 模式中，组件提供以下功能：

- 使用 TCP 或 UDP 协议（具有可选 SSL 支持）侦听指定的套接字。
- 使用基于 text/xml、二进制和序列化对象的有效负载在套接字上接收请求，
- 在路由上将它们作为消息交换发送。

消费者模式支持基于单向和请求响应的操作。

240.6. HEADERS

以下标头填充了 Netty 使用者创建的交换：

标头键	类	描述
NettyConstants.NETTY_CHANNEL_HANDLER_CONTEXT / CamelNettyChannelHandlerContext	**org.jboss.netty.channel.ChannelHandlerContext**	'ChannelHandlerContext 'instance 与 netty 收到的连接相关联。
NettyConstants.NETTY_MESSAGE_EVENT / CamelNettyMessageEvent	**org.jboss.netty.channel.MessageEvent**	与 netty 收到的连接关联的 'MessageEvent 'instance。
NettyConstants.NETTY_REMOTE_ADDRESS / CamelNettyRemoteAddress	**java.net.SocketAddress**	传入套接字连接的远程地址。

标头键	类	描述
NettyConstants.NETTY_LOCAL_ADDRESS / CamelNettyLocalAddress	java.net.SocketAddress	传入套接字连接的本地地址。

240.7. 使用示例

240.7.1. 使用 Request-Reply 和 serialized 对象有效负载的 UDP Netty 端点

```
RouteBuilder builder = new RouteBuilder() {
  public void configure() {
    from("netty:udp://0.0.0.0:5155?sync=true")
      .process(new Processor() {
        public void process(Exchange exchange) throws Exception {
          Poetry poetry = (Poetry) exchange.getIn().getBody();
          poetry.setPoet("Dr. Sarojini Naidu");
          exchange.getOut().setBody(poetry);
        }
      }
    }
};
```

240.7.2. 使用单向通信基于 TCP 的 Netty consumer 端点

```
RouteBuilder builder = new RouteBuilder() {
  public void configure() {
      from("netty:tcp://0.0.0.0:5150")
        .to("mock:result");
  }
};
```

240.7.3. 使用 Request-Reply 通信基于 SSL/TCP 的 Netty 消费者端点

使用 JSSE 配置实用程序

自 Camel 2.9 起，Netty 组件通过 Camel JSSE 配置实用程序支持 SSL/TLS 配置。这个实用程序可

大大减少您需要写入的组件特定代码量，并在端点和组件级别进行配置。以下示例演示了如何将 实用程序与 Netty 组件一起使用。

组件的编程配置

```
KeyStoreParameters ksp = new KeyStoreParameters();
ksp.setResource("/users/home/server/keystore.jks");
ksp.setPassword("keystorePassword");

KeyManagersParameters kmp = new KeyManagersParameters();
kmp.setKeyStore(ksp);
kmp.setKeyPassword("keyPassword");

SSLContextParameters scp = new SSLContextParameters();
scp.setKeyManagers(kmp);

NettyComponent nettyComponent = getContext().getComponent("netty",
NettyComponent.class);
nettyComponent.setSslContextParameters(scp);
```

基于 Spring DSL 的端点配置

```
...
  <camel:sslContextParameters
     id="sslContextParameters">
    <camel:keyManagers
       keyPassword="keyPassword">
      <camel:keyStore
         resource="/users/home/server/keystore.jks"
         password="keystorePassword"/>
    </camel:keyManagers>
  </camel:sslContextParameters>...
...
  <to uri="netty:tcp://0.0.0.0:5150?
sync=true&ssl=true&sslContextParameters=#sslContextParameters"/>
...
```

在 Jetty 组件上使用基本 SSL/TLS 配置

```
JndiRegistry registry = new JndiRegistry(createJndiContext());
registry.bind("password", "changeit");
registry.bind("ksf", new File("src/test/resources/keystore.jks"));
registry.bind("tsf", new File("src/test/resources/keystore.jks"));

context.createRegistry(registry);
context.addRoutes(new RouteBuilder() {
  public void configure() {
     String netty_ssl_endpoint =
```

```
        "netty:tcp://0.0.0.0:5150?sync=true&ssl=true&passphrase=#password"
        + "&keyStoreFile=#ksf&trustStoreFile=#tsf";
    String return_string =
        "When You Go Home, Tell Them Of Us And Say,"
        + "For Your Tomorrow, We Gave Our Today.";

    from(netty_ssl_endpoint)
     .process(new Processor() {
        public void process(Exchange exchange) throws Exception {
          exchange.getOut().setBody(return_string);
        }
      }
  }
});
```

访问 SSLSession 和客户端证书

从 Camel 2.12 开始提供

如果需要获取客户端证书的详细信息，您可以访问 javax.net.ssl.SSLSession。当 ssl=true 随后，Netty 组件将 SSLSession 作为标头存储在 Camel Message 上，如下所示： ???

```
SSLSession session = exchange.getIn().getHeader(NettyConstants.NETTY_SSL_SESSION,
SSLSession.class);
// get the first certificate which is client certificate
javax.security.cert.X509Certificate cert = session.getPeerCertificateChain()[0];
Principal principal = cert.getSubjectDN();
```

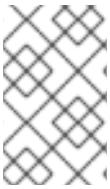
请记住，要设置 needClientAuth=true 以验证客户端，否则 SSLSession 无法访问客户端证书的信息，您可能会收到一个例外 javax.net.ssl.SSLPeerUnverifiedException: peer not authenticated。如果客户端证书过期或无效，您可能还会获得这个异常。

提示

选项 sslClientCertHeaders 可以设置为 true，然后使用带有客户端证书详情的标头增强 Camel 消息。例如，标题 CamelNettySSLClientCertSubjectName 中可随时提供主题名称。

240.7.4. 使用多个代码cs

在某些情况下，可能需要将编码器和解码器链添加到 netty 管道。要将多个 codecs 添加到 camel netty 端点中，应使用 'encoders' 和 'decoders' uri 参数。与 'encoder' 和 'decoder' 参数类似，用于提供应添加到管道的 ChannelUpstreamHandlers 和 ChannelDownstreamHandlers 列表。请注意，如果指定了编码器，则忽略 encoder 参数，类似于解码器和解码器参数。



注意

请参阅上面有关使用不可共享编码器/解码器的进一步的信息。

codecs 列表需要添加到 Camel 的 registry 中，以便在创建端点时解析它们。

Spring 的原生集合支持可用于在应用程序上下文中指定 codec 列表

然后，在 netty 端点定义中使用 bean 名称作为逗号分隔的列表或包含在 List 中，例如：

或通过 spring。

240.8. 完成时关闭频道

当充当服务器时，有时希望在关闭频道时（例如，客户端转换已完成）。
您可以通过设置 endpoint 选项 disconnect=true 来完成此操作。

但是，您也可以根据如下消息指示 Camel：
要指示 Camel 关闭频道，您应该添加一个带有键 CamelNettyCloseChannelWhenComplete 设置为布尔值 true 的标头。
例如，以下示例会在将消息写入客户端后关闭频道：

```
from("netty:tcp://0.0.0.0:8080").process(new Processor() {
    public void process(Exchange exchange) throws Exception {
        String body = exchange.getIn().getBody(String.class);
        exchange.getOut().setBody("Bye " + body);
        // some condition which determines if we should close
        if (close) {

exchange.getOut().setHeader(NettyConstants.NETTY_CLOSE_CHANNEL_WHEN_COMPLETE,
true);
        }
    }
});
```

240.9. 添加自定义频道管道工厂，以获得对创建的管道的完整控制

从 Camel 2.5 开始提供

自定义频道管道通过插入自定义处理程序、编码器和解码器，而无需以非常简单的方式在 Netty Endpoint URL 中指定它们，从而对用户提供完全的控制。

要添加自定义管道，必须通过上下文 registry (JNDIRegistry 或 camel-spring ApplicationContextRegistry 等)创建并注册自定义频道管道工厂。

自定义管道工厂必须构建如下

- Producer 链接的频道管道工厂必须扩展 abstract 类 ClientPipelineFactory。
- 消费者链接的频道管道工厂必须扩展 abstract 类 ServerPipelineFactory。
- 该类应覆盖 getPipeline （）方法，以插入自定义处理程序、编码器和解码器。不覆盖 getPipeline （）方法，创建没有处理程序、编码器或解码器到管道的管道。

以下示例显示如何创建 ServerChannel Pipeline 工厂

使用自定义管道工厂

```
public class SampleServerChannelPipelineFactory extends ServerPipelineFactory {
    private int maxLineSize = 1024;

    public ChannelPipeline getPipeline() throws Exception {
        ChannelPipeline channelPipeline = Channels.pipeline();

        channelPipeline.addLast("encoder-SD", new StringEncoder(CharsetUtil.UTF_8));
        channelPipeline.addLast("decoder-DELIM", new
DelimiterBasedFrameDecoder(maxLineSize, true, Delimiters.lineDelimiter()));
        channelPipeline.addLast("decoder-SD", new StringDecoder(CharsetUtil.UTF_8));
        // here we add the default Camel ServerChannelHandler for the consumer, to allow Camel
to route the message etc.
        channelPipeline.addLast("handler", new ServerChannelHandler(consumer));

        return channelPipeline;
    }
}
```

然后，自定义频道管道工厂可以添加到 registry 中，并使用以下方法在 camel 路由上实例化/使用

```
Registry registry = camelContext.getRegistry();
serverPipelineFactory = new TestServerChannelPipelineFactory();
registry.bind("spf", serverPipelineFactory);
context.addRoutes(new RouteBuilder() {
  public void configure() {
     String netty_ssl_endpoint =
       "netty:tcp://0.0.0.0:5150?serverPipelineFactory=#spf"
     String return_string =
       "When You Go Home, Tell Them Of Us And Say,"
       + "For Your Tomorrow, We Gave Our Today.";

     from(netty_ssl_endpoint)
      .process(new Processor() {
        public void process(Exchange exchange) throws Exception {
          exchange.getOut().setBody(return_string);
        }
      }
  }
});
```

240.10. 重新使用 NETTY BOSS 和 WORKER 线程池

从 Camel 2.12 开始提供

Netty 有两种类型的线程池： boss 和 worker。默认情况下，每个 Netty 使用者和生成者都有自己的专用线程池。如果要在多个消费者或生成者之间重复使用这些线程池，必须在 Registry 中创建并放入线程池。

例如，使用 Spring XML，我们可以使用带有 2 个 worker 线程的 NettyWorkerPoolBuilder 创建共享 worker 线程池，如下所示：

```
<!-- use the worker pool builder to help create the shared thread pool -->
<bean id="poolBuilder" class="org.apache.camel.component.netty.NettyWorkerPoolBuilder">
  <property name="workerCount" value="2"/>
</bean>

<!-- the shared worker thread pool -->
<bean id="sharedPool" class="org.jboss.netty.channel.socket.nio.WorkerPool"
      factory-bean="poolBuilder" factory-method="build" destroy-method="shutdown">
</bean>
```

提示

对于 boss 线程池，有 Netty producers 的 org.apache.camel.component.netty.NettyServerBossPoolBuilder 构建器，以及 Netty producers 的 org.apache.camel.component.netty.NettyClientBossPoolBuilder。

然后，在 Camel 路由中，您可以通过在 URI 中配置 workerPool 选项来引用此 worker 池，如下所示：

```
<route>
  <from uri="netty:tcp://0.0.0.0:5021?
textline=true&amp;sync=true&amp;workerPool=#sharedPool&amp;orderedThreadPoolExecutor=false"
/>
  <to uri="log:result"/>
  ...
</route>
```

如果我们还有另一个路由，我们可以引用共享 worker 池：

```
<route>
  <from uri="netty:tcp://0.0.0.0:5022?
textline=true&amp;sync=true&amp;workerPool=#sharedPool&amp;orderedThreadPoolExecutor=false"
/>
  <to uri="log:result"/>
  ...
</route>
```

i. 以此类推。

240.11. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- Netty HTTP
- MINA

第 241 章 NETTY HTTP 组件（已弃用）

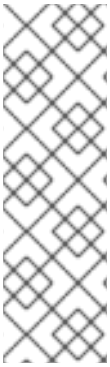
从 Camel 版本 2.12 开始提供

netty-http 组件是 Netty 组件的扩展，用于通过 Netty 协助 HTTP 传输。

此 camel 组件支持生成者和消费者端点。

警告

此组件已弃用。您应使用 Netty4 HTTP。



注意

流.Netty 是基于流的，这意味着它收到的输入作为流提交给 Camel。这意味着您只能够读取一次流的内容。如果您发现了一个情况，消息正文显示为为空，或者您需要多次访问数据（例如：执行多播或重新发送错误处理），您应该使用流缓存或将消息正文转换为 String，这会安全地重新读取多次。注意 Netty4 HTTP 使用 io.netty.handler.codec.http.HttpObjectAggregator 将整个流读入内存中，以构建整个完整 http 消息。但是，生成的消息仍然是基于流的消息，该消息一次是可读的。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-netty-http</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

241.1. URI 格式

netty 组件的 URI 方案如下

```
netty-http:http://0.0.0.0:8080[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

注意

查询参数和端点选项.您可以了解 Camel 如何识别 URI 查询参数和端点选项。例如，您可以创建端点 URI，如下所示 - netty-http:http//example.com?myParam=myValue&compression=true。在本例中，myParam 是 HTTP 参数，而 压缩 是 Camel 端点选项。在这种情况下，Camel 使用的策略是解决可用的端点选项并将其从 URI 中删除。这意味着，对于所讨论的示例，Netty HTTP producer 发送的 HTTP 请求将如下所示 - http//example.com?myParam=myValue，因为 压缩 端点选项将从目标 URL 解析并删除。请记住，您无法使用动态标头（如 CamelHttpQuery）指定端点选项。端点选项只能在端点 URI 定义级别（如 to 或 DSL 元素）指定。

241.2. HTTP 选项

更多选项

此组件继承 Netty 的所有选项。因此，请务必查看 Netty 文档。
请注意，在使用这个 Netty HTTP 组件时，Netty 中的一些选项不适用，如与 UDP 传输相关的选项。

Netty HTTP 组件支持 7 个选项，如下所列。

Name	描述	默认值	类型
nettyHttpBinding (advanced)	使用自定义 org.apache.camel.component.netty.http.NettyHttpBinding 来绑定到 Netty 和 Camel Message API。		NettyHttpBinding
configuration (common)	在创建端点时，使用 NettyConfiguration 作为配置。		NettyHttpConfiguration
headerFilterStrategy (advanced)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 来过滤标头。		HeaderFilterStrategy
securityConfiguration (security)	指的是 org.apache.camel.component.netty.http.NettyHttpSecurityConfiguration，用于配置安全 Web 资源。		NettyHttpSecurity 配置

Name	描述	默认值	类型
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
maximumPoolSize (advanced)	排序线程池的核心池大小（如果使用）。默认值为 16。	16	int
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Netty HTTP 端点使用 URI 语法进行配置：

```
netty-http:protocol:host:port/path
```

使用以下路径和查询参数：

241.2.1. 路径参数(4 参数)：

Name	描述	默认值	类型
protocol	**必需** 使用的协议是 http 或 https		字符串
主机	当作为消费者时，**需要** 本地主机名，如 localhost 或 0.0.0.0。使用制作者时的远程 HTTP 服务器主机名。		字符串
port	主机端口号		int
path	资源路径		字符串

241.2.2. 查询参数(78 参数)：

Name	描述	默认值	类型

Name	描述	默认值	类型
bridgeEndpoint (common)	如果选项为 true，则生成者将忽略 Exchange.HTTP_URI 标头，并使用端点的 URI 进行请求。您也可以将 throwExceptionOnFailure 设置为 false，以便生成者发送所有错误响应。在网桥模式下工作的用户将跳过 gzip 压缩和 WWW URL 表单编码（通过添加 Exchange.SKIP_ENCODING 和 Exchange.SKIP_WWW_FORM_URLENCODED 标头到已消耗的交换）。	false	布尔值
disconnect (common)	使用后是否从 Netty Channel 断开连接（关闭）。可用于使用者和制作者。	false	布尔值
keepalive (common)	设置以确保因为不活跃而不会关闭套接字	true	布尔值
reuseAddress (common)	设置以方便套接字多路	true	布尔值
sync (common)	将端点设置为单向或请求响应	true	布尔值
tcpNoDelay (common)	设置以提高 TCP 协议性能	true	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
matchOnUriPrefix (consumer)	如果没有找到完全匹配，Camel 是否应该尝试通过匹配 URI 前缀来查找目标消费者。	false	布尔值
send503whenSuspended (consumer)	在使用者被暂停时，是否发送回 HTTP 状态代码 503。如果选项为 false，则 Netty Acceptor 在消费者被暂停时处于未绑定状态，因此客户端无法再连接。	true	布尔值
backlog (consumer)	允许为 netty consumer (server)配置积压。请注意，积压只是取决于操作系统的最佳努力。将此选项设置为值（如 200、500 或 1000）告知 TCP 堆栈如果未配置此选项，则接受队列的时长，则积压取决于操作系统设置。		int
bossCount (consumer)	当 netty 适用于 nio 模式时，它使用 Netty 中的默认 bossCount 参数，即 1。用户可以使用此操作从 Netty 覆盖默认的 bossCount	1	int

Name	描述	默认值	类型
bossPool (consumer)	使用显式 org.jboss.netty.channel.socket.nio.BossPool 作为 boss 线程池。例如，要与多个使用者共享线程池。默认情况下，每个消费者都有自己的带有 1 个核心线程的 boss 池。		BossPool
channelGroup (consumer)	使用显式 ChannelGroup。		ChannelGroup
chunkedMaxContentLength (consumer)	值（以字节为单位）在 Netty HTTP 服务器上接收的每个块帧的最大内容长度（以字节为单位）。	1048576	int
compression (consumer)	如果客户端从 HTTP 标头支持，则允许使用 gzip/定义在 Netty HTTP 服务器上压缩。	false	布尔值
disableStreamCache (consumer)	确定来自 Netty HttpRequest disastergetContent（）的原始输入流是否被缓存(Camel 将把流读取到基于轻量级内存的流缓存)缓存中。默认情况下，Camel 将缓存 Netty 输入流，以支持多次读取，以确保其 Camel 可以从流检索所有数据。但是，当您需要访问原始流时，您可以将此选项设置为 true，例如将其直接流传输到文件或其他持久性存储。请注意，如果您启用这个选项，则无法多次读取 Netty 流，您需要手动重置 Netty 原始流上的读取器索引。	false	布尔值
disconnectOnNoReply (consumer)	如果启用了同步，则此选项将指定 NettyConsumer（如果它应该断开连接），没有回复回来。	true	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
httpMethodRestrict (consumer)	要禁用 Netty HTTP 使用者上的 HTTP 方法。您可以指定用逗号分开的多个。		字符串
mapHeaders (consumer)	如果启用了这个选项，那么在从 Netty 绑定到 Camel Message 的过程中，标头也会被映射（如作为标头添加到 Camel 消息）。您可以关闭这个选项来禁用这个选项。标头仍然可以从 org.apache.camel.component.netty.http.NettyHttpMessage 消息访问，该消息返回 Netty HTTP 请求 org.jboss.netty.handler.codec.http.HttpRequest 实例。	true	布尔值

Name	描述	默认值	类型
maxChannelMemorySize (consumer)	使用 orderedThreadPoolExecutor 时每个频道的最大排队事件总量。指定 0 来禁用。	1048576	long
maxHeaderSize (consumer)	所有标头的最大长度。如果每个标头的长度总和超过这个值，则会引发 TooLongFrameException。	8192	int
maxTotalMemorySize (consumer)	使用 orderedThreadPoolExecutor 时，此池的最大排队事件总量。指定 0 来禁用。	209715200	long
nettyServerBootstrapFactory (consumer)	使用自定义 NettyServerBootstrapFactory		NettyServerBootstrap Factory
nettySharedHttpServer (consumer)	使用共享的 Netty HTTP 服务器。如需了解更多详细信息，请参阅 Netty HTTP 服务器示例。		NettySharedHttpServer
noReplyLogLevel (consumer)	如果启用了同步，这个选项会指示 NettyConsumer，在记录一个没有回复时要使用的日志记录级别。	WARN	LoggingLevel
orderedThreadPoolExecutor (consumer)	是否使用排序的线程池，确保同一通道上按顺序处理事件。如需了解更多详细信息，请参阅 org.jboss.netty.handler.execution.OrderedMemoryAwareThreadPoolExecutor 的 netty javadoc。	true	布尔值
serverClosedChannelExceptionCaughtLogLevel (consumer)	如果服务器(NettyConsumer)捕获 java.nio.channels.ClosedChannelException，然后使用此日志级别进行记录。这用于避免记录关闭的频道异常，因为客户端可以立即断开，然后在 Netty 服务器中造成大量关闭的异常。	DEBUG	LoggingLevel
serverExceptionCaughtLog Level (consumer)	如果服务器(NettyConsumer)捕获异常，则使用此日志级别进行日志记录。	WARN	LoggingLevel
serverPipelineFactory (consumer)	使用自定义 ServerPipelineFactory		ServerPipelineFactory
traceEnabled (consumer)	指定是否为这个 Netty HTTP 使用者启用 HTTP TRACE。默认情况下关闭 TRACE。	false	布尔值
urlDecodeHeaders (consumer)	如果启用了这个选项，那么在从 Netty 绑定到 Camel Message 的过程中，标头值将被解码（例如 %20 将是一个空格字符）。注意此选项由默认的 org.apache.camel.component.netty.http.NettyHttpBinding 使用，因此如果您实施自定义 org.apache.camel.component.netty.http.NettyHttpBinding，则需要相应地将标头解码到此选项。	false	布尔值

Name	描述	默认值	类型
workerCount (consumer)	当 netty 适用于 nio 模式时，它会使用来自 Netty 的默认 workerCount 参数，即 cpu_core_threads2。用户可以使用此操作从 Netty 覆盖默认的 workerCount		int
workerPool (consumer)	使用显式 org.jboss.netty.channel.socket.nio.WorkerPool 作为 worker 线程池。例如，要与多个使用者共享线程池。默认情况下，每个使用者都有自己的 worker 池，有 2 个 x cpu 数内核线程。		WorkerPool
connectTimeout (producer)	等待套接字连接可用的时间。价值为 millis。	10000	long
requestTimeout (producer)	在调用远程服务器时，允许 Netty producer 使用超时。默认情况下，没有使用超时。该值在 milli 秒内，如 30000 为 30 秒。requestTimeout 使用 Netty 的 ReadTimeoutHandler 来触发超时。		long
throwExceptionOnFailure (producer)	如果来自远程服务器的失败响应，用于禁用抛出 HttpOperationFailedException。这样，您可以获取所有响应，而不考虑 HTTP 状态代码。	true	布尔值
clientPipelineFactory (producer)	使用自定义 ClientPipelineFactory		ClientPipelineFactory
lazyChannelCreation (producer)	如果远程服务器在 Camel producer 启动时未启动并运行，则可以创建频道以避免异常。	true	布尔值
okStatusCodeRange (producer)	被视为成功响应的状态代码。值包含。可以定义多个范围，用逗号分开，例如 200-204,209,301-304。每个范围必须是单个数字，或从 到，其中包含短划线。默认范围为 200-299	200-299	字符串
producerPoolEnabled (producer)	producer 池是否已启用。重要： 不要关闭此操作，因为处理并发和可靠的请求/回复需要池。	true	布尔值
producerPoolMaxActive (producer)	设置池可分配给客户端或闲置等待签出的对象数量上限。对没有限制，使用负值。	-1	int
producerPoolMaxIdle (producer)	设置池中空闲实例数量上限。	100	int
producerPoolMinEvictable Idle (producer)	设置对象在空闲对象驱除有资格驱除前，对象可能会在池中闲置的最小时间（值为 millis））。	300000	long

Name	描述	默认值	类型
producerPoolMinIdle (producer)	设置制作者池中允许的最小实例数量，然后再驱除线程（如果活跃）生成新对象。		int
useChannelBuffer (producer)	如果 useChannelBuffer 为 true，则netty 生成者会在发送前将消息正文转换为 ChannelBuffer。	false	布尔值
useRelativePath (producer)	设置是否在 HTTP 请求中使用相对路径。有些第三方后端系统（如 IBM Datapower）不支持 HTTP POST 中的绝对 URI，并将这个选项设置为 true 可以解决这个问题。	false	布尔值
bootstrapConfiguration (advanced)	使用自定义配置的 NettyServerBootstrapConfiguration 来配置此端点。		NettyServerBootstrap Configuration
配置 （高级）	使用自定义配置的 NettyHttpConfiguration 来配置此端点。		NettyHttpConfiguration
headerFilterStrategy (advanced)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 来过滤标头。		HeaderFilterStrategy
nettyHttpBinding (advanced)	使用自定义 org.apache.camel.component.netty.http.NettyHttpBinding 来绑定到 Netty 和 Camel Message API。		NettyHttpBinding
选项 (advanced)	允许使用选项配置其他 netty 选项作为前缀。例如：option.child.keepAlive=false 设置 netty 选项 child.keepAlive=false。有关可以使用的选项，请参阅 Netty 文档。		Map
receiveBufferSize (advanced)	在入站通信中使用的 TCP/UDP 缓冲区大小。大小为字节。	65536	long
receiveBufferSizePredictor (advanced)	配置缓冲区大小预测。请参阅 Jetty 文档和此邮件线程的详细信息。		int
sendBufferSize (advanced)	在出站通信中使用的 TCP/UDP 缓冲大小。大小为字节。	65536	long
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

Name	描述	默认值	类型
transferException (advanced)	如果在消费者端启用并交换失败处理，如果原因 Exception 在响应中作为 application/x-java-serialized-object 内容类型发送序列化，则结果为 application/x-java-serialized-object 内容类型。在生产者侧，异常将反序列化并丢弃为原样，而不是 HttpOperationFailedException。原因异常需要按顺序处理。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
transferExchange (advanced)	仅用于 TCP。您可以通过线线传输交换，而不是只传输正文。以下字段被传输： 在 body, Out body, fault body, In headers, Out headers, fault headers, Exchange properties, Exchange exception。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。	false	布尔值
decoder (codec)	**弃用** 以使用单个解码器。这个选项已弃用，改为使用 encoders。		ChannelHandler
decoders (codec)	要使用的解码器列表。您可以使用以逗号分开的值的 String，并在 Registry 中查找值。只需记住使用 # 前缀，以便 Camel 知道它应该查找。		字符串
encoder (codec)	**弃用** 以使用单个编码器。这个选项已弃用，改为使用 encoders。		ChannelHandler
编码器 (codec)	要使用的编码器列表。您可以使用以逗号分开的值的 String，并在 Registry 中查找值。只需记住使用 # 前缀，以便 Camel 知道它应该查找。		字符串
enabledProtocols (security)	使用 SSL 时要启用的协议	TLSv1, TLSv1.1, TLSv1.2	字符串
keyStoreFile (security)	用于加密的客户端侧证书密钥存储		File
keyStoreFormat (security)	用于有效负载加密的密钥存储格式。如果没有设置，则默认为 JKS	JKS	字符串
keyStoreResource (security)	用于加密的客户端侧证书密钥存储。默认情况下从 classpath 加载，但您可以使用 classpath:、file: 或 http: 前缀来加载来自不同系统的资源。		字符串
needClientAuth (security)	配置服务器在使用 SSL 时是否需要客户端身份验证。	false	布尔值
密码短语 (security)	要使用 SSH 加密/解密有效负载的密码设置		字符串

Name	描述	默认值	类型
securityConfiguration (security)	指的是 org.apache.camel.component.netty.http.NettyHttpSecurityConfiguration，用于配置安全 Web 资源。		NettyHttpSecurity 配置
securityOptions (security)	使用映射中的键/值对配置 NettyHttpSecurityConfiguration		Map
securityProvider (security)	用于有效负载加密的安全供应商。如果没有设置，则默认为 SunX509。	SunX509	字符串
SSL (security)	设置以指定是否将 SSL 加密应用到此端点	false	布尔值
sslClientCertHeaders (security)	启用和 SSL 模式时，Netty 使用者将增强 Camel 消息，其中包含客户端证书的信息，如主题名称、签发者名称、序列号和有效日期范围。	false	布尔值
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters
sslHandler (security)	引用可用于返回 SSL 处理程序的类		SslHandler
trustStoreFile (security)	用于加密的服务器端证书密钥存储		File
trustStoreResource (security)	用于加密的服务器端证书密钥存储。默认情况下从 classpath 加载，但您可以使用 classpath:、file: 或 http: 前缀来加载来自不同系统的资源。		字符串

241.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 31 个选项，如下所列。

Name	描述	默认值	类型
camel.component.netty-http.configuration.allow-default-codec			布尔值

Name	描述	默认值	类型
camel.component.netty-http.configuration.bridge-endpoint	如果选项为 true，则生成者将忽略 Exchange.HTTP_URI 标头，并使用端点的 URI 进行请求。您也可以将 throwExceptionOnFailure 设置为 false，以便生成者发送所有错误响应。在网桥模式下工作的用户将跳过 gzip 压缩和 WWW URL 表单编码（通过添加 Exchange.SKIP_ENCODING 和 Exchange.SKIP_WWW_FORM_URLENCODED 标头到已消耗的交换）。	false	布尔值
camel.component.netty-http.configuration.chunked-max-content-length	值（以字节为单位）在 Netty HTTP 服务器上接收的每个块帧的最大内容长度（以字节为单位）。	1048576	整数
camel.component.netty-http.configuration.compression	如果客户端从 HTTP 标头支持，则允许使用 gzip/定义在 Netty HTTP 服务器上压缩。	false	布尔值
camel.component.netty-http.configuration.disable-stream-cache	确定来自 Netty HttpRequest disastergetContent（）的原始输入流是否被缓存(Camel 将把流读取到基于轻量级内存的流缓存)缓存中。默认情况下，Camel 将缓存 Netty 输入流，以支持多次读取，以确保其 Camel 可以从流检索所有数据。但是，当您需要访问原始流时，您可以将此选项设置为 true，例如将其直接流传输到文件或其他持久性存储。请注意，如果您启用这个选项，则无法多次读取 Netty 流，您需要手动重置 Netty 原始流上的读取器索引。	false	布尔值
camel.component.netty-http.configuration.host	作为消费者，本地主机名，如 localhost, 或 0.0.0.0。使用制作者时的远程 HTTP 服务器主机名。		字符串
camel.component.netty-http.configuration.map-headers	如果启用了这个选项，那么在从 Netty 绑定到 Camel Message 的过程中，标头也会被映射（如作为标头添加到 Camel 消息）。您可以关闭这个选项来禁用这个选项。标头仍然可以从 org.apache.camel.component.netty.http.NettyHttpMessage 消息访问，该消息返回 Netty HTTP 请求 org.jboss.netty.handler.codec.http.HttpRequest 实例。	true	布尔值
camel.component.netty-http.configuration.match-on-uri-prefix	如果没有找到完全匹配，Camel 是否应该尝试通过匹配 URI 前缀来查找目标消费者。	false	布尔值

Name	描述	默认值	类型
camel.component.netty-http.configuration.max-header-size	所有标头的最大长度。如果每个标头的长度总和超过这个值，则会引发 TooLongFrameException。	8192	整数
camel.component.netty-http.configuration.ok-status-code-range	被视为成功响应的状态代码。值包含。可以定义多个范围，用逗号分开，例如 200-204,209,301-304。每个范围必须是单个数字，或从 到，其中包含短划线。默认范围为 200-299	200-299	字符串
camel.component.netty-http.configuration.path	资源路径		字符串
camel.component.netty-http.configuration.port	端口号。http 为 80，https 为 443。		整数
camel.component.netty-http.configuration.protocol	使用的协议(http 或 https)		字符串
camel.component.netty-http.configuration.send503when-suspended	在使用者被暂停时，是否发送回 HTTP 状态代码 503。如果选项为 false，则 Netty Acceptor 在消费者被暂停时处于未绑定状态，因此客户端无法再连接。	true	布尔值
camel.component.netty-http.configuration.throw-exception-on-failure	如果来自远程服务器的失败响应，用于禁用抛出 HttpOperationFailedException。这样，您可以获取所有响应，而不考虑 HTTP 状态代码。	true	布尔值
camel.component.netty-http.configuration.transfer-exception	如果在消费者端启用并交换失败处理，如果原因 Exception 在响应中作为 application/x-java-serialized-object 内容类型发送序列化，则结果为 application/x-java-serialized-object 内容类型。在生产者侧，异常将反序列化并丢弃为原样，而不是 HttpOperationFailedException。原因异常需要按顺序处理。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值

Name	描述	默认值	类型
camel.component.netty-http.configuration.url-decode-headers	如果启用了这个选项，那么在从 Netty 绑定到 Camel Message 的过程中，标头值将被解码（例如 %20 将是一个空格字符）。注意此选项由默认的 org.apache.camel.component.netty.http.NettyHttpBinding 使用，因此如果您实施自定义 org.apache.camel.component.netty.http.NettyHttpBinding，则需要相应地将标头解码到此选项。	false	布尔值
camel.component.netty-http.configuration.use-relative-path	设置是否在 HTTP 请求中使用相对路径。有些第三方后端系统（如 IBM Datapower）不支持 HTTP POST 中的绝对 URI，并将这个选项设置为 true 可以解决这个问题。	false	布尔值
camel.component.netty-http.enabled	启用 netty-http 组件	true	布尔值
camel.component.netty-http.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 来过滤标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		字符串
camel.component.netty-http.maximum-pool-size	排序线程池的核心池大小（如果使用）。默认值为 16。	16	整数
camel.component.netty-http.netty-http-binding	使用自定义 org.apache.camel.component.netty.http.NettyHttpBinding 来绑定到 Netty 和 Camel Message API。选项是 org.apache.camel.component.netty.http.NettyHttpBinding 类型。		字符串
camel.component.netty-http.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.netty-http.security-configuration.authenticate	是否启用身份验证 <p/>，这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.component.netty-http.security-configuration.constraint	支持的 restricted. <p/> 当前只支持 Basic。		字符串
camel.component.netty-http.security-configuration.login-denied-logging-level	设置用于日志记录被拒绝的登录尝试(incl stacktraces) <p/> （默认为 DEBUG）的日志记录级别。		LoggingLevel
camel.component.netty-http.security-configuration.realm	设置要使用的域的名称。		字符串
camel.component.netty-http.security-configuration.role-class-name			字符串
camel.component.netty-http.security-configuration.security-authenticator	设置 {@link SecurityAuthenticator}，用于验证 {@link HttpPrincipal}。		SecurityAuthenticator
camel.component.netty-http.security-configuration.security-constraint	将一个 {@link SecurityConstraint} 设置为用来检查 Web 资源是否受限制或没有 <p/>，默认为 <tt>null</tt>，这意味着所有资源都受到限制。		SecurityConstraint
camel.component.netty-http.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

241.4. 消息标头

以下标头可用于制作者来控制 HTTP 请求。

Name	类型	描述
CamelHttpMethod	**字符串**	允许控制使用什么 HTTP 方法，如 GET、POST 和 TRACE 等。类型也可以是 **org.jboss.netty.handler.codec.http.HttpMethod** 实例。
CamelHttpQuery	**字符串**	允许将 URI 查询参数作为 **String** 值提供，用于覆盖端点配置。使用 & 符号分隔多个参数。例如： **foo=bar&beer=yes**。
CamelHttpPath	**字符串**	Camel 2.13.1/2.12.4 ： 允许提供 URI context-path 和查询参数作为**String** 值，用于覆盖端点配置。这允许重复使用同一制作者来调用同一远程 http 服务器，但使用动态上下文路径和查询参数。
Content-Type	**字符串**	设置 HTTP 正文的内容类型：例如： **text/plain; charset="UTF-8"。**
CamelHttpResponseCode	**int**	允许设置要使用的 HTTP 状态代码。默认情况下，200 用于成功，使用 500 个失败。

当路由从 Netty HTTP 端点开始时，以下标头作为 meta-data 提供：

表中的描述在带有的路由中使用偏移量： from ("netty-http:http:0.0.0.0:8080/myapp")...

Name	类型	描述
CamelHttpMethod	**字符串**	使用的 HTTP 方法，如 GET、POST 和 TRACE 等。
CamelHttpUrl	**字符串**	URL，包括协议、主机和端口等
CamelHttpUri	**字符串**	没有协议、主机和端口的 URI，等等

Name	类型	描述
CamelHttpQuery	**字符串**	任何查询参数，如 **foo=bar&beer=yes**
CamelHttpRawQuery	**字符串**	Camel 2.13.0: 任何查询参数，如 **foo=bar&beer=yes**。以原始形式存储，因为它们到达消费者（例如，在 URL 解码之前）。
CamelHttpPath	**字符串**	其他 context-path。如果名为 context-path **/myapp** 的客户端，则该值为空。如果客户端调用 **/myapp/mystuff**，则此标题值为 **/mystuff**。换句话说，在路由端点上配置的 context-path 后，其值代表。
CamelHttpCharacterEncoding	**字符串**	content-type 标头中的 charset。
CamelHttpAuthentication	**字符串**	如果用户是使用 HTTP Basic 进行身份验证的，则使用值 **Basic** 添加此标头。
Content-Type	**字符串**	如果提供，内容类型。例如： **text/plain; charset="UTF-8"。**

241.5. 访问 NETTY 类型

此组件使用 org.apache.camel.component.netty.http.NettyHttpMessage 作为 Exchange 上的消息实施。这样，最终用户可以根据需要访问原始的 Netty 请求/响应实例，如下所示。请注意，原始响应可能无法始终访问。

```
org.jboss.netty.handler.codec.http.HttpRequest request =
exchange.getIn(NettyHttpMessage.class).getHttpRequest();
```

241.6. 例子

在以下路由中，我们使用 Netty HTTP 作为 HTTP 服务器，它返回一个硬编码的"Bye World"消息。

```
    from("netty-http:http://0.0.0.0:8080/foo")
      .transform().constant("Bye World");
```

我们也可以使用 Camel 调用此 HTTP 服务器，以及 ProducerTemplate，如下所示：

```
    String out = template.requestBody("netty-http:http://0.0.0.0:8080/foo", "Hello World",
String.class);
    System.out.println(out);
```

我们以输出形式返回 "Bye World"。

241.7. 如何让 NETTY 匹配通配符

默认情况下，Netty HTTP 仅在确切的 uri 上匹配。但是，您可以指示 Netty 匹配前缀。例如：

```
 from("netty-http:http://0.0.0.0:8123/foo").to("mock:foo");
```

在上面的 Netty HTTP 中，只有在 uri 完全匹配时才会匹配。因此，如果您输入 http://0.0.0.0:8123/foo，但如果您输入 http://0.0.0.0:8123/foo/bar，则它将匹配。

因此，如果要启用通配符匹配，如下所示：

```
 from("netty-http:http://0.0.0.0:8123/foo?matchOnUriPrefix=true").to("mock:foo");
```

因此，Netty 与以 foo 开头的任何端点匹配。

要匹配 任何 端点，您可以执行以下操作：

```
 from("netty-http:http://0.0.0.0:8123?matchOnUriPrefix=true").to("mock:foo");
```

241.8. 在同一端口使用多个路由

在同一 CamelContext 中，您可以使用来自 Netty HTTP 的多个路由来共享同一端口（如 org.jboss.netty.bootstrap.ServerBootstrap 实例）。执行此操作需要在路由中有多个 bootstrap 选项，因为路由将共享相同的 org.jboss.netty.bootstrap.ServerBootstrap 实例。实例将使用创建的第一个路由中的选项进行配置。

路由必须相同配置的选项是 org.apache.camel.component.netty.NettyServerBootstrapConfiguration 配置类中定义的所有选项。如果您使用不同的选项配置了另一个路由，Camel 会在启动时抛出异常，表示选项不相同。要缓解这个问题，请确保所有选项都相同。

下面是有两个路由共享同一端口的示例：

共享同一端口的两个路由

```
from("netty-http:http://0.0.0.0:{{port}}/foo")
  .to("mock:foo")
  .transform().constant("Bye World");

from("netty-http:http://0.0.0.0:{{port}}/bar")
  .to("mock:bar")
  .transform().constant("Bye Camel");
```

下面是一个错误配置的 2nd 路由的示例，它没有相同的 org.apache.camel.component.netty.NettyServerBootstrapConfiguration 选项作为第 1st 路由。这将导致 Camel 在启动时失败。

两个路由共享同一端口，但第二路由配置错误，并将在启动时失败

```
from("netty-http:http://0.0.0.0:{{port}}/foo")
  .to("mock:foo")
  .transform().constant("Bye World");

// we cannot have a 2nd route on same port with SSL enabled, when the 1st route is NOT
from("netty-http:http://0.0.0.0:{{port}}/bar?ssl=true")
  .to("mock:bar")
  .transform().constant("Bye Camel");
```

241.8.1. 使用多个路由重复使用相同的服务器 bootstrap 配置

通过在 org.apache.camel.component.netty.NettyServerBootstrapConfiguration 类型的单个实例中配置通用服务器 bootstrap 选项，我们可以对 Netty HTTP 用户使用 bootstrapConfiguration 选项来引用和重复使用所有用户相同的选项。

```
<bean id="nettyHttpBootstrapOptions"
class="org.apache.camel.component.netty.NettyServerBootstrapConfiguration">
  <property name="backlog" value="200"/>
```

```
  <property name="connectTimeout" value="20000"/>
  <property name="workerCount" value="16"/>
</bean>
```

在您引用此选项的路由中，如下所示

```
<route>
  <from uri="netty-http:http://0.0.0.0:{{port}}/foo?
bootstrapConfiguration=#nettyHttpBootstrapOptions"/>
  ...
</route>

<route>
  <from uri="netty-http:http://0.0.0.0:{{port}}/bar?
bootstrapConfiguration=#nettyHttpBootstrapOptions"/>
  ...
</route>

<route>
  <from uri="netty-http:http://0.0.0.0:{{port}}/beer?
bootstrapConfiguration=#nettyHttpBootstrapOptions"/>
  ...
</route>
```

241.8.2. 使用 OSGi 容器中多个捆绑包的多个路由重复使用相同的服务器引导配置

详情请查看 Netty HTTP 服务器 示例。

241.9. 使用 HTTP 基本身份验证

Netty HTTP 使用者通过指定要使用的安全域名称来支持 HTTP 基本身份验证，如下所示

```
<route>
  <from uri="netty-http:http://0.0.0.0:{{port}}/foo?securityConfiguration.realm=karaf"/>
  ...
</route>
```

域名称是必需的，才能启用基本身份验证。默认情况下，使用了基于 JAAS 的验证器，它使用指定的域名称（上例中的karaf），并使用此域的 JAAS 域和 JAAS \{{LoginModule}}s 进行身份验证。

Apache Karaf / ServiceMix 的最终用户开箱即用有一个 karaf 域，因此上面的示例将在这些容器中开箱即用。

241.9.1. 在 Web 资源中指定 ACL

org.apache.camel.component.netty.http.SecurityConstraint 允许定义对 Web 资源的约束。并且提供了 org.apache.camel.component.netty.http.SecurityConstraintMapping，允许使用角色轻松定义包含和排除项。

例如，在 XML DSL 中，我们定义约束 bean：

```
<bean id="constraint" class="org.apache.camel.component.netty.http.SecurityConstraintMapping">
  <!-- inclusions defines url -> roles restrictions -->
  <!-- a * should be used for any role accepted (or even no roles) -->
  <property name="inclusions">
    <map>
      <entry key="/*" value="*"/>
      <entry key="/admin/*" value="admin"/>
      <entry key="/guest/*" value="admin,guest"/>
    </map>
  </property>
  <!-- exclusions is used to define public urls, which requires no authentication -->
  <property name="exclusions">
    <set>
      <value>/public/*</value>
    </set>
  </property>
</bean>
```

上面的约束已定义，以便

- 对 Thycotic 的访问会被限制，并且接受任何角色（也如果没有角色，也接受任何角色）
- 访问 /admin86] 需要 admin 角色
- 访问 /guest205 需要 admin 或 guest 角色
- 对 /public netobserv 的访问是一个排除项，这意味着不需要身份验证，因此对于没有登录的任何人都公开了身份验证

要使用此约束，我们只需要引用 bean id，如下所示：

```
<route>
```

```
    <from uri="netty-http:http://0.0.0.0:{{port}}/foo?
matchOnUriPrefix=true&amp;securityConfiguration.realm=karaf&amp;securityConfiguration.securityCon
straint=#constraint"/>
    ...
</route>
```

241.10. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- Netty
- Netty HTTP 服务器示例
- Jetty

第 242 章 NETTY4 组件

从 Camel 版本 2.14 开始提供

Camel 中的 netty4 组件是一个套接字通信组件，基于 Netty 项目版本 4。
Netty 是一个 NIO 客户端服务器框架，能够快速轻松地开发 networkServerInitializerFactory 应用程序，如协议服务器和客户端。
Netty 大大简化了网络编程，如 TCP 和 UDP 套接字服务器。

此 camel 组件支持生成者和消费者端点。

Netty 组件有多个选项，允许对多个 TCP/UDP 通信参数（缓冲大小、keepAlives、tcpNoDelay 等）进行精细控制，并促进 Camel 路由上的 In-Only 和 In-Out 通信。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-netty4</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

242.1. URI 格式

netty 组件的 URI 方案如下

```
netty4:tcp://0.0.0.0:99999[?options]
netty4:udp://remotehost:99999/[?options]
```

此组件支持 TCP 和 UDP 的制作者和消费者端点。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

242.2. 选项

Netty4 组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
maximumPoolSize (advanced)	如果正在使用 EventExecutorGroup 的线程池大小。默认值为 16。	16	int
配置 （高级）	在创建端点时，使用 NettyConfiguration 作为配置。		NettyConfiguration
executorService (advanced)	使用给定的 EventExecutorGroup。		EventExecutorGroup
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Netty4 端点使用 URI 语法进行配置：

```
netty4:protocol:host:port
```

使用以下路径和查询参数：

242.2.1. 路径参数(3 参数)：

Name	描述	默认值	类型
protocol	**必需** 使用的协议，可以是 tcp 或 udp。		字符串
主机	**必需** 的主机名。对于消费者，主机名是 localhost 或 0.0.0.0。对于生成者，主机名是要连接的远程主机		字符串
port	**需要** 主机端口号		int

242.2.2. 查询参数(72 参数)：

Name	描述	默认值	类型
disconnect (common)	使用后是否从 Netty Channel 断开连接（关闭）。可用于使用者和制作者。	false	布尔值
keepalive (common)	设置以确保因为不活跃而不会关闭套接字	true	布尔值
reuseAddress (common)	设置以方便套接字多路	true	布尔值
reuseChannel (common)	此选项允许生成者和消费者（在客户端模式中）在处理交换生命周期中重复使用相同的 Netty Channel。如果您需要在 Camel 路由中多次调用服务器并希望使用相同的网络连接，这非常有用。使用此选项时，频道不会返回到连接池，直到 Exchange 完成后；如果 disconnect 选项设为 true，则断开连接。重复使用的频道以一个交换属性的形式存储在 Exchange 中，它带有一个键 NettyConstants SerialNETTY_CHANNEL，它允许您在路由过程中获取频道并使用它。	false	布尔值
sync (common)	将端点设置为单向或请求响应	true	布尔值
tcpNoDelay (common)	设置以提高 TCP 协议性能	true	布尔值
bridgeErrorHandl er (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
broadcast (consumer)	将 设置为选择通过 UDP 的多播	false	布尔值
clientMode (consumer)	如果 clientMode 为 true，netty 使用者会将地址连接到 TCP 客户端。	false	布尔值
reconnect (consumer)	仅在消费者中的 clientMode 中使用，如果用户启用，消费者将尝试在断开连接时重新连接	true	布尔值
reconnectInterval (consumer)	如果启用了 reconnect 和 clientMode，则使用。尝试重新连接的时间间隔（以 milli 秒为单位）	10000	int
backlog (consumer)	允许为 netty consumer (server)配置积压。请注意，积压只是取决于操作系统的最佳努力。将此选项设置为值（如 200、500 或 1000）告知 TCP 堆栈如果未配置此选项，则接受队列的时长，则积压取决于操作系统设置。		int

Name	描述	默认值	类型
bossCount (consumer)	当 netty 适用于 nio 模式时，它使用 Netty 中的默认 bossCount 参数，即 1。用户可以使用此操作从 Netty 覆盖默认的 bossCount	1	int
bossGroup (consumer)	设置 BossGroup，可用于处理 NettyEndpoint 中服务器端的新连接		EventLoopGroup
disconnectOnNoReply (consumer)	如果启用了同步，则此选项将指定 NettyConsumer（如果它应该断开连接），没有回复回来。	true	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
nettyServerBootstrapFactory (consumer)	使用自定义 NettyServerBootstrapFactory		NettyServerBootstrap Factory
networkInterface (consumer)	在使用 UDP 时，可以使用此选项按名称指定网络接口，如 eth0 以加入多播组。		字符串
noReplyLogLevel (consumer)	如果启用了同步，这个选项会指示 NettyConsumer，在记录一个没有回复时要使用的日志记录级别。	WARN	LoggingLevel
serverClosedChannelExceptionCaughtLogLevel (consumer)	如果服务器(NettyConsumer)捕获 java.nio.channels.ClosedChannelException，然后使用此日志级别进行记录。这用于避免记录关闭的频道异常，因为客户端可以立即断开，然后在 Netty 服务器中造成大量关闭的异常。	DEBUG	LoggingLevel
serverExceptionCaughtLog Level (consumer)	如果服务器(NettyConsumer)捕获异常，则使用此日志级别进行日志记录。	WARN	LoggingLevel
serverInitializerFactory (consumer)	使用自定义 ServerInitializerFactory		ServerInitializer Factory
useExecutorService (consumer)	是否使用排序的线程池，确保同一通道上按顺序处理事件。	true	布尔值
connectTimeout (producer)	等待套接字连接可用的时间。值以毫秒为单位。	10000	int

Name	描述	默认值	类型
requestTimeout (producer)	在调用远程服务器时，允许 Netty producer 使用超时。默认情况下，没有使用超时。该值在 milli 秒内，如 30000 为 30 秒。requestTimeout 使用 Netty 的 ReadTimeoutHandler 来触发超时。		long
clientInitializerFactory (producer)	使用自定义 ClientInitializerFactory		ClientInitializerFactory
correlationManager (producer)	使用自定义关联管理器来管理在将请求/回复与 netty producer 搭配使用时如何映射请求和回复消息。只有在您有一个将请求与回复一起映射请求时（如请求和回复中有关联 ID）时，才应使用此设置。如果要在 netty 中的同一频道（也称为连接）上有多个并发消息，则可以使用它。当执行此操作时，您必须有一个与请求和回复相关的方法，以便在继续路由前将正确的回复存储在 inflight Camel Exchange 上。当您构建自定义关联管理器时，我们建议扩展 TimeoutCorrelationManagerSupport。这提供了对超时和其他复杂性的支持。如需了解更多详细信息，请参阅 producerPoolEnabled 选项。		NettyCamelStateCorrelationManager
lazyChannelCreation (producer)	如果远程服务器在 Camel producer 启动时未启动并运行，则可以创建频道以避免异常。	true	布尔值
producerPoolEnabled (producer)	producer 池是否已启用。重要：如果您关闭此设置，则为生成者使用单个共享连接，如果您执行请求/回复。这意味着，如果回复超出顺序，则存在交错响应潜在的问题。因此，您需要在请求和回复消息中有一个关联 ID，以便您可以将回复与负责继续处理 Camel 中消息的 Camel 回调关联。为此，您需要将 NettyCamelStateCorrelationManager 实施为关联管理器，并通过 correlationManager 选项进行配置。如需了解更多详细信息，请参阅 correlationManager 选项。	true	布尔值
producerPoolMaxActive (producer)	设置池可分配给客户端或闲置等待签出的对象数量上限。对没有限制，使用负值。	-1	int
producerPoolMaxIdle (producer)	设置池中空闲实例数量上限。	100	int
producerPoolMinEvictable Idle (producer)	设置对象在空闲对象驱除有资格驱除前，对象可能会在池中闲置的最小时间（值为 millis））。	300000	long
producerPoolMinIdle (producer)	设置制作者池中允许的最小实例数量，然后再驱除线程（如果活跃）生成新对象。		int

Name	描述	默认值	类型
udpConnectionlessSending (producer)	这个选项支持连接较少的 udp 发送，这是实际触发和忘记的。如果没有端口侦听接收端口，则已连接的 udp 发送 PortUnreachableException。	false	布尔值
useByteBuf (producer)	如果 useByteBuf 为 true，netty producer 会在发送前将消息正文转换为 ByteBuf。	false	布尔值
allowSerializedHeaders (advanced)	仅在 transferExchange 为 true 时用于 TCP。当设置为 true 时，标头和属性中的串行对象将添加到交换中。否则，Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。	false	布尔值
bootstrapConfiguration (advanced)	使用自定义配置的 NettyServerBootstrapConfiguration 来配置此端点。		NettyServerBootstrap Configuration
channelGroup (advanced)	使用显式 ChannelGroup。		ChannelGroup
nativeTransport (advanced)	是否使用原生传输而不是 NIO。原生传输利用主机操作系统，且仅在某些平台上受支持。您需要为正在使用的主机操作系统添加 netty JAR。详情请查看 ：http://netty.io/wiki/native-transports.html	false	布尔值
选项 (advanced)	允许使用选项配置其他 netty 选项作为前缀。例如：option.child.keepAlive=false 设置 netty 选项 child.keepAlive=false。有关可以使用的选项，请参阅 Netty 文档。		Map
receiveBufferSize (advanced)	在入站通信中使用的 TCP/UDP 缓冲区大小。大小为字节。	65536	int
receiveBufferSizePredictor (advanced)	配置缓冲区大小预测。请参阅 Jetty 文档和此邮件线程的详细信息。		int
sendBufferSize (advanced)	在出站通信中使用的 TCP/UDP 缓冲大小。大小为字节。	65536	int
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
transferExchange (advanced)	仅用于 TCP。您可以通过线线传输交换，而不是只传输正文。以下字段被传输： 在 body, Out body, fault body, In headers, Out headers, fault headers, Exchange properties, Exchange exception。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。	false	布尔值

Name	描述	默认值	类型
udpByteArrayCodec (advanced)	仅限 UDP。如果启用了，则使用 byte 数组 codec 而不是 Java 序列化协议。	false	布尔值
workerCount (advanced)	当 netty 适用于 nio 模式时，它会使用来自 Netty 的默认 workerCount 参数，即 cpu_core_threads x 2。用户可以使用此操作从 Netty 覆盖默认的 workerCount。		int
workerGroup (advanced)	使用显式 EventLoopGroup 作为 boss 线程池。例如，要与多个消费者或生成者共享线程池。默认情况下，每个消费者或生成者都有自己的 worker 池，有 2 个 x cpu count 内核线程。		EventLoopGroup
allowDefaultCodec (codec)	如果两者都是 null，netty 组件会安装默认的 codec，则 encoder/decoder 为 null，文本为 false。将 allowDefaultCodec 设置为 false 可防止 netty 组件将默认 codec 安装为过滤器链中的第一个元素。	true	布尔值
autoAppendDelimiter (codec)	使用文本 codec 发送时是否自动附加缺少的结束分隔符。	true	布尔值
decoder (codec)	**弃用了** 一个自定义 ChannelHandler 类，可用于执行入站有效负载的特殊汇总。		ChannelHandler
decoderMaxLineLength (codec)	用于文本行 codec 的最大行长度。	1024	int
decoders (codec)	要使用的解码器列表。您可以使用以逗号分开的值的 String，并在 Registry 中查找值。只需记住使用 # 前缀，以便 Camel 知道它应该查找。		字符串
delimiter (codec)	用于文本行代码的分隔符。可能的值有 LINE 和 NULL。	行	TextLineDelimiter
encoder (codec)	**弃用了** 一个自定义 ChannelHandler 类，可用于执行出站有效负载的特殊汇总。		ChannelHandler
编码器 (codec)	要使用的编码器列表。您可以使用以逗号分开的值的 String，并在 Registry 中查找值。只需记住使用 # 前缀，以便 Camel 知道它应该查找。		字符串
编码 (codec)	用于文本编码的编码(charset 名称)。如果没有提供，Camel 将使用 JVM 默认 Charset。		字符串
文本行 (codec)	仅用于 TCP。如果没有指定 codec，您可以使用此标志来指示基于文本的 codec；如果没有指定，或者值为 false，则通过 TCP 假设 Object Serialization。	false	布尔值

Name	描述	默认值	类型
enabledProtocols (security)	使用 SSL 时要启用的协议	TLSv1, TLSv1.1, TLSv1.2	字符串
keyStoreFile (security)	用于加密的客户端侧证书密钥存储		File
keyStoreFormat (security)	用于有效负载加密的密钥存储格式。如果没有设置，则默认为 JKS		字符串
keyStoreResource (security)	用于加密的客户端侧证书密钥存储。默认情况下从 classpath 加载，但您可以使用 classpath:、file: 或 http: 前缀来加载来自不同系统的资源。		字符串
needClientAuth (security)	配置服务器在使用 SSL 时是否需要客户端身份验证。	false	布尔值
密码短语 (security)	要使用 SSH 加密/解密有效负载的密码设置		字符串
securityProvider (security)	用于有效负载加密的安全供应商。如果没有设置，则默认为 SunX509。		字符串
SSL (security)	设置以指定是否将 SSL 加密应用到此端点	false	布尔值
sslClientCertHeaders (security)	启用和 SSL 模式时，Netty 使用者将增强 Camel 消息，其中包含客户端证书的信息，如主题名称、签发者名称、序列号和有效日期范围。	false	布尔值
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters
sslHandler (security)	引用可用于返回 SSL 处理程序的类		SslHandler
trustStoreFile (security)	用于加密的服务器端证书密钥存储		File
trustStoreResource (security)	用于加密的服务器端证书密钥存储。默认情况下从 classpath 加载，但您可以使用 classpath:、file: 或 http: 前缀来加载来自不同系统的资源。		字符串

242.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 78 选项，如下所列。

Name	描述	默认值	类型
camel.component.netty4.configuration.allow-default-codec	如果两者都是 null，netty 组件会安装默认的 codec，则 encoder/decoder 为 null，文本为 false。将 allowDefaultCodec 设置为 false 可防止 netty 组件将默认 codec 安装为过滤器链中的第一个元素。	true	布尔值
camel.component.netty4.configuration.allow-serialized-headers	仅在 transferExchange 为 true 时用于 TCP。当设置为 true 时，标头和属性中的串行对象将添加到交换中。否则，Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。	false	布尔值
camel.component.netty4.configuration.auto-append-delimiter	使用文本 codec 发送时是否自动附加缺少的结束分隔符。	true	布尔值
camel.component.netty4.configuration.backlog	允许为 netty consumer (server)配置积压。请注意，积压只是取决于操作系统的最佳努力。将此选项设置为值（如 200、500 或 1000）告知 TCP 堆栈如果未配置此选项，则接受队列的时长，则积压取决于操作系统设置。		整数
camel.component.netty4.configuration.boss-count	当 netty 适用于 nio 模式时，它使用 Netty 中的默认 bossCount 参数，即 1。用户可以使用此操作从 Netty 覆盖默认的 bossCount	1	整数
camel.component.netty4.configuration.boss-group	设置 BossGroup，可用于处理 NettyEndpoint 中服务器端的新连接		EventLoopGroup
camel.component.netty4.configuration.broadcast	将 设置为选择通过 UDP 的多播	false	布尔值
camel.component.netty4.configuration.channel-group	使用显式 ChannelGroup。		ChannelGroup
camel.component.netty4.configuration.client-initializer-factory	使用自定义 ClientInitializerFactory		ClientInitializer Factory
camel.component.netty4.configuration.client-mode	如果 clientMode 为 true，netty 使用者会将地址连接到 TCP 客户端。	false	布尔值

Name	描述	默认值	类型
camel.component.netty4.configuration.connect-timeout	等待套接字连接可用的时间。值以毫秒为单位。	10000	整数
camel.component.netty4.configuration.correlation-manager	使用自定义关联管理器来管理在将请求/回复与 netty producer 搭配使用时如何映射请求和回复消息。只有在您有一个将请求与回复一起映射请求时（如请求和回复中有关联 ID）时，才应使用此设置。如果要在 netty 中的同一频道（也称为连接）上有多个并发消息，则可以使用它。当执行此操作时，您必须有一个与请求和回复相关的方法，以便在继续路由前将正确的回复存储在 inflight Camel Exchange 上。当您构建自定义关联管理器时，我们建议扩展 TimeoutCorrelationManagerSupport。这提供了对超时和其他复杂性的支持。如需了解更多详细信息，请参阅 producerPoolEnabled 选项。		NettyCamelStateCorrelationManager
camel.component.netty4.configuration.decoder-max-line-length	用于文本行 codec 的最大行长度。	1024	整数
camel.component.netty4.configuration.decoders	要使用的解码器列表。您可以使用以逗号分开的值的 String，并在 Registry 中查找值。只需记住使用 # 前缀，以便 Camel 知道它应该查找。		list
camel.component.netty4.configuration.delimiter	用于文本行代码的分隔符。可能的值有 LINE 和 NULL。		TextLineDelimiter
camel.component.netty4.configuration.disconnect	使用后是否从 Netty Channel 断开连接（关闭）。可用于使用者和制作者。	false	布尔值
camel.component.netty4.configuration.disconnect-on-no-reply	如果启用了同步，则此选项将指定 NettyConsumer（如果它应该断开连接），没有回复回来。	true	布尔值
camel.component.netty4.configuration.enabled-protocols	使用 SSL 时要启用的协议	TLSv1, TLSv1.1, TLSv1.2	字符串

Name	描述	默认值	类型
camel.component.netty4.configuration.encoders	要使用的编码器列表。您可以使用以逗号分开的值的 String，并在 Registry 中查找值。只需记住使用 # 前缀，以便 Camel 知道它应该查找。		list
camel.component.netty4.configuration.encoding	用于文本编码的编码(charset 名称)。如果没有提供，Camel 将使用 JVM 默认 Charset。		字符串
camel.component.netty4.configuration.host	主机名。对于消费者，主机名是 localhost 或 0.0.0.0。对于生成者，主机名是要连接的远程主机		字符串
camel.component.netty4.configuration.keep-alive	设置以确保因为不活跃而不会关闭套接字	true	布尔值
camel.component.netty4.configuration.key-store-format	用于有效负载加密的密钥存储格式。如果没有设置，则默认为 JKS		字符串
camel.component.netty4.configuration.key-store-resource	用于加密的客户端侧证书密钥存储。默认情况下从 classpath 加载，但您可以使用 classpath:、file: 或 http: 前缀来加载来自不同系统的资源。		字符串
camel.component.netty4.configuration.lazy-channel-creation	如果远程服务器在 Camel producer 启动时未启动并运行，则可以创建频道以避免异常。	true	布尔值
camel.component.netty4.configuration.native-transport	是否使用原生传输而不是 NIO。原生传输利用主机操作系统，且仅在某些平台上受支持。您需要为正在使用的主机操作系统添加 netty JAR。详情请查看 ：http://netty.io/wiki/native-transports.html	false	布尔值
camel.component.netty4.configuration.need-client-auth	配置服务器在使用 SSL 时是否需要客户端身份验证。	false	布尔值
camel.component.netty4.configuration.netty-server-bootstrap-factory	使用自定义 NettyServerBootstrapFactory		NettyServerBootstrap Factory

Name	描述	默认值	类型
camel.component.netty4.configuration.network-interface	在使用 UDP 时，可以使用此选项按名称指定网络接口，如 eth0 以加入多播组。		字符串
camel.component.netty4.configuration.no-reply-log-level	如果启用了同步，这个选项会指示 NettyConsumer，在记录一个没有回复时要使用的日志记录级别。		LoggingLevel
camel.component.netty4.configuration.options	允许使用选项配置其他 netty 选项作为前缀。例如：option.child.keepAlive=false 设置 netty 选项 child.keepAlive=false。有关可以使用的选项，请参阅 Netty 文档。		Map
camel.component.netty4.configuration.passphrase	要使用 SSH 加密/解密有效负载的密码设置		字符串
camel.component.netty4.configuration.port	主机端口号		整数
camel.component.netty4.configuration.producer-pool-enabled	producer 池是否已启用。重要：如果您关闭此设置，则为生成者使用单个共享连接，如果您执行请求/回复。这意味着，如果回复超出顺序，则存在交错响应潜在的问题。因此，您需要在请求和回复消息中有一个关联 ID，以便您可以将回复与负责继续处理 Camel 中消息的 Camel 回调关联。为此，您需要将 NettyCamelStateCorrelationManager 实施为关联管理器，并通过 correlationManager 选项进行配置。如需了解更多详细信息，请参阅 correlationManager 选项。	true	布尔值
camel.component.netty4.configuration.producer-pool-max-active	设置池可分配给客户端或闲置等待签出的对象数量上限。对没有限制，使用负值。	-1	整数
camel.component.netty4.configuration.producer-pool-max-idle	设置池中空闲实例数量上限。	100	整数

Name	描述	默认值	类型
camel.component.netty4.configuration.producer-pool-min-evictable-idle	设置对象在空闲对象驱除有资格驱除前，对象可能会在池中闲置的最小时间（值为 millis））。	300000	Long
camel.component.netty4.configuration.producer-pool-min-idle	设置制作者池中允许的最小实例数量，然后再驱除线程（如果活跃）生成新对象。		整数
camel.component.netty4.configuration.protocol	要使用的协议，可以是 tcp 或 udp。		字符串
camel.component.netty4.configuration.receive-buffer-size	在入站通信中使用的 TCP/UDP 缓冲区大小。大小为字节。	65536	整数
camel.component.netty4.configuration.receive-buffer-size-predictor	配置缓冲区大小预测。请参阅 Jetty 文档和此邮件线程的详细信息。		整数
camel.component.netty4.configuration.reconnect	仅在消费者中的 clientMode 中使用，如果用户启用，消费者将尝试在断开连接时重新连接	true	布尔值
camel.component.netty4.configuration.reconnect-interval	如果启用了 reconnect 和 clientMode，则使用。尝试重新连接的时间间隔（以 milli 秒为单位）	10000	整数
camel.component.netty4.configuration.request-timeout	在调用远程服务器时，允许 Netty producer 使用超时。默认情况下，没有使用超时。该值在 milli 秒内，如 30000 为 30 秒。requestTimeout 使用 Netty 的 ReadTimeoutHandler 来触发超时。		Long
camel.component.netty4.configuration.reuse-address	设置以方便套接字多路	true	布尔值

Name	描述	默认值	类型
camel.component.netty4.configuration.reuse-channel	此选项允许生成者和消费者（在客户端模式中）在处理交换生命周期中重复使用相同的 Netty Channel。如果您需要在 Camel 路由中多次调用服务器并希望使用相同的网络连接，这非常有用。使用此选项时，频道不会返回到连接池，直到 Exchange 完成后；如果 disconnect 选项设为 true，则断开连接。重复使用的频道以一个交换属性的形式存储在 Exchange 中，它带有一个键 NettyConstants SerialNETTY_CHANNEL，它允许您在路由过程中获取频道并使用它。	false	布尔值
camel.component.netty4.configuration.security-provider	用于有效负载加密的安全供应商。如果没有设置，则默认为 SunX509。		字符串
camel.component.netty4.configuration.send-buffer-size	在出站通信中使用的 TCP/UDP 缓冲大小。大小为字节。	65536	整数
camel.component.netty4.configuration.server-closed-channel-exception-caught-log-level	如果服务器(NettyConsumer)捕获 java.nio.channels.ClosedChannelException，然后使用此日志级别进行记录。这用于避免记录关闭的频道异常，因为客户端可以立即断开，然后在 Netty 服务器中造成大量关闭的异常。		LoggingLevel
camel.component.netty4.configuration.server-exception-caught-log-level	如果服务器(NettyConsumer)捕获异常，则使用此日志级别进行日志记录。		LoggingLevel
camel.component.netty4.configuration.server-initializer-factory	使用自定义 ServerInitializerFactory		ServerInitializer Factory
camel.component.netty4.configuration.ssl	设置以指定是否将 SSL 加密应用到此端点	false	布尔值
camel.component.netty4.configuration.ssl-client-cert-headers	启用和 SSL 模式时，Netty 使用者将增强 Camel 消息，其中包含客户端证书的信息，如主题名称、签发者名称、序列号和有效日期范围。	false	布尔值

Name	描述	默认值	类型
camel.component.netty4.configuration.ssl-context-parameters	使用 SSLContextParameters 配置安全性		SSLContextParameters
camel.component.netty4.configuration.ssl-handler	引用可用于返回 SSL 处理程序的类		SslHandler
camel.component.netty4.configuration.sync	将端点设置为单向或请求响应	true	布尔值
camel.component.netty4.configuration.tcp-no-delay	设置以提高 TCP 协议性能	true	布尔值
camel.component.netty4.configuration.textline	仅用于 TCP。如果没有指定 codec，您可以使用此标志来指示基于文本的 codec；如果没有指定，或者值为 false，则通过 TCP 假设 Object Serialization。	false	布尔值
camel.component.netty4.configuration.transfer-exchange	仅用于 TCP。您可以通过线线传输交换，而不是只传输正文。以下字段被传输： 在 body, Out body, fault body, In headers, Out headers, fault headers, Exchange properties, Exchange exception。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。	false	布尔值
camel.component.netty4.configuration.trust-store-resource	用于加密的服务器端证书密钥存储。默认情况下从 classpath 加载，但您可以使用 classpath:、file: 或 http: 前缀来加载来自不同系统的资源。		字符串
camel.component.netty4.configuration.udp-byte-array-codec	仅限 UDP。如果启用了，则使用 byte 数组 codec 而不是 Java 序列化协议。	false	布尔值
camel.component.netty4.configuration.udp-connectionless-sending	这个选项支持连接较少的 udp 发送，这是实际触发和忘记的。如果没有端口侦听接收端口，则已连接的 udp 发送 PortUnreachableException。	false	布尔值
camel.component.netty4.configuration.use-byte-buf	如果 useByteBuf 为 true，netty producer 会在发送前将消息正文转换为 ByteBuf。	false	布尔值

Name	描述	默认值	类型
camel.component.netty4.configuration.using-executor-service	是否使用排序的线程池，确保同一通道上按顺序处理事件。	true	布尔值
camel.component.netty4.configuration.worker-count	当 netty 适用于 nio 模式时，它会使用来自 Netty 的默认 workerCount 参数，即 cpu_core_threads x 2。用户可以使用此操作从 Netty 覆盖默认的 workerCount。		整数
camel.component.netty4.configuration.worker-group	使用显式 EventLoopGroup 作为 boss 线程池。例如，要与多个消费者或生成者共享线程池。默认情况下，每个消费者或生成者都有自己的 worker 池，有 2 个 x cpu count 内核线程。		EventLoopGroup
camel.component.netty4.enabled	启用 netty4 组件	true	布尔值
camel.component.netty4.executor-service	使用给定的 EventExecutorGroup。选项是一个 io.netty.util.concurrent.EventExecutorGroup 类型。		字符串
camel.component.netty4.maximum-pool-size	如果正在使用 EventExecutorGroup 的线程池大小。默认值为 16。	16	整数
camel.component.netty4.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.netty4.ssl-context-parameters	使用 SSLContextParameters 配置安全性。选项是 org.apache.camel.util.jsse.SSLContextParameters 类型。		字符串
camel.component.netty4.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值
camel.component.netty4.configuration.client-pipeline-factory	@deprecated 使用 #setClientInitializerFactory		ClientInitializerFactory

Name	描述	默认值	类型
camel.component.netty4.configuration.decoder	此自定义 ChannelHandler 类可用于执行入站有效负载的特殊汇总。		ChannelHandler
camel.component.netty4.configuration.encoder	此自定义 ChannelHandler 类可用于执行出站有效负载的特殊汇总。		ChannelHandler
camel.component.netty4.configuration.key-store-file	用于加密的客户端侧证书密钥存储		File
camel.component.netty4.configuration.server-pipeline-factory	@deprecated 使用 #setServerInitializerFactory		ServerInitializerFactory
camel.component.netty4.configuration.trust-store-file	用于加密的服务器端证书密钥存储		File

242.4. 基于 REGISTRY 的选项

codec Handlers 和 SSL Keystores 可以在 Registry 中列出，如 Spring XML 文件中。可以传递的值如下：

Name	描述
passphrase	要使用 SSH 加密/解密有效负载的密码设置
keyStoreFormat	用于有效负载加密的密钥存储格式。如果没有设置，则默认为 "JKS"
securityProvider	用于有效负载加密的安全供应商。如果没有设置，则默认为 "SunX509"。
keyStoreFile	**弃用：** 用于加密的客户端侧证书密钥存储

Name	描述
trustStoreFile	**弃用：** 用于加密的服务器侧证书密钥存储
keyStoreResource	Camel 2.11.1： 用于加密的客户端侧证书密钥存储。默认情况下从 classpath 加载，但您可以使用 **"classpath:"**, **"file:"** 或 **"http:"** 前缀来加载来自不同系统的资源。
trustStoreResource	Camel 2.11.1： 用于加密的服务器侧证书密钥存储。默认情况下从 classpath 加载，但您可以使用 **"classpath:"**, **"file:"** 或 **"http:"** 前缀来加载来自不同系统的资源。
sslHandler	引用可用于返回 SSL 处理程序的类
encoder	此自定义 **ChannelHandler** 类可用于执行出站有效负载的特殊汇总。必须覆盖 io.netty.channel.ChannelInboundHandlerAdapter。
encoders	要使用的编码器列表。您可以使用以逗号分开的值的 String，并在 Registry 中查找值。只需记住使用 # 前缀，以便 Camel 知道它应该查找。
decoder	此自定义 **ChannelHandler** 类可用于执行入站有效负载的特殊汇总。必须覆盖 io.netty.channel.ChannelOutboundHandlerAdapter。
decoders	要使用的解码器列表。您可以使用以逗号分开的值的 String，并在 Registry 中查找值。只需记住为值添加前缀 #，因此 Camel 知道它应该进行查找。

注意

请阅读以下有关使用不可共享编码器/解码器的信息。

242.4.1. 使用不可共享编码器或解码器

如果您的编码器或解码器不可共享（例如，它们没有 @Shareable 类注释），则您的编码器/解码器必须实施 org.apache.camel.component.netty.ChannelHandlerFactory 接口，并在 newChannelHandler 方法返回新实例。这是为了确保可以安全地使用 encoder/decoder。否则，Netty 组件会在创建端点时记录 WARN。

Netty 组件提供了一个 org.apache.camel.component.netty.ChannelHandlerFactories 工厂类，它有多个常用的方法。

242.5. 将消息发送到/从 NETTY 端点发送

242.5.1. Netty Producer

在 Producer 模式中，组件提供使用 TCP 或 UDP 协议（具有可选 SSL 支持）将有效负载发送到套接字端点的功能。

producer 模式支持基于单向和请求响应的操作。

242.5.2. Netty Consumer

在 Consumer 模式中，组件提供以下功能：

- 使用 TCP 或 UDP 协议（具有可选 SSL 支持）侦听指定的套接字。
- 使用基于 text/xml、二进制和序列化对象的有效负载在套接字上接收请求，
- 在路由上将它们作为消息交换发送。

消费者模式支持基于单向和请求响应的操作。

242.6. 例子

242.6.1. 使用 Request-Reply 和 serialized 对象有效负载的 UDP Netty 端点

```
RouteBuilder builder = new RouteBuilder() {
  public void configure() {
    from("netty4:udp://0.0.0.0:5155?sync=true")
      .process(new Processor() {
        public void process(Exchange exchange) throws Exception {
          Poetry poetry = (Poetry) exchange.getIn().getBody();
          poetry.setPoet("Dr. Sarojini Naidu");
          exchange.getOut().setBody(poetry);
        }
      }
    }
};
```

–

242.6.2. 使用单向通信基于 TCP 的 Netty consumer 端点

```
RouteBuilder builder = new RouteBuilder() {
  public void configure() {
      from("netty4:tcp://0.0.0.0:5150")
          .to("mock:result");
  }
};
```

242.6.3. 使用 Request-Reply 通信基于 SSL/TCP 的 Netty 消费者端点

使用 JSSE 配置实用程序

自 Camel 2.9 起，Netty 组件通过 Camel JSSE 配置实用程序支持 SSL/TLS 配置。这个实用程序可大大减少您需要写入的组件特定代码量，并在端点和组件级别进行配置。以下示例演示了如何将 实用程序与 Netty 组件一起使用。

组件的编程配置

```
KeyStoreParameters ksp = new KeyStoreParameters();
ksp.setResource("/users/home/server/keystore.jks");
ksp.setPassword("keystorePassword");

KeyManagersParameters kmp = new KeyManagersParameters();
kmp.setKeyStore(ksp);
kmp.setKeyPassword("keyPassword");

SSLContextParameters scp = new SSLContextParameters();
scp.setKeyManagers(kmp);

NettyComponent nettyComponent = getContext().getComponent("netty4",
NettyComponent.class);
nettyComponent.setSslContextParameters(scp);
```

基于 Spring DSL 的端点配置

```
...
  <camel:sslContextParameters
     id="sslContextParameters">
    <camel:keyManagers
       keyPassword="keyPassword">
      <camel:keyStore
         resource="/users/home/server/keystore.jks"
         password="keystorePassword"/>
    </camel:keyManagers>
```

```
  </camel:sslContextParameters>...
...
  <to uri="netty4:tcp://0.0.0.0:5150?
sync=true&ssl=true&sslContextParameters=#sslContextParameters"/>
...
```

[[Netty4-UsingBasicSSL/TLSconfigurationontheJettyComponent]] 在 Jetty 组件中使用基本 SSL/TLS 配置

```
JndiRegistry registry = new JndiRegistry(createJndiContext());
registry.bind("password", "changeit");
registry.bind("ksf", new File("src/test/resources/keystore.jks"));
registry.bind("tsf", new File("src/test/resources/keystore.jks"));

context.createRegistry(registry);
context.addRoutes(new RouteBuilder() {
  public void configure() {
     String netty_ssl_endpoint =
       "netty4:tcp://0.0.0.0:5150?sync=true&ssl=true&passphrase=#password"
       + "&keyStoreFile=#ksf&trustStoreFile=#tsf";
     String return_string =
       "When You Go Home, Tell Them Of Us And Say,"
       + "For Your Tomorrow, We Gave Our Today.";

     from(netty_ssl_endpoint)
      .process(new Processor() {
        public void process(Exchange exchange) throws Exception {
          exchange.getOut().setBody(return_string);
        }
      }
  }
});
```

访问 SSLSession 和客户端证书

如果需要获取客户端证书的详细信息，您可以访问 javax.net.ssl.SSLSession。当 ssl=true 随后，Netty 4 组件会将 SSLSession 存储为 Camel 消息上的标头，如下所示：

```
SSLSession session = exchange.getIn().getHeader(NettyConstants.NETTY_SSL_SESSION,
SSLSession.class);
// get the first certificate which is client certificate
javax.security.cert.X509Certificate cert = session.getPeerCertificateChain()[0];
Principal principal = cert.getSubjectDN();
```

请记住，要设置 needClientAuth=true 以验证客户端，否则 SSLSession 无法访问客户端证书的信息，您可能会收到一个例外 javax.net.ssl.SSLPeerUnverifiedException: peer not authenticated。如果客户端证书过期或无效，您可能还会获得这个异常。

提示

选项 sslClientCertHeaders 可以设置为 true，然后使用带有客户端证书详情的标头增强 Camel 消息。例如，标题 CamelNettySSLClientCertSubjectName 中可随时提供主题名称。

242.6.4. 使用多个代码cs

在某些情况下，可能需要将编码器和解码器链添加到 netty 管道。要将多个 codecs 添加到 camel netty 端点中，应使用 'encoders' 和 'decoders' uri 参数。与 'encoder' 和 'decoder' 参数类似，它们用来提供应该添加到管道中的引用(ChannelUpstreamHandlers 和 ChannelDownstreamHandlers 列表)。请注意，如果指定了编码器，则忽略 encoder 参数，类似于解码器和解码器参数。

注意

请参阅上面有关使用不可共享编码器/解码器的进一步的信息。

codecs 列表需要添加到 Camel 的 registry 中，以便在创建端点时解析它们。

```
ChannelHandlerFactory lengthDecoder =
ChannelHandlerFactories.newLengthFieldBasedFrameDecoder(1048576, 0, 4, 0, 4);

StringDecoder stringDecoder = new StringDecoder();
registry.bind("length-decoder", lengthDecoder);
registry.bind("string-decoder", stringDecoder);

LengthFieldPrepender lengthEncoder = new LengthFieldPrepender(4);
StringEncoder stringEncoder = new StringEncoder();
registry.bind("length-encoder", lengthEncoder);
registry.bind("string-encoder", stringEncoder);

List<ChannelHandler> decoders = new ArrayList<ChannelHandler>();
decoders.add(lengthDecoder);
decoders.add(stringDecoder);

List<ChannelHandler> encoders = new ArrayList<ChannelHandler>();
encoders.add(lengthEncoder);
encoders.add(stringEncoder);

registry.bind("encoders", encoders);
registry.bind("decoders", decoders);
```

Spring 的原生集合支持可用于在应用程序上下文中指定 codec 列表

```
<util:list id="decoders" list-class="java.util.LinkedList">
```

```
        <bean class="org.apache.camel.component.netty4.ChannelHandlerFactories" factory-
method="newLengthFieldBasedFrameDecoder">
            <constructor-arg value="1048576"/>
            <constructor-arg value="0"/>
            <constructor-arg value="4"/>
            <constructor-arg value="0"/>
            <constructor-arg value="4"/>
        </bean>
        <bean class="io.netty.handler.codec.string.StringDecoder"/>
    </util:list>

    <util:list id="encoders" list-class="java.util.LinkedList">
        <bean class="io.netty.handler.codec.LengthFieldPrepender">
            <constructor-arg value="4"/>
        </bean>
        <bean class="io.netty.handler.codec.string.StringEncoder"/>
    </util:list>

    <bean id="length-encoder" class="io.netty.handler.codec.LengthFieldPrepender">
        <constructor-arg value="4"/>
    </bean>
    <bean id="string-encoder" class="io.netty.handler.codec.string.StringEncoder"/>

    <bean id="length-decoder" class="org.apache.camel.component.netty4.ChannelHandlerFactories"
factory-method="newLengthFieldBasedFrameDecoder">
        <constructor-arg value="1048576"/>
        <constructor-arg value="0"/>
        <constructor-arg value="4"/>
        <constructor-arg value="0"/>
        <constructor-arg value="4"/>
    </bean>
    <bean id="string-decoder" class="io.netty.handler.codec.string.StringDecoder"/>
```

然后，在 netty 端点定义中使用 bean 名称作为逗号分隔的列表或包含在 List 中，例如：

```
 from("direct:multiple-codec").to("netty4:tcp://0.0.0.0:{{port}}?
encoders=#encoders&sync=false");

 from("netty4:tcp://0.0.0.0:{{port}}?decoders=#length-decoder,#string-
decoder&sync=false").to("mock:multiple-codec");
```

或通过 XML.

```
<camelContext id="multiple-netty-codecs-context" xmlns="http://camel.apache.org/schema/spring">
    <route>
        <from uri="direct:multiple-codec"/>
        <to uri="netty4:tcp://0.0.0.0:5150?encoders=#encoders&amp;sync=false"/>
    </route>
    <route>
        <from uri="netty4:tcp://0.0.0.0:5150?decoders=#length-decoder,#string-
decoder&amp;sync=false"/>
```

```
        <to uri="mock:multiple-codec"/>
    </route>
</camelContext>
```

242.7. 完成时关闭频道

当充当服务器时，有时希望在关闭频道时（例如，客户端转换已完成）。
您可以通过设置 endpoint 选项 disconnect=true 来完成此操作。

但是，您也可以根据如下消息指示 Camel：
要指示 Camel 关闭频道，您应该添加一个带有键 CamelNettyCloseChannelWhenComplete 设置为布尔值 true 的标头。
例如，以下示例会在将消息写入客户端后关闭频道：

```
from("netty4:tcp://0.0.0.0:8080").process(new Processor() {
    public void process(Exchange exchange) throws Exception {
        String body = exchange.getIn().getBody(String.class);
        exchange.getOut().setBody("Bye " + body);
        // some condition which determines if we should close
        if (close) {

exchange.getOut().setHeader(NettyConstants.NETTY_CLOSE_CHANNEL_WHEN_COMPLETE,
true);
        }
    }
});
```

添加自定义频道管道工厂，以获得对创建的管道的完整控制

242.8. 自定义管道

自定义频道管道通过插入自定义处理程序、编码器和解码器，而无需以非常简单的方式在 Netty Endpoint URL 中指定它们，从而对用户提供完全的控制。

要添加自定义管道，必须通过上下文 registry (JNDIRegistry 或 camel-spring ApplicationContextRegistry 等)创建并注册自定义频道管道工厂。

自定义管道工厂必须构建如下

- Producer 链接的频道管道工厂必须扩展 abstract 类 ClientPipelineFactory。

- 消费者链接的频道管道工厂必须扩展抽象类 ServerInitializerFactory。

- 类应覆盖 initChannel （）方法，以插入自定义处理程序、编码器和解码器。不覆盖 initChannel （） 方法创建没有处理程序、编码器或解码器到管道的管道。

以下示例显示如何创建 ServerInitializerFactory 工厂

242.8.1. 使用自定义管道工厂

```
import io.netty.channel.Channel;
import io.netty.channel.ChannelPipeline;
import io.netty.handler.codec.DelimiterBasedFrameDecoder;
import io.netty.handler.codec.Delimiters;
import io.netty.handler.codec.string.StringDecoder;
import io.netty.handler.codec.string.StringEncoder;
import io.netty.util.CharsetUtil;
import org.apache.camel.component.netty4.NettyConsumer;
import org.apache.camel.component.netty4.ServerInitializerFactory;
import org.apache.camel.component.netty4.handlers.ServerChannelHandler;

public class SampleServerInitializerFactory extends ServerInitializerFactory {
    private int maxLineSize = 1024;
    NettyConsumer consumer;

    public SampleServerInitializerFactory(NettyConsumer consumer) {
        this.consumer = consumer;
    }

    @Override
    public ServerInitializerFactory createPipelineFactory(NettyConsumer consumer) {
        return new SampleServerInitializerFactory(consumer);
    }

    @Override
    protected void initChannel(Channel channel) throws Exception {
        ChannelPipeline channelPipeline = channel.pipeline();

        channelPipeline.addLast("encoder-SD", new StringEncoder(CharsetUtil.UTF_8));
        channelPipeline.addLast("decoder-DELIM", new
DelimiterBasedFrameDecoder(maxLineSize, true, Delimiters.lineDelimiter()));
        channelPipeline.addLast("decoder-SD", new StringDecoder(CharsetUtil.UTF_8));
        // here we add the default Camel ServerChannelHandler for the consumer, to allow Camel
to route the message etc.
        channelPipeline.addLast("handler", new ServerChannelHandler(consumer));
    }
}
```

然后，自定义频道管道工厂可以添加到 registry 中，并使用以下方法在 camel 路由上实例化/使用

```
Registry registry = camelContext.getRegistry();
 ServerInitializerFactory factory = new TestServerInitializerFactory(nettyConsumer);
 registry.bind("spf", factory);
context.addRoutes(new RouteBuilder() {
  public void configure() {
     String netty_ssl_endpoint =
       "netty4:tcp://0.0.0.0:5150?serverInitializerFactory=#spf"
     String return_string =
       "When You Go Home, Tell Them Of Us And Say,"
       + "For Your Tomorrow, We Gave Our Today.";

     from(netty_ssl_endpoint)
      .process(new Processor() {
        public void process(Exchange exchange) throws Exception {
          exchange.getOut().setBody(return_string);
        }
      }
  }
});
```

242.9. 重新使用 NETTY BOSS 和 WORKER 线程池

Netty 有两种类型的线程池： boss 和 worker。默认情况下，每个 Netty 使用者和生成者都有自己的专用线程池。如果要在多个消费者或生成者之间重复使用这些线程池，必须在 Registry 中创建并放入线程池。

例如，使用 Spring XML，我们可以使用带有 2 个 worker 线程的 NettyWorkerPoolBuilder 创建共享 worker 线程池，如下所示：

```
<!-- use the worker pool builder to help create the shared thread pool -->
<bean id="poolBuilder" class="org.apache.camel.component.netty.NettyWorkerPoolBuilder">
  <property name="workerCount" value="2"/>
</bean>

<!-- the shared worker thread pool -->
<bean id="sharedPool" class="org.jboss.netty.channel.socket.nio.WorkerPool"
      factory-bean="poolBuilder" factory-method="build" destroy-method="shutdown">
</bean>
```

提示

对于 boss 线程池，有 Netty producers 的 org.apache.camel.component.netty4.NettyServerBossPoolBuilder 构建器，以及 Netty producers 的 org.apache.camel.component.netty4.NettyClientBossPoolBuilder。

然后，在 Camel 路由中，您可以通过在 URI 中配置 workerPool 选项来引用此 worker 池，如下所示：

```
<route>
  <from uri="netty4:tcp://0.0.0.0:5021?
textline=true&amp;sync=true&amp;workerPool=#sharedPool&amp;usingExecutorService=false"/>
  <to uri="log:result"/>
  ...
</route>
```

如果我们还有另一个路由，我们可以引用共享 worker 池：

```
<route>
  <from uri="netty4:tcp://0.0.0.0:5022?
textline=true&amp;sync=true&amp;workerPool=#sharedPool&amp;usingExecutorService=false"/>
  <to uri="log:result"/>
  ...
</route>
```

以此类推。

242.10. 在单个连接与请求/回复之间多路并发消息

当通过 netty producer 使用 Netty 进行请求/回复消息传递时，默认情况下，每个消息都会通过非共享连接（池）发送。这样可确保回复自动映射到正确的请求线程，以便在 Camel 中进一步路由。在非箱中，请求/回复消息之间的其他词语关联发生，因为回复会返回用于发送请求的同一连接；并且此连接不与他人共享。当响应返回时，连接将返回到连接池，以供其他人重复使用。

但是，如果您想要在单个共享连接上进行多路并发请求/响应，则需要通过设置 producerPoolEnabled=false 来关闭连接池。现在，如果回复超出顺序，则可能会出现交错响应潜在的问题。因此，您需要在请求和回复消息中有一个关联 ID，以便您可以将回复与负责继续处理 Camel 中消息的 Camel 回调关联。要做到这一点，您需要将 NettyCamelStateCorrelationManager 实现为关联管理器，并通过 correlationManager= problemmyManager 选项进行配置。

注意

当您构建自定义关联管理器时，我们建议扩展 TimeoutCorrelationManagerSupport。这提供了对超时和其他复杂性的支持。

您可以在 camel-example-netty-custom-correlation 目录下的示例目录中找到 Apache Camel 源代码的示例。

242.11. 另请参阅

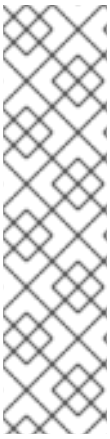
- *Netty HTTP*
- *MINA*

第 243 章 NETTY4 HTTP 组件

从 Camel 版本 2.14 开始提供

netty4-http 组件是 Netty4 组件的扩展，它可通过 Netty4 进行 HTTP 传输。

此 camel 组件支持生成者和消费者端点。



STREAM

Netty 是基于流的，这意味着它收到的输入作为流提交给 Camel。这意味着您只能够读取一次流的内容。

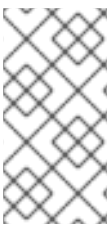
Netty4 HTTP 使用 io.netty.handler.codec.http.HttpObjectAggregator 将整个流读到内存中，以构建整个完整 http 消息。但是，生成的消息仍然是基于流的消息，该消息一次是可读的。

提示

如果消息正文显示为空，或者您需要多次访问数据（例如：执行多播或重新发送错误处理）使用流缓存或将消息正文转换为 String，这会安全地重新读取多次。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-netty4-http</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```



INPUTSTREAM

如果将 InputStream 用作消息正文，并且您想要写入或读取大型数据流（例如 > 2 GB），则必须通过将 disableStreamCache 参数设置为 true 来使用流支持。

示例 1：将大型数据流上传到服务器

```
// Upload a large data stream to the server
from("direct:upstream-call")
        .bean(Helper.class, "prepareStream")
        .to("netty-http:http://localhost:{{port}}/upstream?disableStreamCache=true")
        .log("get ${body}");

// Read a large data stream from the client
from("netty-http:http://0.0.0.0:{{port}}/upstream?disableStreamCache=true")
        .bean(Helper.class, "processStream")
        .to("mock:stream-size");
```

示例 2：从服务器下载大型数据流

```
// Download a large data stream from the server
from("direct:download-call")
        .to("netty-http:http://localhost:{{port}}/downstream?disableStreamCache=true")
        .bean(Helper.class, "asyncProcessStream")
        .log("get ${body}");

// Write a large data stream to the client
from("netty-http:http://0.0.0.0:{{port}}/downstream?disableStreamCache=true")
        .bean(Helper.class, "prepareStream");
```

在下载示例中，您必须从其他线程中的 InputStream 读取，以避免阻止底层流处理程序（请参阅 asyncProcessStream）。

```
    public static void processStream(Exchange exchange) throws Exception {
        InputStream is = exchange.getIn().getBody(InputStream.class);

        byte[] buffer = new byte[1024];
        long read = 0;
        long total = 0;
        while ((read = is.read(buffer, 0, buffer.length)) != -1) {
            total += read;
        }

        exchange.getIn().setBody(new Long(total));
    }

    public static CompletableFuture<Void> asyncProcessStream(Exchange exchange) {
        return CompletableFuture.runAsync(() -> {
            try {
                processStream(exchange);
            } catch (Exception e) {
                exchange.setException(e);
```

```
            }
        });
    }
```

243.1. URI 格式

netty 组件的 URI 方案如下

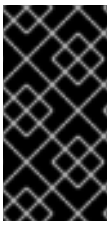
```
netty4-http:http://0.0.0.0:8080[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

注意

查询参数和端点选项.您可以了解 Camel 如何识别 URI 查询参数和端点选项。例如，您可以创建端点 URI，如下所示 - netty4-http:http/example.com?myParam=myValue&compression=true。在本例中，myParam 是 HTTP 参数，而 压缩 是 Camel 端点选项。在这种情况下，Camel 使用的策略是解决可用的端点选项并将其从 URI 中删除。这意味着，对于所讨论的示例，Netty HTTP producer 发送的 HTTP 请求将如下所示 - http//example.com?myParam=myValue，因为 压缩 端点选项将从目标 URL 解析并删除。请记住，您无法使用动态标头（如 CamelHttpQuery）指定端点选项。端点选项只能在端点 URI 定义级别（如 to 或 DSL 元素）指定。

243.2. HTTP 选项



更多选项

此组件从 Netty4 继承所有选项。因此，请务必查看 Netty4 文档。使用此 Netty4 HTTP 组件（如与 UDP 传输相关的选项）时，Netty4 中的一些选项不适用。

Netty4 HTTP 组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
nettyHttpBinding (advanced)	使用自定义 org.apache.camel.component.netty4.http.NettyHttpBinding 来绑定到 Netty 和 Camel Message API。		NettyHttpBinding
configuration (common)	在创建端点时，使用 NettyConfiguration 作为配置。		NettyHttpConfiguration

Name	描述	默认值	类型
headerFilterStrategy (advanced)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 来过滤标头。		HeaderFilterStrategy
securityConfiguration (security)	指的是 org.apache.camel.component.netty4.http.NettyHttpSecurityConfiguration，用于配置安全 Web 资源。		NettyHttpSecurity 配置
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
maximumPoolSize (advanced)	如果正在使用 EventExecutorGroup 的线程池大小。默认值为 16。	16	int
executorService (advanced)	使用给定的 EventExecutorGroup。		EventExecutorGroup
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Netty4 HTTP 端点使用 URI 语法进行配置：

```
netty4-http:protocol:host:port/path
```

使用以下路径和查询参数：

243.2.1. 路径参数(4 参数)：

Name	描述	默认值	类型
protocol	**必需** 使用的协议是 http 或 https		字符串
主机	当作为消费者时，**需要** 本地主机名，如 localhost 或 0.0.0.0。使用制作者时的远程 HTTP 服务器主机名。		字符串
port	主机端口号		int
path	资源路径		字符串

243.2.2. 查询参数(79 参数)：

Name	描述	默认值	类型
bridgeEndpoint (common)	如果选项为 true，则生成者将忽略 Exchange.HTTP_URI 标头，并使用端点的 URI 进行请求。您也可以将 throwExceptionOnFailure 设置为 false，以便生成者发送所有错误响应。在网桥模式下工作的用户将跳过 gzip 压缩和 WWW URL 表单编码（通过添加 Exchange.SKIP_ENCODING 和 Exchange.SKIP_WWW_FORM_URLENCODED 标头到已消耗的交换）。	false	布尔值
disconnect (common)	使用后是否从 Netty Channel 断开连接（关闭）。可用于使用者和制作者。	false	布尔值
keepalive (common)	设置以确保因为不活跃而不会关闭套接字	true	布尔值
reuseAddress (common)	设置以方便套接字多路	true	布尔值
reuseChannel (common)	此选项允许生成者和消费者（在客户端模式中）在处理交换生命周期中重复使用相同的 Netty Channel。如果您需要在 Camel 路由中多次调用服务器并希望使用相同的网络连接，这非常有用。使用此选项时，频道不会返回到连接池，直到 Exchange 完成后；如果 disconnect 选项设为 true，则断开连接。重复使用的频道以一个交换属性的形式存储在 Exchange 中，它带有一个键 NettyConstants SerialNETTY_CHANNEL，它允许您在路由过程中获取频道并使用它。	false	布尔值
sync (common)	将端点设置为单向或请求响应	true	布尔值
tcpNoDelay (common)	设置以提高 TCP 协议性能	true	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
matchOnUriPrefix (consumer)	如果没有找到完全匹配，Camel 是否应该尝试通过匹配 URI 前缀来查找目标消费者。	false	布尔值
send503whenSuspended (consumer)	在使用者被暂停时，是否发送回 HTTP 状态代码 503。如果选项为 false，则 Netty Acceptor 在消费者被暂停时处于未绑定状态，因此客户端无法再连接。	true	布尔值

Name	描述	默认值	类型
backlog (consumer)	允许为 netty consumer (server)配置积压。请注意，积压只是取决于操作系统的最佳努力。将此选项设置为值（如 200、500 或 1000）告知 TCP 堆栈如果未配置此选项，则接受队列的时长，则积压取决于操作系统设置。		int
bossCount (consumer)	当 netty 适用于 nio 模式时，它使用 Netty 中的默认 bossCount 参数，即 1。用户可以使用此操作从 Netty 覆盖默认的 bossCount	1	int
bossGroup (consumer)	设置 BossGroup，可用于处理 NettyEndpoint 中服务器端的新连接		EventLoopGroup
chunkedMaxContentLength (consumer)	值（以字节为单位）在 Netty HTTP 服务器上接收的每个块帧的最大内容长度（以字节为单位）。	1048576	int
compression (consumer)	如果客户端从 HTTP 标头支持，则允许使用 gzip/定义在 Netty HTTP 服务器上压缩。	false	布尔值
disconnectOnNoReply (consumer)	如果启用了同步，则此选项将指定 NettyConsumer（如果它应该断开连接），没有回复回来。	true	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
httpMethodRestrict (consumer)	要禁用 Netty HTTP 使用者上的 HTTP 方法。您可以指定用逗号分开的多个。		字符串
mapHeaders (consumer)	如果启用了这个选项，那么在从 Netty 绑定到 Camel Message 的过程中，标头也会被映射（如作为标头添加到 Camel 消息）。您可以关闭这个选项来禁用这个选项。标头仍然可以从 org.apache.camel.component.netty.http.NettyHttpMessage 消息访问，该消息返回 Netty HTTP 请求 io.netty.handler.codec.http.HttpRequest 实例。	true	布尔值
maxHeaderSize (consumer)	所有标头的最大长度。如果每个标头的长度总和超过这个值，则会引发 io.netty.handler.codec.TooLongFrameException。	8192	int
nettyServerBootstrapFactory (consumer)	使用自定义 NettyServerBootstrapFactory		NettyServerBootstrap Factory

Name	描述	默认值	类型
nettySharedHttpServer (consumer)	使用共享的 Netty HTTP 服务器。如需了解更多详细信息，请参阅 Netty HTTP 服务器示例。		NettySharedHttpServer
noReplyLogLevel (consumer)	如果启用了同步，这个选项会指示 NettyConsumer，在记录一个没有回复时要使用的日志记录级别。	WARN	LoggingLevel
serverClosedChannelExceptionCaughtLogLevel (consumer)	如果服务器(NettyConsumer)捕获 java.nio.channels.ClosedChannelException，然后使用此日志级别进行记录。这用于避免记录关闭的频道异常，因为客户端可以立即断开，然后在 Netty 服务器中造成大量关闭的异常。	DEBUG	LoggingLevel
serverExceptionCaughtLog Level (consumer)	如果服务器(NettyConsumer)捕获异常，则使用此日志级别进行日志记录。	WARN	LoggingLevel
serverInitializerFactory (consumer)	使用自定义 ServerInitializerFactory		ServerInitializerFactory
traceEnabled (consumer)	指定是否为这个 Netty HTTP 使用者启用 HTTP TRACE。默认情况下关闭 TRACE。	false	布尔值
urlDecodeHeaders (consumer)	如果启用了这个选项，那么在从 Netty 绑定到 Camel Message 的过程中，标头值将被解码（例如 %20 将是一个空格字符）。注意此选项由默认的 org.apache.camel.component.netty.http.NettyHttpBinding 使用，因此如果您实施自定义 org.apache.camel.component.netty4.http.NettyHttpBinding，则需要相应地将标头解码到此选项。	false	布尔值
useExecutorService (consumer)	是否使用排序的线程池，确保同一通道上按顺序处理事件。	true	布尔值
connectTimeout (producer)	等待套接字连接可用的时间。值以毫秒为单位。	10000	int
cookieHandler (producer)	配置 Cookie 处理程序，以维护 HTTP 会话		CookieHandler
requestTimeout (producer)	在调用远程服务器时，允许 Netty producer 使用超时。默认情况下，没有使用超时。该值在 milli 秒内，如 30000 为 30 秒。requestTimeout 使用 Netty 的 ReadTimeoutHandler 来触发超时。		long
throwExceptionOnFailure (producer)	如果来自远程服务器的失败响应，用于禁用抛出 HttpOperationFailedException。这样，您可以获取所有响应，而不考虑 HTTP 状态代码。	true	布尔值

Name	描述	默认值	类型
clientInitializerFactory (producer)	使用自定义 ClientInitializerFactory		ClientInitializerFactory
lazyChannelCreation (producer)	如果远程服务器在 Camel producer 启动时未启动并运行，则可以创建频道以避免异常。	true	布尔值
okStatusCodeRange (producer)	被视为成功响应的状态代码。值包含。可以定义多个范围，用逗号分开，例如 200-204,209,301-304。每个范围必须是单个数字，或从 到，其中包含短划线。默认范围为 200-299	200-299	字符串
producerPoolEnabled (producer)	producer 池是否已启用。重要：如果您关闭此设置，则为生成者使用单个共享连接，如果您执行请求/回复。这意味着，如果回复超出顺序，则存在交错响应潜在的问题。因此，您需要在请求和回复消息中有一个关联 ID，以便您可以将回复与负责继续处理 Camel 中消息的 Camel 回调关联。为此，您需要将 NettyCamelStateCorrelationManager 实施为关联管理器，并通过 correlationManager 选项进行配置。如需了解更多详细信息，请参阅 correlationManager 选项。	true	布尔值
producerPoolMaxActive (producer)	设置池可分配给客户端或闲置等待签出的对象数量上限。对没有限制，使用负值。	-1	int
producerPoolMaxIdle (producer)	设置池中空闲实例数量上限。	100	int
producerPoolMinEvictable Idle (producer)	设置对象在空闲对象驱除有资格驱除前，对象可能会在池中闲置的最小时间（值为 millis））。	300000	long
producerPoolMinIdle (producer)	设置制作者池中允许的最小实例数量，然后再驱除线程（如果活跃）生成新对象。		int
useRelativePath (producer)	设置是否在 HTTP 请求中使用相对路径。	false	布尔值
allowSerializedHeaders (advanced)	仅在 transferExchange 为 true 时用于 TCP。当设置为 true 时，标头和属性中的串行对象将添加到交换中。否则，Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。	false	布尔值
bootstrapConfiguration (advanced)	使用自定义配置的 NettyServerBootstrapConfiguration 来配置此端点。		NettyServerBootstrap Configuration
channelGroup (advanced)	使用显式 ChannelGroup。		ChannelGroup

Name	描述	默认值	类型
配置 （高级）	使用自定义配置的 NettyHttpConfiguration 来配置此端点。		NettyHttpConfiguration
disableStreamCache (advanced)	确定来自 Netty HttpRequest CPUfreqgetContent （）或 HttpResponset"getContent （）的原始输入流是否被缓存(Camel 会将流读取到基于轻量级内存的流缓存)缓存中。默认情况下，Camel 将缓存 Netty 输入流，以支持多次读取，以确保其 Camel 可以从流检索所有数据。但是，当您需要访问原始流时，您可以将此选项设置为 true，例如将其直接流传输到文件或其他持久性存储。请注意，如果您启用这个选项，则无法多次读取 Netty 流，您需要手动重置 Netty 原始流上的读取器索引。另外，Netty 会在 Netty HTTP 服务器/HTTP 客户端被处理时自动关闭 Netty 流，这意味着如果异步路由引擎正在使用，则任何可继续路由 org.apache.camel.Exchange 可能无法读取 Netty 流，因为 Netty 已关闭它。	false	布尔值
headerFilterStrategy (advanced)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 来过滤标头。		HeaderFilterStrategy
nativeTransport (advanced)	是否使用原生传输而不是 NIO。原生传输利用主机操作系统，且仅在某些平台上受支持。您需要为正在使用的主机操作系统添加 netty JAR。详情请查看 ：http://netty.io/wiki/native-transports.html	false	布尔值
nettyHttpBinding (advanced)	使用自定义 org.apache.camel.component.netty4.http.NettyHttpBinding 来绑定到 Netty 和 Camel Message API。		NettyHttpBinding
选项 (advanced)	允许使用选项配置其他 netty 选项作为前缀。例如：option.child.keepAlive=false 设置 netty 选项 child.keepAlive=false。有关可以使用的选项，请参阅 Netty 文档。		Map
receiveBufferSize (advanced)	在入站通信中使用的 TCP/UDP 缓冲区大小。大小为字节。	65536	int
receiveBufferSizePredictor (advanced)	配置缓冲区大小预测。请参阅 Jetty 文档和此邮件线程的详细信息。		int
sendBufferSize (advanced)	在出站通信中使用的 TCP/UDP 缓冲大小。大小为字节。	65536	int
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

Name	描述	默认值	类型
transferException (advanced)	如果在消费者端启用并交换失败处理，如果原因 Exception 在响应中作为 application/x-java-serialized-object 内容类型发送序列化，则结果为 application/x-java-serialized-object 内容类型。在生产者侧，异常将反序列化并丢弃为原样，而不是 HttpOperationFailedException。原因异常需要按顺序处理。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
transferExchange (advanced)	仅用于 TCP。您可以通过线线传输交换，而不是只传输正文。以下字段被传输： 在 body, Out body, fault body, In headers, Out headers, fault headers, Exchange properties, Exchange exception。这要求对象是序列化的。Camel 将排除任何非序列化对象，并将其记录在 WARN 级别。	false	布尔值
workerCount (advanced)	当 netty 适用于 nio 模式时，它会使用来自 Netty 的默认 workerCount 参数，即 cpu_core_threads x 2。用户可以使用此操作从 Netty 覆盖默认的 workerCount。		int
workerGroup (advanced)	使用显式 EventLoopGroup 作为 boss 线程池。例如，要与多个消费者或生成者共享线程池。默认情况下，每个消费者或生成者都有自己的 worker 池，有 2 个 x cpu count 内核线程。		EventLoopGroup
decoder (codec)	**弃用** 以使用单个解码器。这个选项已弃用，改为使用 encoders。		ChannelHandler
decoders (codec)	要使用的解码器列表。您可以使用以逗号分开的值的 String，并在 Registry 中查找值。只需记住使用 # 前缀，以便 Camel 知道它应该查找。		字符串
encoder (codec)	**弃用** 以使用单个编码器。这个选项已弃用，改为使用 encoders。		ChannelHandler
编码器 (codec)	要使用的编码器列表。您可以使用以逗号分开的值的 String，并在 Registry 中查找值。只需记住使用 # 前缀，以便 Camel 知道它应该查找。		字符串
enabledProtocols (security)	使用 SSL 时要启用的协议	TLSv1, TLSv1.1, TLSv1.2	字符串
keyStoreFile (security)	用于加密的客户端侧证书密钥存储		File

Name	描述	默认值	类型
keyStoreFormat (security)	用于有效负载加密的密钥存储格式。如果没有设置，则默认为 JKS		字符串
keyStoreResource (security)	用于加密的客户端侧证书密钥存储。默认情况下从 classpath 加载，但您可以使用 classpath:、file: 或 http: 前缀来加载来自不同系统的资源。		字符串
needClientAuth (security)	配置服务器在使用 SSL 时是否需要客户端身份验证。	false	布尔值
密码短语 (security)	要使用 SSH 加密/解密有效负载的密码设置		字符串
securityConfiguration (security)	指的是 org.apache.camel.component.netty4.http.NettyHttpSecurityConfiguration，用于配置安全 Web 资源。		NettyHttpSecurity 配置
securityOptions (security)	使用映射中的键/值对配置 NettyHttpSecurityConfiguration		Map
securityProvider (security)	用于有效负载加密的安全供应商。如果没有设置，则默认为 SunX509。		字符串
SSL (security)	设置以指定是否将 SSL 加密应用到此端点	false	布尔值
sslClientCertHeaders (security)	启用和 SSL 模式时，Netty 使用者将增强 Camel 消息，其中包含客户端证书的信息，如主题名称、签发者名称、序列号和有效日期范围。	false	布尔值
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters
sslHandler (security)	引用可用于返回 SSL 处理程序的类		SslHandler
trustStoreFile (security)	用于加密的服务器端证书密钥存储		File
trustStoreResource (security)	用于加密的服务器端证书密钥存储。默认情况下从 classpath 加载，但您可以使用 classpath:、file: 或 http: 前缀来加载来自不同系统的资源。		字符串

243.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 33 选项，如下所列。

Name	描述	默认值	类型
camel.component.netty4-http.configuration.allow-default-codec			布尔值
camel.component.netty4-http.configuration.bridge-endpoint	如果选项为 true，则生成者将忽略 Exchange.HTTP_URI 标头，并使用端点的 URI 进行请求。您也可以将 throwExceptionOnFailure 设置为 false，以便生成者发送所有错误响应。在网桥模式下工作的用户将跳过 gzip 压缩和 WWW URL 表单编码（通过添加 Exchange.SKIP_ENCODING 和 Exchange.SKIP_WWW_FORM_URLENCODED 标头到已消耗的交换）。	false	布尔值
camel.component.netty4-http.configuration.chunked-max-content-length	值（以字节为单位）在 Netty HTTP 服务器上接收的每个块帧的最大内容长度（以字节为单位）。	1048576	整数
camel.component.netty4-http.configuration.compression	如果客户端从 HTTP 标头支持，则允许使用 gzip/定义在 Netty HTTP 服务器上压缩。	false	布尔值
camel.component.netty4-http.configuration.disable-stream-cache	确定来自 Netty HttpRequest CPUfreqgetContent （）或 HttpResponset"getContent （）的原始输入流是否被缓存(Camel 会将流读取到基于轻量级内存的流缓存)缓存中。默认情况下，Camel 将缓存 Netty 输入流，以支持多次读取，以确保其 Camel 可以从流检索所有数据。但是，当您需要访问原始流时，您可以将此选项设置为 true，例如将其直接流传输到文件或其他持久性存储。请注意，如果您启用这个选项，则无法多次读取 Netty 流，您需要手动重置 Netty 原始流上的读取器索引。另外，Netty 会在 Netty HTTP 服务器/HTTP 客户端被处理时自动关闭 Netty 流，这意味着如果异步路由引擎正在使用，则任何可继续路由 org.apache.camel.Exchange 可能无法读取 Netty 流，因为 Netty 已关闭它。	false	布尔值
camel.component.netty4-http.configuration.host	作为消费者，本地主机名，如 localhost, 或 0.0.0.0。使用制作者时的远程 HTTP 服务器主机名。		字符串

Name	描述	默认值	类型
camel.component.netty4-http.configuration.map-headers	如果启用了这个选项，那么在从 Netty 绑定到 Camel Message 的过程中，标头也会被映射（如作为标头添加到 Camel 消息）。您可以关闭这个选项来禁用这个选项。标头仍然可以从 org.apache.camel.component.netty.http.NettyHttpMessage 消息访问，该消息返回 Netty HTTP 请求 io.netty.handler.codec.http.HttpRequest 实例。	true	布尔值
camel.component.netty4-http.configuration.match-on-uri-prefix	如果没有找到完全匹配，Camel 是否应该尝试通过匹配 URI 前缀来查找目标消费者。	false	布尔值
camel.component.netty4-http.configuration.max-header-size	所有标头的最大长度。如果每个标头的长度总和超过这个值，则会引发 io.netty.handler.codec.TooLongFrameException。	8192	整数
camel.component.netty4-http.configuration.ok-status-code-range	被视为成功响应的状态代码。值包含。可以定义多个范围，用逗号分开，例如 200-204,209,301-304。每个范围必须是单个数字，或从 到，其中包含短划线。默认范围为 200-299	200-299	字符串
camel.component.netty4-http.configuration.path	资源路径		字符串
camel.component.netty4-http.configuration.port	端口号。http 为 80，https 为 443。		整数
camel.component.netty4-http.configuration.protocol	使用的协议(http 或 https)		字符串
camel.component.netty4-http.configuration.send503when-suspended	在使用者被暂停时，是否发送回 HTTP 状态代码 503。如果选项为 false，则 Netty Acceptor 在消费者被暂停时处于未绑定状态，因此客户端无法再连接。	true	布尔值

Name	描述	默认值	类型
camel.component.netty4-http.configuration.throw-exception-on-failure	如果来自远程服务器的失败响应，用于禁用抛出 HttpOperationFailedException。这样，您可以获取所有响应，而不考虑 HTTP 状态代码。	true	布尔值
camel.component.netty4-http.configuration.transfer-exception	如果在消费者端启用并交换失败处理，如果原因 Exception 在响应中作为 application/x-java-serialized-object 内容类型发送序列化，则结果为 application/x-java-serialized-object 内容类型。在生产者侧，异常将反序列化并丢弃为原样，而不是 HttpOperationFailedException。原因异常需要按顺序处理。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
camel.component.netty4-http.configuration.url-decode-headers	如果启用了这个选项，那么在从 Netty 绑定到 Camel Message 的过程中，标头值将被解码（例如 %20 将是一个空格字符）。注意此选项由默认的 org.apache.camel.component.netty.http.NettyHttpBinding 使用，因此如果您实施自定义 org.apache.camel.component.netty4.http.NettyHttpBinding，则需要相应地将标头解码到此选项。	false	布尔值
camel.component.netty4-http.configuration.use-relative-path	设置是否在 HTTP 请求中使用相对路径。	false	布尔值
camel.component.netty4-http.enabled	启用 netty4-http 组件	true	布尔值
camel.component.netty4-http.executor-service	使用给定的 EventExecutorGroup。选项是一个 io.netty.util.concurrent.EventExecutorGroup 类型。		字符串
camel.component.netty4-http.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 来过滤标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		字符串
camel.component.netty4-http.maximum-pool-size	如果正在使用 EventExecutorGroup 的线程池大小。默认值为 16。	16	整数

Name	描述	默认值	类型
camel.component.netty4-http.netty-http-binding	使用自定义 org.apache.camel.component.netty4.http.NettyHttpBinding 来绑定到 Netty 和 Camel Message API。选项是 org.apache.camel.component.netty4.http.NettyHttpBinding 类型。		字符串
camel.component.netty4-http.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.netty4-http.security-configuration.authenticate	是否启用身份验证 <p/>，这默认是启用的。		布尔值
camel.component.netty4-http.security-configuration.constraint	支持的 restricted. <p/> 当前只支持 Basic。		字符串
camel.component.netty4-http.security-configuration.login-denied-logging-level	设置用于日志记录被拒绝的登录尝试(incl stacktraces) <p/> （默认为 DEBUG）的日志记录级别。		LoggingLevel
camel.component.netty4-http.security-configuration.realm	设置要使用的域的名称。		字符串
camel.component.netty4-http.security-configuration.role-class-name			字符串

Name	描述	默认值	类型
camel.component.netty4-http.security-configuration.security-authenticator	设置 {@link SecurityAuthenticator}，用于验证 {@link HttpPrincipal}。		SecurityAuthenticator
camel.component.netty4-http.security-configuration.security-constraint	将一个 {@link SecurityConstraint} 设置为用来检查 Web 资源是否受限制或没有 <p/>，默认为 <tt>null</tt>，这意味着所有资源都受到限制。		SecurityConstraint
camel.component.netty4-http.ssl-context-parameters	使用 SSLContextParameters 配置安全性。选项是 org.apache.camel.util.jsse.SSLContextParameters 类型。		字符串
camel.component.netty4-http.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

243.4. 消息标头

以下标头可用于制作者来控制 HTTP 请求。

Name	类型	描述
CamelHttpMethod	**字符串**	允许控制使用什么 HTTP 方法，如 GET、POST 和 TRACE 等。类型也可以是 **io.netty.handler.codec.http.HttpMethod** 实例。
CamelHttpQuery	**字符串**	允许将 URI 查询参数作为 **String** 值提供，用于覆盖端点配置。使用 & 符号分隔多个参数。例如： **foo=bar&beer=yes**。
CamelHttpPath	**字符串**	允许将 URI 上下文路径和查询参数作为 **String** 值提供，用于覆盖端点配置。这允许重复使用同一制作者来调用同一远程 http 服务器，但使用动态上下文路径和查询参数。
Content-Type	**字符串**	设置 HTTP 正文的内容类型：例如： **text/plain; charset="UTF-8"。**

Name	类型	描述
CamelHttpResponseCode	**int**	允许设置要使用的 HTTP 状态代码。默认情况下，200 用于成功，使用 500 个失败。

当路由从 Netty4 HTTP 端点开始时，以下标头作为 meta-data 提供：

表中的描述在带有的路由中使用偏移量： from ("netty4-http:http:0.0.0.0:8080/myapp")...

Name	类型	描述
CamelHttpMethod	**字符串**	使用的 HTTP 方法，如 GET、POST 和 TRACE 等。
CamelHttpUrl	**字符串**	URL 包括协议、主机和端口等： **http://0.0.0.0:8080/myapp**
CamelHttpUri	**字符串**	没有协议、主机和端口的 URI，例如： **/myapp**
CamelHttpQuery	**字符串**	任何查询参数，如 **foo=bar&beer=yes**
CamelHttpRawQuery	**字符串**	任何查询参数，如 **foo=bar&beer=yes**。以原始形式存储，因为它们到达消费者（例如，在 URL 解码之前）。
CamelHttpPath	**字符串**	其他 context-path。如果名为 context-path **/myapp** 的客户端，则该值为空。如果客户端调用 **/myapp/mystuff**，则此标题值为 **/mystuff**。换句话说，在路由端点上配置的 context-path 后，其值代表。
CamelHttpCharacterEncoding	**字符串**	content-type 标头中的 charset。

Name	类型	描述
CamelHttpAuthentication	**字符串**	如果用户是使用 HTTP Basic 进行身份验证的，则使用值 **Basic** 添加此标头。
Content-Type	**字符串**	如果提供，内容类型。例如： **text/plain; charset="UTF-8"**。

243.5. 访问 NETTY 类型

此组件使用 org.apache.camel.component.netty4.http.NettyHttpMessage 作为 Exchange 上的消息实施。这样，最终用户可以根据需要访问原始的 Netty 请求/响应实例，如下所示。请注意，原始响应可能无法始终访问。

```
io.netty.handler.codec.http.HttpRequest request =
exchange.getIn(NettyHttpMessage.class).getHttpRequest();
```

243.6. 例子

在以下路由中，我们使用 Netty4 HTTP 作为 HTTP 服务器，它返回一个硬编码的"Bye World"消息。

```
from("netty4-http:http://0.0.0.0:8080/foo")
  .transform().constant("Bye World");
```

我们也可以使用 Camel 调用此 HTTP 服务器，以及 ProducerTemplate，如下所示：

```
String out = template.requestBody("netty4-http:http://0.0.0.0:8080/foo", "Hello World",
String.class);
System.out.println(out);
```

我们以输出形式返回 "Bye World"。

243.7. 如何让 NETTY 匹配通配符

默认情况下，Netty4 HTTP 仅在确切的 uri 上匹配。但是，您可以指示 Netty 匹配前缀。例如：

```
from("netty4-http:http://0.0.0.0:8123/foo").to("mock:foo");
```

在上述 Netty4 HTTP 的路由中，只有 uri 完全匹配时才会匹配。因此，如果您输入 http://0.0.0.0:8123/foo，它将匹配，但如果是 http://0.0.0.0:8123/foo/bar，则不会匹配。

因此，如果要启用通配符匹配，如下所示：

```
from("netty4-http:http://0.0.0.0:8123/foo?matchOnUriPrefix=true").to("mock:foo");
```

因此，Netty 与以 foo 开头的任何端点匹配。

要匹配 任何 端点，您可以执行以下操作：

```
from("netty4-http:http://0.0.0.0:8123?matchOnUriPrefix=true").to("mock:foo");
```

243.8. 在同一端口使用多个路由

在同一 CamelContext 中，您可以拥有来自共享同一端口的 Netty4 HTTP 的多个路由（如 io.netty.bootstrap.ServerBootstrap 实例）。执行此操作需要在路由中有多个 bootstrap 选项，因为路由将共享相同的 io.netty.bootstrap.ServerBootstrap 实例。实例将使用创建的第一个路由中的选项进行配置。

路由必须相同配置的选项是 org.apache.camel.component.netty4.NettyServerBootstrapConfiguration 配置类中定义的所有选项。如果您使用不同的选项配置了另一个路由，Camel 会在启动时抛出异常，表示选项不相同。要缓解这个问题，请确保所有选项都相同。

下面是有两个路由共享同一端口的示例：

共享同一端口的两个路由

```
from("netty4-http:http://0.0.0.0:{{port}}/foo")
  .to("mock:foo")
  .transform().constant("Bye World");

from("netty4-http:http://0.0.0.0:{{port}}/bar")
  .to("mock:bar")
  .transform().constant("Bye Camel");
```

下面是一个错误配置的 2nd 路由的示例，它没有相同的 org.apache.camel.component.netty4.NettyServerBootstrapConfiguration 选项作为 1st 路由。这将导致 Camel 在启动时失败。

两个路由共享同一端口，但第二路由配置错误，并将在启动时失败

```
from("netty4-http:http://0.0.0.0:{{port}}/foo")
  .to("mock:foo")
  .transform().constant("Bye World");

// we cannot have a 2nd route on same port with SSL enabled, when the 1st route is NOT
from("netty4-http:http://0.0.0.0:{{port}}/bar?ssl=true")
  .to("mock:bar")
  .transform().constant("Bye Camel");
```

243.8.1. 使用多个路由重复使用相同的服务器 bootstrap 配置

通过在 org.apache.camel.component.netty4.NettyServerBootstrapConfiguration 类型的单个实例中配置通用服务器 bootstrap 选项，我们可以对 Netty4 HTTP 用户使用 bootstrapConfiguration 选项来引用和重复使用所有消费者的相同选项。

```
<bean id="nettyHttpBootstrapOptions"
class="org.apache.camel.component.netty4.NettyServerBootstrapConfiguration">
  <property name="backlog" value="200"/>
  <property name="connectionTimeout" value="20000"/>
  <property name="workerCount" value="16"/>
</bean>
```

在您引用此选项的路由中，如下所示

```
<route>
  <from uri="netty4-http:http://0.0.0.0:{{port}}/foo?
bootstrapConfiguration=#nettyHttpBootstrapOptions"/>
  ...
</route>

<route>
  <from uri="netty4-http:http://0.0.0.0:{{port}}/bar?
bootstrapConfiguration=#nettyHttpBootstrapOptions"/>
  ...
</route>

<route>
  <from uri="netty4-http:http://0.0.0.0:{{port}}/beer?
```

```
bootstrapConfiguration=#nettyHttpBootstrapOptions"/>
  ...
</route>
```

243.8.2. 使用 OSGi 容器中多个捆绑包的多个路由重复使用相同的服务器引导配置

详情请查看 Netty HTTP 服务器 示例。

243.9. 使用 HTTP 基本身份验证

Netty HTTP 使用者通过指定要使用的安全域名称来支持 HTTP 基本身份验证，如下所示

```
<route>
  <from uri="netty4-http:http://0.0.0.0:{{port}}/foo?securityConfiguration.realm=karaf"/>
  ...
</route>
```

域名称是必需的，才能启用基本身份验证。默认情况下，使用了基于 JAAS 的验证器，它使用指定的域名称（上例中的karaf），并使用此域的 JAAS 域和 JAAS \{{LoginModule}}s 进行身份验证。

Apache Karaf / ServiceMix 的最终用户开箱即用有一个 karaf 域，因此上面的示例将在这些容器中开箱即用。

243.9.1. 在 Web 资源中指定 ACL

org.apache.camel.component.netty4.http.SecurityConstraint 允许定义对 Web 资源的约束。并且提供了 org.apache.camel.component.netty.http.SecurityConstraintMapping，允许使用角色轻松定义包含和排除项。

例如，在 XML DSL 中，我们定义约束 bean：

```
<bean id="constraint" class="org.apache.camel.component.netty4.http.SecurityConstraintMapping">
  <!-- inclusions defines url -> roles restrictions -->
  <!-- a * should be used for any role accepted (or even no roles) -->
  <property name="inclusions">
    <map>
      <entry key="/*" value="*"/>
      <entry key="/admin/*" value="admin"/>
      <entry key="/guest/*" value="admin,guest"/>
    </map>
  </property>
  <!-- exclusions is used to define public urls, which requires no authentication -->
```

```
    <property name="exclusions">
      <set>
        <value>/public/*</value>
      </set>
    </property>
  </bean>
```

上面的约束已定义，以便

- 对 Thycotic 的访问会被限制，并且接受任何角色（也如果没有角色，也接受任何角色）
- 访问 /admin86] 需要 admin 角色
- 访问 /guest205 需要 admin 或 guest 角色
- 对 /public netobserv 的访问是一个排除项，这意味着不需要身份验证，因此对于没有登录的任何人都公开了身份验证

要使用此约束，我们只需要引用 bean id，如下所示：

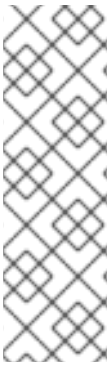
```
<route>
  <from uri="netty4-http:http://0.0.0.0:{{port}}/foo?
matchOnUriPrefix=true&amp;securityConfiguration.realm=karaf&amp;securityConfiguration.securityCon
straint=#constraint"/>
  ...
</route>
```

243.10. 实施反向代理

Netty4 组件可以充当反向代理。这意味着这些标头从 HTTP 请求请求行中收到的绝对 URL 填充：

- Exchange.HTTP_SCHEME
- Exchange.HTTP_HOST

- *Exchange.HTTP_PORT*



HTTP 标头干扰

HTTP 标头可能会干扰客户端或下游处理。

例如，如果更改了 HTTP 正文，则需要重新计算或删除 Content-Length 标头来反映更改。

在以下示例中，HTTP 代理将响应从原始服务器转换为大写：

Example

```
from("netty-http:proxy://0.0.0.0:8080")
   .toD("netty-http:"
      + "${headers." + Exchange.HTTP_SCHEME + "}://"
      + "${headers." + Exchange.HTTP_HOST + "}:"
      + "${headers." + Exchange.HTTP_PORT + "}")
   .process(this::processResponse);

void processResponse(final Exchange exchange) {
   final NettyHttpMessage message = exchange.getIn(NettyHttpMessage.class);
   final FullHttpResponse response = message.getHttpResponse();

   final ByteBuf buf = response.content();
   final String string = buf.toString(StandardCharsets.UTF_8);

   buf.resetWriterIndex();
   ByteBufUtil.writeUtf8(buf, string.toUpperCase(Locale.US));
}
```

243.11. 另请参阅

- *配置 Camel*
- *组件*

- *端点*
- *开始使用*
- *Netty*
- *Netty HTTP 服务器示例*
- *Jetty*

第 244 章 NSQ 组件

从 Camel 版本 2.23 开始提供

从 Camel 版本 2.23 开始提供

NSQ 是一个实时分布式消息传递平台。

Maven 用户需要将以下依赖项添加到此组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-nsq</artifactId>
  <!-- use the same version as your Camel core version -->
  <version>x.y.z</version>
</dependency>
```

244.1. URI 格式

```
nsq:servers[?options]
```

其中 servers 代表 NSQ 服务器列表 - 如果生成者为消费者和 nsqd 服务器，则 nsqlookupd 服务器。

244.2. 选项

NSQ 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

NSQ 端点使用 URI 语法进行配置：

```
nsq:servers
```

使用以下路径和查询参数：

244.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
服务器	**必需** 一个或多个 nsqlookupd servers (consumer)或 nsqd servers (producer)的主机名。		字符串

244.2.2. 查询参数(16 参数)：

Name	描述	默认值	类型
topic (common)	**必需** 要使用的主题名称		字符串
userAgent (common)	用于标识客户端类型的 String		字符串
autoFinish (consumer)	当从 quese 和处理 Exchange 之前检索，它会自动完成 NSQ 消息。	true	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
channel (consumer)	要使用的频道名称		字符串
lookupInterval (consumer)	查找重试间隔	5000	long
lookupServerPort (consumer)	nsqdlookupd 服务器的端口	4161	int
messageTimeout (consumer)	消费者的 NSQ 消息超时。	-1	long
poolSize (consumer)	消费者池大小	10	int

Name	描述	默认值	类型
requeueInterval (consumer)	requeue 间隔	-1	long
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
port (producer)	nsqd 服务器的端口	4150	int
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
安全 （安全）	设置 secure 选项表示需要 TLS	false	布尔值
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters

244.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.nsq.enabled	是否启用 nsq 组件的自动配置。这默认是启用的。		布尔值
camel.component.nsq.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.nsq.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

第 245 章 OGNL 语言

从 Camel 版本 1.1 开始提供

Camel 允许将 OGNL 用作 Expression 或 Predicate the DSL 或 Xml 配置。

您可以使用 OGNL 在 Message Filter 中创建一个 predicates，或作为 Recipient List 的 Expression

您可以使用 OGNL dot 表示法调用操作。如果实例有一个正文，其中包含具有 getFamilyName 方法的 POJO，如下所示：

```
"request.body.familyName"
  // or
"getRequest().getBody().getFamilyName()"
```

245.1. OGNL 选项

OGNL 语言支持 1 个选项，如下所列。

Name	默认值	Java 类型	描述
trim	**true**	**布尔值**	是否修剪值以移除前导和结尾的空格和换行符

245.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.language.ognl.enabled	启用 ognl 语言	true	布尔值
camel.language.ognl.trim	是否修剪值以移除前导和结尾的空格和换行符	true	布尔值

245.3. 变量

变量	类型	描述
这	Exchange	Exchange 是根对象
交换	Exchange	Exchange 对象
例外	Throwable	Exchange 异常（如果有）
exchangeId	字符串	交换 ID
Fulting	消息	Fault 消息（如果有）
request	消息	exchange.in 消息
response	消息	exchange.out 消息（如果有）
属性	Map	交换属性
property (name)	对象	给定名称的属性
property (name, type)	类型	给定名称中的属性作为给定类型

245.4. SAMPLES

例如，您可以在 XML 中的 Message Filter 中使用 OGNL

```
<route>
  <from uri="seda:foo"/>
  <filter>
    <ognl>request.headers.foo == 'bar'</ognl>
    <to uri="seda:bar"/>
  </filter>
</route>
```

和使用 Java DSL 的示例：

```
from("seda:foo").filter().ognl("request.headers.foo == 'bar'").to("seda:bar");
```

245.5. 从外部资源载入脚本

从 Camel 2.11 开始提供

您可以对脚本进行外部化，并让 Camel 从资源（如 "classpath:"、"file:" 或 "http:" ）加载它。这可以通过以下语法完成：" resource:scheme:location"，例如引用您可以进行的类路径上的文件：

```
.setHeader("myHeader").ognl("resource:classpath:myognl.txt")
```

245.6. 依赖项

要在 camel 路由中使用 OGNL，您需要对实现 OGNL 语言的 camel-ognl 添加依赖项。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-ognl</artifactId>
  <version>x.x.x</version>
</dependency>
```

否则，您还需要 OGNL

第 246 章 OLINGO2 组件

从 Camel 版本 2.14 开始提供

Olingo2 组件利用 Apache Olingo 版本 2.0 API 与 OData 2.0 兼容服务交互。许多流行的商业和企业供应商和产品支持 OData 协议。有关支持产品的示例列表可在 OData 网站 中找到。

Olingo2 组件支持读取源、增量源、实体、简单和复杂属性、链接、计数、使用 custom 和 OData 系统查询参数。它支持更新实体、属性和关联链接。它还支持以单一 OData batch 操作形式提交查询并更改请求。

组件支持为 OData 服务连接配置 HTTP 连接参数和标头。这允许根据目标 OData 服务的要求配置 SSL、OAuth2.0 等。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-olingo2</artifactId>
  <version>${camel-version}</version>
</dependency>
```

246.1. URI 格式

```
olingo2://endpoint/<resource-path>?[options]
```

246.2. OLINGO2 选项

Olingo2 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	使用共享配置		Olingo2Configuration
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Olingo2 端点使用 URI 语法进行配置：

```
olingo2:apiName/methodName
```

使用以下路径和查询参数：

246.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
apiName	**需要** 执行什么操作		Olingo2ApiName
methodName	**必需的** 所选操作使用哪些子操作		字符串

246.2.2. 查询参数(14 参数)：

Name	描述	默认值	类型
connectTimeout (common)	HTTP 连接创建超时（以毫秒为单位），默认为 30,000 (30 秒)	30000	int
contentType (common)	content-Type 标头值可用于指定 JSON 或 XML 消息格式，默认为 application/json;charset=utf-8	application/json;charset=utf-8	字符串
httpAsyncClientBuilder (common)	自定义 HTTP async 客户端构建器用于更复杂的 HTTP 客户端配置，覆盖 connectionTimeout, socketTimeout, proxy 和 sslContext。请注意，在构建器中指定一个 socketTimeout MUST，否则 OData 请求可能会无限期阻断		HttpAsyncClientBuilder

Name	描述	默认值	类型
httpClientBuilder (common)	自定义 HTTP 客户端构建器用于更复杂的 HTTP 客户端配置，覆盖 connectionTimeout, socketTimeout, proxy 和 sslContext。请注意，在构建器中指定一个 socketTimeout MUST，否则 OData 请求可能会无限期阻断		HttpClientBuilder
httpHeaders (common)	自定义 HTTP 标头来注入每个请求，这可能包括 OAuth 令牌等。		Map
inBody (common)	设置要在交换 In Body 中传递的参数名称		字符串
proxy (common)	HTTP 代理服务器配置		HttpHost
serviceUri (common)	目标 OData 服务基础 URI，例如 http://services.odata.org/OData/OData.svc		字符串
socketTimeout (common)	HTTP 请求超时（以毫秒为单位），默认为 30,000 (30 秒)	30000	int
sslContextParameters (common)	使用 SSLContextParameters 配置安全性		SSLContextParameters
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

246.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 14 个选项，如下所列。

Name	描述	默认值	类型
camel.component.olingo2.configuration.api-name	要执行的操作类型		Olingo2ApiName
camel.component.olingo2.configuration.connect-timeout	HTTP 连接创建超时（以毫秒为单位），默认为 30,000 (30 秒)	30000	整数
camel.component.olingo2.configuration.content-type	content-Type 标头值可用于指定 JSON 或 XML 消息格式，默认为 application/json;charset=utf-8	application/json;charset=utf-8	字符串
camel.component.olingo2.configuration.http-async-client-builder	自定义 HTTP async 客户端构建器用于更复杂的 HTTP 客户端配置，覆盖 connectionTimeout, socketTimeout, proxy 和 sslContext。请注意，在构建器中指定一个 socketTimeout MUST，否则 OData 请求可能会无限期阻断		HttpAsyncClientBuilder
camel.component.olingo2.configuration.http-client-builder	自定义 HTTP 客户端构建器用于更复杂的 HTTP 客户端配置，覆盖 connectionTimeout, socketTimeout, proxy 和 sslContext。请注意，在构建器中指定一个 socketTimeout MUST，否则 OData 请求可能会无限期阻断		HttpClientBuilder
camel.component.olingo2.configuration.http-headers	自定义 HTTP 标头来注入每个请求，这可能包括 OAuth 令牌等。		Map
camel.component.olingo2.configuration.method-name	用于所选操作的子操作		字符串
camel.component.olingo2.configuration.proxy	HTTP 代理服务器配置		HttpHost
camel.component.olingo2.configuration.service-uri	目标 OData 服务基础 URI，例如 http://services.odata.org/OData/OData.svc		字符串
camel.component.olingo2.configuration.socket-timeout	HTTP 请求超时（以毫秒为单位），默认为 30,000 (30 秒)	30000	整数

Name	描述	默认值	类型
camel.component.olingo2.configuration.ssl-context-parameters	使用 SSLContextParameters 配置安全性		SSLContextParameters
camel.component.olingo2.enabled	启用 olingo2 组件	true	布尔值
camel.component.olingo2.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.olingo2.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

246.4. 制作者端点

生产者端点可以使用接下来列出的端点名称和选项。生成者端点也可以使用特殊选项 inBody，它应包含端点选项的名称，其值将包含在 Camel Exchange In 消息中。inBody 选项默认为使用该选项的端点的数据。

246.5. 端点选项

任何端点选项都可以在端点 URI 中提供，或者在消息标头中动态提供。消息标头名称必须是 CamelOlingo2.<option> 格式。请注意，inBody 选项会覆盖消息标头，即 Body=option 中的端点选项会覆盖 CamelOlingo2.option 标头。此外，还可指定查询参数。

Name	类型	描述
data	对象	具有适当类型的数据，用于创建或修改 OData 资源
keyPredicate	字符串	创建参数化 OData 资源端点的关键 predicate。对于在标头中动态提供 key predicate 值的 create/update 操作很有用
queryParams	java.util.Map<String,String>	OData 系统选项和自定义查询选项。如需更多信息，请参阅 OData 2.0 URI 约定

Name	类型	描述
resourcePath	字符串	OData 资源路径，可能包含密钥 predicate
endpointHttpHeaders	java.util.Map<String, String>	要发送到端点的动态 HTTP 标头
responseHttpHeaders	java.util.Map<String, String>	来自端点的动态 HTTP 响应标头

请注意，resourcePath 选项可以在 URI 中指定的作为 URI 路径的一部分，作为端点选项？resourcePath=<resource-path> 或作为标头值 CamelOlingo2.resourcePath。OData entity key predicate 可以是资源路径的一部分，如 Manufacturers ('1')，其中 '__1' 是 key predicate，也可以使用 resource path Manufacturers 和 keyPredicate 选项 '1' 单独指定。

端点	选项	HTTP 方法	结果正文类型
batch	data, endpointHttpHeaders	POST 带有 multipart/mixed batch 请求	java.util.List<org.apache.camel.component.olingo2.api.batch.Olingo2BatchResponse>
create	data, resourcePath, endpointHttpHeaders	POST	org.apache.olingo.odata2.api.ep.entry.ODataEntry 用于其他 OData 资源 org.apache.olingo.odata2.api.commons.HttpStatusCodes
delete	resourcePath, endpointHttpHeaders	DELETE	org.apache.olingo.odata2.api.commons.HttpStatusCodes

端点	选项	HTTP 方法	结果正文类型
merge	data, resourcePath, endpointHttpHeaders	合并	org.apache.olingo.odata2.api.commons.HttpStatusCodes
patch	data, resourcePath, endpointHttpHeaders	PATCH	org.apache.olingo.odata2.api.commons.HttpStatusCodes
读取	queryParams, resourcePath, endpointHttpHeaders	GET	依赖于正在查询的 OData 资源，如下一步所述
update	data, resourcePath, endpointHttpHeaders	PUT	org.apache.olingo.odata2.api.commons.HttpStatusCodes

246.6. 端点 HTTP 标头(SINCE 2.20)

组件级配置属性 httpHeaders 提供静态 HTTP 标头信息。但是，有些系统需要从端点传递和接收动态标头信息。示例用例是需要动态安全令牌的系统。endpointHttpHeaders 和 responseHttpHeaders 端点属性提供此功能。设置需要传递给 CamelOlingo2.endpointHttpHeaders 属性中的端点的标头，并在 CamelOlingo2.responseHttpHeaders 属性中返回响应标头。这两个属性都是类型 java.util.Map<String, String>。

246.7. ODATA 资源类型映射

读取端点和 数据类型 的结果取决于正在查询、创建或修改的 OData 资源。

OData 资源类型	来自 resourcePath 和 keyPredicate 的资源 URI	in 或 Out Body Type
实体数据模型	$metadata	org.apache.olingo.odata2.api.edm.Edm
服务文档	/	org.apache.olingo.odata2.api.servicedocument.ServiceDocument
OData feed	<entity-set>	org.apache.olingo.odata2.api.ep.feed.ODataFeed
OData 条目	<entity-set>(<key-predicate>)	org.apache.olingo.odata2.api.ep.entry.ODataEntry for Out body (response) java.util.Map<String, Object> for In body (request)
simple 属性	<entity-set>(<key-predicate>)/<simple-property>	适当的 Java 数据类型，如 Olingo EdmProperty 所述
简单属性值	<entity-set>(<key-predicate>)/<simple-property>/$value	适当的 Java 数据类型，如 Olingo EdmProperty 所述
复杂属性	<entity-set>(<key-predicate>)/<complex-property>	java.util.Map<String, Object>

OData 资源类型	来自 resourcePath 和 keyPredicate 的资源 URI	in 或 Out Body Type
零或一个关联链接	<entity-set>(<key-predicate>/$link/<one-to-one-entity-set-property>	响应 java.util.Map<String, Object> 的字符串，带有键属性名称和请求的值
零个或多个关联链接	<entity-set>(<key-predicate>/$link/<one-to-many-entity-set-property>	java.util.List<String> 用于响应 java.util.List<java.util.Map<String, Object>> 包含用于请求的关键属性名称和值列表
数量	<resource-uri>/$count	java.lang.Long

246.8. 消费者端点

只有 read 端点才能用作消费者端点。消费者端点可以使用 Scheduled Poll Consumer Options 和 consumer. 前缀来调度端点调用。默认情况下，返回数组或集合的消费者端点将为每个元素生成一个交换，它们的路由将为每个交换执行一次。可以通过设置 endpoint 属性 consumer.splitResult=false 来禁用此行为。

246.9. 消息标头

任何 URI 选项可以在带有 CamelOlingo2. 前缀的制作者端点的消息标头中提供。

246.10. 消息正文

所有结果消息正文都使用 Olingo2Component 使用的底层 Apache Olingo 2.0 API 提供的对象。生产者端点可以在 inBody 端点 URI 参数中为传入消息正文指定选项名称。对于返回数组或集合的端点，消费者端点会将每个元素映射到不同的消息，除非将 consumer.splitResult 设置为 false。

246.11. 使用案例

以下路由从 Manufacturer 源以升为 Name 属性的顺序读取前 5 个条目。

```
from("direct:...")
    .setHeader("CamelOlingo2.$top", "5");
    .to("olingo2://read/Manufacturers?orderBy=Name%20asc");
```

以下路由使用传入 id 标头中的 key 属性值来读取 Manufacturer 条目。

```
from("direct:...")
    .setHeader("CamelOlingo2.keyPredicate", header("id"))
    .to("olingo2://read/Manufacturers");
```

以下路由使用正文消息中的 java.util.Map<String, Object> 创建 Manufacturer 条目。

```
from("direct:...")
    .to("olingo2://create/Manufacturers");
```

以下路由每 30 秒轮询 Manufacturer delta feed。bean blah 更新 bean paramsBean，以添加更新的 !deltatoken 属性，并在 ODataDeltaFeed 结果中返回的值。因为初始 delta 令牌未知，因此消费者端点第一次会生成一个 ODataFeed 值，并在后续轮询时 ODataDeltaFeed 值。

```
from("olingo2://read/Manufacturers?
queryParams=#paramsBean&consumer.timeUnit=SECONDS&consumer.delay=30")
    .to("bean:blah");
```

第 247 章 OLINGO4 组件

从 Camel 版本 2.19 开始提供

Olingo4 组件利用 Apache Olingo 版本 4.0 API 与兼容 OData 4.0 的服务交互。因为版本 4.0 OData 是 OASIS 标准，因此流行开源和商业供应商和产品的数量支持这个协议。有关支持产品的示例列表可在 OData 网站 中找到。

Olingo4 组件支持读取实体集、实体、简单和复杂的属性、计数、使用自定义和 OData 系统查询参数。它支持更新实体和属性。它还支持以单一 OData batch 操作形式提交查询并更改请求。

组件支持为 OData 服务连接配置 HTTP 连接参数和标头。这允许根据目标 OData 服务的要求配置 SSL、OAuth2.0 等。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-olingo4</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

247.1. URI 格式

```
olingo4://endpoint/<resource-path>?[options]
```

247.2. OLINGO4 选项

Olingo4 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	使用共享配置		Olingo4Configuration
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Olingo4 端点使用 URI 语法进行配置：

```
olingo4:apiName/methodName
```

使用以下路径和查询参数：

247.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
apiName	**需要** 执行什么操作		Olingo4ApiName
methodName	**必需的** 所选操作使用哪些子操作		字符串

247.2.2. 查询参数(14 参数)：

Name	描述	默认值	类型
connectTimeout (common)	HTTP 连接创建超时（以毫秒为单位），默认为 30,000 (30 秒)	30000	int
contentType (common)	content-Type 标头值可用于指定 JSON 或 XML 消息格式，默认为 application/json;charset=utf-8	application/json;charset=utf-8	字符串
httpAsyncClientBuilder (common)	自定义 HTTP async 客户端构建器用于更复杂的 HTTP 客户端配置，覆盖 connectionTimeout, socketTimeout, proxy 和 sslContext。请注意，在构建器中指定一个 socketTimeout MUST，否则 OData 请求可能会无限期阻断		HttpAsyncClientBuilder

Name	描述	默认值	类型
httpClientBuilder (common)	自定义 HTTP 客户端构建器用于更复杂的 HTTP 客户端配置，覆盖 connectionTimeout, socketTimeout, proxy 和 sslContext。请注意，在构建器中指定一个 socketTimeout MUST，否则 OData 请求可能会无限期阻断		HttpClientBuilder
httpHeaders (common)	自定义 HTTP 标头来注入每个请求，这可能包括 OAuth 令牌等。		Map
inBody (common)	设置要在交换 In Body 中传递的参数名称		字符串
proxy (common)	HTTP 代理服务器配置		HttpHost
serviceUri (common)	目标 OData 服务基础 URI，例如 http://services.odata.org/OData/OData.svc		字符串
socketTimeout (common)	HTTP 请求超时（以毫秒为单位），默认为 30,000 (30 秒)	30000	int
sslContextParameters (common)	使用 SSLContextParameters 配置安全性		SSLContextParameters
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

247.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 14 个选项，如下所列。

Name	描述	默认值	类型
camel.component.olingo4.configuration.api-name	要执行的操作类型		Olingo4ApiName
camel.component.olingo4.configuration.connect-timeout	HTTP 连接创建超时（以毫秒为单位），默认为 30,000 (30 秒)	30000	整数
camel.component.olingo4.configuration.content-type	content-Type 标头值可用于指定 JSON 或 XML 消息格式，默认为 application/json;charset=utf-8	application/json;charset=utf-8	字符串
camel.component.olingo4.configuration.http-async-client-builder	自定义 HTTP async 客户端构建器用于更复杂的 HTTP 客户端配置，覆盖 connectionTimeout, socketTimeout, proxy 和 sslContext。请注意，在构建器中指定一个 socketTimeout MUST，否则 OData 请求可能会无限期阻断		HttpAsyncClientBuilder
camel.component.olingo4.configuration.http-client-builder	自定义 HTTP 客户端构建器用于更复杂的 HTTP 客户端配置，覆盖 connectionTimeout, socketTimeout, proxy 和 sslContext。请注意，在构建器中指定一个 socketTimeout MUST，否则 OData 请求可能会无限期阻断		HttpClientBuilder
camel.component.olingo4.configuration.http-headers	自定义 HTTP 标头来注入每个请求，这可能包括 OAuth 令牌等。		Map
camel.component.olingo4.configuration.method-name	用于所选操作的子操作		字符串
camel.component.olingo4.configuration.proxy	HTTP 代理服务器配置		HttpHost
camel.component.olingo4.configuration.service-uri	目标 OData 服务基础 URI，例如 http://services.odata.org/OData/OData.svc		字符串
camel.component.olingo4.configuration.socket-timeout	HTTP 请求超时（以毫秒为单位），默认为 30,000 (30 秒)	30000	整数

Name	描述	默认值	类型
camel.component.olingo4.configuration.ssl-context-parameters	使用 SSLContextParameters 配置安全性		SSLContextParameters
camel.component.olingo4.enabled	启用 olingo4 组件	true	布尔值
camel.component.olingo4.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.olingo4.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

247.4. 制作者端点

生产者端点可以使用接下来列出的端点名称和选项。生成者端点也可以使用特殊选项 inBody，它应包含端点选项的名称，其值将包含在 Camel Exchange In 消息中。inBody 选项默认为使用该选项的端点的数据。

任何端点选项都可以在端点 URI 中提供，或者在消息标头中动态提供。消息标头名称的格式必须是 CamelOlingo4.<option>。请注意，inBody 选项会覆盖消息标头，即 Body=option 中的端点选项会覆盖 CamelOlingo4.option 标头。另外，还可指定查询参数

请注意，resourcePath 选项可以在 URI 中指定的作为 URI 路径的一部分，作为端点选项 ? resourcePath=<resource-path> 或作为标头值 CamelOlingo4.resourcePath。OData entity key predicate 可以是资源路径的一部分，如 Manufacturers ('1')，其中 '__1' 是 key predicate，也可以使用 resource path Manufacturers 和 keyPredicate 选项 '1' 单独指定。

端点	选项	HTTP 方法	结果正文类型

端点	选项	HTTP 方法	结果正文类型
batch	data, endpointHttpHeaders	POST 带有 multipart/mixed batch 请求	java.util.List<org.apache.camel.component.olingo4.api.batch.Olingo4BatchResponse>
create	data, resourcePath, endpointHttpHeaders	POST	org.apache.olingo.client.api.domain.ClientEntity 用于其他 OData 资源 org.apache.olingo.commons.api.http.HttpStatusCode
delete	resourcePath, endpointHttpHeaders	DELETE	org.apache.olingo.commons.api.http.HttpStatusCode
merge	data, resourcePath, endpointHttpHeaders	合并	org.apache.olingo.commons.api.http.HttpStatusCode
patch	data, resourcePath, endpointHttpHeaders	PATCH	org.apache.olingo.commons.api.http.HttpStatusCode
读取	queryParams, resourcePath, endpointHttpHeaders	GET	依赖于正在查询的 OData 资源，如下一步所述

端点	选项	HTTP 方法	结果正文类型
update	data, resourcePath, endpointHttpHeaders	PUT	org.apache.olingo.commons.api.http.HttpStatusCode

247.5. 端点 HTTP 标头（自 CAMEL 2.20开始）

组件级配置属性 httpHeaders 提供静态 HTTP 标头信息。但是，有些系统需要从端点传递和接收动态标头信息。示例用例是需要动态安全令牌的系统。endpointHttpHeaders 和 responseHttpHeaders 端点属性提供此功能。设置需要传递给 CamelOlingo4.endpointHttpHeaders 属性中的端点的标头，并在 CamelOlingo4.responseHttpHeaders 属性中返回响应标头。这两个属性都是类型 java.util.Map<String, String>。

247.6. ODATA 资源类型映射

读取端点和 数据类型 的结果取决于正在查询、创建或修改的 OData 资源。

OData 资源类型	来自 resourcePath 和 keyPredicate 的资源 URI	in 或 Out Body Type
实体数据模型	$metadata	org.apache.olingo.commons.api.edm.Edm
服务文档	/	org.apache.olingo.client.api.domain.ClientServiceDocument
OData 实体集	<entity-set>	org.apache.olingo.client.api.domain.ClientEntitySet
OData 实体	<entity-set>(<key-predicate>)	org.apache.olingo.client.api.domain.ClientEntity for Out body (response) java.util.Map<String, Object> for In body (request)

OData 资源类型	来自 resourcePath 和 keyPredicate 的资源 URI	in 或 Out Body Type
simple 属性	<entity-set>(<key-predicate>)/<simple-property>	org.apache.olingo.client.api.domain.ClientPrimitiveValue
简单属性值	<entity-set>(<key-predicate>)/<simple-property>/$value	org.apache.olingo.client.api.domain.ClientPrimitiveValue
复杂属性	<entity-set>(<key-predicate>)/<complex-property>	org.apache.olingo.client.api.domain.ClientComplexValue
数量	<resource-uri>/$count	java.lang.Long

247.7. 消费者端点

只有 read 端点才能用作消费者端点。消费者端点可以使用 Scheduled Poll Consumer Options 和 consumer. 前缀来调度端点调用。默认情况下，返回数组或集合的消费者端点将为每个元素生成一个交换，它们的路由将为每个交换执行一次。可以通过设置 endpoint 属性 consumer.splitResult=false 来禁用此行为。

247.8. 消息标头

任何 URI 选项可以在带有 CamelOlingo4. 前缀的制作者端点的消息标头中提供。

247.9. 消息正文

所有结果消息正文都使用 Olingo4Component 使用的底层 Apache Olingo 4.0 API 提供的对象。生产者端点可以在 inBody 端点 URI 参数中为传入消息正文指定选项名称。对于返回数组或集合的端点，消费者端点会将每个元素映射到不同的消息，除非将 consumer.splitResult 设置为 false。

247.10. 使用案例

以下路由从由升序 FirstName 属性排序的 People 实体读取前 5 个条目。

```
from("direct:...")
    .setHeader("CamelOlingo4.$top", "5");
    .to("olingo4://read/People?orderBy=FirstName%20asc");
```

以下路由使用传入 id 标头中的 key 属性值读取 Airports 实体。

```
from("direct:...")
    .setHeader("CamelOlingo4.keyPredicate", header("id"))
    .to("olingo4://read/Airports");
```

以下路由使用正文消息中的 ClientEntity 创建 People 实体。

```
from("direct:...")
    .to("olingo4://create/People");
```

第 248 章 OPENSHIFT 组件（已弃用）

从 Camel 版本 2.14 开始提供

openshift 组件是用于管理 OpenShift 应用的组件。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-openshift</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

248.1. URI 格式

```
openshift:clientId[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

248.2. 选项

OpenShift 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
用户名 (security)	登录到 openshift 服务器的用户名。		字符串
密码 (security)	登录 openshift 服务器的密码。		字符串
domain (common)	域名.如果没有指定，则使用默认域。		字符串
server (common)	openshift 服务器的 URL。如果没有指定，则使用本地 openshift 配置文件 /.openshift/express.conf 中的默认值。如果此操作失败，则使用 openshift.redhat.com。		字符串

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

OpenShift 端点使用 URI 语法进行配置：

```
openshift:clientId
```

使用以下路径和查询参数：

248.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
clientId	**必需** 客户端 ID		字符串

248.2.2. 查询参数(26 参数)：

Name	描述	默认值	类型
domain (common)	域名.如果没有指定，则使用默认域。		字符串
password (common)	需要登录 openshift 服务器的密码。		字符串
server (common)	openshift 服务器的 URL。如果没有指定，则使用本地 openshift 配置文件 /.openshift/express.conf 中的默认值。如果此操作失败，则使用 openshift.redhat.com。		字符串
username (common)	**需要** 登录到 openshift 服务器的用户名。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值

Name	描述	默认值	类型
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
application (producer)	用于启动、停止、重启或获取状态的应用程序名称。		字符串
mode (producer)	是否将消息正文输出为 pojo 或 json。对于 pojo，消息是 List 类型。		字符串
operation (producer)	要执行的操作可以是： list、start、stop、restart 和 state。list 操作返回 json 格式的所有应用程序的信息。state 操作返回状态，如： started、stop 等。其他操作不会返回任何值。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long

Name	描述	默认值	类型
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

248.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
camel.component.openshift.domain	域名.如果没有指定，则使用默认域。		字符串
camel.component.openshift.enabled	启用 openshift 组件	true	布尔值

Name	描述	默认值	类型
camel.component.openshift.password	登录 openshift 服务器的密码。		字符串
camel.component.openshift.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.openshift.server	openshift 服务器的 URL。如果没有指定，则使用本地 openshift 配置文件 /.openshift/express.conf 中的默认值。如果此操作失败，则使用 openshift.redhat.com。		字符串
camel.component.openshift.username	登录到 openshift 服务器的用户名。		字符串

248.4. 例子

248.4.1. 列出所有应用程序

```
// sending route
from("direct:apps")
    .to("openshift:myClient?username=foo&password=secret&operation=list");
    .to("log:apps");
```

在这种情况下，所有应用程序的信息返回为 pojo。如果需要 json 响应，请设置 mode=json。

248.4.2. 停止应用程序

```
// stopping the foobar application
from("direct:control")
    .to("openshift:myClient?
username=foo&password=secret&operation=stop&application=foobar");
```

在上例中，我们停止名为 foobar 的应用程序。

轮询 gear 状态更改

consumer 用于轮询 gear 中的状态更改。例如，当添加/删除/或者其生命周期改变、正在启动或停止等时。

```
// trigger when state changes on our gears
from("openshift:myClient?username=foo&password=secret&delay=30s")
    .log("Event ${header.CamelOpenShiftEventType} on application ${body.name} changed
state to ${header.CamelOpenShiftEventNewState}");
```

当消费者发出交换时，正文包含 com.openshift.client.IApplication 作为消息正文。包括以下标头。

标头	可以是 null	描述
CamelOpenShiftEventType	否	事件的类型可以是：添加、删除或更改。
CamelOpenShiftEventOldState	是	当事件类型改变时，旧状态。
CamelOpenShiftEventNewState	否	任何事件类型的新状态

248.5. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 249 章 OPENSHIFT 构建配置组件

从 Camel 版本 2.17 开始提供

OpenShift Build Config 组件是 Kubernetes 组件 之一，它为执行 kubernetes 构建配置操作提供一个制作者。

249.1. 组件选项

Openshift Build Config 组件没有选项。

249.2. 端点选项

Openshift Build Config 端点使用 URI 语法进行配置：

```
openshift-build-configs:masterUrl
```

使用以下路径和查询参数：

249.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
masterUrl	**所需的** Kubernetes API 服务器 URL		字符串

249.2.2. 查询参数(20 参数)：

Name	描述	默认值	类型
apiVersion (producer)	要使用的 Kubernetes API 版本		字符串
dnsDomain (producer)	用于 ServiceCall EIP 的 dns 域		字符串
kubernetesClient (producer)	如果提供，则使用默认 KubernetesClient		KubernetesClient

Name	描述	默认值	类型
operation (producer)	在 Kubernetes 上执行的操作		字符串
PORTNAME (producer)	用于 ServiceCall EIP 的端口名称		字符串
portProtocol (producer)	端口协议，用于 ServiceCall EIP	tcp	字符串
connectionTimeout (advanced)	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
caCertData (security)	CA 认证数据		字符串
caCertFile (security)	CA 认证文件		字符串
clientCertData (security)	客户端证书数据		字符串
clientCertFile (security)	客户端认证文件		字符串
clientKeyAlgo (security)	客户端使用的密钥算法		字符串
ClientKeyData (security)	客户端密钥数据		字符串
clientKeyFile (security)	客户端密钥文件		字符串
clientKeyPassphrase (security)	客户端密钥密码		字符串
oauthToken (security)	Auth Token		字符串
密码 (security)	连接到 Kubernetes 的密码		字符串
trustCerts (security)	定义我们使用的证书是否被信任		布尔值

Name	描述	默认值	类型
用户名 (security)	连接到 Kubernetes 的用户名		字符串

第 250 章 OPENSHIFT 构建组件

从 Camel 版本 2.17 开始提供

Kubernetes 构建 组件是 Kubernetes 组件 之一，它为执行 kubernetes 构建操作提供一个制作者。

250.1. 组件选项

Openshift Builds 组件没有选项。

250.2. 端点选项

Openshift Builds 端点使用 URI 语法进行配置：

```
openshift-builds:masterUrl
```

使用以下路径和查询参数：

250.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
masterUrl	所需的 Kubernetes API 服务器 URL		字符串

250.2.2. 查询参数(20 参数)：

Name	描述	默认值	类型
apiVersion (producer)	要使用的 Kubernetes API 版本		字符串
dnsDomain (producer)	用于 ServiceCall EIP 的 dns 域		字符串
kubernetesClient (producer)	如果提供，则使用默认 KubernetesClient		KubernetesClient

Name	描述	默认值	类型
operation (producer)	在 Kubernetes 上执行的操作		字符串
PORTNAME (producer)	用于 ServiceCall EIP 的端口名称		字符串
portProtocol (producer)	端口协议，用于 ServiceCall EIP	tcp	字符串
connectionTimeout (advanced)	在向 Kubernetes API 服务器发出请求时使用的连接超时（以毫秒为单位）。		整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
caCertData (security)	CA 认证数据		字符串
caCertFile (security)	CA 认证文件		字符串
clientCertData (security)	客户端证书数据		字符串
clientCertFile (security)	客户端认证文件		字符串
clientKeyAlgo (security)	客户端使用的密钥算法		字符串
ClientKeyData (security)	客户端密钥数据		字符串
clientKeyFile (security)	客户端密钥文件		字符串
clientKeyPassphrase (security)	客户端密钥密码		字符串
oauthToken (security)	Auth Token		字符串
密码 (security)	连接到 Kubernetes 的密码		字符串
trustCerts (security)	定义我们使用的证书是否被信任		布尔值

Name	描述	默认值	类型
用户名 (security)	连接到 Kubernetes 的用户名		字符串

250.3. OPENSTACK 组件

从 Camel 2.19 开始提供

openstack 组件是用于管理您的 OpenStack 应用程序的组件。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-openstack</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

OpenStack 服务	Camel 组件	描述
OpenStack Cinder	openstack-cinder	用于维护 OpenStack Cinder 的组件。
OpenStack Glance	openstack-glance	用于维护 OpenStack glance 的组件。
OpenStack Keystone	openstack-keystone	维护 OpenStack Keystone 的组件。
OpenStack Neutron	openstack-neutron	用于维护 OpenStack neutron 的组件。
OpenStack Nova	openstack-nova	用于维护 OpenStack nova 的组件。
OpenStack Swift	openstack-swift	用于维护 OpenStack swift 的组件。

第 251 章 OPENSTACK CINDER 组件

从 Camel 版本 2.19 开始提供

openstack-cinder 组件允许消息发送到 OpenStack 块存储服务。

251.1. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-openstack</artifactId>
    <version>${camel-version}</version>
</dependency>
```

其中 ${camel-version} 必须替换为 Camel 的实际版本。

251.2. URI 格式

```
openstack-cinder://hosturl[?options]
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&…

251.3. URI 选项

OpenStack Cinder 组件没有选项。

OpenStack Cinder 端点使用 URI 语法进行配置：

```
openstack-cinder:host
```

使用以下路径和查询参数：

251.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
主机	**所需的** OpenStack 主机 url		字符串

251.3.2. 查询参数(9 参数)：

Name	描述	默认值	类型
apiVersion (producer)	OpenStack API 版本	V3	字符串
config (producer)	OpenStack 配置		config
domain (producer)	身份验证域	default	字符串
operation (producer)	要执行的操作		字符串
password (producer)	**所需的** OpenStack 密码		字符串
project (producer)	**必需的** 项目 ID		字符串
subsystem (producer)	**所需的** OpenStack Cinder 子系统		字符串
username (producer)	**所需的** OpenStack 用户名		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

251.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.openstack-cinder.enabled	启用 openstack-cinder 组件	true	布尔值
camel.component.openstack-cinder.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

251.5. 使用方法

您可以对每个子系统使用以下设置：

251.6. 卷

251.6.1. 您可以使用 Volume producer 执行的操作

操作	描述
create	创建新卷。
get	获取卷。
getAll	获取所有卷。
getAllTypes	获取卷类型。
update	更新卷。
delete	删除卷。

251.6.2. 由 Volume producer 评估的消息标头

标头	类型	描述
operation	**字符串**	要执行的操作。
ID	**字符串**	卷的 ID。

标头	类型	描述
name	**字符串**	卷名称。
description	**字符串**	卷描述.
size	**整数**	卷的大小。
volumeType	**字符串**	卷类型。
imageRef	**字符串**	镜像的 ID。
snapshotId	**字符串**	快照 ID。
isBootable	**布尔值**	可启动。

如果您需要更精确的卷设置，您可以创建类型为 org.openstack4j.model.storage.block.Volume 的新对象，并在消息正文中发送。

251.7. 快照

251.7.1. 您可以使用 Snapshot producer 执行的操作

操作	描述
create	创建新快照。
get	获取快照。
getAll	获取所有快照。
update	获取更新快照。
delete	删除快照。

251.7.2. Snapshot producer 评估的消息标头

标头	类型	描述
operation	**字符串**	要执行的操作。
ID	**字符串**	服务器的 ID。
name	**字符串**	服务器名称。
description	**字符串**	快照描述。
VolumeId	**字符串**	卷 ID。
force	**布尔值**	强制.

如果您需要更精确的服务器设置，您可以创建类型为 org.openstack4j.model.storage.block.VolumeSnapshot 的新对象，并在消息正文中发送。

251.8. 另请参阅

- ***配置 Camel***
- *组件*
- ***端点***
- ***开始使用***
- ***OpenStack 组件***

第 252 章 OPENSTACK GLANCE 组件

从 Camel 版本 2.19 开始提供

openstack-glance 组件允许消息发送到 OpenStack 镜像服务。

252.1. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-openstack</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 ${camel-version} 必须替换为 Camel 的实际版本。

252.2. URI 格式

```
openstack-glance://hosturl[?options]
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&…

252.3. URI 选项

OpenStack Glance 组件没有选项。

OpenStack Glance 端点使用 URI 语法进行配置：

```
openstack-glance:host
```

使用以下路径和查询参数：

252.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
主机	**所需的** OpenStack 主机 url		字符串

252.3.2. 查询参数(8 参数)：

Name	描述	默认值	类型
apiVersion (producer)	OpenStack API 版本	V3	字符串
config (producer)	OpenStack 配置		config
domain (producer)	身份验证域	default	字符串
operation (producer)	要执行的操作		字符串
password (producer)	**所需的** OpenStack 密码		字符串
project (producer)	**必需的** 项目 ID		字符串
username (producer)	**所需的** OpenStack 用户名		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

252.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.openstack-glance.enabled	启用 openstack-glance 组件	true	布尔值

Name	描述	默认值	类型
camel.component.openstack-glance.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

252.5. 使用方法

操作	描述
reserve	保留镜像。
create	创建新镜像。
update	更新镜像。
上传	上传镜像。
get	获取镜像。
getAll	获取所有镜像。
delete	删除镜像。

252.5.1. Glance producer 评估的消息标头

标头	类型	描述
operation	**字符串**	要执行的操作。
ID	**字符串**	类别的 ID。
name	**字符串**	类别名称。
diskFormat	**org.openstack4j.model.image.DiskFormat**	类别 VCPU 数量。

标头	类型	描述
containerFormat	**org.openstack4j.model.image.ContainerFormat**	RAM 大小。
owner	**字符串**	镜像所有者。
isPublic	**布尔值**	是公共的。
minRam	**Long**	最小 RAM。
minDisk	**Long**	最小磁盘。
size	**Long**	大小。
checksum	**字符串**	checksum。
属性	**Map**	镜像属性.

252.6. 另请参阅

- ***配置 Camel***
- ***组件***
- ***端点***
- ***开始使用***

- ***OpenStack 组件***

第 253 章 OPENSTACK KEYSTONE 组件

从 Camel 版本 2.19 开始提供

openstack-keystone 组件允许消息发送到 OpenStack 身份服务。

openstack-keystone 组件只支持 Identity API v3!

253.1. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-openstack</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 ${camel-version} 必须替换为 Camel 的实际版本。

253.2. URI 格式

```
openstack-keystone://hosturl[?options]
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&…

253.3. URI 选项

OpenStack Keystone 组件没有选项。

OpenStack Keystone 端点使用 URI 语法进行配置：

```
openstack-keystone:host
```

使用以下路径和查询参数：

253.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
主机	**所需的** OpenStack 主机 url		字符串

253.3.2. 查询参数(8 参数)：

Name	描述	默认值	类型
config (producer)	OpenStack 配置		config
domain (producer)	身份验证域	default	字符串
operation (producer)	要执行的操作		字符串
password (producer)	**所需的** OpenStack 密码		字符串
project (producer)	**必需的** 项目 ID		字符串
subsystem (producer)	**所需的** OpenStack Keystone 子系统		字符串
username (producer)	**所需的** OpenStack 用户名		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

253.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.openstack-keystone.enabled	启用 openstack-keystone 组件	true	布尔值
camel.component.openstack-keystone.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

253.5. 使用方法

您可以对每个子系统使用以下设置：

253.6. DOMAINS

253.6.1. 您可以使用 Domain producer 执行的操作

操作	描述
create	创建新域。
get	获取域。
getAll	获取所有域。
update	更新域。
delete	删除域。

253.6.2. 由 Domain producer 评估的消息标头

标头	类型	描述
operation	**字符串**	要执行的操作。
ID	**字符串**	域的 ID。
name	**字符串**	域名。

标头	类型	描述
description	**字符串**	域描述.

如果您需要更精确的域设置，您可以创建类型为 org.openstack4j.model.identity.v3.Domain 的新对象，并在消息正文中发送。

253.7. GROUPS

253.7.1. 您可以使用 Group producer 执行的操作

操作	描述
create	创建新组。
get	获取组。
getAll	获取所有组。
update	更新组。
delete	删除组。
addUserToGroup	将用户添加到组。
checkUserGroup	检查组中的用户是否是用户。
removeUserFromGroup	从组中删除该用户。

253.7.2. 由 Group producer 评估的消息标头

标头	类型	描述
operation	**字符串**	要执行的操作。
groupId	**字符串**	组的 ID。

标头	类型	描述
name	**字符串**	组名称。
userId	**字符串**	用户的 ID。
domainId	**字符串**	域的 ID。
description	**字符串**	组描述.

如果您需要更精确的组设置，您可以创建类型为 org.openstack4j.model.identity.v3.Group 的新对象，并在消息正文中发送。

253.8. PROJECTS

253.8.1. 您可以使用项目制作者执行的操作

操作	描述
create	创建新项目。
get	获取项目。
getAll	获取所有项目。
update	更新项目。
delete	删除项目。

253.8.2. 项目制作者评估的消息标头

标头	类型	描述
operation	**字符串**	要执行的操作。
ID	**字符串**	项目的 ID。
name	**字符串**	项目名称。

标头	类型	描述
description	**字符串**	项目描述.
domainId	**字符串**	域的 ID。
parentId	**字符串**	父项目 ID。

如果您需要更精确的项目设置，您可以创建类型为 org.openstack4j.model.identity.v3.Project 的新对象，并在消息正文中发送。

253.9. 区域

253.9.1. 您可以使用 Region producer 执行的操作

操作	描述
create	创建新区域。
get	获取区域。
getAll	获取所有区域。
update	更新区域。
delete	删除区域。

253.9.2. 由 Region producer 评估的消息标头

标头	类型	描述
operation	**字符串**	要执行的操作。
ID	**字符串**	区域的 ID。
description	**字符串**	区域描述.

如果您需要更精确的区域设置，您可以创建类型为 org.openstack4j.model.identity.v3.Region 的新

对象，并在消息正文中发送。

253.10. USERS

253.10.1. 您可以使用用户制作者执行的操作

操作	描述
create	创建新用户。
get	获取用户。
getAll	获取所有用户。
update	更新用户。
delete	删除用户。

253.10.2. 由 User producer 评估的消息标头

标头	类型	描述
operation	**字符串**	要执行的操作。
ID	**字符串**	用户的 ID。
name	**字符串**	用户名。
description	**字符串**	用户描述.
domainId	**字符串**	域的 ID。
password	**字符串**	用户的密码。
email	**字符串**	用户电子邮件.

如果您需要更精确的用户设置，您可以创建类型为 org.openstack4j.model.identity.v3.User 的新对象，并在消息正文中发送。

253.11. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- OpenStack 组件

第 254 章 OPENSTACK NEUTRON 组件

从 Camel 版本 2.19 开始提供

openstack-neutron 组件允许消息发送到 OpenStack 网络服务。

254.1. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-openstack</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 ${camel-version} 必须替换为 Camel 的实际版本。

254.2. URI 格式

```
openstack-neutron://hosturl[?options]
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&…

254.3. URI 选项

OpenStack Neutron 组件没有选项。

OpenStack Neutron 端点使用 URI 语法进行配置：

```
openstack-neutron:host
```

使用以下路径和查询参数：

254.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
主机	**所需的** OpenStack 主机 url		字符串

254.3.2. 查询参数(9 参数)：

Name	描述	默认值	类型
apiVersion (producer)	OpenStack API 版本	V3	字符串
config (producer)	OpenStack 配置		config
domain (producer)	身份验证域	default	字符串
operation (producer)	要执行的操作		字符串
password (producer)	**所需的** OpenStack 密码		字符串
project (producer)	**必需的** 项目 ID		字符串
subsystem (producer)	**所需的** OpenStack Neutron 子系统		字符串
username (producer)	**所需的** OpenStack 用户名		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

254.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.openstack-neutron.enabled	启用 openstack-neutron 组件	true	布尔值
camel.component.openstack-neutron.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

254.5. 使用方法

您可以对每个子系统使用以下设置：

254.6. NETWORKS

254.6.1. 您可以使用 Network producer 执行的操作

操作	描述
create	创建新网络。
get	获取网络。
getAll	获取所有网络。
delete	删除网络。

254.6.2. Network producer 评估的消息标头

标头	类型	描述
operation	**字符串**	要执行的操作。
ID	**字符串**	网络的 ID。
name	**字符串**	网络名称。
tenantId	**字符串**	租户 ID。

标头	类型	描述
adminStateUp	**布尔值**	AdminStateUp 标头。
networkType	**org.openstack4j.model.network.NetworkType**	网络类型。
physicalNetwork	**字符串**	物理网络.
segmentId	**字符串**	片段 ID。
isShared	**布尔值**	共享。
isRouterExternal	**布尔值**	是外部路由器。

如果您需要更精确的网络设置，您可以创建类型为 org.openstack4j.model.network.Network 的新对象，并在消息正文中发送。

254.7. SUBNETS

254.7.1. 您可以使用 Subnet producer 执行的操作

操作	描述
create	创建新子网。
get	获取子网。
getAll	获取所有子网。
delete	删除子网。

操作	描述
action	对子网执行操作。

254.7.2. 由 Subnet producer 评估的消息标头

标头	类型	描述
operation	**字符串**	要执行的操作。
ID	**字符串**	子网的 ID。
name	**字符串**	子网名称。
networkId	**字符串**	网络 ID。
enableDHCP	**布尔值**	启用 DHCP。
gateway	**字符串**	网关.

如果您需要更精确的子网设置，您可以创建类型为 org.openstack4j.model.network.Subnet 的新对象，并在消息正文中发送。

254.8. PORTS

254.8.1. 您可以使用 Port producer 执行的操作

操作	描述
create	创建新端口。
get	获取端口。
getAll	获取所有端口。
update	更新端口。
delete	删除端口。

254.8.2. 由 Port producer 评估的消息标头

标头	类型	描述
operation	**字符串**	要执行的操作。
name	**字符串**	端口名称。
networkId	**字符串**	网络 ID。
tenantId	**字符串**	租户 ID。
deviceId	**字符串**	设备 ID。
macAddress	**字符串**	MAC 地址。

254.9. 路由器

254.9.1. 您可以使用路由器制作者执行的操作

操作	描述
create	创建新路由器。
get	获取路由器。
getAll	获取所有路由器。
update	更新路由器。
delete	删除路由器。
attachInterface	连接接口。
detachInterface	分离接口。

254.9.2. 由 Port producer 评估的消息标头

标头	类型	描述
operation	**字符串**	要执行的操作。
name	**字符串**	路由器名称。
routerId	**字符串**	路由器 ID.
subnetId	**字符串**	子网 ID。
portId	**字符串**	端口 ID.
interfaceType	**org.openstack4j.model.network.AttachInterfaceType**	接口类型。
tenantId	**字符串**	租户 ID。

254.10. 另请参阅

- ***配置 Camel***
- ***组件***
- ***端点***
- ***开始使用***
- ***OpenStack 组件***

第 255 章 OPENSTACK NOVA 组件

从 Camel 版本 2.19 开始提供

openstack-nova 组件允许消息发送到 OpenStack 计算服务。

255.1. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-openstack</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 ${camel-version} 必须替换为 Camel 的实际版本。

255.2. URI 格式

```
openstack-nova://hosturl[?options]
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&…

255.3. URI 选项

OpenStack Nova 组件没有选项。

OpenStack Nova 端点使用 URI 语法进行配置：

```
openstack-nova:host
```

使用以下路径和查询参数：

255.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
主机	**所需的** OpenStack 主机 url		字符串

255.3.2. 查询参数(9 参数)：

Name	描述	默认值	类型
apiVersion (producer)	OpenStack API 版本	V3	字符串
config (producer)	OpenStack 配置		config
domain (producer)	身份验证域	default	字符串
operation (producer)	要执行的操作		字符串
password (producer)	**所需的** OpenStack 密码		字符串
project (producer)	**必需的** 项目 ID		字符串
subsystem (producer)	**所需的** OpenStack Nova 子系统		字符串
username (producer)	**所需的** OpenStack 用户名		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

255.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.openstack-nova.enabled	启用 openstack-nova 组件	true	布尔值
camel.component.openstack-nova.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

255.5. 使用方法

您可以对每个子系统使用以下设置：

255.6. FLAVORS

255.6.1. 您可以使用 Flavor producer 执行的操作

操作	描述
create	创建新类别.
get	获取类别。
getAll	获取所有类别。
delete	删除类别。

255.6.2. 由 Flavor producer 评估的消息标头

标头	类型	描述
operation	**字符串**	要执行的操作。
ID	**字符串**	类别的 ID。
name	**字符串**	类别名称。
VCPU	**整数**	类别 VCPU 数量。

标头	类型	描述
RAM	**整数**	RAM 大小。
disk	**整数**	磁盘大小。
swap	**整数**	swap 的大小。
rxtxFactor	**整数**	Rxtx Factor.

如果您需要更精确的类别设置，您可以创建类型为 org.openstack4j.model.compute.Flavor 的新对象，并在消息正文中发送。

255.7. 服务器

255.7.1. 您可以使用 Server producer 执行的操作

操作	描述
create	创建新服务器。
createSnapshot	创建服务器快照。
get	获取服务器。
getAll	获取所有服务器。
delete	删除服务器。
action	对服务器执行操作。

255.7.2. 由 Server producer 评估的消息标头

标头	类型	描述
operation	**字符串**	要执行的操作。
ID	**字符串**	服务器的 ID。
name	**字符串**	服务器名称。

标头	类型	描述
ImageId	**字符串**	镜像 ID。
FlavorId	**字符串**	要使用的类别 ID。
KeypairName	**字符串**	密钥对名称。
NetworkId	**字符串**	网络 ID。
AdminPassword	**字符串**	新服务器的管理员密码。
action	**org.openstack4j.model.compute.Action**	要执行的操作。

如果您需要更精确的服务器设置，您可以创建类型为 org.openstack4j.model.compute.ServerCreate 的新对象，并在消息正文中发送。

255.8. 密钥对

255.8.1. 您可以使用 Keypair producer 执行的操作

操作	描述
create	创建新密钥对。
get	获取密钥对。
getAll	获取所有密钥对。
delete	删除密钥对。

255.8.2. 由 Keypair producer 评估的消息标头

标头	类型	描述
operation	**字符串**	要执行的操作。
name	**字符串**	密钥对名称。

255.9. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*
- *OpenStack 组件*

第 256 章 OPENSTACK SWIFT 组件

从 Camel 版本 2.19 开始提供

openstack-swift 组件允许消息发送到 OpenStack 对象存储服务。

256.1. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-openstack</artifactId>
    <version>${camel-version}</version>
</dependency>
```

其中 ${camel-version} 必须替换为 Camel 的实际版本。

256.2. URI 格式

```
openstack-swift://hosturl[?options]
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&…

256.3. URI 选项

OpenStack Swift 组件没有选项。

OpenStack Swift 端点使用 URI 语法进行配置：

```
openstack-swift:host
```

使用以下路径和查询参数：

256.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
主机	**所需的** OpenStack 主机 url		字符串

256.3.2. 查询参数(9 参数)：

Name	描述	默认值	类型
apiVersion (producer)	OpenStack API 版本	V3	字符串
config (producer)	OpenStack 配置		config
domain (producer)	身份验证域	default	字符串
operation (producer)	要执行的操作		字符串
password (producer)	**所需的** OpenStack 密码		字符串
project (producer)	**必需的** 项目 ID		字符串
subsystem (producer)	**所需的** OpenStack Swift 子系统		字符串
username (producer)	**所需的** OpenStack 用户名		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

256.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.openstack-swift.enabled	启用 openstack-swift 组件	true	布尔值
camel.component.openstack-swift.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

256.5. 使用方法

您可以对每个子系统使用以下设置：

256.6. CONTAINERS

256.6.1. 您可以使用 Container producer 执行的操作

操作	描述
create	创建新容器。
get	获取容器。
getAll	获取所有容器。
update	更新容器。
delete	删除容器。
getMetadata	获取元数据。
createUpdateMetadata	创建/更新元数据。
deleteMetadata	删除元数据。

256.6.2. 由 Volume producer 评估的消息标头

标头	类型	描述
operation	**字符串**	要执行的操作。
name	**字符串**	容器名称。
X-Container-Meta-	**Map**	容器元数据前缀。
x-Versions-Location	**字符串**	版本位置.
x-Container-Read	**字符串**	acl - 容器读取。
X-Container-Write	**字符串**	acl - 容器写入。
limit	**整数**	列出选项 - 限制.
marker	**字符串**	列出选项 - 标记.
end_marker	**字符串**	列表选项 - 结束标记。
delimiter	**字符**	列出选项 - 分隔符.
path	**字符串**	列出选项 - 路径.

如果需要更精确的容器设置，您可以创建类型为 org.openstack4j.model.storage.object.options.CreateUpdateContainerOptions （在创建或更新操作的情况下）或 org.openstack4j.model.storage.object.options.ContainerListOptions 类型的新对象，以列出容器并在消息正文中发送。

256.7. 对象(OBJECT)

256.7.1. 您可以使用 Object producer 执行的操作

操作	描述
create	创建新对象。
get	获取对象。
getAll	获取所有对象。

操作	描述
update	获取更新对象。
delete	删除对象。
getMetadata	获取元数据。
createUpdateMetadata	创建/更新元数据。

256.7.2. Object producer 评估的消息标头

标头	类型	描述
operation	**字符串**	要执行的操作。
containerName	**字符串**	容器名称。
objectName	**字符串**	对象名称。

256.8. 另请参阅

- ***配置 Camel***
- ***组件***
- ***端点***
- ***开始使用***
- ***OpenStack 组件***

第 257 章 OPENTRACING 组件

从 Camel 2.19 开始提供

重要

从 Camel 2.21 开始，需要使用与 OpenTracing Java API 版本 0.31 或更高版本兼容的 OpenTracing complaint tracer。

camel-opentracing 组件用于追踪和计时使用 OpenTracing 的传入和传出 Camel 消息。

为要发送到/来自 Camel 的传入和传出消息捕获事件(spans)。

如需支持的 tracer 列表，请参阅 OpenTracing 网站。

257.1. 配置

OpenTracing tracer 的配置属性有：

选项	默认	描述
excludePatterns		设置将针对与模式匹配的 Camel 消息禁用追踪的排除模式。内容是一个 Set<String>，其中键是一个模式。该模式使用来自 Intercept 的规则。
编码	false	设置标头密钥是否需要编码（特定于connector）。该值是一个布尔值。短划线需要针对 JMS 属性密钥编码实例。

可以通过三种方式配置 OpenTracing tracer 来为 Camel 应用程序提供分布式追踪：

257.1.1. explicit

在您的 POM 中包含 camel-opentracing 组件，以及与所选 OpenTracing 兼容 Tracer 关联的任何特定依赖项。

若要显式配置 OpenTracing 支持，请实例化 OpenTracingTracer 并初始化 camel 上下文。您可以选择指定 Tracer，也可以使用 Registry 或 ServiceLoader 隐式发现。

```
OpenTracingTracer ottracer = new OpenTracingTracer();
// By default it uses a Noop Tracer, but you can override it with a specific OpenTracing implementation.
ottracer.setTracer(...);
// And then initialize the context
ottracer.init(camelContext);
```

要在 XML 中使用 OpenTracingTracer，您需要做的就是定义 OpenTracing tracer Bean。Camel 将自动发现并使用它们。

```
<bean id="tracer" class="..."/>
<bean id="ottracer" class="org.apache.camel.opentracing.OpenTracingTracer">
  <property name="tracer" ref="tracer"/>
</bean>
```

257.2. SPRING BOOT

如果您使用 Spring Boot，您可以添加 camel-opentracing-starter 依赖项，并通过使用 @CamelOpenTracing 注解主类来打开 OpenTracing。

Tracer 将隐式从 camel 上下文的 Registry 或 ServiceLoader 获取，除非应用程序定义了 Tracer bean。

257.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.opentracing.encoding	在标头中激活或停用短划线编码（由 JMS 需要）用于消息传递		布尔值
camel.opentracing.exclude-patterns	设置将针对与模式匹配的 Camel 消息禁用追踪的排除模式。		Set

257.4. JAVA 代理

第三种方法是使用 Java 代理来自动配置 OpenTracing 支持。

在您的 POM 中包含 camel-opentracing 组件，以及与所选 OpenTracing 兼容 Tracer 关联的任何特定依赖项。

OpenTracing Java 代理与以下依赖项关联：

```
<dependency>
  <groupId>io.opentracing.contrib</groupId>
  <artifactId>opentracing-agent</artifactId>
</dependency>
```

使用的 Tracer 将隐式从 camel 上下文 Registry 加载，或使用 ServiceLoader。

使用此代理的方式将特定于您如何执行应用程序。camel-example-opentracing 中的 Service2 将代理下载到本地文件夹中，然后使用 exec-maven-plugin 启动具有 -javaagent 命令行选项的服务。

257.5. EXAMPLE

您可以在此处找到一个演示三种配置 OpenTracing 的方法的示例： camel-example-opentracing

第 258 章 OPTAPLANNER 组件

从 Camel 版本 2.13 开始提供

optaplanner: 组件解决了带有 OptaPlanner 消息中包含的规划问题。
例如：为它提供未解决的 Vehicle Routing 问题，它解决了它。

组件支持 consumer 作为 BestSolutionChangedEvent 侦听器和制作者来处理 Solution 和 ProblemFactChange

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-optaplanner</artifactId>
  <version>x.x.x</version><!-- use the same version as your Camel core version -->
</dependency>
```

258.1. URI 格式

```
optaplanner:solverConfig[?options]
```

solverConfig 是 solverConfig 的 classpath-local URI，如 /org/foo/barSolverConfig.xml。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

258.2. OPTAPLANNER 选项

OptaPlanner 组件没有选项。

OptaPlanner 端点使用 URI 语法进行配置：

```
optaplanner:configFile
```

使用以下路径和查询参数：

258.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
configFile	***必需的*** 指定到 solver 文件的位置		字符串

258.2.2. 查询参数(7 参数)：

Name	描述	默认值	类型
solverId (common)	为 solver 实例密钥指定用户 solverId	DEFAULT_SOLVER	字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
async (producer)	指定在 async 模式下执行操作	false	布尔值
threadPoolSize (producer)	指定在 async 为 true 时使用的线程池大小	10	int
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

258.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.optaplanner.enabled	启用 optaplanner 组件	true	布尔值
camel.component.optaplanner.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

258.4. 消息标头

Name	默认值	类型	Context	描述
CamelOptaPlannerSolverId	null	字符串	共享	指定要使用的 solverId
CamelOptaPlannerIsAsync	PUT	字符串	制作者	指定是否使用另一个线程提交 Solution 实例，而不是阻塞当前的线程。

258.5. 消息正文

Camel 对 IN 正文进行规划问题，解决并在 OUT 正文上返回。（自 v 2.16），IN 正文对象支持以下用例：

- ***如果正文是 Solution 实例，则会使用 solverId 标识的解决者以及同步或异步或异步进行解决。***
- ***如果正文是 ProblemFactChange 的实例，它将触发 addProblemFactChange。如果处理异步，则在返回结果前会等待 till isEveryProblemFactChangeProcessed。***
- ***如果正文不是上述类型，则生成者将从 solverId 识别的 solver 返回最佳结果***

258.6. TERMINATION

只要在 solverConfig 中指定，其处理就会需要。

```
<solver>
  ...
  <termination>
    <!-- Terminate after 10 seconds, unless it's not feasible by then yet -->
    <terminationCompositionStyle>AND</terminationCompositionStyle>
    <secondsSpentLimit>10</secondsSpentLimit>
    <bestScoreLimit>-1hard/0soft</bestScoreLimit>
  </termination>
  ...
<solver>
```

258.6.1. Samples

使用 OptaPlanner 解决 ActiveMQ 队列上的规划问题：

```
from("activemq:My.Queue").
  .to("optaplanner:/org/foo/barSolverConfig.xml");
```

将 OptaPlanner 作为 REST 服务公开：

```
from("cxfrs:bean:rsServer?bindingStyle=SimpleConsumer")
  .to("optaplanner:/org/foo/barSolverConfig.xml");
```

258.7. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 259 章 PAHO 组件

从 Camel 版本 2.16 开始提供

paho 组件使用 Eclipse Paho 库为 MQTT 消息传递协议提供连接器。paho 是最流行的 MQTT 库之一，因此，如果您要将其与您的 Java 项目集成 - Camel Paho 连接器是采用的方法。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-paho</artifactId>
  <version>x.y.z</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

请记住，Paho 工件没有托管在 Maven Central 中，因此您需要将 Eclipse Paho 存储库添加到 POM xml 文件中：

```
<repositories>
 <repository>
  <id>eclipse-paho</id>
  <url>https://repo.eclipse.org/content/repositories/paho-releases</url>
  <snapshots>
   <enabled>false</enabled>
  </snapshots>
 </repository>
</repositories>
```

259.1. URI 格式

```
paho:topic[?options]
```

其中 topic 是主题的名称。

259.2. 选项

Paho 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
brokerUrl (common)	MQTT 代理的 URL。		字符串
clientId (common)	MQTT 客户端标识符。		字符串
connectOptions (advanced)	客户端连接选项		MqttConnectOptions
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Paho 端点使用 URI 语法进行配置：

```
paho:topic
```

使用以下路径和查询参数：

259.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
topic	主题的 **必需** 名称		字符串

259.2.2. 查询参数(15 参数)：

Name	描述	默认值	类型
autoReconnect (common)	如果连接丢失，客户端将自动尝试重新连接到服务器	true	布尔值
brokerUrl (common)	MQTT 代理的 URL。	tcp://localhost:1883	字符串
clientId (common)	MQTT 客户端标识符。		字符串
connectOptions (common)	客户端连接选项		MqttConnectOptions

Name	描述	默认值	类型
filePersistenceDirectory (common)	文件持久性提供程序使用的基础目录。		字符串
password (common)	用于对 MQTT 代理进行身份验证的密码		字符串
Persistence (common)	要使用的客户端持久性 - 内存或文件。	内存	PahoPersistence
QoS (common)	客户端服务质量级别(0-2)。	2	int
resolveMqttConnectOptions (common)	定义是否不想从 registry 解析 MQTT Connect 选项	true	布尔值
reserved (common)	retain 选项	false	布尔值
username (common)	用于对 MQTT 代理进行身份验证的用户名		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

259.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.component.paho.broker-url	MQTT 代理的 URL。		字符串
camel.component.paho.client-id	MQTT 客户端标识符。		字符串
camel.component.paho.connect-options	客户端连接选项。选项是一个 org.eclipse.paho.client.mqttv3.MqttConnectOptions 类型。		字符串
camel.component.paho.enabled	启用 paho 组件	true	布尔值
camel.component.paho.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

259.4. HEADERS

以下标头可以被 Paho 组件识别：

标头	Java 常数	端点类型	值类型	描述
CamelMqttTopic	PahoConstants.MQTT_TOPIC	消费者	字符串	主题的名称
CamelPahoOverrideTopic	PahoConstants.CAMEL_PAHO_OVERRIDE_TOPIC	制作者	字符串	覆盖和发送到的主题名称，而不是在端点中指定的主题

259.5. 默认有效负载类型

默认情况下，Camel Paho 组件在提取出出的二进制有效负载上运行（或放入 MQTT 消息）：

-

```
// Receive payload
byte[] payload = (byte[]) consumerTemplate.receiveBody("paho:topic");

// Send payload
byte[] payload = "message".getBytes();
producerTemplate.sendBody("paho:topic", payload);
```

但是，Camel build-in 类型转换 API 可以为您执行自动数据类型转换。在以下示例中，Camel 会自动将二进制有效负载转换为 String （反之亦然）：

```
// Receive payload
String payload = consumerTemplate.receiveBody("paho:topic", String.class);

// Send payload
String payload = "message";
producerTemplate.sendBody("paho:topic", payload);
```

259.6. SAMPLES

例如，以下片段从与 Camel 路由器相同的主机上安装的 MQTT 代理读取信息：

```
from("paho:some/queue")
    .to("mock:test");
```

以下片段将消息发送到 MQTT 代理：

```
from("direct:test")
    .to("paho:some/target/queue");
```

例如，这是如何从远程 MQTT 代理读取信息：

```
from("paho:some/queue?brokerUrl=tcp://iot.eclipse.org:1883")
    .to("mock:test");
```

在这里，我们覆盖默认主题，并设置为动态主题

```
from("direct:test")
    .setHeader(PahoConstants.CAMEL_PAHO_OVERRIDE_TOPIC,
simple("${header.customerId}"))
    .to("paho:some/target/queue");
```

第 260 章 OSGI PAX LOGGING 组件

从 Camel 版本 2.6 开始提供

paxlogging 组件可以在 OSGi 环境中使用，以接收 PaxLogging 事件并处理它们。

260.1. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml中

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-paxlogging</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 ${camel-version} 必须替换为 Camel 的实际版本(2.6.0 或更高版本)。

260.2. URI 格式

```
paxlogging:appender[?options]
```

其中 appender 是需要在 PaxLogging 服务配置中配置的 pax appender 的名称。

260.3. URI 选项

OSGi PAX Logging 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
bundleContext (consumer)	OSGi BundleContext 由 Camel 自动注入		BundleContext
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

OSGi PAX Logging 端点使用 URI 语法进行配置：

```
paxlogging:appender
```

使用以下路径和查询参数：

260.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
appender	**必需的** Appender 是需要在 PaxLogging 服务配置中配置的 pax appender 的名称。		字符串

260.3.2. 查询参数(4 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

260.4. 消息正文

消息正文 中的 将设置为接收的 PaxLoggingEvent。

260.5. 用法示例

```
<route>
  <from uri="paxlogging:camel"/>
  <to uri="stream:out"/>
</route>
```

配置：

```
log4j.rootLogger=INFO, out, osgi:VmLogAppender, osgi:camel
```

第 261 章 PDF COMPONENT

从 Camel 版本 2.16 开始提供

PDF ：组件提供从 PDF 文档创建、修改或提取内容的功能。此组件使用 Apache PDFBox 作为底层库来使用 PDF 文档。

要使用 PDF 组件，Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-pdf</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

261.1. URI 格式

PDF 组件仅支持生成者端点。

```
pdf:operation[?options]
```

261.2. 选项

PDF 组件没有选项。

PDF 端点使用 URI 语法进行配置：

```
pdf:operation
```

使用以下路径和查询参数：

261.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
operation	**所需的操作** 类型		PdfOperation

261.2.2. 查询参数(9 参数)：

Name	描述	默认值	类型
font (producer)	字体	Helvetica	PDFont
fontSize (producer)	pixels 中的字体大小	14	浮点值
marginBottom (producer)	边缘底部点数	20	int
marginLeft (producer)	边缘（以像素）	20	int
marginRight (producer)	边缘（像素）	40	int
marginTop (producer)	边缘 top in pixels	20	int
pageSize (producer)	页面大小	A4	PDRectangle
textProcessingFactory (producer)	要使用的文本处理。自动格式：文本按词语进行分片，然后每行中适合的最大词语数将写入到 pdf 文档中。通过此策略，行中不适合的所有词语都将移至新行。lineTermination：为 line-termination 编写策略构建一组类。文本按行终止符号进行分片，然后写入，无论它适合行。	lineTermination	TextProcessingFactory
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

261.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.pdf.enabled	启用 pdf 组件	true	布尔值
camel.component.pdf.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

261.4. HEADERS

标头	描述
pdf-document	**附加** 操作 **必需** 标头，并在所有其他操作中忽略。预期类型是 PDDocument。存储用于附加操作的 PDF 文档。
protection-policy	预期类型为 https://pdfbox.apache.org/docs/1.8.10/javadocs/org/apache/pdfbox/pdmodel/encryption/ProtectionPolicy.html[ProtectionPolicy]。如果指定，则将用它加密 PDF 文档。
解密	预期类型为 https://pdfbox.apache.org/docs/1.8.10/javadocs/org/apache/pdfbox/pdmodel/encryption/DecryptionMaterial.html[DecryptionMaterial]。如果 PDF 文档被加密，则 **强制** 标头。

261.5. 另请参阅

- ***配置 Camel***
- ***组件***
- ***端点***
- ***开始使用***

-

-

第 262 章 POSTGRESSQL 事件组件

从 Camel 版本 2.15 开始提供

这是 Apache Camel 的一个组件，允许从 PostgreSQL 8.3 开始添加的与 LISTEN/NOTIFY 命令相关的 Producing/Consuming PostgreSQL 事件。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-pgevent</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

URI 格式

pgevent 组件使用以下两种端点 URI 表示法样式：

```
pgevent:datasource[?parameters]
pgevent://host:port/database/channel[?parameters]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

262.1. 选项

PostgresSQL 事件组件没有选项。

PostgresSQL 事件端点使用 URI 语法进行配置：

```
pgevent:host:port/database/channel
```

使用以下路径和查询参数：

262.1.1. 路径参数(4 参数)：

Name	描述	默认值	类型
主机	使用主机名和端口连接到数据库。	localhost	字符串
port	使用主机名和端口连接到数据库。	5432	整数
database	**所需的** 数据库名称		字符串
channel	**所需的** 频道名称		字符串

262.1.2. 查询参数(7 参数)：

Name	描述	默认值	类型
DataSource (common)	使用给定的 javax.sql.DataSource 进行连接，而不使用主机名和端口。		DataSource
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
pass (security)	用于登录的密码		字符串
用户 （安全）	登录的用户名	postgres	字符串

262.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.pgevent.enabled	启用 pgevent 组件	true	布尔值
camel.component.pgevent.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

262.3. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 263 章 PGP DATAFORMAT

从 Camel 版本 2.9 开始提供

PGP 数据格式将 Java Cryptographic 扩展集成到 Camel 中，允许使用 Camel 熟悉的 marshall 和 unmarshal 格式化机制来简单而灵活的加密和解密消息。它假定要对 cyphertext 和 unmarshalling 进行加密，以意味着解密回原始的纯文本。此数据格式仅实施对称（共享密钥）加密和去除。

263.1. PGPDATAFORMAT OPTIONS

PGP dataformat 支持 15 个选项，如下所列。

Name	默认值	Java 类型	描述
keyUserid		**字符串**	加密期间使用的 PGP 密钥环中的密钥用户 ID。也可以是用户 ID 的一部分。例如，如果用户 ID 是 Test User，您可以使用 part Test User 或处理用户 ID。
signatureKeyUserid		**字符串**	PGP 密钥环中的密钥 ID，用于签名（加密）或签名验证（解密）。在签名验证过程中，指定的用户 ID 会限制可用于验证的公钥。如果没有为签名指定用户 ID，则使用公共密钥环中的任何公钥进行验证。也可以是用户 ID 的一部分。例如，如果用户 ID 是 Test User，您可以使用 part Test User 或处理用户 ID。
password		**字符串**	打开私钥时使用的密码（未用于加密）。
signaturePassword		**字符串**	打开用于签名的私钥时使用的密码（加密）。
keyFileName		**字符串**	密钥环的文件名；必须作为类路径资源访问（但您可以使用 file: 前缀指定文件系统中的位置）。
signatureKeyFileName		**字符串**	用于签名（加密加密）或签名验证（解密）的密钥环文件名；必须作为类路径资源访问（但您可以使用 file: prefix 指定文件系统中的位置）。
signatureKeyRing		**字符串**	用于签名/验证为字节阵列的密钥环。您不能同时设置 signatureKeyFileName 和 signatureKeyRing。
armored	**false**	**布尔值**	此选项将使 PGP 对加密文本进行 base64 编码，使它可用于复制/粘贴等。
完整性	**true**	**布尔值**	在加密文件中添加完整性检查/签名。默认值为 true。

Name	默认值	Java 类型	描述
provider		字符串	Java Cryptography Extension (JCE)供应商，默认为 Bouncy Castle (BC)。或者，您可以使用 IAIK JCE 供应商；在这种情况下，必须事先注册供应商，而且不能事先注册 Bouncy Castle 供应商。Sun JCE 供应商无法正常工作。
algorithm		整数	对称密钥加密算法；可能的值在 org.bouncycastle.bcpg.SymmetricKeyAlgorithmTags 中定义；例如 2 (= TRIPLE DES), 3 (= CAST5), 4 (= BLOWFISH), 6 (= DES), 7 (= AES_128)。仅适用于加密。
compressionAlgorithm		整数	压缩算法；可能的值在 org.bouncycastle.bcpg.CompressionAlgorithmTags 中定义；例如 0 (= UNCOMPRESSED), 1 (= ZIP), 2 (= ZLIB), 3 (= BZIP2)。仅适用于加密。
hashAlgorithm		整数	签名哈希算法；可能的值在 org.bouncycastle.bcpg.HashAlgorithmTags 中定义；例如 2 (= SHA1), 8 (= SHA256), 9 (= SHA384), 10 (= SHA512), 11 (=SHA224)。仅与签名相关。
signatureVerificationOption		字符串	控制在 unmarshaling 期间验证签名的行为。可能有 4 个值：可选：PGP 消息可能包含签名；如果包含签名，则执行签名验证。需要：PGP 消息必须至少包含一个签名；如果这不是异常(PGPException)的情况。执行签名验证。忽略：PGP 消息中的包含签名被忽略；没有执行签名验证。no_signature_allowed: PGP 消息不得包含签名；否则抛出异常(PGPException)。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

263.2. PGPDATAFORMAT MESSAGE HEADERS

您可以通过将以下标头应用到消息来覆盖 PGPDataFormat 选项。

Name	类型	描述
CamelPGPDataFormatKeyFileName	字符串	自 Camel 2.11.0 ；密钥环文件名以来，将直接覆盖 PGPDataFormat 上的现有设置。

Name	类型	描述
CamelPGPDataFormatEncryptionKeyRing	**byte[]**	**自 Camel 2.12.1 以来，**加密密钥环； 将直接覆盖 PGPDataFormat 上的现有设置。
CamelPGPDataFormatKeyUserid	**字符串**	**自 Camel 2.11.0 起，**PGP 密钥环中的密钥的用户 ID； 将直接覆盖 PGPDataFormat 上的现有设置。
CamelPGPDataFormatKeyUserids	**List<String>**	从 **camel 2.12.2 开始** ：PGP 密钥环中的密钥的用户 ID ； 将直接覆盖 PGPDataFormat 上的现有设置。
CamelPGPDataFormatKeyPassword	**字符串**	**自 Camel 2.11.0 以来，**打开私钥时使用的密码；将直接覆盖 PGPDataFormat 上的现有设置。
CamelPGPDataFormatSignatureKeyFileName	**字符串**	**自 Camel 2.11.0** ；签名密钥环的文件名； 将直接覆盖 PGPDataFormat 上的现有设置。
CamelPGPDataFormatSignatureKeyRing	**byte[]**	**自 Camel 2.12.1 以来，**签名密钥环； 将直接覆盖 PGPDataFormat 上的现有设置。

Name	类型	描述
CamelPGPDataFormatSignatureKeyUserid	**字符串**	**自 Camel 2.11.0 起，**PGP 密钥环中的签名密钥的用户 ID ； 将直接覆盖 PGPDataFormat 上的现有设置。
CamelPGPDataFormatSignatureKeyUserids	**List<String>**	从 **Camel 2.12.3 开始，**PGP 密钥环中的签名密钥的用户 ID ； 将直接覆盖 PGPDataFormat 上的现有设置。
CamelPGPDataFormatSignatureKeyPassword	**字符串**	**自 Camel 2.11.0 以来，**打开签名私钥时使用的密码； 将直接覆盖 PGPDataFormat 上的现有设置。
CamelPGPDataFormatEncryptionAlgorithm	**int**	**由于 Camel 2.12.2 ；**对称密钥加密算法； 将直接覆盖 PGPDataFormat 上的现有设置。
CamelPGPDataFormatSignatureHashAlgorithm	**int**	**由于 Camel 2.12.2;** 签名哈希算法； 将直接覆盖 PGPDataFormat 上的现有设置。
CamelPGPDataFormatCompressionAlgorithm	**int**	**由于 Camel 2.12.2;** 压缩算法； 将直接覆盖 PGPDataFormat 上的现有设置。

Name	类型	描述
CamelPGPDataFormatNumberOfEncryptionKeys	**整数**	从 *Camel 2.12.3 **开始**; 用于加密 symmectric 密钥的公钥数，在加密过程中由 PGPDataFormat 设置
CamelPGPDataFormatNumberOfSigningKeys	**整数**	从 *Camel 2.12.3 **开始**; 用于创建签名的私钥数，在签名过程中由 PGPDataFormat 设置

263.3. 使用 PGPDATAFORMAT 加密

以下示例使用 popular PGP 格式通过 Bouncy Castle Java 库 加密/解密文件：

以下示例执行签名 + 加密，然后签名验证 + 解密。它对签名和加密使用相同的密钥环，但您可以明显使用不同的密钥：

或使用 Spring:

263.3.1. 要使用前面的示例，您需要以下内容

- *包含用于加密数据的公钥的公共密钥环文件*
- *私有密钥环文件，其中包含用于解密数据的密钥*
- *密钥环密码*

263.3.2. 管理密钥环

若要管理密钥环，我使用命令行工具，我发现这是管理密钥的最简单方法。如果您希望通过这个方法进行此操作，还提供 http://www.bouncycastle.org/java.html 中的 Java 库。

在 linux 中安装命令行工具

```
apt-get install gnupg
```

创建密钥环，输入安全密码

```
gpg --gen-key
```

如果您需要导入其他人的公钥，以便您可以为其加密文件。

```
gpg --import <filename.key
```

以下文件现在应存在，可用于运行示例

```
ls -l ~/.gnupg/pubring.gpg ~/.gnupg/secring.gpg
```

[[crypto-PGPDecrypting/VerifyingofMessagesEncrypted/SignedbyDifferentPrivate/PublicKeys] PGP Decrypting/Verifying of Messages Encrypted/Signed by different # Private/Public Keys

从 Camel 2.12.2 开始。

PGP Data Formater 可以解密/验证由不同公钥加密或由不同私钥签名的消息。只需在 secret keyring 中提供对应的私钥、公共密钥环中的对应公钥，以及密码短语访问器中的密码短语。

```
Map<String, String> userId2Passphrase = new HashMap<String, String>(2);
// add passphrases of several private keys whose corresponding public keys have been used
to encrypt the messages
userId2Passphrase.put("UserIdOfKey1","passphrase1"); // you must specify the exact User
ID!
userId2Passphrase.put("UserIdOfKey2","passphrase2");
PGPPassphraseAccessor passphraseAccessor = new
PGPPassphraseAccessorDefault(userId2Passphrase);

PGPDataFormat pgpVerifyAndDecrypt = new PGPDataFormat();
pgpVerifyAndDecrypt.setPassphraseAccessor(passphraseAccessor);
// the method getSecKeyRing() provides the secret keyring as byte array containing the
```

```
private keys
pgpVerifyAndDecrypt.setEncryptionKeyRing(getSecKeyRing()); // alternatively you can use
setKeyFileName(keyfileName)
// the method getPublicKeyRing() provides the public keyring as byte array containing the
public keys
pgpVerifyAndDecrypt.setSignatureKeyRing((getPublicKeyRing()); // alternatively you can use
setSignatureKeyFileName(signatgureKeyfileName)
// it is not necessary to specify the encryption or signer  User Id

from("direct:start")
     ...
    .unmarshal(pgpVerifyAndDecrypt) // can decrypt/verify messages encrypted/signed by
different private/public keys
    ...
```

- 功能对于支持密钥交换特别有用。如果要交换私钥以进行解密，您可以接受点以旧或新的公钥加密的时间消息。或者，如果发送者希望交换签名人私钥，您可在一段时间内接受旧签名者密钥。

- 技术背景：PGP 加密的数据包含用于加密数据的公钥的密钥 ID。此密钥 ID 可用于查找机密密钥环中的私钥，以解密数据。相同的机制也用于查找验证签名的公钥。因此，您不再必须为 unmarshaling 指定用户 ID。

263.4. 在 PGP 签名验证过程中限制签名身份

从 Camel 2.12.3 开始。

如果您验证了一个签名，您不仅要验证签名的正确性，并且您希望检查签名是否来自特定的身份或一组特定的身份。因此，可以限制可用于验证签名的公共密钥环中的公钥数量。

签名用户 ID

```
// specify the User IDs of the expected signer identities
 List<String> expectedSigUserIds = new ArrayList<String>();
 expectedSigUserIds.add("Trusted company1");
 expectedSigUserIds.add("Trusted company2");

 PGPDataFormat pgpVerifyWithSpecificKeysAndDecrypt = new PGPDataFormat();
 pgpVerifyWithSpecificKeysAndDecrypt.setPassword("my password"); // for decrypting with
private key
 pgpVerifyWithSpecificKeysAndDecrypt.setKeyFileName(keyfileName);
 pgpVerifyWithSpecificKeysAndDecrypt.setSignatureKeyFileName(signatgureKeyfileName);
 pgpVerifyWithSpecificKeysAndDecrypt.setSignatureKeyUserids(expectedSigUserIds); // if
you have only one signer identity then you can also use setSignatureKeyUserid("expected
Signer")
```

```
from("direct:start")
        ...
        .unmarshal(pgpVerifyWithSpecificKeysAndDecrypt)
        ...
```

- 如果 PGP 内容有几个签名，验证一个签名后就会成功验证。

- 如果您不想限制签名者身份进行验证，请不要指定签名密钥用户 ID。在这种情况下，将考虑公共密钥环中的所有公钥。

263.5. ONE PGP DATA FORMAT 的几个签名

从 Camel 2.12.3 开始。

PGP 规范允许一个 PGP 数据格式包含来自不同密钥的多个签名。由于 Camel 2.13.3，可以通过指定与 secret keyring 中多个私钥相关的签名用户 ID 来创建这类 PGP 内容。

多个签名

```
PGPDataFormat pgpSignAndEncryptSeveralSignerKeys = new PGPDataFormat();
pgpSignAndEncryptSeveralSignerKeys.setKeyUserid(keyUserid); // for encrypting, you can
also use setKeyUserids if you want to encrypt with several keys
pgpSignAndEncryptSeveralSignerKeys.setKeyFileName(keyfileName);
pgpSignAndEncryptSeveralSignerKeys.setSignatureKeyFileName(signatgureKeyfileName);
pgpSignAndEncryptSeveralSignerKeys.setSignaturePassword("sdude"); // here we assume
that all private keys have the same password, if this is not the case then you can use
setPassphraseAccessor

List<String> signerUserIds = new ArrayList<String>();
signerUserIds.add("company old key");
signerUserIds.add("company new key");
pgpSignAndEncryptSeveralSignerKeys.setSignatureKeyUserids(signerUserIds);

from("direct:start")
        ...
        .marshal(pgpSignAndEncryptSeveralSignerKeys)
        ...
```

263.6. 支持 PGP DATA FORMAT MARSHALER 中的 SUB-KEYS 和 KEY FLAGS

从 *Camel 2.12.3.
*An OpenPGP V4 密钥 可以有一个主密钥和子密钥。密钥的使用通过所谓的 Key Flags 来指示。例如，

您可以有一个带有两个子键的主键：主键只用于认证其他密钥(Key Flag 0x01)，第一个子密钥只用于签名(Key Flag 0x02)，第二个子密钥应该仅用于加密(Key Flag 0x04 或 0x08)。PGP 数据格式 marshaler 会考虑主密钥和子键的密钥标记，以确定签名和加密的正确密钥。这是必要的，因为主密钥及其子键具有相同的用户 ID。

263.7. 支持自定义密钥访问器

从 *Camel 2.13.0.
开始，您可以为加密/签名实施自定义密钥访问器。上面的 PGPDataFormat 类在某些预定义方式选择，用于签名/加密或验证/解密的密钥。如果您有特殊要求，您应该选择您的密钥，而是使用 PGPKeyAccessDataFormat 类，并实施 interfaces PGPPublicKeyAccessor 和 PGPSecretKeyAccessor 作为 Bean。有默认的实现 DefaultPGPPublicKeyAccessor 和 DefaultPGPSecretKeyAccessor 来缓存密钥，以便在处理器调用时不会在每次解析密钥环时进行解析。

PGPKeyAccessDataFormat 与 PGPDataFormat 除 password, keyFileName, encryptionKeyRing, signaturePassword, signatureKeyFileName, 和 signatureKeyRing 的选项相同。

263.8. 依赖项

要在 camel 路由中使用 PGP dataformat，您需要将以下依赖项添加到 pom 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-crypto</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

263.9. 另请参阅

- 数据格式
- crypto (Digital Signatures)
- http://www.bouncycastle.org/java.html

第 264 章 属性组件

从 Camel 版本 2.3 开始提供

264.1. URI 格式

```
properties:key[?options]
```

其中 key 是要查找的属性的密钥

264.2. 选项

Properties 组件支持 17 个选项，如下所列。

Name	描述	默认值	类型
locations (common)	加载属性的位置列表。此选项将覆盖任何默认位置，并且仅使用此选项中的位置。		list
location (common)	加载属性的位置列表。您可以使用逗号分隔多个位置。此选项将覆盖任何默认位置，并且仅使用此选项中的位置。		字符串
encoding (common)	从文件系统或 classpath 加载属性文件时使用的编码。如果没有设置编码，则使用 ISO-8859-1 编码(latin-1)加载属性文件，如 java.util.Properties#load (java.io.InputStream)中所述。		字符串
propertiesResolver (common)	使用自定义属性Resolver		PropertiesResolver
propertiesParser (common)	使用自定义 PropertiesParser		PropertiesParser
cache (common)	是否缓存加载的属性。默认值为 true。	true	布尔值
propertyPrefix (advanced)	在解析前，可选前缀会添加到属性名称前。		字符串
propertySuffix (advanced)	在解析前，可选后缀附加到属性名称。		字符串

Name	描述	默认值	类型
fallbackToUnaugmented Property (advanced)	如果为 true，首先尝试使用 propertyPrefix 和 propertySuffix 解析属性名称，然后再回退到指定的普通属性名称。如果为 false，则只搜索 augmented 属性名称。	true	布尔值
defaultFallbackEnabled (common)	如果为 false，则组件不会在冒号分隔符后尝试查找密钥的默认密钥。	true	布尔值
ignoreMissingLocation (common)	是否明确忽略某个位置是否无法找到，例如未找到属性文件。	false	布尔值
prefixToken (advanced)	设置用于识别要替换的属性的前缀令牌的值。设置 null 值可恢复默认令牌（链接 DEFAULT_PREFIX_TOKEN）。	{{	字符串
suffixToken (advanced)	设置用于标识要替换的属性的后缀令牌值。设置 null 值可恢复默认令牌（链接 DEFAULT_SUFFIX_TOKEN）。	}}	字符串
initialProperties (advanced)	设置在解析任何位置之前使用的初始属性。		Properties
overrideProperties (advanced)	设置优先覆盖属性的特殊列表，并在属性存在时首先使用。		Properties
systemPropertiesMode (common)	设置系统属性模式。	2	int
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Properties 端点使用 URI 语法进行配置：

```
properties:key
```

使用以下路径和查询参数：

264.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
key	用作占位符 **所需的** Property 键		字符串

264.2.2. 查询参数(6 参数)：

Name	描述	默认值	类型
ignoreMissingLocation (common)	是否明确忽略某个位置是否无法找到，例如未找到属性文件。	false	布尔值
locations (common)	加载属性的位置列表。您可以使用逗号分隔多个位置。此选项将覆盖任何默认位置，并且仅使用此选项中的位置。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外，该处理程序记录在 WARN/ERROR 级别并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，该例外记录在 WARN/ERROR 级别并忽略。		ExceptionHandler
exchangePattern (consumer)	在创建交换时设置默认交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

提示

从 Java 代码解析属性
您可以使用 CamelContext 上的方法 resolvePropertyPlaceholders 来解析任何 Java 代码中的属性。

264.3. 使用 PROPERTYPLACEHOLDER

从 Camel 2.3 开始提供

Camel 现在在 camel-core 中提供一个新的 PropertiesComponent，允许您在定义 Camel Endpoint URI 时使用属性占位符。

这在使用 Spring 的 < property-placeholder& gt; 标签时可以正常工作。但是 Spring 有一个限制，可防止第三方框架利用 Spring 属性占位符获得完整的内容。请参阅如何在 Camel XML 中使用 Spring Property Placeholder。

提示

桥接 Spring 和 Camel 属性占位符
从 Camel 2.10 开始，您可以使用 Camel 桥接 Spring 属性占位符，更多详情请参阅以下内容。

执行此操作时通常使用属性占位符：

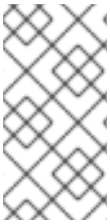
- 查找或创建端点
- 在 Registry 中查找 Bean
- Spring XML 中支持其他支持（请参阅以下示例）
- 将 Blueprint PropertyPlaceholder 与 Camel Properties 组件一起使用
- 使用 @PropertyInject 注入 POJO 中的属性
- 如果属性不存在，则使用默认值
- Camel 2.14.1 不包括开箱即用的功能，用于从 OS 环境变量、JVM 系统属性或服务 idiom 中查找属性值。
- 使用自定义功能的 Camel 2.14.1，可插入到属性组件中。

264.4. 语法

使用 Camel 的属性占位符的语法是使用 {{key}}，例如 {{file.uri}}，其中 file.uri 是 property 键。

您可以在端点 URI 的部分中使用属性占位符，例如，您可以将占位符用于 URI 中的参数。

从 Camel 2.14.1 开始，您可以指定一个默认值，如果带有键的属性不存在，如 file.url:/some/path，其中默认值是冒号后的文本（如 /some/path）。



注意

不要在 property 键中使用冒号。当您提供默认值时，冒号用作分隔符令牌，该值在 Camel 2.14.1 起被支持。

264.5. PROPERTYRESOLVER

Camel 提供了一个可插拔机制，允许您为查询属性提供自己的解析器。Camel 提供了一个默认的实现 org.apache.camel.component.properties.DefaultPropertiesResolver，它能够从文件系统、classpath 或 Registry 加载属性。您可以使用以下内容为位置添加前缀：

- ref: Camel 2.4: to lookup in the Registry
- file: 从文件系统中载入
- classpath：从 classpath 加载（如果没有提供前缀，则这也是默认设置）
- 蓝图：Camel 2.7：使用特定的 OSGi 蓝图占位符服务

264.6. 定义位置

PropertiesResolver 需要知道在哪个位置解析属性的位置。您可以定义 1 到多个位置。如果您在单个 String 属性中定义位置，您可以使用逗号分隔多个位置，例如：

```
pc.setLocation("com/mycompany/myprop.properties,com/mycompany/other.properties");
```

从 Camel 2.19.0 开始提供

您可以通过设置 optional 属性（默认为 false ）来设置在缺失时可以丢弃哪些位置，例如：

```
pc.setLocations(
    "com/mycompany/override.properties;optional=true"
    "com/mycompany/defaults.properties");
```

264.7. 在位置中使用系统和环境变量

从 Camel 2.7 开始提供

该位置现在支持将占位符用于 JVM 系统属性和 OS 环境变量。

例如：

```
location=file:${karaf.home}/etc/foo.properties
```

在上面的位置，我们使用带有键 karaf.home 的 JVM 系统属性来定义使用文件方案的位置。

要使用 OS 环境变量，您必须用 env 前缀：

```
location=file:${env:APP_HOME}/etc/foo.properties
```

其中 APP_HOME 是 OS 环境。

您可以在同一位置有多个占位符，例如：

```
location=file:${env:APP_HOME}/etc/${prop.name}.properties
```

ip=== 使用系统和环境变量来配置属性前缀和后缀

从 Camel 2.12.5, 2.13.3, 2.14.0 开始提供

propertyPrefix,propertySuffix 配置属性支持使用 JVM 系统属性和 OS 环境变量的占位符。

例如，如果 PropertiesComponent 配置了以下属性文件：

```
dev.endpoint = result1
test.endpoint = result2
```

然后，使用以下路由定义：

```
PropertiesComponent pc = context.getComponent("properties", PropertiesComponent.class);
pc.setPropertyPrefix("${stage}.");
// ...
context.addRoutes(new RouteBuilder() {
    @Override
    public void configure() throws Exception {
        from("direct:start").to("properties:mock:{{endpoint}}");
    }
});
```

可以通过将系统属性 阶段 更改为 dev 来更改目标端点（消息将路由到 mock:result1）或 test （消息将路由到 mock:result2）。

264.8. 在 JAVA DSL 中配置

您必须在 name 属性下创建并注册 PropertiesComponent， 例如：

```
PropertiesComponent pc = new PropertiesComponent();
pc.setLocation("classpath:com/mycompany/myprop.properties");
context.addComponent("properties", pc);
```

264.9. 在 SPRING XML 中配置

Spring XML 提供了两种配置变体。您可以将 spring bean 定义为 PropertiesComponent，它类似于 Java DSL 中完成的方式。或者，您可以使用 < propertyPlaceholder> 标签。

```
<bean id="properties" class="org.apache.camel.component.properties.PropertiesComponent">
    <property name="location" value="classpath:com/mycompany/myprop.properties"/>
</bean>
```

使用 & lt;propertyPlaceholder > 标签会使配置更加全新的，例如：

```
<camelContext ...>
  <propertyPlaceholder id="properties" location="com/mycompany/myprop.properties"/>
</camelContext>
```

通过 location 标签设置属性位置可以正常工作，但有些资源需要考虑并从 Camel 2.19.0 开始，您可以使用专用 propertiesLocation 设置属性位置：

```
<camelContext ...>
  <propertyPlaceholder id="myPropertyPlaceholder">
    <propertiesLocation
      resolver = "classpath"
      path     = "com/my/company/something/my-properties-1.properties"
      optional = "false"/>
    <propertiesLocation
      resolver = "classpath"
      path     = "com/my/company/something/my-properties-2.properties"
      optional = "false"/>
    <propertiesLocation
      resolver = "file"
      path     = "${karaf.home}/etc/my-override.properties"
      optional = "true"/>
  </propertyPlaceholder>
</camelContext>
```

提示

在 XML
Camel 2.10 中指定 cache 选项支持在 Spring 和 Blueprint XML 中为 cache 选项指定一个值。

264.10. 使用 REGISTRY 中的属性

从 Camel 2.4
开始，例如在 OSGi 中，您可能希望公开服务，该服务将属性返回为 java.util.Properties 对象。

然后您可以按如下方式设置 **Properties** 组件：

```
<propertyPlaceholder id="properties" location="ref:myProperties"/>
```

其中 myProperties 是在 OSGi 注册表中用于查找的 id。请注意，我们使用 ref: 前缀来告知 Camel 它应该查找 Registry 的属性。

264.11. 使用属性组件的示例

在端点 URI 中使用属性占位符时，您可以使用 properties: 组件，或者在 URI 中直接定义占位符。我们将显示这两种情况的示例，从前者开始。

```
// properties
cool.end=mock:result

// route
from("direct:start").to("properties:{{cool.end}}");
```

您还可以使用占位符作为端点 uri 的一部分：

```
// properties
cool.foo=result

// route
from("direct:start").to("properties:mock:{{cool.foo}}");
```

在上例中，to 端点将解析为 mock:result。

您还可以具有相互引用的属性，例如：

```
// properties
cool.foo=result
cool.concat=mock:{{cool.foo}}

// route
from("direct:start").to("properties:mock:{{cool.concat}}");
```

注意 cool.concat 如何引用另一个属性。

属性： 组件还提供了使用 location 选项覆盖并提供给定 uri 中 的位置 ：

```
from("direct:start").to("properties:bar.end?locations=com/mycompany/bar.properties");
```

264.12. 例子

您也可以直接在端点 uris 中使用属性占位符，而无需使用 属性：

```
// properties
cool.foo=result

// route
from("direct:start").to("mock:{{cool.foo}}");
```

您可以按原样使用它们：

```
// properties
cool.start=direct:start
cool.showid=true
cool.result=result

// route
from("{{cool.start}}")
    .to("log:{{cool.start}}?showBodyType=false&showExchangeId={{cool.showid}}")
    .to("mock:{{cool.result}}");
```

您还可以使用 ProducerTemplate 时的属性占位符，例如：

```
template.sendBody("{{cool.start}}", "Hello World");
```

264.13. 使用 简单 语言的示例

Simple 语言现在支持使用属性占位符，例如在下面的路由中：

```
// properties
cheese.quote=Camel rocks

// route
from("direct:start")
    .transform().simple("Hi ${body} do you think ${properties:cheese.quote}?");
```

您还可以使用 Simple 语言指定位置，例如：

```
// bar.properties
bar.quote=Beer tastes good
```

```
// route
from("direct:start")
    .transform().simple("Hi ${body}. ${properties:com/mycompany/bar.properties:bar.quote}.");
```

264.14. SPRING XML 中支持的其他属性占位符

在许多 Camel Spring XML 标签中也支持属性占位符，如 <package>, &lt ;packageScan>, <contextScan>, <jmxAgent>, <endpoint>, <routeBuilder>, <proxy > 等。

以下示例在 < jmxAgent> 标签中包含属性占位符：

您还可以在 < camelContext> 标签上的 各种属性中定义属性占位符，如 trace，如下所示：

264.15. 使用 JVM 系统属性覆盖属性设置

从 Camel 2.5开始，可以使用 JVM 系统属性在运行时覆盖属性值，而无需重启应用程序来获取更改。这也可以通过从命令行创建与用新值替换的属性相同的 JVM 系统属性来完成。下面是一个示例

```
PropertiesComponent pc = context.getComponent("properties", PropertiesComponent.class);
pc.setCache(false);

System.setProperty("cool.end", "mock:override");
System.setProperty("cool.result", "override");

context.addRoutes(new RouteBuilder() {
    @Override
    public void configure() throws Exception {
        from("direct:start").to("properties:cool.end");
        from("direct:foo").to("properties:mock:{{cool.result}}");
    }
});
context.start();

getMockEndpoint("mock:override").expectedMessageCount(2);

template.sendBody("direct:start", "Hello World");
template.sendBody("direct:foo", "Hello Foo");

System.clearProperty("cool.end");
System.clearProperty("cool.result");

assertMockEndpointsSatisfied();
```

264.16. 将属性占位符用于 XML DSL 中任何类型的属性

从 Camel 2.7 开始提供

注意

如果您使用 OSGi 蓝图，则这仅适用于 2.11.1 或 2.10.5。

在以前的版本中，它只在支持占位符的 XML DSL 中的 xs:string 类型属性。例如，timeout 属性会是一个 xs:int 类型，因此您无法将字符串值设置为占位符键。现在，可以使用特殊的占位符命名空间进行 Camel 2.7。

在以下示例中，我们为命名空间 http://camel.apache.org/schema/placeholder 使用 prop 前缀，在 XML DSLs 中的属性中使用 prop 前缀。请注意，在 Multicast 中使用此选项代表，选项 stopOnException 应该是带有键"stop"的占位符值。

在我们的属性文件中，我们将值定义为

```
stop=true
```

264.17. 在 CAMEL 路由中使用 BLUEPRINT 属性占位符

从 Camel 2.7 开始提供

Camel 支持提供属性占位符服务的 Blueprint。Camel 支持对配置的惯例，因此您只需要在 XML 文件中定义 OSGi Blueprint 属性占位符，如下所示：

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns:cm="http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.0.0"
         xsi:schemaLocation="
         http://www.osgi.org/xmlns/blueprint/v1.0.0
https://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd">

   <!-- OSGI blueprint property placeholder -->
   <cm:property-placeholder id="myblueprint.placeholder" persistent-id="camel.blueprint">
      <!-- list some properties as needed -->
      <cm:default-properties>
         <cm:property name="result" value="mock:result"/>
      </cm:default-properties>
   </cm:property-placeholder>
```

```
    <camelContext xmlns="http://camel.apache.org/schema/blueprint">
      <!-- in the route we can use {{ }} placeholders which will lookup in blueprint
           as Camel will auto detect the OSGi blueprint property placeholder and use it -->
      <route>
        <from uri="direct:start"/>
        <to uri="mock:foo"/>
        <to uri="{{result}}"/>
      </route>
    </camelContext>
  </blueprint>
```

264.17.1. 在 Camel 路由中使用 OSGi 蓝图属性占位符

默认情况下，Camel 会检测和使用 OSGi 蓝图属性占位符服务。您可以通过在 < camelContext > 定义中将属性 useBlueprintPropertyResolver 设置为 false 来禁用此功能。

264.17.2. 关于占位符语法

注意我们如何将 Camel 语法用于 Camel 路由中 的占位符 {{ 和 }}，后者将从 OSGi 蓝图中查找值。

占位符的蓝图语法为 ${ }。因此，在 & lt;camelContext& gt; 之外，您必须使用 ${ } 语法。其中，如 & lt;camelContext> 内，您必须使用 { { 和 }} 语法。

OSGi 蓝图允许您配置语法，因此如果需要，您可以真正保持一致。

您还可以根据 id 明确引用特定的 OSGi 蓝图属性占位符。为此，您需要使用 Camel 的 < propertyPlaceholder& gt;，如下例所示：

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:cm="http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.0.0"
           xsi:schemaLocation="
           http://www.osgi.org/xmlns/blueprint/v1.0.0
https://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd">

    <!-- OSGI blueprint property placeholder -->
    <cm:property-placeholder id="myblueprint.placeholder" persistent-id="camel.blueprint">
      <!-- list some properties as needed -->
      <cm:default-properties>
          <cm:property name="prefix.result" value="mock:result"/>
      </cm:default-properties>
    </cm:property-placeholder>

    <camelContext xmlns="http://camel.apache.org/schema/blueprint">
      <!-- using Camel properties component and refer to the blueprint property placeholder by
```

```
its id -->
        <propertyPlaceholder id="properties" location="blueprint:myblueprint.placeholder"
                             prefixToken="[[" suffixToken="]]"
                             propertyPrefix="prefix."/>

        <!-- in the route we can use {{ }} placeholders which will lookup in blueprint -->
        <route>
            <from uri="direct:start"/>
            <to uri="mock:foo"/>
            <to uri="[[result]]"/>
        </route>
    </camelContext>
</blueprint>
```

264.18. 在 CAMEL 中明确引用 OSGI 蓝图占位符

请注意，我们如何使用 蓝图 方案根据其 id 指向 OSGi 蓝图占位符。这可让您混合和匹配，例如，您也可以在位置有额外的方案。例如，要从 classpath 加载一个文件，您可以执行以下操作：

```
location="blueprint:myblueprint.placeholder,classpath:myproperties.properties"
```

每个位置用逗号分开。

264.19. 覆盖 CAMELCONTEXT 之外的蓝图属性占位符

从 Camel 2.10.4 开始提供

在 Blueprint XML 文件中使用 Blueprint 属性占位符时，您可以在 XML 文件中直接声明属性，如下所示：

请注意，我们有一个 &lt ;bean >，它引用其中一个属性。在 Camel 路由中，我们使用 {{ and }} 表示法引用另一个。

现在，如果要从单元测试覆盖这些蓝图属性，您可以执行以下操作，如下所示：

要做到这一点，我们覆盖并实施 useOverridePropertiesWithConfigAdmin 方法。然后，我们可以将想要覆盖的属性放在给定的 props 参数中。返回 值必须是 < cm:property-placeholder> 标签的 persistence- id，您在蓝图 XML 文件中定义。

264.20. 将 .CFG 或 .PROPERTIES 文件用于 BLUEPRINT 属性占位符

从 Camel 2.10.4 开始提供

当在 Blueprint XML 文件中使用 Blueprint 属性占位符时，您可以在 .properties 或 .cfg 文件中声明属性。如果您使用 Apache ServieMix / Karaf，则此容器有惯例，它从 etc 目录中的一个文件加载属性，其使用命名 etc/pid.cfg，其中 pid 是 persistence-id。

例如，在蓝图 XML 文件中，我们具有 persistence-id="stuff"，这意味着它将作为 etc/stuff.cfg 加载配置文件。

现在，如果要单元测试这个蓝图 XML 文件，您可以覆盖 loadConfigAdminConfigurationFile，并告诉 Camel 要载入哪个文件，如下所示：

请注意，此方法需要返回具有 2 值的 String[]。第 1st 值是要加载的配置文件的路径。2nd 值是 <cm:property-placeholder> 标签的 persistence- id。

material .cfg 文件只是带有属性占位符的普通属性文件，例如：

```
== this is a comment
greeting=Bye
```

264.21. 使用 .CFG 文件并覆盖 BLUEPRINT 属性占位符的属性

您也可以两者兼而有之。下面是一个完整的示例。首先，我们有蓝图 XML 文件：

在单元测试类中，我们按如下方式进行：

etc/stuff.cfg 配置文件包含

```
greeting=Bye
echo=Yay
destination=mock:result
```

264.22. 桥接 SPRING 和 CAMEL 属性占位符

从 Camel 2.10 开始提供

Spring Framework 不允许 Apache Camel 等第三方框架在 Spring 属性占位符机制中无缝 hook。但是，您可以通过声明类型为 org.apache.camel.spring.BridgePropertyPlaceholderConfigurer 类型的 Spring bean 来轻松地桥接 Spring 和 Camel，这是 Spring org.springframework.beans.factory.config.PropertyPlaceholderConfigurer 类型。

要桥接 Spring 和 Camel，您必须定义一个 Bean，如下所示：

桥接 Spring 和 Camel 属性占位符

您不能 同时使用 spring <context:property-placeholder> 命名空间；这不可能。

声明此 bean 后，您可以使用 Spring 风格定义属性占位符，并在 <camelContext> 标签中定义 Camel 样式，如下所示：

使用 bridge 属性占位符

注意 hello bean 如何使用 ${ } 表示法使用纯的 Spring 属性占位符。在 Camel 路由中，我们使用带有 {{ 和 }} 的 Camel 占位符表示法。

264.23. 使用 CAMELS SIMPLE 语言 CLASHING SPRING 属性占位符

在使用 Spring 桥接占位符时请注意，spring ${ } 语法与 Camel 中的 简单 内容一致，因此请谨慎操作。例如：

```
<setHeader headerName="Exchange.FILE_NAME">
  <simple>{{file.rootdir}}/${in.header.CamelFileName}</simple>
</setHeader>
```

clashes 与 Spring 属性占位符，您应该使用 $simple{ } 来指明在 Camel 中使用 Simple 语言。

```
<setHeader headerName="Exchange.FILE_NAME">
  <simple>{{file.rootdir}}/$simple{in.header.CamelFileName}</simple>
</setHeader>
```

另一种方法是，将 PropertyPlaceholderConfigurer 使用 ignoreUnresolvablePlaceholders 选项配置为 true。

264.24. 覆盖 CAMEL TEST KIT 中的属性

从 Camel 2.10 开始提供

使用 Camel 测试并使用 Properties 组件时，您可能需要提供直接从单元测试源代码中使用的属性。现在，从 Camel 2.10 开始，作为 Camel test kits，如 CamelTestSupport 类提供了以下方法

- useOverridePropertiesWithPropertiesComponent
- ignoreMissingLocationWithPropertiesComponent

例如，在单元测试类中，您可以覆盖 useOverridePropertiesWithPropertiesComponent 方法，并返回 java.util.Properties，其中包含应该首选使用的属性。

264.24.1. 从单元测试源内提供属性

这可以从任何 Camel Test kits 来完成，如 camel-test、camel-test-spring 和 camel-test-blueprint。

ignoreMissingLocationWithPropertiesComponent 可用于指示 Camel 忽略任何不可发现的位置，例如，在运行单元测试时，在无法访问属性位置的环境中。

264.25. 使用 @PROPERTYINJECT

从 Camel 2.12 开始提供

Camel 允许使用 @PropertyInject 注释（可以在字段和 setter 方法上设置）注入 POJO 中的属性占位符。

例如，您可以将该类与 RouteBuilder 类一起使用，如下所示：

```
public class MyRouteBuilder extends RouteBuilder {

    @PropertyInject("hello")
    private String greeting;

    @Override
    public void configure() throws Exception {
        from("direct:start")
            .transform().constant(greeting)
            .to("{{result}}");
    }

}
```

请注意，我们使用 @PropertyInject 标注了 greeting 字段，并将它定义为使用键 "hello "。然后，Camel 会使用这个键查找属性并注入其值，并转换为 String 类型。

您还可以在键中使用多个占位符和文本，例如，我们可以执行以下操作：

```
@PropertyInject("Hello {{name}} how are you?")
private String greeting;
```

这将使用它们 "名称" 键查找占位符。

如果键不存在，您还可以添加一个默认值，例如：

```
@PropertyInject(value = "myTimeout", defaultValue = "5000")
private int timeout;
```

264.26. 使用开箱即用的功能

从 Camel 2.14.1 开始提供

Properties 组件包括开箱即用的以下功能

- env - 用于从 OS 环境变量查找属性的功能
- sys - 从 Java JVM 系统属性查找属性的功能

- service - 使用服务命名 idiom 从 OS 环境变量查找属性的功能
- service.name - Camel 2.16.1 ： 使用服务命名 idiom 返回 hostname 部分的从 OS 环境变量中查找属性的功能
- service.port - Camel 2.16.1： 使用服务命名 idiom 仅返回端口部分的从 OS 环境变量中查找属性的功能

正如您所见，这些功能旨在方便从环境中查找值。在开箱即用的情况下，可以轻松使用它们，如下所示：

```
<camelContext xmlns="http://camel.apache.org/schema/blueprint">

  <route>
    <from uri="direct:start"/>
    <to uri="{{env:SOMENAME}}"/>
    <to uri="{{sys:MyJvmPropertyName}}"/>
  </route>
</camelContext>
```

您还可以使用默认值，如果属性不存在，您可以定义一个默认值，如下所示，其中默认值是 log:foo 和 log:bar 值。

```
<camelContext xmlns="http://camel.apache.org/schema/blueprint">

  <route>
    <from uri="direct:start"/>
    <to uri="{{env:SOMENAME:log:foo}}"/>
    <to uri="{{sys:MyJvmPropertyName:log:bar}}"/>
  </route>
</camelContext>
```

服务功能用于查找使用服务命名 idiom 的 OS 环境变量定义的服务，以使用 主机名：端口引用服务位置：

- 名称_SERVICE_HOST
- NAME_SERVICE_PORT

换句话说，服务使用 _SERVICE_HOST 和 _SERVICE_PORT 作为前缀。因此，如果服务命名为 FOO，则 OS 环境变量应设置为

```
export $FOO_SERVICE_HOST=myserver
export $FOO_SERVICE_PORT=8888
```

例如，如果 FOO 服务远程 HTTP 服务，则我们可以引用 Camel 端点 uri 中的服务，并使用 HTTP 组件进行 HTTP 调用：

```
<camelContext xmlns="http://camel.apache.org/schema/blueprint">
  <route>
    <from uri="direct:start"/>
    <to uri="http://{{service:FOO}}/myapp"/>
  </route>
</camelContext>
```

如果服务尚未定义，我们可以使用默认值，例如在 localhost 上调用服务，对于单元测试等，我们可以使用默认值。

```
<camelContext xmlns="http://camel.apache.org/schema/blueprint">
  <route>
    <from uri="direct:start"/>
    <to uri="http://{{service:FOO:localhost:8080}}/myapp"/>
  </route>
</camelContext>
```

264.27. 使用自定义功能

从 Camel 2.14.1 开始提供

Properties 组件允许插件第三方功能，它们可在解析属性占位符时使用。然后，这些功能能够执行自定义逻辑来解析占位符，如在数据库中查找、执行自定义计算或不用处。函数的名称变为占位符中使用的前缀。下面是以下示例代码中的最佳描述

```
<bean id="beerFunction" class="MyBeerFunction"/>

<camelContext xmlns="http://camel.apache.org/schema/blueprint">
  <propertyPlaceholder id="properties">
    <propertiesFunction ref="beerFunction"/>
  </propertyPlaceholder>

  <route>
    <from uri="direct:start"/>
    <to uri="{{beer:FOO}}"/>
```

```
    <to uri="{{beer:BAR}}"/>
  </route>
</camelContext>
```

注意

从 camel 2.19.0，location 属性(on propertyPlaceholder tag)不是更多强制的

在这里，我们有一个 Camel XML 路由，它定义了 <propertyPlaceholder> 来使用一个自定义功能，我们引用为 bean id - eg the beerFunction。由于 beer 函数使用 "beer" 作为其名称，因此占位符语法可以通过以 beer:value 开始触发 beer 功能。

功能的实现只是两个方法，如下所示：

```
public static final class MyBeerFunction implements PropertiesFunction {

    @Override
    public String getName() {
        return "beer";
    }

    @Override
    public String apply(String remainder) {
        return "mock:" + remainder.toLowerCase();
    }
}
```

函数必须实施 org.apache.camel.component.properties.PropertiesFunction 接口。method getName 是函数的名称，如 beer。应用 方法就是我们实施自定义逻辑的位置。因为示例代码来自单元测试，它只是返回一个指向模拟端点的值。

要从 Java 代码注册自定义功能，如下所示：

```
PropertiesComponent pc = context.getComponent("properties", PropertiesComponent.class);
pc.addFunction(new MyBeerFunction());
```

264.28. 另请参阅

- 属性 组件

- ***Jasypt 在属性中使用加密值（例如密码）***

第 265 章 PROTOBUF DATAFORMAT

从 Camel 版本 2.2.0 开始提供

第 266 章 PROTOBUF - 协议缓冲

"protocol Buffers - Google 的数据交换格式"

Camel 提供了一个数据格式，用于 Java 和协议缓冲协议之间的序列化。项目站点详细信息，为什么您可能想 通过 xml 选择此格式。协议缓冲是语言中立且平台中立，因此您的 Camel 路由生成的消息可能会被其他语言实现使用。

API 站点
Protobuf 实现

protobuf Java 教程

266.1. PROTOBUF 选项

Protobuf dataformat 支持 3 个选项，如下所列。

Name	默认值	Java 类型	描述
instanceClass		字符串	unarmshalling 时使用的类名称
contentTypeFormat	原生	字符串	定义内容类型格式，其中 protobuf 消息将从(to) Java 序列化/反序列化。格式可以是 native 或 json，可以是原生 protobuf 或 json 字段。默认值为 native。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

266.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.protobuf.content-type-format	定义内容类型格式，其中 protobuf 消息将从(to) Java 序列化/反序列化。格式可以是 native 或 json，可以是原生 protobuf 或 json 字段。默认值为 native。	原生	字符串
camel.dataformat.protobuf.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.protobuf.enabled	启用 protobuf dataformat	true	布尔值
camel.dataformat.protobuf.instance-class	unarmshalling 时使用的类名称		字符串

ND

266.3. 内容类型格式（从 CAMEL 2.19开始）

可以解析 JSON 消息，将其转换为 protobuf 格式，并使用原生 util 转换程序将其返回。要使用这个选项，请将 contentTypeFormat 值设置为 'json' 或调用带有第二个参数的 protobuf。如果没有指定默认实例，请始终使用原生 protobuf 格式。示例代码显示如下：

```
from("direct:marshal")
    .unmarshal()
    .protobuf("org.apache.camel.dataformat.protobuf.generated.AddressBookProtos$Person",
"json")
    .to("mock:reverse");
```

266.4. PROTOBUF 概述

这一快速概述如何使用 Protobuf。详情请查看 完整的教程

266.5. 定义原型格式

第一步是定义交换正文的格式。这在 .proto 文件中定义，如下所示：

addressbook.proto

```
syntax = "proto2";

package org.apache.camel.component.protobuf;

option java_package = "org.apache.camel.component.protobuf";
option java_outer_classname = "AddressBookProtos";

message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    required string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }

  repeated PhoneNumber phone = 4;
}

message AddressBook {
  repeated Person person = 1;
}
```

266.6. 生成 JAVA 类

Protobuf SDK 提供了一个编译器，它将为在 .proto 文件中定义的格式生成 Java 类。如果您的操作系统受 Protobuf Java 代码生成器 maven 插件支持，您可以通过将以下配置添加到 pom.xml 来自动化生成 protobuf Java 代码：

在项目 pom.xml 的 < build> 标签中插入操作系统和 CPU 架构检测扩展，或者手动设置 ${os.detected.classifier} 参数

```
<extensions>
  <extension>
    <groupId>kr.motd.maven</groupId>
    <artifactId>os-maven-plugin</artifactId>
    <version>1.4.1.Final</version>
  </extension>
</extensions>
```

插入 gRPC 和 protobuf Java 代码生成器插件 & lt;plugins > 项目 pom.xml 标签

```
<plugin>
  <groupId>org.xolstice.maven.plugins</groupId>
  <artifactId>protobuf-maven-plugin</artifactId>
  <version>0.5.0</version>
  <extensions>true</extensions>
  <executions>
    <execution>
      <goals>
        <goal>test-compile</goal>
        <goal>compile</goal>
      </goals>
      <configuration>
        <protocArtifact>com.google.protobuf:protoc:${protobuf-
version}:exe:${os.detected.classifier}</protocArtifact>
      </configuration>
    </execution>
  </executions>
</plugin>
```

您还可以为手动所需的任何其他支持语言运行编译器。

```
protoc --java_out=. ./proto/addressbook.proto
```

这将生成一个名为 AddressBookProtos 的 Java 类，其中包含 Person 和 AddressBook 的内部类。还将为您实施构建器。生成的类实施 com.google.protobuf.Message，这是序列化机制所需的。因此，您的交换正文中只使用这些类非常重要。如果您试图使用未实现 com.google.protobuf.Message 的类，Camel 会在路由创建时抛出异常。使用生成的构建器从任何现有域类转换数据。

266.7. JAVA DSL

您可以使用创建 ProtobufDataFormat 实例，并将其传递给 Camel DataFormat marshal 和 unmarshal API，如下所示：

```
ProtobufDataFormat format = new ProtobufDataFormat(Person.getDefaultInstance());

from("direct:in").marshal(format);
from("direct:back").unmarshal(format).to("mock:reverse");
```

或者，使用 DSL protobuf （）传递 unmarshal 默认实例或默认实例类名称，如下所示。

```
    // You don't need to specify the default instance for protobuf marshaling
    from("direct:marshal").marshal().protobuf();
    from("direct:unmarshalA").unmarshal()

.protobuf("org.apache.camel.dataformat.protobuf.generated.AddressBookProtos$Person")
        .to("mock:reverse");


from("direct:unmarshalB").unmarshal().protobuf(Person.getDefaultInstance()).to("mock:revers
e");
```

266.8. SPRING DSL

以下示例演示了如何使用 Protobuf 来使用 Spring 配置 protobuf 数据类型

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="direct:start"/>
    <unmarshal>
      <protobuf
instanceClass="org.apache.camel.dataformat.protobuf.generated.AddressBookProtos$Perso
n" />
    </unmarshal>
    <to uri="mock:result"/>
  </route>
</camelContext>
```

266.9. 依赖项

要在 camel 路由中使用 Protobuf，您需要添加对实现此数据格式的 camel-protobuf 的依赖关系。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-protobuf</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

266.10. 另请参阅

- Camel gRPC 组件

第 267 章 PUBNUB 组件

从 Camel 版本 2.19 开始提供

Camel PubNub 组件可用于与已连接设备的 PubNub 数据流网络进行通信。此组件使用 pubnub java 库。

使用案例包括：

- 聊天房间：发送和接收信息
- 位置和连接车：分配税务级
- 智能传感器：为数据视觉化从传感器中接收数据
- 健康：监控来自病人的可疑设备的核心率
- Multiplayer gamings
- 交互式介质：观众管理投票系统

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-pubnub</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

267.1. URI 格式

```
pubnub:channel[?options]
```

其中 channel 是要发布或订阅的 PubNub 频道。

267.2. 选项

PubNub 组件没有选项。

PubNub 端点使用 URI 语法进行配置：

```
pubnub:channel
```

使用以下路径和查询参数：

267.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
channel	**所需** 频道用于订阅/发布事件		字符串

267.2.2. 查询参数(14 参数)：

Name	描述	默认值	类型
UUID (common)	要用作设备标识符的 UUID，如果没有通过，则会生成一个默认的 UUID。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
withPresence (consumer)	另外，订阅相关的存在信息	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler

Name	描述	默认值	类型
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
operation (producer)	要执行的操作。PUBLISH：默认.向频道的所有订阅者发送消息。FIRE ：允许客户端向 BLOCKS Event Handlers 发送消息。这些消息将直接进入频道中注册的任何事件处理程序。HERENOW ：获取有关频道当前状态的信息，包括当前订阅了频道的唯一 user-ids 列表以及总八进制数。WHERENOW ：获取有关订阅 uuid 的当前频道列表的信息。GETSTATE: 用来获取特定于订阅者 uuid 的键/值对。状态信息作为键/值对的 JSON 对象提供：用于设置特定于订阅者 uuid GETHISTORY 的键/值对：Fetches 历史消息。		字符串
pubnub (advanced)	对 registry 中的 Pubnub 客户端的引用。		PubNub
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
authkey (security)	如果使用 Access Manager，客户端将在所有受限请求中使用这个 authKey。		字符串
cipherKey (security)	如果传递密码，则会加密所有 communicatons to/from PubNub。		字符串
publishKey (security)	从您的 PubNub 帐户获取的发布密钥。发布消息时需要。		字符串
secretKey (security)	用于消息签名的 secret 密钥。		字符串
安全 （安全）	使用 SSL 进行安全传输。	true	布尔值
subscribeKey (security)	从您的 PubNub 帐户获取的 subscribe 密钥。订阅频道或侦听存在事件时需要		字符串

267.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component .pubnub.enabled	启用 pubnub 组件	true	布尔值

Name	描述	默认值	类型
camel.component.pubnub.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

267.4. 订阅时的消息标头

Name	描述
CamelPubNubTimeToken	事件的 Timestamp。
CamelPubNubChannel	消息所属的频道。

267.5. 消息正文

消息正文可以包含任何 JSON 序列化数据，包括：Objects、Arrays、Ints 和 Strings。消息数据不应包含特殊的 Java V4 类或功能，因为它们不会序列化。字符串内容可以包含任何单字节或多字节 UTF-8

发送时自动执行对象序列化。只需传递完整对象作为消息有效负载。PubNub 将负责对象序列化。

当接收消息正文时，会使用 PubNub API 提供的对象。

267.6. 例子

267.6.1. 发布事件

生成时的默认操作.以下片段将 PojoBean 生成的事件发布到频道 iot。

```
from("timer:mytimer")
  // generate some data as POJO.
  .bean(PojoBean.class)
  .to("pubnub:iot?publishKey=mypublishKey");
```

267.6.2. 触发事件 aka BLOCKS Event Handlers

有关可调用的所有无服务器功能类型，请参阅 https://www.pubnub.com/blocks-catalog/。地理位置查找示例

```
from("timer:geotimer")
    .process(exchange -> exchange.getIn().setBody(new Foo("bar", "TEXT")))
    .to("pubnub:eon-maps-geolocation-input?
operation=fire&publishKey=mypubkey&subscribeKey=mysubkey");

from("pubnub:eon-map-geolocation-output?subscribeKey=mysubkey)
    // geolocation output will be logged here
    .log("${body}");
```

267.6.3. 订阅事件

以下片段侦听 iot 频道上的事件。如果您可以使用Presens 添加选项，您也会接收频道 Join，Leaave asf 事件。

```
from("pubnub:iot?subscribeKey=mySubscribeKey")
    .log("${body}")
    .to("mock:result");
```

267.6.4. 执行操作

在这里：获取有关频道当前状态的信息，包括当前订阅了频道的唯一 user-ids 列表以及频道总数

```
from("direct:control")
    .to("pubnub:myChannel?
publishKey=mypublishKey&subscribeKey=mySubscribeKey&operation=herenow")
    .to("mock:result");
```

其中现在：获取有关订阅 uuid 的当前频道列表的信息

```
from("direct:control")
    .to("pubnub:myChannel?
publishKey=mypublishKey&subscribeKey=mySubscribeKey&operation=wherenow&uuid=spy
onme")
    .to("mock:result");
```

setstate : used 用来设置特定于订阅者 uuid 的键/值对。

```
from("direct:control")
    .bean(StateGenerator.class)
    .to("pubnub:myChannel?
```

```
publishKey=mypublishKey&subscribeKey=mySubscribeKey&operation=setstate&uuid=myuuid");
```

gethistory ：显示频道的历史消息。

```
from("direct:control")
   .to("pubnub:myChannel?
publishKey=mypublishKey&subscribeKey=mySubscribeKey&operation=gethistory");
```

测试目录中有几个示例显示了 PubNub 功能。它们需要一个 PubNub 帐户，您可以在其中获取发布和订阅密钥。

示例 PubNubSensorExample 已经包含 PubNub 提供的订阅密钥，因此可以在没有帐户的情况下运行。这个示例演示了 PubNub 组件订阅一个无限的传感器数据流。

267.7. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- RSS

第 268 章 APACHE PULSAR 组件

从 Camel 版本 2.24 开始提供

Maven 用户需要将以下依赖项添加到此组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-pulsar</artifactId>
  <!-- use the same version as your Camel core version -->
  <version>x.y.z</version>
</dependency>
```

268.1. URI 格式

```
pulsar:[persistent|non-persistent]://tenant/namespace/topic
```

268.2. 选项

Apache Pulsar 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
自动配置(common)	pulsar 自动配置		自动配置
pulsarClient (common)	pulsar 客户端		PulsarClient
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Apache Pulsar 端点使用 URI 语法进行配置：

```
pulsar:uri
```

使用以下路径和查询参数：

268.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
topicUri	主题的完整 URI 路径，包括类型、租户和命名空间		字符串

268.2.2. 查询参数(11 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
consumerName (consumer)	当订阅为 EXCLUSIVE 时，消费者的名称	sole-consumer	字符串
consumerNamePrefix (consumer)	当使用 SHARED 或 FAILOVER 订阅时，要添加到消费者名称的前缀	cons	字符串
consumerQueueSize (consumer)	消费者队列的大小 - 默认为 10	10	int
numberOfConsumers (consumer)	消费者数量 - 默认为 1	1	int
subscriptionName (consumer)	要使用的订阅名称	subscription	字符串
subscriptionType (consumer)	订阅类型 EXCLUSIVESHAREDFAILOVER，默认为 EXCLUSIVE	EXCLUSIVE	SubscriptionType
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
producerName (producer)	producer 的名称	default-producer	字符串

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

268.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.pulsar.enabled	是否启用 pulsar 组件的自动配置。这默认是启用的。		布尔值
camel.component.pulsar.pulsar-client	pulsar 客户端。选项是 org.apache.pulsar.client.api.PulsarClient 类型。		字符串
camel.component.pulsar.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 269 章 QUARTZ 组件（已弃用）

从 Camel 版本 1.0 开始提供

quartz: 组件使用 Quartz Scheduler 1.x 提供预定的消息交付。
每个端点代表不同的计时器（在 Quartz 术语，一个 Trigger 和 JobDetail）。

提示

如果您使用 Quartz 2.x，则从 Camel 2.12 开始，您应该使用 Quartz2 组件

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-quartz</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

269.1. URI 格式

```
quartz://timerName?options
quartz://groupName/timerName?options
quartz://groupName/timerName?cron=expression
quartz://timerName?cron=expression
```

组件使用 CronTrigger 或 SimpleTrigger。如果没有提供 cron 表达式，则组件将使用一个简单的触发器。如果没有提供 groupName，quartz 组件将使用 Camel 组名称。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

269.2. 选项

Quartz 组件支持 8 个选项，如下所列。

Name	描述	默认值	类型
factory (advanced)	使用用于创建调度程序的自定义 SchedulerFactory。		SchedulerFactory
scheduler (advanced)	使用自定义配置的 Quartz 调度程序，而不是创建新的调度程序。		scheduler
properties (consumer)	配置 Quartz 调度程序的属性。		Properties
propertiesFile (consumer)	从 classpath 加载的属性的文件名		字符串
startDelayedSeconds (scheduler)	启动 quartz 调度程序前等待的秒数。		int
autoStartScheduler (consumer)	调度程序是否应自动启动。这个选项是默认的 true	true	布尔值
enableJmx (consumer)	是否启用 Quartz JMX，允许从 JMX 管理 Quartz 调度程序。这个选项是默认的 true	true	布尔值
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Quartz 端点使用 URI 语法进行配置：

```
quartz:groupName/timerName
```

使用以下路径和查询参数：

269.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
groupName	要使用的 quartz 组名称。组名称和计时器名称的组合应该是唯一的。	Camel	字符串
timerName	**必需** 要使用的 quartz 计时器名称。组名称和计时器名称的组合应该是唯一的。		字符串

269.2.2. 查询参数(13 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
cron (consumer)	指定定义要触发的时间的 cron 表达式。		字符串
deleteJob (consumer)	如果设置为 true，则触发器会在路由停止时自动删除。其他设为 false，它将保留在调度程序中。当设置为 false 时，它还意味着用户可以使用 camel Uri 重复使用预先配置的触发器。只需确保名称匹配。请注意，您无法同时将 deleteJob 和 pauseJob 设置为 true。	true	布尔值
fireNow (consumer)	使用简单触发器时触发调度程序sap （这个选项不支持 cron）	false	布尔值
pauseJob (consumer)	如果设置为 true，则触发器会在路由停止时自动暂停。其他设为 false，它将保留在调度程序中。当设置为 false 时，它还意味着用户可以使用 camel Uri 重复使用预先配置的触发器。只需确保名称匹配。请注意，您无法同时将 deleteJob 和 pauseJob 设置为 true。	false	布尔值
startDelayedSeconds (consumer)	启动 quartz 调度程序前等待的秒数。		int
stateful (consumer)	使用 Quartz StatefulJob 而不是默认作业。	false	布尔值
usingFixedCamelContextName (consumer)	如果为 true，JobDataMap 直接使用 CamelContext 名称来引用 CamelContext，如果为 false，则 JobDataMap 使用 CamelContext 管理名称，可在部署期间更改。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
jobParameters (advanced)	配置作业上的附加选项。		Map

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
triggerParameters (advanced)	要在触发器中配置附加选项：		Map

使用 StatefulJob 时，JobDataMap 每次执行作业后都会重新具有持久性，从而为下一次执行保留状态。

INFO：在 OSGi 中运行并使用 quartz 路由的多个捆绑包，如果您在 OSGi 中（如 Apache ServiceMix 或 Apache Karaf）中运行，并且具有从 Quartz 端点开始的 Camel 路由的多个捆绑包，请确保将
分配给这个 id 唯一的 <camelContext>，因为 OSGi 容器中 QuartzScheduler 需要它。如果您没有在 <camelContext> 上设置任何 id，则会自动分配唯一的 id，且没有问题。

269.3. 配置 QUARTZ.PROPERTIES 文件

默认情况下，Quartz 将在 classpath 的 org/quartz 目录中查找 quartz.properties 文件。如果您使用 WAR 部署，这意味着只丢弃 WEB-INF/classes/org/quartz 中的 quartz.properties。

但是，Camel Quartz 组件还允许您配置属性：

参数	默认值	类型	描述
属性	**null**	**Properties**	Camel 2.4 ：您可以配置 **java.util.Properties** 实例。
propertiesFile	**null**	**字符串**	Camel 2.4: 从 classpath 加载的属性的文件名

要做到这一点，您可以在 Spring XML 中配置它，如下所示

```
<bean id="quartz" class="org.apache.camel.component.quartz.QuartzComponent">
    <property name="propertiesFile" value="com/mycompany/myquartz.properties"/>
</bean>
```

269.4. 在 JMX 中启用 QUARTZ 调度程序

您需要配置 quartz 调度程序属性来启用 JMX。
这通常将选项 "org.quartz.scheduler.jmx.export" 设置为配置文件中的 true 值。

从 Camel 2.13 开始，Camel 会自动将此选项设置为 true，除非明确禁用。

269.5. 启动 QUARTZ 调度程序

这是一个示例：

```
<bean id="quartz" class="org.apache.camel.component.quartz.QuartzComponent">
    <property name="startDelayedSeconds" value="5"/>
</bean>
```

269.6. 集群

从 Camel 2.4 开始提供

如果您在集群模式中使用 Quartz，如 JobStore 集群。然后，在 Quartz 组件的 Camel 2.4 中，当节点停止/shutdown 时，不会 暂停/删除触发器。这允许触发器在集群中的其他节点上保持运行。

注意 ：在集群节点中运行时，不会检查以确保端点的唯一作业名称/组。

269.7. 消息标头

Camel 将 Quartz Execution Context 中的 getters 添加为标头值。以下标头会被添加：
calendar, fireTime, jobDetail, jobInstance, jobRuntTime, mergedJobDataMap, nextFireTime, previousFireTime, refireCount, result, scheduledFireTime, scheduler, trigger, triggerName, triggerGroup.

fireTime 标头包含触发交换时的 java.util.Date。

269.8. 使用 CRON TRIGGERS

quartz 支持与 Cron 类似的表达式，以手写格式指定计时器。您可以在 cron URI 参数中使用这些表达

式，但要保留有效的 URI 编码，我们允许使用 + 而不是空格。quartz 提供了有关如何使用 cron 表达式的小教程。

例如，以下内容每五分钟触发一条消息，从早上 12pm (noon)开始，在工作日的 6pm 开始：

```
from("quartz://myGroup/myTimerName?cron=0+0/5+12-18+?+*+MON-FRI").to("activemq:Totally.Rocks");
```

这等同于使用 cron 表达式

```
0 0/5 12-18 ? * MON-FRI
```

下表显示了我们用来保留有效 URI 语法的 URI 字符编码：

URI Character	Cron 字符
+	space

269.9. 指定时区

从 Camel 2.8.1 开始，您可以为每个触发器配置时区。例如，要使用您所在国家的时区，您可以按如下方式执行：

```
quartz://groupName/timerName?cron=0+0/5+12-18+?+*+MON-FRI&trigger.timeZone=Europe/Stockholm
```

timeZone 值是 java.util.TimeZone 接受的值。

在 Camel 2.8.0 或旧版本中，您必须将自定义 String 提供给 java.util.TimeZone Type Converter，以便从端点 uri 中进行配置。
从 Camel 2.8.1 开始，我们已在 camel-core 中包含此类类型转换器。

269.10. 另请参阅

- 配置 Camel

- *组件*
- *端点*
- *开始使用*
- *Quartz2*
- *计时器*

第 270 章 QUARTZ2 组件

从 Camel 版本 2.12 开始提供

quartz2： 组件使用 Quartz Scheduler 2.x 提供预定的消息发送。
每个端点代表不同的计时器（在 Quartz 术语， 一个 Trigger 和 JobDetail）。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-quartz2</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

注意： Quartz 2.x API 与 Quartz 1.x 不兼容。如果您需要保留旧的 Quartz 1.x，
使用旧的 Quartz 组件。

270.1. URI 格式

```
quartz2://timerName?options
quartz2://groupName/timerName?options
quartz2://groupName/timerName?cron=expression
quartz2://timerName?cron=expression
```

组件使用 CronTrigger 或 SimpleTrigger。如果没有提供 cron 表达式，则组件将使用一个简单的触发器。如果没有提供 groupName， quartz 组件将使用 Camel 组名称。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

270.2. 选项

Quartz2 组件支持 11 个选项，如下所列。

Name	描述	默认值	类型
autoStartScheduler (scheduler)	调度程序是否应自动启动。这个选项是默认的 true	true	布尔值
startDelayedSeconds (scheduler)	启动 quartz 调度程序前等待的秒数。		int
prefixJobNameWith EndpointId (consumer)	是否使用端点 ID 为 quartz 作业添加前缀。这个选项是默认的 false。	false	布尔值
enableJmx (consumer)	是否启用 Quartz JMX，允许从 JMX 管理 Quartz 调度程序。这个选项是默认的 true	true	布尔值
properties (consumer)	配置 Quartz 调度程序的属性。		Properties
propertiesFile (consumer)	从 classpath 加载的属性的文件名		字符串
prefixInstanceName (consumer)	是否使用 CamelContext 名称为 Quartz Scheduler 实例名称添加前缀。这默认是启用的，让每个 CamelContext 默认使用自己的 Quartz 调度程序实例。您可以将这个选项设置为 false，以在多个 CamelContext 之间重复使用 Quartz 调度程序实例。	true	布尔值
interruptJobsOn Shutdown (scheduler)	是否在关闭时中断作业，它会强制调度程序更快地关闭，并尝试中断任何正在运行的作业。如果启用了，则任何正在运行的作业都可能会因为中断而失败。	false	布尔值
schedulerFactory (advanced)	使用用于创建调度程序的自定义 SchedulerFactory。		SchedulerFactory
scheduler (advanced)	使用自定义配置的 Quartz 调度程序，而不是创建新的调度程序。		scheduler
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Quartz2 端点使用 URI 语法进行配置：

```
quartz2:groupName/triggerName
```

使用以下路径和查询参数：

270.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
groupName	要使用的 quartz 组名称。组名称和计时器名称的组合应该是唯一的。	Camel	字符串
triggerName	**必需** 要使用的 quartz 计时器名称。组名称和计时器名称的组合应该是唯一的。		字符串

270.2.2. 查询参数(19 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
cron (consumer)	指定定义要触发的时间的 cron 表达式。		字符串
deleteJob (consumer)	如果设置为 true，则触发器会在路由停止时自动删除。其他设为 false，它将保留在调度程序中。当设置为 false 时，它还意味着用户可以使用 camel Uri 重复使用预先配置的触发器。只需确保名称匹配。请注意，您无法同时将 deleteJob 和 pauseJob 设置为 true。	true	布尔值
durableJob (consumer)	作业在孤立后是否应保留存储（没有触发器指向它）。	false	布尔值
pauseJob (consumer)	如果设置为 true，则触发器会在路由停止时自动暂停。其他设为 false，它将保留在调度程序中。当设置为 false 时，它还意味着用户可以使用 camel Uri 重复使用预先配置的触发器。只需确保名称匹配。请注意，您无法同时将 deleteJob 和 pauseJob 设置为 true。	false	布尔值
recoverableJob (consumer)	如果遇到 'recovery' 或 'fail-over' 情况，则指示调度程序是否应重新执行该作业。	false	布尔值
stateful (consumer)	使用 Quartz PersistJobDataAfterExecution 和 DisallowConcurrentExecution 而不是默认作业。	false	布尔值

Name	描述	默认值	类型
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
customCalendar (advanced)	指定自定义日历以避免特定日期范围		日历
jobParameters (advanced)	配置作业上的附加选项。		Map
prefixJobNameWithEndpoint Id (advanced)	作业名称是否应以端点 ID 前缀	false	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
triggerParameters (advanced)	要在触发器中配置附加选项：		Map
usingFixedCamelContextName (advanced)	如果为 true，JobDataMap 直接使用 CamelContext 名称来引用 CamelContext，如果为 false，则 JobDataMap 使用 CamelContext 管理名称，可在部署期间更改。	false	布尔值
autoStartScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
fireNow (scheduler)	如果为 true，在使用 SimpleTrigger 时路由启动时将触发触发器。	false	布尔值
startDelayedSeconds (scheduler)	启动 quartz 调度程序前等待的秒数。		int
triggerStartDelay (scheduler)	如果调度程序已经启动，我们希望在当前时间后触发器启动稍有变化，以确保端点在作业启动之前完全启动。	500	long

270.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 12 个选项，如下所列。

Name	描述	默认值	类型
camel.component.quartz2.auto-start-scheduler	调度程序是否应自动启动。这个选项是默认的 true	true	布尔值
camel.component.quartz2.enable-jmx	是否启用 Quartz JMX，允许从 JMX 管理 Quartz 调度程序。这个选项是默认的 true	true	布尔值
camel.component.quartz2.enabled	启用 quartz2 组件	true	布尔值
camel.component.quartz2.interrupt-jobs-on-shutdown	是否在关闭时中断作业，它会强制调度程序更快地关闭，并尝试中断任何正在运行的作业。如果启用了，则任何正在运行的作业都可能会因为中断而失败。	false	布尔值
camel.component.quartz2.prefix-instance-name	是否使用 CamelContext 名称为 Quartz Scheduler 实例名称添加前缀。这默认是启用的，让每个 CamelContext 默认使用自己的 Quartz 调度程序实例。您可以将这个选项设置为 false，以在多个 CamelContext 之间重复使用 Quartz 调度程序实例。	true	布尔值
camel.component.quartz2.prefix-job-name-with-endpoint-id	是否使用端点 ID 为 quartz 作业添加前缀。这个选项是默认的 false。	false	布尔值
camel.component.quartz2.properties	配置 Quartz 调度程序的属性。选项是一个 java.util.Properties 类型。		字符串
camel.component.quartz2.properties-file	从 classpath 加载的属性的文件名		字符串
camel.component.quartz2.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.quartz2.scheduler	使用自定义配置的 Quartz 调度程序，而不是创建新的调度程序。选项是 org.quartz.Scheduler 类型。		字符串
camel.component.quartz2.scheduler-factory	使用用于创建调度程序的自定义 SchedulerFactory。选项是 org.quartz.SchedulerFactory 类型。		字符串

Name	描述	默认值	类型
camel.component.quartz2.start-delayed-seconds	启动 quartz 调度程序前等待的秒数。		整数

例如，以下路由规则将触发两个计时器事件到 mock:results 端点：

```
from("quartz2://myGroup/myTimerName?trigger.repeatInterval=2&trigger.repeatCount=1").routeId("myRoute")
    .to("mock:result");
```

使用 stateful=true 时，JobDataMap 每次执行作业后都会重新具有持久性，从而为下一次执行保留状态。

INFO：在 OSGi 中运行并使用 quartz 路由的多个捆绑包，如果您运行在 OSGi 中，如 Apache ServiceMix 或 Apache Karaf，并且具有从 Quartz2 端点开始的 Camel 路由的多个捆绑包，然后请确保如果您为这个 id 的 <camelContext> 分配了一个 ID，因为 OSGi 容器中 QuartzScheduler 需要它。如果您没有在 <camelContext> 上设置任何 id，则会自动分配唯一的 id，且没有问题。

270.4. 配置 QUARTZ.PROPERTIES 文件

默认情况下，Quartz 将在 classpath 的 org/quartz 目录中查找 quartz.properties 文件。如果您使用 WAR 部署，这意味着只丢弃 WEB-INF/classes/org/quartz 中的 quartz.properties。

但是，Camel Quartz2 组件还允许您配置属性：

参数	默认值	类型	描述
属性	**null**	**Properties**	您可以配置 **java.util.Properties** 实例。
propertiesFile	**null**	**字符串**	从 classpath 加载的属性的文件名

要做到这一点，您可以在 Spring XML 中配置它，如下所示

```
<bean id="quartz2" class="org.apache.camel.component.quartz2.QuartzComponent">
    <property name="propertiesFile" value="com/mycompany/myquartz.properties"/>
</bean>
```

270.5. 在 JMX 中启用 QUARTZ 调度程序

您需要配置 quartz 调度程序属性来启用 JMX。
这通常将选项 "org.quartz.scheduler.jmx.export" 设置为配置文件中的 true 值。

从 Camel 2.13 开始，Camel 会自动将此选项设置为 true，除非明确禁用。

270.6. 启动 QUARTZ 调度程序

Quartz2 组件提供了一个使 Quartz 调度程序启动延迟或根本不自动启动的选项。

这是一个示例：

```
<bean id="quartz2" class="org.apache.camel.component.quartz2.QuartzComponent">
    <property name="startDelayedSeconds" value="5"/>
</bean>
```

270.7. 集群

如果您在集群模式中使用 Quartz，如 JobStore 集群。然后，当节点停止/关闭时，Qartz 2 组件 不会暂停/删除触发器。这允许触发器在集群中的其他节点上保持运行。

注意 ： 在集群节点中运行时，不会检查以确保端点的唯一作业名称/组。

270.8. 消息标头

Camel 将 Quartz Execution Context 中的 getters 添加为标头值。以下标头会被添加：
calendar, fireTime, jobDetail, jobInstance, jobRuntTime, mergedJobDataMap, nextFireTime, previousFireTime, refireCount, result, scheduledFireTime, scheduler, trigger, triggerName, triggerGroup.

fireTime 标头包含触发交换时的 java.util.Date。

270.9. 使用 CRON TRIGGERS

quartz 支持与 Cron 类似的表达式，以手写格式指定计时器。您可以在 cron URI 参数中使用这些表达式，但要保留有效的 URI 编码，我们允许使用 + 而不是空格。

例如，以下内容每五分钟触发一条消息，从早上 12pm (noon)开始，在工作日的 6pm 开始：

```
from("quartz2://myGroup/myTimerName?cron=0+0/5+12-18+?+*+MON-FRI")
    .to("activemq:Totally.Rocks");
```

这等同于使用 cron 表达式

```
0 0/5 12-18 ? * MON-FRI
```

下表显示了我们用来保留有效 URI 语法的 URI 字符编码：

URI Character	Cron 字符
+	space

270.10. 指定时区

Quartz 调度程序允许您为每个触发器配置时区。例如，要使用您所在国家的时区，您可以按如下方式执行：

```
quartz2://groupName/timerName?cron=0+0/5+12-18+?+*+MON-
FRI&trigger.timeZone=Europe/Stockholm
```

timeZone 值是 java.util.TimeZone 接受的值。

270.11. 使用 QUARTZSCHEDULEDPOLLCONSUMERSCHEDULER

Quartz2 组件提供了一个 Polling Consumer 调度程序，它允许将基于 cron 的调度用于 Polling Consumer，如 File 和 FTP 用户。

例如，要使用基于 cron 的表达式每第 2 秒轮询一次文件，可将 Camel 路由定义为：

```
from("file:inbox?scheduler=quartz2&scheduler.cron=0/2+*+*+*+*+?")
  .to("bean:process");
```

请注意，我们定义 scheduler=quartz2，以指示 Camel 使用基于 Quartz2 的调度程序。然后，我们使用 scheduler.xxx 选项来配置调度程序。Quartz2 调度程序需要设置 cron 选项。

支持以下选项：

参数	默认值	类型	描述
quartzScheduler	null	org.quartz.Scheduler	使用自定义 Quartz 调度程序。如果没有配置，则使用 Quartz2 组件的共享调度程序。
cron	null	字符串	**强制**：定义用于触发轮询的 cron 表达式。
triggerId	null	字符串	指定触发器 ID：如果没有提供，则生成并使用 UUID。
triggerGroup	QuartzScheduledPollConsumerScheduler	字符串	指定触发器组。
timeZone	默认值	TimeZone	用于 CRON 触发器的时区。

重要：从端点 URI 配置这些选项必须以 调度程序前缀。例如，配置触发器 id 和 group：

```
from("file:inbox?scheduler=quartz2&scheduler.cron=0/2+*+*+*+*+?
&scheduler.triggerId=myId&scheduler.triggerGroup=myGroup")
  .to("bean:process");
```

Spring 中还有一个 CRON 调度程序，因此您也可以使用以下内容：

```
from("file:inbox?scheduler=spring&scheduler.cron=0/2+*+*+*+*+?")
  .to("bean:process");
```

第 271 章 QUICKFIX 组件

从 Camel 版本 2.1 开始提供

快速修复 组件调整了在 Camel 中使用的 QuickFIX/J FIX 引擎。此组件使用标准特性 交换(FIX)协议进行 消息传输。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-quickfix</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

271.1. URI 格式

```
quickfix:configFile[?sessionID=sessionID&lazyCreateEngine=true|false]
```

configFile 是用于 FIX 引擎的 QuickFIX/J 配置的名称（在您的 classpath 中找到的资源）。可选的 sessionID 标识特定的 FIX 会话。sessionID 的格式是：

```
(BeginString):(SenderCompID)[/(SenderSubID)[/(SenderLocationID)]]->(TargetCompID)
[/(TargetSubID)[/(TargetLocationID)]]
```

可选的 lazyCreateEngine (Camel 2.12.3+)参数允许按需创建 QuickFIX/J 引擎。值 true 意味着当发送第一个消息或路由定义中配置消费者时，引擎将启动。使用 false 值时，引擎会在创建端点时启动。当缺少此参数时，会使用组件属性 lazyCreateEngines 的值。

URI 示例：

```
quickfix:config.cfg

quickfix:config.cfg?sessionID=FIX.4.2:MyTradingCompany->SomeExchange

quickfix:config.cfg?sessionID=FIX.4.2:MyTradingCompany->SomeExchange&lazyCreateEngine=true
```

271.2. ENDPOINTS

FIX 会话是 快速修复 组件的端点。端点 URI 可以指定单个会话，或者由特定 QuickFIX/J 引擎管理的所有会话。典型的应用程序将只使用一个 FIX 引擎，但高级用户可以通过引用 快速修复 组件端点 URI 中的不同配置文件来创建多个 FIX 引擎。

当消费者在端点 URI 中不包含会话 ID 时，它将接收由 URI 中指定的配置文件所管理的所有会话的交换。如果生成者没有在端点 URI 中指定会话，则必须在要发送的 FIX 消息中包含与会话相关的字段。如果在 URI 中指定会话，则组件将自动将与会话相关的字段注入 FIX 消息。

271.3. 选项

QuickFix 组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
messageFactory (advanced)	使用给定的 MessageFactory		MessageFactory
logFactory (advanced)	使用给定的 LogFactory		LogFactory
messageStoreFactory (advanced)	使用给定的 MessageStoreFactory		MessageStoreFactory
configuration (common)	使用给定配置的 QuickFix 配置映射到键的映射		Map
lazyCreateEngines (common)	如果设置为 true，则在需要时将创建和启动引擎（发送第一个消息时）	false	布尔值
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

QuickFix 端点使用 URI 语法进行配置：

```
quickfix:configurationName
```

使用以下路径和查询参数：

271.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
configurationName	**必需** The configFile 是用于 FIX 引擎的 QuickFIX/J 配置的名称（在您的 classpath 中找到的资源）。		字符串

271.3.2. 查询参数(6 参数)：

Name	描述	默认值	类型
lazyCreateEngine (common)	这个选项允许按需创建 QuickFIX/J 引擎。值 true 意味着当发送第一个消息或路由定义中配置消费者时，引擎将启动。使用 false 值时，引擎会在创建端点时启动。当缺少此参数时，会使用组件属性 lazyCreateEngines 的值。	false	布尔值
sessionID (common)	可选的 sessionID 标识特定的 FIX 会话。sessionID 的格式是：(BeginString):(SenderCompID)/(SenderSubID)/(SenderLocationID)-(TargetCompID)/(TargetSubID)/(TargetLocationID)		SessionID
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

271.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 7 个选项，如下所列。

Name	描述	默认值	类型
camel.component.quickfix.configurations	使用给定配置的 QuickFix 配置映射到键的映射		Map
camel.component.quickfix.enabled	启用快速修复组件	true	布尔值
camel.component.quickfix.lazy-create-engines	如果设置为 true，则在需要时将创建和启动引擎（发送第一个消息时）	false	布尔值
camel.component.quickfix.log-factory	使用给定的 LogFactory。选项是一个 quickfix.LogFactory 类型。		字符串
camel.component.quickfix.message-factory	使用给定的 MessageFactory。选项是一个 quickfix.MessageFactory 类型。		字符串
camel.component.quickfix.message-store-factory	使用给定的 MessageStoreFactory。选项是一个 quickfix.MessageStoreFactory 类型。		字符串
camel.component.quickfix.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

271.5. EXCHANGE FORMAT

交换标头包括有助于交换过滤、路由和其他处理的信息。可用的标头如下：

标头名称	描述
EventCategory	**AppMessageReceived**,**AppMessageSent**,**AdminMessageReceived**,**AdminMessageSent**,**SessionCreated**,**SessionLogon**,**SessionLogoff**.请参阅 **QuickfixjEventCategory** enum。
SessionID	FIX 消息 SessionID
MessageType	FIX MsgType 标签值

标头名称	描述
DataDictionary	指定用于解析传入消息的数据字典。可以是数据字典实例，也可以是 QuickFIX/J 数据字典文件的资源路径

如果收到字符串消息，且需要在路由中解析字符串消息，DataDictionary 标头很有用。QuickFIX/J 需要一个数据字典来解析某些类型的消息（例如，重复组）。在收到消息字符串后，FIX 引擎可以正确地解析数据，在路由中注入 DataDictionary 标头。

271.6. QUICKFIX/J 配置扩展

当直接使用 QuickFIX/J 时，通常会写入代码来创建日志记录适配器实例、消息存储和通信连接器。quickfix 组件将根据配置文件中的信息自动创建这些类的实例。它还提供了许多常见必要设置的默认值，并添加其他功能（例如激活 JMX 支持）。

以下小节描述了 快速修复 组件如何处理 QuickFIX/J 配置。有关 QuickFIX/J 配置的综合信息，请参阅 QFJ 用户手册。

271.6.1. 通信连接器

当组件检测到 QuickFIX/J 配置文件中的 initiator 或 acceptor 会话设置时，它将自动创建对应的启动器和/或接收器连接器。这些设置可以位于 default 或配置文件的特定会话部分中。

会话设置	组件操作
CONNECTIONTYPE=initiator	创建 initiator 连接器
CONNECTIONTYPE=acceptor	创建接受者连接器

也可以指定 QuickFIX/J 会话连接器的线程模型。这些设置会影响配置文件中的所有会话，并必须放在设置 default 部分中。

默认/全球设置	组件操作
ThreadModel=ThreadPerConnector	使用 **SocketInitiator** 或 **SocketAcceptor** （默认）
ThreadModel=ThreadPerSession	使用 **ThreadedSocketInitiator** 或 **ThreadedSocketAcceptor**

271.6.2. 日志记录

***QuickFIX/J logger** 实现可以通过在配置文件的 **default** 部分中包含以下设置来指定：如果配置中不存在以下设置，则 **ScreenLog** 是默认设置。这是包含表示多个日志实施的设置的错误。日志工厂实现也可以直接在 **Quickfix** 组件上设置。这将覆盖 **QuickFIX/J** 设置文件中任何相关值。*

默认/全球设置	组件操作
ScreenLogShowEvents	使用 **ScreenLog**
ScreenLogShowIncoming	使用 **ScreenLog**
ScreenLogShowOutgoing	使用 **ScreenLog**
SLF4J*	Camel 2.6+.使用 **SLF4JLog**。任何 SLF4J 设置都将使用此日志。

默认/全球设置	组件操作
FileLogPath	使用 **FileLog**
JdbcDriver	使用 **JdbcLog**

271.6.3. 消息存储

QuickFIX/J 消息存储实现可以通过在配置文件 default 部分包含以下设置来指定。如果配置中不存在以下设置，则 MemoryStore 是默认设置。这是包含表示多个消息存储实施的设置的错误。消息存储工厂也可以直接在 Quickfix 组件上设置。这将覆盖 QuickFIX/J 设置文件中任何相关值。

默认/全球设置	组件操作
JdbcDriver	使用 **JdbcStore**
FileStorePath	使用 **FileStore**
SleepycatDatabaseDir	使用 **SleepcatStore**

271.6.4. Message Factory

消息工厂用于从原始 FIX 消息构造域对象。默认消息工厂是 DefaultMessageFactory。但是，高级应用程序可能需要自定义消息工厂。这可以在 QuickFIX/J 组件上设置。

271.6.5. JMX

默认/全球设置	组件操作

默认/全球设置	组件操作
UseJmx	如果为 **Y**，则启用 QuickFIX/J JMX

271.6.6. 其他默认值

组件为 QuickFIX/J 配置文件中通常需要的设置提供了一些默认设置。SessionStartTime 和 SessionEndTime 默认为 "00:00:00"，这意味着会话不会自动启动和停止。HeartBtInt (heartbeat 间隔) 默认为 30 秒。

271.6.7. 最小启动器配置示例

```
[SESSION]
ConnectionType=initiator
BeginString=FIX.4.4
SenderCompID=YOUR_SENDER
TargetCompID=YOUR_TARGET
```

271.7. 使用 INOUT 消息交换模式

Camel 2.8+

虽然 FIX 协议是事件驱动的和异步的，但有代表请求回复消息交换的特定对消息。要使用 InOut Exchange 模式，应该有一个请求消息，以及对请求的单一回复消息。示例包括 OrderStatusRequest message 和 UserRequest。

271.7.1. 为消费者实施 InOut Exchanges

将 "exchangePattern=InOut" 添加到 QuickFIX/J enpoint URI。以下示例中的 MessageOrderStatusService 是带有同步服务方法的 bean。该方法返回响应请求（本例中为 ExecutionReport），然后发回到 requestor 会话。

```
    from("quickfix:examples/inprocess.cfg?sessionID=FIX.4.2:MARKET-
>TRADER&exchangePattern=InOut")

.filter(header(QuickfixjEndpoint.MESSAGE_TYPE_KEY).isEqualTo(MsgType.ORDER_STATUS
_REQUEST))
        .bean(new MarketOrderStatusService());
```

271.7.2. 为生成实施 InOut Exchanges

对于生产者，发送消息将阻止到收到回复或超时为止。在 FIX 中无法关联回复消息的标准方法。因此，必须为每种类型的 InOut 交换定义一个关联标准。可以使用 Exchange 属性来指定关联条件和超时。

描述	键字符串	关键的 Constant	default
关联标准	"CorrelationCriteria"	QuickfixjProducer.CORRELATION_CRITERIA_KEY	None
在 Milliseconds 中关联超时	"CorrelationTimeout"	QuickfixjProducer.CORRELATION_TIMEOUT_KEY	1000

关联条件与 MessagePredicate 对象定义。以下示例将处理指定会话中的 FIX ExecutionReport，其中事务类型是 STATUS，Order ID 与我们的请求匹配。会话 ID 应该用于 请求者，在查找回复时，发送者和目标 CompID 字段将被撤销。

```
exchange.setProperty(QuickfixjProducer.CORRELATION_CRITERIA_KEY,
    new MessagePredicate(new SessionID(sessionID), MsgType.EXECUTION_REPORT)
        .withField(ExecTransType.FIELD, Integer.toString(ExecTransType.STATUS))
        .withField(OrderID.FIELD, request.getString(OrderID.FIELD)));
```

271.7.3. Example

源代码包含一个名为 RequestReplyExample 的示例，它演示了消费者和制作者的 InOut 交换。本例创建一个接受订购状态请求的简单 HTTP 服务器端点。HTTP 请求转换为 FIX OrderStatusRequestMessage，以关联标准增强，然后路由到快速修复端点。然后，响应将转换为 JSON 格式的字符串，并发回到作为 Web 响应来提供的 HTTP 服务器端点。

Spring 配置已从 Camel 2.9 开始。请参阅以下示例。

271.8. SPRING 配置

Camel 2.6 - 2.8.x

QuickFIX/J 组件包含一个 Spring FactoryBean，用于在 Spring 上下文中配置会话设置。也包括了 QuickFIX/J 会话 ID 字符串的类型转换器。以下示例显示了一个 acceptor 和 initiator 会话的简单配置，

以及两个会话的默认设置。

```
<!-- camel route -->
<camelContext xmlns="http://camel.apache.org/schema/spring" id="quickfixjContext">
    <route>
        <from uri="quickfix:example"/>
        <filter>
            <simple>${in.header.EventCategory} == 'AppMessageReceived'</simple>
            <to uri="log:test"/>
        </filter>
    </route>
</camelContext>
        <!-- quickfix component -->
<bean id="quickfix" class="org.apache.camel.component.quickfixj.QuickfixjComponent">
<property name="engineSettings">
    <util:map>
        <entry key="quickfix:example" value-ref="quickfixjSettings"/>
    </util:map>
</property>
<property name="messageFactory">
    <bean
class="org.apache.camel.component.quickfixj.QuickfixjSpringTest.CustomMessageFactory"/>
</property>
</bean>
        <!-- quickfix settings -->
<bean id="quickfixjSettings" class="org.apache.camel.component.quickfixj.QuickfixjSettingsFactory">
<property name="defaultSettings">
    <util:map>
        <entry key="SocketConnectProtocol" value="VM_PIPE"/>
        <entry key="SocketAcceptProtocol" value="VM_PIPE"/>
        <entry key="UseDataDictionary" value="N"/>
    </util:map>
</property>
<property name="sessionSettings">
    <util:map>
        <entry key="FIX.4.2:INITIATOR->ACCEPTOR">
            <util:map>
                <entry key="ConnectionType" value="initiator"/>
                <entry key="SocketConnectHost" value="localhost"/>
                <entry key="SocketConnectPort" value="5000"/>
            </util:map>
        </entry>
        <entry key="FIX.4.2:ACCEPTOR->INITIATOR">
            <util:map>
                <entry key="ConnectionType" value="acceptor"/>
                <entry key="SocketAcceptPort" value="5000"/>
            </util:map>
        </entry>
    </util:map>
</property>
</bean>
```

Camel 2.9 以后

QuickFIX/J 组件包括用于配置会话设置的 QuickfixjConfiguration 类。也包括了 QuickFIX/J 会话 ID 字符串的类型转换器。以下示例显示了一个 acceptor 和 initiator 会话的简单配置，以及两个会话的默认设置。

```
<!-- camel route -->
<camelContext id="quickfixjContext" xmlns="http://camel.apache.org/schema/spring">
    <route>
        <from uri="quickfix:example"/>
        <filter>
            <simple>${in.header.EventCategory} == 'AppMessageReceived'</simple>
            <to uri="log:test"/>
        </filter>
    </route>
    <route>
        <from uri="vm:test"/>
        <to uri="lazyQuickfix:example"/>
    </route>
</camelContext>

<!-- quickfix component -->
<bean id="quickfix" class="org.apache.camel.component.quickfixj.QuickfixjComponent">
    <property name="configurations">
        <util:map>
            <entry key="example" value-ref="quickfixjConfiguration"/>
        </util:map>
    </property>
    <property name="messageFactory">
        <bean
class="org.apache.camel.component.quickfixj.QuickfixjSpringTest.CustomMessageFactory"/>
    </property>
</bean>

<!-- lazy quickfix component -->
<bean id="lazyQuickfix" class="org.apache.camel.component.quickfixj.QuickfixjComponent">
    <property name="lazyCreateEngines" value="true" />
    <property name="configurations">
        <util:map>
            <entry key="example" value-ref="lazyQuickfixjConfiguration"/>
        </util:map>
    </property>
    <property name="messageFactory">
        <bean
class="org.apache.camel.component.quickfixj.QuickfixjSpringTest.CustomMessageFactory"/>
    </property>
</bean>

<!-- quickfix settings -->
<bean id="quickfixjConfiguration"
class="org.apache.camel.component.quickfixj.QuickfixjConfiguration">
    <property name="defaultSettings">
        <util:map>
            <entry key="SocketConnectProtocol" value="VM_PIPE"/>
            <entry key="SocketAcceptProtocol" value="VM_PIPE"/>
            <entry key="UseDataDictionary" value="N"/>
```

```
        </util:map>
    </property>
    <property name="sessionSettings">
        <util:map>
            <entry key="FIX.4.2:INITIATOR->ACCEPTOR">
                <util:map>
                    <entry key="ConnectionType" value="initiator"/>
                    <entry key="SocketConnectHost" value="localhost"/>
                    <entry key="SocketConnectPort" value="5000"/>
                </util:map>
            </entry>
            <entry key="FIX.4.2:ACCEPTOR->INITIATOR">
                <util:map>
                    <entry key="ConnectionType" value="acceptor"/>
                    <entry key="SocketAcceptPort" value="5000"/>
                </util:map>
            </entry>
        </util:map>
    </property>
</bean>
```

271.9. 异常处理

如果在处理消息的过程中抛出某些异常，可以修改 QuickFIX/J 行为。如果在处理传入的登录管理消息时抛出 RejectLogon 异常，则登录将被拒绝。

通常，QuickFIX/J 会自动处理登录过程。但是，有时必须修改传出日志消息，使其包含 FIX 计数器所需的凭据。如果在发送登录消息时修改 FIX log-in 消息正文(EventCategory=AdminMessageSent，修改的消息将发送到计数器)。重要的是，传出的登录消息是 同步 处理的。如果异步处理（在另一个线程中），FIX 引擎将在回调方法返回时立即发送未修改的传出消息。

271.10. FIX 序列号管理

如果在 同步 交换处理过程中抛出应用程序异常，这会导致 QuickFIX/J 不会递增传入的 FIX 消息序列号，并可能导致计数器消息的重新发送。此 FIX 协议行为主要用于处理 传输 错误，而不是应用程序错误。使用这个机制来处理应用程序错误的风险。主要风险是，消息会在每次重新收到时重复导致应用程序错误。更好的解决方法是，在处理前立即保留传入的消息（数据库、JMS 队列）。这也允许应用异步处理消息，而不会在发生错误时丢失消息。

虽然可以在登录前将消息发送到 FIX 会话（消息在日志中发送），但通常最好等待会话登录。这消除了登录时所需的序列号重新同步步骤。可以通过设置处理 SessionLogon 事件类别的路由并给应用程序发送消息，等待会话日志。

有关 FIX 序列号管理的详情，请查看 FIX 协议规格和 QuickFIX/J 文档。

271.11. 路由示例

QuickFIX/J 组件源代码（测试子目录）中包括几个示例。这些示例之一实施了三倍的交易模拟。这个示例定义了一个应用程序组件，它使用 URI 方案 "trade-executor"。

以下路由接收了交易执行器会话的消息，并将应用消息传递给 executor 组件。

```
from("quickfix:examples/inprocess.cfg?sessionID=FIX.4.2:MARKET->TRADER").

filter(header(QuickfixjEndpoint.EVENT_CATEGORY_KEY).isEqualTo(QuickfixjEventCategory.
AppMessageReceived)).
    to("trade-executor:market");
```

交易执行器组件会生成回交易会话的消息。会话 ID 必须在 FIX 消息本身中设置，因为端点 URI 中没有指定会话 ID。

```
from("trade-executor:market").to("quickfix:examples/inprocess.cfg");
```

交易器会话消耗了来自市场的执行报告信息并处理它们。

```
from("quickfix:examples/inprocess.cfg?sessionID=FIX.4.2:TRADER->MARKET").

filter(header(QuickfixjEndpoint.MESSAGE_TYPE_KEY).isEqualTo(MsgType.EXECUTION_REP
ORT)).
    bean(new MyTradeExecutionProcessor());
```

271.12. CAMEL 2.5 的 QUICKFIX/J 组件优先级

quickfix 组件是 Java 的 QuickFIX/J 引擎的实现。此引擎允许连接到 FIX 服务器，该服务器用于根据 FIX 协议 标准交换财务消息。

注意： 组件可用于向 FIX 服务器发送/接收消息。

271.13. URI 格式

```
quickfix-server:config file
quickfix-client:config file
```

其中 配置文件是用来在启动时配置引擎的快速修复配置文件的位置（在您的类路径中）。

注：有关快速修复参数的信息，请参阅 QuickFIX/J 网站。

如果您要发送消息到 FIX 网关时，必须使用 quickfix-server 端点来接收来自 FIX 服务器 FIX 消息和 quickfix-client 端点。

271.14. 交换数据格式

QuickFIX/J 引擎类似于 CXF 组件使用 MINA 作为协议层的消息传递总线，以使用 FIX 引擎网关创建套接字连接。

当 QuickFIX/J 引擎收到消息时，它会创建一个由 camel 端点接收的 QuickFix.Message 实例。此对象是从 FIX 消息格式化的"映射对象"创建，最初为键值对数据集合。您可以使用此对象，也可以使用方法 'toString' 来检索原始的 FIX 消息。

注：另外，您可以使用 camel bindy dataformat 将 FIX 消息转换为您自己的 java POJO

当必须向 QuickFix 发送消息时，您必须创建一个 QuickFix.Message 实例。

271.15. LAZY 创建引擎

从 Camel 2.12.3 开始，您可以将 QuickFixComponent 配置为 lazy 创建并启动引擎，然后仅启动这些按需。例如，当您在带有 master/slaves 的集群中有多个 Camel 应用程序时，您可以使用它。并且希望从设备处于备用状态。

271.16. SAMPLES

方向：到 FIX 网关

```
<route>
  <from uri="activemq:queue:fix"/>
  <bean ref="fixService" method="createFixMessage"/> // bean method in charge to transform message into a QuickFix.Message
  <to uri="quickfix-client:META-INF/quickfix/client.cfg"/> // Quickfix engine who will send the FIX messages to the gateway
</route>
```

方向：来自 FIX 网关

```
<route>
  <from uri="quickfix-server:META-INF/quickfix/server.cfg"/> // QuickFix engine who will receive the message from FIX gateway
  <bean ref="fixService" method="parseFixMessage"/> // bean method parsing the QuickFix.Message
  <to uri="uri="activemq:queue:fix"/>"
</route>
```

271.17. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 272 章 RABBITMQ 组件

从 Camel 版本 2.12 开始提供

rabbitmq: 组件允许您从 RabbitMQ 实例生成和使用消息。通过使用 RabbitMQ AMQP 客户端，此组件通过通用 AMQP 组件提供纯 RabbitMQ 方法。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-rabbitmq</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

272.1. URI 格式

旧语法 已弃用 ：

```
rabbitmq://hostname[:port]/exchangeName?[options]
```

相反，主机名和端口在组件级别上配置，或者作为 uri 查询参数提供。

新语法为：

```
rabbitmq:exchangeName?[options]
```

其中 hostname 是正在运行的 rabbitmq 实例或集群的主机名。port 是可选的；如果未指定，则默认为 RabbitMQ 客户端默认值(5672)。交换名称决定了生成的消息将发送到哪些交换。如果是消费者，交换名称决定了队列将绑定到的交换。

272.2. 选项

RabbitMQ 组件支持 50 个选项，如下所列。

Name	描述	默认值	类型
hostname (common)	运行 RabbitMQ 实例或集群的主机名。		字符串
portnumber (common)	带有正在运行的 rabbitmq 实例或集群的主机的端口号。	5672	int
用户名 (security)	身份验证访问时的用户名	Guest	字符串
密码 (security)	验证访问的密码	Guest	字符串
vhost (common)	频道的 vhost	/	字符串
addresses (common)	如果设置了这个选项，camel-rabbitmq 将尝试根据选项地址的设置来创建连接。addresses 值是一个字符串，它类似于 server1:12345, server2:12345		字符串
ConnectionFactory (common)	使用自定义 RabbitMQ 连接工厂。当设定这个选项时，不会使用在 URI 上设置的所有连接选项 (connectionTimeout, requestedChannelMax...)		ConnectionFactory
threadPoolSize (consumer)	使用者使用带有固定线程数的线程池可执行文件。此设置允许您设置该线程数量。	10	int
autoDetectConnection Factory (advanced)	是否自动检测是否从注册表查找 RabbitMQ 连接工厂。启用并找到连接工厂的单一实例后，它将被使用。可以在组件或端点级别配置显式连接工厂。	true	布尔值
connectionTimeout (advanced)	连接超时	60000	int
requestedChannelMax (advanced)	请求的连接频道最大数（提供的最大频道数）	2047	int
requestedFrameMax (advanced)	请求的连接帧最大范围（提供的帧的最大大小）	0	int
requestedHeartbeat (advanced)	请求的连接心跳（以秒为单位）	60	int
自动恢复启用 (advanced)	启用连接自动恢复（使用在应用程序不启动连接关闭时执行自动恢复）		布尔值
networkRecoveryInterval (advanced)	以毫秒为单位的网络恢复间隔（从网络故障中恢复时使用的interval）	5000	整数

Name	描述	默认值	类型
topologyRecoveryEnabled (advanced)	启用连接拓扑恢复（应该执行拓扑恢复）		布尔值
prefetchEnabled (consumer)	在 RabbitMQConsumer 端启用服务质量。您需要同时指定 prefetchSize、prefetchCount、prefetchGlobal 的选项	false	布尔值
prefetchSize (consumer)	服务器将提供的最大内容量（以八位字节表示），如果无限，则会提供 0。您需要同时指定 prefetchSize、prefetchCount、prefetchGlobal 的选项		int
prefetchCount (consumer)	服务器将发送的最大消息数为 0 （如果没有限制）。您需要同时指定 prefetchSize、prefetchCount、prefetchGlobal 的选项		int
prefetchGlobal (consumer)	如果设置应该应用到整个频道，而不是每个消费者，您需要同时指定 prefetchSize、prefetchCount、prefetchCount、prefetchGlobal	false	布尔值
channelPoolMaxSize (producer)	获取池中打开的最大频道数	10	int
channelPoolMaxWait (producer)	设置从池中等待频道的最大毫秒数	1000	long
requestTimeout (advanced)	设置在使用 InOut Exchange Pattern 时等待回复的超时（以毫秒为单位）	20000	long
requestTimeoutCheckerInterval (advanced)	为 inOut 交换设置 requestTimeoutCheckerInterval	1000	long
transferException (advanced)	当对消费者进行 true 和 inOut Exchange 时，发送 caused Exception 返回响应中	false	布尔值
publisherAcknowledgements (producer)	为 true 时，消息将发布并打开发布发布者确认信息	false	布尔值
publisherAcknowledgementsTimeout (producer)	从 RabbitMQ 服务器等待基本.ack 响应的时间（以毫秒为单位）		long

Name	描述	默认值	类型
guaranteedDeliveries (producer)	为 true 时，当无法发送消息时(basic.return)并且消息被标记为 mandatory 时，会抛出异常。在这种情况下，发布者也会被激活。另请参阅发布者确认 - 消息将被确认的时间。	false	布尔值
mandatory (producer)	此标志告诉服务器如何在消息无法路由到队列时做出反应。如果设置了此标志，服务器将使用返回方法返回无法路由的消息。如果此标志为零，则服务器会静默丢弃消息。如果标题为 rabbitmq.MANDATORY，它将覆盖此选项。	false	布尔值
immediate (producer)	此标志告诉服务器如何在消息无法立即路由到队列消费者时做出反应。如果设置了此标志，服务器将使用返回方法返回一个 undeliverable 消息。如果此标志为零，服务器会将该消息排队，但不保证使用它。如果标头存在 rabbitmq.IMMEDIATE，它将覆盖此选项。	false	布尔值
args (advanced)	指定用于配置不同的 RabbitMQ 概念的参数，每个概念都需要一个不同的前缀： Exchange: arg.exchange。 queue：arg.queue.binding：arg.binding.例如，使用 message ttl 参数声明队列：http://localhost:5672/exchange/queueeargs=arg.queue.x-message-ttl=60000		Map
clientProperties (advanced)	连接客户端属性（在服务器中使用的客户端信息）		Map
SSLProtocol (security)	在连接中启用 SSL，接受的值为 true、TLS 和 'SSLv3		字符串
trustmanager (security)	配置 SSL 信任管理器，应该启用 SSL 才能使此选项生效		TrustManager
autoAck (consumer)	如果应该自动确认消息	true	布尔值
auto delete (common)	如果为 true，则在不再使用交换时会删除交换	true	布尔值
durable (common)	如果我们声明持久化的交换（服务器重启后交换将保留）	true	布尔值
exclusive (common)	独占队列只能由当前连接访问，并在该连接关闭时删除。	false	布尔值
exclusiveConsumer (consumer)	请求对队列进行独占访问（注意只有此使用者可以访问队列）。当您希望一个长期共享队列被一个消费者临时访问时，这很有用。	false	布尔值

Name	描述	默认值	类型
passive (common)	被动队列依赖于 RabbitMQ 上已经可用的队列。	false	布尔值
skipQueueDeclare (common)	如果为 true，则生成者不会声明和绑定队列。这可用于通过现有的路由密钥指示消息。	false	布尔值
skipQueueBind (common)	如果为 true，则在声明后队列不会绑定到交换	false	布尔值
skipExchangeDeclare (common)	如果需要声明队列而不是交换，可以使用它	false	布尔值
声明 (common)	如果选项为 true，camel 声明交换和队列名称并将其绑定在一起。如果选项为 false，则 camel 不会声明服务器上的交换和队列名称。	true	布尔值
deadLetterExchange (common)	死信交换的名称		字符串
deadLetterQueue (common)	死信队列的名称		字符串
deadLetterRoutingKey (common)	死信交换的路由密钥		字符串
deadLetterExchangeType (common)	死信交换的类型	direct	字符串
allowNullHeaders (producer)	允许将 null 值传递给标头	false	布尔值
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

RabbitMQ 端点使用 URI 语法进行配置：

```
rabbitmq:exchangeName
```

使用以下路径和查询参数：

272.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
exchangeName	**必需** The Exchange name 决定将发送到的交换生成的消息。如果是消费者，交换名称决定了队列将绑定到的交换。		字符串

272.2.2. 查询参数(62 参数)：

Name	描述	默认值	类型
addresses (common)	如果设置了这个选项，camel-rabbitmq 将尝试根据选项地址的设置来创建连接。addresses 值是一个字符串，它类似于 server1:12345, server2:12345		Address[]
auto delete (common)	如果为 true，则在不再使用交换时会删除交换	true	布尔值
ConnectionFactory (common)	使用自定义 RabbitMQ 连接工厂。当设定这个选项时，不会使用在 URI 上设置的所有连接选项 (connectionTimeout, requestedChannelMax...)		ConnectionFactory
deadLetterExchange (common)	死信交换的名称		字符串
deadLetterExchangeType (common)	死信交换的类型	direct	字符串
deadLetterQueue (common)	死信队列的名称		字符串
deadLetterRoutingKey (common)	死信交换的路由密钥		字符串
声明 (common)	如果选项为 true，camel 声明交换和队列名称并将其绑定在一起。如果选项为 false，则 camel 不会声明服务器上的交换和队列名称。	true	布尔值
durable (common)	如果我们声明持久化的交换（服务器重启后交换将保留）	true	布尔值
exchangeType (common)	交换类型，如 direct 或 topic。	direct	字符串
exclusive (common)	独占队列只能由当前连接访问，并在该连接关闭时删除。	false	布尔值

Name	描述	默认值	类型
hostname (common)	正在运行的 rabbitmq 实例或集群的主机名。		字符串
passive (common)	被动队列依赖于 RabbitMQ 上已经可用的队列。	false	布尔值
portnumber (common)	带有正在运行的 rabbitmq 实例或集群的主机的端口号。默认值为 5672。		int
queue (common)	从其中接收消息的队列		字符串
routingKey (common)	将消费者队列绑定到交换时使用的路由密钥。对于生成者路由密钥，您可以设置标头 rabbitmq.ROUTING_KEY。		字符串
skipExchangeDeclare (common)	如果需要声明队列而不是交换，可以使用它	false	布尔值
skipQueueBind (common)	如果为 true，则在声明后队列不会绑定到交换	false	布尔值
skipQueueDeclare (common)	如果为 true，则生成者不会声明和绑定队列。这可用于通过现有的路由密钥指示消息。	false	布尔值
vhost (common)	频道的 vhost	/	字符串
autoAck (consumer)	如果应该自动确认消息	true	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
concurrentConsumers (consumer)	从代理消耗时并发消费者数量。（与 JMS 组件相同的选项类似）。	1	int
exclusiveConsumer (consumer)	请求对队列进行独占访问（注意只有此使用者可以访问队列）。当您希望一个长期共享队列被一个消费者临时访问时，这很有用。	false	布尔值
prefetchCount (consumer)	服务器将发送的最大消息数为 0 （如果没有限制）。您需要同时指定 prefetchSize、prefetchCount、prefetchGlobal 的选项		int

Name	描述	默认值	类型
prefetchEnabled (consumer)	在 RabbitMQConsumer 端启用服务质量。您需要同时指定 prefetchSize、prefetchCount、prefetchGlobal 的选项	false	布尔值
prefetchGlobal (consumer)	如果设置应该应用到整个频道，而不是每个消费者，您需要同时指定 prefetchSize、prefetchCount、prefetchCount、prefetchGlobal	false	布尔值
prefetchSize (consumer)	服务器将提供的最大内容量（以八位字节表示），如果无限，则会提供 0。您需要同时指定 prefetchSize、prefetchCount、prefetchGlobal 的选项		int
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
threadPoolSize (consumer)	使用者使用带有固定线程数的线程池可执行文件。此设置允许您设置该线程数量。	10	int
allowNullHeaders (producer)	允许将 null 值传递给标头	false	布尔值
bridgeEndpoint (producer)	如果 bridgeEndpoint 为 true，则生成者将忽略 rabbitmq.EXCHANGE_NAME 和 rabbitmq.ROUTING_KEY 的消息标头	false	布尔值
channelPoolMaxSize (producer)	获取池中打开的最大频道数	10	int
channelPoolMaxWait (producer)	设置从池中等待频道的最大毫秒数	1000	long
guaranteedDeliveries (producer)	为 true 时，当无法发送消息时(basic.return)并且消息被标记为 mandatory 时，会抛出异常。在这种情况下，发布者也会被激活。另请参阅发布者确认 - 消息将被确认的时间。	false	布尔值
immediate (producer)	此标志告诉服务器如何在消息无法立即路由到队列消费者时做出反应。如果设置了此标志，服务器将使用返回方法返回一个 undeliverable 消息。如果此标志为零，服务器会将该消息排队，但不保证使用它。如果标头存在 rabbitmq.IMMEDIATE，它将覆盖此选项。	false	布尔值

Name	描述	默认值	类型
mandatory (producer)	此标志告诉服务器如何在消息无法路由到队列时做出反应。如果设置了此标志，服务器将使用返回方法返回无法路由的消息。如果此标志为零，则服务器会静默丢弃消息。如果标题为 rabbitmq.MANDATORY，它将覆盖此选项。	false	布尔值
publisherAcknowledgements (producer)	为 true 时，消息将发布并打开发布发布者确认信息	false	布尔值
publisherAcknowledgementsTimeout (producer)	从 RabbitMQ 服务器等待基本.ack 响应的时间（以毫秒为单位）		long
args (advanced)	指定用于配置不同的 RabbitMQ 概念的参数，每个概念都需要一个不同的前缀： Exchange: arg.exchange。queue：arg.queue.binding：arg.binding.例如，使用 message ttl 参数声明队列：http://localhost:5672/exchange/queueargs=arg.queue.x-message-ttl=60000		Map
automaticRecoveryEnabled (advanced)	启用连接自动恢复（使用在应用程序不启动连接关闭时执行自动恢复）		布尔值
bindingArgs (advanced)	**弃用了** Key/value args，用于在 declare=true 时配置队列绑定参数		Map
clientProperties (advanced)	连接客户端属性（在服务器中使用的客户端信息）		Map
connectionTimeout (advanced)	连接超时	60000	int
exchangeArgs (advanced)	**弃用了** Key/value args，用于在 declare=true 时配置交换参数		Map
exchangeArgsConfigurer (advanced)	**弃用的** 设置配置器，以在 Channel.exchangeDeclare 中设置交换参数		ArgsConfigurer
networkRecoveryInterval (advanced)	以毫秒为单位的网络恢复间隔（从网络故障中恢复时使用的interval）	5000	整数
queueArgs (advanced)	**弃用了** Key/value args，用于在 declare=true 时配置队列参数		Map

Name	描述	默认值	类型
queueArgsConfigurer (advanced)	**弃用的** 设置配置器，以在 Channel.queueDeclare 中设置 queue args		ArgsConfigurer
requestedChannelMax (advanced)	请求的连接频道最大数（提供的最大频道数）	2047	int
requestedFrameMax (advanced)	请求的连接帧最大范围（提供的帧的最大大小）	0	int
requestedHeartbeat (advanced)	请求的连接心跳（以秒为单位）	60	int
requestTimeout (advanced)	设置在使用 InOut Exchange Pattern 时等待回复的超时（以毫秒为单位）	20000	long
requestTimeoutChecker Interval (advanced)	为 inOut 交换设置 requestTimeoutCheckerInterval	1000	long
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
topologyRecoveryEnabled (advanced)	启用连接拓扑恢复（应该执行拓扑恢复）		布尔值
transferException (advanced)	当对消费者进行 true 和 inOut Exchange 时，发送 caused Exception 返回响应中	false	布尔值
密码 (security)	验证访问的密码	Guest	字符串
SSLProtocol (security)	在连接中启用 SSL，接受的值为 true、TLS 和 'SSLv3		字符串
trustmanager (security)	配置 SSL 信任管理器，应该启用 SSL 才能使此选项生效		TrustManager
用户名 (security)	身份验证访问时的用户名	Guest	字符串

272.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 51 选项，如下所列。

Name	描述	默认值	类型
camel.component.rabbitmq.addresses	如果设置了这个选项，camel-rabbitmq 将尝试根据选项地址的设置来创建连接。addresses 值是一个字符串，它类似于 server1:12345, server2:12345		字符串
camel.component.rabbitmq.allow-null-headers	允许将 null 值传递给标头	false	布尔值
camel.component.rabbitmq.args	指定用于配置不同的 RabbitMQ 概念的参数，每个概念都需要一个不同的前缀： Exchange: arg.exchange。queue：arg.queue.binding：arg.binding.例如，使用 message ttl 参数声明队列：http://localhost:5672/exchange/queueeargs=arg.queue.x-message-ttl=60000		Map
camel.component.rabbitmq.auto-ack	如果应该自动确认消息	true	布尔值
camel.component.rabbitmq.auto-delete	如果为 true，则在不再使用交换时会删除交换	true	布尔值
camel.component.rabbitmq.auto-detect-connection-factory	是否自动检测是否从注册表查找 RabbitMQ 连接工厂。启用并找到连接工厂的单一实例后，它将被使用。可以在组件或端点级别配置显式连接工厂。	true	布尔值
camel.component.rabbitmq.automatic-recovery-enabled	启用连接自动恢复（使用在应用程序不启动连接关闭时执行自动恢复）		布尔值
camel.component.rabbitmq.channel-pool-max-size	获取池中打开的最大频道数	10	整数
camel.component.rabbitmq.channel-pool-max-wait	设置从池中等待频道的最大毫秒数	1000	Long
camel.component.rabbitmq.client-properties	连接客户端属性（在服务器中使用的客户端信息）		Map

Name	描述	默认值	类型
camel.component.rabbitmq.connection-factory	使用自定义 RabbitMQ 连接工厂。当设定这个选项时，不会使用在 URI 上设置的所有连接选项 (connectionTimeout, requestedChannelMax...)。选项是一个 com.rabbitmq.client.ConnectionFactory 类型。		字符串
camel.component.rabbitmq.connection-timeout	连接超时	60000	整数
camel.component.rabbitmq.dead-letter-exchange	死信交换的名称		字符串
camel.component.rabbitmq.dead-letter-exchange-type	死信交换的类型	direct	字符串
camel.component.rabbitmq.dead-letter-queue	死信队列的名称		字符串
camel.component.rabbitmq.dead-letter-routing-key	死信交换的路由密钥		字符串
camel.component.rabbitmq.declare	如果选项为 true，camel 声明交换和队列名称并将其绑定在一起。如果选项为 false，则 camel 不会声明服务器上的交换和队列名称。	true	布尔值
camel.component.rabbitmq.durable	如果我们声明持久化的交换（服务器重启后交换将保留）	true	布尔值
camel.component.rabbitmq.enabled	启用 rabbitmq 组件	true	布尔值
camel.component.rabbitmq.exclusive	独占队列只能由当前连接访问，并在该连接关闭时删除。	false	布尔值
camel.component.rabbitmq.exclusive-consumer	请求对队列进行独占访问（注意只有此使用者可以访问队列）。当您希望一个长期共享队列被一个消费者临时访问时，这很有用。	false	布尔值

Name	描述	默认值	类型
camel.component.rabbitmq.guaranteed-deliveries	为 true 时，当无法发送消息时(basic.return)并且消息被标记为 mandatory 时，会抛出异常。在这种情况下，发布者也会被激活。另请参阅发布者确认 - 消息将被确认的时间。	false	布尔值
camel.component.rabbitmq.hostname	运行 RabbitMQ 实例或集群的主机名。		字符串
camel.component.rabbitmq.immediate	此标志告诉服务器如何在消息无法立即路由到队列消费者时做出反应。如果设置了此标志，服务器将使用返回方法返回一个 undeliverable 消息。如果此标志为零，服务器会将该消息排队，但不保证使用它。如果标头存在 rabbitmq.IMMEDIATE，它将覆盖此选项。	false	布尔值
camel.component.rabbitmq.mandatory	此标志告诉服务器如何在消息无法路由到队列时做出反应。如果设置了此标志，服务器将使用返回方法返回无法路由的消息。如果此标志为零，则服务器会静默丢弃消息。如果标题为 rabbitmq.MANDATORY，它将覆盖此选项。	false	布尔值
camel.component.rabbitmq.network-recovery-interval	以毫秒为单位的网络恢复间隔（从网络故障中恢复时使用的interval）	5000	整数
camel.component.rabbitmq.passive	被动队列依赖于 RabbitMQ 上已经可用的队列。	false	布尔值
camel.component.rabbitmq.password	验证访问的密码	Guest	字符串
camel.component.rabbitmq.port-number	带有正在运行的 rabbitmq 实例或集群的主机的端口号。	5672	整数
camel.component.rabbitmq.prefetch-count	服务器将发送的最大消息数为 0 （如果没有限制）。您需要同时指定 prefetchSize、prefetchCount、prefetchGlobal 的选项		整数
camel.component.rabbitmq.prefetch-enabled	在 RabbitMQConsumer 端启用服务质量。您需要同时指定 prefetchSize、prefetchCount、prefetchGlobal 的选项	false	布尔值
camel.component.rabbitmq.prefetch-global	如果设置应该应用到整个频道，而不是每个消费者，您需要同时指定 prefetchSize、prefetchCount、prefetchCount、prefetchGlobal	false	布尔值

Name	描述	默认值	类型
camel.component.rabbitmq.prefetch-size	服务器将提供的最大内容量（以八位字节表示），如果无限，则会提供 0。您需要同时指定 prefetchSize、prefetchCount、prefetchGlobal 的选项		整数
camel.component.rabbitmq.publisher-acknowledgements	为 true 时，消息将发布并打开发布发布者确认信息	false	布尔值
camel.component.rabbitmq.publisher-acknowledgements-timeout	从 RabbitMQ 服务器等待基本.ack 响应的时间（以毫秒为单位）		Long
camel.component.rabbitmq.request-timeout	设置在使用 InOut Exchange Pattern 时等待回复的超时（以毫秒为单位）	20000	Long
camel.component.rabbitmq.request-timeout-checker-interval	为 inOut 交换设置 requestTimeoutCheckerInterval	1000	Long
camel.component.rabbitmq.requested-channel-max	请求的连接频道最大数（提供的最大频道数）	2047	整数
camel.component.rabbitmq.requested-frame-max	请求的连接帧最大范围（提供的帧的最大大小）	0	整数
camel.component.rabbitmq.requested-heartbeat	请求的连接心跳（以秒为单位）	60	整数
camel.component.rabbitmq.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.rabbitmq.skip-exchange-declare	如果需要声明队列而不是交换，可以使用它	false	布尔值
camel.component.rabbitmq.skip-queue-bind	如果为 true，则在声明后队列不会绑定到交换	false	布尔值

Name	描述	默认值	类型
camel.component.rabbitmq.skip-queue-declare	如果为 true，则生成者不会声明和绑定队列。这可用于通过现有的路由密钥指示消息。	false	布尔值
camel.component.rabbitmq.ssl-protocol	在连接中启用 SSL，接受的值为 true、TLS 和 'SSLv3		字符串
camel.component.rabbitmq.thread-pool-size	使用者使用带有固定线程数的线程池可执行文件。此设置允许您设置该线程数量。	10	整数
camel.component.rabbitmq.topology-recovery-enabled	启用连接拓扑恢复（应该执行拓扑恢复）		布尔值
camel.component.rabbitmq.transfer-exception	当对消费者进行 true 和 inOut Exchange 时，发送 caused Exception 返回响应中	false	布尔值
camel.component.rabbitmq.trust-manager	配置 SSL 信任管理器，应该启用 SSL 才能使此选项生效。选项是 javax.net.ssl.TrustManager 类型。		字符串
camel.component.rabbitmq.username	身份验证访问时的用户名	Guest	字符串
camel.component.rabbitmq.vhost	频道的 vhost	/	字符串

有关连接选项的更多信息，请参阅 http://www.rabbitmq.com/releases/rabbitmq-java-client/current-javadoc/com/rabbitmq/client/ConnectionFactory.html 和 AMQP 规格。

272.4. 使用连接工厂

要连接到 RabbitMQ，您可以使用登录详细信息设置 ConnectionFactory （与 JMS 相同）：

```
<bean id="rabbitConnectionFactory" class="com.rabbitmq.client.ConnectionFactory">
  <property name="host" value="localhost"/>
  <property name="port" value="5672"/>
  <property name="username" value="camel"/>
```

```
  <property name="password" value="bugsbunny"/>
</bean>

And then refer to the connection factory in the endpoint uri as shown below:

<camelContext>
  <route>
    <from uri="direct:rabbitMQEx2"/>
    <to uri="rabbitmq:ex2?connectionFactory=#rabbitConnectionFactory"/>
  </route>
</camelContext>
```

默认情况下，Camel 2.21 中会自动探测到 ConnectionFactory，因此您只需执行

```
<camelContext>
  <route>
    <from uri="direct:rabbitMQEx2"/>
    <to uri="rabbitmq:ex2"/>
  </route>
</camelContext>
```

272.5. 消息标头

在使用消息时，在交换上设置以下标头。

属性	value
rabbitmq.ROUTING_KEY	用于接收消息的路由密钥或生成消息时使用的路由密钥
rabbitmq.EXCHANGE_NAME	接收消息的交换
rabbitmq.DELIVERY_TAG	接收消息的 rabbitmq delivery 标签

属性	value
rabbitmq.REDELIVERY_TAG	消息是否为红色
rabbitmq.REQUEUE	Camel 2.14.2： 这供消费者用来控制邮件的拒绝。当消费者完成处理交换时，如果交换失败，则消费者将拒绝来自 RabbitMQ 代理的消息。此标头的值控制此行为。如果值为 false （默认为），则消息将被丢弃/死字母。如果值为 true，则消息会被重新排队。

生成者使用以下标头。如果在 camel 交换上设置它们，则将在 RabbitMQ 消息上设置。

属性	value
rabbitmq.ROUTING_KEY	发送消息时使用的路由密钥
rabbitmq.EXCHANGE_NAME	接收消息的交换
rabbitmq.EXCHANGE_OVERRIDE_NAME	Camel 2.21： 用于强制向此交换发送消息，而不是在生成者上配置的端点
rabbitmq.CONTENT_TYPE	在 RabbitMQ 消息上设置的 contentType
rabbitmq.PRIORITY	在 RabbitMQ 消息上设置的优先级标头

属性	value
rabbitmq.CORRELATIONID	在 RabbitMQ 消息上设置的 correlationId
rabbitmq.MESSAGE_ID	在 RabbitMQ 消息上设置的消息 ID
rabbitmq.DELIVERY_MODE	如果消息应该是持久的
rabbitmq.USERID	在 RabbitMQ 消息上设置的 userId
rabbitmq.CLUSTERID	在 RabbitMQ 消息上设置的 clusterId
rabbitmq.REPLY_TO	在 RabbitMQ 消息上设置的 replyTo
rabbitmq.CONTENT_ENCODING	在 RabbitMQ 消息上设置的 contentEncoding
rabbitmq.TYPE	在 RabbitMQ 消息上设置的类型
rabbitmq.EXPIRATION	在 RabbitMQ 消息上设置的过期时间

属性	value
rabbitmq.TIMESTAMP	在 RabbitMQ 消息上设置的时间戳
rabbitmq.APP_ID	在 RabbitMQ 消息上设置的 appId

标头由消费者设置，在收到消息后。在交换发生后，生成者还会为下游处理器设置标头。生产之前设置的任何标头都会被覆盖。

272.6. 消息正文

组件将使用正文中的 camel Exchange 作为 rabbit mq 消息正文。对象中的 camel Exchange 必须转换为字节数组。否则，生成者会抛出不支持的正文类型例外。

272.7. SAMPLES

要从绑定到路由密钥 B 的交换 A 的队列接收消息，

```
from("rabbitmq:A?routingKey=B")
```

要接收来自带有自动确认禁用的单一线程的队列的消息。

```
from("rabbitmq:A?routingKey=B&threadPoolSize=1&autoAck=false")
```

将消息发送到名为 C 的交换

```
to("rabbitmq:C")
```

声明标头交换和队列

```
from("rabbitmq:ex?exchangeType=headers&queue=q&bindingArgs=#bindArgs")
```

并将对应的 Map<String, Object > 替换为 Registry 中的 "bindArgs" 的 id。

例如，在 spring 中声明方法

```
@Bean(name="bindArgs")
public Map<String, Object> bindArgsBuilder() {
    return Collections.singletonMap("foo", "bar");
}
```

272.7.1. 在交换之间路由时出现问题（在 Camel 2.20.x 或更早版本中）

例如，如果要将信息从一个 Rabbit 交换路由到另一个，如下例所示，带有 foo → bar ：

```
from("rabbitmq:foo")
  .to("rabbitmq:bar")
```

然后请注意，Camel 会将消息路由到其自身，如 foo → foo。那么为什么如此？这是因为接收消息的使用者（例如 from ）提供消息标题 rabbitmq.EXCHANGE_NAME，其交换名称为 eg foo。当 Camel 生成者将消息发送到 bar 时，标头 rabbitmq.EXCHANGE_NAME 将覆盖此操作，而发送消息到 foo。

要避免这种情况，您需要：

- 删除标头：

```
from("rabbitmq:foo")
  .removeHeader("rabbitmq.EXCHANGE_NAME")
  .to("rabbitmq:bar")
```

- 或者打开 producer 上的 bridgeEndpoint 模式：

```
from("rabbitmq:foo")
  .to("rabbitmq:bar?bridgeEndpoint=true")
```

从 Camel 2.21 开始有所改进，因此您可以在交换之间轻松路由。标头 rabbitmq.EXCHANGE_NAME 不再被生成者用来覆盖目标交换。相反，可以使用一个新的标头 rabbitmq.EXCHANGE_OVERRIDE_NAME 发送到不同的交换。例如，要发送到 cheese 交换，您可以这样做

```
from("rabbitmq:foo")
  .setHeader("rabbitmq.EXCHANGE_OVERRIDE_NAME", constant("cheese"))
  .to("rabbitmq:bar")
```

第 273 章 被动流组件

从 Camel 版本 2.19 开始提供

reactive-streams: 组件允许您与 reactive 流标准兼容被动流处理库来交换消息。 http://www.reactive-streams.org/

组件支持后退化，并使用 被动流技术兼容性工具包(TCK) 进行测试。

Camel 模块提供了一个 reactive-streams 组件，允许用户在 Camel 路由中定义进入和传出流，以及一个直接客户端 API，允许将 Camel 端点直接用于任何外部被动框架。

Camel 使用了重新主动流 发布者和订阅 商的内部实施，因此它不与任何特定的框架关联。集成测试中使用了以下被动框架： Reactor Core 3、 RxJava 2。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-reactive-streams</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

273.1. URI 格式

```
reactive-streams://stream?[options]
```

其中 stream 是一个逻辑流名称，用于将 Camel 路由绑定到外部流处理系统。

273.2. 选项

被动流组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
internalEngine 配置 (advanced)	为 Reactive Streams 配置内部引擎。		ReactiveStreamsEngine 配置
backpressureStrategy (producer)	将事件推送到较慢的订阅者时使用的后退策略。	BUFFER	ReactiveStreamsBackpressureStrategy
ServiceType (advanced)	设置要使用的底层被动流实施的类型。实现从 registry 查找或使用 ServiceLoader，默认的实现是 DefaultCamelReactiveStreamsService		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Reactive Streams 端点使用 URI 语法进行配置：

```
reactive-streams:stream
```

使用以下路径和查询参数：

273.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
stream	端点用于交换消息的流频道名称。		字符串

273.2.2. 查询参数(10 parameters):

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
concurrentConsumers (consumer)	在 Camel 路由中处理交换的线程数量。	1	int

Name	描述	默认值	类型
exchangesRefillLowWatermark (consumer)	将请求的交换的低水位线设置为 maxInflightExchanges 的百分比。当来自上游源的待处理项目数量低于水位线时，可以向订阅请求新项目。如果设置为 0，则订阅者将在处理之前批处理的所有项目后以 maxInflightExchanges 批量请求项目。如果设置为 1，则订阅者每次处理交换时可以请求一个新项目（聊天）。可以使用任何中间值。	0.25	double
forwardOnComplete (consumer)	确定在Complete 事件上是否应推送到 Camel 路由。	false	布尔值
forwardOnError (consumer)	决定 onError 事件是否应该推送到 Camel 路由。例外将设置为消息正文。	false	布尔值
maxInflightExchanges (consumer)	Camel 同时处理的最大交换数。此参数控制流上的后退。设置非正数值将禁用后退。	128	整数
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
backpressureStrategy (producer)	将事件推送到较慢的订阅者时使用的后退策略。		ReactiveStreamsBackpressureStrategy
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

273.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 7 个选项，如下所列。

Name	描述	默认值	类型
camel.component.reactive-streams.backpressure-strategy	将事件推送到较慢的订阅者时使用的后退策略。		ReactiveStreamsBackpressureStrategy

Name	描述	默认值	类型
camel.component.reactive-streams.enabled	启用 reactive-streams 组件	true	布尔值
camel.component.reactive-streams.internal-engine-configuration.thread-pool-max-size	被动流内部引擎使用的最大线程数。		整数
camel.component.reactive-streams.internal-engine-configuration.thread-pool-min-size	被动流内部引擎使用的最小线程数量。		整数
camel.component.reactive-streams.internal-engine-configuration.thread-pool-name	被动流内部引擎使用的线程池名称。		字符串
camel.component.reactive-streams.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.reactive-streams.service-type	设置要使用的底层被动流实施的类型。实现从 registry 查找或使用 ServiceLoader，默认的实现是 DefaultCamelReactiveStreamsService		字符串

273.4. 使用方法

这个程序库的目的是支持应用程序与 Camel 数据交互所需的所有通信模式：

- *从 Camel 路由获取数据（仅来自 Camel）*
- *将数据发送到 Camel 路由（仅适用于 Camel）*

- 请求 向 Camel 路由转换(In-Out 到 Camel)

- 使用被动 处理 步骤（从 Camel 中分离）处理来自 Camel 路由的数据流

273.5. 从 CAMEL 获取数据

为了订阅来自 Camel 路由的数据流，应将交换重定向到命名流，如以下片段所示：

```
from("timer:clock")
.setBody().header(Exchange.TIMER_COUNTER)
.to("reactive-streams:numbers");
```

也可以使用 XML DSL 编写路由。

在示例中，未绑定的数字流与名称 号 关联。该流可以使用 CamelReactiveStreams 实用程序类来访问。

```
CamelReactiveStreamsService camel = CamelReactiveStreams.get(context);

// Getting a stream of exchanges
Publisher<Exchange> exchanges = camel.fromStream("numbers");

// Getting a stream of Integers (using Camel standard conversion system)
Publisher<Integer> numbers = camel.fromStream("numbers", Integer.class);
```

流可用于任何兼容被动流的被动流。以下是如何将它与 RxJava 2 搭配使用的示例（尽管任何被动框架可用于处理事件）。

```
Flowable.fromPublisher(integers)
   .doOnNext(System.out::println)
   .subscribe();
```

这个示例将 Camel 生成的所有数字打印到 system .out 中。

273.5.1. 使用直接 API 从 Camel 获取数据

对于希望使用被动框架功能构造定义整个处理流程的用户（无需使用 Camel DSL），也可以使用 Camel URI 定义流。

```
CamelReactiveStreamsService camel = CamelReactiveStreams.get(context);

// Get a stream from all the files in a directory
Publisher<String> files = camel.from("file:folder", String.class);

// Use the stream in RxJava2
Flowable.fromPublisher(files)
    .doOnNext(System.out::println)
    .subscribe();
```

273.6. 将数据发送到 CAMEL

当外部库需要将事件推送到 Camel 路由时，必须将 Reactive Streams 端点设置为消费者。

```
from("reactive-streams:elements")
.to("log:INFO");
```

元素 流的句柄可以从 CamelReactiveStreams 实用程序类获得。

```
CamelReactiveStreamsService camel = CamelReactiveStreams.get(context);

Subscriber<String> elements = camel.streamSubscriber("elements", String.class);
```

订阅者可用于将事件推送到从 元素 流消耗的 Camel 路由。

以下是如何将它与 RxJava 2 搭配使用的示例（尽管任何被动框架都可用于发布事件）。

```
Flowable.interval(1, TimeUnit.SECONDS)
    .map(i -> "Item " + i)
    .subscribe(elements);
```

字符串项由示例中的 RxJava 每秒生成，它们被推送到上面定义的 Camel 路由中。

273.6.1. 使用直接 API 将数据发送到 Camel

另外，在本例中，直接 API 可用于从端点 URI 获取 Camel 订阅者。

```
CamelReactiveStreamsService camel = CamelReactiveStreams.get(context);

// Send two strings to the "seda:queue" endpoint
```

```
Flowable.just("hello", "world")
    .subscribe(camel.subscriber("seda:queue", String.class));
```

273.7. 请求到 CAMEL 的转换

在一些 Camel DSL 中定义的路由可以在被动流框架中使用来执行特定的转换（相同的机制也可用于将数据发送到 http 端点并继续）。

以下代码片段演示了 RxJava 功能代码如何向 Camel 请求加载和汇总文件的任务。

```
CamelReactiveStreamsService camel = CamelReactiveStreams.get(context);

// Process files starting from their names
Flowable.just(new File("file1.txt"), new File("file2.txt"))
    .flatMap(file -> camel.toStream("readAndMarshal", String.class))
    // Camel output will be converted to String
    // other steps
    .subscribe();
```

为了正常工作，应在 Camel 上下文中定义类似以下内容的路由：

```
from("reactive-streams:readAndMarshal")
.marshal() // ... other details
```

273.7.1. 使用直接 API 请求到 Camel 的转换

另一种方法包括在被动流中直接使用 URI 端点：

```
CamelReactiveStreamsService camel = CamelReactiveStreams.get(context);

// Process files starting from their names
Flowable.just(new File("file1.txt"), new File("file2.txt"))
    .flatMap(file -> camel.to("direct:process", String.class))
    // Camel output will be converted to String
    // other steps
    .subscribe();
```

当使用 to （） 方法而不是 toStream 时，不需要使用 "reactive-streams:" 端点（尽管它们在 hood 下使用）来定义路由。

在这种情况下，Camel 转换只能：

```
from("direct:process")
.marshal() // ... other details
```

273.8. 将 CAMEL 数据处理到被动框架中

虽然被动流发布软件允许以单向方式交换数据，但 Camel 路由通常使用 in-out 交换模式（例如，定义 REST 端点和一般来说，每个请求都需要回复）。

在这些情况下，用户可以向流添加被动处理步骤，以增强 Camel 路由或使用被动框架定义整个转换。

例如，给定以下路由：

```
from("timer:clock")
.setBody().header(Exchange.TIMER_COUNTER)
.to("direct:reactive")
.log("Continue with Camel route... n=${body}");
```

被动处理步骤可以与 "direct:reactive" 端点关联：

```
CamelReactiveStreamsService camel = CamelReactiveStreams.get(context);

camel.process("direct:reactive", Integer.class, items ->
    Flowable.fromPublisher(items) // RxJava2
        .map(n -> -n)); // make every number negative
```

Camel 路由中的数据流将由外部被动框架处理，然后继续 Camel 内处理流程。

这种机制也可用于以完全被动的方式定义 In-Out 交换。

```
CamelReactiveStreamsService camel = CamelReactiveStreams.get(context);

// requires a rest-capable Camel component
camel.process("rest:get:orders", exchange ->
                Flowable.fromPublisher(exchange)
                        .flatMap(ex -> allOrders())); // retrieve orders asynchronously
```

详情请查看 Camel 示例(camel-example-reactive-streams)。

273.9. 高级主题

273.9.1. 控制 Backpressure (producer side)

当路由 Camel 交换到外部订阅者时，会由缓存在交付前交换的内部缓冲区处理后向。如果订阅者比交换速率慢，则缓冲区可能会变得太大。在很多情况下，必须避免这种情况。

考虑以下路由：

```
from("jms:queue")
.to("reactive-streams:flow");
```

如果 JMS 队列包含大量消息，并且与 流流 关联的 Subscriber 太慢，消息会从 JMS 分离并附加到缓冲区中，可能会导致"内存不足"错误。要避免这样的问题，可以在路由中设置 ThrottlingInflightRoutePolicy。

```
ThrottlingInflightRoutePolicy policy = new ThrottlingInflightRoutePolicy();
policy.setMaxInflightExchanges(10);

from("jms:queue")
.routePolicy(policy)
.to("reactive-streams:flow");
```

策略限制活跃交换的最大数量（以及最大缓冲区大小），保持其低于阈值（示例中为10 ）。当有 10 个信息处于 flight 状态时，路由会被暂停，等待订阅者处理它们。

通过这种机制，订阅者通过后退自动控制路由暂停/恢复。当多个订阅者消耗来自同一流的项目时，会自动控制路由状态的速度。

在其他情况下，例如使用 http 消费者时，路由挂起会导致 http 服务不可用，因此应该优先使用默认配置（无策略、未绑定的缓冲）。用户应尝试避免内存问题，方法是限制对 http 服务的请求数（例如，横向扩展）。

在可以接受一定数量的数据丢失的情况下，设置 BUFFER 以外的后退策略可能是处理快速源的解决方案。

```
from("direct:thermostat")
.to("reactive-streams:flow?backpressureStrategy=LATEST");
```

使用 LATEST 后端策略时，仅保留从路由接收的最后交换程序，同时删除旧数据（另一个选项可用）。

273.9.2. 控制 Backpressure (consumer side)

当 Camel 使用来自 reactive-streams 发布者的项目时，可以的最大动态交换数设为 endpoint 选项。

与消费者关联的订阅者与发布者交互，以保持路由中的消息数量低于阈值。

backpressure-aware 路由示例：

```
from("reactive-streams:numbers?maxInflightExchanges=10")
.to("direct:endpoint");
```

Camel 向源发布者请求的项目数（通过 reactive 流回转机制）始终小于 10。消息由 Camel 端的单个线程处理。

并发消费者（线程）数也可以设置为端点选项(concurrentConsumers)。使用 1 消费者（默认）时，源流中的项目顺序会被维护。当这个值增加时，项目将由多个线程同时处理（不会保留顺序）。

273.10. CAMEL REACTIVE STREAMS STARTER

spring-boot 用户提供了一个初学者模块。使用初学者时，CamelReactiveStreamsService 可以直接注入到 Spring 组件中。

要使用启动程序，将以下内容添加到 spring boot pom.xml 文件中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-reactive-streams-starter</artifactId>
  <version>${camel.version}</version> <!-- use the same version as your Camel core version --
>
</dependency>
```

273.11. 另请参阅

- 配置 Camel
- 组件

- ***端点***

- ***开始使用***

第 274 章 REACTOR 组件

从 Camel 版本 2.20 开始提供

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-reactor</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

第 275 章 REF 组件

从 Camel 版本 1.2 开始提供

ref: 组件用于查找 Registry 中绑定的现有端点。

275.1. URI 格式

```
ref:someName[?options]
```

其中 someName 是 Registry 中的端点名称（通常为，但并不总是是 Spring registry）。如果您使用 Spring registry，someName 将是 Spring registry 中端点的 bean ID。

275.2. REF 选项

Ref 组件没有选项。

Ref 端点使用 URI 语法进行配置：

```
ref:name
```

使用以下路径和查询参数：

275.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	必须在 registry 中查询的端点名称。		字符串

275.2.2. 查询参数(4 参数)：

Name	描述	默认值	类型

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外，该处理程序记录在 WARN/ERROR 级别并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，该例外记录在 WARN/ERROR 级别并忽略。		ExceptionHandler
exchangePattern (consumer)	在创建交换时设置默认交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

275.3. 运行时查找

当您需要在 Registry 中动态发现端点，您可以在运行时计算 URI 时使用此组件。然后，您可以使用以下代码查找端点：

```
// lookup the endpoint
String myEndpointRef = "bigspenderOrder";
Endpoint endpoint = context.getEndpoint("ref:" + myEndpointRef);

Producer producer = endpoint.createProducer();
Exchange exchange = producer.createExchange();
exchange.getIn().setBody(payloadToSend);
// send the exchange
producer.process(exchange);
```

您可以有一个在 Registry 中定义的端点列表，例如：

```
<camelContext id="camel" xmlns="http://activemq.apache.org/camel/schema/spring">
    <endpoint id="normalOrder" uri="activemq:order.slow"/>
    <endpoint id="bigspenderOrder" uri="activemq:order.high"/>
</camelContext>
```

275.4. 示例

在以下示例中，我们使用 URI 中的 ref: 来引用 spring ID endpoint2 的端点：

当然，您可以使用 ref 属性替代：

```
<to ref="endpoint2"/>
```

这是编写它的一种更常见方法。

第 276 章 REST 组件

从 Camel 版本 2.14 开始提供

其余组件允许使用 Rest DSL 和插件来定义 REST 端点(consumer)，作为 REST 传输。

在其他组件的 Camel 2.18 中，也可以用作客户端(producer)来调用 REST 服务。

276.1. URI 格式

```
rest://method:path[:uriTemplate]?[options]
```

276.2. URI 选项

REST 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
componentName (common)	用于 REST 传输的 Camel Rest 组件，如 restlet、spark-rest。如果没有明确配置组件，则 Camel 将查找是否有与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestConsumerFactory (consumer)或 org.apache.camel.spi.RestProducerFactory (producer)是否在 registry 中注册。如果找到其中任一个，则使用它。		字符串
apidoc (producer)	要使用的 OpenAPI api doc 资源。默认情况下，资源从 classpath 加载，必须采用 JSon 格式。		字符串
host (producer)	要使用的 HTTP 服务的主机和端口（在 OpenAPI 模式中设置主机）		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

REST 端点使用 URI 语法进行配置：

```
rest:method:path:uriTemplate
```

使用以下路径和查询参数：

276.2.1. 路径参数(3 参数)：

Name	描述	默认值	类型
方法	要使用 **所需的** HTTP 方法。		字符串
path	**必需** 的基本路径		字符串
uriTemplate	uri 模板		字符串

276.2.2. 查询参数(15 参数)：

Name	描述	默认值	类型
componentName (common)	用于 REST 传输的 Camel Rest 组件，如 restlet、spark-rest。如果没有明确配置组件，则如果有一个与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestConsumerFactory 在 registry 中注册，则 Camel 将查找。如果找到其中任一个，则使用它。		字符串
consume (common)	介质类型，如 'text/xml' 或 'application/json' this REST 服务接受。默认情况下，我们接受所有类型。		字符串
inType (common)	将传入的 POJO 绑定类型声明为 FQN 类名称		字符串
outType (common)	将传出的 POJO 绑定类型声明为 FQN 类名称		字符串
generate (common)	介质类型，如 'text/xml' 或 'application/json' this REST 服务返回。		字符串
routeId (common)	此 REST 服务创建的路由的名称		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外，该处理程序记录在 WARN/ERROR 级别并忽略。	false	布尔值
description (consumer)	记录此 REST 服务的人员描述		字符串

Name	描述	默认值	类型
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，该例外记录在 WARN/ERROR 级别并忽略。		ExceptionHandler
exchangePattern (consumer)	在创建交换时设置默认交换模式。		ExchangePattern
apidoc (producer)	要使用的 OpenAPI api doc 资源。默认情况下，资源从 classpath 加载，必须采用 JSon 格式。		字符串
bindingMode (producer)	配置制作者的绑定模式。如果设置为 'off' 以外的任何一个，则生成者将尝试将传入消息的正文从 inType 转换为 json 或 xml，并将 json 或 xml 的响应转换为 outType。		RestBindingMode
host (producer)	要使用的 HTTP 服务的主机和端口（在 OpenAPI 模式中设置主机）		字符串
queryParameters (producer)	要调用的 HTTP 服务的查询参数		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

276.3. 支持的其他组件

以下组件支持其余消费者(Rest DSL)：

- ***camel-coap***
- ***camel-netty-http***
- ***camel-netty4-http***
- ***camel-jetty***

- *camel-restlet*
- *camel-servlet*
- *camel-spark-rest*
- *camel-undertow*

以下组件支持 rest producer：

- *camel-http*
- *camel-http4*
- *camel-netty4-http*
- *camel-jetty*
- *camel-restlet*
- *camel-undertow*

276.4. PATH 和 URITEMPLATE 语法

path 和 uriTemplate 选项使用 REST 语法来定义，您可以在其中使用支持参数定义 REST 上下文路径。

TIP: 如果没有配置 uriTemplate，则 path 选项的工作方式相同。如果您只配置路径或者配置这两个选项，则这无关紧要。虽然配置路径和 uriTemplate 是使用 REST 的一个更常见的做法。

以下是只使用路径的 Camel 路由

```
from("rest:get:hello")
  .transform().constant("Bye World");
```

以下路由使用参数，该参数映射到带有键"me"的 Camel 标头。

```
from("rest:get:hello/{me}")
  .transform().simple("Bye ${header.me}");
```

以下示例配置了基础路径为 "hello"，然后使用 uriTemplates 配置两个 REST 服务。

```
from("rest:get:hello:/{me}")
  .transform().simple("Hi ${header.me}");

from("rest:get:hello:/french/{me}")
  .transform().simple("Bonjour ${header.me}");
```

276.5. REST 生成者示例

您可以使用其他组件调用 REST 服务，如任何其他 Camel 组件。

例如，要使用 hello/{me} 调用 上的 REST 服务，您可以这样做

```
from("direct:start")
  .to("rest:get:hello/{me}");
```

然后，动态值 {me} 被映射到具有相同名称的 Camel 消息。要调用此 REST 服务，您可以发送空消息正文和标头，如下所示：

```
template.sendBodyAndHeader("direct:start", null, "me", "Donald Duck");
```

Rest producer 需要知道 REST 服务的主机名和端口，您可以使用 host 选项进行配置，如下所示：

```
from("direct:start")
  .to("rest:get:hello/{me}?host=myserver:8080/foo");
```

您可以在 restConfiguration 上配置主机，而不使用 host 选项：

```
restConfiguration().host("myserver:8080/foo");

from("direct:start")
  .to("rest:get:hello/{me}");
```

您可以使用 producerComponent 选择将哪个 Camel 组件用作 HTTP 客户端，例如，使用 http4，您可以执行以下操作：

```
restConfiguration().host("myserver:8080/foo").producerComponent("http4");

from("direct:start")
  .to("rest:get:hello/{me}");
```

276.6. REST 生成者绑定

REST 制作者支持像 rest-dsl 一样使用 JSon 或 XML 进行绑定。

例如，要在启用 json 绑定模式中使用 jetty 来在 rest 配置中进行配置：

```
restConfiguration().component("jetty").host("localhost").port(8080).bindingMode(RestBinding
Mode.json);

from("direct:start")
  .to("rest:post:user");
```

然后，在使用 rest producer 调用 REST 服务时，它会在调用 REST 服务前自动将任何 POJO 绑定到 json ：

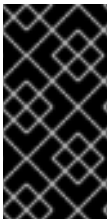
```
UserPojo user = new UserPojo();
user.setId(123);
user.setName("Donald Duck");

template.sendBody("direct:start", user);
```

在上例中，我们发送一个 POJO 实例 UserPojo 作为消息正文。由于我们在其他配置中开启了 JSon 绑定，因此在调用 REST 服务之前，POJO 将从 POJO 放入 JSon。

但是，如果您还想对响应消息（例如 REST 服务发送回响应）执行绑定，您需要配置 outType 选项，以指定 POJO 到 JSon 到 POJO 的类名称。

例如，如果 REST 服务返回一个 JSon 有效负载，它绑定到 com.foo.MyResponsePojo，您可以将其配置为如下所示：

```
restConfiguration().component("jetty").host("localhost").port(8080).bindingMode(RestBindingMode.json);

  from("direct:start")
    .to("rest:post:user?outType=com.foo.MyResponsePojo");
```



重要

如果您希望为调用 REST 服务收到的响应消息发生 POJO 绑定，您必须配置 outType 选项。

276.7. 更多示例

请参阅 Rest DSL，它们提供了更多示例，以及如何使用 Rest DSL 以 nicer RESTful 方式定义它们。

Apache Camel 分发中有一个 camel-example-servlet-rest-tomcat 示例，它演示了如何使用带有 SERVLET 的 Rest DSL 作为传输，这些传输可以在 Apache Tomcat 或类似的 Web 容器上部署。

276.8. 另请参阅

- REST DSL
- SERVLET

部分 I. REST OPENAPI COMPONENT

Since Camel 3.1

仅支持生成者

REST OpenApi 配置 OpenApi (Open API)规格文档中的 rest producers，并委派给实现 RestProducerFactory 接口的组件。目前已知的工作组件是：*

- *http*
- *netty-http*
- *undertow*

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-rest-openapi</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

第 277 章 URI 格式

```
rest-openapi:[specificationPath#]operationId
```

其中 operationId 是 OpenApi 规格中的操作 ID，specificationPath 是规格的路径。如果没有指定 specificationPath，则默认为 openapi.json。查找机制使用 CamelS ResourceHelper 来加载资源，这意味着您可以使用 CLASSPATH 资源(classpath:my-specification.json), 文件(file:/some/path.json), web (http://api.example.com/openapi.json)或引用 bean (ref:nameOfBean)或使用 bean (bean:nameOfBean.methodName)来获取规格资源，失败，失败：

此组件不充当 HTTP 客户端，它将该客户端委派给上述另一个组件。查找机制搜索实施 RestProducerFactory 接口并使用的单一组件。如果 CLASSPATH 包含多个，则应设置 property componentName 来指示要委派哪些组件。

大多数配置都从 OpenApi 规范获取，但存在选项，通过在组件或端点中指定它们来覆盖它们。通常，如果主机 或 basePath 与规格不同，则您只需要覆盖主机或 basePath。

注意

host 参数应包含包含方案、主机名和端口号的绝对 URI，例如：https://api.example.com

使用 componentName，您可以指定用于执行请求的组件，此命名组件需要存在于 Camel 上下文中，并实现所需的 RestProducerFactory 接口 swig-wagonas 处理顶部列出的组件。

如果您没有在组件或端点级别指定 componentName，则会搜索 CLASSPATH 以获得合适的委派。CLASSPATH 上应该只有一个组件，它实现了 RestProducerFactory 接口，以便它正常工作。

此组件的端点 URI 是很方便的，这意味着除了消息标头外，您还可以将 REST 操作的参数指定为端点参数，对于所有后续调用来说，它们会保持恒定的，因此仅将此功能用于对所有调用时常数的参数，如 /api/{version}/users/{id}。

第 278 章 选项

REST OpenApi 组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
basePath (producer)	API basePath，如 /v2。默认是 unset，如果设置会覆盖 OpenApi 规格中存在的值。		字符串
componentName (producer)	执行请求的 Camel 组件的名称。组件必须存在于 Camel registry 中，且必须实现 RestProducerFactory 服务供应商接口。如果没有设置 CLASSPATH，则会搜索实现 RestProducerFactory SPI 的单个组件。可以在端点配置中覆盖。		字符串
consumes (producer)	此组件可以使用的有效负载类型。可以是一种类型，如 application/json 或多种类型，如 application/json、application/xml; q=0.5，根据 RFC7231。这等同于 Accept HTTP 标头的值。如果设置会覆盖 OpenApi 规格中找到的任何值。可以在端点配置中覆盖		字符串
host (producer)	以 https://hostname:port 的形式将 HTTP 请求定向到的方案主机名和端口。可以在端点、组件或 Camel 上下文中的相应 REST 配置中配置。如果您将这个组件命名为 （如 petstore），则 REST 配置首先被查询，下一个 rest-openapi，最后为全局配置指定。如果设置覆盖 OpenApi 规格(RestConfiguration)中找到的任何值。可以在端点配置中覆盖。		字符串
generate (producer)	此组件生成的有效负载类型。例如，根据 RFC7231 的 application/json。这等同于 Content-Type HTTP 标头的值。如果设置会覆盖 OpenApi 规格中存在的任何值。可以在端点配置中覆盖。		字符串
specificationUri (producer)	OpenApi 规格文件的路径。方案、主机基础路径从此规格中获取，但可以通过组件或端点级别的属性覆盖。如果没有给定组件尝试加载 openapi.json 资源。请注意，在这个组件的组件和端点中定义的主机应包含方案、主机名和可选的 URI 语法中的端口（例如 https://api.example.com:8080）。可以在端点配置中覆盖。	openapi.json	URI
sslContextParameters (security)	自定义组件使用的 TLS 参数。如果没有设置，则默认为 Camel 上下文中设置的 TLS 参数		SSLContextParameters
useGlobalSslContextParameters (security)	启用使用全局 SSL 上下文参数。	false	布尔值

Name	描述	默认值	类型
basicPropertyBinding (advanced)	组件是否应该使用基本属性绑定(Camel 2.x)或较新的属性绑定	false	布尔值
lazyStartProducer (producer)	生成者是否应懒惰启动（在第一个消息中）。通过懒惰启动，您可以使用此选项来允许 CamelContext 和路由在生成者启动期间启动，并导致路由启动失败。通过懒惰启动，启动失败可以在路由信息时通过 Camel 的路由错误处理程序进行处理。请注意，在处理第一个消息时，创建并启动生成者可能需要稍等时间，并延长处理的总处理时间。	false	布尔值

REST OpenApi 端点使用 URI 语法进行配置：

```
rest-openapi:specificationUri#operationId
```

使用以下路径和查询参数：

278.1. 路径参数(2 参数)：

Name	描述	默认值	类型
specificationUri	OpenApi 规格文件的路径。方案、主机基础路径从此规格中获取，但可以通过组件或端点级别的属性覆盖。如果没有提供组件，则组件尝试从 classpath 加载 openapi.json 资源。请注意，在这个组件的组件和端点中定义的主机应包含方案、主机名和可选的 URI 语法中的端口（例如 http://api.example.com:8080）。覆盖组件配置。OpenApi 规格可以通过 file: classpath: http: https: 前缀从不同的源加载。对 https 的支持仅限于使用 JDK 安装的 UrlHandler，因此为 https 设置 TLS/SSL 证书（如设置很多 javax.net.ssl JVM 系统属性）。如何参考 UrlHandler 的 JDK 文档。	openapi.json	URI
operationId	OpenApi 规格中的操作 **所需的** ID。		字符串

278.2. 查询参数(8 参数)：

Name	描述	默认值	类型
basePath (producer)	API basePath，如 /v2。默认是 unset，如果设置会覆盖 OpenApi 规格和组件配置中存在的值。		字符串

Name	描述	默认值	类型
componentName (producer)	执行请求的 Camel 组件的名称。组件必须存在于 Camel registry 中，且必须实现 RestProducerFactory 服务供应商接口。如果没有设置 CLASSPATH，则会搜索实现 RestProducerFactory SPI 的单个组件。覆盖组件配置。		字符串
consumes (producer)	此组件可以使用的有效负载类型。可以是一种类型，如 application/json 或多种类型，如 application/json、application/xml; q=0.5，根据 RFC7231。这等同于 Accept HTTP 标头的值。如果设置覆盖 OpenApi 规格和组件配置中找到的任何值。		字符串
host (producer)	以 https://hostname:port 的形式将 HTTP 请求定向到的方案主机名和端口。可以在端点、组件或 Camel 上下文中的相应 REST 配置中配置。如果您将这个组件命名为 （如 petstore），则 REST 配置首先被查询，下一个 rest-openapi，最后为全局配置指定。如果设置覆盖 OpenApi 规格(RestConfiguration)中找到的任何值。覆盖所有其他配置。		字符串
lazyStartProducer (producer)	生成者是否应懒惰启动 （在第一个消息中）。通过懒惰启动，您可以使用此选项来允许 CamelContext 和路由在生成者启动期间启动，并导致路由启动失败。通过懒惰启动，启动失败可以在路由信息时通过 Camel 的路由错误处理程序进行处理。请注意，在处理第一个消息时，创建并启动生成者可能需要稍等时间，并延长处理的总处理时间。	false	布尔值
generate (producer)	此组件生成的有效负载类型。例如，根据 RFC7231 的 application/json。这等同于 Content-Type HTTP 标头的值。如果设置会覆盖 OpenApi 规格中存在的任何值。覆盖所有其他配置。		字符串
basicPropertyBinding (advanced)	端点是否应该使用基本属性绑定(Camel 2.x)或较新的属性绑定	false	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

第 279 章 SPRING BOOT AUTO-CONFIGURATION

当使用 Spring Boot 时，请确保使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-rest-openapi-starter</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

组件支持 12 个选项，如下所列。

Name	描述	默认值	类型
camel.component.rest-openapi.base-path	API basePath，如 /v2。默认是 unset，如果设置会覆盖 OpenApi 规格中存在的值。		字符串
camel.component.rest-openapi.basic-property-binding	组件是否应该使用基本属性绑定(Camel 2.x)或较新的属性绑定	false	布尔值
camel.component.rest-openapi.bridge-error-handler	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
camel.component.rest-openapi.component-name	执行请求的 Camel 组件的名称。组件必须存在于 Camel registry 中，且必须实现 RestProducerFactory 服务供应商接口。如果没有设置 CLASSPATH，则会搜索实现 RestProducerFactory SPI 的单个组件。可以在端点配置中覆盖。		字符串
camel.component.rest-openapi.consumes	此组件可以使用的有效负载类型。可以是一种类型，如 application/json 或多种类型，如 application/json、application/xml; q=0.5，根据 RFC7231。这等同于 Accept HTTP 标头的值。如果设置会覆盖 OpenApi 规格中找到的任何值。可以在端点配置中覆盖		字符串

Name	描述	默认值	类型
camel.component.rest-openapi.enabled	是否启用 rest-openapi 组件的自动配置。这默认是启用的。		布尔值
camel.component.rest-openapi.host	以 https://hostname:port 的形式将 HTTP 请求定向到的方案主机名和端口。可以在端点、组件或 Camel 上下文中的相应 REST 配置中配置。如果您将这个组件命名为 （如 petstore），则 REST 配置首先被查询，下一个 rest-openapi，最后为全局配置指定。如果设置覆盖 OpenApi 规格(RestConfiguration)中找到的任何值。可以在端点配置中覆盖。		字符串
camel.component.rest-openapi.lazy-start-producer	生成者是否应懒惰启动 （在第一个消息中）。通过懒惰启动，您可以使用此选项来允许 CamelContext 和路由在生成者启动期间启动，并导致路由启动失败。通过懒惰启动，启动失败可以在路由信息时通过 Camel 的路由错误处理程序进行处理。请注意，在处理第一个消息时，创建并启动生成者可能需要稍等时间，并延长处理的总处理时间。	false	布尔值
camel.component.rest-openapi.produces	此组件生成的有效负载类型。例如，根据 RFC7231 的 application/json。这等同于 Content-Type HTTP 标头的值。如果设置会覆盖 OpenApi 规格中存在的任何值。可以在端点配置中覆盖。		字符串
camel.component.rest-openapi.specification-uri	OpenApi 规格文件的路径。方案、主机基础路径从此规格中获取，但可以通过组件或端点级别的属性覆盖。如果没有给定组件尝试加载 openapi.json 资源。请注意，在这个组件的组件和端点中定义的主机应包含方案、主机名和可选的 URI 语法中的端口（例如 https://api.example.com:8080）。可以在端点配置中覆盖。		URI
camel.component.rest-openapi.ssl-context-parameters	自定义组件使用的 TLS 参数。如果没有设置，则默认为 Camel 上下文中设置的 TLS 参数。选项是 org.apache.camel.support.jsse.SSLContextParameters 类型。		字符串
camel.component.rest-openapi.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

第 280 章 示例：PETSTORE

签出 examples 目录中的 camel-example-rest-openapi 项目中的示例。

例如，如果您想要使用 PetStore 提供的 REST API 仅引用 OpenApi 规格中的规格 URI 和所需操作 id，或者下载规格并将其存储为 CLASSPATH 的 openapi.json （在 root 中），它将被自动使用。让我们使用 Undertow 组件来执行对 Spring Boot 的所有请求和 Camel excelent 支持。

以下是 Maven POM 文件中定义的依赖项：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-undertow-starter</artifactId>
</dependency>

<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-rest-openapi-starter</artifactId>
</dependency>
```

首先定义 Undertow 组件和 RestOpenApiComponent ：

```
@Bean
public Component petstore(CamelContext camelContext, UndertowComponent undertow) {
    RestOpenApiComponent petstore = new RestOpenApiComponent(camelContext);
    petstore.setSpecificationUri("https://petstore3.swagger.io/api/v3/openapi.json");
    petstore.setDelegate(undertow);

    return petstore;
}
```

注意

对 Spring Boot 的支持将自动创建 UndertowComponent Spring bean，您可以使用前缀 camel.component.undertow.，使用 application.properties （或 application.yml）进行配置。我们在此处定义 petstore 组件，以便在 Camel 上下文中具有命名组件，我们可以使用它来与 PetStore REST API 交互（如果这是我们使用的唯一 rest-openapi 组件，则可能会以同样的方式配置它（使用 application.properties）。

现在，在应用程序中，我们简单地使用 ProducerTemplate 调用 PetStore REST 方法：

```
@Autowired
ProducerTemplate template;

String getPetJsonById(int petId) {
    return template.requestBodyAndHeaders("petstore:getPetById", null, "petId", petId);
}
```

第 281 章 REST SWAGGER 组件

从 Camel 版本 2.19 开始提供

rest-swagger 配置来自 Swagger (Open API)规格文档的 rest producer，并委派给实现 RestProducerFactory 接口的组件。目前已知的工作组件是：

- http
- http4
- netty4-http
- restlet
- jetty
- undertow

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-rest-swagger</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

281.1. URI 格式

```
rest-swagger:[specificationPath#]operationId
```

其中 operationId 是 Swagger 规格中的操作 ID，specificationPath 是规格的路径。如果没有指定 specificationPath，则默认为 swagger.json。查找机制使用 Camels ResourceHelper 来加载资源，这意味着您可以使用 CLASSPATH 资源(classpath:my-specification.json)、文件

(file:/some/path.json)、web (http://api.example.com/swagger.json)或引用 bean (ref:nameOfBean)或使用 bean (bean:nameOfBean.methodName)的方法来获取规格资源，失败于 Swagger 的资源，失败。

此组件不充当 HTTP 客户端，它将该客户端委派给上述另一个组件。查找机制搜索实施 RestProducerFactory 接口并使用的单一组件。如果 CLASSPATH 包含多个，则应设置 property componentName 来指示要委派哪些组件。

大多数配置都从 Swagger 规范获取，但存在 选项，通过在组件或端点中指定它们来覆盖它们。通常，如果主机 或 basePath 与规格不同，则您只需要覆盖主机或 basePath。

注意

host 参数应包含包含方案、主机名和端口号的绝对 URI，例如：https://api.example.com

使用 componentName，您可以指定用于执行请求的组件，此命名组件需要存在于 Camel 上下文中，并实现所需的 RestProducerFactory 接口 swig-wagonas 处理顶部列出的组件。

如果您没有在组件或端点级别指定 componentName，则会搜索 CLASSPATH 以获得合适的委派。CLASSPATH 上应该只有一个组件，它实现了 RestProducerFactory 接口，以便它正常工作。

此组件的端点 URI 是很宽松的，这意味着除了消息标头外，您还可以将 REST 操作的参数指定为端点参数，对于所有后续调用来说，这些将是恒定的，因此，仅在所有后续调用中持续使用此功能的参数（如 /api/7.13/users/{id} ）。

281.2. 选项

REST Swagger 组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
basePath (producer)	API basePath，如 /v2。默认会被取消设置，如果设置会覆盖 Swagger 规格中存在的值。		字符串

Name	描述	默认值	类型
componentName (producer)	执行请求的 Camel 组件的名称。compnent 必须存在于 Camel registry 中，且必须实现 RestProducerFactory 服务供应商接口。如果没有设置 CLASSPATH，则会搜索实现 RestProducerFactory SPI 的单个组件。可以在端点配置中覆盖。		字符串
consumes (producer)	此组件可以使用的有效负载类型。可以是一种类型，如 application/json 或多种类型，如 application/json、application/xml; q=0.5，根据 RFC7231。这等同于 Accept HTTP 标头的值。如果设置会覆盖 Swagger 规格中找到的任何值。可以在端点配置中覆盖		字符串
host (producer)	以 https://hostname:port 的形式将 HTTP 请求定向到的方案主机名和端口。可以在端点、组件或 Camel 上下文中的 correspoding REST 配置中配置。如果您将这个组件命名为（如 petstore），则 REST 配置将首先查询，rest-swagger 下一个，最后是全局配置。如果设置会覆盖 Swagger 规格中找到的任何值 RestConfiguration。可以在端点配置中覆盖。		字符串
generate (producer)	此组件生成的有效负载类型。例如，根据 RFC7231 的 application/json。这等同于 Content-Type HTTP 标头的值。如果设置覆盖 Swagger 规格中存在的任何值。可以在端点配置中覆盖。		字符串
specificationUri (producer)	Swagger 规格文件的路径。方案、主机基础路径从此规格中获取，但可以使用组件或端点级别的属性覆盖。如果没有给定组件尝试加载 swagger.json 资源。请注意，在这个组件的组件和端点中定义的主机应包含方案、主机名和可选的 URI 语法中的端口（例如 https://api.example.com:8080）。可以在端点配置中覆盖。	swagger.json	URI
sslContextParameters (security)	自定义组件使用的 TLS 参数。如果没有设置，则默认为 Camel 上下文中设置的 TLS 参数		SSLContextParameters
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

REST Swagger 端点使用 URI 语法进行配置：

rest-swagger:specificationUri#operationId

使用以下路径和查询参数：

281.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
specificationUri	Swagger 规格文件的路径。方案、主机基础路径从此规格中获取，但可以使用组件或端点级别的属性覆盖。如果没有给定组件尝试加载 swagger.json 资源。请注意，在这个组件的组件和端点中定义的主机应包含方案、主机名和可选的 URI 语法中的端口（例如 https://api.example.com:8080）。覆盖组件配置。	swagger.json	URI
operationId	Swagger 规格中的操作 **所需的** ID。		字符串

281.2.2. 查询参数(6 参数)：

Name	描述	默认值	类型
basePath (producer)	API basePath，如 /v2。默认会被取消设置，如果设置会覆盖 Swagger 规格和组件配置中存在的值。		字符串
componentName (producer)	执行请求的 Camel 组件的名称。compnent 必须存在于 Camel registry 中，且必须实现 RestProducerFactory 服务供应商接口。如果没有设置 CLASSPATH，则会搜索实现 RestProducerFactory SPI 的单个组件。覆盖组件配置。		字符串
consumes (producer)	此组件可以使用的有效负载类型。可以是一种类型，如 application/json 或多种类型，如 application/json、application/xml; q=0.5，根据 RFC7231。这等同于 Accept HTTP 标头的值。如果设置覆盖了 Swagger 规格和组件配置中找到的任何值。		字符串
host (producer)	以 https://hostname:port 的形式将 HTTP 请求定向到的方案主机名和端口。可以在端点、组件或 Camel 上下文中的 correspoding REST 配置中配置。如果您将这个组件命名为（如 petstore），则 REST 配置将首先查询，rest-swagger 下一个，最后是全局配置。如果设置会覆盖 Swagger 规格中找到的任何值 RestConfiguration。覆盖所有其他配置。		字符串

Name	描述	默认值	类型
generate (producer)	此组件生成的有效负载类型。例如，根据 RFC7231 的 application/json。这等同于 Content-Type HTTP 标头的值。如果设置覆盖 Swagger 规格中存在的任何值。覆盖所有其他配置。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

281.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
camel.component.rest-swagger.base-path	API basePath，如 /v2。默认会被取消设置，如果设置会覆盖 Swagger 规格中存在的值。		字符串
camel.component.rest-swagger.component-name	执行请求的 Camel 组件的名称。compnent 必须存在于 Camel registry 中，且必须实现 RestProducerFactory 服务供应商接口。如果没有设置 CLASSPATH，则会搜索实现 RestProducerFactory SPI 的单个组件。可以在端点配置中覆盖。		字符串
camel.component.rest-swagger.consumes	此组件可以使用的有效负载类型。可以是一种类型，如 application/json 或多种类型，如 application/json、application/xml; q=0.5，根据 RFC7231。这等同于 Accept HTTP 标头的值。如果设置会覆盖 Swagger 规格中找到的任何值。可以在端点配置中覆盖		字符串
camel.component.rest-swagger.enabled	启用 rest-swagger 组件	true	布尔值
camel.component.rest-swagger.host	以 https://hostname:port 的形式将 HTTP 请求定向到的方案主机名和端口。可以在端点、组件或 Camel 上下文中的 correspoding REST 配置中配置。如果您将这个组件命名为（如 petstore），则 REST 配置将首先查询，rest-swagger 下一个，最后是全局配置。如果设置会覆盖 Swagger 规格中找到的任何值 RestConfiguration。可以在端点配置中覆盖。		字符串

Name	描述	默认值	类型
camel.component.rest-swagger.produces	此组件生成的有效负载类型。例如，根据 RFC7231 的 application/json。这等同于 Content-Type HTTP 标头的值。如果设置覆盖 Swagger 规格中存在的任何值。可以在端点配置中覆盖。		字符串
camel.component.rest-swagger.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.rest-swagger.specification-uri	Swagger 规格文件的路径。方案、主机基础路径从此规格中获取，但可以使用组件或端点级别的属性覆盖。如果没有给定组件尝试加载 swagger.json 资源。请注意，在这个组件的组件和端点中定义的主机应包含方案、主机名和可选的 URI 语法中的端口（例如 https://api.example.com:8080）。可以在端点配置中覆盖。		URI
camel.component.rest-swagger.ssl-context-parameters	自定义组件使用的 TLS 参数。如果没有设置，则默认为 Camel 上下文中设置的 TLS 参数。选项是 org.apache.camel.util.jsse.SSLContextParameters 类型。		字符串
camel.component.rest-swagger.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

281.4. 示例：PETSTORE

在 examples 目录中的 camel-example-rest-swagger 项目中签出示例。

例如，如果您想要使用 PetStore 提供的 REST API 仅引用 Swagger 规格中的规格 URI 和所需操作 id，或者下载规格并将其存储为 CLASS .json （在 root 中），它将被自动使用。我们使用 undertow 组件来执行对 Spring Boot 的所有请求和 Camel excelent 支持。

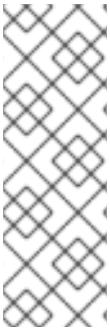
以下是 Maven POM 文件中定义的依赖项：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-undertow-starter</artifactId>
```

```
</dependency>

<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-rest-swagger-starter</artifactId>
</dependency>
```

首先定义 Undertow 组件和 RestSwaggerComponent ：

```
@Bean
public Component petstore(CamelContext camelContext, UndertowComponent undertow) {
    RestSwaggerComponent petstore = new RestSwaggerComponent(camelContext);
    petstore.setSpecificationUri("http://petstore.swagger.io/v2/swagger.json");
    petstore.setDelegate(undertow);

    return petstore;
}
```



注意

对 Spring Boot 的支持将自动创建 UndertowComponent Spring bean，您可以使用前缀 camel.component.undertow.，使用 application.properties （或 application.yml）进行配置。我们在此处定义 petstore 组件，以便在 Camel 上下文中具有命名组件，我们可以使用它来与 PetStore REST API 交互（如果这是我们使用的唯一 rest-swagger 组件，则可能会以同样的方式配置它（使用 application.properties）。

现在，在应用程序中，我们简单地使用 ProducerTemplate 调用 PetStore REST 方法：

```
@Autowired
ProducerTemplate template;

String getPetJsonById(int petId) {
    return template.requestBodyAndHeaders("petstore:getPetById", null, "petId", petId);
}
```

第 282 章 RESTLET 组件

从 Camel 版本 2.0 开始提供

Restlet 组件为消耗和生成 RESTful 资源提供基于 Restlet 的端点。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-restlet</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

另外，您需要在 pom.xml 文件中添加对 restlet maven 存储库的依赖关系：

```
<repository>
  <id>maven-restlet</id>
  <name>Public online Restlet repository</name>
  <url>https://maven.restlet.com</url>
</repository>
```

282.1. URI 格式

```
restlet:restletUrl[?options]
```

restletUrl 格式：

```
protocol://hostname[:port][/resourcePattern]
```

restlet 促进了协议和应用程序顾虑的分离。Restlet Engine 的参考实现支持很多协议。但是，我们只测试了 HTTP 协议。默认端口为端口 80。我们还没有根据协议自动切换默认端口。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

INFO ：在了解标头时，看似 Restlet 区分大小写。例如，要使用 content-type，使用 Content-Type，对于 location 使用 Location 等。

警告

在 Camel 2.14.0 和 2.14.1 中，我们收到有关在 camel-restlet 中丢弃性能的报告。我们在 问题 996 中报告给 Restlet 团队。为了解决这个问题，从 Camel 2.14.2 开始，您可以在 endpoint uris 上将 synchronous=true 设置为选项，或者在 RestletComponent 上将其设置为全局选项，以便所有端点都继承此选项。

282.2. 选项

Restlet 组件支持 23 个选项，如下所列。

Name	描述	默认值	类型
controllerDaemon (consumer)	指明控制器线程应为守护进程（而不是阻塞 JVM 退出）。		布尔值
controllerSleepTimeMs (consumer)	在各个控制之间休眠控制器线程的时间。		整数
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
inboundBufferSize (consumer)	读取消息时的缓冲区的大小。		整数
maxConnectionsPerHost (common)	每个主机(IP 地址)的最大并发连接数。		整数
maxThreads (consumer)	将服务请求的最大线程。		整数
lowThreads (consumer)	确定连接器何时被视为过载的 worker 线程数量。		整数
maxTotalConnections (common)	总的并发连接的最大数量。		整数
minThreads (consumer)	等待服务请求的最小线程。		整数

Name	描述	默认值	类型
outboundBufferSize (consumer)	编写消息时的缓冲区的大小。		整数
persistingConnections (consumer)	表示在调用后是否应保持连接。		布尔值
pipeliningConnections (consumer)	指明是否支持 pipelining 连接。		布尔值
threadMaxIdleTimeMs (consumer)	在收集前，空闲线程等待操作的时间。		整数
useForwardedForHeader (consumer)	查找常见代理和缓存支持的 X-Forwarded-For 标头，并使用它来填充 Request.getClientAddresses （）方法结果。此信息仅适用于您的本地网络中的中介组件。通过设置一个假的标头来轻松地更改其他地址，且不应因为严重的安全检查而被信任。		布尔值
reuseAddress (consumer)	启用/禁用 SO_REUSEADDR 套接字选项。如需了解更多详细信息，请参阅 java.io.ServerSocket SerialreuseAddress 属性。		布尔值
maxQueued (consumer)	如果没有任何 worker 线程可用于服务，则可排队的最大调用数。如果值为 '0'，则不使用队列，并在没有 worker 线程立即可用时拒绝调用。如果值为 '-1'，则使用未绑定队列，并且调用永远不会被拒绝。		整数
disableStreamCache (consumer)	确定是否缓存来自 Restlet 的原始输入流(Camel will read the stream in memory/overflow to file, Stream cache)缓存。默认情况下，Camel 将缓存 Restlet 输入流，以支持多次读取，以确保 Camel 可以从流检索所有数据。但是，当您需要访问原始流时，您可以将此选项设置为 true，例如将其直接流传输到文件或其他持久性存储。DefaultRestletBinding 会将请求输入流复制到流缓存中，如果此选项为 false，则将其放入消息正文中，以便支持多次读取流。	false	布尔值
port (consumer)	为 restlet 消费者路由配置端口号。这允许配置一次，以便为这些使用者重复使用相同的端口。		int
同步 (producer)	是否将同步 Restlet 客户端用于制作者。将这个选项设置为 true 可获得更快的性能，因为它认为 Restlet 同步客户端可以更好地工作。		布尔值
enabledConverters (advanced)	作为完整类名称或简单类名称启用的转换器列表。如果为空或 null，则会自动注册的所有转换器		list
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值

Name	描述	默认值	类型
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Restlet 端点使用 URI 语法进行配置：

```
restlet:protocol:host:port/uriPattern
```

使用以下路径和查询参数：

282.2.1. 路径参数(4 参数)：

Name	描述	默认值	类型
protocol	**必需** 使用的协议是 http 或 https		字符串
主机	**必需** restlet 服务的主机名		字符串
port	**必需** restlet 服务的端口号	80	int
uriPattern	资源模式，如 /customer/id		字符串

282.2.2. 查询参数(18 参数)：

Name	描述	默认值	类型
restletMethod (common)	在制作者端点上，指定要使用的请求方法。在消费者端点上，指定端点只消耗 restletMethod 请求。	GET	方法
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值

Name	描述	默认值	类型
restletMethods (consumer)	指定一个或多个由 restlet 消费者端点提供服务的逗号（如 restletMethods=post、put）的方法。如果指定了 restletMethod 和 restletMethods 选项，则忽略 restletMethod 设置。可能的方法有： ALL、CONNECT、DELETE、GET、HEAD、OPTIONS、PATCH、POST、PUT、TRACE		字符串
disableStreamCache (consumer)	确定是否缓存来自 Restlet 的原始输入流(Camel will read the stream in memory/overflow to file, Stream cache)缓存。默认情况下，Camel 将缓存 Restlet 输入流，以支持多次读取，以确保 Camel 可以从流检索所有数据。但是，当您需要访问原始流时，您可以将此选项设置为 true，例如将其直接流传输到文件或其他持久性存储。DefaultRestletBinding 会将请求输入流复制到流缓存中，如果此选项为 false，则将其放入消息正文中，以便支持多次读取流。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
restletUriPatterns (consumer)	**弃用的** 指定一个或多个 URI 模板由 restlet 消费者端点提供服务，使用 # 表示法引用 Camel Registry 中的列表。如果在端点 URI 中定义了 URI 模式，则端点中定义的 URI 模式和 restletUriPatterns 选项都将满足。		list
connectTimeout (producer)	如果连接是超时，客户端将放弃连接，0 代表无限等待。	30000	int
cookieHandler (producer)	配置 Cookie 处理程序，以维护 HTTP 会话		CookieHandler
socketTimeout (producer)	客户端套接字接收超时，0 代表无限等待。	30000	int
throwExceptionOnFailure (producer)	是否在生成者失败时抛出异常。如果此选项为 false，则 http 状态代码被设置为消息标头，如果其具有错误值，则可以检查该标头。	true	布尔值

Name	描述	默认值	类型
autoCloseStream (producer)	是否使用 restlet 生成者自动关闭流表示，作为来自调用 REST 服务的响应。如果响应被流传输，并且启用了选项 streamRepresentation，则您可能希望从流响应中自动关闭 InputStream，以确保在 Camel Exchange 完成路由时关闭输入流。但是，如果您需要读取 Camel 路由外的流，您可能需要自动关闭流。	false	布尔值
streamRepresentation (producer)	是否支持流表示，作为来自使用 restlet 生成者调用 REST 服务的响应。如果响应被流传输，则可以启用此选项来使用 java.io.InputStream 作为 Camel Message body 上的消息正文。如果使用这个选项，您可能需要启用 autoCloseStream 选项，确保在 Camel Exchange 完成后关闭输入流。但是，如果您需要读取 Camel 路由外的流，您可能需要自动关闭流。	false	布尔值
headerFilterStrategy (advanced)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
restletBinding (advanced)	使用自定义 RestletBinding 和 Camel 消息之间的绑定。		RestletBinding
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
restletRealm (security)	将 restlet 的安全域配置为映射：		Map
sslContextParameters (security)	使用 SSLContextParameters 配置安全性。		SSLContextParameters

282.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 24 个选项，如下所列。

Name	描述	默认值	类型
camel.component.restlet.controller-daemon	指明控制器线程应为守护进程（而不是阻塞 JVM 退出）。		布尔值
camel.component.restlet.controller-sleep-time-ms	在各个控制之间休眠控制器线程的时间。		整数

Name	描述	默认值	类型
camel.component.restlet.disable-stream-cache	确定是否缓存来自 Restlet 的原始输入流(Camel will read the stream in memory/overflow to file, Stream cache)缓存。默认情况下，Camel 将缓存 Restlet 输入流，以支持多次读取，以确保 Camel 可以从流检索所有数据。但是，当您需要访问原始流时，您可以将此选项设置为 true，例如将其直接流传输到文件或其他持久性存储。DefaultRestletBinding 会将请求输入流复制到流缓存中，如果此选项为 false，则将其放入消息正文中，以便支持多次读取流。	false	布尔值
camel.component.restlet.enabled	启用 restlet 组件	true	布尔值
camel.component.restlet.enabled-converters	作为完整类名称或简单类名称启用的转换器列表。如果为空或 null，则会自动注册的所有转换器		list
camel.component.restlet.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		字符串
camel.component.restlet.inbound-buffer-size	读取消息时的缓冲区的大小。		整数
camel.component.restlet.low-threads	确定连接器何时被视为过载的 worker 线程数量。		整数
camel.component.restlet.max-connections-per-host	每个主机(IP 地址)的最大并发连接数。		整数
camel.component.restlet.max-queued	如果没有任何 worker 线程可用于服务，则可排队的最大调用数。如果值为 '0'，则不使用队列，并在没有 worker 线程立即可用时拒绝调用。如果值为 '-1'，则使用未绑定队列，并且调用永远不会被拒绝。		整数
camel.component.restlet.max-threads	将服务请求的最大线程。		整数
camel.component.restlet.max-total-connections	总的并发连接的最大数量。		整数

Name	描述	默认值	类型
camel.component.restlet.min-threads	等待服务请求的最小线程。		整数
camel.component.restlet.outbound-buffer-size	编写消息时的缓冲区的大小。		整数
camel.component.restlet.persisting-connections	表示在调用后是否应保持连接。		布尔值
camel.component.restlet.pipelining-connections	指明是否支持 pipelining 连接。		布尔值
camel.component.restlet.port	为 restlet 消费者路由配置端口号。这允许配置一次，以便为这些使用者重复使用相同的端口。		整数
camel.component.restlet.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.restlet.reuse-address	启用/禁用 SO_REUSEADDR 套接字选项。如需了解更多详细信息，请参阅 java.io.ServerSocket SerialreuseAddress 属性。		布尔值
camel.component.restlet.ssl-context-parameters	使用 SSLContextParameters 配置安全性。选项是 org.apache.camel.util.jsse.SSLContextParameters 类型。		字符串
camel.component.restlet.synchronous	是否将同步 Restlet 客户端用于制作者。将这个选项设置为 true 可获得更快的性能，因为它认为 Restlet 同步客户端可以更好地工作。		布尔值
camel.component.restlet.thread-max-idle-time-ms	在收集前，空闲线程等待操作的时间。		整数
camel.component.restlet.use-forwarded-for-header	查找常见代理和缓存支持的 X-Forwarded-For 标头，并使用它来填充 Request.getClientAddresses （）方法结果。此信息仅适用于您的本地网络中的中介组件。通过设置一个假的标头来轻松地更改其他地址，且不应因为严重的安全检查而被信任。		布尔值

Name	描述	默认值	类型
camel.component.restlet.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

282.4. 消息标头

Name	类型	描述
Content-Type	**字符串**	指定内容类型，可以通过 application/processor 在 OUT 消息上设置。该值是响应消息的 **content-type**。如果没有设置此标头，则内容类型基于 OUT 消息正文的对象类型。在 Camel 2.3 中，如果 Camel IN 消息中指定 Content-Type 标头，则标头的值决定了 Restlet 请求消息的内容类型。否则，默认为 "application/x-www-form-urlencoded'。在 2.3 版本之前，无法更改请求内容类型。
CamelAcceptContentType	**字符串**	从 **Camel 2.9.3 开始，2.10.0：** HTTP Accept 请求标头。
CamelHttpMethod	**字符串**	HTTP 请求方法。这在 IN 消息标头中设置。
CamelHttpQuery	**字符串**	请求 URI 的查询字符串。当 restlet 组件收到请求时，它会被 **DefaultRestletBinding** 设置为 IN 信息。
CamelHttpResponseCode	**字符串或整数**	响应代码可以在应用程序/处理器的 OUT 消息上设置。该值是响应消息的响应代码。如果没有设置此标头，则响应代码由 restlet 运行时引擎设置。
CamelHttpUri	**字符串**	HTTP 请求 URI。这在 IN 消息标头中设置。
CamelRestletLogin	**字符串**	基本身份验证的登录名称。它由应用程序在 IN 消息上设置，并在 Camel 的 restlet 请求标头之前过滤。
CamelRestletPassword	**字符串**	用于基本身份验证的密码名称。它由应用程序在 IN 消息上设置，并在 Camel 的 restlet 请求标头之前过滤。

Name	类型	描述
CamelRestletRequest	**Request（请求）**	Camel 2.8： 所有请求详细信息的 **org.restlet.Request** 对象。
CamelRestletResponse	**响应**	Camel 2.8： **org.restlet.Response** 对象。您可以使用它从 Restlet 创建响应。请参见以下示例。
org.restlet.*		被传播到 Camel IN 标头的 Restlet 消息的属性。
cache-control	**字符串或 List<CacheDirective>**	Camel 2.11： 用户可以使用 String 值或 camel 消息标头中的 CacheDirective of Restlet 设置 cache-control 列表。

282.5. 消息正文

Camel 会将来自外部服务器的 restlet 响应存储在 OUT 正文上。IN 消息中的所有标头都将复制到 OUT 消息，以便在路由过程中保留标头。

282.6. SAMPLES

282.6.1. 带有身份验证的 restlet 端点

以下路由启动一个 restlet 消费者端点，该端点侦听 http://localhost:8080 上的 POST 请求。处理器会创建一个响应，以回显请求正文和 id 标头的值。

URI 查询中的 restletRealm 设置用于在注册表中查找 Realm Map。如果指定了这个选项，restlet 使用者将使用信息来验证用户登录。只有 经过身份验证的 请求才能访问资源。在本例中，我们创建一个作为 registry 的 Spring 应用程序上下文。Realm Map 的 bean ID 应与 restletRealmRef 匹配。

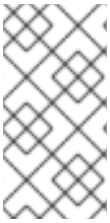
以下示例启动一个 直接 端点，它向 http://localhost:8080 中的服务器发送请求（即 restlet 消费者端

点）。

这是我们需要的。我们已准备好发送请求并尝试出 restlet 组件：

示例客户端使用以下标头向 direct:start-auth 端点发送请求：

- CamelRestletLogin （由 Camel 内部使用）
- CamelRestletPassword （由 Camel 内部使用）
- ID （应用程序标头）



注意

org.apache.camel.restlet.auth.login 和 org.apache.camel.restlet.auth.password 不会被传播为 Restlet 标头。

示例客户端获得类似如下的响应：

```
received [<order foo='1'/>] as an order id = 89531
```

282.6.2. 单个 restlet 端点为多个方法和 URI 模板提供服务（已弃用）

这个功能 已弃用，请不要使用！

可以使用 restletMethods 选项创建单个路由来服务多个 HTTP 方法。这个片段还演示了如何从标头检索请求方法：

除了为多种方法提供服务外，下一个代码片段还演示了如何创建一个端点，该端点使用 restletUriPatterns 选项支持多个 URI 模板。请求 URI 在 IN 消息的标头中可用。如果在端点 URI 中定义了 URI 模式（本例中不是这种情况），则端点中定义的 URI 模式和 restletUriPatterns 选项都将满足。

restletUriPatterns=#uriTemplates 选项引用 Spring XML 配置中定义的 List<String > bean。

```
<util:list id="uriTemplates">
   <value>/users/{username}</value>
   <value>/atom/collection/{id}/component/{cid}</value>
</util:list>
```

282.6.3. 使用 Restlet API 填充响应

从 Camel 2.8 开始提供

您可能希望使用 org.restlet.Response API 来填充响应。这可让您完全访问 Restlet API 和对响应的精细控制。请参阅下面的路由片断，从内联 Camel 处理器生成响应：

使用 Restlet Response API 生成响应

282.6.4. 在组件上配置最大线程

要配置最大线程选项，您必须在组件上执行此操作，例如：

```
<bean id="restlet" class="org.apache.camel.component.restlet.RestletComponent">
  <property name="maxThreads" value="100"/>
</bean>
```

282.6.5. 在 webapp 中使用 Restlet servlet

从 Camel 2.8开始，
可以通过三种方式在 servlet 容器中配置 Restlet 应用程序，并使用子类的 SpringServerServlet 在 Camel 中通过注入 Restlet 组件来启用配置。http://www.restlet.org/documentation/2.0/jee/ext/org/restlet/ext/servlet/ServerServlet.html

在 servlet 容器中使用 Restlet servlet 可让路由配置 URI 中的相对路径（删除硬编码绝对 URI 的限制）和托管 servlet 容器来处理传入的请求（而不是在新端口上生成单独的服务器进程）。

先决条件

- *您需要将对 Spring 扩展的依赖项添加到 Maven pom.xml 文件中的 restlet 中：*
  -

```
<dependency>
  <groupId>org.restlet.jee</groupId>
  <artifactId>org.restlet.ext.spring</artifactId>
  <version>${restlet-version}</version>
</dependency>
```

流程

1. *要配置 Restlet 应用程序，请将以下内容添加到 camel-context.xml 中；*

```
<camelContext>
  <route id="RS_RestletDemo">
    <from uri="restlet:/demo/{id}" />
    <transform>
      <simple>Request type : ${header.CamelHttpMethod} and ID : ${header.id}</simple>
    </transform>
  </route>
</camelContext>

<bean id="RestletComponent" class="org.restlet.Component" />

<bean id="RestletComponentService"
class="org.apache.camel.component.restlet.RestletComponent">
  <constructor-arg index="0">
    <ref bean="RestletComponent" />
  </constructor-arg>
</bean>
```

2. *将以下内容添加到 web.xml 中；*

```
<!-- Restlet Servlet -->
<servlet>
  <servlet-name>RestletServlet</servlet-name>
  <servlet-class>org.restlet.ext.spring.SpringServerServlet</servlet-class>
  <init-param>
    <param-name>org.restlet.component</param-name>
    <param-value>RestletComponent</param-value>
  </init-param>
</servlet>

<servlet-mapping>
  <servlet-name>RestletServlet</servlet-name>
  <url-pattern>/rs/*</url-pattern>
</servlet-mapping>
```

结果

您可以在 http://localhost:8080/mywebapp/rs/demo/1234 中访问部署的路由，其中 localhost:8080 是 servlet 容器的服务器和端口，mywebapp 是部署的 web 应用程序的名称。

您的浏览器将显示以下内容：

> *"Request type : GET and ID : 1234"*

第 283 章 RIBBON 组件

从 Camel 版本 2.18 开始提供

ribbon 组件为客户端侧负载均衡提供 Netflix Ribbon 使用。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-ribbon</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

此组件有助于在使用 ServiceCall EIP 时在客户端上应用负载均衡功能。

283.1. 配置

- programmatic

```
RibbonConfiguration configuration = new RibbonConfiguration();
configuration.addProperties("ServerListRefreshInterval", "250");

RibbonLoadBalancer loadBalancer = new RibbonLoadBalancer(configuration);

from("direct:start")
   .serviceCall()
      .name("myService")
      .loadBalancer(loadBalancer)
      .consulServiceDiscovery()
      .end()
   .to("mock:result");
```

- Spring Boot

application.properties

```
camel.cloud.ribbon.properties[ServerListRefreshInterval] = 250
```

Routes

```
from("direct:start")
    .serviceCall()
        .name("myService")
        .ribbonLoadBalancer()
        .consulServiceDiscovery()
        .end()
    .to("mock:result");
```

- *XML*

```
<route>
    <from uri="direct:start"/>
    <serviceCall name="myService">
        <!-- enable ribbon load balancer -->
        <ribbonLoadBalancer>
            <properties key="ServerListRefreshInterval" value="250"/>
        </ribbonLoadBalancer>
    </serviceCall>
</route>
```

283.2. 另请参阅

- *ServiceCall EIP*

第 284 章 RMI COMPONENT

从 Camel 版本 1.0 开始提供

rmi: 组件将 Exchanges 绑定到 RMI 协议(JRMP)。

由于此绑定只使用 RMI，因此普通的 RMI 规则仍应用于可调用哪些方法。此组件只支持从扩展远程接口的接口执行方法调用的 Exchange。方法中的所有参数都应是 Serializable 或 Remote 对象。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-rmi</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

284.1. URI 格式

```
rmi://rmi-regisitry-host:rmi-registry-port/registry-path[?options]
```

例如：

```
rmi://localhost:1099/path/to/service
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

284.2. 选项

RMI 组件没有选项。

RMI 端点使用 URI 语法进行配置：

```
rmi:hostname:port/name
```

使用以下路径和查询参数：

284.2.1. 路径参数(3 参数)：

Name	描述	默认值	类型
hostname	RMI 服务器的主机名	localhost	字符串
name	绑定到 RMI 服务器时使用的 **必需** 名称		字符串
port	RMI 服务器的端口号	1099	int

284.2.2. 查询参数(6 参数)：

Name	描述	默认值	类型
方法 (common)	您可以设置要调用的方法的名称。		字符串
remoteInterfaces (common)	特定远程接口。		list
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

284.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.rmi.enabled	启用 rmi 组件	true	布尔值
camel.component.rmi.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

284.4. 使用

要调用在 RMI registry 中注册的现有 RMI 服务，请创建一个类似如下的路由：

```
from("pojo:foo").to("rmi://localhost:1099/foo");
```

要在 RMI registry 中绑定现有的 camel 处理器或服务，请定义一个 RMI 端点，如下所示：

```
RmiEndpoint endpoint= (RmiEndpoint) endpoint("rmi://localhost:1099/bar");
endpoint.setRemoteInterfaces(ISay.class);
from(endpoint).to("pojo:bar");
```

请注意，当绑定 RMI 消费者端点时，您必须指定公开的 远程 接口。

在 XML DSL 中，您可以按以下方式从 Camel 2.7 开始：

```
<camel:route>
    <from uri="rmi://localhost:37541/helloServiceBean?
remoteInterfaces=org.apache.camel.example.osgi.HelloService"/>
    <to uri="bean:helloServiceBean"/>
</camel:route>
```

284.5. 另请参阅

- *配置 Camel*
- *组件*

- *端点*
- *开始使用*

第 285 章 ROUTEBOX 组件（已弃用）

从 Camel 版本 2.6 开始提供

已更改的 Routebox 主题

routebox 组件允许创建专用端点，为在自动创建或用户注入的 camel 上下文中托管的 camel 路由提供封装和基于策略的间接服务集合。

Routebox 端点是 camel 端点，可以在 camel 路由上直接调用。routebox 端点执行以下关键功能

- 封装 - 充当黑box，托管存储在内部 camel 上下文中的 camel 路由的集合。内部上下文完全受 routebox 组件的控制，并且是 绑定的 JVM。
- 基于策略间接 - 发送到 routebox 端点的直接有效负载以及根据用户定义的内部路由或分配映射的 camel 路由。
- Exchange propagation - 将 routebox 端点修改的交换转发到 camel 路由的下一段。

routebox 组件支持使用者和制作者端点。

生产者端点有两种类型

- 将传入请求发送到外部 routebox 消费者端点的制作者
- 在内部嵌入式 camel 上下文中直接调用路由的制作者不会向外部消费者发送请求。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-routebox</artifactId>
```

```
    <version>x.x.x</version>
    <!-- use the same version as your Camel core version -->
</dependency>
```

285.1. 对 CAMEL ROUTEBOX 端点的需求

routebox 组件旨在简化需要复杂环境中的集成

- 大量路由和
- 涉及一组广泛的端点技术，需要以不同方式集成

在这种环境中，通常需要通过创建在 camel 路由之间分层来制作集成解决方案，从而有效地将它们组织到

- 粗粒度或更高级别的路由 - 公开为代表集成重点区域的 Routebox 端点的聚合或较低级别路由。例如：

关注区域	粗粒度路由示例
departmentocus	HR 路由、销售路由等
供应链和 B2B 重点	发运路由、填充路由、第三方服务等
技术重点	数据库路由、JMS 路由、Scheduled batch 路由等

- 细粒度路由 - 执行单调和特定业务和/或集成模式的路由。

发送到 grained 粒度路由上的 Routebox 端点的请求可以将请求委派给内粒度路由，以实现特定的集成目标，收集最终结果，并继续进入下一步以及粗粒度路由。

285.2. URI 格式

```
routebox:routeboxname[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

285.3. 选项

RouteBox 组件没有选项。

RouteBox 端点使用 URI 语法进行配置：

```
routebox:routeboxName
```

使用以下路径和查询参数：

285.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
routeboxName	routebox **所需的** 逻辑名称（如队列名称）		字符串

285.3.2. 查询参数(17 参数)：

Name	描述	默认值	类型
assignMap (common)	在 Camel Registry 中代表键的字符串，与类型为 HashMap 的对象值匹配。HashMap 键应包含字符串，可与为交换标头 ROUTE_DISPATCH_KEY 设置的值匹配。HashMap 值应包含内部路由消费者 URI，该 URI 应该定向到哪一请求。		Map
assignStrategy (common)	使用自定义 RouteboxDispatchStrategy，允许使用自定义分配而不是默认值。		RouteboxDispatch 策略
forkContext (common)	是否分叉并创建一个新的 inner CamelContext，而不是重复使用相同的 CamelContext。	true	布尔值
innerProtocol (common)	Routebox 组件内部使用的协议。可以是 Direct 或 SEDA。Routebox 组件目前提供 JVM 绑定的协议。	direct	字符串
queueSize (common)	创建固定大小队列以接收请求。		int

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
pollInterval (consumer)	从 seda 轮询时使用的超时。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让使用者在关闭时更快地做出反应。	1000	long
threads (consumer)	routebox 用来接收请求的线程数。	20	int
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
connectionTimeout (producer)	发送消息时，生成者使用的超时(millis)。	20000	long
sendToConsumer (producer)	指定 Producer 端点是否发送请求到外部 routebox 消费者。如果设置为 false，Producer 会创建一个嵌入的内上下文，并在内部处理请求。	true	布尔值
nerContext (advanced)	在 Camel Registry 中代表键的字符串，与类型为 org.apache.camel.CamelContext 的对象值匹配。如果用户没有提供一个 CamelContext，则会自动创建一个 CamelContext 来部署内部路由。		CamelContext
innerProducerTemplate (advanced)	内部嵌入的 CamelContext 使用的 ProducerTemplate		ProducerTemplate
innerRegistry (advanced)	将自定义 registry 用于内部嵌入式 CamelContext。		容器镜像仓库（Registry）
routeBuilders (advanced)	在 Camel Registry 中代表键的字符串，与类型 List 的对象值匹配。如果用户没有提供内部Context 预处理路由，则 routeBuilders 选项必须作为包含内部路由的 RouteBuilders 的非空列表提供		字符串

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

285.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.routebox.enabled	启用 routebox 组件	true	布尔值
camel.component.routebox.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

285.5. 将消息发送到/从 ROUTEBOX 发送

在发送请求前，需要通过将所需的 URI 参数加载到 Registry 中来正确配置 routebox，如下所示。对于 Spring，如果正确声明了所需的 Bean，则 registry 将自动由 Camel 填充。

285.5.1. 第 1 步：在 Registry 中加载内部路由详情

```
@Override
protected JndiRegistry createRegistry() throws Exception {
    JndiRegistry registry = new JndiRegistry(createJndiContext());

    // Wire the routeDefinitions & dispatchStrategy to the outer camelContext where the
routebox is declared
    List<RouteBuilder> routes = new ArrayList<RouteBuilder>();
    routes.add(new SimpleRouteBuilder());
    registry.bind("registry", createInnerRegistry());
    registry.bind("routes", routes);

    // Wire a dispatch map to registry
    HashMap<String, String> map = new HashMap<String, String>();
    map.put("addToCatalog", "seda:addToCatalog");
    map.put("findBook", "seda:findBook");
    registry.bind("map", map);

    // Alternatively wiring a dispatch strategy to the registry
    registry.bind("strategy", new SimpleRouteDispatchStrategy());
```

```
    return registry;
}

private JndiRegistry createInnerRegistry() throws Exception {
    JndiRegistry innerRegistry = new JndiRegistry(createJndiContext());
    BookCatalog catalogBean = new BookCatalog();
    innerRegistry.bind("library", catalogBean);

    return innerRegistry;
}
...
CamelContext context = new DefaultCamelContext(createRegistry());
```

285.5.2. 第 2 步：可选使用 Dispatch 策略而不是 Dispatch Map

使用分配策略涉及实施接口 org.apache.camel.component.routebox.strategy.RouteboxDispatchStrategy，如下例所示。

```
public class SimpleRouteDispatchStrategy implements RouteboxDispatchStrategy {

    /* (non-Javadoc)
     * @see
org.apache.camel.component.routebox.strategy.RouteboxDispatchStrategy#selectDestination
Uri(java.util.List, org.apache.camel.Exchange)
     */
    public URI selectDestinationUri(List<URI> activeDestinations,
            Exchange exchange) {
        URI dispatchDestination = null;

        String operation = exchange.getIn().getHeader("ROUTE_DISPATCH_KEY", String.class);
        for (URI destination : activeDestinations) {
            if (destination.toASCIIString().equalsIgnoreCase("seda:" + operation)) {
                dispatchDestination = destination;
                break;
            }
        }

        return dispatchDestination;
    }
}
```

285.5.3. 第 2 步：启动 routebox 消费者

在创建路由消费者时，请注意 routeboxUri 中的 # 条目与 CamelContext Registry 中创建的内部 registry, routebuilder list 和 assignStrategy/dispatchMap 匹配。请注意，所有 routebuilder 和关联的路由都在创建的内部上下文中的 routebox 中启动

```
private String routeboxUri = "routebox:multipleRoutes?
innerRegistry=#registry&routeBuilders=#routes&dispatchMap=#map";
```

```
public void testRouteboxRequests() throws Exception {
    CamelContext context = createCamelContext();
    template = new DefaultProducerTemplate(context);
    template.start();

    context.addRoutes(new RouteBuilder() {
        public void configure() {
            from(routeboxUri)
                .to("log:Routes operation performed?showAll=true");
        }
    });
    context.start();

    // Now use the ProducerTemplate to send the request to the routebox
    template.requestBodyAndHeader(routeboxUri, book, "ROUTE_DISPATCH_KEY",
"addToCatalog");
}
```

285.5.4. 第 3 步：使用 routebox producer

当向 routebox 发送请求时，生成者不需要知道内部路由端点 URI，它们只需使用分配策略或分配 Map 调用 Routebox URI 端点，如下所示

需要设置一个特殊的交换标头，名为 ROUTE_DISPATCH_KEY （用于 Dispatch Strategy）的密钥，其键与分配映射中的键匹配，以便可将请求发送到正确的内部路由

```
from("direct:sendToStrategyBasedRoutebox")
    .to("routebox:multipleRoutes?
innerRegistry=#registry&routeBuilders=#routes&dispatchStrategy=#strategy")
    .to("log:Routes operation performed?showAll=true");

from ("direct:sendToMapBasedRoutebox")
    .setHeader("ROUTE_DISPATCH_KEY", constant("addToCatalog"))
    .to("routebox:multipleRoutes?
innerRegistry=#registry&routeBuilders=#routes&dispatchMap=#map")
    .to("log:Routes operation performed?showAll=true");
```

第 286 章 RSS 组件

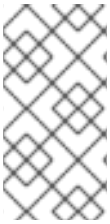
从 Camel 版本 2.0 开始提供

rss: 组件用于轮询 RSS 源。Camel 将每 60 秒轮询一次源。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-rss</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

注意： 组件目前仅支持轮询（耗时）源。



注意

Camel-rss 内部使用在 ServiceMix 上托管的 ROME 补丁版本 来解决一些 OSGi 类加载问题。

286.1. URI 格式

```
rss:rssUri
```

其中 rssUri 是要轮询的 RSS 源的 URI。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

286.2. 选项

RSS 组件没有选项。

RSS 端点使用 URI 语法进行配置：

rss:feedUri

使用以下路径和查询参数：

286.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
feedUri	**需要** 要轮询的源的 URI。		字符串

286.2.2. 查询参数(27 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
feedHeader (consumer)	设置是否将 source 对象添加为标头	true	布尔值
filter (consumer)	设置是否使用过滤或条目。	true	布尔值
lastUpdate (consumer)	设置用于从 atom 源过滤条目的时间戳。这个选项只与 splitEntries 结合使用。		Date
password (consumer)	设置在从 HTTP 源轮询时用于基本身份验证的密码		字符串
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
sortEntries (consumer)	设置是否按公布的日期对条目进行排序。仅在 splitEntries = true 时才有效。	false	布尔值
splitEntries (consumer)	设置条目是否应单独发送，还是将整个源作为单个消息发送	true	布尔值
throttleEntries (consumer)	设置在单个源轮询中识别的所有条目是否应立即发送。如果为 true，则每个 consumer.delay 只处理一个条目。仅在 splitEntries = true 时才适用。	true	布尔值

Name	描述	默认值	类型
用户名 (consumer)	设置在从 HTTP 源轮询时用于基本身份验证的用户名		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service

Name	描述	默认值	类型
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

286.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.rss.enabled	启用 rss 组件	true	布尔值
camel.component.rss.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.dataformat.rss.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.rss.enabled	启用 rss dataformat	true	布尔值

286.4. EXCHANGE 数据类型

Camel 使用 ROME SyndFeed 在交换上初始化 In 正文。根据 splitEntries 标志的值，Camel 返回一个带有 SyndEntrys 的 SyndFeed，或一个带有 SyndEntry s 的 java.util.List。

选项	value	行为
splitEntries	**true**	当前源中的一个条目在交换中设置。
splitEntries	**false**	当前源的整个条目列表在交换中设置。

286.5. 消息标头

标头	描述
CamelRssFeed	整个 **SyncFeed** 对象。

286.6. RSS DATAFORMAT

RSS 组件附带一个 RSS dataformat，可用于在 String （作为 XML）和 ROME RSS 模型对象之间进行转换。

- ***marshal = 从 ROME SyndFeed 到 XML 字符串***
- ***unmarshal = 从 XML 字符串 到 ROME SyndFeed***

使用 RSS dataformat 的路由类似如下： from ("rss:file:src/test/data/rss20.xml?splitEntries=false&consumer.delay=1000").marshal （）.rss （）.to ("mock:marshal");

此功能的目的是使用 Camel 的内置表达式来操作 RSS 消息。如下所示，可以使用 XPath 表达式来过滤 RSS 消息。在以下示例中，在标题中带有 Camel 的 ly 条目将通过过滤器进行。

> ***`from("rss:file:src/test/data/rss20.xml?splitEntries=true&consumer.delay=100").marshal().rss().filter().xpath("//item/title[contains(.,'Camel')]").to("mock:result");`***

提示

如果 RSS 源的 URL 使用查询参数，则此组件将解析它们。例如，如果源使用 alt=rss，则以下示例将解决： from ("rss:http://someserver.com/feeds/posts/default?alt=rss&splitEntries=false&consumer.delay=1000").to"bean:rss"）；

286.7. 过滤条目

您可以使用 XPath 过滤掉条目，如上面的 data format 部分所示。您还可以利用 Camel 的 Bean 集成来实施自己的条件。例如，与上述 XPath 示例对应的过滤器将是：

```
from ("rss:file:src/test/data/rss20.xml?splitEntries=true&consumer.delay=100"). filter
（）.method ("myFilterBean", "titleContainsCamel").to ("mock:result");
```

其自定义 bean 为：

```
public static class FilterBean {
    public boolean titleContainsCamel(@Body SyndFeed feed) {
        SyndEntry firstEntry = (SyndEntry) feed.getEntries().get(0);
return firstEntry.getTitle().contains("Camel");
```

286.8. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*
- *Atom*

第 287 章 RSS DATAFORMAT

从 Camel 版本 2.1 开始提供

RSS 组件附带一个 RSS dataformat，可用于在 String （作为 XML）和 ROME RSS 模型对象之间进行转换。

- marshal = 从 ROME SyndFeed 到 XML 字符串
- unmarshal = 从 XML 字符串 到 ROME SyndFeed

使用它的路由类似如下：

这个功能的目的是为了使用 Camel 的喜欢内置的表达式来处理 RSS 信息。如下所示，可以使用 XPath 表达式来过滤 RSS 消息：

提示

如果 RSS 源的 URL 使用查询参数，则此组件也会了解它们，例如，如果源使用 alt=rss，那么您可以例如，从("rss:http://someserver.com/feeds/posts/default?alt=rss&splitEntries=false&consumer.delay=1000").to ("bean:rss").to ("bean:rss");

287.1. 选项

RSS dataformat 支持 1 个选项，如下所列。

Name	默认值	Java 类型	描述
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

287.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.rss.enabled	启用 rss 组件	true	布尔值
camel.component.rss.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.dataformat.rss.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.rss.enabled	启用 rss dataformat	true	布尔值

ND

第 288 章 RXJAVA2 COMPONENT

从 Camel 版本 2.22 开始提供

基于 RxJava2 的后端，用于 Camel 的被动流组件。

请参阅 camel-streams-component 文档中的更多详细信息。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-rxjava2</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

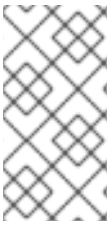
第 289 章 SALESFORCE 组件

从 Camel 版本 2.12 开始提供

此组件支持生成者和消费者端点，以便使用 Java DTO 与 Salesforce 进行通信。
有生成这些 DTO 的 companion maven 插件 Camel Salesforce 插件（请参阅以下）。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-salesforce</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```



注意

希望向组件贡献的开发人员将指示查看 README.md 文件，了解如何启动和设置您的环境以运行集成测试。

289.1. 向 SALESFORCE 进行身份验证

组件支持三个 OAuth 身份验证流：

- OAuth 2.0 Username-Password Flow
- OAuth 2.0 刷新令牌流
- OAuth 2.0 JWT 持有者令牌流

对于每个流，需要设置不同的属性集合：

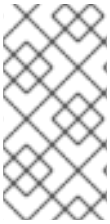
表 289.1. 为每个身份验证流设置的属性

属性	在 Salesforce 上找到它的位置	流
clientId	连接的 App, Consumer Key	所有流
clientSecret	连接的 App, Consumer Secret	username-Password, Refresh Token
userName	salesforce 用户用户名	username-Password, JWT Bearer Token
password	salesforce 用户密码	username-Password
refreshToken	从 OAuth 流回调	刷新令牌
keystore	连接的应用程序、数字证书	JWT Bearer 令牌

组件自动决定您要配置的流，要删除设置 authenticationType 属性。

注意

不建议在生产环境中使用 Username-Password Flow。



注意

JWT Bearer Token Flow 中使用的证书可以是自签名证书。KeyStore 包含证书和私钥必须只包含单个证书私钥条目。

289.2. URI 格式

当用作消费者时，接收流事件时，URI 方案为：

```
salesforce:topic?options
```

当作为制作者使用时，调用 Salesforce RSET API，URI 方案为：

```
salesforce:operationName?options
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

289.3. 传递 SALESFORCE 标头并获取 SALESFORCE 响应标头

使用 Camel 2.21 支持通过入站 消息标头传递 Salesforce 标头，标头名称以 Camel 消息上的 Sforce 或 x-sfdc 开头，并在请求中传递以 Sforce 开头的响应标头。

例如，要获取 API 限制，您可以指定：

```
// in your Camel route set the header before Salesforce endpoint
//...
  .setHeader("Sforce-Limit-Info", constant("api-usage"))
  .to("salesforce:getGlobalObjects")
  .to(myProcessor);

// myProcessor will receive `Sforce-Limit-Info` header on the outbound
// message
class MyProcessor implements Processor {
    public void process(Exchange exchange) throws Exception {
        Message in = exchange.getIn();
        String apiLimits = in.getHeader("Sforce-Limit-Info", String.class);
    }
}
```

289.4. 支持的 SALESFORCE API

组件支持以下 Salesforce API

生产者端点可以使用以下 API：大多数 API 会一次处理一个记录，Query API 可以检索多个记录。

289.4.1. REST API

您可以将以下内容用于 operationName ：

- getVersions - 获取受支持的 Salesforce REST API 版本
- GetResources - 获取可用的 Salesforce REST 资源端点
- getGlobalObjects - 获取所有可用 SObject 类型的元数据

- *getBasicInfo - 获取特定 SObject 类型的基本元数据*
- *getDescription - 获取特定 SObject 类型的综合元数据*
- *getSObject - 使用其 Salesforce Id 获取 SObject*
- *createSObject - 创建 SObject*
- *updateSObject - 使用 Id 更新 SObject*
- *deleteSObject - 使用 Id 删除 SObject*
- *getSObjectWithId - 使用外部（用户定义的）id 字段获取 SObject*
- *upsertSObject - Updates 或使用外部 ID 插入 SObject*
- *deleteSObjectWithId - 使用外部 ID 删除 SObject*
- *query - 运行 Salesforce SOQL 查询*
- *queryMore - 使用从 'query' API 返回的结果链接，获取更多结果（如果大量结果）*
- *搜索 - 运行 Salesforce SOSL 查询*
- *limits - 获取机构 API 用量限制*
- *recent - 获取最新项目*

- Approval - 提交用于批准过程的记录或记录（批量）
- Approvals - 获取所有批准过程列表
- 复合 - 提交至 25 个可能相关的 REST 请求并接收单个响应
- composite-tree - 创建最多 200 个记录，带有父子关系（最多 5 级）
- composite-batch - 提交批处理中的请求组成
- queryAll - 运行 SOQL 查询。它返回因为合并或删除而删除的结果。另外，返回有关归档任务和事件记录的信息。
- getBlobField - 从单个记录检索指定的 blob 字段。
- apexCall - 执行用户定义的 APEX REST API 调用。

例如，以下制作者端点使用 upsertSObject API，并将 'Name' 指定为外部 id 字段。请求消息正文应该是使用 maven 插件生成的 SObject DTO。如果更新现有记录，响应消息为 null，或者 CreateSObjectResult 带有新记录的 id，或者在创建新对象时出现错误列表。

```
...to("salesforce:upsertSObject?sObjectIdName=Name")...
```

289.4.2. 批量 2.0 API

Bulk 2.0 API 通过原始 Bulk API 有一个简化的模型。使用它快速将大量数据加载到 Salesforce 中，或者从 Salesforce 中查询大量数据。数据必须以 CSV 格式提供。Bulk 2.0 的最低 API 版本是 v41.0。Bulk Queries 的最低 API 版本是 v47.0。下面提到的 DTO 类来自 org.apache.camel.component.salesforce.api.dto.bulkv2 软件包。支持以下操作：

- bulk2CreateJob - 创建一个批量作业。在消息正文中提供 作业 实例。

- *bulk2GetJob - 获取现有作业。jobId 参数是必需的。*
- *bulk2CreateBatch - 向作业中添加 CSV 记录的批处理。在消息正文中提供 CSV 数据。第一行必须包含标头。jobId 参数是必需的。*
- *bulk2CloseJob - Close 一个作业。您必须关闭该作业，以便它被处理或中止/删除。jobId 参数是必需的。*
- *bulk2AbortJob - Abort 一个作业。jobId 参数是必需的。*
- *bulk2DeleteJob - 删除作业。jobId 参数是必需的。*
- *bulk2GetSuccessfulResults - 获取作业成功结果。返回的消息正文将包含 CSV 数据的 InputStream。jobId 参数是必需的。*
- *bulk2GetFailedResults - 获取作业失败结果。返回的消息正文将包含 CSV 数据的 InputStream。jobId 参数是必需的。*
- *bulk2GetUnprocessedRecords - 获取作业的未处理记录。返回的消息正文将包含 CSV 数据的 InputStream。jobId 参数是必需的。*
- *bulk2GetAllJobs - 获取所有作业。响应正文是 作业 实例。如果 done 属性为 false，则还有额外的页面来获取，而 nextRecordsUrl 属性包含后续调用的 queryLocator 参数中要设置的值。*
- *bulk2CreateQueryJob - 创建批量查询作业。在消息正文中提供 QueryJob 实例。*
- *bulk2GetQueryJob - 获取批量查询作业。jobId 参数是必需的。*
- *bulk2GetQueryJobResults - 获取批量查询作业结果。jobId 参数是必需的。*

- *bulk2AbortQueryJob - Abort 一个批量查询作业。jobId 参数是必需的。*
- *bulk2DeleteQueryJob - 删除批量查询作业。jobId 参数是必需的。*
- *bulk2GetAllQueryJobs - 获取所有作业。响应正文是一个 QueryJobs 实例。如果 done 属性为 false，则还有额外的页面来获取，而 nextRecordsUrl 属性包含后续调用的 queryLocator 参数中要设置的值。*

289.4.3. REST Bulk (original) API

生产者端点可以使用以下 API：所有作业数据格式，如 xml、csv、zip/xml 和 zip/csv 都被支持。请求和响应必须由路由进行 marshalled/unmarshalled。通常，请求将是一些流源，如 CSV 文件，响应也可以保存到要与请求关联的文件中。

您可以将以下内容用于 operationName ：

- *CreateJob - 创建一个 Salesforce Bulk 作业*
- *GetJob - 使用其 Salesforce Id 获取作业*
- *CloseJob - 关闭作业*
- *AbortJob - Aborts a Job*
- *CreateBatch - 在 Bulk 任务中提交一个批处理*
- *getBatch - 使用 Id 获取批处理*
- *getAllBatches - 获取 Bulk Job Id 的所有批处理*
- *getRequest - 获取批处理的请求数据(XML/CSV)*

- *getResults - 在 Batch 完成后获取结果*
- *createBatchQuery - 从 SOQL 查询创建批处理*
- *getQueryResultIds - 获取 Batch Query 的 Result Ids 列表*
- *getQueryResult - 获取结果 Id*
- *getRecentReports - 通过向报告列表资源发送 GET 请求，获取您最近查看的报告 200 个。*
- *getReportDescription - 以表格或摘要或列表格式检索报告、报告类型和相关元数据。*
- *executeSyncReport - 使用或不更改过滤器同时运行报告，并返回最新的摘要数据。*
- *executeAsyncReport - 使用或不使用过滤器异步运行报告实例，并返回带有或没有详情的摘要数据。*
- *getReportInstances - 为您请求异步运行的报告返回实例列表。列表中的每一项被视为报告的一个单独实例。*
- *getReportResults ：包含运行报告的结果。*

例如，以下制作者端点使用 createBatch API 来创建作业批处理。消息中的主体必须包含一个正文，它可以被转换为 InputStream （通常是 UTF-8 CSV 或 XML 内容），以及来自文件的 Job 和 'contentType' 的标头字段 'jobId'，可以是 XML、CSV、ZIP_XML 或 ZIP_CSV。放置消息正文将包含成功上的 BatchInfo，或者抛出 SalesforceException （错误）。

```
...to("salesforce:createBatchJob")..
```

289.4.4. REST Streaming API

消费者端点可使用以下 sytax 来处理端点，以便在创建/更新时接收 Salesforce 通知。

创建并订阅一个主题

```
from("salesforce:CamelTestTopic?
notifyForFields=ALL&notifyForOperations=ALL&sObjectName=Merchandise__c&updateTopic
=true&sObjectQuery=SELECT Id, Name FROM Merchandise__c")...
```

订阅现有主题

```
from("salesforce:CamelTestTopic&sObjectName=Merchandise__c")...
```

289.4.5. 平台事件

要发送平台事件，请使用 createSObject 操作。并设置消息正文可以是 JSON 字符串或 InputStream，它带有 key-value data5-4-rhacmin，case sObjectName 需要设为事件的 API 名称，或者从 AbstractDTOBase 扩展为事件的适当类名称的类。

例如，使用 DTO：

```
class Order_Event__e extends AbstractDTOBase {
 @JsonProperty("OrderNumber")
 private String orderNumber;
 // ... other properties and getters/setters
}

from("timer:tick")
   .process(exchange -> {
       final Message in = exchange.getIn();
       String orderNumber = "ORD" +
String.valueOf(in.getHeader(Exchange.TIMER_COUNTER));
       Order_Event__e event = new Order_Event__e();
       event.setOrderNumber(orderNumber);
       in.setBody(event);
   })
   .to("salesforce:createSObject");
```

或使用 JSON 事件数据：

```
from("timer:tick")
   .process(exchange -> {
       final Message in = exchange.getIn();
       String orderNumber = "ORD" +
String.valueOf(in.getHeader(Exchange.TIMER_COUNTER));
```

```
        in.setBody("{\"OrderNumber\":\"" + orderNumber + "\"}");
    })
    .to("salesforce:createSObject?sObjectName=Order_Event__e");
```

要接收平台事件，请使用带有 event/ （或 / event/ ）前缀的平台事件的 API 名称的消费者端点，如 salesforce:events/Order_Event__e。从该端点消耗的处理器将分别接收 org.apache.camel.component.salesforce.api.dto.PlatformEvent 对象或 org.cometd.bayeux.Message，具体取决于正文中的 rawPayload 分别为 false 或 true。

例如，在最简单的形式中，消耗一个事件：

```
PlatformEvent event = consumer.receiveBody("salesforce:event/Order_Event__e",
PlatformEvent.class);
```

289.5. 例子

289.5.1. 将文档上传到内容工作空间

使用 Processor 实例，在 Java 中创建 ContentVersion：

```
public class ContentProcessor implements Processor {
    public void process(Exchange exchange) throws Exception {
        Message message = exchange.getIn();

        ContentVersion cv = new ContentVersion();
        ContentWorkspace cw = getWorkspace(exchange);
        cv.setFirstPublishLocationId(cw.getId());
        cv.setTitle("test document");
        cv.setPathOnClient("test_doc.html");
        byte[] document = message.getBody(byte[].class);
        ObjectMapper mapper = new ObjectMapper();
        String enc = mapper.convertValue(document, String.class);
        cv.setVersionDataUrl(enc);
        message.setBody(cv);
    }

    protected ContentWorkspace getWorkSpace(Exchange exchange) {
        // Look up the content workspace somehow, maybe use enrich() to add it to a
        // header that can be extracted here
        ....
    }
}
```

将处理器的输出从处理器提供给 Salesforce 组件：

```
from("file:///home/camel/library")
    .to(new ContentProcessor())    // convert bytes from the file into a ContentVersion SObject
                                   // for the salesforce component
    .to("salesforce:createSObject");
```

289.6. 使用 SALESFORCE LIMITS API

通过 salesforce:limits 操作，您可以从 Salesforce 获取 API 限制，然后对收到的数据进行操作。salesforce:limits 操作的结果映射到 org.apache.camel.component.salesforce.api.dto.Limits 类，并可在自定义处理器或表达式中使用。

例如，请考虑您需要限制 Salesforce 的 API 使用情况，以便每日 API 请求的 10% 用于其他路由。输出消息的正文包含 org.apache.camel.component.salesforce.api.dto.Limits 对象实例，可与 Content Based Router 和 Content Based Router and Spring Expression Language (SpEL) 结合使用，用于选择执行查询。

请注意，在 body.dailyApiRequests.remaining 中保存的整数值乘以 1.0 如何使表达式评估为与浮点算一样的表达式评估，而不包括浮动点，则最终最终产生集成块，从而导致有 0 （消耗一些 API 限制）或 1 (没有 API 限制)。

```
from("direct:querySalesforce")
    .to("salesforce:limits")
    .choice()
    .when(spel("#{1.0 * body.dailyApiRequests.remaining / body.dailyApiRequests.max < 0.1}"))
        .to("salesforce:query?...")
    .otherwise()
        .setBody(constant("Used up Salesforce API limits, leaving 10% for critical routes"))
    .endChoice()
```

289.7. 使用批准

所有属性都与在带有 批准前缀的 Salesforce REST API 中命名完全相同。您可以通过设置 Endpoint 的 approval.PropertyName 设置批准属性，这些属性将用作 template swig-5-4mean，表示正文或标头中没有存在的任何属性都将从 Endpoint 配置中获取。或者，您可以通过将 approval 属性分配给 Registry 中 bean 的引用来设置 Endpoint 上的批准模板。

您还可以在传入的消息标头中使用相同的 approval.PropertyName 提供标头值。

最后，正文可以包含一个 AprovalRequest 或 ApprovalRequest 对象的 Iterable，以作为批处理处理。

请记住，重要的是这三个机制中指定的值的优先级：

1. 正文中的值在任何其他之前具有优先权
2. 消息标头中的值在模板值前具有优先权
3. 如果未指定标头或正文中的其他值，则模板中的值会被设置

例如，要使用标头中的值发送一条用于批准的记录：

给定路由：

```
from("direct:example1")//
    .setHeader("approval.ContextId", simple("${body['contextId']}"))
    .setHeader("approval.NextApproverIds", simple("${body['nextApproverIds']}"))
    .to("salesforce:approval?"//
      + "approval.actionType=Submit"//
      + "&approval.comments=this is a test"//
      + "&approval.processDefinitionNameOrId=Test_Account_Process"//
      + "&approval.skipEntryCriteria=true");
```

您可以发送一条用于批准的记录：

```
final Map<String, String> body = new HashMap<>();
body.put("contextId", accountIds.iterator().next());
body.put("nextApproverIds", userId);

final ApprovalResult result = template.requestBody("direct:example1", body,
ApprovalResult.class);
```

289.8. 使用 SALESFORCE RECENT ITEMS API

要获取最近的项目，请使用 salesforce:recent 操作。此操作返回一个 org.apache.camel.component.salesforce.api.dto.RecentItem 对象的 java.util.List (List<RecentItem>)，它包括 Id、Name 和 Attributes (带有 type 和 url 属性)。您可以通过将 limit 参数设置为要返回的最大记录数来限制返回的项目数量。例如：

```
from("direct:fetchRecentItems")
  to("salesforce:recent")
```

```
        .split().body()
            .log("${body.name} at ${body.attributes.url}");
```

289.9. 使用批准

所有属性都与在带有 批准前缀的 Salesforce REST API 中命名完全相同。您可以通过设置 Endpoint 的 approval.PropertyName 设置批准属性，这些属性将用作 template swig-5-4mean，表示正文或标头中没有存在的任何属性都将从 Endpoint 配置中获取。或者，您可以通过将 approval 属性分配给 Registry 中 bean 的引用来设置 Endpoint 上的批准模板。

您还可以在传入的消息标头中使用相同的 approval.PropertyName 提供标头值。

最后，正文可以包含一个 AprovalRequest 或 ApprovalRequest 对象的 Iterable，以作为批处理处理。

请记住，重要的是这三个机制中指定的值的优先级：

1. 正文中的值在任何其他之前具有优先权
2. 消息标头中的值在模板值前具有优先权
3. 如果未指定标头或正文中的其他值，则模板中的值会被设置

例如，要使用标头中的值发送一条用于批准的记录：

给定路由：

```
from("direct:example1")//
        .setHeader("approval.ContextId", simple("${body['contextId']}"))
        .setHeader("approval.NextApproverIds", simple("${body['nextApproverIds']}"))
        .to("salesforce:approval?"//
            + "approvalActionType=Submit"//
            + "&approvalComments=this is a test"//
            + "&approvalProcessDefinitionNameOrId=Test_Account_Process"//
            + "&approvalSkipEntryCriteria=true");
```

您可以发送一条用于批准的记录：

```
final Map<String, String> body = new HashMap<>();
body.put("contextId", accountIds.iterator().next());
body.put("nextApproverIds", userId);

final ApprovalResult result = template.requestBody("direct:example1", body,
ApprovalResult.class);
```

289.10. 使用 SALESFORCE COMPOSITE API 提交 SOBJECT 树

要创建包括 parent-child 关系在内的最多 200 记录，请使用 salesforce:composite-tree 操作。这要求输入消息中的 org.apache.camel.component.salesforce.api.dto.composite.SObjectTree 实例，并返回输出消息中的相同对象树。树中的 org.apache.camel.component.salesforce.api.dto.AbstractSObjectBase 实例会使用标识符值 (Id 属性) 或对应的 org.apache.camel.component.salesforce.api.dto.composite.SObjectNode 实例进行更新，并在失败时生成 errors。

请注意，对于某些记录操作，一些记录操作可能会成功，因此您可能需要手动检查错误。

使用此功能的最简单方法是使用 camel-salesforce-maven-plugin 生成的 DTOs，但您也可以选择自定义在树中标识每个对象的引用（用于您的数据库中的实例主键）。

让我们来看一个示例：

```
Account account = ...
Contact president = ...
Contact marketing = ...

Account anotherAccount = ...
Contact sales = ...
Asset someAsset = ...

// build the tree
SObjectTree request = new SObjectTree();
request.addObject(account).addChildren(president, marketing);
request.addObject(anotherAccount).addChild(sales).addChild(someAsset);

final SObjectTree response = template.requestBody("salesforce:composite-tree", tree,
SObjectTree.class);
final Map<Boolean, List<SObjectNode>> result = response.allNodes()
                                    .collect(Collectors.groupingBy(SObjectNode::hasErrors));

final List<SObjectNode> withErrors = result.get(true);
```

```
final List<SObjectNode> succeeded = result.get(false);

final String firstId = succeeded.get(0).getId();
```

289.11. 使用 SALESFORCE COMPOSITE API 提交批处理中的多个请求

Composite API batch 操作(composite-batch)允许您在批处理中累积多个请求，然后一次性提交请求，从而节省多个单独请求的往返成本。然后，每个响应都会在带有保留顺序的响应列表中接收，以便第 n 个请求响应位于响应的第 n 个位置。

注意

结果可能与 API 到 API，因此请求的结果以 java.lang.Object 形式提供。在大多数情况下，结果将是 java.util.Map，带有字符串键和值或其他 java.util.Map 作为值。以 JSON 格式发出的请求保存了一些类型信息（即，知道哪个值是字符串以及哪些值是数字），因此通常它们会更友好。请注意，响应在 XML 和 JSON 之间有所不同，这源自来自 Salesforce API 的响应。因此，如果您在不更改响应处理代码的情况下在格式间切换，请小心。

让我们来看一个示例：

```
final String acountId = ...
final SObjectBatch batch = new SObjectBatch("38.0");

final Account updates = new Account();
updates.setName("NewName");
batch.addUpdate("Account", accountId, updates);

final Account newAccount = new Account();
newAccount.setName("Account created from Composite batch API");
batch.addCreate(newAccount);

batch.addGet("Account", accountId, "Name", "BillingPostalCode");

batch.addDelete("Account", accountId);

final SObjectBatchResponse response = template.requestBody("salesforce:composite-batch?
format=JSON", batch, SObjectBatchResponse.class);

boolean hasErrors = response.hasErrors(); // if any of the requests has resulted in either 4xx
or 5xx HTTP status
final List<SObjectBatchResult> results = response.getResults(); // results of three operations
sent in batch

final SObjectBatchResult updateResult = results.get(0); // update result
final int updateStatus = updateResult.getStatusCode(); // probably 204
final Object updateResultData = updateResult.getResult(); // probably null
```

```
final SObjectBatchResult createResult = results.get(1); // create result
@SuppressWarnings("unchecked")
final Map<String, Object> createData = (Map<String, Object>) createResult.getResult();
final String newAccountId = createData.get("id"); // id of the new account, this is for JSON, for
XML it would be createData.get("Result").get("id")

final SObjectBatchResult retrieveResult = results.get(2); // retrieve result
@SuppressWarnings("unchecked")
final Map<String, Object> retrieveData = (Map<String, Object>) retrieveResult.getResult();
final String accountName = retrieveData.get("Name"); // Name of the retrieved account, this is
for JSON, for XML it would be createData.get("Account").get("Name")
final String accountBillingPostalCode = retrieveData.get("BillingPostalCode"); // Name of the
retrieved account, this is for JSON, for XML it would be
createData.get("Account").get("BillingPostalCode")

final SObjectBatchResult deleteResult = results.get(3); // delete result
final int updateStatus = deleteResult.getStatusCode(); // probably 204
final Object updateResultData = deleteResult.getResult(); // probably null
```

289.12. 使用 SALESFORCE COMPOSITE API 提交多个链请求

复合 操作允许提交最多 25 个请求，这些请求可以串联在一起，以便在后续请求中使用之前生成的实例标识符。单个请求和响应与提供的 参考链接。

注意

复合 API 仅支持 JSON 有效负载。

注意

与批处理 API 一样，结果可能会因 API 而异，因此请求的结果会被指定为 java.lang.Object。在大多数情况下，结果将是 java.util.Map，带有字符串键和值或其他 java.util.Map 作为值。以 JSON 格式发出的请求保存了一些类型信息（即，知道哪个值是字符串以及哪些值是数字），因此通常它们会更友好。

让我们来看一个示例：

```
SObjectComposite composite = new SObjectComposite("38.0", true);

// first insert operation via an external id
final Account updateAccount = new TestAccount();
updateAccount.setName("Salesforce");
updateAccount.setBillingStreet("Landmark @ 1 Market Street");
updateAccount.setBillingCity("San Francisco");
updateAccount.setBillingState("California");
updateAccount.setIndustry(Account_IndustryEnum.TECHNOLOGY);
```

```
composite.addUpdate("Account", "001xx000003DIpcAAG", updateAccount,
"UpdatedAccount");

final Contact newContact = new TestContact();
newContact.setLastName("John Doe");
newContact.setPhone("1234567890");
composite.addCreate(newContact, "NewContact");

final AccountContactJunction__c junction = new AccountContactJunction__c();
junction.setAccount__c("001xx000003DIpcAAG");
junction.setContactId__c("@{NewContact.id}");
composite.addCreate(junction, "JunctionRecord");

final SObjectCompositeResponse response = template.requestBody("salesforce:composite?
format=JSON", composite, SObjectCompositeResponse.class);
final List<SObjectCompositeResult> results = response.getCompositeResponse();

final SObjectCompositeResult accountUpdateResult = results.stream().filter(r ->
"UpdatedAccount".equals(r.getReferenceId())).findFirst().get()
final int statusCode = accountUpdateResult.getHttpStatusCode(); // should be 200
final Map<String, ?> accountUpdateBody = accountUpdateResult.getBody();

final SObjectCompositeResult contactCreationResult = results.stream().filter(r ->
"JunctionRecord".equals(r.getReferenceId())).findFirst().get()
```

289.13. 生成 SOQL 查询字符串

org.apache.camel.component.salesforce.api.utils.QueryHelper 包含生成 SOQL 查询的帮助程序方法。例如，若要从 Account SObject 获取所有自定义字段，只需通过调用以下内容即可生成 SOQL SELECT：

```
String allCustomFieldsQuery = QueryHelper.queryToFetchFilteredFieldsOf(new Account(),
SObjectField::isCustom);
```

289.14. CAMEL SALESFORCE MAVEN 插件

此 Maven 插件为 Camel Salesforce 生成一个 DTO。

因此，建议 clientId、clientSecret、userName 和 password 字段在 pom.xml 中设置。该插件应该为其余属性配置，并可通过以下命令来执行：

```
mvn camel-salesforce:generate -DcamelSalesforce.clientId=<clientid> -
DcamelSalesforce.clientSecret=<clientsecret> \
   -DcamelSalesforce.userName=<username> -DcamelSalesforce.password=<password>
```

生成的 DTOs 使用 Jackson 和 XStream 注解。所有 Salesforce 字段类型都支持。默认情况下，日

期和时间字段映射到 java.time.ZonedDateTime，而 picklist 字段则映射到生成的 Java Enumerations。

289.15. 选项

Salesforce 组件支持 31 个选项，如下所列。

Name	描述	默认值	类型
authenticationType (security)	明确使用的身份验证方法，其中一个 USERNAME_PASSWORD、REFRESH_TOKEN 或 JWT。salesforce 组件可以自动确定要从属性集中使用的身份验证方法，将此属性设置为消除任何不确定性。		AuthenticationType
loginConfig (security)	在一个嵌套的 bean 中的所有身份验证配置，也可以直接在组件上设置的所有属性		SalesforceLoginConfig
instanceUrl (security)	身份验证后使用的 Salesforce 实例的 URL，默认从 Salesforce 网站接收了成功身份验证		字符串
loginUrl (security)	用于身份验证的 Salesforce 实例 **所需的** URL，默认设置为 https://login.salesforce.com	https://login.salesforce.com	字符串
clientId (security)	**必需** 在 Salesforce 实例设置中配置的连接应用程序的 OAuth 消费者密钥。通常，需要配置连接的应用程序，但可以通过安装软件包来提供。		字符串
clientSecret (security)	在 Salesforce 实例设置中配置的已连接应用程序的 OAuth 消费者 Secret。		字符串
keystore (security)	在 OAuth JWT 流中使用的密钥存储参数。KeyStore 应该只包含一个带有私钥和证书的条目。salesforce 不会验证证书链，因此这很容易是自签名证书。确保将证书上传到对应的连接的应用程序。		KeyStoreParameters
refreshToken (security)	刷新令牌已在刷新令牌 OAuth 流中获取。一个需要设置 Web 应用程序并配置回调 URL 以接收刷新令牌，或使用 https://login.salesforce.com/services/oauth2/success 或 https://test.salesforce.com/services/oauth2/success 的内置回调进行配置，然后从流末尾的 URL 中重试 refresh_token。请注意，在开发组织中，Salesforce 允许在 localhost 处托管回调 Web 应用程序。		字符串

Name	描述	默认值	类型
用户名 (security)	OAuth 流中使用的用户名，以获取对访问令牌的访问权限。使用密码 OAuth 流很容易，但通常应该避免它，因为它被视为不如其他流的安全性。		字符串
密码 (security)	OAuth 流中使用的密码，以获取对访问令牌的访问权限。使用密码 OAuth 流很容易，但通常应该避免它，因为它被视为不如其他流的安全性。如果使用，请确保将安全令牌附加到密码的末尾。		字符串
lazyLogin (security)	如果设置为 true，则组件可以防止组件通过启动组件向 Salesforce 进行身份验证。您通常会将其设置为（默认） false，并提前进行身份验证，并立即了解任何身份验证问题。	false	布尔值
config (common)	全局端点配置 - 使用 设置所有端点通用的值		SalesforceEndpoint Config
httpClientProperties (common)	用于设置可以在底层 HTTP 客户端上配置的任何属性。查看 SalesforceHttpClient 的属性，以及适用于所有可用选项的 Jetty HttpClient。		Map
longPollingTransport Properties (common)	用于在流传输 api 使用的 BayeuxClient (CometD)上使用的 LongPollingTransport 上配置的任何属性		Map
sslContextParameters (security)	要使用的 SSL 参数，请参阅所有可用选项的 SSLContextParameters 类。		SSLContextParameters
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数	false	布尔值
httpProxyHost (proxy)	要使用的 HTTP 代理服务器的主机名。		字符串
httpProxyPort (proxy)	要使用的 HTTP 代理服务器的端口号。		整数
httpProxyUsername (security)	用于对 HTTP 代理服务器进行身份验证的用户名。		字符串
httpProxyPassword (security)	用于对 HTTP 代理服务器进行身份验证的密码。		字符串
isHttpProxySocks4 (proxy)	如果设置为 true，则将 HTTP 代理配置为用作 SOCKS4 代理。	false	布尔值

Name	描述	默认值	类型
isHttpProxySecure (security)	如果设置为 false，则禁用在访问 HTTP 代理时使用 TLS。	true	布尔值
httpProxyIncluded Addresses (proxy)	应该使用 HTTP 代理服务器的地址列表。		Set
httpProxyExcluded Addresses (proxy)	不应使用 HTTP 代理服务器的地址列表。		Set
httpProxyAuthUri (security)	用于针对 HTTP 代理服务器进行身份验证，需要与代理服务器的 URI 匹配，以便使用 httpProxyUsername 和 httpProxyPassword 进行身份验证。		字符串
httpProxyRealm (security)	代理服务器的域，用于针对 HTTP 代理服务器抢占 Basic/Digest 身份验证方法。		字符串
httpProxyUseDigest Auth (security)	如果设置为 true Digest 身份验证，则在向 HTTP 代理进行身份验证时使用，否则将使用基本授权方法	false	布尔值
packages (common)	在哪些软件包中是生成的 DTO 类。通常，类将使用 camel-salesforce-maven-plugin 生成。如果使用生成的 DTOs 来获得在 parameters/header 值中使用简短 SObject 名称的好处，则设置它。		string[]
queryLocator (common)	当查询结果比单个调用检索更多记录时，查询 salesforce 提供的查询查找器。在后续调用中使用这个值来检索额外的记录。		字符串
jobType (common)	仅获取有关与指定作业类型匹配的作业信息。可能的值有： **Classic** Bulk API 作业（包括查询作业和 ingest 作业）。 **V2Query** Bulk API 2.0 查询作业。 **V2Ingest** Bulk API 2.0 ingest （上传和 upsert）作业。		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Salesforce 端点使用 URI 语法进行配置：

```
salesforce:operationName:topicName
```

使用以下路径和查询参数：

289.15.1. 路径参数(2 参数)：

Name	描述	默认值	类型
operationName	要使用的操作。有 59 enums，值可以是：getVersions, getResources, getGlobalObjects, getBasicInfo, getDescription, getSObject, createSObject, updateSObject, deleteSObject, getSObject, getSObjectWithId, upsertSObject, deleteSObjectWithId, getBlobField, query, queryMore, queryAll, search, apexCall, recent, createJob, getJob, closeJob, abortJob, createBatch, getBatch, getAllBatches, getRequest, getResults, createBatchQuery, getQueryResultIds, getQueryResult, getRecentReports, getReportDescription, executeSyncReport, executeAsyncReport, getReportInstances, getReportResults, limits, approval, approvals, composite-tree, composite-batch, composite, bulk2GetAllJobs, bulk2CreateJob, bulk2GetJob, bulk2CreateBatch, bulk2CloseJob, bulk2AbortJob, bulk2DeleteJob, bulk2GetSuccessfulResults, bulk2GetFailedResults, bulk2GetUnprocessedRecords, bulk2GetUnprocessedRecords, bulk2CreateQueryJob, bulk2GetQueryJob, bulk2GetAllQueryJobs, bulk2GetQueryJobs, bulk2AbortQueryJob, bulk2DeleteQueryJob		OperationName
topicName	要使用的主题的名称		字符串

289.15.2. 查询参数(46 参数)：

Name	描述	默认值	类型
apexMethod (common)	APEX 方法名称		字符串
apexQueryParams (common)	查询 APEX 方法的参数		Map
apexUrl (common)	APEX 方法 URL		字符串

Name	描述	默认值	类型
apiVersion (common)	salesforce API 版本，默认为 SalesforceEndpointConfig.DEFAULT_VERSION		字符串
backoffIncrement (common)	对于流连接重启尝试超过 CometD auto-reconnect 的 backoff 间隔递增。		long
batchId (common)	批量 API 批处理 ID		字符串
contentType (common)	批量 API 内容类型, XML, CSV, ZIP_XML, ZIP_CSV 之一		ContentType
defaultReplayId (common)	如果没有在 initialReplayIdMap 中找到值，则默认 replayId 设置		Long
格式 (common)	用于 Salesforce API 调用的 payload 格式(JSON 或 XML)默认为 JSON		PayloadFormat
httpclient (common)	自定义 Jetty Http 客户端用于连接到 Salesforce。		SalesforceHttpClient
includeDetails (common)	在 Salesforce1 分析报告中包括详细信息，默认为 false。		布尔值
initialReplayIdMap (common)	重播 ID 从每个频道名称开始。		Map
instanceId (common)	Salesforce1 Analytics 报告执行实例 ID		字符串
jobId (common)	批量 API 作业 ID		字符串
jobType (common)	仅获取有关与指定作业类型匹配的作业信息。可能的值有： **Classic** Bulk API 作业（包括查询作业和 ingest 作业）。 **V2Query** Bulk API 2.0 查询作业。 **V2Ingest** Bulk API 2.0 ingest （上传和 upsert）作业。		字符串
limit (common)	返回的记录数量的限制。适用于某些 API，请查看 Salesforce 文档。		整数
maxBackoff (common)	Streaming connection restart attempt for failures beyond CometD auto-reconnect 的最大 backoff 间隔。		long

Name	描述	默认值	类型
notFoundBehaviour (common)	设置从 Salesforce API 收到的 404 未找到状态的行为。如果正文设置为 NULL NotFoundBehaviour#NULL，或者应该在交换 NotFoundBehaviour CPUfreqEXCEPTION 上发送异常。		NotFoundBehaviour
notifyForFields (common)	notify for fields, options are ALL, REFERENCED, SELECT, WHERE		NotifyForFieldsEnum
notifyForOperationCreate (common)	notify 用于 create 操作，默认为 false (API version = 29.0)		布尔值
notifyForOperationDelete (common)	notify 用于 delete 操作，默认为 false (API version = 29.0)		布尔值
notifyForOperations (common)	notify for operations, options are ALL, CREATE, EXTENDED, UPDATE (API 版本 29.0)		NotifyForOperations Enum
notifyForOperationUndelete (common)	notify 为 un-delete 操作，默认为 false (API version = 29.0)		布尔值
notifyForOperationUpdate (common)	notify 用于 update 操作，默认为 false (API version = 29.0)		布尔值
ObjectMapper (common)	自定义 Jackson ObjectMapper 在序列化/撤销 Salesforce 对象时使用。		ObjectMapper
queryLocator (common)	当查询结果比单个调用检索更多记录时，查询 salesforce 提供的查询查找器。在后续调用中使用这个值来检索额外的记录。		字符串
rawPayload (common)	将原始有效负载字符串用于请求和响应(JSON 或 XML 根据格式)，而不是 DTOs，默认为 false	false	布尔值
reportId (common)	Salesforce1 Analytics 报告 Id		字符串
reportMetadata (common)	Salesforce1 Analytics 报告用于过滤的元数据		ReportMetadata
resultId (common)	批量 API 结果 ID		字符串
serializeNulls (common)	如果给定 DTO 的 NULL 值被序列化为空(NULL)值。这只会影响 JSON 数据格式。	false	布尔值

Name	描述	默认值	类型
sObjectBlobFieldName (common)	SObject blob 字段名称		字符串
sObjectClass (common)	完全限定的 SObject 类名称，通常使用 camel-salesforce-maven-plugin 生成		字符串
sObjectFields (common)	用于检索的 SObject 字段		字符串
sObjectId (common)	API 需要 SObject ID		字符串
sObjectIdName (common)	SObject 外部 ID 字段名称		字符串
sObjectIdValue (common)	SObject 外部 ID 字段值		字符串
sObjectName (common)	SObject 名称（如果 API 需要或支持）		字符串
sObjectQuery (common)	salesforce SOQL 查询字符串		字符串
sObjectSearch (common)	salesforce SOSL 搜索字符串		字符串
updateTopic (common)	使用 Streaming API 时是否更新现有的 Push Topic，默认为 false	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
replayId (consumer)	订阅时要使用的 replayId 值		Long
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler

Name	描述	默认值	类型
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

289.16. SPRING BOOT AUTO-CONFIGURATION

组件支持 85 个选项，如下所列。

Name	描述	默认值	类型
camel.component.salesforce.authentication-type	明确使用的身份验证方法，其中一个 USERNAME_PASSWORD、REFRESH_TOKEN 或 JWT。salesforce 组件可以自动确定要从属性集中使用的身份验证方法，将此属性设置为消除任何不确定性。		AuthenticationType
camel.component.salesforce.client-id	在 Salesforce 实例设置中配置的已连接应用程序的 OAuth 消费者密钥。通常，需要配置连接的应用程序，但可以通过安装软件包来提供。		字符串
camel.component.salesforce.client-secret	在 Salesforce 实例设置中配置的已连接应用程序的 OAuth 消费者 Secret。		字符串
camel.component.salesforce.config.apex-method	APEX 方法名称		字符串
camel.component.salesforce.config.apex-query-params	查询 APEX 方法的参数		Map
camel.component.salesforce.config.apex-url	APEX 方法 URL		字符串
camel.component.salesforce.config.api-version	salesforce API 版本，默认为 SalesforceEndpointConfig.DEFAULT_VERSION		字符串

Name	描述	默认值	类型
camel.component.salesforce.config.approval	Approval API 的批准请求。@param approval		ApprovalRequest
camel.component.salesforce.config.approval-action-type	代表要执行的操作类型： Submit、Approve 或 Reject。@param actionType		ApprovalRequest$Action
camel.component.salesforce.config.approval-comments	要添加到与此请求关联的历史记录步骤的注释。@param 注释		字符串
camel.component.salesforce.config.approval-context-actor-id	请求批准记录的提交者 ID。@param contextActorId		字符串
camel.component.salesforce.config.approval-context-id	正在处理的项目的 ID。@param contextId		字符串
camel.component.salesforce.config.approval-next-approver-ids	如果进程需要指定下一个批准，则需要为下一个请求分配用户的 ID。@param nextApproverIds		list
camel.component.salesforce.config.approval-process-definition-name-or-id	进程定义的开发人员名称或 ID。@param processDefinitionNameOrId		字符串
camel.component.salesforce.config.approval-skip-entry-criteria	如果进程定义名称或 ID 不为空，则决定是否评估进程的条目条件(true)或 not (false)。如果没有指定进程定义名称或 ID，则忽略此参数，并根据进程顺序进行标准评估。默认情况下，如果没有由此请求设置，则条目条件不会被跳过。@param skipEntryCriteria		布尔值
camel.component.salesforce.config.backoff-increment	对于流连接重启尝试超过 CometD auto-reconnect 的 backoff 间隔递增。		Long

Name	描述	默认值	类型
camel.component.salesforce.config.batch-id	批量 API 批处理 ID		字符串
camel.component.salesforce.config.content-type	批量 API 内容类型, XML, CSV, ZIP_XML, ZIP_CSV 之一		ContentType
camel.component.salesforce.config.default-replay-id	如果没有在 initialReplayIdMap 中找到值，则默认 replayId 设置		Long
camel.component.salesforce.config.format	用于 Salesforce API 调用的 payload 格式(JSON 或 XML)默认为 JSON		PayloadFormat
camel.component.salesforce.config.http-client	自定义 Jetty Http 客户端用于连接到 Salesforce。		SalesforceHttpClient
camel.component.salesforce.config.include-details	在 Salesforce1 分析报告中包括详细信息，默认为 false。		布尔值
camel.component.salesforce.config.initial-replay-id-map	重播 ID 从每个频道名称开始。		Map
camel.component.salesforce.config.instance-id	Salesforce1 Analytics 报告执行实例 ID		字符串
camel.component.salesforce.config.job-id	批量 API 作业 ID		字符串
camel.component.salesforce.config.limit	返回的记录数量的限制。适用于某些 API，请查看 Salesforce 文档。		整数
camel.component.salesforce.config.max-backoff	Streaming connection restart attempt for failures beyond CometD auto-reconnect 的最大 backoff 间隔。		Long

Name	描述	默认值	类型
camel.component.salesforce.config.not-found-behaviour	设置从 Salesforce API 收到的 404 未找到状态的行为。如果正文设置为 NULL NotFoundBehaviour#NULL，或者应该在交换 NotFoundBehaviour CPUfreqEXCEPTION 上发送异常。		NotFoundBehaviour
camel.component.salesforce.config.notify-for-fields	notify for fields, options are ALL, REFERENCED, SELECT, WHERE		NotifyForFieldsEnum
camel.component.salesforce.config.notify-for-operation-create	notify 用于 create 操作，默认为 false (API version = 29.0)		布尔值
camel.component.salesforce.config.notify-for-operation-delete	notify 用于 delete 操作，默认为 false (API version = 29.0)		布尔值
camel.component.salesforce.config.notify-for-operation-undelete	notify 为 un-delete 操作，默认为 false (API version = 29.0)		布尔值
camel.component.salesforce.config.notify-for-operation-update	notify 用于 update 操作，默认为 false (API version = 29.0)		布尔值
camel.component.salesforce.config.notify-for-operations	notify for operations, options are ALL, CREATE, EXTENDED, UPDATE (API 版本 29.0)		NotifyForOperations Enum
camel.component.salesforce.config.object-mapper	自定义 Jackson ObjectMapper 在序列化/撤销 Salesforce 对象时使用。		ObjectMapper
camel.component.salesforce.config.raw-payload	将原始有效负载字符串用于请求和响应(JSON 或 XML 根据格式)，而不是 DTOs，默认为 false	false	布尔值
camel.component.salesforce.config.report-id	Salesforce1 Analytics 报告 Id		字符串

Name	描述	默认值	类型
camel.component.salesforce.config.report-metadata	Salesforce1 Analytics 报告用于过滤的元数据		ReportMetadata
camel.component.salesforce.config.result-id	批量 API 结果 ID		字符串
camel.component.salesforce.config.s-object-blob-field-name	SObject blob 字段名称		字符串
camel.component.salesforce.config.s-object-class	完全限定的 SObject 类名称，通常使用 camel-salesforce-maven-plugin 生成		字符串
camel.component.salesforce.config.s-object-fields	用于检索的 SObject 字段		字符串
camel.component.salesforce.config.s-object-id	API 需要 SObject ID		字符串
camel.component.salesforce.config.s-object-id-name	SObject 外部 ID 字段名称		字符串
camel.component.salesforce.config.s-object-id-value	SObject 外部 ID 字段值		字符串
camel.component.salesforce.config.s-object-name	SObject 名称（如果 API 需要或支持）		字符串
camel.component.salesforce.config.s-object-query	salesforce SOQL 查询字符串		字符串
camel.component.salesforce.config.s-object-search	salesforce SOSL 搜索字符串		字符串
camel.component.salesforce.config.serialize-nulls	如果给定 DTO 的 NULL 值被序列化为空(NULL)值。这只会影响 JSON 数据格式。	false	布尔值

Name	描述	默认值	类型
camel.component.salesforce.config.update-topic	使用 Streaming API 时是否更新现有的 Push Topic，默认为 false	false	布尔值
camel.component.salesforce.enabled	启用 salesforce 组件	true	布尔值
camel.component.salesforce.http-client-properties	用于设置可以在底层 HTTP 客户端上配置的任何属性。查看 SalesforceHttpClient 的属性，以及适用于所有可用选项的 Jetty HttpClient。		Map
camel.component.salesforce.http-proxy-auth-uri	用于针对 HTTP 代理服务器进行身份验证，需要与代理服务器的 URI 匹配，以便使用 httpProxyUsername 和 httpProxyPassword 进行身份验证。		字符串
camel.component.salesforce.http-proxy-excluded-addresses	不应使用 HTTP 代理服务器的地址列表。		Set
camel.component.salesforce.http-proxy-host	要使用的 HTTP 代理服务器的主机名。		字符串
camel.component.salesforce.http-proxy-included-addresses	应该使用 HTTP 代理服务器的地址列表。		Set
camel.component.salesforce.http-proxy-password	用于对 HTTP 代理服务器进行身份验证的密码。		字符串
camel.component.salesforce.http-proxy-port	要使用的 HTTP 代理服务器的端口号。		整数
camel.component.salesforce.http-proxy-realm	代理服务器的域，用于针对 HTTP 代理服务器抢占 Basic/Digest 身份验证方法。		字符串
camel.component.salesforce.http-proxy-use-digest-auth	如果设置为 true Digest 身份验证，则在向 HTTP 代理进行身份验证时使用，否则将使用基本授权方法	false	布尔值

Name	描述	默认值	类型
camel.component.salesforce.http-proxy-username	用于对 HTTP 代理服务器进行身份验证的用户名。		字符串
camel.component.salesforce.instance-url	身份验证后使用的 Salesforce 实例的 URL，默认从 Salesforce 网站接收了成功身份验证		字符串
camel.component.salesforce.is-http-proxy-secure	如果设置为 false，则禁用在访问 HTTP 代理时使用 TLS。	true	布尔值
camel.component.salesforce.is-http-proxy-socks4	如果设置为 true，则将 HTTP 代理配置为用作 SOCKS4 代理。	false	布尔值
camel.component.salesforce.keystore	在 OAuth JWT 流中使用的密钥存储参数。KeyStore 应该只包含一个带有私钥和证书的条目。salesforce 不会验证证书链，因此这很容易是自签名证书。确保将证书上传到对应的连接的应用程序。选项是 org.apache.camel.util.jsse.KeyStoreParameters 类型。		字符串
camel.component.salesforce.lazy-login	如果设置为 true，则组件可以防止组件通过启动组件向 Salesforce 进行身份验证。您通常会将其设置为（默认） false，并提前进行身份验证，并立即了解任何身份验证问题。	false	布尔值
camel.component.salesforce.login-config.client-id	salesforce connected application Consumer Key		字符串
camel.component.salesforce.login-config.client-secret	salesforce connected application Consumer Secret		字符串
camel.component.salesforce.login-config.instance-url			字符串
camel.component.salesforce.login-config.keystore	密钥存储的密钥存储参数，其中包含 OAuth 2.0 JWT Bearer Token 流所需的证书和私钥。		KeyStoreParameters

Name	描述	默认值	类型
camel.component.salesforce.login-config.lazy-login	用于启用/禁用 lazy OAuth 的标记，默认为 false。启用后，在第一个 API 调用前，不会进行 OAuth 令牌检索或生成		布尔值
camel.component.salesforce.login-config.login-url	salesforce 登录 URL，默认为 https://login.salesforce.com		字符串
camel.component.salesforce.login-config.password	salesforce account password		字符串
camel.component.salesforce.login-config.refresh-token	salesforce connected application Consumer token		字符串
camel.component.salesforce.login-config.type			AuthenticationType
camel.component.salesforce.login-config.user-name	salesforce 帐户用户名		字符串
camel.component.salesforce.login-url	用于身份验证的 Salesforce 实例的 URL，默认设置为 https://login.salesforce.com	https://login.salesforce.com	字符串
camel.component.salesforce.long-polling-transport-properties	用于在流传输 api 使用的 BayeuxClient (CometD)上使用的 LongPollingTransport 上配置的任何属性		Map
camel.component.salesforce.packages	在哪些软件包中是生成的 DTO 类。通常，类将使用 camel-salesforce-maven-plugin 生成。如果使用生成的 DTOs 来获得在 parameters/header 值中使用简短 SObject 名称的好处，则设置它。		string[]
camel.component.salesforce.password	OAuth 流中使用的密码，以获取对访问令牌的访问权限。使用密码 OAuth 流很容易，但通常应该避免它，因为它被视为不如其他流的安全性。如果使用，请确保将安全令牌附加到密码的末尾。		字符串

Name	描述	默认值	类型
camel.component.salesforce.refresh-token	刷新令牌已在刷新令牌 OAuth 流中获取。一个需要设置 Web 应用程序并配置回调 URL 以接收刷新令牌，或使用 https://login.salesforce.com/services/oauth2/success 或 https://test.salesforce.com/services/oauth2/success 的内置回调进行配置，然后从流末尾的 URL 中重试 refresh_token。请注意，在开发组织中，Salesforce 允许在 localhost 处托管回调 Web 应用程序。		字符串
camel.component.salesforce.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.salesforce.ssl-context-parameters	要使用的 SSL 参数，请参阅所有可用选项的 SSLContextParameters 类。选项是 org.apache.camel.util.jsse.SSLContextParameters 类型。		字符串
camel.component.salesforce.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数	false	布尔值
camel.component.salesforce.user-name	OAuth 流中使用的用户名，以获取对访问令牌的访问权限。使用密码 OAuth 流很容易，但通常应该避免它，因为它被视为不如其他流的安全性。		字符串

第 290 章 SAP 组件

SAP 组件是由十个不同的 SAP 组件组成的软件包。有远程功能调用(RFC)组件支持 sRFC、tRFC 和 qRFC 协议；存在用于以 IDoc 格式使用消息的 IDoc 组件。组件使用 SAP Java Connector (SAP JCo) 库来促进与 SAP 和 SAP IDoc 库的双向通信，以 Intermediate Document (IDoc)格式传输文档。

290.1. 概述

依赖项

Maven 用户需要把以下依赖项添加到其 pom.xml 文件中，才能使用此组件：

```
<dependency>
  <groupId>org.fusesource</groupId>
  <artifactId>camel-sap</artifactId>
  <version>x.x.x</version>
<dependency>
```

SAP 组件的其他平台限制

由于 SAP 组件依赖于第三方 JCo 3 和 IDoc 3 库，因此只能安装到这些库支持的平台上。

有关支持的库版本和平台限制的详情，请参阅 Red Hat JBoss Fuse 支持的配置。

SAP JCo 和 SAP IDoc 库

使用 SAP 组件的先决条件是 SAP Java Connector (SAP JCo)库和 SAP IDoc 库被安装到 Java 运行时的 lib/ 目录中。

您可以从 SAP Service Marketplace 下载适用于您的目标操作系统的适当 SAP 库集合。

注意

您必须有一个 SAP Service Marketplace 帐户 才能下载和使用这些库。

库文件的名称因目标操作系统而异，如 表 290.1 “所需的 SAP 库” 所示。

表 290.1. 所需的 SAP 库

SAP 组件	Linux 和 UNIX	Windows
SAP JCo 3	**sapjco3.jar** **libsapjco3.so**	**sapjco3.jar** **sapjco3.dll**
SAP IDoc	**sapidoc3.jar**	**sapidoc3.jar**

如需更多信息，请参阅 SAP Java Connector 文档。

```
:experimental:
// Standard document attributes to be used in the documentation
//
// The following are shared by all documents
:toc:
:toclevels: 4
:numbered:
```

290.2. 安装所需的 SAP LIBRARIES

290.2.1. 在 Fuse OSGi 容器中部署

您可以将 SAP JCo 库和 SAP IDoc 库安装到 JBoss Fuse OSGi 容器中，如下所示：

1. 从 SAP Service Marketplace 下载 SAP JCo 库和 SAP IDoc 库，确保为您的操作系统选择适当的库版本。http://service.sap.com/public/connectorshttp://service.sap.com/public/connectors

注意

您必须有一个 SAP Service Marketplace 帐户才能下载和使用这些库。

2. 将 sapjco3.jar、libsapjco3.so （或 Windows 上的 sapjco3.dll ）和 sapidoc3.jar 库文件复制到 Fuse 安装的 lib/ 目录中。

3. 在文本编辑器中打开配置属性文件 etc/config.properties，以及自定义属性文件 etc/custom.properties。在 etc/config.properties 文件中，查找 org.osgi.framework.system.packages.extra 属性并复制完整属性设置（此设置会扩展多个

行，带有反斜杠字符 \，用于表示行继续）。现在，将此设置粘贴到 etc/custom.properties 文件中。

现在，您可以添加支持 SAP 库所需的额外软件包。在 etc/custom.properties 文件中，将所需的软件包添加到 org.osgi.framework.system.packages.extra 设置中，如下所示：

```
org.osgi.framework.system.packages.extra  \
... , \
com.sap.conn.idoc, \
com.sap.conn.idoc.jco, \
com.sap.conn.jco, \
com.sap.conn.jco.ext, \
com.sap.conn.jco.monitor, \
com.sap.conn.jco.rt, \
com.sap.conn.jco.server
```

不要忘记在新条目前面的每行的末尾包含一个逗号和反斜杠 \，以便正确执行列表。

4. 重启容器以使这些更改生效。

5. 在容器中安装 camel-sap 功能。在 Karaf 控制台中输入以下命令：

```
JBossFuse:karaf@root> features:install camel-sap
```

290.2.2. 在 JBoss EAP 容器中部署

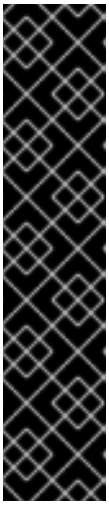
要在 JBoss EAP 容器中部署 SAP 组件，请执行以下步骤：

1. 从 SAP Service Marketplace 下载 SAP JCo 库和 SAP IDoc 库，确保为您的操作系统选择适当的库版本。http://service.sap.com/public/connectorshttp://service.sap.com/public/connectors

注意

您必须有一个 SAP Service Marketplace 帐户才能下载和使用这些库。

2. 将 JCo 库文件和 IDoc 库文件复制到 JBoss EAP 安装的子目录中。



重要

按照命名规则进行操作

原生库必须安装在遵循命名标准的子目录中，格式为 < osname>-<cpuname& gt;。

JBoss 模块手册中 提供了信息和允许名称的完整列表。

例如，如果您的主机平台是 64 位 Linux (linux-x86_64)，请按如下所示安装库文件：

Example

```
cp sapjco3.jar sapidoc3.jar
$JBOSS_HOME/modules/system/layers/fuse/com/sap/conn/jco/main/
mkdir -p $JBOSS_HOME/modules/system/layers/fuse/com/sap/conn/jco/main/lib/linux-x86_64
cp libsapjco3.so
$JBOSS_HOME/modules/system/layers/fuse/com/sap/conn/jco/main/lib/linux-x86_64/
```

3. 创建名为 **$JBOSS_HOME/modules/system/layers/fuse/org/wildfly/camel/extras/main/module.xml** 的新文件，并添加以下内容 ：

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.1" name="org.wildfly.camel.extras">

 <dependencies>
  <module name="org.fusesource.camel.component.sap" export="true" services="export" />
 </dependencies>

</module>
```

290.2.3. 在 Spring Boot 和 OpenShift Container Platform 中部署

要使用 maven-resources 和 maven-jar 插件使用 Maven 部署 SAP，请按照以下步骤操作：

1. 下载库
2. 添加依赖项
3. 将库放在项目中
4. 为库添加配置
5. 部署到 OpenShift

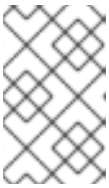
290.2.3.1. 下载库

您需要三个库：

- 所有环境的通用库：
  - sapidoc3.jar
- 架构库：
  - sapjco3.jar
  - sapjco3.so

  如需更多信息，请参阅 SAP Java Connector 文档。

1. 从 SAP Service Marketplace 下载 SAP JCo 库和 SAP IDoc 库，确保为您的操作系统选择适当的库版本。http://service.sap.com/public/connectorshttp://service.sap.com/public/connectors



注意

您必须有一个 SAP Service Marketplace 帐户才能下载和使用这些库。

290.2.3.2. 添加依赖项

1. **Maven 用户需要把以下依赖项添加到其 pom.xml 文件中，才能使用此组件：**

```
<dependency>
 <groupId>org.fusesource</groupId>
 <artifactId>camel-sap-starter</artifactId>
 <exclusions>
  <exclusion>
   <groupId>com.sap.conn.idoc</groupId>
   <artifactId>sapidoc3</artifactId>
  </exclusion>
  <exclusion>
   <groupId>com.sap.conn.jco</groupId>
   <artifactId>sapjco3</artifactId>
  </exclusion>
 </exclusions>
</dependency>
```

290.2.3.3. 放置库

1. **将 SAP 库文件复制到相对于 pom.xml的 lib 目录中**

   运行 Maven 时，它会遵循 pom.xml 中的指令，并将文件复制到指定的位置。

   示例：AMD64

```
src
└── lib
    └── amd64.com.sap.conn
        ├── idoc
        │   └── sapidoc3.jar
        └── jco
            ├── sapjco3.jar
            └── sapjco3.so
```

警告

不要将 SAP 库文件添加到自定义 Maven 存储库中

SAP Java Connector 对 JAR 文件 sapjco3.jar 和 sapidoc3.jar 的名称执行验证。

如果您将 JAR 文件复制到 Maven 存储库，spring-boot-maven-plugin 将通过附加版本号来重命名它们。

这会导致验证失败，从而导致应用程序正确部署。

290.2.3.4. 配置插件

1. *将 maven 配置添加到 pom.xml 中，在 spring-boot-maven-plugin 下：*

   添加 maven-jar-plugin，并将 Class-Path 条目设置为 lib 文件夹位置：

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-jar-plugin</artifactId>
  <configuration>
    <archive>
      <manifestEntries>
        <Class-Path>lib/${os.arch}/sapjco3.jar lib/${os.arch}/sapidoc3.jar</Class-
Path>
      </manifestEntries>
    </archive>
  </configuration>
</plugin>
```

   这会为必要的工件创建正确的结构，并使 SAP 库部署到所需的目标目录中。

2. *使用 pom.xml 中的 maven-resources-plugin 来复制库*

```
<plugin>
```

```
    <artifactId>maven-resources-plugin</artifactId>
    <executions>
      <execution>
        <id>copy-resources01</id>
        <phase>process-classes</phase>
        <goals>
          <goal>copy-resources</goal>
        </goals>
        <configuration>
          <outputDirectory>${basedir}/target/lib</outputDirectory>
          <encoding>UTF-8</encoding>
          <resources>
            <resource>
              <directory>${basedir}/lib</directory>
              <includes>
                <include>**/**</include>
              </includes>
            </resource>
          </resources>
        </configuration>
      </execution>
    </executions>
  </plugin>
```

这会在 运行 'oc 命令时将相对 lib 中的库目录复制到 target/lib。

290.2.3.5. 部署到 Openshift

完成以下步骤以部署到 OpenShift， 并触发 Maven 构建。

1. 运行 oc 以创建并配置构建：

```
oc new-build --binary=true
 --image-stream="<current_Fuse_Java_OpenShift_Imagestream_version>"
 --name=<application_name>
 -e  "ARTIFACT_COPY_ARGS=-a ."
 -e "MAVEN_ARGS_APPEND=<additional_args>
 -e "ARTIFACT_DIR=<relative_path_of_target_directory>"
```

根据需要替换值：

- <current_Fuse_Java_OpenShift_Imagestream_version& gt; : 当前镜像流。

- <APPLICATION_NAME > :您选择的应用程序名称。
- <additional_args > : 附加到 maven 的参数。
- <relative_path_of_target_directory& gt; : path of app to target。

Example

在本例中，使用 MAVEN_ARGS_APPEND 指定仅在 spring-boot 目录中构建特定项目：

```
oc new-build --binary=true --image-stream="fuse7-java-openshift:1.4"
--name=sapik6 -e  "ARTIFACT_COPY_ARGS=-a ." -e "MAVEN_ARGS_APPEND=-pl
spring-boot/sap-srfc-destination-spring-boot" -e "ARTIFACT_DIR=spring-boot/sap-srfc-
destination-spring-boot/target"
```

2. 启动构建（从多模块父目录）

```
oc start-build sapik6 --from-dir=.
```

这会将本地主机源发送到运行 maven 构建的 OpenShift。

3. 启动应用程序

```
oc new-app --image-stream=<name>:<version>
```

Example

```
oc new-app --image-stream=sapik6:latest
```

290.2.4. 使用 JKube 在 Spring Boot 和 OpenShift Container Platform 中部署

要使用 openshift-maven-plugin 插件在项目中部署 SAP，请按照以下步骤操作：

https://www.eclipse.org/jkube/

1. 将连接器放在项目的 lib 目录中：

示例：AMD64

```
src
└── lib
    └── amd64.com.sap.conn
        ├── idoc
        │   └── sapidoc3.jar
        └── jco
            ├── sapjco3.jar
            └── sapjco3.so
```

警告

不要将 SAP 库文件添加到自定义 Maven 存储库中

SAP Java Connector 对 JAR 文件 sapjco3.jar 和 sapidoc3.jar 的名称执行验证。

如果您将 JAR 文件复制到 Maven 存储库，spring-boot-maven-plugin 将通过附加版本号来重命名它们。

这会导致验证失败，从而导致应用程序正确部署。

2. 从 pom.xml 中的初学者中排除嵌入的连接器：

```
<dependency>
  <groupId>org.fusesource</groupId>
```

```
    <artifactId>camel-sap-starter</artifactId>
    <exclusions>
      <exclusion>
        <groupId>com.sap.conn.idoc</groupId>
        <artifactId>sapidoc3</artifactId>
      </exclusion>
      <exclusion>
        <groupId>com.sap.conn.jco</groupId>
        <artifactId>sapjco3</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
```

3. 在 pom.xml 中将本地连接器定义为静态资源：

```
<resources>
  <resource>
    <directory>src/lib/${os.arch}/com/sap/conn/idoc</directory>
    <targetPath>BOOT-INF/lib</targetPath>
    <includes>
      <include>*.jar</include>
    </includes>
  </resource>
  <resource>
    <directory>src/lib/${os.arch}/com/sap/conn/jco</directory>
    <targetPath>BOOT-INF/lib</targetPath>
    <includes>
      <include>*.jar</include>
    </includes>
  </resource>
</resources>
```

4. 配置资源和部署配置，在 pom.xml 中指定连接器路径：

```
<plugin>
  <groupId>org.eclipse.jkube</groupId>
  <artifactId>openshift-maven-plugin</artifactId>
  <version>1.4.0</version>
  <configuration>
    <images>
      <image>
        <name>${project.artifactId}:${project.version}</name>
        <build>
          <from>${java.docker.image}</from>
          <assembly>
            <targetDir>/deployments</targetDir>
            <layers>
              <layer>
                <id>static-files</id>
                <fileSets>
                  <fileSet>
```

```
                    <directory>src/lib/${os.arch}/com/sap/conn/jco</directory>
                    <outputDirectory>static</outputDirectory>
                    <includes>
                      <include>*.so</include>
                    </includes>
                  </fileSet>
                </fileSets>
              </layer>
            </layers>
          </assembly>
        </build>
      </image>
    </images>
  </configuration>
  <executions>
    <execution>
      <goals>
        <goal>resource</goal>
        <goal>build</goal>
        <goal>apply</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

290.2.4.1. 在 Openshift 上部署

在 pom.xml 中配置了 openshift-maven-plugin 后，您可以将 fuse spring-boot 镜像导入到特定命名空间中，作为我们的应用程序的构建器镜像。

1. 在应用程序路径中启动：

```
cd <sap_application_path>
```

2. 创建项目流：

```
oc new-project streams
```

3. 导入镜像流：

```
oc import-image streams/fuse7-java-openshift:1.11 --from=registry.redhat.io/fuse7/fuse-java-openshift-rhel8:1.11-32 --confirm -n streams (JDK8)
```

4. 创建项目

```
oc new-project <your_project>
```

5. 使用 maven 部署应用程序：

```
mvn clean oc:deploy -Djkube.docker.imagePullPolicy=Always -Popenshift -
Djkube.generator.from=streams/fuse7-java-openshift:1.11 -
Djkube.resourceDir=./src/main/jkube -Djkube.openshiftManifest=target/classes/META-
INF/jkube/openshift.yml -Djkube.generator.fromMode=istag
```

290.3. URI 格式

SAP 组件提供两种不同类型的端点： Remote Function Call (RFC)端点和 Intermediate Document (IDoc)端点。

RFC 端点的 URI 格式如下：

```
sap-srfc-destination:destinationName:rfcName
sap-trfc-destination:destinationName:rfcName
sap-qrfc-destination:destinationName:queueName:rfcName
sap-srfc-server:serverName:rfcName[?options]
sap-trfc-server:serverName:rfcName[?options]
```

IDoc 端点的 URI 格式如下：

```
sap-idoc-
destination:destinationName:idocType[:idocTypeExtension[:systemRelease[:applicationRelease]]]
sap-idoclist-
destination:destinationName:idocType[:idocTypeExtension[:systemRelease[:applicationRelease]]]
sap-qidoc-
destination:destinationName:queueName:idocType[:idocTypeExtension[:systemRelease[:applicationR
elease]]]
sap-qidoclist-
destination:destinationName:queueName:idocType[:idocTypeExtension[:systemRelease[:applicationR
elease]]]
sap-idoclist-server:serverName:idocType[:idocTypeExtension[:systemRelease[:applicationRelease]]]
[?options]
```

前缀为 sap-endpointKind-destination 的 URI 格式定义目的地端点（换句话说，Camel producer 端点），而 destinationName 是到 SAP 实例的特定出站连接的名称。出站连接在组件级别命名和配置，如 第 290.6.2 节 “目标配置” 所述。

前缀为 sap-endpointKind-server 的 URI 格式定义服务器端点（换句话说，Camel consumer endpoint）和 serverName 是 SAP 实例的特定入站连接的名称。入站连接在组件级别命名和配置，如

第 290.6.3 节 “服务器配置” 所述。

RFC 端点 URI 的其他组件如下：

rfcName

（必需） 在目标端点 URI 中，由连接的 SAP 实例中端点调用的 RFC 名称。在服务器端点 URI 中，从连接的 SAP 实例调用时由端点处理的 RFC 名称。

queueName

指定此端点向发送 SAP 请求的队列。

IDoc 端点 URI 的其他组件如下：

idocType

（必需） 指定此端点生成的 IDoc 的 Basic IDoc 类型。

idocTypeExtension

指定此端点生成的 IDoc 类型 Extension （若有）。

systemRelease

指定关联的 SAP Basis 发行版本（如果有），此端点生成的 IDoc。

applicationRelease

Specifes 关联的应用程序发行版本（如果有），此端点生成的 IDoc。

queueName

指定此端点向发送 SAP 请求的队列。

290.4. 选项

290.4.1. RFC 目标端点的选项

RFC 目标端点(sap-srfc-destination,sap-trfc-destination, 和 sap-qrfc-destination)支持以下 URI 选项：

Name	default	描述
有状态	**false**	如果为 **true**，指定此端点启动 SAP 有状态会话
Transacted	**false**	如果为 **true**，指定此端点启动 SAP 事务

RFC 服务器端点的选项

SAP RFC 服务器端点(sap-srfc-server 和 sap-trfc-server)支持以下 URI 选项：

Name	default	描述
有状态	**false**	如果为 **true**，请指定此端点启动 SAP 有状态会话。
propagateExceptions	**false**	**（仅SAP-trfc-server 端点）** 如果为 **true**，指定此端点将异常传播到 SAP 中的调用者，而不是交换的异常处理程序

IDoc List Server 端点的选项

SAP IDoc List Server 端点(sap-idoclist-server)支持以下 URI 选项：

Name	default	描述
有状态	**false**	如果为 **true**，请指定此端点启动 SAP 有状态会话。
propagateExceptions	**false**	如果为 **true**，指定此端点将异常传播到 SAP 中的调用者，而不是交换的异常处理程序

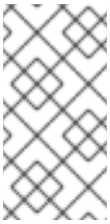
290.5. RFC 和 IDOC 端点概述

SAP 组件软件包提供以下 RFC 和 IDoc 端点：

sap-srfc-destination

JBoss Fuse SAP Synchronous Remote Function Call Destination Camel 组件.如果 Camel

路由需要同步向 SAP 系统的请求和响应，请使用此端点。



注意

此组件使用的 sRFC 协议为 SAP 系统以及 最佳工作 提供请求和响应。如果在发送请求时出现通信错误，接收 SAP 系统的远程功能调用的完成状态将保留在问题 中。

sap-trfc-destination

JBoss Fuse SAP Transactional Remote Function Call Destination Camel 组件.当请求必须最多需要 一次发送到接收的 SAP 系统时，请使用此端点。要达到此目的，组件会生成一个事务 ID tid，它将通过路由交换中组件发送的每个请求相配。接收 SAP 系统在发出请求之前记录与请求相关的 tid ；如果 SAP 系统以同样的方式 接收请求，它不会发送请求。因此，如果在通过此组件的端点发送请求时遇到通信错误，它可以在同一交换中重试发送请求，知道它仅发送并执行一次。

注意

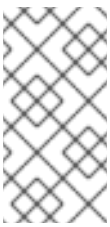
此组件使用的 tRFC 协议是异步的，不会返回响应。因此，此组件的端点不会返回响应消息。

注意

此组件不能保证一系列通过其端点的请求顺序，这些请求的交付和执行顺序可能因通信错误和重新发送请求而在接收 SAP 系统中有所不同。有关保证交付顺序，请参阅 JBoss Fuse SAP Queued Remote Function Call Destination Camel 组件。

sap-qrfc-destination

JBoss Fuse SAP Queued Remote Function Call Destination Camel 组件.此组件通过添加 以便 交付保证通过其端点来扩展 JBoss Fuse Transactional Remote Function Call Destination camel 组件的功能。当一系列请求相互依赖且必须 最多一次 发送到接收 SAP 系统时，请使用 此端点。组件最多可完成 一次 交付保证，使用与 JBoss Fuse SAP Transactional Remote Function Call Destination Camel 组件相同的机制。排序保证是通过按照 SAP 系统收到的顺序将请求序列化到入站队列 来实现的。入站队列由 SAP 中的 QIN 调度程序 处理。激活 入站队列后，QIN 调度程序按顺序执行队列请求。



注意

此组件使用的 qRFC 协议是异步的，不会返回响应。因此，此组件的端点不会返回响应消息。

sap-srfc-server

JBoss Fuse SAP Synchronous Remote Function Call Server Camel 组件.如果需要 Camel 路由来同步处理来自 SAP 系统和响应的请求，请使用此组件及其端点。

sap-trfc-server

JBoss Fuse SAP Transactional Remote Function Call Server Camel 组件.如果发送 SAP 系统最多需要 一次 将其请求发送到 Camel 路由，请使用此端点。要达到此目的，发送 SAP 系统会生成一个事务 ID tid，它会与其发送到组件端点的每个请求相配。发送 SAP 系统首先检查组件是否被指定 tid 接收，然后再发送与 tid 关联的一系列请求。组件会检查所接收的 tid列表，如果其不在该列表中，记录发送的 tid，然后响应发送 SAP 系统，然后响应发送 SAP 系统，表示 tid 是否已记录。如果之前未记录过 tid，发送 SAP 系统将传输一系列请求。这可让发送 SAP 系统一次可靠地向 camel 路由发送一系列请求。

sap-idoc-destination

JBoss Fuse SAP IDoc Destination Camel 组件.如果需要 Camel 路由向 SAP 系统发送中间文档(IDocs)列表时，请使用此端点。

sap-idoclist-destination

JBoss Fuse SAP IDoc List Destination Camel 组件.如果需要 Camel 路由向 SAP 系统发送中间文档(IDocs)列表时，请使用此端点。

sap-qidoc-destination

JBoss Fuse SAP Queued IDoc Destination Camel 组件.如果需要 Camel 路由 才能将中间文档(IDocs)列表发送到 SAP 系统，请使用此组件及其端点。

sap-qidoclist-destination

JBoss Fuse SAP Queued IDoc List Destination Camel 组件.如果需要 camel 路由 才能将中间文档(IDocs)列表发送到 SAP 系统，请使用此组件及其端点。

sap-idoclist-server

JBoss Fuse SAP IDoc List Server Camel 组件.在发送 SAP 系统需要向 Camel 路由发送 Intermediate Document 列表时，请使用此端点。此组件使用 tRFC 协议与 SAP 通信，如 sap-trfc-server-standalone 快速启动中所述。

SAP RFC 目标端点

RFC 目标端点支持到 SAP 的出站通信，这使得这些端点能够调用 SAP 中的 ABAP 功能模块。RFC 目标端点被配置为通过特定连接 SAP 实例发出对特定 ABAP 功能的 RFC 调用。RFC 目的地是出站连接的逻辑设计，具有唯一名称。RFC 目的地由一组名为 目标数据 的连接参数指定。

RFC 目标端点从 IN-OUT 交换的输入消息中提取 RFC 请求，并在对 SAP 的函数调用中接收和分配该请求。交换的输出消息包含来自函数调用的响应。由于 SAP RFC 目标端点只支持出站通信，因此 RFC 目标端点只支持创建制作者。

SAP RFC 服务器端点

RFC 服务器端点支持 SAP 的入站通信，它允许 SAP 中的 ABAP 应用程序向服务器端点发出 RFC 调用。ABAP 应用程序与 RFC 服务器端点交互，就像它是远程功能模块一样。RFC 服务器端点被配置为通过 SAP 实例的特定连接接收对特定 RFC 功能的 RFC 调用。RFC 服务器是入站连接的逻辑设计，具有唯一的名称。RFC 服务器由一组名为 服务器数据 的连接参数指定。

RFC 服务器端点处理传入的 RFC 请求，并将其分配为 IN-OUT 交换的输入消息。交换的输出消息返回为 RFC 调用的响应。由于 SAP RFC 服务器端点只支持入站通信，因此 RFC 服务器端点只支持创建消费者。

SAP IDoc 和 IDoc 列表目标端点

IDoc 目标端点支持到 SAP 的出站通信，然后可以对 IDoc 消息执行进一步处理。IDoc 文档表示业务事务，可与非 SAP 系统轻松交换。IDoc 目的地由一组称为 目标数据 的连接参数指定。

IDoc 列表目标端点与 IDoc 目标端点类似，但它处理的消息由 IDoc 文档列表组成。

SAP IDoc list server endpoint

IDoc list 服务器端点支持 SAP 的入站通信，使 Camel 路由能够接收来自 SAP 系统的 IDoc 文档列表。IDoc list 服务器通过一组称为 服务器数据 的连接参数来指定。

元数据软件仓库

元数据存储库用于存储以下元数据：

功能模块的接口描述

此元数据由 JCo 和 ABAP 运行时用于检查 RFC 调用，以确保通信合作伙伴之间数据的类型安全传输，然后再分配这些调用。仓库填充了存储库数据。仓库数据是命名功能模板的映射。功能模板包含描述所有参数的元数据，以及传递给函数模块的输入信息，并具有它描述的功能模块的唯一名称。

idoc 类型描述

IDoc 运行时使用此元数据来确保在发送到通信合作伙伴之前正确格式化 IDoc 文档。基本 IDoc

类型由名称、允许的片段列表以及片段之间分层关系的描述组成。一些额外的限制可以在网段上实施：一个片段可以是强制的或可选；并可为每个片段指定最小/最大范围（定义那个片段允许的重复次数）。

因此，SAP 目的地和服务器端点需要访问存储库来发送和接收 RFC 调用，并发送和接收 IDoc 文档。对于 RFC 调用，端点调用和处理的所有函数模块的元数据必须位于存储库中；对于 IDoc 端点，由端点处理的所有 IDoc 类型和 IDoc 类型扩展的元数据必须位于存储库中。目的地和服务器端点使用的存储库的位置在目标数据和各自连接的服务器数据中指定。

如果是 SAP 目标端点，它所使用的存储库通常驻留在 SAP 系统中，默认为它所连接的 SAP 系统。此默认要求目标数据中没有显式配置。另外，目标端点发出的远程函数调用的元数据已存在于它调用的任何现有功能模块存储库中。目标端点发出的调用元数据，因此不需要在 SAP 组件中进行配置。

另一方面，由服务器端点处理的功能调用的元数据通常不在 SAP 系统存储库中，必须由驻留于 SAP 组件的存储库提供。SAP 组件维护指定元数据存储库的映射。存储库的名称对应于它提供元数据的服务器的名称。

290.6. 配置

SAP 组件维护三个映射，以存储目标数据、服务器数据和存储库数据。目标数据存储和 服务器数据存储 使用特殊的配置对象 SapConnectionConfiguration，它会自动注入到 SAP 组件（在 Blueprint XML 配置或 Spring XML 配置文件上下文中）。存储库数据存储 必须在相关的 SAP 组件上直接配置。

290.6.1. 配置概述

概述

SAP 组件维护三个映射，以存储目标数据、服务器数据和存储库数据。组件的属性 destinationDataStore 存储目标数据，按目标名称密钥。属性 serverDataStore 存储按服务器名称密钥的服务器数据。属性 repositoryDataStore 存储按存储库名称密钥的存储库数据。您必须在初始化过程中将这些配置传递给组件。

Example

以下示例演示了如何在 Blueprint XML 文件中配置示例目标数据存储和示例服务器数据存储。sap-configuration bean （类型为 SapConnectionConfiguration）会自动注入到此 XML 文件中使用的任何 SAP 组件中。

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint ... >
    ...
```

```
    <!-- Configures the Inbound and Outbound SAP Connections -->
    <bean id="sap-configuration"
        class="org.fusesource.camel.component.sap.SapConnectionConfiguration">
        <property name="destinationDataStore">
            <map>
                <entry key="quickstartDest" value-ref="quickstartDestinationData" />
            </map>
        </property>
        <property name="serverDataStore">
            <map>
                <entry key="quickstartServer" value-ref="quickstartServerData" />
            </map>
        </property>
    </bean>

    <!-- Configures an Outbound SAP Connection -->
    <!-- *** Please enter the connection property values for your environment *** -->
    <bean id="quickstartDestinationData"
        class="org.fusesource.camel.component.sap.model.rfc.impl.DestinationDataImpl">
        <property name="ashost" value="example.com" />
        <property name="sysnr" value="00" />
        <property name="client" value="000" />
        <property name="user" value="username" />
        <property name="passwd" value="passowrd" />
        <property name="lang" value="en" />
    </bean>

    <!-- Configures an Inbound SAP Connection -->
    <!-- *** Please enter the connection property values for your environment ** -->
    <bean id="quickstartServerData"
        class="org.fusesource.camel.component.sap.model.rfc.impl.ServerDataImpl">
        <property name="gwhost" value="example.com" />
        <property name="gwserv" value="3300" />
        <!-- Do not change the following property values -->
        <property name="progid" value="QUICKSTART" />
        <property name="repositoryDestination" value="quickstartDest" />
        <property name="connectionCount" value="2" />
    </bean>
</blueprint>
```

290.6.2. 目标配置

概述

目的地的配置在 SAP 组件的 **destinationDataStore** 属性中维护。此映射中的每个条目都配置与 SAP 实例不同的出站连接。每个条目的键是出站连接的名称，并在目标端点 URI 的 **destinationName** 组件中使用，如 URI 格式部分所述。

每个条目的值都是目标数据配置对象 - **org.fusesource.camel.component.sap.model.rfc.impl.DestinationDataImpl** -，用于指定出站 SAP 连接的配置。

目标配置示例

以下蓝图 XML 代码演示了如何使用名称 QuickstartDest 配置示例目的地。

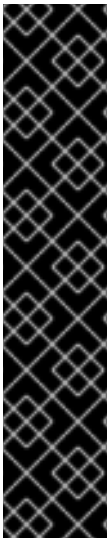
```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint ... >
   ...
   <!-- Create interceptor to support tRFC processing -->
   <bean id="currentProcessorDefinitionInterceptor"
        class="org.fusesource.camel.component.sap.CurrentProcessorDefinitionInterceptStrategy" />

   <!-- Configures the Inbound and Outbound SAP Connections -->
   <bean id="sap-configuration"
      class="org.fusesource.camel.component.sap.SapConnectionConfiguration">
      <property name="destinationDataStore">
         <map>
            <entry key="quickstartDest" value-ref="quickstartDestinationData" />
         </map>
      </property>
   </bean>

   <!-- Configures an Outbound SAP Connection -->
   <!-- *** Please enter the connection property values for your environment *** -->
   <bean id="quickstartDestinationData"
      class="org.fusesource.camel.component.sap.model.rfc.impl.DestinationDataImpl">
      <property name="ashost" value="example.com" />
      <property name="sysnr" value="00" />
      <property name="client" value="000" />
      <property name="user" value="username" />
      <property name="passwd" value="password" />
      <property name="lang" value="en" />
   </bean>

</blueprint>
```

例如，在配置目的地（如前面的 Blueprint XML 文件所示），您可以使用以下 URI 在 QuickstartDest 目标上调用 BAPI_FLCUST_GETLIST 远程函数调用：

```
sap-srfc-destination:quickstartDest:BAPI_FLCUST_GETLIST
```

TRFC 和 qRFC 目的地的拦截器

前面的目标配置示例显示了 CurrentProcessorDefinitionInterceptStrategy 对象的实例化。此对象在 Camel 运行时中安装一个拦截器，使 Camel SAP 组件能够在处理 RFC 事务时跟踪其在 Camel 路由内的位置。如需了解更多详细信息，请参阅 “事务 RFC 目标端点”一节。



重要

此拦截器必须安装在 Camel 运行时，才能正确管理出站事务 RFC 通信。

对于事务性 RFC 目标端点（如 sap-trfc-destination 和 sap-qrfc-destination）非常重要。

如果在运行时找不到策略，则 Destination RFC Transaction Handlers 会发出警告，在这种情况下，必须重新置备并重启 Camel 运行时才能正确管理出站事务 RFC 通信。

登录和验证选项

下表列出了用于在 SAP 目标数据存储中配置目的地的 登录和验证 选项：

Name	默认值	描述
client		SAP 客户端，强制登录参数
user		log-in user, log-in 参数用于基于密码的身份验证
aliasUser		可以使用登录用户别名而不是登录用户
userId		用于登录 ABAP AS 的用户身份。如果目标配置使用 SSO/assertion ticket、证书、当前用户或 SNC 环境进行身份验证，则由 JCo 运行时使用。如果没有用户或用户别名，则需要用户 ID。SAP 后端不使用此 ID，JCo 运行时在本地使用它。
passwd		用于基于密码的身份验证的登录密码、登录参数
lang		使用日志语言而不是用户语言
mysapsso2		使用指定的 SAP Cookie Version 2 作为基于 SSO 的身份验证的登录票据
x509cert		使用指定的 X509 证书进行基于证书的身份验证

lcheck		在首次调用 1（启用）之前，请提醒身份验证。仅在特例中使用 **lcheck**。
useSapGui		使用可见、隐藏或不使用 SAP GUI
codePage		其他 log-in 参数，用于定义用于转换日志参数的代码页面。仅在特殊情况下使用 **codePage**。
getsso2		在登录后订购 SSO 票据，目标属性中提供了获取的票据
denyInitialPassword		如果设置为 **1**，则使用初始密码会导致异常（默认为 **0**）。

连接选项

下表列出了用于在 SAP 目标数据存储中配置目的地的连接选项：

Name	**默认值**	**描述**
saprouter		SAP Router 字符串，用于连接到 SAP 路由器后面的系统。SAP Router string 包含 SAP Routers 及其端口号链，格式为：**(/H/<host>[/S/<port>])+**
sysnr		SAP ABAP 应用服务器的系统数，直接连接强制使用
ashost		SAP ABAP 应用服务器，直接连接强制
mshost		SAP 消息服务器，负载平衡连接的强制属性
msserv		SAP 消息服务器端口，可选属性，用于负载平衡连接。为了解析服务名称 sapmsXXX，操作系统的网络层在 **etc/services** 中执行查找。如果使用端口号而不是符号服务名称，则不需要查找，且不需要额外的条目。

gwhost		允许指定 Concrete 网关。使用它来建立与应用服务器的连接。如果未使用应用服务器上的网关指定网关。
gwserv		在使用 **gwhost** 时设置此项。允许指定该网关上使用的端口。如果没有指定应用服务器上的网关端口。为了解析服务名称 sapgwXXX，操作系统的网络层在 **etc/services** 中执行查找。如果使用端口号而不是符号服务名称，则不需要查找，且不需要额外的条目。
r3name		SAP 系统的系统 ID，负载平衡连接的强制属性。
group		SAP 应用服务器组，负载平衡连接的必要属性
network	**LAN**	根据 JCo 和您的目标系统之间的网络质量，以优化性能来设置这个值。有效值为 **LAN** 或 **WAN**（仅适用于快速序列化）。**WAN** 使用较慢但效率更高的压缩算法，数据分析用于进一步压缩选项。**LAN** 使用非常快的压缩算法，它只进行基本数据分析。使用 **LAN** 选项时，压缩率并不高，但网络传输时间被视为不太显著。默认设置为 **LAN**。
serializationFormat	**rowBased**	序列化格式。可以是 **基于行**（默认）或基于 **列**（快速序列化）。

连接池选项

下表列出了用于在 SAP 目标数据存储中配置目的地的 连接池 选项：

Name	默认值	描述
peakLimit	**0**	目的地同时活跃的出站连接的最大数量。值 **0** 允许无限数量的活跃连接。否则，它会自动增加到 **poolCapacity**。如果配置，默认设置为 **poolCapacity** 的值。如果没有指定 **poolCapacity**，则默认为 **0**（无限）。

poolCapacity	**1**	目标打开的最大空闲出站连接数。值 **0** 表示没有连接池（默认为 **1**）。
expirationTime		目的地在内部持有的空闲连接的最小时间（毫秒）必须保持打开状态。
expirationPeriod		目标检查已发布的连接的过期时间（毫秒）。
maxGetTime		如果应用程序已分配最多允许的连接数，等待连接的最长时间，以毫秒为单位。

保护网络连接选项

下表列出了用于在 SAP 目标数据存储中配置目的地 的安全网络选项 ：

Name	**默认值**	**描述**
sncMode		安全网络连接(SNC)模式、**0** （关闭）或 **1** (on)
sncPartnername		SNC 合作伙伴，例如：**p:CN=R3, O=XYZ-INC, C=EN**
sncQop		SNC 安全级别： **1** 到 **9**
sncMyname		自己的 SNC 名称。覆盖环境设置
sncLibrary		提供 SNC 服务的库的路径

仓库选项

下表列出了用于在 SAP 目标数据存储中配置目的地 的存储库 选项：

Name	**默认值**	**描述**
repositoryDest		指定要用作存储库的目标位置。
repositoryUser		如果尚未定义存储库目的地，这定义了用于存储库调用的用户。这可让您将其他用户用于存储库查找。

repositoryPasswd		仓库用户的密码。使用仓库用户时必需。
repositorySnc		**（可选）** 如果将 SNC 用于此目的地，则可以将其关闭用于存储库连接，如果此属性设置为 0。**默认设置为 jco.client.snc_mode。只适用于特殊情况。**
repositoryRoundtripOptimization		启用 **RFC_METADATA_GET** API，它会在一次往返中提供存储库数据。 **1** 激活在 ABAP 系统中使用 **RFC_METADATA_GET**， **0** 在 ABAP 系统中停用 **RFC_METADATA_GET**。 如果没有设置属性，目的地最初会进行远程调用，以检查 **RFC_METADATA_GET** 是否可用。如果可用，则目标将使用它。 **注意** 如果存储库已初始化（例如，由于被某些其他目标使用），则此属性没有任何效果。通常，此属性与 ABAP 系统相关，在指向同一 ABAP 系统的所有目的地上都应具有相同的值。有关后端先决条件，请参阅备注 1456826。

跟踪配置选项

下表列出了用于在 SAP 目标数据存储中配置目的地的 trace 配置选项 ：

Name	默认值	描述

trace		启用/禁用 RFC trace (**0** 或 **1**)
cpicTrace		启用/禁用 CPIC trace **[0..3]**

290.6.3. 服务器配置

概述

服务器的配置在 SAP 组件的 serverDataStore 属性中维护。此映射中的每个条目都配置与 SAP 实例不同的入站连接。每个条目的键是出站连接的名称，并在服务器端点 URI 的 serverName 组件中使用，如 URI 格式部分所述。

每个条目的值都是 服务器数据配置对象 org.fusesource.camel.component.sap.model.rfc.impl.ServerDataImpl，用于定义入站 SAP 连接的配置。

服务器配置示例

以下蓝图 XML 代码演示了如何使用名称 QuickstartServer 创建示例服务器配置。

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint ... >
  ...
  <!-- Configures the Inbound and Outbound SAP Connections -->
  <bean id="sap-configuration"
    class="org.fusesource.camel.component.sap.SapConnectionConfiguration">
    <property name="destinationDataStore">
      <map>
        <entry key="quickstartDest" value-ref="quickstartDestinationData" />
      </map>
    </property>
    <property name="serverDataStore">
      <map>
        <entry key="quickstartServer" value-ref="quickstartServerData" />
      </map>
    </property>
  </bean>

  <!-- Configures an Outbound SAP Connection -->
  <!-- *** Please enter the connection property values for your environment *** -->
  <bean id="quickstartDestinationData"
    class="org.fusesource.camel.component.sap.model.rfc.impl.DestinationDataImpl">
    <property name="ashost" value="example.com" />
    <property name="sysnr" value="00" />
    <property name="client" value="000" />
    <property name="user" value="username" />
    <property name="passwd" value="passowrd" />
```

```
        <property name="lang" value="en" />
    </bean>

    <!-- Configures an Inbound SAP Connection -->
    <!-- *** Please enter the connection property values for your environment ** -->
    <bean id="quickstartServerData"
        class="org.fusesource.camel.component.sap.model.rfc.impl.ServerDataImpl">
        <property name="gwhost" value="example.com" />
        <property name="gwserv" value="3300" />
        <!-- Do not change the following property values -->
        <property name="progid" value="QUICKSTART" />
        <property name="repositoryDestination" value="quickstartDest" />
        <property name="connectionCount" value="2" />
    </bean>
</blueprint>
```

注意本示例如何配置目标连接 QuickstartDest，服务器使用它来从远程 SAP 实例检索元数据。此目的地通过 repositoryDestination 选项在服务器数据中配置。如果您没有配置这个选项，则需要创建本地元数据存储库（请参阅 第 290.6.4 节 “仓库配置”）。

例如，在配置目的地（如前面的 Blueprint XML 文件所示），您可以使用以下 URI 处理来自调用客户端的 BAPI_FLCUST_GETLIST 远程功能调用：

```
sap-srfc-server:quickstartServer:BAPI_FLCUST_GETLIST
```

所需选项

服务器数据配置对象所需的选项有，如下所示：

Name	默认值	描述
gwhost		要注册服务器连接的网关主机。
gwserv		网关服务，这是可以进行注册的端口。要解析服务名称 **sapgwXXX**，操作系统的网络层在 **etc/services** 中执行查找。如果使用端口号而不是符号服务名称，则不需要查找，且不需要额外的条目。
progid		完成注册的程序 ID。作为网关和 ABAP 系统目的地中的标识符。

repositoryDestination		指定服务器可用于从远程 SAP 服务器托管的元数据存储库检索元数据的目的地名称。
connectionCount		使用网关注册的连接数量。

保护网络连接选项

服务器数据配置对象的安全连接选项是，如下所示：

Name	默认值	描述
sncMode		安全网络连接(SNC)模式、**0** （关闭）或 **1** (on)
sncQop		SNC 安全级别，**1** 到 **9**
sncMyname		服务器的 SNC 名称。覆盖默认 SNC 名称。通常，比如 **p:CN=JCoServer、O=ACompany、C=EN**.
sncLib		提供 SNC 服务的库的路径。如果没有提供此属性，则使用 **jco.middleware.snc_lib** 属性的值

其他选项

服务器数据配置对象的其他选项有如下：

Name	默认值	描述
saprouter		SAP 路由器字符串用于受防火墙保护的系统，因此只能在在该 ABAP 系统的网关中注册服务器时通过 SAProuter 访问。典型的路由器字符串为 **/H/firewall.hostname/H/**
maxStartupDelay		失败时两个启动尝试之间的最大时间（以秒为单位）。最初，在每个启动失败后，等待的时间从 1 秒加倍，直到达到最大值为止，或者服务器可以成功启动。

trace		启用/禁用 RFC trace (**0** 或 **1**)
workerThreadCount		服务器连接使用的最大线程数。如果没有设置，则 **connectionCount** 的值将用作 **workerThreadCount**。线程的最大数量不能超过 99。
workerThreadMinCount		服务器连接使用的最小线程数量。如果没有设置，则 **connectionCount** 的值将用作 **workerThreadMinCount**。

290.6.4. 仓库配置

概述

存储库的配置在 SAP 组件的 repositoryDataStore 属性中维护。此映射中的每个条目都配置不同的存储库。每个条目的键是存储库的名称，此键也对应于附加到此存储库的服务器的名称。

每个条目的值都是存储库数据配置对象 org.fusesource.camel.component.sap.model.rfc.impl.RepositoryDataImpl，用于定义元数据存储库的内容。存储库数据对象是功能模板配置对象 org.fuesource.camel.component.sap.model.rfc.impl.FunctionTemplateImpl 的映射。此映射中的每个条目都指定函数模块的接口，每个条目的键是指定的功能模块的名称。

仓库数据示例

以下代码显示了配置元数据存储库的简单示例：

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint ... >
  ...
  <!-- Configures the sap-srfc-server component -->
  <bean id="sap-configuration"
    class="org.fusesource.camel.component.sap.SapConnectionConfiguration">
    <property name="repositoryDataStore">
      <map>
        <entry key="nplServer" value-ref="nplRepositoryData" />
      </map>
    </property>
  </bean>

  <!-- Configures a metadata Repository -->
  <bean id="nplRepositoryData"
    class="org.fusesource.camel.component.sap.model.rfc.impl.RepositoryDataImpl">
```

```
        <property name="functionTemplates">
            <map>
                <entry key="BOOK_FLIGHT" value-ref="bookFlightFunctionTemplate" />
            </map>
        </property>
    </bean>
    ...
</blueprint>
```

功能模板属性

function 模块的接口由四个参数列表组成，其数据会重新传输到 RFC 调用中的 function 模块。每个参数列表由一个或多个字段组成，每个字段都是在 RFC 调用中传输的命名参数。支持以下参数列表和异常列表：

- *import 参数列表包含发送到 RFC 调用中 function 模块的参数值 ；*
- *export 参数列表包含 RFC 调用中的 function 模块返回的参数值 ；*
- *更改参数列表包含在 RFC 调用中发送到和返回的参数值 ；*
- *table 参数列表 包含发送到 RFC 调用中的函数模块和返回的内部表值。*
- *function 模块的接口也由 ABAP 例外列表组成，在 RFC 调用中调用该模块时可能会引发。*

功能模板描述了函数接口的每个参数列表中的名称和参数类型，以及函数引发的 ABAP 异常。功能模板对象维护五个元数据对象的属性列表，如下表所述。

属性	描述
importParameterList	列表字段元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.ListFieldMeataDataImpl**。指定将 RFC 调用中发送到函数模块的参数。
changingParameterList	列表字段元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.ListFieldMeataDataImpl**。指定在 RFC 调用中向函数模块发送和返回的参数。

exportParameterList	列表字段元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.ListFieldMeataDataImpl**。指定来自 function 模块的 RFC 调用中返回的参数。
tableParameterList	列表字段元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.ListFieldMeataDataImpl**。指定在 RFC 调用中向函数模块发送和返回的表参数。
exceptionList	ABAP 异常元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.AbapExceptionImpl** 的列表。指定在功能模块的 RFC 调用中可能会引发的 ABAP 异常。

功能模板示例

以下示例演示了如何配置功能模板：

```
<bean id="bookFlightFunctionTemplate"
    class="org.fusesource.camel.component.sap.model.rfc.impl.FunctionTemplateImpl">
    <property name="importParameterList">
        <list>
            ...
        </list>
    </property>
    <property name="changingParameterList">
        <list>
            ...
        </list>
    </property>
    <property name="exportParameterList">
        <list>
            ...
        </list>
    </property>
    <property name="tableParameterList">
        <list>
            ...
        </list>
    </property>
    <property name="exceptionList">
        <list>
            ...
        </list>
    </property>
</bean>
```

列出字段元数据属性

列表字段元数据对象 org.fusesource.camel.component.sap.model.rfc.impl.ListFieldMeataDataImpl 指定参数列表中字段的名称和类型。对于 elementary 参数字段(CHAR,DATE,BCD,TIME,BYTE,NUM,FLOAT,INT 1 ,INT1,INT2,DECF16,DECF34,STRING,XSTRING)），下表列出了可能在列表元数据上设置的配置属性：

Name	默认值	描述
name	-	参数字段的名称。
type	-	字段的参数类型。
byteLength	-	非统一代码布局的字段长度（以字节为单位）。这个值取决于参数类型。请参阅 第 290.9 节 “RFC 的消息正文”。
unicodeByteLength	-	Unicode 布局的字段长度（以字节为单位）。这个值取决于参数类型。请参阅 第 290.9 节 “RFC 的消息正文”。
十进制	**0**	字段值中的十进制数；只有参数类型 BCD 和 FLOAT 时需要。请参阅 第 290.9 节 “RFC 的消息正文”。
optional	**false**	如果为 **true**，该字段是可选的，且不需要在 RFC 调用中设置

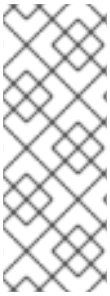
注意

所有 elementary 参数字段要求在字段 metadata 对象中指定 name、type、bytesLength 和 unicodeByteLength 属性。此外，BCD、FLOAT、DECF16 和 DECF34 字段需要在字段 metadata 对象中指定十进制属性。

对于类型为 TABLE 或 STRUCTURE 的复杂参数字段，下表列出了列表字段 metadata 对象可能设置的配置属性：

Name	默认值	描述
name	-	参数字段的名称

type	-	字段的参数类型
recordMetaData	-	结构或表的元数据。将传递记录元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.RecordMetaDataImpl** 以指定结构或表行中的字段。
optional	**false**	如果为 **true**，该字段是可选的，且不需要在 RFC 调用中设置



注意

所有复杂的参数字段要求在字段元数据对象中 指定名称、类型 和记录MetaData 属性。recordMetaData 属性的值是一个记录字段元数据对象 org.fusesource.camel.component.sap.model.rfc.impl.RecordMetaDataImpl，它指定嵌套结构的结构或表结构的结构。

Elementary 列表字段元数据示例

以下元数据配置指定一个一个可选的 24 位打包的 BCD number 参数，它有两个十进制位置，名为 TICKET_PRICE ：

```
<bean class="org.fusesource.camel.component.sap.model.rfc.impl.ListFieldMetaDataImpl">
    <property name="name" value="TICKET_PRICE" />
    <property name="type" value="BCD" />
    <property name="byteLength" value="12" />
    <property name="unicodeByteLength" value="24" />
    <property name="decimals" value="2" />
    <property name="optional" value="true" />
</bean>
```

复杂的列表字段元数据示例

以下元数据配置指定所需的 TABLE 参数，名为 CONNINFO，其行结构由 connectionInfo 记录元数据对象指定：

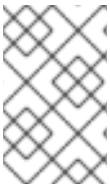
```
<bean    class="org.fusesource.camel.component.sap.model.rfc.impl.ListFieldMetaDataImpl">
    <property name="name" value="CONNINFO" />
    <property name="type" value="TABLE" />
    <property name="recordMetaData" ref="connectionInfo" />
</bean>
```

记录元数据属性

记录元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.RecordMetaDataImpl** 指定嵌套 **STRUCTURE** 或 **TABLE** 参数的行。记录元数据对象维护一个记录字段元数据对象列表 **org.fusesource.camel.component.sap.model.rfc.impl.FieldMetaDataImpl**，用于指定嵌套结构或表行中的参数。

下表列出了可以在记录元数据对象上设置的配置属性：

Name	默认值	描述
name	-	记录的名称。
recordFieldMetaData	-	记录字段元数据对象列表 **org.fusesource.camel.component.sap.model.rfc.impl.FieldMetaDataImpl**。指定结构中包含的字段。



注意

记录元数据对象的所有属性都是必需的。

记录元数据示例

以下示例演示了如何配置记录元数据对象：

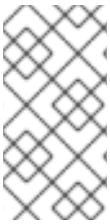
```
<bean id="connectionInfo"
      class="org.fusesource.camel.component.sap.model.rfc.impl.RecordMetaDataImpl">
  <property name="name" value="CONNECTION_INFO" />
  <property name="recordFieldMetaData">
    <list>
      ...
    </list>
  </property>
</bean>
```

记录字段元数据属性

记录字段元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.FieldMetaDataImpl** 指定具有结构的参数字段的名称和类型。

记录字段 **metadata** 对象与参数字段 **metadata** 对象类似，但必须额外指定嵌套结构或表行中单个字

段位置的偏移。单个字段的非统一代码和 Unicode 偏移量必须从结构或行中上述字段的总和和指定。



注意

如果不正确指定嵌套结构和表行中的字段偏移，会导致底层 JCo 和 ABAP 运行时中的参数字段存储重叠，并防止 RFC 调用中的正确传输值。

对于 elementary 参数字段(CHAR,DATE,BCD,TIME,BYTE,NUM,FLOAT,INT 1 ,INT1,INT2,DECF16,DECF34,STRING,XSTRING)），下表列出了可能在记录元数据上设置的配置属性：

Name	默认值	描述
name	-	参数字段的名称
type	-	字段的参数类型
byteLength	-	非统一代码布局的字段长度（以字节为单位）。这个值取决于参数类型。请参阅 第 290.9 节 “RFC 的消息正文”。
unicodeByteLength	-	Unicode 布局的字段长度（以字节为单位）。这个值取决于参数类型。请参阅 第 290.9 节 “RFC 的消息正文”。
byteOffset	-	非Unicode 布局的字段偏移（以字节为单位）。这个偏移是在封闭结构中字段的字节位置。
unicodeByteOffset	-	Unicode 布局的字段偏移（以字节为单位）。这个偏移是在封闭结构中字段的字节位置。
十进制	**0**	字段值中的十进制数；只有参数类型 **BCD** 和 **FLOAT** 才需要。请参阅 第 290.9 节 “RFC 的消息正文”。

对于类型为 TABLE 或 STRUCTURE 的复杂参数字段，下表列出了可以在记录字段 metadata 对象上设置的配置属性：

Name	默认值	描述

name	-	参数字段的名称
type	-	字段的参数类型
byteOffset	-	非Unicode 布局的字段偏移（以字节为单位）。这个偏移是在封闭结构中字段的字节位置。
unicodeByteOffset	-	Unicode 布局的字段偏移（以字节为单位）。这个偏移是在封闭结构中字段的字节位置。
recordMetaData	-	结构或表的元数据。将传递记录元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.RecordMetaDataImpl** 以指定结构或表行中的字段。

Elementary 记录字段元数据示例

以下元数据配置指定名为 ARR DATE 的 85 字节的 DATE 字段参数，到非Unicode 布局，将位置 170 字节到 Unicode 布局的内线结构中：

```
<bean    class="org.fusesource.camel.component.sap.model.rfc.impl.FieldMetaDataImpl">
   <property name="name" value="ARRDATE" />
   <property name="type" value="DATE" />
   <property name="byteLength" value="8" />
   <property name="unicodeByteLength" value="16" />
   <property name="byteOffset" value="85" />
   <property name="unicodeByteOffset" value="170" />
</bean>
```

复杂的记录字段元数据示例

以下元数据配置指定了名为 FLTINFO 的 STRUCTURE 字段参数，其结构由 flightInfo 记录元数据对象指定。在非统一代码和 Unicode 布局的情况下，参数位于封闭结构的开头。

```
<bean    class="org.fusesource.camel.component.sap.model.rfc.impl.FieldMetaDataImpl">
   <property name="name" value="FLTINFO" />
   <property name="type" value="STRUCTURE" />
   <property name="byteOffset" value="0" />
   <property name="unicodeByteOffset" value="0" />
   <property name="recordMetaData" ref="flightInfo" />
</bean>
```

290.7. 消息标头

SAP 组件支持以下消息标头：

标头	描述
CamelSap.scheme	处理消息的最后一个端点的 URI 方案。使用以下值之一： **sap-srfc-destination** **sap-trfc-destination** **sap-qrfc-destination** **sap-srfc-server** **sap-trfc-server** **sap-idoc-destination** **sap-idoclist-destination** **sap-qidoc-destination** **sap-qidoclist-destination** **sap-idoclist-server**
CamelSap.destinationName	处理消息的最后一个目标端点的目标名称。
CamelSap.serverName	用于处理消息的最后一个服务器端点的服务器名称。
CamelSap.queueName	最后一次排队端点的队列名称，以处理消息。
CamelSap.rfcName	处理消息的最后一个 RFC 端点的 RFC 名称。
CamelSap.idocType	处理消息的最后一个 IDoc 端点的 IDoc 类型。
CamelSap.idocTypeExtension	用于处理消息的最后一个 IDoc 端点的 IDoc 类型扩展（若有）。
CamelSap.systemRelease	系统发行版本（若有）用于处理消息。
CamelSap.applicationRelease	应用程序发行版本（若有）用于处理消息。

290.8. 交换属性

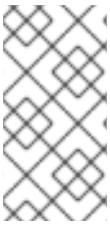
SAP 组件添加以下交换属性：

属性	描述
CamelSap.destinationPropertiesMap	包含交换遇到的每个 SAP 目的地属性的映射。映射按目标名称键，每个条目都是包含该目的地配置属性的 **java.util.Properties** 对象。
CamelSap.serverPropertiesMap	包含交换遇到的每个 SAP 服务器属性的映射。映射按服务器名称键，每个条目都是包含该服务器配置属性的 **java.util.Properties** 对象。

290.9. RFC 的消息正文

请求和响应对象

SAP 端点预期收到含有 SAP 请求对象的消息正文的消息，并返回含有 SAP 响应对象的消息正文。SAP 请求和响应是固定的映射数据结构，其中包含命名字段，每个字段都有预定义的数据类型。



注意

SAP 请求和响应中的命名字段特定于 SAP 端点，每个端点在 SAP 请求中定义参数以及可接受的响应。SAP 端点提供创建特定于它的请求和响应对象的工厂方法。

```
public class SAPEndpoint {
    ...
    public Structure getRequest() throws Exception;

    public Structure getResponse() throws Exception;
    ...
}
```

结构对象

SAP 请求和响应对象以 Java 表示为支持 org.fusesource.camel.component.sap.model.rfc.Structure 接口的结构对象。此接口同时扩展了 java.util.Map 和 org.eclipse.emf.ecore.EObject 接口。

```
public interface Structure extends org.eclipse.emf.ecore.EObject,
                                   java.util.Map<String, Object> {
```

```
    <T> T get(Object key, Class<T> type);

}
```

结构对象中的字段值可以通过映射接口上字段的 getter 方法访问。此外，结构接口还提供了类型限制方法来检索字段值。

使用 Eclipse Modeling Framework (EMF)在组件运行时实施结构对象，并支持该框架的 EObject 接口。结构对象的实例已附加元数据，它们定义和限制它提供的字段映射的结构和内容。可以使用 EMF 提供的标准方法访问和内省此元数据。详情请参考 EMF 文档。

注意

尝试获取结构对象上未定义的参数返回 null。尝试设置结构上未定义的参数会抛出异常，并尝试设置具有不正确类型的参数值。

如以下部分中所述，结构对象可以包含包含复杂字段类型值、STRUCTURE 和 TABLE 的字段。

注意

创建这些类型的实例并将其添加到结构中，不需要创建它们。在封闭结构中访问时，需要根据需要创建这些字段值的实例。

字段类型

驻留在 SAP 请求或响应结构对象内的字段可能是 元素 或复杂。elementary 字段包含一个 scalar 值，而复杂的字段包含元素或复杂类型的一个或多个字段。

Elementary 字段类型

elementary 字段可以是字符、数字、十六进制或字符串字段类型。下表总结了可能位于结构对象中的元素字段类型：

字段类型	对应的 Java 类型	字节长度	Unicode Byte Length	number Decimals Digits	描述

CHAR	**java.lang.String**	1 到 65535	1 到 65535	-	ABAP 类型 'C': Fixed sized character string
DATE	**java.util.Date**	8	16	-	ABAP 类型 'D': Date (format: YYYYMMDD)
BCD	**java.math.BigDecimal**	1 到 16	1 到 16	0 到 14	ABAP 类型"P"：打包的 BCD 号.BCD 号包含每个字节的两位数字。
时间	**java.util.Date**	6	12	-	ABAP 类型 'T': Time (format: HHMMSS)
BYTE	**byte[]**	1 到 65535	1 到 65535	-	ABAP 类型 'X':Fixed sized byte array
NUM	**java.lang.String**	1 到 65535	1 到 65535	-	ABAP 类型 'N': Fixed sized numeric character string
浮点值	**java.lang.Double**	8	8	0 到 15	ABAP 类型 'F': 浮动点号
INT	**java.lang.Integer**	4	4	-	ABAP 类型"I"：4 字节整数
INT2	**java.lang.Integer**	2	2	-	ABAP 类型 'S': 2 字节整数
INT1	**java.lang.Integer**	1	1	-	ABAP 类型 'B': 1 字节整数
DECF16	**java.match.BigDecimal**	8	8	16	ABAP 类型 'decfloat16': 8-byte Decimal Floating Point Number

DECF34	**java.math.BigDecimal**	16	16	34	ABAP 类型 'decfloat34': 16 字节 Decimal Floating Point Number
字符串	**java.lang.String**	8	8	-	ABAP 类型 'G': 变量长度字符字符串
XSTRING	**byte[]**	8	8	-	ABAP 类型 'Y': Variable length byte array

字符字段类型

字符字段包含一个固定大小的字符字符串，可在底层 JCo 和 ABAP 运行时中使用非Unicode 或 Unicode 字符编码。非统一代码字符字符串对每字节一个字符进行编码。Unicode 字符字符串使用 UTF-16 编码以两字节编码。character 字段值在 Java 中表示为 java.lang.String 对象，底层 JCo 运行时负责转换到其 ABAP 表示。

character 字段在其关联的 byteLength 和 unicodeByteLength 属性中声明其字段长度，它决定了每个编码系统中字段字符字符串的长度。

CHAR

CHAR 字符字段是一个包含字母数字字符的文本字段，对应于 ABAP 类型 C。

NUM

NUM 字符字段是一个数字文本字段，仅包含数字字符，对应于 ABAP 类型 N。

DATE

DATE 字符字段是年份、月份和日的 8 个字符日期字段，格式为 YYYYMMDD，对应于 ABAP 类型 D。

时间

TIME 字符字段是 6 个字符时间字段，小时、分钟和秒格式为 HHMMSS，对应于 ABAP 类型 T。

数字字段类型

数字字段包含一个数字。支持以下数字字段类型：

INT

INT 数字字段是一个整数值，作为底层 JCo 和 ABAP 运行时中的 4 字节整数值，对应于 ABAP 类型 I。INT 字段值以 java.lang.Integer 对象表示。

INT2

INT2 numeric 字段是一个整数字段，存储为底层 JCo 和 ABAP 运行时中的 2 字节整数值，对应于 ABAP 类型 S。INT2 字段值在 Java 中以 java.lang.Integer 对象表示。

INT1

INT1 字段是一个整数值，作为底层 JCo 和 ABAP 运行时值中的 1 字节整数值，对应于 ABAP 类型 B。INT1 字段值在 Java 中以 java.lang.Integer 对象表示。

浮点值

FLOAT 字段是一个二进制浮点数字段，作为底层 JCo 和 ABAP 运行时中的 8 字节双值存储，对应于 ABAP 类型 F。FLOAT 字段声明字段在关联十进制属性中包含的十进制数字数。如果是 FLOAT 字段，这个十进制属性的值可以介于 1 到 15 位之间。FLOAT 字段值以 Java 表示，作为 java.lang.Double 对象。

BCD

BCD 字段是一个二进制编码的十进制字段，存储为底层 JCo 和 ABAP 运行时中的 1 到 16 字节的十进制字段，对应于 ABAP 类型 P。打包的数字存储了两个十进制数字。BCD 字段在其关联的 byteLength 和 unicodeByteLength 属性中声明其字段长度。对于 BCD 字段，这些属性可以有 1 到 16 字节的值，这两个属性具有相同的值。BCD 字段声明字段值在其关联的十进制属性中包含的十进制数字数。对于 BCD 字段，这个十进制属性可以有 1 到 14 个数字之间的值。BCD 字段值以 Java 表示为 java.math.BigDecimal。

DECF16

DECF16 字段是一个十进制浮点，存储在底层 JCo 和 ABAP 运行时中的 8 字节 IEEE 754 decimal64 浮点值，对应于 ABAP 类型 decfloat16。DECF16 字段的值具有 16 个十进制数字。DECF16 字段的值以 Java 表示，表示为 java.math.BigDecimal。

DECF34

DECF34 字段是作为底层 JCo 和 ABAP 运行时 16 字节 IEEE 754 十进制128 浮点值存储的十进制浮动点，对应于 ABAP 类型 decfloat34。DECF34 字段的值为 34 个十进制数字。DECF34 字段的值在 Java 中以 java.math.BigDecimal 表示。

十六进制字段类型

十六进制字段包含原始二进制数据。支持以下十六进制字段类型：

BYTE

BYTE 字段是一个固定大小字节字符串，存储为底层 JCo 和 ABAP 运行时中的字节数组，对应于 ABAP 类型 X。BYTE 字段在关联的 byteLength 和 unicodeByteLength 属性中声明其字段长度。如果是 BYTE 字段，这些属性可以有 1 到 65535 字节之间的值，这两个属性具有相同的值。BYTE 字段的值在 Java 中作为 byte[] 对象表示。

字符串字段类型

string 字段引用变量长度字符串值。该字符串值的长度在运行时前不会被修复。字符串值的存储是在底层 JCo 和 ABAP 运行时中动态创建的。字符串字段本身的存储已被修复，仅包含字符串标头。

字符串

STRING 字段引用字符字符串，并存储在底层 JCo 和 ABAP 运行时中，作为 8 字节值。它对应于 ABAP 类型 G。STRING 字段的值在 Java 中以 java.lang.String 对象表示。

XSTRING

XSTRING 字段引用字节字符串，并作为 8 字节值存储在底层 JCo 和 ABAP 运行时中。它对应于 ABAP 类型 Y。STRING 字段的值在 Java 中以 byte[] 对象表示。

复杂字段类型

复杂的字段可以是结构或表字段类型。下表总结了这些复杂的字段类型。

字段类型	对应的 Java 类型	字节长度	Unicode Byte Length	number Decimals Digits	描述
结构	**org.fusesource.camel.component.sap.model.rfc.Structure**	单个字段字节长度总量	单个字段 Unicode 字节长度总数	-	ABAP 类型 'u' 和 'v': Heterogeneous Structure
表	**org.fusesource.camel.component.sap.model.rfc.Table**	行结构的字节长度	行结构的 Unicode 字节长度	-	ABAP 类型"h"：表

structure 字段类型

STRUCTURE 字段包含一个结构对象，并作为 ABAP 结构记录存储在底层 JCo 和 ABAP 运行时中。它对应于 ABAP 类型 u 或 v。STRUCTURE 字段的值以 Java 表示为具有接口 org.fusesource.camel.component.sap.model.rfc.Structure 的结构对象。

表字段类型

TABLE 字段包含一个表对象，并作为 ABAP 内部表存储在底层 JCo 和 ABAP 运行时中。它对应于 ABAP 类型 h。字段的值在 Java 中由接口 org.fusesource.camel.component.sap.model.rfc.Table 的表对象表示。

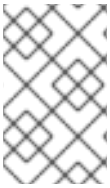
表对象

表对象是一个同构列表数据结构，其中包含具有相同结构的结构对象行。这个接口扩展了 java.util.List 和 org.eclipse.emf.ecore.EObject 接口。

```
public interface Table<S extends Structure>
    extends org.eclipse.emf.ecore.EObject,
    java.util.List<S> {

    /**
     * Creates and adds table row at end of row list
     */
    S add();

    /**
     * Creates and adds table row at index in row list
     */
    S add(int index);

}
```

使用列表接口中定义的标准方法访问和管理表对象中的行列表。另外，表接口提供了在行列表中创建和添加结构对象的两个工厂方法。

表对象使用 Eclipse Modeling Framework (EMF)在组件运行时实现，并支持该框架的 EObject 接口。表对象的实例已附加元数据，它们定义和限制它提供的行的结构和内容。可以使用 EMF 提供的标准方法访问和内省此元数据。详情请参考 EMF 文档。



注意

尝试添加或设置错误类型的行结构值会抛出异常。

290.10. IDOC 的消息正文

idoc 消息类型

当使用 IDoc Camel SAP 端点时，消息正文的类型取决于您正在使用的特定端点。

对于 **sap-idoc-destination** 端点或 **sap-qidoc-destination** 端点，消息正文是 **Document** 类型：

```
org.fusesource.camel.component.sap.model.idoc.Document
```

对于 **sap-idoclist-destination** 端点，**sap-qidoclist-destination** 端点或 **sap-idoclist-server** 端点，消息正文为 **DocumentList** 类型：

```
org.fusesource.camel.component.sap.model.idoc.DocumentList
```

IDoc 文档模型

对于 **Camel SAP** 组件，使用 **Eclipse Modelling Framework (EMF)**进行建模，该文档围绕底层 **SAP IDoc API** 提供打包程序 **API**。这个模型中最重要的类型是：

```
org.fusesource.camel.component.sap.model.idoc.Document
org.fusesource.camel.component.sap.model.idoc.Segment
```

文档 类型表示 **IDoc** 文档实例。在概述中，**Document** 接口会公开以下方法：

```
// Java
package org.fusesource.camel.component.sap.model.idoc;
...
public interface Document extends EObject {
    // Access the field values from the IDoc control record
    String getArchiveKey();
    void setArchiveKey(String value);
    String getClient();
    void setClient(String value);
    ...
```

```
    // Access the IDoc document contents
    Segment getRootSegment();
}
```

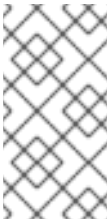
以下方法通过 Document 接口公开：

访问控制记录的方法

大多数方法都是访问或修改 IDoc 控制记录的字段值。这些方法是格式为 AttributeName,AttributeName，其中 AttributeName 是字段值的名称（请参阅 表 290.2 “idoc 文档属性”）。

访问文档内容的方法

getRootSegment 方法提供对文档内容的访问(IDoc 数据存储)，将内容返回为 Segment 对象。每个分段对象可以包含任意数量的子段，段可以嵌套到任意程度。



注意

片段层次结构的确切布局由文档的特定 IDoc 类型定义。在创建（或读取）片段层次结构时，您必须确定遵循 IDoc 类型定义的确切结构。

Segment 类型用于访问 IDoc 文档的数据记录，其中片段会根据文档的 IDoc 类型所定义的结构进行布局。在概述中，Segment 接口会公开以下方法：

```
// Java
package org.fusesource.camel.component.sap.model.idoc;
...
public interface Segment extends EObject, java.util.Map<String, Object> {
    // Returns the value of the '<em><b>Parent</b></em>' reference.
    Segment getParent();

    // Return a immutable list of all child segments
    <S extends Segment> EList<S> getChildren();

    // Returns a list of child segments of the specified segment type.
    <S extends Segment> SegmentList<S> getChildren(String segmentType);

    EList<String> getTypes();

    Document getDocument();

    String getDescription();

    String getType();
```

```
    String getDefinition();

    int getHierarchyLevel();

    String getIdocType();

    String getIdocTypeExtension();

    String getSystemRelease();

    String getApplicationRelease();

    int getNumFields();

    long getMaxOccurrence();

    long getMinOccurrence();

    boolean isMandatory();

    boolean isQualified();

    int getRecordLength();

    <T> T get(Object key, Class<T> type);
}
```

getChildren (String segmentType) 方法对于向网段添加新的（嵌套）子对象特别有用。它返回一个类型为 SegmentList 的对象，它定义如下：

```
// Java
package org.fusesource.camel.component.sap.model.idoc;
...
public interface SegmentList<S extends Segment> extends EObject, EList<S> {
    S add();

    S add(int index);
}
```

因此，要创建 E1SCU_CRE 类型的数据记录，您可以使用类似如下的 Java 代码：

```
Segment rootSegment = document.getRootSegment();

Segment E1SCU_CRE_Segment = rootSegment.getChildren("E1SCU_CRE").add();
```

IDoc 如何与 Document 对象相关

根据 SAP 文档，IDoc 文档由以下主要部分组成：

控制记录

控制记录（包含 IDoc 文档的元数据）由 Document 对象可能性对象中的属性表示，详情请参阅表 290.2 “idoc 文档属性”。

数据记录

数据记录由 Segment 对象表示，这些对象被构建为片段的嵌套层次结构。您可以通过 Document.getRootSegment 方法访问根段。

状态记录

在 Camel SAP 组件中，状态记录 不由 文档模型表示。但是，您可以通过控制记录上的 status 属性访问最新的状态值。

创建文档实例示例

例如，例 290.1 “在 Java 中创建 IDoc 文档” 如何使用 Java 中的 IDoc 模型 API，使用 IDoc 类型 FLCUSTOMER_CREATEFROMDATA01 创建 IDoc 文档。

例 290.1. 在 Java 中创建 IDoc 文档

```
// Java
import org.fusesource.camel.component.sap.model.idoc.Document;
import org.fusesource.camel.component.sap.model.idoc.Segment;
import org.fusesource.camel.component.sap.util.IDocUtil;

import org.fusesource.camel.component.sap.model.idoc.Document;
import org.fusesource.camel.component.sap.model.idoc.DocumentList;
import org.fusesource.camel.component.sap.model.idoc.IdocFactory;
import org.fusesource.camel.component.sap.model.idoc.IdocPackage;
import org.fusesource.camel.component.sap.model.idoc.Segment;
import org.fusesource.camel.component.sap.model.idoc.SegmentChildren;
...
//
// Create a new IDoc instance using the modelling classes
//

// Get the SAP Endpoint bean from the Camel context.
// In this example, it's a 'sap-idoc-destination' endpoint.
SapTransactionalIDocDestinationEndpoint endpoint =
    exchange.getContext().getEndpoint(
        "bean:SapEndpointBeanID",
        SapTransactionalIDocDestinationEndpoint.class
    );

// The endpoint automatically populates some required control record attributes
Document document = endpoint.createDocument()

// Initialize additional control record attributes
```

```
document.setMessageType("FLCUSTOMER_CREATEFROMDATA");
document.setRecipientPartnerNumber("QUICKCLNT");
document.setRecipientPartnerType("LS");
document.setSenderPartnerNumber("QUICKSTART");
document.setSenderPartnerType("LS");

Segment rootSegment = document.getRootSegment();

Segment E1SCU_CRE_Segment = rootSegment.getChildren("E1SCU_CRE").add();

Segment E1BPSCUNEW_Segment =
E1SCU_CRE_Segment.getChildren("E1BPSCUNEW").add();
E1BPSCUNEW_Segment.put("CUSTNAME", "Fred Flintstone");
E1BPSCUNEW_Segment.put("FORM", "Mr.");
E1BPSCUNEW_Segment.put("STREET", "123 Rubble Lane");
E1BPSCUNEW_Segment.put("POSTCODE", "01234");
E1BPSCUNEW_Segment.put("CITY", "Bedrock");
E1BPSCUNEW_Segment.put("COUNTR", "US");
E1BPSCUNEW_Segment.put("PHONE", "800-555-1212");
E1BPSCUNEW_Segment.put("EMAIL", "fred@bedrock.com");
E1BPSCUNEW_Segment.put("CUSTTYPE", "P");
E1BPSCUNEW_Segment.put("DISCOUNT", "005");
E1BPSCUNEW_Segment.put("LANGU", "E");
```

文档属性

表 290.2 “idoc 文档属性” 显示您可以在 **Document** 对象上设置的控制记录属性。

表 290.2. idoc 文档属性

属性	length	SAP Field	描述
archiveKey	70	**ARCKEY**	EDI 归档密钥
client	3	**MANDT**	客户端
creationDate	8	**CREDAT**	创建日期 IDoc
creationTime	6	**CRETIM**	创建时间 IDoc
direction	1	**DIRECT**	方向
eDIMessage	14	**REFMES**	对消息的引用
eDIMessageGroup	14	**REFGRP**	对消息组的引用
eDIMessageType	6	**STDMES**	EDI 消息类型

属性	length	SAP Field	描述
eDIStandardFlag	1	**STD**	EDI 标准
eDIStandardVersion	6	**STDVRS**	EDI 标准版本
eDITransmissionFile	14	**REFINT**	对交换文件的引用
iDocCompoundType	8	**DOCTYP**	idoc 类型
iDocNumber	16	**文档NUM**	idoc 号
iDocSAPRelease	4	**DOCREL**	SAP IDoc 版本
iDocType	30	**IDOCTP**	基本 IDoc 类型的名称
iDocTypeExtension	30	**CIMTYP**	扩展类型的名称
messageCode	3	**MESCOD**	逻辑消息代码
messageFunction	3	**MESFCT**	逻辑消息功能
messageType	30	**MESTYP**	逻辑消息类型
outputMode	1	**OUTMOD**	输出模式
recipientAddress	10	**RCVSAD**	接收器地址(SADR)
recipientLogicalAddress	70	**RCVLAD**	接收器的逻辑地址
recipientPartnerFunction	2	**RCVPFC**	接收器合作伙伴功能
recipientPartnerNumber	10	**RCVPRN**	合作伙伴接收器数
recipientPartnerType	2	**RCVPRT**	合作伙伴接收器类型
recipientPort	10	**RCVPOR**	接收器端口(SAP 系统、EDI 子系统)
senderAddress		**SNDSAD**	发送者地址(SADR)
senderLogicalAddress	70	**SNDLAD**	发送者的逻辑地址

属性	length	SAP Field	描述
senderPartnerFunction	2	**SNDPFC**	发送者的合作伙伴功能
senderPartnerNumber	10	**SNDPRN**	合作伙伴发送者数
senderPartnerType	2	**SNDPRT**	合作伙伴发送者类型
senderPort	10	**SNDPOR**	发送者端口(SAP 系统、EDI 子系统)
serialization	20	**SERIAL**	EDI/ALE: Serialization 字段
status	2	**状态**	IDoc 的状态
testFlag	1	**测试**	test 标记

在 Java 中设置文档属性

在 Java 中设置控制记录属性时（来自 表 290.2 “idoc 文档属性”），则遵循 Java bean 属性的常规约定。也就是说，可以通过 getName 和 setName 方法访问 name 属性，用于获取和设置属性值。例如，i Doc Type、iDocTypeExtension 和 messageType 属性可以设置为在 Document 对象中如下所示：

```
// Java
document.setIDocType("FLCUSTOMER_CREATEFROMDATA01");
document.setIDocTypeExtension("");
document.setMessageType("FLCUSTOMER_CREATEFROMDATA");
```

在 XML 中设置文档属性

在 XML 中设置控制记录属性时，必须在 idoc:Document 元素上设置属性。例如，i Doc Type、iDocTypeExtension 和 messageType 属性可以设置如下：

```
<?xml version="1.0" encoding="ASCII"?>
<idoc:Document ...
            iDocType="FLCUSTOMER_CREATEFROMDATA01"
            iDocTypeExtension=""
            messageType="FLCUSTOMER_CREATEFROMDATA" ... >
   ...
</idoc:Document>
```

290.11. 事务支持

BAPI 事务模型

SAP 组件支持 BAPI 事务模型，用于与 SAP 的出站通信。具有包含 transacted 选项设为 true 的 URL 的目的地端点，在端点的出站连接上启动有状态会话，并使用交换注册 Camel 同步对象。

此同步对象调用 BAPI_TRANSACTION_COMMIT，并在消息交换处理完成后结束有状态会话。如果消息交换的处理失败，同步对象调用 BAPI 服务器方法 BAPI_TRANSACTION_ROLLBACK，并结束有状态会话。

RFC 事务模型

tRFC 协议通过唯一事务标识符(TID)识别每个事务请求(TID)来实现 AT-MOST-ONCE 交付和处理保证。TID 遵循协议中发送的每个请求。使用 tRFC 协议发送应用程序必须识别在发送请求时具有唯一 TID 的请求的每个实例。应用程序可以多次发送带有给定 TID 的请求，但协议可确保最多在接收系统中发送和处理请求。当发送请求时，应用程序可能会选择重新发送带有给定 TID 的请求，因此是否向接收系统中发送和处理请求。通过在遇到通信错误时重新发送请求，使用 tRFC 协议的客户端应用程序可以确保 EXACTLY-ONCE 交付和处理保证。

要使用哪个事务模型？

BAPI 事务是一个应用程序级别的事务，在某种程度上，它会对 BAPI 方法或 SAP 数据库中 BAPI 方法或 RFC 功能执行的持久性数据更改施加 ACID 保证。RFC 事务是一个通信事务，它对 BAPI 方法和/或 RFC 功能的请求施加交付保证(AT-MOST-ONCE、EXACTLY-ONCE-IN-ORDER)。

事务 RFC 目标端点

以下目的地端点支持 RFC 事务：

- *sap-trfc-destination*
- *sap-qrfc-destination*

单个 Camel 路由可以包含多个事务 RFC 目标端点，将消息发送到多个 RFC 目的地，甚至多次将消息发送到同一 RFC 目的地。这意味着 Camel SAP 组件可能需要跟踪每个 Exchange 对象的事务 ID (TID)。现在，如果路由处理失败且必须重试，则情况会非常复杂。RFC 事务语义要求每个 RFC 目的地都必须使用第一次 使用的同一 TID 调用（以及每个目的地的 TID 相互不同）。换句话说，Camel SAP 组件必须跟踪在路由中使用的 TID，并记住此信息，以便 TID 能够按照正确顺序重新执行。

默认情况下，Camel 不提供一个机制，它允许交换知道路由中的位置。要提供这样的机制，需要将 CurrentProcessorDefinitionInterceptStrategy 拦截器安装到 Camel 运行时。此拦截器必须安装在 Camel 运行时，才能跟踪使用 Camel SAP 组件的路由中的 TID。有关如何配置拦截器的详情，请参考 "TRFC 和 qRFC 目的地的拦截器"一节。

事务 RFC 服务器端点

以下服务器端点支持 RFC 事务：

- sap-trfc-server

当 Camel 交换处理事务请求遇到处理错误时，Camel 通过其标准错误处理机制处理处理错误。如果 Camel 路由处理被配置为将错误传播到调用者，则发起交换的 SAP 服务器端点会记录失败，发送 SAP 系统会收到错误通知。然后，发送 SAP 系统可以通过发送另一个具有相同 TID 的事务请求来响应请求，以再次处理请求。

290.12. RFC 的 XML SERIALIZATION

概述

SAP 请求和响应对象支持 XML 序列化格式，从而使这些对象可以序列化至 XML 文档。

XML 命名空间

每个软件仓库中的 RFC 都定义了特定的 XML 命名空间，用于编写其 Request 和 Response 对象的序列化形式。此命名空间 URL 的形式如下：

```
http://sap.fusesource.org/rfc/<Repository Name>/<RFC Name>
```

RFC 命名空间 URL 具有通用 http://sap.fusesource.org/rfc 前缀，后跟定义 RFC 元数据的存储库的名称。URL 中的最后一个组件是 RFC 本身的名称。

请求和响应 XML 文档

SAP 请求对象使用名为 Request 的 root 元素序列化为 XML 文档，并由请求的 RFC 的命名空间的范围。

```
<?xml version="1.0" encoding="ASCII"?>
```

```
<BOOK_FLIGHT:Request
    xmlns:BOOK_FLIGHT="http://sap.fusesource.org/rfc/nplServer/BOOK_FLIGHT">
   ...
</BOOK_FLIGHT:Request>
```

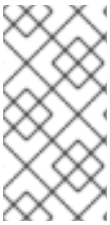
SAP 响应对象使用名为 Response 的 root 元素序列化为 XML 文档，并由响应的 RFC 的命名空间限定。

```
<?xml version="1.0" encoding="ASCII"?>
<BOOK_FLIGHT:Response
    xmlns:BOOK_FLIGHT="http://sap.fusesource.org/rfc/nplServer/BOOK_FLIGHT">
   ...
</BOOK_FLIGHT:Response>
```

结构字段

参数列表或嵌套结构中的结构字段被序列化为元素。序列化结构的元素名称对应于它所驻留的参数列表、结构或表行中的结构的字段名称。

```
<BOOK_FLIGHT:FLTINFO
    xmlns:BOOK_FLIGHT="http://sap.fusesource.org/rfc/nplServer/BOOK_FLIGHT">
   ...
</BOOK_FLIGHT:FLTINFO>
```



注意

RFC 命名空间中的 structure 元素的类型名称对应于定义结构的记录元数据对象的名称，如下例所示：

```
<xs:schema
    targetNamespace="http://sap.fusesource.org/rfc/nplServer/BOOK_FLIGHT">
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
   ...
   <xs:complexType name="FLTINFO_STRUCTURE”>
   ...
   </xs:complexType>
   ...
</xs:schema>
```

在指定 JAXB bean to marshal 和 unmarshal the 结构时，这种区别很重要，如 第 290.14.3 节 “示例 3：从 SAP 处理请求” 中所示。

表字段

参数列表或嵌套结构中的表字段作为元素序列化。序列化结构的元素名称对应于它所驻留的参数列表、结构或表行中的表的字段名称。table 元素包含一系列行元素，用于保存表行条目的序列化值。

```
<BOOK_FLIGHT:CONNINFO
    xmlns:BOOK_FLIGHT="http://sap.fusesource.org/rfc/nplServer/BOOK_FLIGHT">
   <row ... > ... </row>
   ...
   <row ... > ... </row>
</BOOK_FLIGHT:CONNINFO>
```

注意

RFC 命名空间中的表元素的类型名称对应于记录元数据对象的名称，该对象定义由 _TABLE 后缀的表行结构。RFC 名称中表行元素的类型名称对应于定义表行结构的记录元数据对象的名称，如下例所示：

```
<xs:schema
    targetNamespace="http://sap.fusesource.org/rfc/nplServer/BOOK_FLIGHT"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
   ...
   <xs:complextType name="CONNECTION_INFO_STRUCTURE_TABLE”>
      <xs:sequence>
         <xs:element
            name="row”
            minOccures="0”
            maxOccurs="unbounded”
            type="CONNECTION_INFO_STRUCTURE”/>
         ...
         <xs:sequence>
      </xs:sequence>
   </xs:complexType>

   <xs:complextType name="CONNECTION_INFO_STRUCTURE”>
         ...
   </xs:complexType>
   ...
</xs:schema>
```

在指定 JAXB bean to marshal 和 unmarshal the 结构时，这种区别很重要，如 第 290.14.3 节 “示例 3：从 SAP 处理请求” 中所示。

Elementary 字段

参数列表或嵌套结构中的 Elementary 字段被序列化为封闭参数列表或结构元素的属性。serialized 字段的属性名称对应于它所在的参数列表、结构或表行条目中字段的字段名称，如下例所示：

```
<?xml version="1.0" encoding="ASCII"?>
<BOOK_FLIGHT:Request
    xmlns:BOOK_FLIGHT="http://sap.fusesource.org/rfc/nplServer/BOOK_FLIGHT"
    CUSTNAME="James Legrand"
   PASSFORM="Mr"
   PASSNAME="Travelin Joe"
   PASSBIRTH="1990-03-17T00:00:00.000-0500"
   FLIGHTDATE="2014-03-19T00:00:00.000-0400"
   TRAVELAGENCYNUMBER="00000110"
   DESTINATION_FROM="SFO"
   DESTINATION_TO="FRA"/>
```

日期和时间格式

日期和时间字段使用以下格式序列化为属性值：

```
yyyy-MM-dd'T'HH:mm:ss.SSSZ
```

日期字段只对年、月、天和时区组件进行了序列化：

```
DEPDATE="2014-03-19T00:00:00.000-0400"
```

时间字段只序列化为小时、分钟、第二个、millisecond 和 timezone 组件设置：

```
DEPTIME="1970-01-01T16:00:00.000-0500"
```

290.13. 用于 IDOC 的 XML SERIALIZATION

概述

IDoc 消息正文可以序列化为 XML 字符串格式，以帮助内置类型转换器。

XML 命名空间

每个序列化 IDoc 都与 XML 命名空间关联，其具有以下通用格式：

```
http://sap.fusesource.org/idoc/repositoryName/idocType/idocTypeExtension/systemRelease/applicationRelease
```

repositoryName （远程 SAP 元数据存储库的名称）和 idocType (IDoc document type)都是强制

的，但命名空间的其他组件可以留空。例如，您可以有一个类似如下的 XML 命名空间：

```
http://sap.fusesource.org/idoc/MY_REPO/FLCUSTOMER_CREATEFROMDATA01///
```

内置类型转换器

Camel SAP 组件具有一个内置类型转换器，能够将 Document 对象或 DocumentList 对象转换为 String 类型或从 String 类型转换。

例如，要将 Document 对象序列化为 XML 字符串，只需将以下行添加到 XML DSL 中的路由中：

```
<convertBodyTo type="java.lang.String"/>
```

您还可以使用此方法将 XML 消息序列化为 Document 对象。例如，如果当前消息正文是序列化 XML 字符串，您可以通过将以下行添加到 XML DSL 中的路由来将其转换为 Document 对象：

```
<convertBodyTo type="org.fusesource.camel.component.sap.model.idoc.Document"/>
```

XML 格式的 IDoc 消息正文示例

当您将 IDoc 消息转换为 String 时，它将序列化为 XML 文档，其中 root 元素是 idoc:Document（单个文档）或 idoc:DocumentList （用于文档列表）。例 290.2 “XML 中的 idoc 消息正文” 显示被序列化为 idoc:Document 元素的单个 IDoc 文档。

例 290.2. XML 中的 idoc 消息正文

```
<?xml version="1.0" encoding="ASCII"?>
<idoc:Document
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:FLCUSTOMER_CREATEFROMDATA01--
-="http://sap.fusesource.org/idoc/XXX/FLCUSTOMER_CREATEFROMDATA01///"
   xmlns:idoc="http://sap.fusesource.org/idoc"
   creationDate="2015-01-28T12:39:13.980-0500"
   creationTime="2015-01-28T12:39:13.980-0500"
   iDocType="FLCUSTOMER_CREATEFROMDATA01"
   iDocTypeExtension=""
   messageType="FLCUSTOMER_CREATEFROMDATA"
   recipientPartnerNumber="QUICKCLNT"
   recipientPartnerType="LS"
   senderPartnerNumber="QUICKSTART"
   senderPartnerType="LS">
  <rootSegment xsi:type="FLCUSTOMER_CREATEFROMDATA01---:ROOT" document="/">
    <segmentChildren parent="//@rootSegment">
      <E1SCU_CRE parent="//@rootSegment" document="/">
```

```
        <segmentChildren parent="//@rootSegment/@segmentChildren/@E1SCU_CRE.0">
          <E1BPSCUNEW parent="//@rootSegment/@segmentChildren/@E1SCU_CRE.0"
              document="/"
              CUSTNAME="Fred Flintstone" FORM="Mr."
              STREET="123 Rubble Lane"
              POSTCODE="01234"
              CITY="Bedrock"
              COUNTR="US"
              PHONE="800-555-1212"
              EMAIL="fred@bedrock.com"
              CUSTTYPE="P"
              DISCOUNT="005"
              LANGU="E"/>
        </segmentChildren>
      </E1SCU_CRE>
    </segmentChildren>
  </rootSegment>
</idoc:Document>
```

290.14. SAP 示例

290.14.1. 示例 1：从 SAP 读取数据

概述

本例演示了一个从 SAP 读取 FlightCustomer 业务对象数据的路由。该路由调用 FlightCustomer BAPI 方法，即 BAPI_FLCUST_GETLIST，使用 SAP 同步 RFC 目标端点来检索数据。

用于路由的 Java DSL

示例路由的 Java DSL 如下：

```
from("direct:getFlightCustomerInfo")
    .to("bean:createFlightCustomerGetListRequest")
    .to("sap-srfc-destination:nplDest:BAPI_FLCUST_GETLIST")
    .to("bean:returnFlightCustomerInfo");
```

用于路由的 XML DSL

同一路由的 Spring DSL 如下所示：

```
<route>
    <from uri="direct:getFlightCustomerInfo"/>
    <to uri="bean:createFlightCustomerGetListRequest"/>
```

```
    <to uri="sap-srfc-destination:nplDest:BAPI_FLCUST_GETLIST"/>
    <to uri="bean:returnFlightCustomerInfo"/>
</route>
```

createFlightCustomerGetListRequest bean

createFlightCustomerGetListRequest bean 负责在后续 SAP 端点的 RFC 调用中使用的交换方法中构建 SAP 请求对象。以下代码片段演示了构建请求对象的操作序列：

```
public void create(Exchange exchange) throws Exception {

    // Get SAP Endpoint to be called from context.
    SapSynchronousRfcDestinationEndpoint endpoint =
        exchange.getContext().getEndpoint("sap-srfc-
destination:nplDest:BAPI_FLCUST_GETLIST",
                                    SapSynchronousRfcDestinationEndpoint.class);

    // Retrieve bean from message containing Flight Customer name to
    // look up.
    BookFlightRequest bookFlightRequest =
        exchange.getIn().getBody(BookFlightRequest.class);

    // Create SAP Request object from target endpoint.
    Structure request = endpoint.getRequest();

    // Add Customer Name to request if set
    if (bookFlightRequest.getCustomerName() != null &&
        bookFlightRequest.getCustomerName().length() > 0) {
            request.put("CUSTOMER_NAME",
                    bookFlightRequest.getCustomerName());
        }
    } else {
        throw new Exception("No Customer Name");
    }

    // Put request object into body of exchange message.
    exchange.getIn().setBody(request);
}
```

returnFlightCustomerInfo bean

returnFlightCustomerInfo bean 负责从 SAP 响应对象中提取数据，在其从之前的 SAP 端点接收的 Exchange 方法中提取数据。以下代码片段演示了从响应对象中提取数据的操作序列：

```
public void createFlightCustomerInfo(Exchange exchange) throws Exception {

    // Retrieve SAP response object from body of exchange message.
    Structure flightCustomerGetListResponse =
        exchange.getIn().getBody(Structure.class);
```

```
    if (flightCustomerGetListResponse == null) {
        throw new Exception("No Flight Customer Get List Response");
    }

    // Check BAPI return parameter for errors
    @SuppressWarnings("unchecked")
    Table<Structure> bapiReturn =
        flightCustomerGetListResponse.get("RETURN", Table.class);
    Structure bapiReturnEntry = bapiReturn.get(0);
    if (bapiReturnEntry.get("TYPE", String.class) != "S") {
        String message = bapiReturnEntry.get("MESSAGE", String.class);
        throw new Exception("BAPI call failed: " + message);
    }

    // Get customer list table from response object.
    @SuppressWarnings("unchecked")
    Table<? extends Structure> customerList =
        flightCustomerGetListResponse.get("CUSTOMER_LIST", Table.class);

    if (customerList == null || customerList.size() == 0) {
        throw new Exception("No Customer Info.");
    }

    // Get Flight Customer data from first row of table.
    Structure customer = customerList.get(0);

    // Create bean to hold Flight Customer data.
    FlightCustomerInfo flightCustomerInfo = new FlightCustomerInfo();

    // Get customer id from Flight Customer data and add to bean.
    String customerId = customer.get("CUSTOMERID", String.class);
    if (customerId != null) {
        flightCustomerInfo.setCustomerNumber(customerId);
    }

    ...

    // Put bean into body of exchange message.
    exchange.getIn().setHeader("flightCustomerInfo", flightCustomerInfo);

}
```

290.14.2. 示例 2：将数据写入 SAP

概述

本例演示了一个路由，它在 SAP 中创建 FlightTrip business object 实例。路由调用 FlightTrip BAPI 方法 BAPI_FLTRIP_CREATE，使用目标端点来创建对象。

用于路由的 Java DSL

示例路由的 Java DSL 如下：

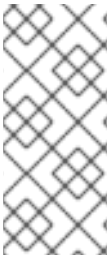
```
from("direct:createFlightTrip")
    .to("bean:createFlightTripRequest")
    .to("sap-srfc-destination:nplDest:BAPI_FLTRIP_CREATE?transacted=true")
    .to("bean:returnFlightTripResponse");
```

用于路由的 XML DSL

同一路由的 Spring DSL 如下所示：

```
<route>
    <from uri="direct:createFlightTrip"/>
    <to uri="bean:createFlightTripRequest"/>
    <to uri="sap-srfc-destination:nplDest:BAPI_FLTRIP_CREATE?transacted=true"/>
    <to uri="bean:returnFlightTripResponse"/>
</route>
```

事务支持



注意

SAP 端点的 URL 将 transacted 选项设置为 true。如 第 290.11 节 “事务支持” 所述，当启用这个选项时，端点可确保在调用 RFC 调用前启动 SAP 事务会话。由于此端点的 RFC 在 SAP 中创建新数据，因此需要此选项才能在 SAP 中进行路由的更改。

填充请求参数

createFlightTripRequest 和 returnFlightTripResponse Bean 负责将请求参数填充到 SAP 请求中，再从 SAP 响应中提取响应参数，其遵循上例中所示的操作序列。

290.14.3. 示例 3：从 SAP 处理请求

概述

本例演示了一个路由，它处理从 SAP 到 BOOK_FLIGHT RFC 的请求，该请求由路由实现。此外，它还演示了组件的 XML 序列化支持，使用 JAXB 到 unmarshal 和 marshal SAP 请求对象，并将对象响应到自定义 Bean。

此路由代表旅行代理 FlightTrip 业务对象 FlightCustomer。路由首先由 SAP 服务器端点接收的 SAP 请求对象到自定义 JAXB bean 中。然后，这个自定义 Bean 在交换到三个子路由中多播，它会收集创建

旅行行程所需的旅行代理、flight 连接和乘客信息。最后的子路由会在 SAP 中创建 flight trip 对象，如上例中所示。最后的子路由也创建并返回自定义 JAXB bean，它被放入 SAP 响应对象中，并由服务器端点返回。

用于路由的 Java DSL

示例路由的 Java DSL 如下：

```
DataFormat jaxb = new JaxbDataFormat("org.fusesource.sap.example.jaxb");

from("sap-srfc-server:nplserver:BOOK_FLIGHT")
    .unmarshal(jaxb)
    .multicast()
    .to("direct:getFlightConnectionInfo",
        "direct:getFlightCustomerInfo",
        "direct:getPassengerInfo")
    .end()
    .to("direct:createFlightTrip")
    .marshal(jaxb);
```

用于路由的 XML DSL

同一路由的 XML DSL 如下：

```
<route>
    <from uri="sap-srfc-server:nplserver:BOOK_FLIGHT"/>
    <unmarshal>
        <jaxb contextPath="org.fusesource.sap.example.jaxb"/>
    </unmarshal>
    <multicast>
        <to uri="direct:getFlightConnectionInfo"/>
        <to uri="direct:getFlightCustomerInfo"/>
        <to uri="direct:getPassengerInfo"/>
    </multicast>
    <to uri="direct:createFlightTrip"/>
    <marshal>
        <jaxb contextPath="org.fusesource.sap.example.jaxb"/>
    </marshal>
</route>
```

BookFlightRequest bean

以下列表演示了一个 JAXB bean，它来自 SAP BOOK_FLIGHT 请求对象的序列化形式：

```
@XmlRootElement(name="Request",
namespace="http://sap.fusesource.org/rfc/nplServer/BOOK_FLIGHT")
```

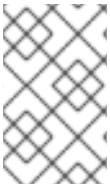
```
@XmlAccessorType(XmlAccessType.FIELD)
public class BookFlightRequest {

    @XmlAttribute(name="CUSTNAME")
    private String customerName;

    @XmlAttribute(name="FLIGHTDATE")
    @XmlJavaTypeAdapter(DateAdapter.class)
    private Date flightDate;

    @XmlAttribute(name="TRAVELAGENCYNUMBER")
    private String travelAgencyNumber;

    @XmlAttribute(name="DESTINATION_FROM")
    private String startAirportCode;

    @XmlAttribute(name="DESTINATION_TO")
    private String endAirportCode;

    @XmlAttribute(name="PASSFORM")
    private String passengerFormOfAddress;

    @XmlAttribute(name="PASSNAME")
    private String passengerName;

    @XmlAttribute(name="PASSBIRTH")
    @XmlJavaTypeAdapter(DateAdapter.class)
    private Date passengerDateOfBirth;

    @XmlAttribute(name="CLASS")
    private String flightClass;

    ...
}
```

BookFlightResponse bean

以下列表演示了一个 JAXB bean，它类似于 SAP BOOK_FLIGHT 响应对象的序列化形式：

```
@XmlRootElement(name="Response",
namespace="http://sap.fusesource.org/rfc/nplServer/BOOK_FLIGHT")
@XmlAccessorType(XmlAccessType.FIELD)
public class BookFlightResponse {

    @XmlAttribute(name="TRIPNUMBER")
    private String tripNumber;

    @XmlAttribute(name="TICKET_PRICE")
    private BigDecimal ticketPrice;

    @XmlAttribute(name="TICKET_TAX")
    private BigDecimal ticketTax;
```

```
    @XmlAttribute(name="CURRENCY")
    private String currency;

    @XmlAttribute(name="PASSFORM")
    private String passengerFormOfAddress;

    @XmlAttribute(name="PASSNAME")
    private String passengerName;

    @XmlAttribute(name="PASSBIRTH")
    @XmlJavaTypeAdapter(DateAdapter.class)
    private Date passengerDateOfBirth;

    @XmlElement(name="FLTINFO")
    private FlightInfo flightInfo;

    @XmlElement(name="CONNINFO")
    private ConnectionInfoTable connectionInfo;

    ...
}
```



注意

响应对象的复杂参数字段被序列化为响应的子元素。

FlightInfo bean

以下列表演示了一个 JAXB bean，它类似于复杂结构参数的序列化形式 FLTINFO ：

```
@XmlRootElement(name="FLTINFO",
namespace="http://sap.fusesource.org/rfc/nplServer/BOOK_FLIGHT")
@XmlAccessorType(XmlAccessType.FIELD)
public class FlightInfo {

    @XmlAttribute(name="FLIGHTTIME")
    private String flightTime;

    @XmlAttribute(name="CITYFROM")
    private String cityFrom;

    @XmlAttribute(name="DEPDATE")
    @XmlJavaTypeAdapter(DateAdapter.class)
    private Date departureDate;

    @XmlAttribute(name="DEPTIME")
    @XmlJavaTypeAdapter(DateAdapter.class)
    private Date departureTime;

    @XmlAttribute(name="CITYTO")
    private String cityTo;
```

```
    @XmlAttribute(name="ARRDATE")
    @XmlJavaTypeAdapter(DateAdapter.class)
    private Date arrivalDate;

    @XmlAttribute(name="ARRTIME")
    @XmlJavaTypeAdapter(DateAdapter.class)
    private Date arrivalTime;

    ...
}
```

ConnectionInfoTable bean

以下列表演示了一个 JAXB bean，它类似于复杂 table 参数的序列化形式 CONNINFO。

注意

JAXB bean 的根元素类型的名称对应于后缀为 _TABLE 的行结构类型的名称，而 bean 包含行元素的列表。

```
@XmlRootElement(name="CONNINFO_TABLE",
namespace="http://sap.fusesource.org/rfc/nplServer/BOOK_FLIGHT")
@XmlAccessorType(XmlAccessType.FIELD)
public class ConnectionInfoTable {

    @XmlElement(name="row")
    List<ConnectionInfo> rows;

    ...
}
```

ConnectionInfo bean

以下列表演示了一个 JAXB bean，它类似于上述表行元素的序列化格式：

```
@XmlRootElement(name="CONNINFO",
namespace="http://sap.fusesource.org/rfc/nplServer/BOOK_FLIGHT")
@XmlAccessorType(XmlAccessType.FIELD)
public class ConnectionInfo {

    @XmlAttribute(name="CONNID")
    String connectionId;

    @XmlAttribute(name="AIRLINE")
    String airline;
```

```
    @XmlAttribute(name="PLANETYPE")
    String planeType;

    @XmlAttribute(name="CITYFROM")
    String cityFrom;

    @XmlAttribute(name="DEPDATE")
    @XmlJavaTypeAdapter(DateAdapter.class)
    Date departureDate;

    @XmlAttribute(name="DEPTIME")
    @XmlJavaTypeAdapter(DateAdapter.class)
    Date departureTime;

    @XmlAttribute(name="CITYTO")
    String cityTo;

    @XmlAttribute(name="ARRDATE")
    @XmlJavaTypeAdapter(DateAdapter.class)
    Date arrivalDate;

    @XmlAttribute(name="ARRTIME")
    @XmlJavaTypeAdapter(DateAdapter.class)
    Date arrivalTime;

    ...
}
```

第 291 章 SAP NETWEAVER 组件

从 Camel 版本 2.12 开始提供

sap-netweaver 使用 HTTP 传输与 SAP NetWeaver 网关 集成。

此 camel 组件仅支持生成者端点。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-sap-netweaver</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

291.1. URI 格式

sap netweaver 网关组件的 URI 方案如下

```
sap-netweaver:https://host:8080/path?username=foo&password=secret
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

291.2. 先决条件

您需要拥有一个 SAP NetWeaver 系统帐户才能利用这个组件。SAP 提供了一个 演示设置，您可以在其中需要帐户。

此组件使用基本身份验证方案登录到 SAP NetWeaver。

291.3. SAPNETWEAVER 选项

SAP NetWeaver 组件没有选项。

SAP NetWeaver 端点使用 URI 语法进行配置：

```
sap-netweaver:url
```

使用以下路径和查询参数：

291.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
url	**需要** 到 SAP net-weaver 网关服务器的 Url。		字符串

291.3.2. 查询参数(6 参数)：

Name	描述	默认值	类型
flatternMap (producer)	如果 JSON Map 仅包含单个条目，那么通过将该单一条目值存储为消息正文来扁平。	true	布尔值
json (producer)	是否以 JSON 格式返回数据。如果此选项为 false，则以 Atom 格式返回 XML。	true	布尔值
jsonAsMap (producer)	将 JSON 从 String 转换为消息正文中的 Map：	true	布尔值
password (producer)	**帐户** 所需的密码。		字符串
username (producer)	**帐户** 所需的用户名。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

291.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.sap-netweaver.enabled	启用 sap-netweaver 组件	true	布尔值
camel.component.sap-netweaver.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

291.5. 消息标头

生产者可以使用以下标头：

Name	类型	描述
CamelNetWeaverCommand	字符串	强制 ：以 MS ADO.Net Data Service 格式执行的命令。

291.6. 例子

这个示例使用 SAP 的 flight 演示示例，该示例可通过互联网在线使用。http://scn.sap.com/docs/DOC-31221

在以下路由中，我们使用以下 url 请求 SAP NetWeaver demo 服务器

```
https://sapes1.sapdevcenter.com/sap/opu/odata/IWBEP/RMTSAMPLEFLIGHT_2/
```

我们希望执行以下命令

```
FlightCollection(AirLineID='AA',FlightConnectionID='0017',FlightDate=datetime'2012-08-29T00%3A00%3A00')
```

要获得给定动态的动态详情。命令语法采用 MS ADO.Net Data Service 格式。

我们有以下 Camel 路由

```
from("direct:start")
    .setHeader(NetWeaverConstants.COMMAND, constant(command))
    .toF("sap-netweaver:%s?username=%s&password=%s", url, username, password)
    .to("log:response")
    .to("velocity:flight-info.vm")
```

其中 url、username、password 和 command 定义为：

```
    private String username = "P1909969254";
    private String password = "TODO";
    private String url =
"https://sapes1.sapdevcenter.com/sap/opu/odata/IWBEP/RMTSAMPLEFLIGHT_2/";
    private String command =
"FlightCollection(AirLineID='AA',FlightConnectionID='0017',FlightDate=datetime'2012-08-
29T00%3A00%3A00')";
```

密码无效。您需要在 SAP 上创建帐户，首先运行演示。

velocity 模板用于格式化对基本 HTML 页面的响应

```
<html>
  <body>
  Flight information:

  <p/>
  <br/>Airline ID: $body["AirLineID"]
  <br/>Aircraft Type: $body["AirCraftType"]
  <br/>Departure city: $body["FlightDetails"]["DepartureCity"]
  <br/>Departure airport: $body["FlightDetails"]["DepartureAirPort"]
  <br/>Destination city: $body["FlightDetails"]["DestinationCity"]
  <br/>Destination airport: $body["FlightDetails"]["DestinationAirPort"]

  </body>
</html>
```

在运行应用程序时，您会获得 sampel 输出：

```
Flight information:
Airline ID: AA
```

```
Aircraft Type: 747-400
Departure city: new york
Departure airport: JFK
Destination city: SAN FRANCISCO
Destination airport: SFO
```

291.7. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- HTTP

第 292 章 调度程序组件

从 Camel 版本 2.15 开始提供

scheduler: 组件用于在调度程序触发时生成消息交换。此组件与 Timer 组件类似，但它在调度方面提供更多功能。另外，此组件也使用 JDK ScheduledExecutorService。其中，因为计时器使用 JDK Timer。

您只能消耗来自此端点的事件。

292.1. URI 格式

```
scheduler:name[?options]
```

其中 name 是调度程序的名称，它在端点之间创建和共享。因此，如果您对所有调度程序端点使用相同的名称，则只使用一个调度程序线程池和线程，但您可以将线程池配置为允许更多的并发线程。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

注：所生成的交换的 IN 正文为 null。因此，exchange.getIn （）.getBody （） 返回 null。

292.2. 选项

Scheduler 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
concurrentTasks (scheduler)	调度线程池使用的线程数。默认情况下，使用单个线程	1	int
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Scheduler 端点使用 URI 语法进行配置：

scheduler:name

使用以下路径和查询参数：

292.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	**必需** 调度程序的名称		字符串

292.2.2. 查询参数(20 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外，该处理程序记录在 WARN/ERROR 级别并忽略。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，该例外记录在 WARN/ERROR 级别并忽略。		ExceptionHandler
exchangePattern (consumer)	在创建交换时设置默认交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。换句话说，在轮询收集信息时发生错误，例如，对文件网络的访问失败，因此 Camel 无法访问它来扫描文件。默认实施将记录 WARN 级别导致的异常并忽略它。		PollingConsumerPoll Strategy
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int

Name	描述	默认值	类型
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
concurrentTasks (scheduler)	调度线程池使用的线程数。默认情况下，使用单个线程	1	int
delay (scheduler)	下一次轮询前的时间（毫秒）。默认值为 500。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。默认值为 1000。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。此选项允许您在多个消费者之间共享线程池。		ScheduledExecutor Service
scheduler (scheduler)	允许插件自定义 org.apache.camel.spi.ScheduledPollConsumerScheduler，以用作在轮询消费者运行时触发的调度程序。默认实施使用 ScheduledExecutorService，有一个 Quartz2 和 Spring，它支持 CRON 表达式。注意：如果使用自定义调度程序，则不能使用 initialDelay 的选项，则使用FixedDelay、timeUnit 和 scheduledExecutorService。使用文本 quartz2 来引用使用 Quartz2 调度程序；并使用文本 Spring 来使用基于 Spring 的；并使用文本 #myScheduler 来引用 registry 中的 id 的自定义调度程序。如需示例，请参阅 Quartz2 页。	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值

Name	描述	默认值	类型
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

292.3. 更多信息

这个组件是一个调度程序 Polling Consumer，您可以在其中找到有关以上选项的更多信息，以及 Polling Consumer 页面的示例。

292.4. 交换属性

当触发计时器时，它会将以下信息作为属性添加到 交换 中：

Name	类型	描述
Exchange.TIMER_NAME	**字符串**	**name** 选项的值。
Exchange.TIMER_FIRED_TIME	**Date**	消费者触发的时间。

292.5. 示例

要设置一个路由，该路由每 60 秒生成事件：

```
from("scheduler://foo?delay=60s").to("bean:myBean?method=someMethodName");
```

以上路由将生成一个事件，然后在 Registry （如 JNDI 或 Spring）中调用名为 myBean 的 bean 上的 someMethodName 方法。

以及 Spring DSL 中的路由：

```
<route>
  <from uri="scheduler://foo?delay=60s"/>
  <to uri="bean:myBean?method=someMethodName"/>
</route>
```

292.6. 完成后强制调度程序立即触发

要让调度程序在上一个任务完成后立即触发，您可以设置选项 greedy=true。但要注意，调度程序将始终保持触发所有时间。因此请谨慎使用这个选项。

292.7. 强制调度程序闲置

在有些用例中，您可能希望调度程序触发并有 greedy。但有时您希望"省略调度程序"，没有任务轮询，因此调度程序可以使用 backoff 选项更改为空闲模式。要做到这一点，您需要将密钥 Exchange.SCHEDULER_POLLED_MESSAGES 的交换属性设置为布尔值 false。这将导致使用者表示没有轮询消息。

默认情况下，消费者都会返回到调度程序，每次消费者完成交换时，都会返回 1 消息轮询到调度程序。

292.8. 另请参阅

- 计时器
- quartz

第 293 章 SCHEMATRON 组件

从 Camel 版本 2.15 开始提供

Schematron 是用于验证 XML 实例文档的基于 XML 的语言。它用于在 XML 文档中对数据进行断言，也用于表达操作和业务规则。Schematron 是 ISO 标准。schematron 组件使用 ISO 模式的主要实现。它是一个基于 XSLT 的实施。模式功能规则 通过四个 XSLT 管道运行，它会生成一个最终的 XSLT，它将用作针对 XML 文档运行断言的基础。组件使用 Schematron 规则在端点开始时加载的方式（仅一次），这是减去代表规则的 Java 模板对象的开销。

293.1. URI 格式

```
schematron://path?[options]
```

293.2. URI 选项

Schematron 组件没有选项。

Schematron 端点使用 URI 语法进行配置：

```
schematron:path
```

使用以下路径和查询参数：

293.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
path	**必需** schematron 规则文件的路径。可以位于类路径或文件系统中的位置。		字符串

293.2.2. 查询参数(4 参数)：

Name	描述	默认值	类型
abort (producer)	中止路由的标志，并抛出 schematron 验证异常。	false	布尔值
rules (producer)	使用给定的 schematron 规则而不是从路径加载		模板

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
uriResolver (advanced)	将 URIResolver 设置为用于解析 schematron 包括的规则文件中。		URIResolver

293.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.schematron.enabled	启用 schematron 组件	true	布尔值
camel.component.schematron.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

293.4. HEADERS

Name	描述	类型	in/Out
CamelSchematronValidationStatus	schematron 验证状态：SUCCESS / FAILED	字符串	IN
CamelSchematronValidationReport	XML 格式的 schematron 报告正文。请参见以下示例	字符串	IN

293.5. URI 和路径语法

以下示例演示了如何在 Java DSL 中调用 schematron 处理器。schematron 规则文件从类路径提供：

```
from("direct:start").to("schematron://sch/schematron.sch").to("mock:result")
```

以下示例演示了如何在 XML DSL 中调用 schematron 处理器。schematron 规则文件从文件系统中提供：

```
<route>
  <from uri="direct:start" />
  <to uri="schematron:///usr/local/sch/schematron.sch" />
  <log message="Schematron validation status: ${in.header.CamelSchematronValidationStatus}" />
  <choice>
    <when>
      <simple>${in.header.CamelSchematronValidationStatus} == 'SUCCESS'</simple>
      <to uri="mock:success" />
    </when>
    <otherwise>
      <log message="Failed schematron validation" />
      <setBody>
        <header>CamelSchematronValidationReport</header>
      </setBody>
      <to uri="mock:failure" />
    </otherwise>
  </choice>
</route>
```

提示

在哪里存储架构规则？架构规则可能会随着业务要求而改变，因此建议将这些规则存储在文件系统中。当 schematron 组件端点启动时，规则将编译到 XSLT 中，作为 Java 模板对象。这只执行一次来减去实例化 Java 模板对象的开销，对于大量规则集来说，这可能会产生昂贵的操作，并且假定进程通过 XSLT 转换 的四个管道。因此，如果您发生将规则存储在文件系统中，在更新时，您需要重启路由或组件。然而，在类路径中不会损害这些规则，但您必须构建和部署组件来获取更改。

293.6. 架构规则和报告示例

以下是 schematron 规则的示例

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
  <title>Check Sections 12/07</title>
  <pattern id="section-check">
```

```
    <rule context="section">
      <assert test="title">This section has no title</assert>
      <assert test="para">This section has no paragraphs</assert>
    </rule>
  </pattern>
</schema>
```

以下是 schematron 报告的示例：

```
<?xml version="1.0" encoding="UTF-8"?>
<svrl:schematron-output xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
 xmlns:iso="http://purl.oclc.org/dsdl/schematron"
 xmlns:saxon="http://saxon.sf.net/"
 xmlns:schold="http://www.ascc.net/xml/schematron"
 xmlns:xhtml="http://www.w3.org/1999/xhtml"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema" schemaVersion="" title="">

  <svrl:active-pattern document="" />
  <svrl:fired-rule context="chapter" />
  <svrl:failed-assert test="title" location="/doc[1]/chapter[1]">
    <svrl:text>A chapter should have a title</svrl:text>
  </svrl:failed-assert>
  <svrl:fired-rule context="chapter" />
  <svrl:failed-assert test="title" location="/doc[1]/chapter[2]">
    <svrl:text>A chapter should have a title</svrl:text>
  </svrl:failed-assert>
  <svrl:fired-rule context="chapter" />
</svrl:schematron-output>
```

提示

*有用的链接和资源 - Mulleberry 技术 介绍 Schematron。 PDF 中的良好文档，使您开始实施架构。 * Schematron 官方网站.这包括其他资源的链接*

第 294 章 SCP 组件

从 Camel 版本 2.10 开始提供

camel-jsch 组件支持使用 Jsch 项目的 Client API 的 SCP 协议。jsch 已在 camel 中由 sftp: 协议的 FTP 组件使用。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jsch</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

294.1. URI 格式

```
scp://host[:port]/destination[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

文件名可以在 URI 的 <path> 部分中指定，也可以作为消息上的 "CamelFileName" 标头指定（如果用 code，则为Exchange.FILE_NAME）。

294.2. 选项

SCP 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
verboseLogging (producer)	JSCH 是开箱即用的详细注销。因此，我们默认将日志记录变为 DEBUG 日志记录。但是，将这个选项设置为 true 可再次打开详细日志记录。	false	布尔值
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

SCP 端点使用 URI 语法进行配置：

```
scp:host:port/directoryName
```

使用以下路径和查询参数：

294.2.1. 路径参数(3 参数)：

Name	描述	默认值	类型
主机	需要的 FTP 服务器的主机名		字符串
port	FTP 服务器的端口		int
directoryName	起始目录		字符串

294.2.2. 查询参数(22 参数)：

Name	描述	默认值	类型
disconnect (common)	使用后是否从远程 FTP 服务器断开连接。断开连接将仅断开当前与 FTP 服务器的连接。如果您有要停止的消费者，则需要停止消费者/路由。	false	布尔值
chmod (producer)	允许您在存储的文件上设置 chmod。例如 chmod=664。	664	字符串
filename (producer)	使用 File Language 等表达式来动态设置文件名。对于消费者，它将用作文件名过滤器。对于生成者，它用于评估要写入的文件名。如果设置了表达式，它将优先于 CamelFileName 标头。（注：标题本身也可以是表达式）。表达式选项支持 String 和 Expression 类型。如果表达式是 String 类型，则始终使用 File 语言来评估它。如果表达式是 Expression 类型，则使用指定的 Expression 类型 - 例如，允许您使用 OGNL 表达式。对于消费者，您可以使用它过滤文件名，因此您可以使用 File Language 语法使用当前的文件：mydata-$date:now:yyyyMMdd.txt。生产者支持 CamelOverruleFileName 标头，它优先于任何现有的 CamelFileName 标头；CamelOverruleFileName 是一个只使用一次的标头，它可让您更轻松地临时存储 CamelFileName，且之后必须恢复它。		字符串

Name	描述	默认值	类型
flatten (producer)	flatten 用于扁平化任何前导路径的文件名路径，因此只用于文件名。这可让您以递归方式消耗到子目录中，但当您将文件写入另一个目录时，它们将写入单个目录。在生成者中将其设置为 true 强制执行 CamelFileName 标头中的任何文件名都会被剥离任何前导路径。	false	布尔值
jailStartingDirectory (producer)	用于存放（限制）将文件写入起始目录（和子目录）。这默认是启用的，不允许 Camel 将文件写入外部目录（从而更加安全）。您可以关闭此项，以允许将文件写入起始目录之外的目录，如父目录或 root 文件夹。	true	布尔值
StrictHostKeyChecking (producer)	设置是否使用严格的主机密钥检查。可能的值有：no、yes	否	字符串
allowNullBody (producer)	用于指定在文件写入过程中是否允许 null 正文。如果设置为 true，则会创建一个空文件，当设为 false 时，并尝试向文件组件发送 null 正文，则 'Cannot null body to file.' 的 GenericFileWriteException 将会被抛出。如果将 fileExist 选项设置为 'Override'，则文件将被截断，如果设置为附加文件，则文件将保持不变。	false	布尔值
disconnectOnBatchComplete (producer)	在 Batch upload 完成后是否断开与远程 FTP 服务器的连接。disconnectOnBatchComplete 只会断开当前与 FTP 服务器的连接。	false	布尔值
moveExistingFileStrategy (producer)	策略(Custom Strategy)用于在配置 fileExist=Move 时使用，使用特殊命名令牌来移动文件。默认情况下，如果没有提供自定义策略，则使用实施		FileMoveExisting 策略
connectTimeout (advanced)	设置连接超时，以等待 FTPClient 和 JSCH 使用连接	10000	int
soTimeout (advanced)	仅为 FTPClient 设置 so 超时	300000	int
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
超时 (advanced)	设置用于仅由 FTPClient 使用的数据超时	30000	int
knownHostsFile (security)	设置 known_hosts 文件，以便 jsch 端点可以进行主机密钥验证。您可以使用 classpath: 前缀来加载 classpath 中的文件，而不是文件系统。		字符串

Name	描述	默认值	类型
密码 (security)	用于登录的密码		字符串
preferredAuthentications (security)	设置以逗号分隔的身份验证列表，按首选顺序使用。可能的身份验证方法由 JCraft JSCH 定义。有些示例包括：gssapi-with-mic,publickey,keyboard-interactive,password （如果没有指定 JSCH 和/或系统默认值）。		字符串
privateKeyBytes (security)	将私钥字节设置为端点可以进行私钥验证。这只有在未设置 privateKeyFile 时才使用。否则，该文件将具有优先级。		byte[]
privateKeyFile (security)	将私钥文件设置为端点可以进行私钥验证。您可以使用 classpath: 前缀来加载 classpath 中的文件，而不是文件系统。		字符串
privateKeyFilePassphrase (security)	将私钥文件密语设置为端点可以进行私钥验证。		字符串
用户名 (security)	用于登录的用户名		字符串
useUserKnownHostsFile (security)	如果没有明确配置 knownHostFile，则使用 System.getProperty (user.home)/.ssh/known_hosts 中的主机文件	true	布尔值
密码 (security)	设置一个以逗号分隔的密码列表，按首选顺序使用。可能的密码名称由 JCraft JSCH 定义。某些示例包括：aes128-ctr,aes128-cbc,3des-ctr,3des-cbc,blowfish-cbc,aes192-cbc,aes256-cbc.如果没有指定 JSCH 的默认列表，则将使用。		字符串

294.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.scp.enabled	启用 scp 组件	true	布尔值
camel.component.scp.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Name	描述	默认值	类型
camel.component.scp.verbose-logging	JSCH 是开箱即用的详细注销。因此，我们默认将日志记录变为 DEBUG 日志记录。但是，将这个选项设置为 true 可再次打开详细日志记录。	false	布尔值

294.4. 限制

目前，camel-jsch 只支持 Producer （例如，将文件复制到另一台主机上）。

294.5. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 295 章 CAMEL SCR （已弃用）

从 Camel 2.15 开始提供

SCR 代表服务组件运行时，是 OSGi Declarative Services 规格的实施。SCR 允许任何普通旧 Java 对象公开和使用没有样板代码的 OSGi 服务。

OSGi 框架通过查看其捆绑包中的 SCR 描述符文件来知道您的对象，这些文件通常是通过 org.apache.felix:maven-scr-plugin 等 Java 注解生成的。

在 SCR 捆绑包中运行 Camel 是 Spring DM 和 Blueprint 基于蓝图的解决方案，你和 OSGi 框架之间代码行要少得多。使用 SCR 您的捆绑包可在 Java 世界中完全保留；无需编辑 XML 或属性文件。这可让您完全控制一切，这意味着您选择的 IDE 知道您的项目中的具体内容。

295.1. CAMEL SCR 支持

camel-scr 捆绑包没有包括在 Apache Camel 之前的版本 2.15.0 中，但工件本身可以从 2.12.0 开始用于任何 Camel 版本。

org.apache.camel/camel-scr 捆绑包提供基本类 AbstractCamelRunner，它管理您和帮助程序类 ScrHelper，用于在单元测试中使用 SCR 属性。Camel-scr 功能可用于定义在 SCR 捆绑包中运行 Camel 所需的所有功能和捆绑包。

AbstractCamelRunner 类将 CamelContext 的生命周期与 Service 组件的生命周期相关联，并使用 Camel 的 PropertiesComponent 来处理配置。从 java 类中使一个 Service 组件从 AbstractCamelRunner 进行扩展，并在类级别上添加以下 org.apache.felix.scr.annotations ：

添加所需的注解

```
@Component
@References({
   @Reference(name = "camelComponent",referenceInterface = ComponentResolver.class,
       cardinality = ReferenceCardinality.MANDATORY_MULTIPLE, policy =
ReferencePolicy.DYNAMIC,
       policyOption = ReferencePolicyOption.GREEDY, bind = "gotCamelComponent", unbind =
"lostCamelComponent")
})
```

然后实施 getRouteBuilders （） 方法，该方法返回您要运行的 Camel 路由：

实施 getRouteBuilders （）

```
@Override
protected List<RoutesBuilder> getRouteBuilders() {
    List<RoutesBuilder> routesBuilders = new ArrayList<>();
    routesBuilders.add(new YourRouteBuilderHere(registry));
    routesBuilders.add(new AnotherRouteBuilderHere(registry));
    return routesBuilders;
}
```

最后提供默认配置：

注解中的默认配置

```
@Properties({
  @Property(name = "camelContextId", value = "my-test"),
  @Property(name = "active", value = "true"),
  @Property(name = "...", value = "..."),
  ...
})
```

这一切。并且如果您使用 camel-archetype-scr 来生成一个项目，则这已经需要注意。

以下是由 camel-archetype-scr 生成的完整的 Service 组件类示例：

CamelScrExample.java

```
// This file was generated from org.apache.camel.archetypes/camel-archetype-scr/2.15-
SNAPSHOT
package example;

import java.util.ArrayList;
import java.util.List;

import org.apache.camel.scr.AbstractCamelRunner;
import example.internal.CamelScrExampleRoute;
import org.apache.camel.RoutesBuilder;
import org.apache.camel.spi.ComponentResolver;
import org.apache.felix.scr.annotations.*;
```

```
@Component(label = CamelScrExample.COMPONENT_LABEL, description =
CamelScrExample.COMPONENT_DESCRIPTION, immediate = true, metatype = true)
@Properties({
    @Property(name = "camelContextId", value = "camel-scr-example"),
    @Property(name = "camelRouteId", value = "foo/timer-log"),
    @Property(name = "active", value = "true"),
    @Property(name = "from", value = "timer:foo?period=5000"),
    @Property(name = "to", value = "log:foo?showHeaders=true"),
    @Property(name = "messageOk", value = "Success: {{from}} -> {{to}}"),
    @Property(name = "messageError", value = "Failure: {{from}} -> {{to}}"),
    @Property(name = "maximumRedeliveries", value = "0"),
    @Property(name = "redeliveryDelay", value = "5000"),
    @Property(name = "backOffMultiplier", value = "2"),
    @Property(name = "maximumRedeliveryDelay", value = "60000")
})
@References({
    @Reference(name = "camelComponent",referenceInterface = ComponentResolver.class,
        cardinality = ReferenceCardinality.MANDATORY_MULTIPLE, policy =
ReferencePolicy.DYNAMIC,
        policyOption = ReferencePolicyOption.GREEDY, bind = "gotCamelComponent", unbind =
"lostCamelComponent")
})
public class CamelScrExample extends AbstractCamelRunner {

    public static final String COMPONENT_LABEL = "example.CamelScrExample";
    public static final String COMPONENT_DESCRIPTION = "This is the description for camel-
scr-example.";

    @Override
    protected List<RoutesBuilder> getRouteBuilders() {
        List<RoutesBuilder> routesBuilders = new ArrayList<>();
        routesBuilders.add(new CamelScrExampleRoute(registry));
        return routesBuilders;
    }
}
```

CamelContextId 和 active 属性控制 CamelContext 的名称（默认为 "camel-runner-default"）以及是否要启动它（默认为 "false"。除了这些之外，您还可以添加和使用您想要的相同属性。Camel 的 PropertiesComponent 处理递归属性，并在没有问题的情况下使用回退前缀。

AbstractCamelRunner 将为您的 RouteBuilders 提供这些属性，并提供 Camel 的 PropertiesComponents，它还会在名称匹配时将这些值注入到您的 Service 组件和 RouteBuilder 字段中。可以使用任何可见性级别声明字段，并且支持许多类型（字符串、int、布尔值、URL、...）。

以下是由 camel-archetype-scr 生成的 RouteBuilder 类的示例：

CamelScrExampleRoute.java

```
// This file was generated from org.apache.camel.archetypes/camel-archetype-scr/2.15-SNAPSHOT
package example.internal;

import org.apache.camel.LoggingLevel;
import org.apache.camel.builder.RouteBuilder;
import org.apache.camel.impl.SimpleRegistry;
import org.apache.commons.lang.Validate;

public class CamelScrExampleRoute extends RouteBuilder {

    SimpleRegistry registry;

    // Configured fields
    private String camelRouteId;
    private Integer maximumRedeliveries;
    private Long redeliveryDelay;
    private Double backOffMultiplier;
    private Long maximumRedeliveryDelay;

    public CamelScrExampleRoute(final SimpleRegistry registry) {
        this.registry = registry;
    }

    @Override
    public void configure() throws Exception {
        checkProperties();

        // Add a bean to Camel context registry
        registry.put("test", "bean");

        errorHandler(defaultErrorHandler()
            .retryAttemptedLogLevel(LoggingLevel.WARN)
            .maximumRedeliveries(maximumRedeliveries)
            .redeliveryDelay(redeliveryDelay)
            .backOffMultiplier(backOffMultiplier)
            .maximumRedeliveryDelay(maximumRedeliveryDelay));

        from("{{from}}")
            .startupOrder(2)
            .routeId(camelRouteId)
            .onCompletion()
                .to("direct:processCompletion")
            .end()
            .removeHeaders("CamelHttp*")
            .to("{{to}}");


        from("direct:processCompletion")
            .startupOrder(1)
            .routeId(camelRouteId + ".completion")
            .choice()
                .when(simple("${exception} == null"))
                    .log("{{messageOk}}")
                .otherwise()
                    .log(LoggingLevel.ERROR, "{{messageError}}")
```

```
            .end();
        }
    }

    public void checkProperties() {
        Validate.notNull(camelRouteId, "camelRouteId property is not set");
        Validate.notNull(maximumRedeliveries, "maximumRedeliveries property is not set");
        Validate.notNull(redeliveryDelay, "redeliveryDelay property is not set");
        Validate.notNull(backOffMultiplier, "backOffMultiplier property is not set");
        Validate.notNull(maximumRedeliveryDelay, "maximumRedeliveryDelay property is not
set");
    }
}
```

让我们一起查看 CamelScrExampleRoute。

```
    // Configured fields
    private String camelRouteId;
    private Integer maximumRedeliveries;
    private Long redeliveryDelay;
    private Double backOffMultiplier;
    private Long maximumRedeliveryDelay;
```

这些字段的值通过匹配其名称来使用来自属性的值。

```
        // Add a bean to Camel context registry
        registry.put("test", "bean");
```

如果需要将一些 Bean 添加到 CamelContext 的 registry 中用于路由，您可以执行以下操作：

```
    public void checkProperties() {
        Validate.notNull(camelRouteId, "camelRouteId property is not set");
        Validate.notNull(maximumRedeliveries, "maximumRedeliveries property is not set");
        Validate.notNull(redeliveryDelay, "redeliveryDelay property is not set");
        Validate.notNull(backOffMultiplier, "backOffMultiplier property is not set");
        Validate.notNull(maximumRedeliveryDelay, "maximumRedeliveryDelay property is not
set");
    }
```

最好检查是否设置了所需的参数，并在允许路由启动前具有有意义的值。

```
        from("{{from}}")
            .startupOrder(2)
            .routeId(camelRouteId)
            .onCompletion()
                .to("direct:processCompletion")
```

```
        .end()
        .removeHeaders("CamelHttp*")
        .to("{{to}}");


    from("direct:processCompletion")
        .startupOrder(1)
        .routeId(camelRouteId + ".completion")
        .choice()
            .when(simple("${exception} == null"))
                .log("{{messageOk}}")
            .otherwise()
                .log(LoggingLevel.ERROR, "{{messageError}}")
        .end();
```

请注意，路由中的一切都配置有属性。这基本上使您的 RouteBuilder 成为模板。SCR 允许您通过提供替代配置来创建更多路由实例。有关使用 Camel SCR 捆绑包作为模板 的更多信息。

295.2. SCR 中的 ABSTRACTCAMELRUNNER 的生命周期

1. 当组件的配置策略和强制引用满足 SCR 调用 activate （） 时。这会通过以下调用链创建并设置 CamelContext： activate （） → prepare （） → createCamelContext （） → setupPropertiesComponent （） → configure （） → setupCamelContext （）。最后，上下文在 AbstractCamelRunner.START_DELAY 中定义的延迟后启动，并带有 runWithDelay （）。

2. 当在 OSGi 中注册 Camel 组件（组件Resolver 服务，为了准确）时，SCR 调用 会得到 CamelComponent'（），其重新排期/延迟了 CamelContext 的相同 AbstractCamelRunner.START_DELAY。这种效果会导致 CamelContext 等待，直到载入所有 Camel 组件或它们之间有足够差距。同一逻辑将告知在我们添加更多 Camel 组件时再次尝试失败的 CamelContext。

3. 当 Camel 组件取消注册时，SCR 调用 lostCamelComponent'（）'。此调用什么都不做。

4. 当导致调用 激活（） 的要求之一时，SCR 将调用 deactivate （）。这将关闭 CamelContext。

在(non-OSGi)单元测试中，您应该使用 prepare （） → run （） → stop （） 而不是 activate （） 进行更精细的控制。另外，这使我们可以避免在测试中可能出现 SCR 特定操作。

295.3. 使用 CAMEL-ARCHETYPE-SCR

创建 Camel SCR 捆绑包项目的最简单方法是使用 camel-archetype-scr 和 Maven。

您可以按照以下步骤生成项目：

生成项目

```
$ mvn archetype:generate -Dfilter=org.apache.camel.archetypes:camel-archetype-scr

Choose archetype:
1: local -> org.apache.camel.archetypes:camel-archetype-scr (Creates a new Camel SCR
bundle project for Karaf)
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): : 1
Define value for property 'groupId': : example
[INFO] Using property: groupId = example
Define value for property 'artifactId': : camel-scr-example
Define value for property 'version': 1.0-SNAPSHOT: :
Define value for property 'package': example: :
[INFO] Using property: archetypeArtifactId = camel-archetype-scr
[INFO] Using property: archetypeGroupId = org.apache.camel.archetypes
[INFO] Using property: archetypeVersion = 2.15-SNAPSHOT
Define value for property 'className': : CamelScrExample
Confirm properties configuration:
groupId: example
artifactId: camel-scr-example
version: 1.0-SNAPSHOT
package: example
archetypeArtifactId: camel-archetype-scr
archetypeGroupId: org.apache.camel.archetypes
archetypeVersion: 2.15-SNAPSHOT
className: CamelScrExample
Y: :
```

完成！

现在运行：

```
mvn install
```

该捆绑包已准备好部署。

295.4. 单元测试 CAMEL 路由

Service 组件是一个 POJO，对（非OSGi）单元测试没有特殊要求。然而，有一些特定于 Camel SCR 的技术，或者只是简化测试。

以下是一个单元测试示例，由 camel-archetype-scr 生成：

```
// This file was generated from org.apache.camel.archetypes/camel-archetype-scr/2.15-SNAPSHOT
package example;

import java.util.List;

import org.apache.camel.scr.internal.ScrHelper;
import org.apache.camel.builder.AdviceWithRouteBuilder;
import org.apache.camel.component.mock.MockComponent;
import org.apache.camel.component.mock.MockEndpoint;
import org.apache.camel.model.ModelCamelContext;
import org.apache.camel.model.RouteDefinition;
import org.junit.After;
import org.junit.Before;
import org.junit.Rule;
import org.junit.Test;
import org.junit.rules.TestName;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.junit.runner.RunWith;
import org.junit.runners.JUnit4;

@RunWith(JUnit4.class)
public class CamelScrExampleTest {

    Logger log = LoggerFactory.getLogger(getClass());

    @Rule
    public TestName testName = new TestName();

    CamelScrExample integration;
    ModelCamelContext context;

    @Before
    public void setUp() throws Exception {
        log.info("************************************************************");
        log.info("Test: " + testName.getMethodName());
        log.info("************************************************************");

        // Set property prefix for unit testing
        System.setProperty(CamelScrExample.PROPERTY_PREFIX, "unit");

        // Prepare the integration
        integration = new CamelScrExample();
        integration.prepare(null, ScrHelper.getScrProperties(integration.getClass().getName()));
        context = integration.getContext();

        // Disable JMX for test
```

```
        context.disableJMX();

        // Fake a component for test
        context.addComponent("amq", new MockComponent());
    }

    @After
    public void tearDown() throws Exception {
        integration.stop();
    }

    @Test
    public void testRoutes() throws Exception {
        // Adjust routes
        List<RouteDefinition> routes = context.getRouteDefinitions();

        routes.get(0).adviceWith(context, new AdviceWithRouteBuilder() {
            @Override
            public void configure() throws Exception {
                // Replace "from" endpoint with direct:start
                replaceFromWith("direct:start");
                // Mock and skip result endpoint
                mockEndpoints("log:*");
            }
        });

        MockEndpoint resultEndpoint = context.getEndpoint("mock:log:foo",
MockEndpoint.class);
        // resultEndpoint.expectedMessageCount(1); // If you want to just check the number of
messages
        resultEndpoint.expectedBodiesReceived("hello"); // If you want to check the contents

        // Start the integration
        integration.run();

        // Send the test message
        context.createProducerTemplate().sendBody("direct:start", "hello");

        resultEndpoint.assertIsSatisfied();
    }
}
```

现在，我们来看一个有趣的位。

使用属性前缀

```
        // Set property prefix for unit testing
        System.setProperty(CamelScrExample.PROPERTY_PREFIX, "unit");
```

这样，您可以通过使用 "unit." 前缀来覆盖配置的部分。例如： unit. from 覆盖单元测试。

前缀可用于处理路由可以运行的运行时环境之间的区别。通过开发、测试和生产环境移动未更改的捆绑包是典型的用例。

从注解获取测试配置

```
integration.prepare(null, ScrHelper.getScrProperties(integration.getClass().getName()));
```

在这里，我们使用与 OSGi 环境中使用的相同属性，在测试中配置服务组件。

用于测试的模拟组件

```
// Fake a component for test
context.addComponent("amq", new MockComponent());
```

测试中不可用的组件可以像这样模拟以允许启动路由。

调整测试的路由

```
// Adjust routes
List<RouteDefinition> routes = context.getRouteDefinitions();

routes.get(0).adviceWith(context, new AdviceWithRouteBuilder() {
    @Override
    public void configure() throws Exception {
        // Replace "from" endpoint with direct:start
        replaceFromWith("direct:start");
        // Mock and skip result endpoint
        mockEndpoints("log:*");
    }
});
```

Camel 的 AdviceWith 功能允许修改路由进行测试。

启动路由

```
// Start the integration
integration.run();
```

在这里，我们启动 Service 组件，并为其提供路由。

发送测试信息

```
// Send the test message
context.createProducerTemplate().sendBody("direct:start", "hello");
```

在这里，我们向测试中的路由发送一条信息。

295.5. 在 APACHE KARAF 中运行捆绑包

使用 mvn 安装 构建捆绑包后，就可以部署它。要在 Apache Karaf 上部署捆绑包，请在 Karaf 命令行上执行以下步骤：

在 Apache Karaf 中部署捆绑包

```
# Add Camel feature repository
karaf@root> features:chooseurl camel 2.15-SNAPSHOT

# Install camel-scr feature
karaf@root> features:install camel-scr

# Install commons-lang, used in the example route to validate parameters
karaf@root> osgi:install mvn:commons-lang/commons-lang/2.6

# Install and start your bundle
karaf@root> osgi:install -s mvn:example/camel-scr-example/1.0-SNAPSHOT

# See how it's running
karaf@root> log:tail -n 10

Press ctrl-c to stop watching the log.
```

295.5.1. 覆盖默认配置

默认情况下，Service 组件的配置 PID 等于其类的完全限定名称。您可以使用 Karaf 的 config:* 命令更改示例捆绑包的属性：

覆盖属性

```
# Override 'messageOk' property
karaf@root> config:propset -p example.CamelScrExample messageOk "This is better
logging"
```

或者，您可以编辑 Karaf 的 etc 文件夹中的属性文件来更改配置。

295.5.2. 使用 Camel SCR 捆绑包作为模板

假设您有一个 Camel SCR 捆绑包，它实现了常用的集成模式，例如从 → 到，成功/失败日志记录和重新发送，这也是我们的示例路由实现的模式。您可能不希望为每个实例创建单独的捆绑包。您不会涉及 SCR。

为您的服务组件创建配置 PID，但添加一个带有短划线的尾部，SCR 将使用该配置来创建一个组件的新实例。

创建新 Service 组件实例

```
# Create a PID with a tail
karaf@root> config:edit example.CamelScrExample-anotherone

# Override some properties
karaf@root> config:propset camelContextId my-other-context
karaf@root> config:propset to "file://removeme?fileName=removemetoo.txt"

# Save the PID
karaf@root> config:update
```

这将使用覆盖的属性启动新的 CamelContext。如何方便.

295.6. 注

在设计 Service 组件时，您通常不希望它在没有默认配置的情况下启动它。

要防止您的 Service 组件从默认配置添加 policy = ConfigurationPolicy.REQUIRE 'to the class level '@Component 注解开始。

第 296 章 XML 安全数据格式

从 Camel 版本 2.0 开始提供

XMLSecurity Data Format 在 Document, Element, 和 Element 内容级别（包括使用 XPath 同步多节点加密/解密）促进 XML 有效负载的加密和解密。要使用 XML 签名规格签署消息，请参阅 Camel XML 安全组件。

加密功能基于使用 Apache XML Security (Santuario)项目支持的格式。目前，使用 Triple-DES 和 AES (128、192 和 256)加密格式支持对称加密/解密。可以根据需要轻松添加其他格式。此功能允许在路由中分配或接收时 Camel 用户加密/解密有效负载。

从 Camel 2.9开始，
XMLSecurity Data Format 支持非对称密钥加密。在此加密模型中生成对称密钥，用于执行 XML 内容加密或解密。然后，使用非对称加密算法对"内容加密密钥"进行加密，该算法将接收者的公钥作为"密钥加密密钥"。使用非对称密钥加密算法可确保只有接收者的私钥的持有者才能访问生成的对称加密密钥。因此，只有私钥拥有者才可以解码消息。XMLSecurity 数据格式处理使用非对称密钥加密加密和解密消息内容和加密密钥所需的所有逻辑。

在处理选择用于加密的 XPath 查询时，XMLSecurity Data Format 还改进了对命名空间的支持。命名空间定义映射可以包含作为数据格式配置的一部分。这可启用 true 命名空间匹配，即使 XPath 查询和 target xml 文档中的前缀值不是等同的字符串。

296.1. XMLSECURITY 选项

XML 安全架构支持 13 个选项，如下所列。

Name	默认值	Java 类型	描述
xmlCipherAlgorithm	TRIPLEDES	字符串	用于加密/解密 XML 消息内容的密码算法。可用的选择包括：XMLCipher.TRIPLEDES XMLCipher.AES_128 XMLCipher.AES_128_GCM XMLCipher.AES_192 XMLCipher.AES_192_GCM XMLCipher.AES_256 XMLCipher.AES_256_GCM XMLCipher.SEED_128 XMLCipher.CAMELLIA_128 XMLCipher.CAMELLIA_192 XMLCipher.CAMELLIA_256_256_256_ARCipher.SEED_128 XMLCipher.CAMELLIA_192 XMLCipher.CAMELLIA_256_256_256_256_GCM是 MLCipher.TRIPLEDES。

Name	默认值	Java 类型	描述
passphrase		字符串	用作 passPhrase 用于加密/解密内容的字符串。必须提供 passPhrase。如果没有指定 passPhrase，则使用默认 passPhrase。passPhrase 需要与适当的加密算法一起使用。例如，使用 TRIPLEDES passPhase 只能是一个 24 个字节键
passPhraseByte		byte[]	用作 passPhrase 用于加密/解密内容的字节。必须提供 passPhrase。如果没有指定 passPhrase，则使用默认 passPhrase。passPhrase 需要与适当的加密算法一起使用。例如，使用 TRIPLEDES passPhase 只能是一个 24 个字节键
secureTag		字符串	XPath 引用所选的用于加密/解密的 XML 元素。如果没有指定标签，则整个有效负载会被加密/解密。
secureTagContents	false	布尔值	一个布尔值，用于指定 XML 元素是否被加密或者 XML Element false 的内容 = Element Level true = Element Content Level
keyCipherAlgorithm	RSA_OAEP	字符串	用于加密/解密非对称密钥的密码算法。可用的选择包括：XMLCipher.RSA_v1dot5 XMLCipher.RSA_OAEP XMLCipher.RSA_OAEP_11，默认值为 XMLCipher.RSA_OAEP
recipientKeyAlias		字符串	执行非对称密钥加密或解密时从 KeyStore 检索收件人的公钥或私钥时使用的密钥别名。
keyOrTrustStoreParametersId		字符串	代表 registry 中的 KeyStore 实例，用于创建和加载代表发件人 trustStore 或接收者的 keyStore 的 KeyStore 实例的配置选项。
keyPassword		字符串	用于从 KeyStore 检索私钥的密码。此密钥用于非对称解密。
digestAlgorithm	SHA1	字符串	与 RSA OAEP 算法一起使用的摘要算法。可用的选择有：XMLCipher.SHA1 XMLCipher.SHA256 XMLCipher.SHA512 默认值是 XMLCipher.SHA1
mgfAlgorithm	MGF1_SHA1	字符串	用于 RSA OAEP 算法的 MGF Algorithm。可用的选择包括：EncryptionConstants.MGF1_SHA1 EncryptionConstants.MGF1_SHA256 EncryptionConstants.MGF1_SHA512，默认值为 EncryptionConstants.MGF1_SHA1
addKeyValueForEncryptedKey	true	布尔值	是否在 EncryptedKey 结构中将用于加密会话密钥的公钥添加为 KeyValue。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

296.1.1. 密钥加密算法

从 Camel 2.12.0 开始，默认的 Key Cipher Algorithm 现在是 XMLCipher.RSA_OAEP，而不是 XMLCipher.RSA_v1dot5。由于各种攻击，不建议使用 XMLCipher.RSA_v1dot5。使用 RSA v1.5 作为密钥加密算法的请求将被拒绝，除非已明确配置为密钥加密算法。

296.2. MARSHAL

要加密有效负载，需要在路由上应用 marshal 处理器，后跟 secureXML () 标签。

296.3. UNMARSHAL

要解密有效负载，需要对路由应用 unmarshal 处理器，后跟 secureXML () 标签。

296.4. 例子

以下是如何在文档、元素和内容级别执行 marshalling 的几个示例。

296.4.1. 完整 Payload 加密/解密

```
from("direct:start")
    .marshal().secureXML()
    .unmarshal().secureXML()
    .to("direct:end");
```

296.4.2. 部分 Payload Content Only encryption/decryption

```
String tagXPATH = "//cheesesites/italy/cheese";
boolean secureTagContent = true;
...
from("direct:start")
    .marshal().secureXML(tagXPATH, secureTagContent)
    .unmarshal().secureXML(tagXPATH, secureTagContent)
    .to("direct:end");
```

296.4.3. partial Multi Node Payload Content Only encryption/decryption

```
String tagXPATH = "//cheesesites/*/cheese";
boolean secureTagContent = true;
...
from("direct:start")
```

```
    .marshal().secureXML(tagXPATH, secureTagContent)
    .unmarshal().secureXML(tagXPATH, secureTagContent)
    .to("direct:end");
```

296.4.4. 部分 Payload Content Only encryption/decryption with choice of passPhrase (password)

```
String tagXPATH = "//cheesesites/italy/cheese";
boolean secureTagContent = true;
...
String passPhrase = "Just another 24 Byte key";
from("direct:start")
    .marshal().secureXML(tagXPATH, secureTagContent, passPhrase)
    .unmarshal().secureXML(tagXPATH, secureTagContent, passPhrase)
    .to("direct:end");
```

296.4.5. 部分 Payload Content Only encryption/decryption with passPhrase (password)和 Algorithm

```
import org.apache.xml.security.encryption.XMLCipher;
....
String tagXPATH = "//cheesesites/italy/cheese";
boolean secureTagContent = true;
String passPhrase = "Just another 24 Byte key";
String algorithm= XMLCipher.TRIPLEDES;
from("direct:start")
    .marshal().secureXML(tagXPATH, secureTagContent, passPhrase, algorithm)
    .unmarshal().secureXML(tagXPATH, secureTagContent, passPhrase, algorithm)
    .to("direct:end");
```

296.4.6. 带有命名空间的部分 Payload 内容支持

Java DSL

```
final Map<String, String> namespaces = new HashMap<String, String>();
namespaces.put("cust", "http://cheese.xmlsecurity.camel.apache.org/");

final KeyStoreParameters tsParameters = new KeyStoreParameters();
tsParameters.setPassword("password");
tsParameters.setResource("sender.ts");

context.addRoutes(new RouteBuilder() {
    public void configure() {
        from("direct:start")
          .marshal().secureXML("//cust:cheesesites/italy", namespaces, true, "recipient",
                               testCypherAlgorithm, XMLCipher.RSA_v1dot5, tsParameters)
          .to("mock:encrypted");
    }
}
```

Spring XML

定义作为 **camelContext** 定义一部分的命名空间前缀可以在 **secureXML** 元素的数据格式 **secureTag** 属性中重复使用。

```
<camelContext id="springXmlSecurityDataFormatTestCamelContext"
              xmlns="http://camel.apache.org/schema/spring"
              xmlns:cheese="http://cheese.xmlsecurity.camel.apache.org/">
   <route>
      <from uri="direct://start"/>
         <marshal>
            <secureXML secureTag="//cheese:cheesesites/italy"
                       secureTagContents="true"/>
         </marshal>
         ...
```

296.4.7. 非对称密钥加密

Spring XML Sender

```
<!--  trust store configuration -->
<camel:keyStoreParameters id="trustStoreParams" resource="./sender.ts" password="password"/>

<camelContext id="springXmlSecurityDataFormatTestCamelContext"
              xmlns="http://camel.apache.org/schema/spring"
              xmlns:cheese="http://cheese.xmlsecurity.camel.apache.org/">
   <route>
      <from uri="direct://start"/>
         <marshal>
            <secureXML secureTag="//cheese:cheesesites/italy"
                       secureTagContents="true"
                       xmlCipherAlgorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"
                       keyCipherAlgorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
                       recipientKeyAlias="recipient"
                       keyOrTrustStoreParametersId="trustStoreParams"/>
         </marshal>
         ...
```

Spring XML Recipient

```
<!--  key store configuration -->
<camel:keyStoreParameters id="keyStoreParams" resource="./recipient.ks" password="password" />

<camelContext id="springXmlSecurityDataFormatTestCamelContext"
              xmlns="http://camel.apache.org/schema/spring"
              xmlns:cheese="http://cheese.xmlsecurity.camel.apache.org/">
   <route>
```

```
<from uri="direct://encrypted"/>
  <unmarshal>
    <secureXML secureTag="//cheese:cheesesites/italy"
          secureTagContents="true"
          xmlCipherAlgorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"
          keyCipherAlgorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"
          recipientKeyAlias="recipient"
          keyOrTrustStoreParametersId="keyStoreParams"
          keyPassword="privateKeyPassword" />
  </unmarshal>
  ...
```

296.5. 依赖项

此数据格式在 camel-xmlsecurity 组件中提供。

第 297 章 SEDA 组件

从 Camel 版本 1.1 开始提供

seda: 组件提供异步 SEDA 行为，因此消息交换在 BlockingQueue 上，并且使用者在独立于制作者的线程中调用。

请注意，队列仅在 单个 CamelContext 中可见。如果要跨 CamelContext 实例通信（例如，在 Web 应用程序间通信），请参阅 VM 组件。

在处理消息时，如果虚拟机终止了任何类型的持久性或恢复，则此组件不会实现任何类型的持久性或恢复。如果您需要持久性的可靠性或分布式 SEDA，请尝试使用 JMS 或 ActiveMQ。

TIP:*Synchronous*，当生成者发送消息交换时，直接 组件提供任何消费者的同步调用。

297.1. URI 格式

```
seda:someName[?options]
```

其中 someName 可以是在当前 CamelContext 中唯一标识端点的任何字符串。

您可以以以下格式将查询选项附加到 URI 中： ?option=value&option=value&...

297.2. 选项

SEDA 组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
queueSize (advanced)	设置 SEDA 队列的默认最大容量（例如，它可以保存的消息数）。	1000	int
concurrentConsumers (consumer)	设置默认的并发线程处理交换数。	1	int

Name	描述	默认值	类型
defaultQueueFactory (advanced)	设置默认队列工厂。		BlockingQueueFactory
defaultBlockWhenFull (producer)	将消息发送到完整 SEDA 队列的线程将阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出一个异常，表示队列已满。通过启用这个选项，调用线程将阻止并等待消息被接受。	false	布尔值
defaultOfferTimeout (producer)	将消息发送到完整 SEDA 队列的线程将阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出一个异常，表示队列已满。通过启用这个选项，可以将配置超时添加到块问题单中。使用lining java 队列的 .offer (timeout)方法		long
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

SEDA 端点使用 URI 语法进行配置：

```
seda:name
```

使用以下路径和查询参数：

297.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	**所需** 队列的名称		字符串

297.2.2. 查询参数(17 参数)：

Name	描述	默认值	类型
size (common)	SEDA 队列的最大容量（例如，它可以保存的消息数）。默认情况下，将使用 SEDA 组件上设置的 defaultSize。	1000	int

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外，该处理程序记录在 WARN/ERROR 级别并忽略。	false	布尔值
concurrentConsumers (consumer)	并发线程处理交换的数量。	1	int
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，该例外记录在 WARN/ERROR 级别并忽略。		ExceptionHandler
exchangePattern (consumer)	在创建交换时设置默认交换模式。		ExchangePattern
limitConcurrentConsumers (consumer)	是否将 concurrentConsumers 的数量限制为最多 500。默认情况下，如果端点配置了更大的数字，则会抛出异常。您可以通过关闭这个选项来禁用该检查。	true	布尔值
multipleConsumers (consumer)	指定是否允许多个消费者。如果启用，您可以使用 SEDA 进行 Publish-Subscribe 消息。也就是说，您可以向 SEDA 队列发送消息，并让每个消费者收到邮件的副本。启用后，应在每个消费者端点上指定此选项。	false	布尔值
pollTimeout (consumer)	轮询时使用的超时。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让使用者在关闭时更快地做出反应。	1000	int
purgeWhenStopping (consumer)	在停止 consumer/route 时是否清除任务队列。这允许更快地停止，因为队列中的任何待处理消息都会被丢弃。	false	布尔值
blockWhenFull (producer)	将消息发送到完整 SEDA 队列的线程将阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出一个异常，表示队列已满。通过启用这个选项，调用线程将阻止并等待消息被接受。	false	布尔值
discardIfNoConsumers (producer)	当发送到没有活跃消费者的队列时，生成者是否应该丢弃消息（不要将消息添加到队列）。可以同时启用其中一个选项 discardIfNoConsumers 和 failIfNoConsumers。	false	布尔值

Name	描述	默认值	类型
failIfNoConsumers (producer)	当发送到没有活跃用户的队列时，生成者是否应该通过抛出异常。可以同时启用其中一个选项 discardIfNoConsumers 和 failIfNoConsumers。	false	布尔值
offerTimeout (producer)	当队列已满时，可以使用 offerTimeout （毫秒）添加到块问题单中。您可以使用 0 或负值禁用超时。		long
timeout (producer)	SEDA 生成者停止等待异步任务完成前的超时（以毫秒为单位）。您可以使用 0 或负值禁用超时。	30000	long
waitForTaskToComplete (producer)	选项指定调用者是否应该等待 async 任务在继续前等待。支持以下三个选项： Always、Never 或 IfReplyExpected。前两个值为 self-explanatory。最后的值如果为 Request ReplyExpected，只有在消息是 Request Reply based 时等待。默认选项为 IfReplyExpected。	IfReplyExpected	WaitForTaskToComplete
队列 (advanced)	定义供端点使用的队列实例。这个选项只适用于您要使用自定义队列实例的个别用例。		BlockingQueue
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

297.3. 选择 BLOCKINGQUEUE 实现

从 Camel 2.12 开始提供

默认情况下，SEDA 组件始终是 intantiates LinkedBlockingQueue，但您可以使用不同的实现，您可以引用自己的 BlockingQueue 实现，在这种情况下，不使用 size 选项。

```
<bean id="arrayQueue" class="java.util.ArrayBlockingQueue">
  <constructor-arg index="0" value="10" ><!-- size -->
  <constructor-arg index="1" value="true" ><!-- fairness -->
</bean>

<!-- ... and later -->
<from>seda:array?queue=#arrayQueue</from>
```

或者您可以引用 BlockingQueueFactory 实现，3 个实现提供 LinkedBlockingQueueFactory, ArrayBlockingQueueFactory 和 PriorityBlockingQueueFactory:

```
<bean id="priorityQueueFactory"
class="org.apache.camel.component.seda.PriorityBlockingQueueFactory">
```

```
  <property name="comparator">
    <bean class="org.apache.camel.demo.MyExchangeComparator" />
  </property>
</bean>

<!-- ... and later -->
<from>seda:priority?queueFactory=#priorityQueueFactory&size=100</from>
```

297.4. 重新使用请求

SEDA 组件支持使用 Request Reply，调用者将等待 Async 路由完成。例如：

```
from("mina:tcp://0.0.0.0:9876?textline=true&sync=true").to("seda:input");

from("seda:input").to("bean:processInput").to("bean:createResponse");
```

在上面的路由中，我们在端口 9876 上有一个 TCP 侦听器，它接受传入的请求。请求路由到 seda:input 队列。当它是 Request Reply 消息时，我们会等待响应。当 seda:input 队列上的消费者完成后，它会将响应复制到原始消息响应。

注意

在 2.2: Works 之前，只有使用 Request Reply over SEDA 或 VM 的 2 个端点 只适用于 2 个端点。您无法通过 发送到 A → B → C 等来串联端点。仅在 A → B 之间。原因是实施逻辑非常简单。为了支持 3+ 端点，逻辑更为复杂，可以正确处理等待线程之间的排序和通知。这在 Camel 2.3 以后已进行了改进，它允许您像您想要一样链接多个端点。

297.5. 并发消费者

默认情况下，SEDA 端点使用单一消费者线程，但您可以将其配置为使用并发消费者线程。因此，您可以使用的线程池而不是线程池：

```
from("seda:stageName?concurrentConsumers=5").process(...)
```

与两者之间的差别相同，请注意 线程池 可以在运行时动态增加/shrink，具体取决于负载，而并发用户的数量始终被修复。

297.6. 线程池

请注意，通过执行以下操作在 SEDA 端点中添加线程池：

```
from("seda:stageName").thread(5).process(...)
```

可以利用两个 BlockQueues: one 从 SEDA 端点中取名，一个来自线程池的工作队列，可能不是您想要的。相反，您可能希望使用线程池配置 直接 端点，这样可同步和异步处理消息。例如：

```
from("direct:stageName").thread(5).process(...)
```

您还可以使用 concurrentConsumers 选项直接配置处理 SEDA 端点上消息的线程数量。

297.7. 示例

在以下路由中，我们使用 SEDA 队列将请求发送到此 async 队列，以便能够向此线程中进一步处理发送 fire-and-forget 消息，并将此线程中的恒定回复返回给原始调用者。

在这里，我们发送 Hello World 消息，并期望回复是 OK。

"Hello World"消息将从另一个线程中消耗来自 SEDA 队列，以进行进一步处理。由于这来自单元测试，因此它将发送到一个 模拟 端点，在单元测试中我们可以进行断言。

297.8. 使用 MULTIPLECONSUMERS

从 Camel 2.2 开始提供

在这个示例中，我们定义了两个消费者，并将其注册为 spring Bean。

由于在 seda foo 端点上指定了 multipleConsumers=true，因此我们可以让这两个用户接收自己的消息副本作为 pub-sub 风格的消息传递。

因为 Bean 是单元测试的一部分，它们只是将消息发送到模拟端点，但注意我们可以如何使用 @Consume 从 seda 队列中消耗。

297.9. 提取队列信息。

如果需要，可以在不需要使用 JMX 的情况下获得队列大小等信息：

```
SedaEndpoint seda = context.getEndpoint("seda:xxxx");
int size = seda.getExchanges().size();
```

297.10. 另请参阅

- VM
- Disruptor
- direct
- Async

第 298 章 JAVA OBJECT SERIALIZATION DATAFORMAT

从 Camel 版本 2.12 开始提供

serialization （序列化）是一个 Data Format，它使用标准的 Java Serialization 机制将二进制有效负载 unmarshal 到 Java 对象，或将 Java 对象嵌套到二进制 blob 中。
例如，以下命令使用 Java 序列化来 unmarshal 一个二进制文件，然后将其作为 ObjectMessage 发送到 ActiveMQ

```
from("file://foo/bar").
  unmarshal().serialization().
  to("activemq:Some.Queue");
```

298.1. 选项

Java Object Serialization dataformat 支持 1 个选项，如下所列。

Name	默认值	Java 类型	描述
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

298.2. 依赖项

此数据格式以 camel-core 提供，因此不需要额外的依赖项。

第 299 章 服务组件

从 Camel 版本 2.22 开始提供

299.1. 使用服务端点

299.2. URI 格式

```
service:serviceName:endpoint[?options]
```

299.3. 选项

Service 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
服务 （高级）	注入要使用的服务。		ServiceRegistry
serviceSelector (advanced)	注入用于查找 ServiceRegistry 要使用的服务选择器。		选择器
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Service 端点使用 URI 语法进行配置：

```
service:delegateUri
```

使用以下路径和查询参数：

299.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
delegateUri	**必需的** 端点 uri 作为服务公开		字符串

299.3.2. 查询参数(4 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

299.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.service.enabled	是否启用服务组件的自动配置。这默认是启用的。		布尔值
camel.component.service.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.service.service	注入要使用的服务。选项是 org.apache.camel.cloud.ServiceRegistry 类型。		字符串
camel.component.service.service-selector	注入用于查找 ServiceRegistry 要使用的服务选择器。选项是 org.apache.camel.cloud.ServiceRegistry.Selector 类型。		字符串

299.5. 实现

Camel 提供以下 ServiceRegistry 实现：

- *camel-consul*
- *camel-zookeeper*
- *camel-spring-cloud*

299.6. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 300 章 SERVICENOW 组件

从 Camel 版本 2.18 开始提供

ServiceNow 组件通过其 REST API 访问 ServiceNow 平台。

注意

从 Camel 2.18.1 开始，组件支持多个 ServiceNow 平台版本，默认值为 Helsinki。支持的版本为 表 300.1 “API 映射” 和 表 300.2 “API 映射”

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-servicenow</artifactId>
  <version>${camel-version}</version>
</dependency>
```

300.1. URI 格式

```
servicenow://instanceName?[options]
```

300.2. 选项

ServiceNow 组件支持 14 个选项，如下所列。

Name	描述	默认值	类型
instanceName (advanced)	ServiceNow 实例名称		字符串
配置 （高级）	ServiceNow 默认配置		ServiceNowConfiguration
api url (producer)	ServiceNow REST API url		字符串
用户名 (security)	ServiceNow 用户帐户名称		字符串

Name	描述	默认值	类型
密码 (security)	ServiceNow 帐户密码		字符串
oauthClientId (security)	OAuth2 ClientID		字符串
oauthClientSecret (security)	OAuth2 ClientSecret		字符串
oauthTokenUrl (security)	OAuth 令牌 Url		字符串
proxyHost (advanced)	代理主机名		字符串
proxyPort (advanced)	代理端口号		整数
proxyUserName (security)	用于代理身份验证的用户名		字符串
proxyPassword (security)	用于代理身份验证的密码		字符串
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

ServiceNow 端点使用 URI 语法进行配置：

```
servicenow:instanceName
```

使用以下路径和查询参数：

300.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
instanceName	**必需** ServiceNow 实例名称		字符串

300.2.2. 查询参数(44 参数)：

Name	描述	默认值	类型
display (producer)	将此参数设置为 true 以仅返回选中指示显示字段的 scorecard。将此参数设置为 all，以使用任何 Display 字段值返回 scorecard。默认情况下，此参数为 true。	true	字符串
displayValue (producer)	返回显示值(true)、实际值(false)，或两者(all)用于参考字段（默认为 false）	false	字符串
excludeReferenceLink (producer)	true 要排除参考字段的 Table API 链接（默认为 false）		布尔值
Favorites (producer)	将此参数设置为 true 以仅返回最喜欢查询用户的 scorecard。		布尔值
includeAggregates (producer)	将此参数设置为 true，以始终返回指示符的所有可用聚合，包括何时应用了聚合。如果没有指定值，此参数默认为 false，并返回任何聚合。		布尔值
includeAvailableAggregates (producer)	将此参数设为 true，以便在没有应用聚合时返回指示符的所有可用聚合。如果没有指定值，此参数默认为 false，并返回任何聚合。		布尔值
includeAvailableBreakdowns (producer)	将此参数设置为 true，以返回指示器的所有可用分类。如果没有指定值，此参数默认为 false，并返回 no breakdowns。		布尔值
includeScoreNotes (producer)	将此参数设置为 true，以返回与分数关联的所有备注。备注元素包含备注文本，以及添加备注时的作者和时间戳。		布尔值
includeScores (producer)	将此参数设置为 true，以返回 scorecard 的所有分数。如果没有指定值，此参数默认为 false，并只返回最新的分数值。		布尔值
inputDisplayValue (producer)	true 设置输入字段的原始值（默认为 false）		布尔值
key (producer)	将此参数设置为 true 以仅返回关键指示符的 scorecard。		布尔值
models (producer)	定义请求和响应模型		字符串
perPage (producer)	输入每个查询可以返回的最大 scorecard 数。默认值为 10，最大值为 100。	10	整数
release (producer)	ServiceNow 发行版本到目标，默认为 Helsinki，请参阅 https://docs.servicenow.com	HELSINKI	ServiceNowRelease

Name	描述	默认值	类型
requestModels (producer)	定义请求模型		字符串
resource (producer)	默认资源可以被标头 CamelServiceNowResource 覆盖		字符串
responseModels (producer)	定义响应模型		字符串
sortby (producer)	指定排序结果时要使用的值。默认情况下，查询按值排序记录。		字符串
sortDir (producer)	指定排序方向、升序或降序。默认情况下，查询会以降序排列记录。使用 sysparm_sortdir=asc 以升序排序。		字符串
suppressAutoSysField (producer)	true 阻止自动生成系统字段（默认值：false）		布尔值
suppressPaginationHeader (producer)	将此值设置为 true，以从响应中删除 Link 标头。当与查询匹配的记录数超过查询限制时，Link 标头允许您请求额外的数据页面		布尔值
table (producer)	默认表可以被标头 CamelServiceNowTable 覆盖		字符串
target (producer)	将此参数设置为 true 以仅返回具有目标的 scorecard。		布尔值
topLevelOnly (producer)	仅获取那些父目录为目录的类别。		布尔值
apiVersion (advanced)	ServiceNow REST API 版本，默认 latest		字符串
dateFormat (advanced)	用于 Json 序列化/deserialization 的日期格式	yyyy-MM-dd	字符串
dateTimeFormat (advanced)	用于 Json serialization/deserialization 的 date-time 格式	YYYY-MM-dd HH:mm:s	字符串
httpClientPolicy (advanced)	配置 http-client		HTTPClientPolicy

Name	描述	默认值	类型
mapper (advanced)	设置用于请求/回复的 Jackson 的 ObjectMapper		ObjectMapper
proxyAuthorizationPolicy (advanced)	配置代理身份验证		ProxyAuthorization 策略
retrieveTargetRecordOn Import (advanced)	将此参数设置为 true，在使用导入 set api 时检索目标记录。然后，导入设置结果会被目标记录替换	false	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
timeFormat (advanced)	用于 Json 序列化/deserialization 的时间格式	Hh:mm:ss	字符串
proxyHost (proxy)	代理主机名		字符串
proxyPort (proxy)	代理端口号		整数
apiUrl (security)	ServiceNow REST API url		字符串
oauthClientId (security)	OAuth2 ClientID		字符串
oauthClientSecret (security)	OAuth2 ClientSecret		字符串
oauthTokenUrl (security)	OAuth 令牌 Url		字符串
密码 (security)	**所需的** ServiceNow 帐户密码，必须提供		字符串
proxyPassword (security)	用于代理身份验证的密码		字符串
proxyUserName (security)	用于代理身份验证的用户名		字符串
sslContextParameters (security)	使用 SSLContextParameters 配置安全性。请参阅 http://camel.apache.org/camel-configuration-utilities.html		SSLContextParameters
用户名 (security)	**所需的** ServiceNow 用户帐户名称，必须提供		字符串

300.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 57 选项，如下所列。

Name	描述	默认值	类型
camel.component.servicenow.api-url	ServiceNow REST API url		字符串
camel.component.servicenow.configuration.api-url	ServiceNow REST API url		字符串
camel.component.servicenow.configuration.api-version	ServiceNow REST API 版本，默认 latest		字符串
camel.component.servicenow.configuration.date-format	用于 Json 序列化/deserialization 的日期格式	yyyy-MM-dd	字符串
camel.component.servicenow.configuration.date-time-format	用于 Json serialization/deserialization 的 date-time 格式	YYYY-MM-dd HH:mm:s	字符串
camel.component.servicenow.configuration.display	将此参数设置为 true 以仅返回选中指示显示字段的 scorecard。将此参数设置为 all，以使用任何 Display 字段值返回 scorecard。默认情况下，此参数为 true。	true	字符串
camel.component.servicenow.configuration.display-value	返回显示值(true)、实际值(false)，或两者(all)用于参考字段（默认为 false）	false	字符串
camel.component.servicenow.configuration.exclude-reference-link	true 要排除参考字段的 Table API 链接（默认为 false）		布尔值
camel.component.servicenow.configuration.favorites	将此参数设置为 true 以仅返回最喜欢查询用户的 scorecard。		布尔值

Name	描述	默认值	类型
camel.component.servicenow.configuration.http-client-policy	配置 http-client		HTTPClientPolicy
camel.component.servicenow.configuration.include-aggregates	将此参数设置为 true，以始终返回指示符的所有可用聚合，包括何时应用了聚合。如果没有指定值，此参数默认为 false，并返回任何聚合。		布尔值
camel.component.servicenow.configuration.include-available-aggregates	将此参数设为 true，以便在没有应用聚合时返回指示符的所有可用聚合。如果没有指定值，此参数默认为 false，并返回任何聚合。		布尔值
camel.component.servicenow.configuration.include-available-breakdowns	将此参数设置为 true，以返回指示器的所有可用分类。如果没有指定值，此参数默认为 false，并返回 no breakdowns。		布尔值
camel.component.servicenow.configuration.include-score-notes	将此参数设置为 true，以返回与分数关联的所有备注。备注元素包含备注文本，以及添加备注时的作者和时间戳。		布尔值
camel.component.servicenow.configuration.include-scores	将此参数设置为 true，以返回 scorecard 的所有分数。如果没有指定值，此参数默认为 false，并只返回最新的分数值。		布尔值
camel.component.servicenow.configuration.input-display-value	true 设置输入字段的原始值（默认为 false）		布尔值
camel.component.servicenow.configuration.key	将此参数设置为 true 以仅返回关键指示符的 scorecard。		布尔值
camel.component.servicenow.configuration.mapper	设置用于请求/回复的 Jackson 的 ObjectMapper		ObjectMapper

Name	描述	默认值	类型
camel.component.servicenow.configuration.models	定义请求和响应模型		Map
camel.component.servicenow.configuration.oauth-client-id	OAuth2 ClientID		字符串
camel.component.servicenow.configuration.oauth-client-secret	OAuth2 ClientSecret		字符串
camel.component.servicenow.configuration.oauth-token-url	OAuth 令牌 Url		字符串
camel.component.servicenow.configuration.password	ServiceNow 帐户密码，必须提供		字符串
camel.component.servicenow.configuration.per-page	输入每个查询可以返回的最大 scorecard 数。默认值为 10，最大值为 100。	10	整数
camel.component.servicenow.configuration.proxy-authorization-policy	配置代理身份验证		ProxyAuthorization 策略
camel.component.servicenow.configuration.proxy-host	代理主机名		字符串
camel.component.servicenow.configuration.proxy-password	用于代理身份验证的密码		字符串

Name	描述	默认值	类型
camel.component.servicenow.configuration.proxy-port	代理端口号		整数
camel.component.servicenow.configuration.proxy-user-name	用于代理身份验证的用户名		字符串
camel.component.servicenow.configuration.release	ServiceNow 发行版本到目标，默认为 Helsinki，请参阅 https://docs.servicenow.com		ServiceNowRelease
camel.component.servicenow.configuration.request-models	定义请求模型		Map
camel.component.servicenow.configuration.resource	默认资源可以被标头 CamelServiceNowResource 覆盖		字符串
camel.component.servicenow.configuration.response-models	定义响应模型		Map
camel.component.servicenow.configuration.retrieve-target-record-on-import	将此参数设置为 true，在使用导入 set api 时检索目标记录。然后，导入设置结果会被目标记录替换	false	布尔值
camel.component.servicenow.configuration.sort-by	指定排序结果时要使用的值。默认情况下，查询按值排序记录。		字符串
camel.component.servicenow.configuration.sort-dir	指定排序方向、升序或降序。默认情况下，查询会以降序排列记录。使用 sysparm_sortdir=asc 以升序排序。		字符串
camel.component.servicenow.configuration.ssl-context-parameters	使用 SSLContextParameters 配置安全性。请参阅 http://camel.apache.org/camel-configuration-utilities.html		SSLContextParameters

Name	描述	默认值	类型
camel.component.servicenow.configuration.suppress-auto-sys-field	true 阻止自动生成系统字段（默认值：false）		布尔值
camel.component.servicenow.configuration.suppress-pagination-header	将此值设置为 true，以从响应中删除 Link 标头。当与查询匹配的记录数超过查询限制时，Link 标头允许您请求额外的数据页面		布尔值
camel.component.servicenow.configuration.table	默认表可以被标头 CamelServiceNowTable 覆盖		字符串
camel.component.servicenow.configuration.target	将此参数设置为 true 以仅返回具有目标的 scorecard。		布尔值
camel.component.servicenow.configuration.time-format	用于 Json 序列化/deserialization 的时间格式	Hh:mm:ss	字符串
camel.component.servicenow.configuration.top-level-only	仅获取那些父目录为目录的类别。		布尔值
camel.component.servicenow.configuration.user-name	ServiceNow 用户帐户名称，必须提供		字符串
camel.component.servicenow.enabled	启用 servicenow 组件	true	布尔值
camel.component.servicenow.instance-name	ServiceNow 实例名称		字符串
camel.component.servicenow.oauth-client-id	OAuth2 ClientID		字符串

Name	描述	默认值	类型
camel.component.servicenow.oauth-client-secret	OAuth2 ClientSecret		字符串
camel.component.servicenow.oauth-token-url	OAuth 令牌 Url		字符串
camel.component.servicenow.password	ServiceNow 帐户密码		字符串
camel.component.servicenow.proxy-host	代理主机名		字符串
camel.component.servicenow.proxy-password	用于代理身份验证的密码		字符串
camel.component.servicenow.proxy-port	代理端口号		整数
camel.component.servicenow.proxy-user-name	用于代理身份验证的用户名		字符串
camel.component.servicenow.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.servicenow.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值
camel.component.servicenow.user-name	ServiceNow 用户帐户名称		字符串

300.4. HEADERS

Name	类型	ServiceNow API 参数	端点选项	描述
CamelServiceNowResource	字符串	-	-	用于访问的资源
CamelServiceNowAction	字符串	-	-	要执行的操作
CamelServiceNowActionSubject	-	-	字符串	应应用该操作的主题
CamelServiceNowModel	类	-	-	数据模型
CamelServiceNowRequestModel	类	-	-	请求数据模型
CamelServiceNowResponseModel	类	-	-	响应数据模型
CamelServiceNowOffsetNext	-	-	-	-
CamelServiceNowOffsetPrev	-	-	-	-
CamelServiceNowOffsetFirst	-	-	-	-

Name	类型	ServiceNow API 参数	端点选项	描述
CamelServiceNowOffsetLast	-	-	-	-
CamelServiceNowContentType	-	-	-	-
CamelServiceNowContentEncoding	-	-	-	-
CamelServiceNowContentMeta	-	-	-	-
CamelServiceNowSysId	字符串	sys_id	-	-
CamelServiceNowUserSysId	字符串	user_sysid	-	-
CamelServiceNowUserId	字符串	user_id	-	-
CamelServiceNowCartItemId	字符串	cart_item_id	-	-

Name	类型	ServiceNow API 参数	端点选项	描述
CamelServiceNowFileName	字符串	file_name	-	-
CamelServiceNowTable	字符串	table_name	-	-
CamelServiceNowTableSysId	字符串	table_sys_id	-	-
CamelServiceNowEncryptionContext	字符串	encryption_context	-	-
CamelServiceNowCategory	字符串	sysparm_category	-	-
CamelServiceNowType	字符串	sysparm_type	-	-
CamelServiceNowCatalog	字符串	sysparm_catalog	-	-
CamelServiceNowQuery	字符串	sysparm_query	-	-

Name	类型	ServiceNow API 参数	端点选项	描述
CamelServiceNowDisplayValue	字符串	sysparm_display_value	displayValue	-
CamelServiceNowInputDisplayValue	布尔值	sysparm_input_display_value	inputDisplayValue	-
CamelServiceNowExcludeReferenceLink	布尔值	sysparm_exclude_reference_link	excludeReferenceLink	-
CamelServiceNowFields	字符串	sysparm_fields	-	-
CamelServiceNowLimit	整数	sysparm_limit	-	-
CamelServiceNowText	字符串	sysparm_text	-	-
CamelServiceNowOffset	整数	sysparm_offset	-	-
CamelServiceNowView	字符串	sysparm_view	-	-

Name	类型	ServiceNow API 参数	端点选项	描述
CamelServiceNowSuppressAutoSysField	布尔值	sysparm_suppress_auto_sys_field	suppressAutoSysField	-
CamelServiceNowSuppressPaginationHeader	Boolean	sysparm_suppress_pagination_header	suppressPaginationHeader	-
CamelServiceNowMinFields	字符串	sysparm_min_fields	-	-
CamelServiceNowMaxFields	字符串	sysparm_max_fields	-	-
CamelServiceNowSumFields	字符串	sysparm_sum_fields	-	-
CamelServiceNowAvgFields	字符串	sysparm_avg_fields	-	-
CamelServiceNowCount	布尔值	sysparm_count	-	-
CamelServiceGroupBy	字符串	sysparm_group_by	-	-

Name	类型	ServiceNow API 参数	端点选项	描述
CamelServiceOrderBy	字符串	sysparm_order_by	-	-
CamelServiceHaving	字符串	sysparm_having	-	-
CamelServiceNowUUID	字符串	sysparm_uuid	-	-
CamelServiceNowBreakdown	字符串	sysparm_breakdown	-	-
CamelServiceNowIncludeScores	布尔值	sysparm_include_scores	includeScores	-
CamelServiceNowIncludeScoreNotes	布尔值	sysparm_include_score_notes	includeScoreNotes	-
CamelServiceNowIncludeAggregates	布尔值	sysparm_include_aggregates	includeAggregates	-
CamelServiceNowIncludeAvailableBreakdowns	布尔值	sysparm_include_available_breakdowns	includeAvailableBreakdowns	-

Name	类型	ServiceNow API 参数	端点选项	描述
CamelServiceNowIncludeAvailableAggregates	布尔值	sysparm_include_available_aggregates	includeAvailableAggregates	-
CamelServiceNowFavorites	布尔值	sysparm_favorites	Favorites	-
CamelServiceNowKey	布尔值	sysparm_key	key	-
CamelServiceNowTarget	布尔值	sysparm_target	target	-
CamelServiceNowDisplay	字符串	sysparm_display	显示	-
CamelServiceNowPerPage	整数	sysparm_per_page	perPage	-
CamelServiceNowSortBy	字符串	sysparm_sortby	sortBy	-
CamelServiceNowSortDir	字符串	sysparm_sortdit	sortDir	-
CamelServiceNowContains	字符串	sysparm_contains	-	-

Name	类型	ServiceNow API 参数	端点选项	描述
CamelServiceNowTags	字符串	sysparm_tags	-	-
CamelServiceNowPage	字符串	sysparm_page	-	-
CamelServiceNowElementsFilter	字符串	sysparm_elements_filter	-	-
CamelServiceNowBreakdownRelation	字符串	sysparm_breakdown_relation	-	-
CamelServiceNowDataSource	字符串	sysparm_data_source	-	-
CamelServiceNowTopLevelOnly	布尔值	sysparm_top_level_only	topLevelOnly	-
CamelServiceNowApiVersion	字符串	-	-	REST API 版本
CamelServiceNowResponseMeta	Map	-	-	与响应一起提供的 meta 数据

表 300.1. API 映射

Camel ServiceNowResource	Camel ServiceNowAction	方法	API URI
表	检索	GET	/api/now/v1/table/{table_name}/{sys_id}
	CREATE	POST	/api/now/v1/table/{table_name}
	修改	PUT	/api/now/v1/table/{table_name}/{sys_id}
	DELETE	DELETE	/api/now/v1/table/{table_name}/{sys_id}
	更新	PATCH	/api/now/v1/table/{table_name}/{sys_id}
AGGREGATE	检索	GET	/api/now/v1/stats/{table_name}
IMPORT	检索	GET	/api/now/import/{table_name}/{sys_id}
	CREATE	POST	/api/now/import/{table_name}

注意

Fuji REST API 文档

表 300.2. API 映射

Camel ServiceNowResource	Camel ServiceNowAction	Camel ServiceNowActionSubject	方法	API URI
表	检索		GET	/api/now/v1/table/{table_name}/{sys_id}
	CREATE		POST	/api/now/v1/table/{table_name}
	修改		PUT	/api/now/v1/table/{table_name}/{sys_id}
	DELETE		DELETE	/api/now/v1/table/{table_name}/{sys_id}

Camel ServiceNowResource	Camel ServiceNowAction	Camel ServiceNowActionSubject	方法	API URI
	更新		PATCH	/api/now/v1/table/{table_name}/{sys_id}
AGGREGATE	检索		GET	/api/now/v1/stats/{table_name}
IMPORT	检索		GET	/api/now/import/{table_name}/{sys_id}
	CREATE		POST	/api/now/import/{table_name}
ATTACHMENT	检索		GET	/api/now/api/now/attachment/{sys_id}
	内容		GET	/api/now/attachment/{sys_id}/file
	上传		POST	/api/now/api/now/attachment/file
	DELETE		DELETE	/api/now/attachment/{sys_id}
SCORECARDS	检索	PERFORMANCE_ANALYTICS	GET	/api/now/pa/scorecards
MISC	检索	USER_ROLE_INHERITANCE	GET	/api/global/user_role_inheritance
	CREATE	IDENTIFY_RECONCILE	POST	/api/now/identifyreconcile
SERVICE_CATALOG	检索		GET	/sn_sc/servicecatalog/catalogs/{sys_id}
	检索	类别	GET	/sn_sc/servicecatalog/catalogs/{sys_id}/categories

Camel ServiceNowResource	Camel ServiceNowAction	Camel ServiceNowActionSubject	方法	API URI
SERVICE_CATALOG_ITEMS	检索		GET	/sn_sc/servicecatalog/items/{sys_id}
	检索	SUBMIT_GUIDE	POST	/sn_sc/servicecatalog/items/{sys_id}/submit_guide
	检索	CHECKOUT_GUIDE	POST	/sn_sc/servicecatalog/items/{sys_id}/checkout_guide
	CREATE	SUBJECT_CART	POST	/sn_sc/servicecatalog/items/{sys_id}/add_to_cart
	CREATE	SUBJECT_PRODUCER	POST	/sn_sc/servicecatalog/items/{sys_id}/submit_producer
SERVICE_CATALOG_CARTS	检索		GET	/sn_sc/servicecatalog/cart
	检索	DELIVERY_ADDRESS	GET	/sn_sc/servicecatalog/cart/delivery_address/{user_id}
	检索	CHECKOUT	POST	/sn_sc/servicecatalog/cart/checkout
	更新		POST	/sn_sc/servicecatalog/cart/{cart_item_id}
	更新	CHECKOUT	POST	/sn_sc/servicecatalog/cart/submit_order
	DELETE		DELETE	/sn_sc/servicecatalog/cart/{sys_id}/empty
SERVICE_CATALOG_CATEGORIES	检索		GET	/sn_sc/servicecatalog/categories/{sys_id}

注意

Helsinki REST API 文档

300.5. 使用示例：

检索 10 事件

```
context.addRoutes(new RouteBuilder() {
    public void configure() {
        from("direct:servicenow")
            .to("servicenow:{{env:SERVICENOW_INSTANCE}}"
                + "?userName={{env:SERVICENOW_USERNAME}}"
                + "&password={{env:SERVICENOW_PASSWORD}}"
                + "&oauthClientId={{env:SERVICENOW_OAUTH2_CLIENT_ID}}"
                + "&oauthClientSecret={{env:SERVICENOW_OAUTH2_CLIENT_SECRET}}"
            .to("mock:servicenow");
    }
});

FluentProducerTemplate.on(context)
    .withHeader(ServiceNowConstants.RESOURCE, "table")
    .withHeader(ServiceNowConstants.ACTION, ServiceNowConstants.ACTION_RETRIEVE)
    .withHeader(ServiceNowConstants.SYSPARM_LIMIT.getId(), "10")
    .withHeader(ServiceNowConstants.TABLE, "incident")
    .withHeader(ServiceNowConstants.MODEL, Incident.class)
    .to("direct:servicenow")
    .send();
```

第 301 章 SERVLET 组件

从 Camel 版本 2.0 开始提供

servlet: 组件为消耗 HTTP 请求提供基于 HTTP 的端点，该端点到达与公布的 Servlet 绑定的 HTTP 端点。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-servlet</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

注意

Stream

servlet 基于流，这意味着它收到的输入作为流提交给 Camel。这意味着您只能够读取一次流的内容。如果您发现消息正文似乎为空的情况，或者您需要多次访问数据（例如：执行多播或重新发送错误处理），您应该使用流缓存或将消息正文转换为 字符串，这样可安全地读取多次。

301.1. URI 格式

```
servlet://relative_path[?options]
```

您可以以以下格式将查询选项附加到 URI 中： ?option=value&option=value&...

301.2. 选项

Servlet 组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
servletName (consumer)	要使用的默认 servlet 名称。默认名称为 CamelServlet。	Camel Servlet	字符串
httpRegistry (consumer)	使用自定义 org.apache.camel.component.servlet.HttpRegistry。		HttpRegistry
attachmentMultipart Binding (consumer)	是否自动将 multipart/form-data 作为 Camel Exchange 上的附件进行自动绑定。选项 attachmentMultipartBinding=true 和 disableStreamCache=false 无法一起工作。删除 disableStreamCache 以使用 AttachmentMultipartBinding。默认情况下关闭此设置，因为这可能需要 servlet 特定的配置，以便在使用 Servlet 时启用此功能。	false	布尔值
fileNameExtWhitelist (consumer)	接受上传的文件的已接受文件扩展名白名单。可以使用逗号分隔多个扩展，如 txt、xml。		字符串
httpBinding (advanced)	使用自定义 HttpBinding 来控制 Camel 消息和 HttpClient 之间的映射。		HttpBinding
httpConfiguration (advanced)	将共享 HttpConfiguration 用作基础配置。		HttpConfiguration
允许 JavaSerialized Object (advanced)	当请求使用 context-type=application/x-java-serialized-object 时，是否允许 java serialization。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Servlet 端点使用 URI 语法进行配置：

```
servlet:contextPath
```

使用以下路径和查询参数：

301.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
contextPath	**必需的** context-path		字符串

301.2.2. 查询参数(22 参数)：

Name	描述	默认值	类型
disableStreamCache (common)	确定 Servlet 中的原始输入流是否被缓存(Camel 将会在 memory/overflow 中读取流到文件，流缓存)缓存。默认情况下，Camel 将缓存 Servlet 输入流，以支持多次读取，以确保其 Camel 可以从流检索所有数据。但是，当您需要访问原始流时，您可以将此选项设置为 true，例如将其直接流传输到文件或其他持久性存储。DefaultHttpBinding 将请求输入流复制到流缓存中，如果此选项为 false，则将它放入消息正文，以支持多次读取流。如果您使用 Servlet 到 bridge/proxy 端点，则请考虑启用此选项来提高性能，如果您不需要多次读取消息有效负载。http/http4 producer 默认缓存响应正文流。如果将此选项设置为 true，则生成者不会缓存响应正文流，而是使用响应流作为消息正文。	false	布尔值
headerFilterStrategy (common)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
httpBinding (common)	使用自定义 HttpBinding 来控制 Camel 消息和 HttpClient 之间的映射。		HttpBinding
async (consumer)	将消费者配置为在 async 模式下工作	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
chunked (consumer)	如果此选项为 false，则 Servlet 将禁用 HTTP 流，并在响应上设置 content-length 标头	true	布尔值
httpMethodRestrict (consumer)	仅在 HttpMethod 匹配时才允许使用，如 GET/POST/PUT 等。可以使用逗号分隔多个方法。		字符串
matchOnUriPrefix (consumer)	如果找不到完全匹配，消费者是否应该尝试通过匹配 URI 前缀来查找目标消费者。	false	布尔值

Name	描述	默认值	类型
responseBufferSize (consumer)	在 javax.servlet.ServletResponse 上使用自定义缓冲区大小。		整数
servletName (consumer)	要使用的 servlet 的名称	Camel Servlet	字符串
transferException (consumer)	如果在消费者端启用并交换失败处理，如果原因 Exception 在响应中作为 application/x-java-serialized-object 内容类型发送序列化，则结果为 application/x-java-serialized-object 内容类型。在生产者侧，异常将反序列化并丢弃为原样，而不是 HttpOperationFailedException。原因异常需要按顺序处理。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
attachmentMultipartBinding (consumer)	是否自动将 multipart/form-data 作为 Camel Exchange 上的附件进行自动绑定。选项 attachmentMultipartBinding=true 和 disableStreamCache=false 无法一起工作。删除 disableStreamCache 以使用 AttachmentMultipartBinding。默认情况下关闭此设置，因为这可能需要 servlet 特定的配置，以便在使用 Servlet 时启用此功能。	false	布尔值
eagerCheckContentAvailable (consumer)	如果 content-length 标头为 0，还是 eager 检查 HTTP 请求是否有内容。如果 HTTP 客户端没有发送流数据，则可以打开此项。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
fileNameExtWhitelist (consumer)	接受上传的文件的已接受文件扩展名白名单。可以使用逗号分隔多个扩展，如 txt、xml。		字符串
optionsEnabled (consumer)	指定是否为这个 Servlet 消费者启用 HTTP OPTIONS。默认情况下关闭 OPTIONS。	false	布尔值
traceEnabled (consumer)	指定是否为这个 Servlet 使用者启用 HTTP TRACE。默认情况下关闭 TRACE。	false	布尔值

Name	描述	默认值	类型
mapHttpMessageBody (advanced)	如果此选项为 true，则交换的 IN Exchange Body 将映射到 HTTP 正文。把它设置为 false 可以避免 HTTP 映射。	true	布尔值
mapHttpMessageFormUrlEncodedBody (advanced)	如果此选项为 true，则交换的 IN Exchange Form Encoded body 将映射到 HTTP。把它设置为 false 可以避免 HTTP Form Encoded body 映射。	true	布尔值
mapHttpMessageHeaders (advanced)	如果此选项为 true，则交换的 IN Exchange Headers 将映射到 HTTP 标头。把它设置为 false 将避免 HTTP 标头映射。	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

301.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 13 个选项，如下所列。

Name	描述	默认值	类型
camel.component.servlet.allow-java-serialized-object	当请求使用 context-type=application/x-java-serialized-object 时，是否允许 java serialization。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
camel.component.servlet.attachment-multipart-binding	是否自动将 multipart/form-data 作为 Camel Exchange 上的附件进行自动绑定。选项 attachmentMultipartBinding=true 和 disableStreamCache=false 无法一起工作。删除 disableStreamCache 以使用 AttachmentMultipartBinding。默认情况下关闭此设置，因为这可能需要 servlet 特定的配置，以便在使用 Servlet 时启用此功能。	false	布尔值
camel.component.servlet.enabled	启用 servlet 组件	true	布尔值
camel.component.servlet.file-name-ext-whitelist	接受上传的文件的已接受文件扩展名白名单。可以使用逗号分隔多个扩展，如 txt、xml。		字符串

Name	描述	默认值	类型
camel.component.servlet.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		字符串
camel.component.servlet.http-binding	使用自定义 HttpBinding 来控制 Camel 消息和 HttpClient 之间的映射。选项是 org.apache.camel.http.common.HttpBinding 类型。		字符串
camel.component.servlet.http-configuration	将共享 HttpConfiguration 用作基础配置。选项是 org.apache.camel.http.common.HttpConfiguration 类型。		字符串
camel.component.servlet.http-registry	使用自定义 org.apache.camel.component.servlet.HttpRegistry。选项是 org.apache.camel.component.servlet.HttpRegistry 类型。		字符串
camel.component.servlet.mapping.context-path	servlet 组件用于自动映射的上下文路径。	/camel/*	字符串
camel.component.servlet.mapping.enabled	启用 servlet 组件的自动映射到 Spring web 上下文。	true	布尔值
camel.component.servlet.mapping.servlet-name	Camel servlet 的名称。	CamelServlet	字符串
camel.component.servlet.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.servlet.servlet-name	要使用的默认 servlet 名称。默认名称为 CamelServlet。	CamelServlet	字符串

301.4. 消息标头

Camel 将应用与 HTTP 组件相同的 Message Headers。

Camel 还将填充 所有 request.parameter 和 request.headers。例如，如果客户端请求具有 URL http://myserver/myserver?orderid=123，则该交换将包含名为 orderid 的标头，其值为 123。

301.5. 使用方法

您只能从 Servlet 组件生成的端点消耗。因此，它应该只用作 Camel 路由中的输入。要针对其他 HTTP 端点发出 HTTP 请求，请使用 HTTP 组件。

301.6. 将 CAMEL JAR 放入应用服务器引导类路径

如果您将 Camel JARs （如 camel-core、camel-servlet 等）放在应用服务器的引导类路径中（通常位于其 lib 目录中），则请记住 servlet 映射列表现在在应用服务器中的多个部署的 Camel 应用程序之间共享。

请注意，将 Camel JAR 置于应用服务器的引导类路径中通常不是最佳的做法！

因此，在这些情况下，您必须在每个 Camel 应用程序中定义自定义且唯一的 servlet 名称，例如 web.xml 定义：

```
<servlet>
  <servlet-name>MyServlet</servlet-name>
  <servlet-class>org.apache.camel.component.servlet.CamelHttpTransportServlet</servlet-
class>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>MyServlet</servlet-name>
  <url-pattern>/*</url-pattern>
</servlet-mapping>
```

在 Camel 端点中，也包括 servlet 名称

```
<route>
  <from uri="servlet://foo?servletName=MyServlet"/>
  ...
</route>
```

从 Camel 2.11 开始，Camel 将检测这个重复项，且无法启动应用程序。您可以通过将 servlet init-parameter ignoreDuplicateServletName 设置为 true 来控制忽略这个重复，如下所示：

```
<servlet>
  <servlet-name>CamelServlet</servlet-name>
  <display-name>Camel Http Transport Servlet</display-name>
  <servlet-class>org.apache.camel.component.servlet.CamelHttpTransportServlet</servlet-
class>
  <init-param>
    <param-name>ignoreDuplicateServletName</param-name>
    <param-value>true</param-value>
  </init-param>
</servlet>
```

但 强烈建议 为每个 Camel 应用程序使用唯一的 servlet-name，以避免这种重复冲突，以及任何不可预见的副作用。

301.7. 示例

注意

从 Camel 2.7 开始，在 Spring web 应用程序中使用 Servlet 更容易。详情请查看 Servlet Tomcat 示例。

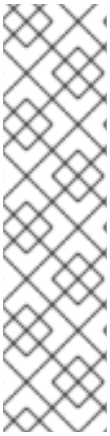
在这个示例中，我们定义在 http://localhost:8080/camel/services/hello 中公开 HTTP 服务的路由。

首先，您需要通过普通的 Web 容器或 OSGi 服务发布 CamelHttpTransportServlet。使用 Web.xml 文件发布 CamelHttpTransportServlet，如下所示：

```
<web-app>

  <servlet>
    <servlet-name>CamelServlet</servlet-name>
    <display-name>Camel Http Transport Servlet</display-name>
    <servlet-class>org.apache.camel.component.servlet.CamelHttpTransportServlet</servlet-
class>
  </servlet>

  <servlet-mapping>
    <servlet-name>CamelServlet</servlet-name>
    <url-pattern>/services/*</url-pattern>
  </servlet-mapping>

</web-app>
```

然后，您可以按照以下方式定义路由：

```
from("servlet:hello?matchOnUriPrefix=true").process(new Processor() {
    public void process(Exchange exchange) throws Exception {
        String contentType = exchange.getIn().getHeader(Exchange.CONTENT_TYPE,
String.class);
        String path = exchange.getIn().getHeader(Exchange.HTTP_URI, String.class);
        path = path.substring(path.lastIndexOf("/"));

        assertEquals("Get a wrong content type", CONTENT_TYPE, contentType);
        // assert camel http header
        String charsetEncoding =
exchange.getIn().getHeader(Exchange.HTTP_CHARACTER_ENCODING, String.class);
        assertEquals("Get a wrong charset name from the message heaer", "UTF-8",
charsetEncoding);
        // assert exchange charset
        assertEquals("Get a wrong charset naem from the exchange property", "UTF-8",
exchange.getProperty(Exchange.CHARSET_NAME));
        exchange.getOut().setHeader(Exchange.CONTENT_TYPE, contentType + "; charset=UTF-
8");
        exchange.getOut().setHeader("PATH", path);
        exchange.getOut().setBody("<b>Hello World</b>");
    }
});
```



注意

指定 camel-servlet 端点的相对路径

由于我们将 HTTP 传输与公布的 servlet 绑定，因此我们不知道 servlet 的应用上下文路径，camel-servlet 端点使用相对路径来指定端点的 URL。客户端可以通过 servlet 发布地址访问 camel-servlet 端点：("http://localhost:8080/camel/services")+ RELATIVE_PATH ("/hello")

301.7.1. 使用 Spring 3.x 时的示例

请参阅 Servlet Tomcat 示例。

301.7.2. 使用 Spring 2.x 时的示例

当在 Camel/Spring 应用程序中使用 Servlet 组件时，通常需要在 Servlet 组件启动后加载 Spring ApplicationContext。这可以通过使用 Spring 的 ContextLoaderServlet 而不是 ContextLoaderListener 来完成。在这种情况下，您需要在 CamelHttpTransportServlet 后启动 ContextLoaderServlet，如下所示：

```
<web-app>
  <servlet>
    <servlet-name>CamelServlet</servlet-name>
```

```
        <servlet-class>
            org.apache.camel.component.servlet.CamelHttpTransportServlet
        </servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet>
        <servlet-name>SpringApplicationContext</servlet-name>
        <servlet-class>
            org.springframework.web.context.ContextLoaderServlet
        </servlet-class>
        <load-on-startup>2</load-on-startup>
    </servlet>
</web-app>
```

301.7.3. 使用 OSGi 时的示例

在 Camel 2.6.0 中，您可以使用类似如下的蓝图发布 CamelHttpTransportServlet 作为 OSGi 服务：

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="
           http://www.osgi.org/xmlns/blueprint/v1.0.0
https://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd">

    <bean id="camelServlet" class="org.apache.camel.component.servlet.CamelHttpTransportServlet"
/>

    <!--
        Enlist it in OSGi service registry.
        This will cause two things:
        1) As the pax web whiteboard extender is running the CamelServlet will
           be registered with the OSGi HTTP Service
        2) It will trigger the HttpRegistry in other bundles so the servlet is
           made known there too
    -->
    <service ref="camelServlet">
        <interfaces>
            <value>javax.servlet.Servlet</value>
            <value>org.apache.camel.http.common.CamelServlet</value>
        </interfaces>
        <service-properties>
            <entry key="alias" value="/camel/services" />
            <entry key="matchOnUriPrefix" value="true" />
            <entry key="servlet-name" value="CamelServlet" />
        </service-properties>
    </service>

</blueprint>
```

然后，在您的 Camel 路由中使用这个服务，如下所示：

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
           xmlns:ext="http://aries.apache.org/blueprint/xmlns/blueprint-ext/v1.0.0"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="
           http://www.osgi.org/xmlns/blueprint/v1.0.0
https://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd">

    <reference id="servletref" ext:proxy-method="classes"
interface="org.apache.camel.http.common.CamelServlet">
        <reference-listener ref="httpRegistry" bind-method="register" unbind-method="unregister" />
    </reference>

    <bean id="httpRegistry" class="org.apache.camel.component.servlet.DefaultHttpRegistry" />

    <bean id="servlet" class="org.apache.camel.component.servlet.ServletComponent">
        <property name="httpRegistry" ref="httpRegistry" />
    </bean>

    <bean id="servletProcessor" class="org.apache.camel.example.servlet.ServletProcessor" />

    <camelContext xmlns="http://camel.apache.org/schema/blueprint">
        <route>
            <!-- Notice how we can use the servlet scheme which is that reference above -->
            <from uri="servlet://hello" />
            <process ref="servletProcessor" />
        </route>
    </camelContext>

</blueprint>
```

对于 Camel 2.6 之前的版本，您可以使用 Activator 在 OSGi 平台上发布 CamelHttpTransportServlet ：

```
import java.util.Dictionary;
import java.util.Hashtable;

import org.apache.camel.component.servlet.CamelHttpTransportServlet;
import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;
import org.osgi.framework.ServiceReference;
import org.osgi.service.http.HttpContext;
import org.osgi.service.http.HttpService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.osgi.context.BundleContextAware;

public final class ServletActivator implements BundleActivator, BundleContextAware {
    private static final Logger LOG = LoggerFactory.getLogger(ServletActivator.class);
    private static boolean registerService;

    /**
     * HttpService reference.
     */
```

```
    private ServiceReference<?> httpServiceRef;

    /**
     * Called when the OSGi framework starts our bundle
     */
    public void start(BundleContext bc) throws Exception {
        registerServlet(bc);
    }

    /**
     * Called when the OSGi framework stops our bundle
     */
    public void stop(BundleContext bc) throws Exception {
        if (httpServiceRef != null) {
            bc.ungetService(httpServiceRef);
            httpServiceRef = null;
        }
    }

    protected void registerServlet(BundleContext bundleContext) throws Exception {
        httpServiceRef = bundleContext.getServiceReference(HttpService.class.getName());

        if (httpServiceRef != null && !registerService) {
            LOG.info("Register the servlet service");
            final HttpService httpService = (HttpService)bundleContext.getService(httpServiceRef);
            if (httpService != null) {
                // create a default context to share between registrations
                final HttpContext httpContext = httpService.createDefaultHttpContext();
                // register the hello world servlet
                final Dictionary<String, String> initParams = new Hashtable<String, String>();
                initParams.put("matchOnUriPrefix", "false");
                initParams.put("servlet-name", "CamelServlet");
                httpService.registerServlet("/camel/services", // alias
                    new CamelHttpTransportServlet(), // register servlet
                    initParams, // init params
                    httpContext // http context
                );
                registerService = true;
            }
        }
    }

    public void setBundleContext(BundleContext bc) {
        try {
            registerServlet(bc);
        } catch (Exception e) {
            LOG.error("Cannot register the servlet, the reason is " + e);
        }
    }

}
```

301.7.4. 使用 Spring-Boot

从 Camel 2.19.0 开始，camel-servlet-starter 库会在 /camel PROFILE 上下文路径下自动绑定所有剩余的端点。下表总结了 camel-servlet-starter 库中提供的其他配置属性。也可以禁用 Camel servlet 的自动映射。

spring-Boot Property	default	描述
camel.component.servlet.mapping.enabled	**true**	启用 servlet 组件的自动映射到 Spring web 上下文
camel.component.servlet.mapping.context-path	**/camel/***	servlet 组件用于自动映射的上下文路径
camel.component.servlet.mapping.servlet-name	**CamelServlet**	Camel servlet 的名称

301.8. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*
- *Servlet Tomcat 示例*
- *servlet Tomcat No Spring Example*
- *HTTP*
- *Jetty*

301.9. SERVLETLISTENER 组件

从 Camel 2.11 开始提供

此组件用于在 web 应用程序中引导 Camel 应用程序。例如，事先人员必须找到自己的引导 Camel 方法，或者依赖 Spring 等第三方框架进行。

注意

侧边栏 此组件支持 Servlet 2.x，这意味着它在较旧的 Web 容器中也可以正常工作，这是此组件的目标。虽然 Servlet 2.x 需要使用 web.xml 文件作为配置。对于 Servlet 3.x 容器，您可以使用注解驱动的配置来使用 @WebListener 来增强 Camel，并实施您自己的类，您可以在其中 boostrap Camel。这样做仍然会面临一些挑战，使最终用户能够轻松配置 Camel，而您可通过旧的 ium web.xml 文件自由配置 Camel。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-servletlistener</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

301.9.1. 使用

您需要选择抽象类 org.apache.camel.component.servletlistener.CamelServletContextListener 的以下实现之一。

- JndiCamelServletContextListener，它使用 JndiRegistry 将 JNDI 用于其 registry。
- SimpleCamelServletContextListener，它使用 SimpleRegistry 将 java.util.Map 用作其 registry。

要使用它，您需要在 WEB-INF/web.xml 文件中配置 org.apache.camel.component.servletlistener.CamelServletContextListener，如下所示：

301.9.2. 选项

org.apache.camel.component.servletlistener.CamelServletContextListener 支持以下选项，这些选项可以在 web.xml 文件中配置为 context-param。

选项	类型	描述
propertyPlaceholder.XXX		在 Camel 中配置属性占位符。您应该为选项添加 "propertyPlaceholder." 前缀，例如要配置位置，使用 propertyPlaceholder.location 作为名称。您可以配置 Properties 组件中的所有选项。
jmx.XXX		配置 JMX，请执行以下操作：您应该为选项添加 "jmx." 前缀，例如禁用 JMX，使用 jmx.disabled 作为名称。您可以配置 **org.apache.camel.spi.ManagementAgent** 中的所有选项。以及 JMX 页面中提到的选项。
name	**字符串**	配置 CamelContext 的名称。
messageHistory	**布尔值**	**Camel 2.12.2：** 是否启用或禁用消息历史记录（默认为启用）。
streamCache	**布尔值**	是否启用流缓存。
trace	**布尔值**	是否启用 Tracer。
delayer	**Long**	要为 Delay Interceptor 设置延迟值。
handleFault	**布尔值**	是否启用句柄错误。
errorHandlerRef	**字符串**	指的是要使用的上下文范围 Error Handler。
autoStartup	**布尔值**	启动 Camel 时是否启动所有路由。
useMDCLogging	**布尔值**	是否使用 MDC 日志记录。
useBreadcrumb	**布尔值**	是否使用面包屑导航栏。

选项	类型	描述
management NamePattern	**字符串**	要为 JMX MBeans 设置自定义命名模式。
threadNamePattern	**字符串**	要为线程设置自定义命名模式，请执行以下操作：
properties.XXX		要在 **CamelContext.getProperties** 上设置自定义属性：这是 seldom。
routebuilder.XXX		要配置要使用的路由，请执行以下操作：详情请查看以下。
CamelContextLifecycle		指的是 **org.apache.camel.component.servletlistener.CamelContextLifecycle** 实施的 FQN 类名称。允许在 CamelContext 启动或停止之前和之后执行自定义代码。详情请查看以下信息。
XXX		要在 CamelContext 上设置任何选项，请执行以下操作：

301.9.3. 例子

请参阅 Servlet Tomcat No Spring Example。

301.9.4. 访问创建的 CamelContext

从 Camel 2.14/2.13.3/2.12.5 开始提供

创建的 CamelContext 作为带有键为 "CamelContext" 的属性存储在 ServletContext 上。如果您可以保存 ServletContext，则可以保存 CamelContext，如下所示：

```
ServletContext sc = ...
CamelContext camel = (CamelContext) sc.getAttribute("CamelContext");
```

301.9.5. 配置路由

您需要配置在 web.xml 文件中使用哪些路由。您可以通过多种方式完成此操作，但所有参数都必须以 "routeBuilder" 前缀。

301.9.5.1. 使用 RouteBuilder 类

默认情况下，Camel 将假定 param-value 是 Camel RouteBuilder 类的 FQN 类名称，如下所示：

```
<context-param>
  <param-name>routeBuilder-MyRoute</param-name>
  <param-value>org.apache.camel.component.servletlistener.MyRoute</param-value>
</context-param>
```

您可以在同一 param-value 中指定多个类，如下所示：

```
<context-param>
  <param-name>routeBuilder-routes</param-name>
  <!-- we can define multiple values separated by comma -->
  <param-value>
    org.apache.camel.component.servletlistener.MyRoute,
    org.apache.camel.component.servletlistener.routes.BarRouteBuilder
  </param-value>
</context-param>
```

参数的名称在运行时没有意义。它只需要是唯一的，并以"routeBuilder"开始。在上例中，我们有"routeBuilder-routes"。但是，您也可以将其命名为 "routeBuilder.foo"。

301.9.5.2. 使用软件包扫描

您还可以告诉 Camel 使用软件包扫描，这意味着它将在给定软件包中查找所有类型的 RouteBuilder 类型，并自动将它们添加为 Camel 路由。要使用 "packagescan:" 为值添加前缀，如下所示：

```
<context-param>
  <param-name>routeBuilder-MyRoute</param-name>
  <!-- define the routes using package scanning by prefixing with packagescan: -->
  <param-value>packagescan:org.apache.camel.component.servletlistener.routes</param-
value>
  </context-param>
```

301.9.5.3. 使用 XML 文件

您还可以使用 XML DSL 定义 Camel 路由，但我们没有使用 Spring 或 Blueprint，XML 文件只能包含 Camel 路由。

在 web.xml 中，您引用 XML 文件，该文件可以是 "classpath", "file" 或 "http" url，如下所示：

```
<context-param>
  <param-name>routeBuilder-MyRoute</param-name>
  <param-value>classpath:routes/myRoutes.xml</param-value>
</context-param>
```

XML 文件为：

routes/myRoutes.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- the xmlns="http://camel.apache.org/schema/spring" is needed -->
<routes xmlns="http://camel.apache.org/schema/spring">

  <route id="foo">
    <from uri="direct:foo"/>
    <to uri="mock:foo"/>
  </route>

  <route id="bar">
    <from uri="direct:bar"/>
    <to uri="mock:bar"/>
  </route>

</routes>
```

请注意，在 XML 文件中，root 标签为 <routes>，它必须使用命名空间 "http://camel.apache.org/schema/spring"。此命名空间的名称中包含 spring，但由于历史原因，因为 Spring 是第一个且唯一的 XML DSL。运行时不需要 Spring JAR。在 Camel 3.0 中，命名空间可以重命名为通用名称。

301.9.5.4. 配置正确的占位符

以下是 web.xml 配置的一个片段，用于将属性占位符设置为从 classpath 加载 myproperties.properties

```
<!-- setup property placeholder to load properties from classpath -->
<!-- we do this by setting the param-name with propertyPlaceholder. as prefix and then any options such as location, cache etc -->
<context-param>
  <param-name>propertyPlaceholder.location</param-name>
  <param-value>classpath:myproperties.properties</param-value>
</context-param>
<!-- for example to disable cache on properties component, you do -->
```

```
<context-param>
  <param-name>propertyPlaceholder.cache</param-name>
  <param-value>false</param-value>
</context-param>
```

301.9.5.5. 配置 JMX

以下是用于配置 JMX 的 web.xml 配置片段，如禁用 JMX。

```
<!-- configure JMX by using names that is prefixed with jmx. -->
<!-- in this example we disable JMX -->
<context-param>
  <param-name>jmx.disabled</param-name>
  <param-value>true</param-value>
</context-param>
```

JNDI 或 Simple as Camel Registry ^^ ^^^^^^^^^^^^^^^^^^^^

此组件使用 JNDI 或 Simple 作为 Registry。
这样，您可以在 JNDI 中查找 Bean和其他服务，并绑定和取消绑定您自己的 Bean。

这可以通过实施 org.apache.camel.component.servletlistener.CamelContextLifecycle 来从 Java 代码完成。

301.9.5.6. 使用自定义 CamelContextLifecycle

在以下代码中，我们使用 Start 和 afterStop 的回调，将自定义 bean 列在 Simple Registry 中，并在我们停止时进行清理。

然后，我们需要使用参数名称"CamelContextLifecycle"在 web.xml 文件中注册此类，如下所示。该值必须是 FQN，它指的是实施 org.apache.camel.component.servletlistener.CamelContextLifecycle 接口的类。

```
<context-param>
  <param-name>CamelContextLifecycle</param-name>
  <param-value>org.apache.camel.component.servletlistener.MyLifecycle</param-value>
</context-param>
```

我们使用名称"my Bean "列出 HelloBean Bean，我们可以在 Camel 路由中引用此 Bean，如下所示：

```
public class MyBeanRoute extends RouteBuilder {
    @Override
    public void configure() throws Exception {
        from("seda:foo").routeId("foo")
            .to("bean:myBean")
            .to("mock:foo");
    }
}
```

重要：如果您使用 org.apache.camel.component.servletlistener.JndiCamelServletContextListener，则 CamelContextLifecycle 还必须使用 JndiRegistry。如果 servlet 是 org.apache.camel.component.servletlistener.SimpleCamelServletContextListener，则 CamelContextLifecycle 必须使用 SimpleRegistry

301.9.6. 另请参阅

- SERVLET
- Servlet Tomcat 示例
- servlet Tomcat No Spring Example

第 302 章 SFTP 组件

从 Camel 版本 1.1 开始提供

这个组件可让您通过 FTP 和 SFTP 协议访问远程文件系统。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-ftp</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

如需更多信息，您可以查看 FTP 组件

302.1. URI 选项

以下选项专用于 FTPS 组件。

SFTP 组件没有选项。

SFTP 端点使用 URI 语法进行配置：

```
sftp:host:port/directoryName
```

使用以下路径和查询参数：

302.1.1. 路径参数(3 参数)：

Name	描述	默认值	类型
主机	需要的 FTP 服务器的主机名		字符串

Name	描述	默认值	类型
port	FTP 服务器的端口		int
directoryName	起始目录		字符串

302.1.2. 查询参数(117 参数)：

Name	描述	默认值	类型
charset (common)	此选项用于指定文件的编码。您可以在消费者上使用此选项，指定文件的编码，允许 Camel 了解在访问文件内容时应加载文件内容。在编写文件时，您可以使用此选项指定写入该文件的 charset。请记住，在编写文件 Camel 时，可能需要将消息内容读取到内存中，以便能将数据转换为配置的 charset，因此如果您有大型消息，请不要使用它。		字符串
disconnect (common)	使用后是否从远程 FTP 服务器断开连接。断开连接将仅断开当前与 FTP 服务器的连接。如果您有要停止的消费者，则需要停止消费者/路由。	false	布尔值
doneFileName (common)	生产者：如果提供，则 Camel 将在写入原始文件时写入第 2 个 done 文件。完成的文件将为空。这个选项配置要使用的文件名。您可以指定一个固定名称。或者，您可以使用动态占位符。完成的文件始终将写入与原始文件相同的文件夹中。consumer：如果提供，Camel 仅在文件存在时使用文件。这个选项配置要使用的文件名。您可以指定一个固定名称。或者您可以使用动态占位符。已完成的文件始终预期位于与原始文件相同的文件夹中。仅支持 $file.name 和 $file.name.noext 作为动态占位符。		字符串
filename (common)	使用 File Language 等表达式来动态设置文件名。对于消费者，它将用作文件名过滤器。对于生成者，它用于评估要写入的文件名。如果设置了表达式，它将优先于 CamelFileName 标头。（注：标题本身也可以是表达式）。表达式选项支持 String 和 Expression 类型。如果表达式是 String 类型，则始终使用 File 语言来评估它。如果表达式是 Expression 类型，则使用指定的 Expression 类型 - 例如，允许您使用 OGNL 表达式。对于消费者，您可以使用它过滤文件名，因此您可以使用 File Language 语法使用当前的文件：mydata-$date:now:yyyyMMdd.txt。生产者支持 CamelOverruleFileName 标头，它优先于任何现有的 CamelFileName 标头；CamelOverruleFileName 是一个只使用一次的标头，它可让您更轻松地临时存储 CamelFileName，且之后必须恢复它。		字符串

Name	描述	默认值	类型
jschLoggingLevel (common)	JSCH 活动日志记录使用的日志记录级别。因为 JSCH 默认在 INFO 级别中详细，阈值默认为 WARN。	WARN	LoggingLevel
分隔符 (common)	设置要使用的路径分隔符。Unix = 使用 unix 样式路径分隔符 Windows = 使用窗口样式路径分隔符 Auto =（默认）在文件名中使用现有路径分隔符	UNIX	PathSeparator
fastExistsCheck (common)	如果将此选项设置为 true，camel-ftp 将直接使用列表文件来检查该文件是否存在。由于某些 FTP 服务器可能不支持直接列出文件，如果选项为 false，camel-ftp 将使用旧方式列出该目录并检查该文件是否存在。这个选项还影响 readLock=changed，以控制它是否执行快速检查来更新文件信息。如果 FTP 服务器包含大量文件，则可以使用它来加快进程。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
delete (consumer)	如果为 true，则在成功处理后会删除该文件。	false	布尔值
moveFailed (consumer)	根据简单语言设置移动失败表达式。例如，要将文件移到 .error 子目录中，请使用： .error。注意：将文件移到故障位置 Camel 时，将处理错误，并且不会再次获取该文件。		字符串
noop (consumer)	如果为 true，则文件不会以任何方式移动或删除。这个选项对只读数据或者 ETL 类型要求很好。如果 noop=true，Camel 也设置 idempotent=true，以避免再次消耗相同的文件。	false	布尔值
preMove (consumer)	表达式（如文件语言）用于在处理之前动态设置文件名。例如，要将 in-progress 文件移到 order 目录中，将此值设置为 order。		字符串
preSort (consumer)	启用预先排序后，消费者将在轮询期间对文件和目录名称进行排序，该名称是从文件系统检索的。如果需要按排序顺序对文件进行操作，您可能需要执行此操作。指示器在消费者开始过滤前执行，并接受 Camel 处理的文件。这个选项是 default=false 表示禁用。	false	布尔值
递归 (consumer)	如果某个目录，将查找所有子目录中的文件。	false	布尔值

Name	描述	默认值	类型
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
streamDownload (consumer)	设置在使用本地工作目录时要使用的下载方法。如果设置为 true，则远程文件会在读取时流到路由。当设置为 false 时，远程文件会在发送到路由之前加载到内存中。	false	布尔值
directoryMustExist (consumer)	与 startingDirectoryMustExist 类似，但这在轮询递归子目录期间应用。	false	布尔值
download (consumer)	FTP 使用者是否应下载 文件。如果此选项设为 false，则消息正文将为空，但消费者仍会触发具有文件详情的 Camel Exchange，如文件名、文件大小等。只是不会下载该文件。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
ignoreFileNotFoundOrPermissionError (consumer)	是否忽略 when （尝试列出目录中的文件或下载文件时），但不存在，还是因为权限错误。默认情况下，当目录或文件不存在或权限不足时，会抛出异常。将这个选项设置为 true 可忽略它。	false	布尔值
inProgressRepository (consumer)	可插拔式 in-progress 存储库 org.apache.camel.spi.IdempotentRepository。in-progress 存储库用于考虑被消耗的当前正在进行的文件。默认使用基于内存的存储库。		IdempotentRepository
localWorkDirectory (consumer)	在消耗时，可以使用本地工作目录直接将远程文件内容存储在本地文件中，以避免将内容加载到内存中。这很有用，如果您使用非常大的远程文件，因此可以节省内存。		字符串
onCompletionException Handler (consumer)	要使用自定义 org.apache.camel.spi.ExceptionHandler 来处理在完成过程中发生的任何抛出异常，消费者在其中执行提交或回滚。默认实施会将任何异常记录在 WARN 级别并忽略。		ExceptionHandler
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy

Name	描述	默认值	类型
processStrategy (consumer)	可插拔 org.apache.camel.component.file.GenericFileProcessStrategy 允许您实施自己的 readLock 选项或类似选项。也可以在使用文件之前满足特殊条件时使用，如存在特殊的就绪文件。如果设置了这个选项，则不会应用 readLock 选项。		GenericFileProcess s Strategy
startingDirectoryMustExist (consumer)	起始目录是否必须存在。请注意，autoCreate 选项是默认启用的，这意味着如果起始目录不存在，则通常会自动创建该目录。您可以禁用 autoCreate 并启用它，以确保起始目录必须存在。如果目录 不存在，将抛出异常。	false	布尔值
useList (consumer)	下载文件时是否允许使用 LIST 命令。默认为 true。在某些情况下，您可能想要下载特定的文件，且不允许使用 LIST 命令，因此您可以将此选项设置为 false。请注意，在使用此选项时，要下载的特定文件不包括诸如文件大小、时间戳、权限等元数据数据信息，因为这些信息只能在 LIST 命令使用时检索。	true	布尔值
fileExist (producer)	如果文件已存在具有相同名称的文件，该怎么办。override （默认值）替换现有文件。Append - 在现有文件中添加内容。fail - 抛出 GenericFileOperationException，表示已存在文件。ignore - 静默忽略问题，且不会覆盖现有文件，但假设一切都正常。Move - 选项需要使用 moveExisting 选项也可以配置。选项 eagerDeleteTargetFile 可用于控制在移动文件时执行什么操作，并且已存在文件，否则会导致移动操作失败。Move 选项将在编写目标文件之前移动任何现有文件。只有在使用 tempFileName 选项时，TryRename 才适用。这允许尝试将文件从临时名称重命名为实际名称，而无需执行任何存在的检查。在某些文件系统（特别是 FTP 服务器中）中，这个检查可能会更快。	override	GenericFileExist
flatten (producer)	flatten 用于扁平化任何前导路径的文件名路径，因此只用于文件名。这可让您以递归方式消耗到子目录中，但当您将文件写入另一个目录时，它们将写入单个目录。在生成者中将其设置为 true 强制执行 CamelFileName 标头中的任何文件名都会被剥离任何前导路径。	false	布尔值
jailStartingDirectory (producer)	用于存放（限制）将文件写入起始目录（和子目录）。这默认是启用的，不允许 Camel 将文件写入外部目录（从而更加安全）。您可以关闭此项，以允许将文件写入起始目录之外的目录，如父目录或 root 文件夹。	true	布尔值

Name	描述	默认值	类型
moveExisting (producer)	当配置了 fileExist=Move 时，用于计算文件名的表达式（如 File Language）。要将文件移动到备份子目录中，请输入 backup。这个选项只支持以下 File Language 令牌：file:name, file:name.ext, file:name.noext, file:onlyname, file:onlyname.noext, file:ext, 和 file:parent。请注意，FTP 组件不支持 file:parent，因为 FTP 组件只能将任何现有文件移到基于当前 dir 的相对目录中。		字符串
tempFileName (producer)	与 tempPrefix 选项相同，但对临时文件名的命名提供更精细的控制，因为它使用 File Language。		字符串
tempPrefix (producer)	此选项用于使用临时名称写入文件，然后在写入完成后将其重命名为实际名称。可用于识别正在写入的文件，并避免消费者（不使用专用读取锁定）读取进度文件。上传大型文件时，FTP 通常使用。		字符串
allowNullBody (producer)	用于指定在文件写入过程中是否允许 null 正文。如果设置为 true，则会创建一个空文件，当设为 false 时，并尝试向文件组件发送 null 正文，则 'Cannot null body to file.' 的 GenericFileWriteException 将会被抛出。如果将 fileExist 选项设置为 'Override'，则文件将被截断，如果设置为附加文件，则文件将保持不变。	false	布尔值
chmod (producer)	允许您在存储的文件上设置 chmod。例如 chmod=640。		字符串
disconnectOnBatchComplete (producer)	在 Batch upload 完成后是否断开与远程 FTP 服务器的连接。disconnectOnBatchComplete 只会断开当前与 FTP 服务器的连接。	false	布尔值
eagerDeleteTargetFile (producer)	是否强制删除任何现有目标文件。这个选项只适用于使用 fileExists=Override 和 tempFileName 选项。您可以使用它来禁用（将其设置为 false）在写入临时文件前删除目标文件。例如，您可以写入大文件，并且希望写入 temp 文件期间存在目标文件。这样可确保目标文件仅在最后一次删除之前，直到 temp 文件被重命名为目标文件名之前。这个选项还用于控制在启用了 fileExist=Move 时是否删除任何现有文件，并且存在现有的文件。如果此选项 copyAndDeleteOnRenameFails false，则会在移动操作前现有文件存在时抛出异常。	true	布尔值
keepLastModified (producer)	将保留源文件的最后修改的时间戳（若有）。将使用 Exchange.FILE_LAST_MODIFIED 标头来定位时间戳。此标头可以包含带有时间戳的 java.util.Date 或 long。如果存在时间戳，并且启用了 选项，它将在写入的文件上设置此时间戳。注：此选项仅适用于文件制作者。您不能将此选项与任何 ftp 生成者一起使用。	false	布尔值

Name	描述	默认值	类型
moveExistingFileStrategy (producer)	策略(Custom Strategy)用于在配置 fileExist=Move 时使用，使用特殊命名令牌来移动文件。默认情况下，如果没有提供自定义策略，则使用实施		FileMoveExisting 策略
sendNoop (producer)	在将文件上传到 FTP 服务器之前，是否将 noop 命令作为预写检查发送。这默认是启用的，因为连接的验证仍然有效，这允许静默重新连接才能上传该文件。但是，如果这会导致问题，您可以关闭这个选项。	true	布尔值
auto create (advanced)	在文件的路径名称中自动创建缺少的目录。对于文件消费者，这意味着创建起始目录。对于生成者文件，这意味着文件应当要写入的目录。	true	布尔值
bindAddress (advanced)	指定连接应绑定的本地接口地址。		字符串
bufferSize (advanced)	写入缓冲区大小（以字节为单位）。	131072	int
bulkRequests (advanced)	指定在任何时间点上可以处理多少个请求。增加这个值可能会稍提高文件传输速度，但会增加内存用量。		整数
压缩 （高级）	使用 压缩方式。指定从 1 到 10 的级别。重要： 您必须手动将所需的 JSCH zlib JAR 添加到用于压缩支持的类路径中。		int
connectTimeout (advanced)	设置连接超时，以等待 FTPClient 和 JSCH 使用连接	10000	int
maximumReconnectAttempts (advanced)	指定在尝试连接到远程 FTP 服务器时执行的最大重新连接尝试 Camel。使用 0 来禁用此行为。		int
代理 (advanced)	使用自定义配置的 com.jcraft.jsch.Proxy。此代理用于使用来自目标 SFTP 主机的信息。		Proxy
reconnectDelay (advanced)	millis Camel 中的延迟将在执行重新连接尝试前等待。		long
serverAliveCountMax (advanced)	允许您设置 sftp 会话的 serverAliveCountMax	1	int
serverAliveInterval (advanced)	允许您设置 sftp 会话的 serverAliveInterval		int
soTimeout (advanced)	仅为 FTPClient 设置 so 超时	300000	int

Name	描述	默认值	类型
stepwise (advanced)	设置我们是否应在下载文件时遍历文件结构的同时，还是将文件上传到目录时逐步更改目录。例如，如果您因为安全原因无法在 FTP 服务器上更改目录时禁用此功能。	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
throwExceptionOnConnect Failed (advanced)	如果连接失败（忽略）时，应该抛出异常，默认异常不会被抛出，并记录 WARN。您可以使用它来启用异常，并处理 org.apache.camel.spi.PollingConsumerPollStrategy rollback 方法所引发的异常。	false	布尔值
antExclude (filter)	Ant 风格过滤器排除。如果使用 antInclude 和 antExclude，则 antExclude 优先于 antInclude。可以使用逗号分隔的格式指定多个排除项。		字符串
antFilterCaseSensitive (filter)	在t 过滤器上设置问题单敏感标志	true	布尔值
antInclude (filter)	Ant 风格过滤器包括：可以使用逗号分隔的格式指定多个包含。		字符串
eagerMaxMessagesPerPoll (filter)	允许控制 maxMessagesPerPoll 的限制是 eager 还是 not。如果 eager，则限制是在扫描文件时。其中 false 会扫描所有文件，然后执行排序。将此选项设置为 false 可首先对所有文件进行排序，然后限制轮询。请注意，这需要更高的内存用量，因为所有文件详细信息都在内存中执行排序。	true	布尔值
exclude (filter)	用于排除文件，如果文件名与 regex 模式匹配（匹配是 in-senstive）。请注意，如果您使用加号等符号，如果将其配置为端点 uri，则需要使用 RAW （）语法进行配置。有关配置端点 uris 时，请参阅更多详情		字符串
Filter (filter)	可插拔过滤器为 org.apache.camel.component.file.GenericFileFilter 类。如果过滤在 accept （）方法中返回 false，则会跳过文件。		GenericFileFilter
filterDirectory (filter)	根据简单语言过滤目录。例如，要过滤当前日期，您可以使用简单的日期模式，如 $date:now:yyMMdd		字符串
filterFile (filter)	根据简单语言过滤文件。例如，要过滤文件大小，您可以使用 $file:size 5000		字符串

Name	描述	默认值	类型
idempotent (filter)	使用 Idempotent Consumer EIP 模式的选项，让 Camel 跳过已处理的文件。默认情况下，将使用基于内存的 LRUCache，其中包含 1000 个条目。如果 noop=true 然后会启用幂等，以避免再次消耗同样的文件。	false	布尔值
idempotentKey (filter)	使用自定义幂等密钥。默认情况下使用文件的绝对路径。您可以使用 File Language，例如使用文件名和文件大小，您可以执行以下操作：idempotentKey=$file:name-$file:size		字符串
idempotentRepository (filter)	一个可插拔式存储库 org.apache.camel.spi.IdempotentRepository，如果未指定 none，则默认使用 MemoryMessageIdRepository，并且幂等性为 true。		IdempotentRepository
include (filter)	用于包含文件，如果文件名匹配 regex 模式（匹配不区分大小写）。请注意，如果您使用加号等符号，如果将其配置为端点 uri，则需要使用 RAW （）语法进行配置。有关配置端点 uris 时，请参阅更多详情		字符串
maxDepth (filter)	递归处理目录时要遍历的最大深度。	2147483647	int
maxMessagesPerPoll (filter)	定义每个轮询要收集的最大消息。默认不设置最大值。可用于设置限制，例如 1000，以避免在启动有数千个文件的服务器时避免。设置一个 0 或 negative 值来禁用它。注意：如果使用这个选项，则 File 和 FTP 组件将在任何排序之前限制。例如，如果您有 100000 文件并使用 maxMessagesPerPoll=500，则只有第一个 500 文件会被获取，然后排序。您可以使用 eagerMaxMessagesPerPoll 选项，并将其设置为 false 以允许先扫描所有文件，然后随后排序。		int
minDepth (filter)	在递归处理目录时开始处理的最小深度。使用 minDepth=1 表示基础目录。使用 minDepth=2 表示第一个子目录。		int
Move (filter)	表达式（如简单语言）用于在处理后动态设置文件名。要将文件移动到 .done 子目录中，只需输入 .done。		字符串
exclusiveReadLockStrategy (lock)	可插拔 read-lock 作为 org.apache.camel.component.file.GenericFileExclusiveReadLockStrategy 实施。		GenericFileExclusive ReadLockStrategy

Name	描述	默认值	类型
readLock (lock)	供消费者使用，只有在文件上具有独占的 read-lock 时才轮询文件（例如，该文件没有正在进行中或被写入）。Camel 将等到授予文件锁定为止。这个选项提供了 build in strategy: none - No read lock is in use markerFile - Camel 创建一个标记文件 (fileName.camelLock)，然后保存锁定。这个选项不适用于 FTP 组件更改 - 更改方法是使用文件长度/修改时间戳来检测当前是否复制该文件。至少将使用 1 sec 确定此选项，因此此选项无法像其他进程一样快速使用文件，但可以更可靠，因为 JDK IO API 始终无法确定文件目前是否被其他进程使用。可以使用 readLockCheckInterval 选项来设置检查频率。 fileLock - 用于使用 java.nio.channels.FileLock。这个选项在 Windows 或 FTP 组件上不可用。当通过 mount/share 访问远程文件系统时，应该避免使用这种方法，除非文件系统支持分布式文件锁定。rename 是使用尝试重命名该文件的测试，如果我们可以获得专用只读锁。幂等的 - （仅限文件组件）幂等性是将 idempotentRepository 用作读锁。这允许使用支持集群读取锁定（如果幂等存储库实现支持集群）。idid-changed - （仅限文件组件）幂等更改，使用 idempotentRepository 并更改作为组合的 read-lock。这允许使用读取锁定，如果幂等存储库实施支持集群。jid-rename - （仅限文件组件） idempotent-rename 使用 idempotentRepository，并重命名作为组合的 read-lock。这允许使用支持集群的读取锁定（如果幂等存储库实现支持）。注意：各种读取锁定并不适用于在集群模式下工作，其中不同节点上的并发用户与共享文件系统上的相同文件竞争。markerFile 使用接近 atomic 操作来创建空的标记文件，但不保证在集群中工作。fileLock 可能会更好地工作，但文件系统需要支持分布式文件锁定，等等。如果幂等性存储库支持集群，如 Hazelcast 组件或 Infinispan，则使用幂等读取锁定支持集群。	none	字符串
readLockCheckInterval (lock)	读锁在 millis 中，如果读锁定支持，则间隔为 millis。这个间隔用于在尝试获取读锁定之间休眠。例如，在使用更改的读锁定时，您可以将更高的间隔周期设置为 cater 以慢速写入。如果生成者非常慢，则默认 1 sec. 可能太快。注意：对于 FTP，默认的 readLockCheckInterval 为 5000。readLockTimeout 值必须高于 readLockCheckInterval，但 thumb 规则是有一个超过 readLockCheckInterval 的 2 次或多次的超时。这是为了确保在达到超时前尝试获取锁进程允许随时读取的时间。	1000	long

Name	描述	默认值	类型
readLockDeleteOrphanLock Files (lock)	如果 Camel 没有正确关闭（如 JVM 崩溃），则应在启动时读取带有标记文件的读取锁定文件（可能已留在文件系统上）读取锁定。如果将此选项设置为 false，则任何孤立锁定文件将导致 Camel 不尝试选择该文件，这可能是因为另一个节点同时从同一个共享目录读取文件。	true	布尔值
readLockIdempotentRelease Async (lock)	延迟发布任务应该是同步还是异步的。请参阅 readLockIdempotentReleaseDelay 选项的详情。	false	布尔值
readLockIdempotentRelease AsyncPoolSize (lock)	使用异步发行任务时，调度线程池中的线程数量。在几乎所有用例中，使用默认值 1 个内核线程应足够足够，只有在更新幂等存储库缓慢或处理大量文件时才将其设置为更高的值。如果您使用共享线程池，通过配置 readLockIdempotentReleaseExecutorService 选项，这个选项不会被使用。请参阅 readLockIdempotentReleaseDelay 选项的详情。		int
readLockIdempotentRelease Delay (lock)	是否将发行任务延迟为 millis。当文件被视为具有共享幂等存储库的主动/主动集群场景时，这可用于延迟发行任务扩展窗口，以确保其他节点因为竞争条件而可能扫描和获取同一文件。通过扩展发行任务的 time-window 有助于防止这些情况。只有在将 readLockRemoveOnCommit 配置为 true 时才需要注意延迟。		int
readLockIdempotentRelease ExecutorService (lock)	使用自定义和共享线程池进行异步发行任务。请参阅 readLockIdempotentReleaseDelay 选项的详情。		ScheduledExecutor Service
readLockLogging Level (lock)	无法获取读取锁定时使用的日志记录级别。默认情况下，会记录 WARN。您可以更改此级别，例如将 OFF 更改为没有任何日志记录。这个选项只适用于 readLock 类型： changed, fileLock, idempotent, idempotent-changed, idempotent-rename, rename。	DEBUG	LoggingLevel
readLockMarkerFile (lock)	是否将标志文件与 changed、rename 或 exclusive read 锁定类型一起使用。默认情况下，还使用标记文件来保护获取相同文件的其他进程。通过将这个选项设置为 false 来关闭此行为。例如，如果您不希望将标记文件写入 Camel 应用程序的文件系统。	true	布尔值

Name	描述	默认值	类型
readLockMinAge (lock)	这个选项只适用于 readLock=changed。它可指定文件在尝试获取读取锁定前必须经过的最短期限。例如，使用 readLockMinAge=300s 来要求文件最新 5 分钟。这可加快更改的读锁定速度，因为它将只尝试获取至少给定年龄的文件。	0	long
readLockMinLength (lock)	这个选项只适用于 readLock=changed。它允许您配置最小文件长度。默认情况下，Camel 预期文件包含数据，因此默认值为 1。您可以将这个选项设置为零，以允许消耗零长度文件。	1	long
readLockRemoveOnCommit (lock)	这个选项只适用于 readLock=idempotent。它允许在处理文件成功并且执行提交时指定是否从幂等存储库中删除文件名条目。默认情况下，文件不会被删除，以确保不会发生任何竞争条件，因此另一个活动节点可能会尝试获取该文件。相反，幂等存储库可能支持驱除策略，在 X 分钟后用于驱除文件名条目 - 这样可保证竞争条件没有问题。请参阅 readLockIdempotentReleaseDelay 选项的详情。	false	布尔值
readLockRemoveOnRollback (lock)	这个选项只适用于 readLock=idempotent。它允许在处理文件失败时指定是否从幂等存储库中删除文件名条目，并发生回滚。如果此选项为 false，则文件名称条目将被确认（就像文件确实是提交一样）。	true	布尔值
readLockTimeout (lock)	read-lock 的 millis 中可选超时（如果 read-lock 支持）。如果无法授予 read-lock，并且触发超时，则 Camel 将跳过该文件。在下一次轮询 Camel 时，将再次尝试文件，此时可能会授予 read-lock。使用 0 或较低值来指示永久。目前，fileLock，改变并重命名支持超时。注意：对于 FTP，默认的 readLockTimeout 值为 20000 而不是 10000。readLockTimeout 值必须高于 readLockCheckInterval，但 thumb 规则是有一个超过 readLockCheckInterval 的 2 次或多次的超时。这是为了确保在达到超时前尝试获取锁进程允许随时读取的时间。	10000	long
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int

Name	描述	默认值	类型
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
Shuff le (sort)	影响文件列表（以随机顺序排序）	false	布尔值
排序(排序)	使用文件语言进行内置排序.支持嵌套排序，以便按文件名排序，并根据修改的日期作为第二组排序。		字符串
排序 （排序）	可插拔排序器作为 java.util.Comparator 类。		比较器
密码 (security)	设置一个以逗号分隔的密码列表，按首选顺序使用。可能的密码名称由 JCraft JSCH 定义。某些示例包括：aes128-ctr,aes128-cbc,3des-ctr,3des-cbc,blowfish-cbc,aes192-cbc,aes256-cbc.如果没有指定 JSCH 的默认列表，则将使用。		字符串

Name	描述	默认值	类型
密钥对 （安全）	设置公钥和私钥的密钥对，以便 SFTP 端点可以进行公钥/私钥验证。		KeyPair
knownHosts (security)	从字节数组设置 known_hosts，以便 SFTP 端点可以执行主机密钥验证。		byte[]
knownHostsFile (security)	设置 known_hosts 文件，以便 SFTP 端点可以验证主机密钥。		字符串
knownHostsUri (security)	设置 known_hosts 文件（默认为从 classpath 加载），以便 SFTP 端点可以执行主机密钥验证。		字符串
密码 (security)	用于登录的密码		字符串
preferredAuthentications (security)	设置将使用 SFTP 端点的首选身份验证。一些示例包括：password,publickey.如果没有指定 JSCH 的默认列表，则将使用。		字符串
PrivateKey (security)	将私钥设置为字节，以便 SFTP 端点可以进行私钥验证。		byte[]
privateKeyFile (security)	设置私钥文件，以便 SFTP 端点可以验证私钥。		字符串
privateKeyPassphrase (security)	设置私钥文件密码短语，以便 SFTP 端点可以进行私钥验证。		字符串
privateKeyUri (security)	设置私钥文件（默认为从 classpath 加载），以便 SFTP 端点可以进行私钥验证。		字符串
StrictHostKeyChecking (security)	设置是否使用严格的主机密钥检查。	否	字符串
用户名 (security)	用于登录的用户名		字符串
useUserKnownHostsFile (security)	如果没有明确配置 knownHostFile，则使用 System.getProperty (user.home)/.ssh/known_hosts 中的主机文件	true	布尔值

第 303 章 SHIRO SECURITY 组件

从 Camel 2.5 开始提供

Camel 中的 shiro-security 组件是一个基于 Apache Shiro 安全项目的安全集中组件。

Apache Shiro 是一个强大而灵活的开源安全框架，可完全处理身份验证、授权、企业会话管理和加密。Apache Shiro 项目的目标是提供最强大、最全面的应用程序安全框架，同时易于理解和使用。

此 camel shiro-security 组件允许身份验证和授权支持应用到 camel 路由的不同片段。

使用 Camel 策略在路由上应用 Shiro 安全性。Camel 中的策略利用策略模式在 Camel 处理器上应用拦截器。它提供对 camel 路由 的部分/网段应用跨快捷方式问题（例如，安全、事务等）的能力。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-shiro</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

303.1. SHIRO 安全基础

要在 camel 路由中使用 Shiro 安全性，ShiroSecurityPolicy 对象必须使用安全配置详情（包括用户、密码、角色等）实例化。然后，此对象必须应用到 camel 路由。此 ShiroSecurityPolicy 对象也可以在 Camel registry (JNDI 或 ApplicationContextRegistry)中注册，然后用于 Camel 上下文中的其他路由。

配置详情由 ShiroSecurityPolicy 使用 Ini 文件(properties 文件)或 Ini 对象提供给 ShiroSecurityPolicy。Ini 文件是一个标准 Shiro 配置文件，其中包含 user/role 详情，如下所示

```
[users]
# user 'ringo' with password 'starr' and the 'sec-level1' role
ringo = starr, sec-level1
george = harrison, sec-level2
john = lennon, sec-level3
```

```
paul = mccartney, sec-level3

[roles]
# 'sec-level3' role has all permissions, indicated by the
# wildcard '*'
sec-level3 = *

# The 'sec-level2' role can do anything with access of permission
# readonly (*) to help
sec-level2 = zone1:*

# The 'sec-level1' role can do anything with access of permission
# readonly
sec-level1 = zone1:readonly:*
```

303.2. 实例化 SHIROSECURITYPOLICY 对象

ShiroSecurityPolicy 对象实例化，如下所示

```
    private final String iniResourcePath = "classpath:shiro.ini";
    private final byte[] passPhrase = {
        (byte) 0x08, (byte) 0x09, (byte) 0x0A, (byte) 0x0B,
        (byte) 0x0C, (byte) 0x0D, (byte) 0x0E, (byte) 0x0F,
        (byte) 0x10, (byte) 0x11, (byte) 0x12, (byte) 0x13,
        (byte) 0x14, (byte) 0x15, (byte) 0x16, (byte) 0x17};
    List<permission> permissionsList = new ArrayList<permission>();
    Permission permission = new WildcardPermission("zone1:readwrite:*");
    permissionsList.add(permission);

    final ShiroSecurityPolicy securityPolicy =
        new ShiroSecurityPolicy(iniResourcePath, passPhrase, true, permissionsList);
```

303.3. SHIROSECURITYPOLICY 选项

Name	默认值	类型	描述
iniResourcePath 或 ini	**none**	资源字符串或 Ini Object	必须将 IniResourcePath 或 Ini 对象的实例的强制资源 String 传递给安全策略。分别以 "file:, classpath:, 或 url:" 前缀时，可以从文件系统、classpath 或 URL 获取资源。例如 "classpath:shiro.ini"
passphrase	**基于 AES 128 的密钥**	byte[]	与消息交换一起发送的解密 ShiroSecurityToken (s)的 passPhrase

Name	默认值	类型	描述
alwaysReauthenticate	**true**	布尔值	设置 以确保对每一个请求重新进行身份验证。如果设置为 false，则用户进行身份验证并锁定，而不仅限于来自同一用户转发的请求。
permissionsList	**none**	List<Permission>	要授权经过身份验证的用户执行进一步操作（例如，继续在路由上进一步执行）所需的权限列表。如果没有为 ShiroSecurityPolicy 对象提供 Permissions 列表或角色列表（请参阅以下），则授权被视为不是必需的。请注意，如果列表中的任何 Permission Objects 适用，则默认是授权。
rolesList	**none**	List<String>	**Camel 2.13：** 一个所需的角色列表，以便经过身份验证的用户被授权执行进一步操作，例如继续对路由进行进一步的操作。如果没有为 ShiroSecurityPolicy 对象提供角色列表或权限列表（请参阅上面的），则授权被视为不是必需的。请注意，如果列表中任何角色都适用，则默认是授权。
cipherService	**AES**	org.apache.shiro.crypto.CipherService	Shiro 附带 AES 和基于 Blowfish 的 CipherServices。您可以使用其中之一或传递您自己的 Cipher 实现
base64	**false**	**布尔值**	**Camel 2.12：** 要将 base64 编码用于安全令牌标头，它允许通过 JMS 等传输标头。这个选项还必须在 **ShiroSecurityTokenInjector** 上设置。
allPermissionsRequired	**false**	**布尔值**	**Camel 2.13：** 如果 permissionsList 参数中的任何 Permission Objects 适用，则默认是授权。把它设置为 true 以要求满足所有权限。
allRolesRequired	**false**	**布尔值**	**Camel 2.13：** 如果 rolesList 参数中的任何角色都适用，则默认是授权。把它设置为 true 以要求满足所有角色。

303.4. 在 CAMEL ROUTE 上应用 SHIRO 身份验证

ShiroSecurityPolicy、测试并允许传入的消息交换在 Message Header 中包含加密的 SecurityToken，以继续以下正确身份验证。SecurityToken 对象包含一个 Username/Password 详

情，用于确定用户是有效用户的位置。

```
protected RouteBuilder createRouteBuilder() throws Exception {
    final ShiroSecurityPolicy securityPolicy =
        new ShiroSecurityPolicy("classpath:shiro.ini", passPhrase);

    return new RouteBuilder() {
        public void configure() {
            onException(UnknownAccountException.class).
                to("mock:authenticationException");
            onException(IncorrectCredentialsException.class).
                to("mock:authenticationException");
            onException(LockedAccountException.class).
                to("mock:authenticationException");
            onException(AuthenticationException.class).
                to("mock:authenticationException");

            from("direct:secureEndpoint").
                to("log:incoming payload").
                policy(securityPolicy).
                to("mock:success");
        }
    };
}
```

303.5. 在 CAMEL ROUTE 上应用 SHIRO 授权

授权可以通过将权限列表与 ShiroSecurityPolicy 关联来应用到 camel 路由。Permissions 列表指定用户执行路由段所需的权限。如果用户没有正确的权限集，则请求不会被授权进一步继续。

```
protected RouteBuilder createRouteBuilder() throws Exception {
    final ShiroSecurityPolicy securityPolicy =
        new ShiroSecurityPolicy("./src/test/resources/securityconfig.ini", passPhrase);

    return new RouteBuilder() {
        public void configure() {
            onException(UnknownAccountException.class).
                to("mock:authenticationException");
            onException(IncorrectCredentialsException.class).
                to("mock:authenticationException");
            onException(LockedAccountException.class).
                to("mock:authenticationException");
            onException(AuthenticationException.class).
                to("mock:authenticationException");

            from("direct:secureEndpoint").
                to("log:incoming payload").
                policy(securityPolicy).
                to("mock:success");
```

```
        }
    };
}
```

303.6. 创建 SHIROSECURITYTOKEN 并将其注入消息交换

ShiroSecurityToken 对象可以使用名为 ShiroSecurityTokenInjector 的 Shiro Processor 创建并注入到 Message Exchange 中。例如，在客户端中使用 ShiroSecurityTokenInjector 注入 ShiroSecurityTokenInjector 的示例如下所示

```
ShiroSecurityToken shiroSecurityToken = new ShiroSecurityToken("ringo", "starr");
ShiroSecurityTokenInjector shiroSecurityTokenInjector =
    new ShiroSecurityTokenInjector(shiroSecurityToken, passPhrase);

from("direct:client").
    process(shiroSecurityTokenInjector).
    to("direct:secureEndpoint");
```

303.7. 将消息发送到 SHIROSECURITYPOLICY 保护的路由

消息和消息交换随 camel 路由一起发送，安全策略需要由 Exchange Header 中的 SecurityToken 提供。SecurityToken 是一个加密对象，其中包含用户名和密码。SecurityToken 默认使用 AES 128 位安全加密，并可改为您选择的任何密码。

下面是如何使用 Camel 中的 ProducerTemplate 以及 SecurityToken 来发送请求的示例

```
@Test
public void testSuccessfulShiroAuthenticationWithNoAuthorization() throws Exception {
    //Incorrect password
    ShiroSecurityToken shiroSecurityToken = new ShiroSecurityToken("ringo", "stirr");

    // TestShiroSecurityTokenInjector extends ShiroSecurityTokenInjector
    TestShiroSecurityTokenInjector shiroSecurityTokenInjector =
        new TestShiroSecurityTokenInjector(shiroSecurityToken, passPhrase);

    successEndpoint.expectedMessageCount(1);
    failureEndpoint.expectedMessageCount(0);

    template.send("direct:secureEndpoint", shiroSecurityTokenInjector);

    successEndpoint.assertIsSatisfied();
    failureEndpoint.assertIsSatisfied();
}
```

303.8. 将消息发送到 SHIROSECURITYPOLICY 保护的路由（以后通过 CAMEL 2.12 更轻松地发送）

从 Camel 2.12 开始，您可以以两种不同的方式提供主题。

303.8.1. 使用 ShiroSecurityToken

您可以向 Camel 路由发送一条信息，其中包含用户名和密码的类型为 org.apache.camel.component.shiro.ShiroSecurityToken 的标头 ShiroSecurityConstants.SHIRO_SECURITY_TOKEN。例如：

```
ShiroSecurityToken shiroSecurityToken = new ShiroSecurityToken("ringo", "starr");

template.sendBodyAndHeader("direct:secureEndpoint", "Beatle Mania",
ShiroSecurityConstants.SHIRO_SECURITY_TOKEN, shiroSecurityToken);
```

您还可以使用两个不同的标头提供用户名和密码，如下所示：

```
Map<String, Object> headers = new HashMap<String, Object>();
headers.put(ShiroSecurityConstants.SHIRO_SECURITY_USERNAME, "ringo");
headers.put(ShiroSecurityConstants.SHIRO_SECURITY_PASSWORD, "starr");
template.sendBodyAndHeaders("direct:secureEndpoint", "Beatle Mania", headers);
```

当您使用用户名和密码标头时，Camel 路由中的 ShiroSecurityPolicy 会使用令牌自动将其转换为带有密钥 ShiroSecurityConstants.SHIRO_SECURITY_TOKEN 的单个标头。然后，token 可以是 ShiroSecurityToken 实例，也可以是一个 String （当您设置了 base64=true 时）的 base64 表示。

第 304 章 简单语言

从 Camel 版本 1.1 开始提供

Simple Expression Language 是创建时非常简单的语言，但自此已发展为更加强大的语言。它主要用于评估 Expressions 和 Predicates，而无需任何新的依赖项或 XPath 知识；因此，最好在 camel-core 中进行测试。当您需要在 Camel 路由中需要一些基于表达式的基于表达式的脚本时，可以满足 95% 的常见用例。

但是，对于更复杂的用例，您通常建议您选择更加表达和强大的语言，例如：

- SpEL
- MVEL
- Groovy
- JavaScript
- OGNL
- 支持的 脚本语言之一

简单语言对复杂表达式使用 ${body} 占位符，其中表达式包含常量文字。如果表达式只是令牌本身，则可以省略 $\{ } 占位符。

提示

稍后，从 Camel 2.5 的替代语法 也可以使用替代语法，其使用 $simple{ } 作为占位符。这可用于避免在将 Spring 属性占位符与 Camel 搭配使用时发生冲突。

304.1. CAMEL 2.9 中的简单语言更改

从 Camel 2.9 开始，简单 语言已被改进，以使用更好的语法解析器，该解析器可以进行索引精确的错误消息，以便您准确了解错误以及问题的位置。例如，如果您在其中一个操作器中创建了拼写错误，那么解析程序将无法检测到这一点，并导致评估为 true。语法中有一些更改，不再向后兼容。当使用 简单 语言作为 predicates 时，字面文本 必须 用单引号或双引号括起来。例如： "${body} == 'Camel'"。注意我们如何通过单引号显示字面上的单引号。使用 "body" 和 "header.foo" 的旧样式来指代消息正文，标头是 @deprecated，我们建议始终对内置功能使用 $\{ } 令牌。range 运算符现在要求范围以单引号括起，显示： "${header.zip} between '30000..39999'"。

要获得邮件的正文："body" 或 " in.body" 或 "${body}"。

复杂的表达式必须使用 $\{ } 占位符，例如 ："Hello ${in.header.name} how are?".

您可以在同一表达式中有多个功能：" Hello ${in.header.name} this is ${in.header.me} talking". 但是，在 Camel 2.8.x 或更早的版本中，您无法嵌套功能（不允许在现有中使用另一个 $\{ } 占位符）。从 Camel 2.9 开始，您可以嵌套功能。

304.2. 简单语言选项

Simple 语言支持 2 个选项，如下所列。

Name	默认值	Java 类型	描述
resultType		**字符串**	设置结果类型的类名称（输出中的类型）
trim	**true**	**布尔值**	是否修剪值以移除前导和结尾的空格和换行符

304.3. 变量

变量	类型	描述
camelId	字符串	**Camel 2.10:** CamelContext 名称
camelContext.**OGNL**	对象	**Camel 2.11：** 使用 Camel OGNL 表达式调用的 CamelContext。
交换	Exchange	**Camel 2.16：** 交换

变量	类型	描述
Exchange.**OGNL**	对象	**Camel 2.16：** 使用 Camel OGNL 表达式调用的 Exchange。
exchangeId	字符串	**Camel 2.3：** 交换 ID
id	字符串	输入消息 id
正文（body）	对象	输入正文
in.body	对象	输入正文
正文。**OGNL**	对象	**Camel 2.3：** 使用 Camel OGNL 表达式调用的输入正文。
in.body.**OGNL**	对象	**Camel 2.3：** 使用 Camel OGNL 表达式调用的输入正文。
bodyAs(*type*)	类型	**Camel 2.3：** 将正文转换为由 classname 决定的给定类型。转换的正文可以是 null。
bodyAs(*type*).**OGNL**	对象	**Camel 2.18:** 将正文转换为由 classname 决定的给定类型，然后使用 Camel OGNL 表达式调用方法。转换的正文可以是 null。
mandatoryBodyAs(*type*)	类型	**Camel 2.5：** 将正文转换为由 classname 决定的给定类型，并期望正文不是 null。
mandatoryBodyAs(*type*).**OGNL**	对象	**Camel 2.18:** 将正文转换为由 classname 决定的给定类型，然后使用 Camel OGNL 表达式调用方法。
out.body	对象	输出正文
header.foo	对象	请参阅输入 foo 标头

变量	类型	描述
header[foo]	对象	**Camel 2.9.2：** 请参阅输入 foo 标头
headers.foo	对象	请参阅输入 foo 标头
headers[foo]	对象	**Camel 2.9.2：** 请参阅输入 foo 标头
in.header.foo	对象	请参阅输入 foo 标头
in.header[foo]	对象	**Camel 2.9.2：** 请参阅输入 foo 标头
in.headers.foo	对象	请参阅输入 foo 标头
in.headers[foo]	对象	**Camel 2.9.2：** 请参阅输入 foo 标头
header.foo[bar]	对象	**Camel 2.3：** 将输入 foo 标头视为映射，并在映射中使用 bar 作为键进行查找
in.header.foo[bar]	对象	**Camel 2.3：** 将输入 foo 标头视为映射，并在映射中使用 bar 作为键进行查找
in.headers.foo[bar]	对象	**Camel 2.3：** 将输入 foo 标头视为映射，并在映射中使用 bar 作为键进行查找
header.foo.**OGNL**	对象	**Camel 2.3：** 引用输入 foo 标头并使用 Camel OGNL 表达式调用其值。
in.header.foo.**OGNL**	对象	**Camel 2.3：** 引用输入 foo 标头并使用 Camel OGNL 表达式调用其值。
in.headers.foo.**OGNL**	对象	**Camel 2.3：** 引用输入 foo 标头并使用 Camel OGNL 表达式调用其值。
out.header.foo	对象	请参阅 out 标头 foo

变量	类型	描述
out.header[foo]	对象	**Camel 2.9.2：** 请参阅 out 标头 foo
out.headers.foo	对象	请参阅 out 标头 foo
out.headers[foo]	对象	**Camel 2.9.2：** 请参阅 out 标头 foo
headerAs(*key*, *type*)	类型	**Camel 2.5：** 将标头转换为由 classname 决定的给定类型
标头	Map	**Camel 2.9：** 请参阅输入标头
in.headers	Map	**Camel 2.9：** 请参阅输入标头
property.foo	对象	**deprecated ：** 引用交换上的 foo 属性
exchangeProperty.foo	对象	**Camel 2.15：** 引用交换上的 foo 属性
property[foo]	对象	**deprecated ：** 引用交换上的 foo 属性
exchangeProperty[foo]	对象	**Camel 2.15：** 引用交换上的 foo 属性
property.foo.**OGNL**	对象	**deprecated ：** 引用交换上的 foo 属性，并使用 Camel OGNL 表达式调用其值。
exchangeProperty.foo.**OGNL**	对象	**Camel 2.15：** 引用交换上的 foo 属性，并使用 Camel OGNL 表达式调用其值。
sys.foo	字符串	请参阅系统属性

变量	类型	描述
sysenv.foo	字符串	**Camel 2.3：** 请参阅系统环境
例外	对象	**Camel 2.4：** 如果交换上未设定异常，请参阅交换上的异常对象为 **null**。如果交换有，将回退和抓取异常(**Exchange.EXCEPTION_CAUGHT**)。
例外。OGNL	对象	**Camel 2.4：** 请参阅使用 Camel OGNL 表达式对象调用的交换异常
exception.message	字符串	如果交换中没有设置异常，请参阅交换上的 exception.message。**如果交换有，将回退和抓取异常(Exchange.EXCEPTION_CAUGHT)。**
exception.stacktrace	字符串	**Camel 2.6.**如果交换中没有设置异常，请参阅交换上的 exception.stacktrace。**如果交换有，将回退和抓取异常(Exchange.EXCEPTION_CAUGHT)。**
date:_command_	Date	评估为 Date 对象。支持的命令包括： **现在为** 当前的时间戳，**in.header.xxx** 或 **header.xxx**，使用带有键 xxx 的 IN 标头中的 Date 对象。**out.header.xxx**，使用带有键 xxx 的 OUT 标头中的 Date 对象。**property.xxx**，使用带有键 xxx 的 Exchange 属性中的 Date 对象。**文件** 上次修改时间戳的文件（可使用文件使用者使用）。命令接受偏移量，例如： **now-24h** 或 **in.header.xxx+1h** 或 **now+1h30m-100**。
date:_command:pattern_	字符串	使用 **java.text.SimpleDataFormat** 模式进行日期格式化。
date-with-timezone:_command:timezone:pattern_	字符串	使用 **java.text.SimpleDataFormat** timezones 和 pattern 进行日期格式化。
bean:_bean expression_	对象	使用 Bean 语言调用 bean 表达式.指定方法名称，您必须使用点作为分隔符。我们还支持 Bean 组件使用的 ?method=methodname 语法。
properties:_locations:key_	字符串	**弃用（使用 properties-location）Camel 2.3：** 使用给定密钥查找属性。**locations** 选项是可选的。请参阅更多使用 PropertyPlaceholder。

变量	类型	描述
properties-location:_http://locationskey[locations:key]_	字符串	**Camel 2.14.1：** 使用给定键查找属性。**locations** 选项是可选的。请参阅更多使用 PropertyPlaceholder。
properties:key:default	字符串	**Camel 2.14.1** ：使用给定键查找属性。如果键不存在或没有值，则可以指定可选的默认值。
routeId	字符串	**Camel 2.11：** 返回交换正在路由的当前路由的 id。
threadName	字符串	**Camel 2.3：** 返回当前线程的名称。可用于日志记录目的。
ref:xxx	对象	**Camel 2.6：** 使用给定 ID 从 Registry 查找 bean。
type:name.field	对象	**Camel 2.11：** 根据 FQN 名称引用类型或字段。要引用一个字段，您可以附加 .FIELD_NAME。例如，您可以将 Exchange 中的常量字段引用为：**org.apache.camel.Exchange.FILE_NAME**
null	null	**Camel 2.12.3: 代表 null**
random_(value)_	整数	*Camel 2.16.0:*returns a random Integer between 0 (included)和 *value* (excluded)
random_(min, max)_	整数	*Camel 2.16.0:*returns a random Integer between *min* (included)和 *max* (excluded)
collate(group)	list	**Camel 2.17：** collate 函数迭代消息正文，并将数据分组到指定大小的子列表中。这可以与 Splitter EIP 一起使用，将消息正文和组/batch 分成一组 N 子列表。这个方法的工作方式与 Groovy 中的 collate 方法类似。
skip(number)	iterator	**Camel 2.19 ：** 跳过功能会迭代消息正文，并跳过第一个项目数量。这可以与 Splitter EIP 一起使用来分割消息正文，并跳过第一个 N 个项目数。
messageHistory	字符串	**Camel 2.17：** 当前交换的消息历史记录，如何路由。这与路由 stack-trace 消息历史记录（如果未处理的异常）错误处理程序日志类似。

变量	类型	描述
messageHistory(false)	字符串	Camel 2.17: As messageHistory，但没有交换详情（仅包含路由 strack-trace）。如果您不想记录来自消息本身的敏感数据，则可以使用它。

304.4. OGNL 表达式支持

从 Camel 2.3 开始提供

INFO:Camel 的 OGNL 支持仅用于调用方法。您无法访问字段。从 Camel 2.11.1 开始，我们添加了对访问 Java 阵列长度字段的特殊支持。

Simple 和 Bean 语言现在支持以类似方式调用 Bean 的 Camel OGNL 表示法。假设 Message IN 正文包含一个 POJO，其具有 getAddress （） 方法。

然后，您可以使用 Camel OGNL 表示法访问地址对象：

```
simple("${body.address}")
simple("${body.address.street}")
simple("${body.address.zip}")
```

Camel 了解 getter 的简写名称，但您可以调用任何方法或使用实际名称，例如：

```
simple("${body.address}")
simple("${body.getAddress.getStreet}")
simple("${body.address.getZip}")
simple("${body.doSomething}")
```

如果正文没有地址，您还可以使用 null secure operator (?.)来避免 NPE

```
simple("${body?.address?.street}")
```

也可以索引 映射 或 列表 类型，以便您可以进行以下操作：

```
simple("${body[foo].name}")
```

要假定正文是基于 映射的，并使用 foo 作为键查找值，并对该值调用 getName 方法。

如果键有空格，则必须使用 引号包括键，如 'foo bar' ：

```
simple("${body['foo bar'].name}")
```

您可以使用其密钥名称（带有或不使用点）直接访问 Map 或 List 对象：

```
simple("${body[foo]}")
simple("${body[this.is.foo]}")
```

假设没有键 foo 的值，您可以使用 null secure operator 来避免 NPE，如下所示：

```
simple("${body[foo]?.name}")
```

您还可以访问 List 类型，例如从地址中获取行，您可以执行以下操作：

```
simple("${body.address.lines[0]}")
simple("${body.address.lines[1]}")
simple("${body.address.lines[2]}")
```

有一个特殊的 last 关键字，可用于从列表中获取最后一个值。

```
simple("${body.address.lines[last]}")
```

另外，要获得第 2 个可以减去数字，因此我们可以使用 last-1 来指示这一点：

```
simple("${body.address.lines[last-1]}")
```

最后第三篇是：

```
simple("${body.address.lines[last-2]}")
```

您可以在列表中调用 size 方法

```
simple("${body.address.lines.size}")
```

从 Camel 2.11.1 开始，我们添加了对 Java 阵列的 length 字段的支持，例如：

```
String[] lines = new String[]{"foo", "bar", "cat"};
exchange.getIn().setBody(lines);

simple("There are ${body.length} lines")
```

和 yes 您可以将其与 Operator 支持相结合，如下所示：

```
simple("${body.address.zip} > 1000")
```

304.5. OPERATOR 支持

解析器仅限于只支持单个操作器。

要启用它，左侧值必须包含在 $\{ } 中。语法是：

```
${leftValue} OP rightValue
```

其中 rightValue 可以是以 ' '、null、恒定值或以 $\{ } 括起的另一个表达式的 String 文字。

重要

Operator 必须 有空格。

Camel 将自动将 rightValue 类型转换为 leftValue 类型，因此它能够 eg. 将字符串转换为数字，以便您可以将 > 比较用于数字值。

支持以下 Operator：

Operator	描述
==	等于

Operator	描述
=~	**Camel 2.16：** 等于忽略条件（在比较字符串值时将忽略大小写）
>	大于
>=	大于或等于
<	小于
<=	小于或等于
!=	不等于
contains	用于测试，如果基于字符串的值中包含
不包含	对于测试，如果没有包含在基于字符串的值中
~~	对于测试，通过在基于字符串的值中忽略问题单敏感度来进行测试
regex	针对定义为 String 值的正则表达式模式匹配
Not regex	对于不与作为 String 值定义的给定正则表达式模式匹配
in	要在一组值中匹配，则每个元素必须以逗号分隔。如果要包含空值，则必须使用双逗号定义，例如 ',bronze,silver,gold'，它是一个带有空值的四个值，然后是三个 medals。
not in	要在一组值中没有匹配，则每个元素必须以逗号分隔。如果要包含空值，则必须使用双逗号定义，例如 ',bronze,silver,gold'，它是一个带有空值的四个值，然后是三个 medals。
is	如果左侧类型是值的实例，则匹配。
not is	如果左侧类型不是值的实例，则匹配。
range	若要匹配，如果左侧位于定义为数字的范围：**from..to**。从 Camel 2.9 开始，范围值必须用单引号括起。
Not range	如果左侧的值不在定义为 number: **from.to** 的数值范围内进行匹配。从 Camel 2.9 开始，范围值必须用单引号括起。

Operator	描述
从开始	**Camel 2.17.1, 2.18**: 用于测试左侧字符串是否以右手字符串开头。
结束	**Camel 2.17.1, 2.18**: 用于测试左侧字符串以右手字符串结尾。

以下 unary operator 可以使用：

Operator	描述
++	**Camel 2.9 ：** 以 1 递增数字。左侧必须是函数，否则作为文字解析。
–	**Camel 2.9：** 减少一个数字。左侧必须是函数，否则作为文字解析。
\	**Camel 2.9.3 到 2.10.x** 要转义一个值，如 \$，以指示 $ 符号。特殊：将 \n 用于新行，\t 表示选项卡，\r 用于回车。**注意： 不支持使用** 文件语言进行转义。**注意：** 从 Camel 2.11 开始，转义字符不再被支持，但替换为以下三个特殊的转义。
\n	**Camel 2.11：** 使用换行符。
\t	**Camel 2.11：** 使用选项卡字符。
\r	**Camel 2.11 ：** 使用回车返回字符。
\}	**Camel 2.18:** 使用 } 字符作为文本

以下逻辑运算符可用于对表达式进行分组：

Operator	描述
and	**弃用的** use && 替代。逻辑 和 运算符用于对两个表达式进行分组。
or	**弃用** || 替代。logical 或 运算符用于对两个表达式进行分组。

Operator	描述
&&	Camel 2.9： 逻辑和运算符用于对两个表达式进行分组。
||	Camel 2.9： 逻辑或运算符用于对两个表达式进行分组。

重要

使用 and,or 运算符 在 Camel 2.4 或更旧的 和 或 中只能在简单的语言表达式中使用一次。从 Camel 2.5 开始，您可以多次使用这些 Operator。

AND 的语法是：

```
${leftValue} OP rightValue and ${leftValue} OP rightValue
```

而 OR 的语法是：

```
${leftValue} OP rightValue or ${leftValue} OP rightValue
```

一些示例：

```
// exact equals match
simple("${in.header.foo} == 'foo'")

// ignore case when comparing, so if the header has value FOO this will match
simple("${in.header.foo} =~ 'foo'")

// here Camel will type convert '100' into the type of in.header.bar and if it is an Integer '100'
will also be converter to an Integer
simple("${in.header.bar} == '100'")

simple("${in.header.bar} == 100")

// 100 will be converter to the type of in.header.bar so we can do > comparison
simple("${in.header.bar} > 100")
```

304.5.1. 与不同类型的比较

当您与不同类型（如 String 和 int）进行比较时，您必须小心。Camel 将使用左侧的类型作为第一优先级。如果无法根据该类型比较这两个值，并且回退到右侧类型。
这意味着您可以颠倒值来强制执行特定类型的值。假设上面的 bar 值是一个 String。然后您可以反转 equation：

```
simple("100 < ${in.header.bar}")
```

然后，这样可确保 int 类型用作第一优先级。

如果 Camel 团队改进二进制比较操作来首选基于 String 的数字类型，则以后可能会更改此设置。最常见的是 String 类型，与数字进行比较时会导致问题。

```
// testing for null
simple("${in.header.baz} == null")

// testing for not null
simple("${in.header.baz} != null")
```

以及一些更高级的示例，其中正确的值是另一个表达式

```
simple("${in.header.date} == ${date:now:yyyyMMdd}")

simple("${in.header.type} == ${bean:orderService?method=getOrderType}")
```

和包含 的例子，测试标题是否包含 Camel 一词

```
simple("${in.header.title} contains 'Camel'")
```

和带有 regex 的示例，测试数字标头是否为 4 位值：

```
simple("${in.header.number} regex '\\d{4}'")
```

如果标头等于列表中的任何值，则最后是一个示例。每个元素必须用逗号分开，并且没有空格。
这也适用于数字等，因为 Camel 会将每个元素转换为左侧类型的类型。

```
simple("${in.header.type} in 'gold,silver'")
```

对于最后 3，我们也支持使用以下内容的 negate 测试：

-

```
simple("${in.header.type} not in 'gold,silver'")
```

您可以测试该类型是否是特定的实例，例如，一个字符串

```
simple("${in.header.type} is 'java.lang.String'")
```

我们为所有 java.lang 类型添加了简写，以便您可以将其写为：

```
simple("${in.header.type} is 'String'")
```

也支持范围。范围间隔需要数字，并且来自 和 end 都包含。例如，测试值是否为 100 到 199 之间：

```
simple("${in.header.number} range 100..199")
```

请注意，我们在没有空格的范围中使用 ..。它采用与 Groovy 相同的语法。

从 Camel 2.9 开始，范围值必须使用单引号

```
simple("${in.header.number} range '100..199'")
```

304.5.2. 使用 Spring XML

由于 Spring XML 没有任何权限作为其各种构建器方法的 Java DSL，因此您必须利用其他语言来测试简单操作器。现在，您可以使用简单语言进行此操作。在以下示例中，我们希望测试标头是否为小部件顺序：

```
<from uri="seda:orders">
  <filter>
    <simple>${in.header.type} == 'widget'</simple>
    <to uri="bean:orderService?method=handleWidget"/>
  </filter>
</from>
```

304.6. 使用 和 / 或

如果您有两个表达式，您可以将它们与 and 或 or 运算符合并。

提示

Camel 2.9 onwards Use && or || from Camel 2.9 onwards.

例如：

```
simple("${in.header.title} contains 'Camel' and ${in.header.type'} == 'gold'")
```

当然也支持 或。示例将是：

```
simple("${in.header.title} contains 'Camel' or ${in.header.type'} == 'gold'")
```

注意：当前 和 或 只能在简单语言表达式中使用 一次。这可能会在以后改变。因此，您无法 ：

```
simple("${in.header.title} contains 'Camel' and ${in.header.type'} == 'gold' and
${in.header.number} range 100..200")
```

304.7. SAMPLES

在下面的 Spring XML 示例中，我们根据标头值进行过滤：

```
<from uri="seda:orders">
  <filter>
    <simple>${in.header.foo}</simple>
    <to uri="mock:fooOrders"/>
  </filter>
</from>
```

Simple 语言可用于上面的 predicate 测试，在 Message Filter 模式中测试 中是否有 foo 标头（带有键 foo 的标头）。如果表达式评估为 true，则消息将路由到 mock:fooOrders 端点，否则消息将被丢弃。

Java DSL 中的相同示例：

```
from("seda:orders")
   .filter().simple("${in.header.foo}")
      .to("seda:fooOrders");
```

您还可以将简单语言用于简单文本串联，例如：

```
from("direct:hello")
   .transform().simple("Hello ${in.header.user} how are you?")
   .to("mock:reply");
```

请注意，我们必须在表达式中使用 $\{ } 占位符，以允许 Camel 正确解析它。

这个示例使用 date 命令输出当前日期。

```
from("direct:hello")
   .transform().simple("The today is ${date:now:yyyyMMdd} and it is a great day.")
   .to("mock:reply");
```

在以下示例中，我们调用 bean 语言来调用要包含在返回的字符串中的 bean 方法：

```
from("direct:order")
   .transform().simple("OrderId: ${bean:orderIdGenerator}")
   .to("mock:reply");
```

其中 orderIdGenerator 是 Registry 中注册的 bean 的 id。如果使用 Spring，则它是 Spring bean id。

如果要声明对顺序 id 生成器 bean 调用的方法，我们必须首先将 .method 名称 放在下方，我们调用 generateId 方法。

```
from("direct:order")
   .transform().simple("OrderId: ${bean:orderIdGenerator.generateId}")
   .to("mock:reply");
```

我们可以使用 ?method=methodname 选项，我们熟悉 Bean 组件本身：

```
from("direct:order")
   .transform().simple("OrderId: ${bean:orderIdGenerator?method=generateId}")
   .to("mock:reply");
```

从 Camel 2.3 开始，您还可以将正文转换为给定类型，例如确保它是一个 String：

```
<transform>
  <simple>Hello ${bodyAs(String)} how are you?</simple>
</transform>
```

有几个类型有一个简写表示法，因此我们可以使用 String 而不是 java.lang.String。这些是： byte[], String, Integer, Long。所有其他类型必须使用其 FQN 名称，如 org.w3c.dom.Document。

也可以从 Camel 2.3 以后的标头 映射 中查找值：

```
<transform>
  <simple>The gold value is ${header.type[gold]}</simple>
</transform>
```

在上面的代码中，我们查找名为 type 的标头，并将其命名为 java.util.Map，然后使用键 gold 进行查找并返回值。如果标头无法转换为 Map，则会抛出异常。如果名为 type 的标头不存在 null，则返回 null。

从 Camel 2.9 开始，您可以嵌套功能，如下所示：

```
<setHeader headerName="myHeader">
  <simple>${properties:${header.someKey}}</simple>
</setHeader>
```

304.8. 引用常数或枚举

从 Camel 2.11 开始提供

假设您有一个面向客户的枚举

在 Content Based Router 中，我们可以使用 Simple 语言来引用此 enum，来检查其匹配的消息。

304.9. 在 XML DSLS 中使用新行或标签页

从 Camel 2.9.3 开始提供

从 Camel 2.9.3 开始，您可以在 XML DSL 中指定新行或标签页，因为您可以现在转义值

```
<transform>
  <simple>The following text\nis on a new line</simple>
</transform>
```

304.10. 前导和结尾的空格处理

从 Camel 2.10.0 开始提供

从 Camel 2.10.0 开始，可以使用表达式的 trim 属性来控制是否删除前导和尾随空格字符。默认值为 true，它会删除空格字符。

```
<setBody>
  <simple trim="false">You get some trailing whitespace characters.    </simple>
</setBody>
```

304.11. 设置结果类型

从 Camel 2.8 开始提供

现在，您可以为 Simple 表达式提供结果类型，这意味着评估的结果将转换为所需的类型。这最常用于定义类型，如布尔值、整数等。

例如，要将标头设置为布尔值类型，您可以执行以下操作：

```
.setHeader("cool", simple("true", Boolean.class))
```

在 XML DSL 中

```
<setHeader headerName="cool">
  <!-- use resultType to indicate that the type should be a java.lang.Boolean -->
  <simple resultType="java.lang.Boolean">true</simple>
</setHeader>
```

304.12. 更改功能启动和端点令牌

从 Camel 2.9.1 开始提供

您可以使用 setters changeFunctionStartToken 和 changeFunctionEndToken on SimpleLanguage 配置功能 start 和 end 令牌 - $\{ }，使用 Java 代码。在 Spring XML 中，您可以使用属性中的新已更改令牌定义 <bean> 标签，如下所示：

```
<!-- configure Simple to use custom prefix/suffix tokens -->
<bean id="simple" class="org.apache.camel.language.simple.SimpleLanguage">
  <property name="functionStartToken" value="["/>
  <property name="functionEndToken" value="]"/>
</bean>
```

在上例中，我们使用 [ ] 作为更改的令牌。

通过更改 start/end 令牌，您可以在所有在 classpath 上共享相同 camel-core 的 Camel 应用程序中更改这些令牌。
例如，在 OSGi 服务器中，这可能会影响许多应用程序，其中 Web 应用程序作为 WAR 文件，它只会影响 Web 应用。

304.13. 从外部资源载入脚本

从 Camel 2.11 开始提供

您可以对脚本进行外部化，并让 Camel 从资源（如 "classpath:"、"file:" 或 "http:" ）加载它。这可以通过以下语法完成：" resource:scheme:location"，例如引用您可以进行的类路径上的文件：

```
.setHeader("myHeader").simple("resource:classpath:mysimple.txt")
```

304.14. 将 SPRING BEAN 设置为 EXCHANGE 属性

从 Camel 2.6 开始提供

您可以将 spring bean 设置为交换属性，如下所示：

```
<bean id="myBeanId" class="my.package.MyCustomClass" />
...
<route>
  ...
  <setProperty propertyName="monitoring.message">
    <simple>ref:myBeanId</simple>
  </setProperty>
  ...
</route>
```

-

304.15. 依赖项

Simple 语言是 camel-core 的一部分。

第 305 章 SIP 组件

从 Camel 版本 2.5 开始提供

Camel 中的 sip 组件是一个通信组件，基于 Jain SIP 实施（在 JCP 许可证下可用）。

会话初始协议(SIP)是一种 IETF 定义信号协议，广泛用来控制多媒体通信会话，如语音和视频会议，如语音和视频调用互联网协议(IP)。SIP 协议是独立于底层传输层的应用程序层协议，它可以在传输控制协议(TCP)、用户数据报协议(UDP)或流控制传输协议(SCTP)上运行。

Jain SIP 实施仅支持 TCP 和 UDP。

Camel SIP 组件 只支持 SIP Publish 和 Subscribe 功能，如 RFC3903 - Session Initiation Protocol (SIP) Extension for Event所述

此 camel 组件支持生成者和消费者端点。

Camel SIP Producers (Event publishers)和 SIP Consumers (Event Subscribers)使用名为 SIP Presence Agent （有状态代理）的中间实体（有状态代理实体）传达事件和状态信息。

对于基于 SIP 的通信，带有侦听器的 SIP 堆栈 必须在 SIP Producer 和 Consumer 上实例化（如果使用 localhost，则使用单独的端口）。这是为了支持在通信期间 SIP 堆栈之间交换的握手和确认功能所必需的。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-sip</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

305.1. URI 格式

sip 端点的 URI 方案如下：

```
sip://johndoe@localhost:99999[?options]
sips://johndoe@localhost:99999/[?options]
```

此组件支持 TCP 和 UDP 的制作者和消费者端点。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

305.2. 选项

SIP 组件提供了一组广泛的配置选项，并可创建通过 SIP 协议传播状态所需的自定义有状态标头。

SIP 组件没有选项。

SIP 端点使用 URI 语法进行配置：

```
sip:uri
```

使用以下路径和查询参数：

305.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
uri	要连接的 SIP 服务器 **所需的** URI （用户名和密码可以包括： john:secretmyserver:9999）		URI

305.2.2. 查询参数(44 参数)：

Name	描述	默认值	类型
cacheConnections (common)	应 SipStack 缓存连接，以降低连接创建成本。如果连接用于长时间运行的对话，这非常有用。	false	布尔值

Name	描述	默认值	类型
contentSubType (common)	为 contentSubType 设置可设置为任何有效的 MimeSubType。	plain	字符串
contentType (common)	contentType 的设置可以设置为任何有效的 MimeType。	text	字符串
eventHeaderName (common)	为基于 String 的事件类型设置。		字符串
eventId (common)	为基于 String 的事件 Id 设置。除非指定了基于 registry 的 FromHeader，否则强制设置		字符串
fromHost (common)	消息 originator 的主机名。除非指定了基于 registry 的 FromHeader，否则强制设置		字符串
FromPort (common)	消息 originator 的端口。除非指定了基于 registry 的 FromHeader，否则强制设置		int
fromUser (common)	消息 originator 的用户名。除非指定了基于 registry 的自定义 FromHeader，否则强制设置。		字符串
msgExpiration (common)	端点收到的消息被视为有效的时间	3600	int
receiveTimeoutMillis (common)	设置以指定等待 Response 和/或 Acknowledgement 的时间，可以从另一个 SIP 堆栈接收	10000	long
stackName (common)	与 SIP 端点关联的 SIP Stack 实例的名称。	NAME_NOT_SET	字符串
toHost (common)	消息接收器的主机名。除非指定了基于 registry 的 ToHeader，否则强制设置		字符串
ToPort (common)	消息接收器的 PORTNAME。除非指定了基于 registry 的 ToHeader，否则强制设置		int
toUser (common)	消息接收器的用户名。除非指定了基于 registry 的自定义 ToHeader，否则强制设置。		字符串
transport (common)	设置，以选择传输协议。有效选择是 tcp 或 udp。	tcp	字符串

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
consumer (consumer)	此设置用于确定是否为此端点创建的标头类型 (FromHeader、ToHeader 等)	false	布尔值
presenceAgent (consumer)	此设置用于区分 Presence Agent 和消费者。这是因为 SIP Camel 组件附带一个基本优先级代理（仅用于测试目的）。消费者必须将此标志设置为 true。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
addressFactory (advanced)	使用自定义 AddressFactory		AddressFactory
callIdHeader (advanced)	包含调用详情的自定义 Header 对象。必须实施类型 javax.sip.header.CallIdHeader		CallIdHeader
contactHeader (advanced)	可选的自定义 Header 对象，其中包含详细联系人详细信息（电子邮件、电话号码等）。必须实施类型 javax.sip.header.ContactHeader		ContactHeader
contentTypeHeader (advanced)	包含消息内容详情的自定义 Header 对象。必须实施类型 javax.sip.header.ContentTypeHeader		ContentTypeHeader
eventHeader (advanced)	包含事件详情的自定义 Header 对象。必须实施类型 javax.sip.header.EventHeader		EventHeader
expiresHeader (advanced)	包含消息过期详情的自定义 Header 对象。必须实施类型 javax.sip.header.ExpiresHeader		ExpiresHeader
extensionHeader (advanced)	包含用户/应用程序特定详情的自定义 Header 对象。必须实施类型 javax.sip.header.ExtensionHeader		ExtensionHeader
fromHeader (advanced)	包含消息 originator 设置的自定义 Header 对象。必须实施类型 javax.sip.header.FromHeader		FromHeader

Name	描述	默认值	类型
headerFactory (advanced)	使用自定义 HeaderFactory		HeaderFactory
listeningPoint (advanced)	使用自定义 ListeningPoint 实现		ListeningPoint
maxForwardsHeader (advanced)	包含最大代理转发详情的自定义 Header 对象。这个标头会对 viaHeaders 施加一个限制。必须实施类型 javax.sip.header.MaxForwardsHeader		MaxForwardsHeader
maxMessageSize (advanced)	设置允许的最大消息大小（以字节为单位）。	1048576	int
messageFactory (advanced)	使用自定义 MessageFactory		MessageFactory
sipFactory (advanced)	使用自定义 SipFactory 创建要使用的 SipStack		SipFactory
sipStack (advanced)	使用自定义 SipStack		SipStack
sipUri (advanced)	使用自定义 SipURI。如果没有配置，则 SipUri fallback 使用选项 toUser toHost:toPort		SipURI
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
toHeader (advanced)	包含消息接收器设置的自定义 Header 对象。必须实施类型 javax.sip.header.ToHeader		ToHeader
viaHeaders (advanced)	类型 javax.sip.header.ViaHeader 的自定义 Header 对象列表。每个 ViaHeader 都包含用于请求转发的代理地址。（请注意，当请求到达其监听器时，每个代理会自动更新此标头。）		list
implementationDebugLogFile (logging)	用于日志记录的客户端调试日志文件的名称		字符串
implementationServerLogFile (logging)	用于日志记录的服务器日志文件的名称		字符串
implementationTraceLevel (logging)	追踪的日志记录级别	0	字符串

Name	描述	默认值	类型
maxForwards (proxy)	最大代理转发数		int
useRouterForAllUris (proxy)	当请求通过代理发送到 Presence 代理时，会使用此设置。	false	布尔值

305.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.sip.enabled	启用 sip 组件	true	布尔值
camel.component.sip.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

305.4. 将消息发送到/从 SIP 端点

305.4.1. 创建 Camel SIP publisher

在以下示例中，创建一个 SIP publisher 来向用户"agent@localhost:5152"发送 SIP 事件发布。这是 SIP Presence Agent 的地址，它充当 SIP publisher 和 Subscriber 之间的代理

- *使用名为 client 的 SIP Stack*
- *使用一个名为 evtHdrName 的基于 registry 的 eventHeader*
- *使用一个名为 evtId 的基于 registry 的 eventId*
- *从带有 Listener 的 SIP 堆栈中，设置为 user2@localhost:3534*

- 发布的事件是 EVENT_A

- 名为 REQUEST_METHOD 的 Mandatory Header 设置为 Request.Publish 因此，将端点设置为 Event publisher"

```
producerTemplate.sendBodyAndHeader(
    "sip://agent@localhost:5152?
stackName=client&eventHeaderName=evtHdrName&eventId=evtid&fromUser=user2&fromHos
t=localhost&fromPort=3534",
    "EVENT_A",
    "REQUEST_METHOD",
    Request.PUBLISH);
```

305.4.2. 创建 Camel SIP Subscriber

在以下示例中，创建一个 SIP Subscriber 来接收发送到的 SIP 事件发布程序"johndoe@localhost:5154"

- 使用名为 Subscriber 的 SIP Stack

- 使用名为 agent@localhost:5152 的 Presence Agent 用户注册

- 使用一个名为 evtHdrName 的基于 registry 的 eventHeader。evtHdrName 包含事件，该事件为 se to "Event_A"

- 使用一个名为 evtId 的基于 registry 的 eventId

```
@Override
protected RouteBuilder createRouteBuilder() throws Exception {
    return new RouteBuilder() {
        @Override
        public void configure() throws Exception {
            // Create PresenceAgent
            from("sip://agent@localhost:5152?
stackName=PresenceAgent&presenceAgent=true&eventHeaderName=evtHdrName&eventId=e
vtid")
                .to("mock:neverland");

            // Create Sip Consumer(Event Subscriber)
            from("sip://johndoe@localhost:5154?
stackName=Subscriber&toUser=agent&toHost=localhost&toPort=5152&eventHeaderName=ev
```

```
tHdrName&eventId=evtid")
                .to("log:ReceivedEvent?level=DEBUG")
                .to("mock:notification");

        }
    };
}
```

Camel SIP 组件也附带了一个 Presence Agent，仅用于测试和演示目的。以上提供了实例化 Presence Agent 的示例。

请注意，Presence Agent 设置为用户 agent@localhost:5152，能够与 publisher 和 Subscriber 通信。它有一个与 publisher 和 Subscriber 不同的 SIP stackName。虽然它被设置为 Camel Consumer，但它实际上不会将任何消息与路由一起发送到端点 "mock:neverland"。

第 306 章 简单的 JMS BATCH 组件

从 Camel 版本 2.16 开始提供

SJMS Batch 是专门的组件，可用于来自 JMS 队列的高性能、交易型批处理消耗。它可以被认为是仅限消费者组件和聚合器的混合。

Camel 中的常见用例是消耗来自队列的消息，并在将聚合状态发送到另一个端点前聚合它们。为确保在系统执行处理失败时数据不会丢失，它通常在队列中的事务中消耗，一次聚合存储在持久 聚合 Repository 中，如 JDBC 组件 中找到的。

聚合器模式的行为涉及从 AggregationRepository 获取数据，然后再聚合传入消息，之后再写入结果。本质上，当聚合的工件数量增加时，读取和写入会逐渐长。使用表示此问题影响的任意时间单位的粗略示例如下：

项	读取时间	写时间	总时间
0	0	1	1
1	1	2	4
2	2	3	9
3	3	4	16
4	4	5	25
5	5	6	36
6	6	7	49
7	7	8	64
8	8	9	81
9	9	10	100

相反，使用 SJMS 批处理组件消耗性能是线性的。在获取下一个消息前，使用 AggregationStrategy 来消耗并聚合每个消息。因为所有消耗和聚合都是在单个 JMS 事务中执行的，因此不需要外部存储才能保持中间状态 - 这可避免上面描述的读写成本。实际上，这会产生多个数量较高的吞吐量顺序。

满足完成条件后，从第一个消息起按大小或周期，聚合的 Exchange 将传递到路由中。在处理此 交换过程中，如果抛出异常或系统关闭，则所有原始使用的消息都会返回到队列（或根据代理配置），所有原始使用的消息都会备份到死信队列。

与使用常规聚合器不同，没有聚合条件的工具，这意味着无法批量使用多个消息组。所有使用的消息都会聚合到一个批处理中。

提供的多个 JMS 使用者支持允许您利用一个路由并行使用，并且同时使用 JMS 消息组等功能来分组相关消息。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-sjms</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

306.1. URI 格式

```
sjms:[queue:]destinationName[?options]
```

其中 destinationName 是 JMS 队列。默认情况下，destinationName 解释为队列名称。

```
sjms:FOO.BAR
```

如果需要，您可以包含可选的 queue: 前缀：

```
sjms:queue:FOO.BAR
```

不支持主题消耗，因为该上下文中无法使用批处理。主题消息通常是非持久性的，可以接受丢失。如果在失败的事务中使用，则代理可能会不会重新显示主题信息。在这种情况下，普通的 SJMS 消费者端点可与常规的非永久性支持聚合器一起使用。

306.2. 组件选项和配置

Simple JMS Batch 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
ConnectionFactory (advanced)	需要 ConnectionFactory 才能启用 SjmsBatchComponent。		ConnectionFactory
asyncStartListener (advanced)	在启动路由时，是否异步启动消费者消息监听程序。例如，如果 JmsConsumer 无法连接到远程 JMS 代理，则在重试和/或故障转移时可能会阻止。这将导致 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 以异步模式使用专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果无法建立连接，那么会记录异常在 WARN 级别，消费者将无法接收消息；然后您可以重启路由来重试。	false	布尔值
recoveryInterval (advanced)	指定恢复尝试之间的间隔，例如当连接被刷新时，以毫秒为单位。默认值为 5000 ms，即 5 秒。	5000	int
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Simple JMS Batch 端点使用 URI 语法进行配置：

```
sjms-batch:destinationName
```

使用以下路径和查询参数：

306.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
destinationName	**必需** 目的地名称。仅支持队列，名称可能以 'queue:' 前缀。		字符串

306.2.2. 查询参数(23 参数)：

Name	描述	默认值	类型
aggregationStrategy (consumer)	**必需** 使用的聚合策略，它会将所有批处理的消息合并到单个消息中		AggregationStrategy
allowNullBody (consumer)	是否允许发送没有正文的消息。如果此选项为 false，并且消息正文为 null，则抛出 JMSException。	true	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
completionInterval (consumer)	millis 中的完成间隔，每个间隔都会以固定率完成批处理。如果触发超时并且批处理中没有消息，则批处理可能为空。请注意，您不能同时使用完成超时和完成间隔，只能配置一个。	1000	int
completionPredicate (consumer)	completion predicate，当 predicate 评估为 true 时，它会导致批处理完成。predicate 也可以使用字符串语法简单的语言进行配置。您可能希望将 eagerCheckCompletion 选项设为 true，以便 predicate 与传入消息匹配，否则它与聚合消息匹配。		字符串
completionSize (consumer)	批处理要完成的消息数量	200	int
completionTimeout (consumer)	当批处理完成后，第一个消息被接收后的 millis 中超时。如果触发超时并且批处理中没有消息，则批处理可能为空。请注意，您不能同时使用完成超时和完成间隔，只能配置一个。	500	int
consumerCount (consumer)	要从其中使用的 JMS 会话数量	1	int
eagerCheckCompletion (consumer)	使用 eager 完成检查，这意味着 completionPredicate 将使用传入的交换。在不强制完成检查 completionPredicate 的情况下，将使用聚合的交换。	false	布尔值
includeAllJMSXProperties (consumer)	从 JMS 映射到 Camel 消息时，是否要包含所有 JMSXxxx 属性。把它设置为 true 将包括 JMSXAppID 和 JMSXUserID 等属性。注： 如果您使用自定义 headerFilterStrategy，则不会应用这个选项。	false	布尔值
mapJmsMessage (consumer)	指定 Camel 是否应该自动将收到的 JMS 消息映射到合适的有效负载类型，如 javax.jms.TextMessage 到 String 等。如需了解更多详细信息，请参阅映射如何工作。	true	布尔值

Name	描述	默认值	类型
pollDuration (consumer)	每个轮询消息的持续时间（毫秒）。如果消息比较短且批处理已启动，则将使用 completionTimeOut。	1000	int
sendEmptyMessageWhenIdle (consumer)	如果使用完成超时或间隔，则在触发超时且没有批处理中没有消息时，批处理可能为空。如果此选项为 true，并且批处理为空，则将向批处理添加一个空消息，以便路由空消息。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
asyncStartListener (advanced)	在启动路由时，是否异步启动消费者消息监听程序。例如，如果 JmsConsumer 无法连接到远程 JMS 代理，则在重试和/或故障转移时可能会阻止。这将导致 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 以异步模式使用专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果无法建立连接，那么会记录异常在 WARN 级别，消费者将无法接收消息；然后您可以重启路由来重试。	false	布尔值
headerFilterStrategy (advanced)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
jmsKeyFormatStrategy (advanced)	可插拔编码和解码 JMS 密钥策略，以便它们能够与 JMS 规范兼容。Camel 提供了两个开箱即用的实现：default 和 passthrough。默认策略可以安全地放入点和连字符(. 和 -)。passthrough 策略将密钥保留原样。可用于 JMS 代理，其不小心是 JMS 标头键是否包含非法字符。您可以提供自己的 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 # 表示法引用它。		JmsKeyFormatStrategy
keepAliveDelay (advanced)	尝试重新建立有效会话之间的延迟。如果这是一个正值，SjmsBatchConsumer 将尝试创建新会话，如果它在消息消耗期间看到 IllegalStateException。此延迟值允许您在尝试防止垃圾日志之间暂停。如果这是一个负值（默认为 -1），则 SjmsBatchConsumer 将像在之前一样的行为，而如果看到 IllegalStateException，则路由将关闭。	-1	int
messageCreatedStrategy (advanced)	要使用给定的 MessageCreatedStrategy，后者在 Camel 在 Camel 发送 JMS 消息时创建 javax.jms.Message 对象的新实例。		MessageCreatedStrategy

Name	描述	默认值	类型
recoveryInterval (advanced)	指定恢复尝试之间的间隔，例如当连接被刷新时，以毫秒为单位。默认值为 5000 ms，即 5 秒。	5000	int
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
timeoutCheckerExecutor Service (advanced)	如果使用 completionInterval 选项，则会创建一个后台线程来触发完成间隔。设置这个选项，以提供要使用的自定义线程池，而不是为每个消费者创建新线程。		ScheduledExecutor Service

306.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
camel.component.sjms-batch.async-start-listener	在启动路由时，是否异步启动消费者消息监听程序。例如，如果 JmsConsumer 无法连接到远程 JMS 代理，则在重试和/或故障转移时可能会阻止。这将导致 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 以异步模式使用专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果无法建立连接，那么会记录异常在 WARN 级别，消费者将无法接收消息；然后您可以重启路由来重试。	false	布尔值
camel.component.sjms-batch.connection-factory	需要 ConnectionFactory 才能启用 SjmsBatchComponent。选项是 javax.jms.ConnectionFactory 类型。		字符串
camel.component.sjms-batch.enabled	启用 sjms-batch 组件	true	布尔值
camel.component.sjms-batch.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		字符串
camel.component.sjms-batch.recovery-interval	指定恢复尝试之间的间隔，例如当连接被刷新时，以毫秒为单位。默认值为 5000 ms，即 5 秒。	5000	整数

Name	描述	默认值	类型
camel.component.sjms-batch.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

completionSize 端点属性与 completionTimeout 结合使用，其中第一个条件将会导致路由发出聚合的 Exchange。

第 307 章 简单的 JMS 组件

从 Camel 版本 2.11 开始提供

Simple JMS 组件或 SJMS 是 JMS 客户端，可与 Camel 一起使用，在涉及 JMS 客户端创建和配置时，使用已知最佳实践。SJMS 包含为 Camel 明确编写的全新的 JMS 客户端 API，消除了第三方消息传递实施可保持其轻性和弹性。包括以下功能：

- Standard Queue and Topic Support (Durable and Non-Durable)
- InOnly and InOut MEP 支持
- 异步 Producer 和 Consumer Processing
- 内部 JMS 事务支持

其他主要功能包括：

- 可插入连接资源管理
- Session, Consumer, 和 Producer Pooling and caching Management
- 批处理消费者和 Producers
- Transacted Batch Consumers & Producers
- 支持可自定义交易提交策略（仅限本地 JMS 事务）

注意

为什么 S在 SJMS 中

s 代表简单和标准，以及 Springless。另外，camel-jms 已被使用。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-sjms</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

307.1. URI 格式

```
sjms:[queue:|topic:]destinationName[?options]
```

其中 destinationName 是 JMS 队列或主题名称。默认情况下，destinationName 解释为队列名称。例如，要连接到队列，使用 FOO.BAR ：

```
sjms:FOO.BAR
```

如果需要，您可以包含可选的 queue: 前缀：

```
sjms:queue:FOO.BAR
```

要连接到主题，您必须 包含 topic: 前缀。例如，要连接到主题 Stocks.Prices，请使用：

```
sjms:topic:Stocks.Prices
```

您可以使用以下格式将查询选项附加到 URI 中，?option=value&option=value&...

307.2. 组件选项和配置

Simple JMS 组件支持 15 个选项，如下所列。

Name	描述	默认值	类型
ConnectionFactory (advanced)	需要 ConnectionFactory 才能启用 SjmsComponent。它可以直接设置，也可以设置为 ConnectionResource 的一部分。		ConnectionFactory
connectionResource (advanced)	ConnectionResource 是一个接口，允许对 ConnectionFactory 进行自定义和容器控制。如需了解更多详细信息，请参阅 插件连接资源管理。		ConnectionResource
connectionCount (common)	此组件下启动的端点的最大可用连接数	1	整数
jmsKeyFormatStrategy (advanced)	可插拔编码和解码 JMS 密钥策略，以便它们能够与 JMS 规范兼容。Camel 提供了开箱即用的一个实施：default.默认策略可以安全地放入点和连字符(. 和 -)。可用于 JMS 代理，其不小心是 JMS 标头键是否包含非法字符。您可以提供自己的 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 # 表示法引用它。		JmsKeyFormatStrategy
transactionCommit 策略 (transaction)	要配置要使用的提交策略类型。Camel 开箱即用提供了两个实现，即 default 和 batch。		TransactionCommit 策略
destinationCreation Strategy (advanced)	使用自定义 DestinationCreationStrategy。		DestinationCreation 策略
timedTaskManager (advanced)	使用自定义 TimedTaskManager		TimedTaskManager
messageCreatedStrategy (advanced)	要使用给定的 MessageCreatedStrategy，后者在 Camel 在 Camel 发送 JMS 消息时创建 javax.jms.Message 对象的新实例。		MessageCreatedStrategy
connectionTestOnBorrow (advanced)	在使用默认的 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时，在从池返回之前，每个 javax.jms.Connection 应经过测试（调用 start）。	true	布尔值
connectionUsername (security)	创建 javax.jms.Connection 时使用的用户名，在使用默认的 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时使用。		字符串

Name	描述	默认值	类型
connectionPassword (security)	创建 javax.jms.Connection 时使用的密码，在使用默认的 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时使用。		字符串
connectionClientId (advanced)	创建 javax.jms.Connection 时使用的客户端 ID，在使用默认的 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时使用。		字符串
connectionMaxWait (advanced)	当池使用默认 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时，池被耗尽时，等待 millis 的最大等待时间。	5000	long
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Simple JMS 端点使用 URI 语法进行配置：

```
sjms:destinationType:destinationName
```

使用以下路径和查询参数：

307.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
destinationType	要使用的目的地类型	queue	字符串
destinationName	**required** DestinationName 是 JMS 队列或主题名称。默认情况下，destinationName 解释为队列名称。		字符串

307.2.2. 查询参数(34 参数)：

Name	描述	默认值	类型
acknowledgementMode (common)	JMS 确认名称，即 SESSION_TRANSACTED、CLIENT_ACKNOWLEDGE、AUTO_ACKNOWLEDGE, DUPS_OK_ACKNOWLEDGE	AUTO_确认	SessionAcknowledgement Type
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
consumerCount (consumer)	设置用于此端点的消费者监听程序数量。	1	int
durableSubscriptionId (consumer)	设置持久主题所需的持久订阅 Id。		字符串
同步 (consumer)	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	true	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
messageSelector (consumer)	设置 JMS Message selector 语法。		字符串
namedReplyTo (producer)	将回复设置为用于 InOut producer 端点的目的地名称。回复目的地的类型可以通过其名称中的起始前缀 (topic: 或 queue:)决定。		字符串
persistent (producer)	用于启用/禁用消息持久性的标志。	true	布尔值
producerCount (producer)	设置用于此端点的制作者数量。	1	int
TTL (producer)	用于调整生成消息的 Time To Live 值的标志。	-1	long
allowNullBody (producer)	是否允许发送没有正文的消息。如果此选项为 false，并且消息正文为 null，则抛出 JMSException。	true	布尔值

Name	描述	默认值	类型
prefillPool (producer)	是否在启动时预先填充制作者连接池，或者根据需要创建连接 lazy。	true	布尔值
responseTimeOut (producer)	设置在超时 InOut 响应前应等待的时间。	5000	long
asyncStartListener (advanced)	在启动路由时，是否异步启动消费者消息监听程序。例如，如果 JmsConsumer 无法连接到远程 JMS 代理，则在重试和/或故障转移时可能会阻止。这将导致 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 以异步模式使用专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果无法建立连接，那么会记录异常在 WARN 级别，消费者将无法接收消息；然后您可以重启路由来重试。	false	布尔值
asyncStopListener (advanced)	在停止路由时，是否异步停止消费者消息监听程序。	false	布尔值
connectionCount (advanced)	此端点可用的最大连接数		整数
ConnectionFactory (advanced)	初始化端点的 connectionFactory，它优先于组件的 connectionFactory （若有）		ConnectionFactory
connectionResource (advanced)	初始化端点的 connectionResource，它优先于组件的 connectionResource （若有）		ConnectionResource
destinationCreationStrategy (advanced)	使用自定义 DestinationCreationStrategy。		DestinationCreation 策略
exceptionListener (advanced)	指定正在获得任何底层 JMS 异常通知的 JMS Exception Listener。		ExceptionListener
headerFilterStrategy (advanced)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
includeAllJMSXProperties (advanced)	从 JMS 映射到 Camel 消息时，是否要包含所有 JMSXxxx 属性。把它设置为 true 将包括 JMSXAppID 和 JMSXUserID 等属性。注： 如果您使用自定义 headerFilterStrategy，则不会应用这个选项。	false	布尔值

Name	描述	默认值	类型
jmsKeyFormatStrategy (advanced)	可插拔编码和解码 JMS 密钥策略，以便它们能够与 JMS 规范兼容。Camel 提供了两个开箱即用的实现：default 和 passthrough。默认策略可以安全地放入点和连字符(. 和 -)。passthrough 策略将密钥保留原样。可用于 JMS 代理，其不小心是 JMS 标头键是否包含非法字符。您可以提供自己的 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 # 表示法引用它。		JmsKeyFormatStrategy
mapJmsMessage (advanced)	指定 Camel 是否应该自动将收到的 JMS 消息映射到合适的有效负载类型，如 javax.jms.TextMessage 到 String 等。如需了解更多详细信息，请参阅映射如何工作。	true	布尔值
messageCreatedStrategy (advanced)	要使用给定的 MessageCreatedStrategy，后者在 Camel 在 Camel 发送 JMS 消息时创建 javax.jms.Message 对象的新实例。		MessageCreatedStrategy
errorHandlerLoggingLevel (logging)	允许为日志记录未捕获的异常配置默认的 errorHandler 日志记录级别。	WARN	LoggingLevel
errorHandlerLogStackTrace (logging)	允许由默认的 errorHandler 控制是否应记录堆栈追踪。	true	布尔值
转换 （事务）	指定是否使用转换模式	false	布尔值
transactionBatchCount (transaction)	如果转换在提交事务前将进程的消息数量设置为进程。	-1	int
transactionBatchTimeout (transaction)	为批处理事务设置超时（在 millis 中），该值应为 1000 或更高。	5000	long
transactionCommitStrategy (transaction)	设置提交策略。		TransactionCommit 策略
sharedJMSSession (transaction)	指定是否与其他 SJMS 端点共享 JMS 会话。如果您的路由正在访问多个 JMS 提供程序，则关闭此项。如果您需要对多个 JMS 提供程序的事务，请使用 jms 组件来利用 XA 事务。	true	布尔值

307.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 15 个选项，如下所列。

Name	描述	默认值	类型
camel.component.sjms.connection-client-id	创建 javax.jms.Connection 时使用的客户端 ID，在使用默认的 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时使用。		字符串
camel.component.sjms.connection-count	此组件下启动的端点的最大可用连接数	1	整数
camel.component.sjms.connection-factory	需要 ConnectionFactory 才能启用 SjmsComponent。它可以直接设置，也可以设置为 ConnectionResource 的一部分。选项是 javax.jms.ConnectionFactory 类型。		字符串
camel.component.sjms.connection-max-wait	当池使用默认 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时，池被耗尽时，等待 millis 的最大等待时间。	5000	Long
camel.component.sjms.connection-password	创建 javax.jms.Connection 时使用的密码，在使用默认的 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时使用。		字符串
camel.component.sjms.connection-resource	ConnectionResource 是一个接口，允许对 ConnectionFactory 进行自定义和容器控制。如需了解更多详细信息，请参阅 插件连接资源管理。选项是 org.apache.camel.component.sjms.jms.ConnectionResource 类型。		字符串
camel.component.sjms.connection-test-on-borrow	在使用默认的 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时，在从池返回之前，每个 javax.jms.Connection 应经过测试（调用 start）。	true	布尔值
camel.component.sjms.connection-username	创建 javax.jms.Connection 时使用的用户名，在使用默认的 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时使用。		字符串
camel.component.sjms.destination-creation-strategy	使用自定义 DestinationCreationStrategy。选项是 org.apache.camel.component.sjms.jms.DestinationCreationStrategy 类型。		字符串
camel.component.sjms.enabled	启用 sjms 组件	true	布尔值

Name	描述	默认值	类型
camel.component.sjms.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		字符串
camel.component.sjms.jms-key-format-strategy	可插拔编码和解码 JMS 密钥策略，以便它们能够与 JMS 规范兼容。Camel 提供了开箱即用的一个实施：default.默认策略可以安全地放入点和连字符(. 和 -)。可用于 JMS 代理，其不小心是 JMS 标头键是否包含非法字符。您可以提供自己的 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 # 表示法引用它。选项是 org.apache.camel.component.sjms.jms.JmsKeyFormatStrategy 类型。		字符串
camel.component.sjms.message-created-strategy	要使用给定的 MessageCreatedStrategy，后者在 Camel 在 Camel 发送 JMS 消息时创建 javax.jms.Message 对象的新实例。选项是 org.apache.camel.component.sjms.jms.MessageCreatedStrategy 类型。		字符串
camel.component.sjms.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.sjms.transaction-commit-strategy	要配置要使用的提交策略类型。Camel 开箱即用提供了两个实现，即 default 和 batch。选项是 org.apache.camel.component.sjms.TransactionCommitStrategy 类型。		字符串

以下是如何使用所需的 ConnectionFactory 提供程序配置 SjmsComponent 的示例。默认情况下，它将创建一个连接，并使用组件内部池 API 来存储它，以确保它能够以线程安全的方式服务会话创建请求。

```
SjmsComponent component = new SjmsComponent();
component.setConnectionFactory(new ActiveMQConnectionFactory("tcp://localhost:61616"));
getContext().addComponent("sjms", component);
```

对于支持持久订阅的 SJMS 组件，您可以覆盖默认的 ConnectionFactoryResource 实例并设置 clientId 属性。

```
ConnectionFactoryResource connectionResource = new ConnectionFactoryResource();
```

```
connectionResource.setConnectionFactory(new
ActiveMQConnectionFactory("tcp://localhost:61616"));
connectionResource.setClientId("myclient-id");

SjmsComponent component = new SjmsComponent();
component.setConnectionResource(connectionResource);
component.setMaxConnections(1);
```

307.4. 生成者使用情况

307.4.1. InOnly Producer - （默认）

InOnly producer 是 SJMS Producer Endpoint 的默认行为。

```
from("direct:start")
   .to("sjms:queue:bar");
```

307.4.2. InOut Producer

要启用 InOut 行为，请将 exchangePattern 属性附加到 URI。默认情况下，它将为每个消费者使用专用的 TemporaryQueue。

```
from("direct:start")
   .to("sjms:queue:bar?exchangePattern=InOut");
```

您可以指定一个 namedReplyTo，它可以提供更好的 monitor 点。

```
from("direct:start")
   .to("sjms:queue:bar?exchangePattern=InOut&namedReplyTo=my.reply.to.queue");
```

307.5. 消费者使用情况

307.5.1. InOnly Consumer -(Default)

InOnly consumer 是 SJMS Consumer Endpoint 的默认 Exchange 行为。

```
from("sjms:queue:bar")
   .to("mock:result");
```

307.5.2. InOut Consumer

要启用 InOut 行为，请将 exchangePattern 属性附加到 URI。

```
from("sjms:queue:in.out.test?exchangePattern=InOut")
    .transform(constant("Bye Camel"));
```

307.6. 高级使用备注

307.6.1. 可插入连接资源管理

SJMS 通过内置连接池提供 JMS 连接资源管理。 http://docs.oracle.com/javaee/5/api/javax/jms/Connection.html这消除了依赖第三方 API 池逻辑的需求。然而，您可能需要使用外部连接资源管理器，如 J2EE 或 OSGi 容器提供的资源管理器。对于此 SJMS，提供了一个接口，可用于覆盖内部 SJMS 连接池。这通过 ConnectionResource 接口来完成。

Connection Resource 提供了根据需要编写和返回 Connections 的方法，是用来向 SJMS 组件提供连接池的合同。必要时，用户应使用 SJMS 与外部连接池管理器集成。

虽然对于标准 ConnectionFactory 提供程序，建议您使用 ConnectionFactoryResource 实施（由 SJMS as-is 提供），或将其扩展为此组件。

以下是使用带有 ActiveMQ PooledConnectionFactory 的可插件 ConnectionResource 的示例：

```
public class AMQConnectionResource implements ConnectionResource {
    private PooledConnectionFactory pcf;

    public AMQConnectionResource(String connectString, int maxConnections) {
        super();
        pcf = new PooledConnectionFactory(connectString);
        pcf.setMaxConnections(maxConnections);
        pcf.start();
    }

    public void stop() {
        pcf.stop();
    }

    @Override
    public Connection borrowConnection() throws Exception {
        Connection answer = pcf.createConnection();
        answer.start();
        return answer;
    }

    @Override
```

```
    public Connection borrowConnection(long timeout) throws Exception {
        // SNIPPED...
    }

    @Override
    public void returnConnection(Connection connection) throws Exception {
        // Do nothing since there isn't a way to return a Connection
        // to the instance of PooledConnectionFactory
        log.info("Connection returned");
    }
}
```

然后，将 ConnectionResource 传递给 SjmsComponent ：

```
CamelContext camelContext = new DefaultCamelContext();
AMQConnectionResource pool = new AMQConnectionResource("tcp://localhost:33333", 1);
SjmsComponent component = new SjmsComponent();
component.setConnectionResource(pool);
camelContext.addComponent("sjms", component);
```

要查看其使用的完整示例，请参阅 ConnectionResourceIT。

307.6.2. 批量消息支持

SjmsProducer 支持通过创建封装列表 的交换来发布一系列消息。然后，此 SjmsProducer 会迭代 List 的内容并单独发布每条消息。

如果生成批处理消息，则需要设置每个消息的唯一标头，您可以使用 SJMS BatchMessage 类。当 SjmsProducer 遇到 BatchMessage 列表时，它将迭代每个 BatchMessage，并发布包含的有效负载和标头。

以下是使用 BatchMessage 类的示例。首先，我们创建一个 BatchMessage 列表：

```
List<BatchMessage<String>> messages = new ArrayList<BatchMessage<String>>();
for (int i = 1; i <= messageCount; i++) {
    String body = "Hello World " + i;
    BatchMessage<String> message = new BatchMessage<String>(body, null);
    messages.add(message);
}
```

然后发布列表：

```
template.sendBody("sjms:queue:batch.queue", messages);
```

-

307.6.3. 可自定义的交易提交策略（仅限本地 JMS 事务）

SJMS 为开发人员提供了利用 TransactionCommitStrategy 接口创建自定义和可插入的事务策略的方法。这允许用户定义 SessionTransactionSynchronization 将用来决定何时提交会话的唯一场景。一个用法示例是 BatchTransactionCommitStrategy，其在下一节中会进一步详细介绍。

307.6.4. Transacted Batch Consumers & Producers

SJMS 组件设计为支持在 Producer 和 Consumer 端点上对本地 JMS 事务的批处理。它们的处理方式都截然不同。

SJMS 使用者端点是一种简单的实现，将在提交 X 消息之前处理 X 消息，然后再提交相关会话。要在消费者上启用批处理事务，首先通过将 transacted 参数设置为 true 来启用事务，然后添加 transactionBatchCount 并将其设置为大于 0 的任何值。例如，以下配置将提交 Session 每 10 个信息：

```
sjms:queue:transacted.batch.consumer?transacted=true&transactionBatchCount=10
```

如果在消费者端点处理批处理期间发生异常，则调用 Session rollback，从而导致消息被重新提供给下一个可用的消费者。对于关联的会话，计数器也会重置为 0 (BatchTransactionCommitStrategy)。用户负责确保 hook 在批处理消息的处理器中放置 hook，以监视 JMSRedelivered 标头设置为 true 的消息。这是指示消息在某个时间点上回滚，并且应该验证成功处理。

转用批处理消费者还将其作为内部计时器实例，在消息上提交开放事务前等待默认时间(5000ms)。默认值 5000ms （最小 1000ms）应该足以满足大多数用例，但如果需要进一步调整，只需设置 transactionBatchTimeout 参数。

```
sjms:queue:transacted.batch.consumer?
transacted=true&transactionBatchCount=10&transactionBatchTimeout=2000
```

接受的最小值为 1000ms，因为上下文切换的数量可能会导致不必要的性能影响，而不会获得好处。

生产者端点的处理方式有很大不同。各个消息发送后生成者都会关闭 Exchange，并且不再引用该消息。要使所有消息都可用于重新发送，您只需在发布 BatchMessages 的 Producer 端点上启用事务。事务将在交换的结论下提交，后者包含批处理列表中的所有消息。不需要配置任何其他操作。例如：

```
List<BatchMessage<String>> messages = new ArrayList<BatchMessage<String>>();
for (int i = 1; i <= messageCount; i++) {
    String body = "Hello World " + i;
```

```
    BatchMessage<String> message = new BatchMessage<String>(body, null);
    messages.add(message);
}
```

现在，在启用了事务的情况下发布列表：

```
template.sendBody("sjms:queue:batch.queue?transacted=true", messages);
```

307.7. 其他备注

307.7.1. Message Header Format

SJMS 组件使用与 Camel JMS 组件中使用的相同标头格式策略。此插件策略确保通过线路发送的消息符合 JMS Message spec。

对于 exchange.in.header，以下规则适用于标头键：

- 从 JMS 或 JMSX 开始的密钥被保留。
- exchange.in.headers 键必须是 literals，且所有是有效的 Java 标识符（不要在键名称中使用点）。
- 在消耗 JMS 消息时，Camel 替换了点和连字符以及反向：
  - 在 Camel 使用消息时，由 DOT 和反向替换替换。
  - 在 Camel 使用消息时，由 HYPHEN 替换为 HYPHEN，反向替换。
    另请参阅 jmsKeyFormatStrategy 选项，它允许使用您自己的自定义策略格式化密钥。

对于 exchange.in.header，以下规则适用于标头值：

307.7.2. Message Content

若要通过线线提供内容，我们必须确保传输的消息正文遵循 JMS Message Specification。因此，生成的所有操作都必须是原语或其计数器对象（如 Integer、Long、Character）。type,

String,CharSequence,Date,BigDecimal 和 BigInteger 都转换为它们的 toString （） 表示。所有其他类型都将被丢弃。

307.7.3. 集群

在集群环境中使用 InOut with SJMS 时，您必须使用 TemporaryQueue 目的地，或使用每个 InOut producer 端点的唯一命名回复。消息关联由端点处理，而不是与代理中的消息选择器处理。InOut Producer Endpoint 使用由 Message JMSCorrelationID 缓存的 Java Concurrency Exchangers。这提供了在减少代理的开销时的一个 nice 性能，因为所有消息都会按照感兴趣的消费者产生的顺序使用。

目前，唯一的关联策略是使用 JMSCorrelationId。InOut Consumer 使用这个策略，同时确保对所含的 JMSReplyTo 目的地的所有响应消息也都会从请求中复制了 JMSCorrelationId。

307.8. 事务支持

SJMS 目前仅支持使用内部 JMS 交易。不支持 Camel Transaction Processor 或 Java Transaction API (JTA)。

307.8.1. Springless Mean I Can't use Spring?

根本不可能。以下是使用 Spring DSL 的 SJMS 组件示例：

```
<route
   id="inout.named.reply.to.producer.route">
   <from
      uri="direct:invoke.named.reply.to.queue" />
   <to
      uri="sjms:queue:named.reply.to.queue?
namedReplyTo=my.response.queue&amp;exchangePattern=InOut" />
</route>
```

Springless 指的是从 Spring JMS API 的依赖中移出。新的 JMS 客户端 API 从底层开发到电源 SJMS。

第 308 章 简单的 JMS2 组件

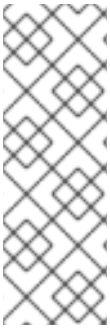
从 Camel 版本 2.19 开始提供

Simple JMS 2.0 组件或 SJMS2 是与 Camel 一起使用的 JMS 客户端，在 JMS 客户端创建和配置时会使用良好最佳实践。SJMS2 包含明确为 Camel 编写的全新的 JMS 2.0 客户端 API，消除了第三方消息传递实施可保持其轻性和弹性。包括以下功能：

- Standard Queue and Topic Support (Durable and Non-Durable)
- InOnly and InOut MEP 支持
- 异步 Producer 和 Consumer Processing
- 内部 JMS 事务支持

其他主要功能包括：

- 可插入连接资源管理
- Session, Consumer, 和 Producer Pooling and caching Management
- 批处理消费者和 Producers
- Transacted Batch Consumers & Producers
- 支持可自定义交易提交策略（仅限本地 JMS 事务）



注意

为什么 S在 SJMS 中

s 代表简单和标准，以及 Springless。另外，camel-jms 已被使用。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-sjms2</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

308.1. URI 格式

```
sjms2:[queue:|topic:]destinationName[?options]
```

其中 destinationName 是 JMS 队列或主题名称。默认情况下，destinationName 解释为队列名称。例如，要连接到队列，使用 FOO.BAR ：

```
sjms2:FOO.BAR
```

如果需要，您可以包含可选的 queue: 前缀：

```
sjms2:queue:FOO.BAR
```

要连接到主题，您必须 包含 topic: 前缀。例如，要连接到主题 Stocks.Prices，请使用：

```
sjms2:topic:Stocks.Prices
```

您可以使用以下格式将查询选项附加到 URI 中，?option=value&option=value&...

308.2. 组件选项和配置

Simple JMS2 组件支持 15 个选项，如下所列。

Name	描述	默认值	类型
ConnectionFactory (advanced)	需要 ConnectionFactory 才能启用 SjmsComponent。它可以直接设置，也可以设置为 ConnectionResource 的一部分。		ConnectionFactory
connectionResource (advanced)	ConnectionResource 是一个接口，允许对 ConnectionFactory 进行自定义和容器控制。如需了解更多详细信息，请参阅 插件连接资源管理。		ConnectionResource
connectionCount (common)	此组件下启动的端点的最大可用连接数	1	整数
jmsKeyFormatStrategy (advanced)	可插拔编码和解码 JMS 密钥策略，以便它们能够与 JMS 规范兼容。Camel 提供了开箱即用的一个实施：default.默认策略可以安全地放入点和连字符(. 和 -)。可用于 JMS 代理，其不小心是 JMS 标头键是否包含非法字符。您可以提供自己的 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 # 表示法引用它。		JmsKeyFormatStrategy
transactionCommit 策略 (transaction)	要配置要使用的提交策略类型。Camel 开箱即用提供了两个实现，即 default 和 batch。		TransactionCommit 策略
destinationCreation Strategy (advanced)	使用自定义 DestinationCreationStrategy。		DestinationCreation 策略
timedTaskManager (advanced)	使用自定义 TimedTaskManager		TimedTaskManager
messageCreatedStrategy (advanced)	要使用给定的 MessageCreatedStrategy，后者在 Camel 在 Camel 发送 JMS 消息时创建 javax.jms.Message 对象的新实例。		MessageCreatedStrategy
connectionTestOnBorrow (advanced)	在使用默认的 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时，在从池返回之前，每个 javax.jms.Connection 应经过测试（调用 start）。	true	布尔值
connectionUsername (security)	创建 javax.jms.Connection 时使用的用户名，在使用默认的 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时使用。		字符串

Name	描述	默认值	类型
connectionPassword (security)	创建 javax.jms.Connection 时使用的密码，在使用默认的 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时使用。		字符串
connectionClientId (advanced)	创建 javax.jms.Connection 时使用的客户端 ID，在使用默认的 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时使用。		字符串
connectionMaxWait (advanced)	当池使用默认 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时，池被耗尽时，等待 millis 的最大等待时间。	5000	long
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Simple JMS2 端点使用 URI 语法进行配置：

```
sjms2:destinationType:destinationName
```

使用以下路径和查询参数：

308.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
destinationType	要使用的目的地类型	queue	字符串
destinationName	**required** DestinationName 是 JMS 队列或主题名称。默认情况下，destinationName 解释为队列名称。		字符串

308.2.2. 查询参数(37 参数)：

Name	描述	默认值	类型
acknowledgementMode (common)	JMS 确认名称，即 SESSION_TRANSACTED、CLIENT_ACKNOWLEDGE、AUTO_ACKNOWLEDGE, DUPS_OK_ACKNOWLEDGE	AUTO_确认	SessionAcknowledgement Type
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
consumerCount (consumer)	设置用于此端点的消费者监听程序数量。	1	int
durable (consumer)	将主题消费者设置为 durable。	false	布尔值
durableSubscriptionId (consumer)	设置持久主题所需的持久订阅 Id。		字符串
shared (consumer)	将消费者设置为 shared。	false	布尔值
subscriptionId (consumer)	设置持久或共享主题所需的订阅 Id。		字符串
同步 (consumer)	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	true	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
messageSelector (consumer)	设置 JMS Message selector 语法。		字符串
namedReplyTo (producer)	将回复设置为用于 InOut producer 端点的目的地名称。		字符串
persistent (producer)	用于启用/禁用消息持久性的标志。	true	布尔值

Name	描述	默认值	类型
producerCount (producer)	设置用于此端点的制作者数量。	1	int
TTL (producer)	用于调整生成消息的 Time To Live 值的标志。	-1	long
allowNullBody (producer)	是否允许发送没有正文的消息。如果此选项为 false，并且消息正文为 null，则抛出 JMSException。	true	布尔值
prefillPool (producer)	是否在启动时预先填充制作者连接池，或者根据需要创建连接 lazy。	true	布尔值
responseTimeOut (producer)	设置在超时 InOut 响应前应等待的时间。	5000	long
asyncStartListener (advanced)	在启动路由时，是否异步启动消费者消息监听程序。例如，如果 JmsConsumer 无法连接到远程 JMS 代理，则在重试和/或故障转移时可能会阻止。这将导致 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 以异步模式使用专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果无法建立连接，那么会记录异常在 WARN 级别，消费者将无法接收消息；然后您可以重启路由来重试。	false	布尔值
asyncStopListener (advanced)	在停止路由时，是否异步停止消费者消息监听程序。	false	布尔值
connectionCount (advanced)	此端点可用的最大连接数		整数
ConnectionFactory (advanced)	初始化端点的 connectionFactory，它优先于组件的 connectionFactory （若有）		ConnectionFactory
connectionResource (advanced)	初始化端点的 connectionResource，它优先于组件的 connectionResource （若有）		ConnectionResource
destinationCreationStrategy (advanced)	使用自定义 DestinationCreationStrategy。		DestinationCreation 策略
exceptionListener (advanced)	指定正在获得任何底层 JMS 异常通知的 JMS Exception Listener。		ExceptionListener
headerFilterStrategy (advanced)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy

Name	描述	默认值	类型
includeAllJMSXProperties (advanced)	从 JMS 映射到 Camel 消息时，是否要包含所有 JMSXxxx 属性。把它设置为 true 将包括 JMSXAppID 和 JMSXUserID 等属性。注： 如果您使用自定义 headerFilterStrategy，则不会应用这个选项。	false	布尔值
jmsKeyFormatStrategy (advanced)	用于编码和解码 JMS 密钥的可插拔策略，以便它们可以符合 JMS 规范。Camel 在开箱即用提供两个实现：default 和 passthrough。默认策略可以安全地放入点和连字符(. 和 -)。passthrough 策略将密钥保留原样。可用于 JMS 代理，其不小心是 JMS 标头键是否包含非法字符。您可以提供自己的 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 # 表示法引用它。		JmsKeyFormatStrategy
mapJmsMessage (advanced)	指定 Camel 是否应该自动将收到的 JMS 消息映射到合适的有效负载类型，如 javax.jms.TextMessage 到 String 等。如需了解更多详细信息，请参阅映射如何工作。	true	布尔值
messageCreatedStrategy (advanced)	要使用给定的 MessageCreatedStrategy，后者在 Camel 在 Camel 发送 JMS 消息时创建 javax.jms.Message 对象的新实例。		MessageCreatedStrategy
errorHandlerLoggingLevel (logging)	允许为日志记录未捕获的异常配置默认的 errorHandler 日志记录级别。	WARN	LoggingLevel
errorHandlerLogStackTrace (logging)	允许由默认的 errorHandler 控制是否应记录堆栈追踪。	true	布尔值
转换 （事务）	指定是否使用转换模式	false	布尔值
transactionBatchCount (transaction)	如果转换在提交事务前将进程的消息数量设置为进程。	-1	int
transactionBatchTimeout (transaction)	为批处理事务设置超时（在 millis 中），该值应为 1000 或更高。	5000	long
transactionCommitStrategy (transaction)	设置提交策略。		TransactionCommit 策略
sharedJMSSession (transaction)	指定是否与其他 SJMS 端点共享 JMS 会话。如果您的路由正在访问多个 JMS 提供程序，则关闭此项。如果您需要对多个 JMS 提供程序的事务，请使用 jms 组件来利用 XA 事务。	true	布尔值

308.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 16 个选项，如下所列。

Name	描述	默认值	类型
camel.component.sjms2.connection-client-id	创建 javax.jms.Connection 时使用的客户端 ID，在使用默认的 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时使用。		字符串
camel.component.sjms2.connection-count	此组件下启动的端点的最大可用连接数	1	整数
camel.component.sjms2.connection-factory	需要 ConnectionFactory 才能启用 SjmsComponent。它可以直接设置，也可以设置为 ConnectionResource 的一部分。选项是 javax.jms.ConnectionFactory 类型。		字符串
camel.component.sjms2.connection-max-wait	当池使用默认 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时，池被耗尽时，等待 millis 的最大等待时间。	5000	Long
camel.component.sjms2.connection-password	创建 javax.jms.Connection 时使用的密码，在使用默认的 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时使用。		字符串
camel.component.sjms2.connection-resource	ConnectionResource 是一个接口，允许对 ConnectionFactory 进行自定义和容器控制。如需了解更多详细信息，请参阅 插件连接资源管理。选项是 org.apache.camel.component.sjms.jms.ConnectionResource 类型。		字符串
camel.component.sjms2.connection-test-on-borrow	在使用默认的 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时，在从池返回之前，每个 javax.jms.Connection 应经过测试（调用 start）。	true	布尔值
camel.component.sjms2.connection-username	创建 javax.jms.Connection 时使用的用户名，在使用默认的 org.apache.camel.component.sjms.jms.ConnectionFactoryResource 时使用。		字符串

Name	描述	默认值	类型
camel.component.sjms2.destination-creation-strategy	使用自定义 DestinationCreationStrategy。选项是 org.apache.camel.component.sjms.jms.DestinationCreationStrategy 类型。		字符串
camel.component.sjms2.enabled	启用 sjms2 组件	true	布尔值
camel.component.sjms2.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		字符串
camel.component.sjms2.jms-key-format-strategy	可插拔编码和解码 JMS 密钥策略，以便它们能够与 JMS 规范兼容。Camel 提供了开箱即用的一个实施：default.默认策略可以安全地放入点和连字符(. 和 -)。可用于 JMS 代理，其不小心是 JMS 标头键是否包含非法字符。您可以提供自己的 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 # 表示法引用它。选项是 org.apache.camel.component.sjms.jms.JmsKeyFormatStrategy 类型。		字符串
camel.component.sjms2.message-created-strategy	要使用给定的 MessageCreatedStrategy，后者在 Camel 在 Camel 发送 JMS 消息时创建 javax.jms.Message 对象的新实例。选项是 org.apache.camel.component.sjms.jms.MessageCreatedStrategy 类型。		字符串
camel.component.sjms2.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.sjms2.timed-task-manager	使用自定义 TimedTaskManager。选项是 org.apache.camel.component.sjms.taskmanager.TimedTaskManager 类型。		字符串
camel.component.sjms2.transaction-commit-strategy	要配置要使用的提交策略类型。Camel 开箱即用提供了两个实现，即 default 和 batch。选项是 org.apache.camel.component.sjms.TransactionCommitStrategy 类型。		字符串

以下是如何将 Sjms2Component 配置为其所需的 ConnectionFactory 提供程序：默认情况下，它将创建一个连接，并使用组件的内部池 API 来存储它，以确保它能够以线程安全的方式服务会话创建请求。

```
Sjms2Component component = new Sjms2Component();
component.setConnectionFactory(new ActiveMQConnectionFactory("tcp://localhost:61616"));
getContext().addComponent("sjms2", component);
```

对于支持持久订阅的 SJMS2 组件，您可以覆盖默认的 ConnectionFactoryResource 实例并设置 clientId 属性。

```
ConnectionFactoryResource connectionResource = new ConnectionFactoryResource();
connectionResource.setConnectionFactory(new
ActiveMQConnectionFactory("tcp://localhost:61616"));
connectionResource.setClientId("myclient-id");

Sjms2Component component = new Sjms2Component();
component.setConnectionResource(connectionResource);
component.setMaxConnections(1);
```

308.4. 生成者使用情况

308.4.1. InOnly Producer - （默认）

InOnly producer 是 SJMS2 Producer 端点的默认行为。

```
from("direct:start")
    .to("sjms2:queue:bar");
```

308.4.2. InOut Producer

要启用 InOut 行为，请将 exchangePattern 属性附加到 URI。默认情况下，它将为每个消费者使用专用的 TemporaryQueue。

```
from("direct:start")
    .to("sjms2:queue:bar?exchangePattern=InOut");
```

您可以指定一个 namedReplyTo，它可以提供更好的 monitor 点。

```
from("direct:start")
    .to("sjms2:queue:bar?exchangePattern=InOut&namedReplyTo=my.reply.to.queue");
```

308.5. 消费者使用情况

308.5.1. 持久化共享订阅

要创建可在一个或多个消费者之间共享的持久化订阅。使用 JMS 2.0 兼容连接工厂并指定通用 subscriptionId。然后，将订阅属性 durable 和 shared 设置为 true。

```
from("sjms2:topic:foo?consumerCount=3&subscriptionId=bar&durable=true&shared=true")
    .to("mock:result");

from("sjms2:topic:foo?consumerCount=2&subscriptionId=bar&durable=true&shared=true")
    .to("mock:result");
```

308.5.2. InOnly Consumer -(Default)

InOnly consumer 是 SJMS2 Consumer Endpoint 的默认 Exchange 行为。

```
from("sjms2:queue:bar")
    .to("mock:result");
```

308.5.3. InOut Consumer

要启用 InOut 行为，请将 exchangePattern 属性附加到 URI。

```
from("sjms2:queue:in.out.test?exchangePattern=InOut")
    .transform(constant("Bye Camel"));
```

308.6. 高级使用备注

308.6.1. 可插入连接资源管理

SJMS2 通过内置连接池提供 JMS 连接资源管理。http://docs.oracle.com/javaee/5/api/javax/jms/Connection.html这消除了依赖第三方 API 池逻辑的需求。然而，您可能需要使用外部连接资源管理器，如 J2EE 或 OSGi 容器提供的资源管理器。对于此 SJMS2，提供了一个接口，可用于覆盖内部 SJMS2 连接池功能。这通过 ConnectionResource 接口来完成。

Connection Resource 提供了根据需要编写和返回 Connections 的方法，是用来向 SJMS2 组件提供连接池的合同。如果需要将 SJMS2 与外部连接池管理器集成，用户应使用它。

虽然对于标准 ConnectionFactory 提供程序，建议您使用 ConnectionFactoryResource 实施，该实施由 SJMS2 提供，或随着此组件进行了优化而扩展。

以下是使用带有 ActiveMQ PooledConnectionFactory 的可插件 ConnectionResource 的示例：

```
public class AMQConnectionResource implements ConnectionResource {
    private PooledConnectionFactory pcf;

    public AMQConnectionResource(String connectString, int maxConnections) {
        super();
        pcf = new PooledConnectionFactory(connectString);
        pcf.setMaxConnections(maxConnections);
        pcf.start();
    }

    public void stop() {
        pcf.stop();
    }

    @Override
    public Connection borrowConnection() throws Exception {
        Connection answer = pcf.createConnection();
        answer.start();
        return answer;
    }

    @Override
    public Connection borrowConnection(long timeout) throws Exception {
        // SNIPPED...
    }

    @Override
    public void returnConnection(Connection connection) throws Exception {
        // Do nothing since there isn't a way to return a Connection
        // to the instance of PooledConnectionFactory
        log.info("Connection returned");
    }
}
```

然后，将 ConnectionResource 传递给 Sjms2Component ：

```
CamelContext camelContext = new DefaultCamelContext();
AMQConnectionResource pool = new AMQConnectionResource("tcp://localhost:33333", 1);
Sjms2Component component = new Sjms2Component();
component.setConnectionResource(pool);
camelContext.addComponent("sjms2", component);
```

要查看其使用的完整示例，请参阅 ConnectionResourceIT。

308.6.2. Session, Consumer, 和 Producer Pooling and caching Management

soon ...

308.6.3. 批量消息支持

Sjms2Producer 支持通过创建封装列表 的 Exchange 发布一系列消息。然后，此 Sjms2Producer 将迭代 List 的内容，并单独发布每个消息。

如果生成批处理消息，则需要设置每个消息的唯一标头，您可以使用 SJMS2 BatchMessage 类。当 Sjms2Producer 遇到 BatchMessage 列表时，它将迭代每个 BatchMessage，并发布包含的有效负载和标头。

以下是使用 BatchMessage 类的示例。首先，我们创建一个 BatchMessage 列表：

```
List<BatchMessage<String>> messages = new ArrayList<BatchMessage<String>>();
for (int i = 1; i <= messageCount; i++) {
    String body = "Hello World " + i;
    BatchMessage<String> message = new BatchMessage<String>(body, null);
    messages.add(message);
}
```

然后发布列表：

```
template.sendBody("sjms2:queue:batch.queue", messages);
```

308.6.4. 可自定义的交易提交策略（仅限本地 JMS 事务）

SJMS2 为开发人员提供了通过使用 TransactionCommitStrategy 接口创建自定义和可插入的事务策略的方法。这允许用户定义 SessionTransactionSynchronization 将用来决定何时提交会话的唯一场景。一个用法示例是 BatchTransactionCommitStrategy，其在下一节中会进一步详细介绍。

308.6.5. Transacted Batch Consumers & Producers

SJMS2 组件设计为支持在 Producer 和 Consumer 端点上对本地 JMS 事务的批处理。它们的处理方式都截然不同。

SJMS2 使用者端点是一种简单的实现，将在提交 X 消息之前进行处理，然后再提交相关会话。要在消费者上启用批处理事务，首先通过将 transacted 参数设置为 true 来启用事务，然后添加

transactionBatchCount 并将其设置为大于 0 的任何值。例如，以下配置将提交 Session 每 10 个信息：

```
sjms2:queue:transacted.batch.consumer?transacted=true&transactionBatchCount=10
```

如果在消费者端点处理批处理期间发生异常，则调用 Session rollback，从而导致消息被重新提供给下一个可用的消费者。对于关联的会话，计数器也会重置为 0 (BatchTransactionCommitStrategy)。用户负责确保 hook 在批处理消息的处理器中放置 hook，以监视 JMSRedelivered 标头设置为 true 的消息。这是指示消息在某个时间点上回滚，并且应该验证成功处理。

转用批处理消费者还将其作为内部计时器实例，在消息上提交开放事务前等待默认时间(5000ms)。默认值 5000ms （最小 1000ms）应该足以满足大多数用例，但如果需要进一步调整，只需设置 transactionBatchTimeout 参数。

```
sjms2:queue:transacted.batch.consumer?
transacted=true&transactionBatchCount=10&transactionBatchTimeout=2000
```

接受的最小值为 1000ms，因为上下文切换的数量可能会导致不必要的性能影响，而不会获得好处。

生产者端点的处理方式有很大不同。各个消息发送后生成者都会关闭 Exchange，并且不再引用该消息。要使所有消息都可用于重新发送，您只需在发布 BatchMessages 的 Producer 端点上启用事务。事务将在交换的结论下提交，后者包含批处理列表中的所有消息。不需要配置任何其他操作。例如：

```
List<BatchMessage<String>> messages = new ArrayList<BatchMessage<String>>();
for (int i = 1; i <= messageCount; i++) {
    String body = "Hello World " + i;
    BatchMessage<String> message = new BatchMessage<String>(body, null);
    messages.add(message);
}
```

现在，在启用了事务的情况下发布列表：

```
template.sendBody("sjms2:queue:batch.queue?transacted=true", messages);
```

308.7. 其他备注

308.7.1. Message Header Format

SJMS2 组件使用与 Camel JMS 组件中使用的相同标头格式策略。此插件策略确保通过线路发送的消息符合 JMS Message spec。

对于 exchange.in.header，以下规则适用于标头键：

- 从 JMS 或 JMSX 开始的密钥被保留。
- exchange.in.headers 键必须是 literals，且所有是有效的 Java 标识符（不要在键名称中使用点）。
- 在消耗 JMS 消息时，Camel 替换了点和连字符以及反向：
  - 在 Camel 使用消息时，由 DOT 和反向替换替换。
  - 在 Camel 使用消息时，由 HYPHEN 替换为 HYPHEN，反向替换。
    另请参阅 jmsKeyFormatStrategy 选项，它允许使用您自己的自定义策略格式化密钥。

对于 exchange.in.header，以下规则适用于标头值：

308.7.2. Message Content

若要通过线线提供内容，我们必须确保传输的消息正文遵循 JMS Message Specification。因此，生成的所有操作都必须是原语或其计数器对象（如 Integer、Long、Character）。type, String,CharSequence,Date,BigDecimal 和 BigInteger 都转换为它们的 toString （） 表示。所有其他类型都将被丢弃。

308.7.3. 集群

在集群环境中使用 InOut 和 SJMS2 时，您必须使用 TemporaryQueue 目的地，或使用每个 InOut producer 端点的唯一命名回复。消息关联由端点处理，而不是与代理中的消息选择器处理。InOut Producer Endpoint 使用由 Message JMSCorrelationID 缓存的 Java Concurrency Exchangers。这提供了在减少代理的开销时的一个 nice 性能，因为所有消息都会按照感兴趣的消费者产生的顺序使用。

目前，唯一的关联策略是使用 JMSCorrelationId。InOut Consumer 使用这个策略，同时确保对所含的 JMSReplyTo 目的地的所有响应消息也都会从请求中复制了 JMSCorrelationId。

308.8. 事务支持

SJMS2 目前仅支持使用内部 JMS 交易。不支持 Camel Transaction Processor 或 Java Transaction API (JTA)。

308.8.1. Springless Mean I Can't use Spring?

根本不可能。以下是使用 Spring DSL 的 SJMS2 组件示例：

```
<route
   id="inout.named.reply.to.producer.route">
   <from
      uri="direct:invoke.named.reply.to.queue" />
   <to
      uri="sjms2:queue:named.reply.to.queue?
namedReplyTo=my.response.queue&amp;amp;exchangePattern=InOut" />
</route>
```

Springless 指的是从 Spring JMS API 的依赖中移出。新的 JMS 客户端 API 从底层开发到电源 SJMS2。

第 309 章 SLACK 组件

从 Camel 版本 2.16 开始提供

slack 组件允许您连接到 Slack 实例，并通过预先建立的 Slack 传入 webhook 提供消息正文中包含的消息。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-slack</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

309.1. URI 格式

要发送消息到频道。

```
slack:#channel[?options]
```

要向 slackuser 发送直接消息，请执行以下操作：

```
slack:@username[?options]
```

309.2. 选项

Slack 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
webhookUrl (common)	传入的 Webhook URL		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Slack 端点使用 URI 语法进行配置：

```
slack:channel
```

使用以下路径和查询参数：

309.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
channel	**需要** 频道名称(syntax #name)或 slackuser (syntax userName)来直接向用户发送消息。		字符串

309.2.2. 查询参数(26 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
maxResults (consumer)	轮询的 Max Result	10	字符串
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
serverUrl (consumer)	Slack 实例的服务器 URL	https://slack.com	字符串
token (consumer)	要使用的令牌		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern

Name	描述	默认值	类型
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
iconEmoji (producer)	使用 Slack emoji 作为 avatar		字符串
iconUrl (producer)	在向频道或用户发送消息时，组件将要使用的 avatar。		字符串
username (producer)	这是在向频道或用户发送消息时 bot 将具有的用户名。		字符串
webhookUrl (producer)	传入的 Webhook URL		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel

Name	描述	默认值	类型
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

309.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.slack.enabled	启用 slack 组件	true	布尔值
camel.component.slack.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.slack.webhook-url	传入的 Webhook URL		字符串

309.4. SLACKCOMPONENT

带有 XML 的 SlackComponent 必须配置为 Spring 或 Blueprint bean，其中包含集成的传入 Webhook url 作为参数。

```
<bean id="slack" class="org.apache.camel.component.slack.SlackComponent">
    <property name="webhookUrl"
value="https://hooks.slack.com/services/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"/>
</bean>
```

对于 Java，您可以使用 Java 代码进行配置。

309.5. 示例

带有 Blueprint 的 CamelContext 可以是：

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0" default-activation="lazy">

    <bean id="slack" class="org.apache.camel.component.slack.SlackComponent">
        <property name="webhookUrl"
value="https://hooks.slack.com/services/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"/>
    </bean>

    <camelContext xmlns="http://camel.apache.org/schema/blueprint">
        <route>
            <from uri="direct:test"/>
            <to uri="slack:#channel?iconEmoji=:camel:&amp;username=CamelTest"/>
        </route>
    </camelContext>

</blueprint>
```

309.6. 消费者

您还可以将消费者用于频道中的信息

```
from("slack://general?token=RAW(<YOUR_TOKEN>)&maxResults=1")
    .to("mock:result");
```

这样，您将从常规频道获取最后一条消息。消费者将跟踪被使用的最后一个消息的时间戳，然后在下一次轮询中，它将从该时间戳中检查。

309.7. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 310 章 SMPP 组件

从 Camel 版本 2.2 开始提供

此组件提供对 SMPP 协议的 SMSC (Short Message Service Center)的访问，以发送和接收 SMS。JSMPP 库用于协议实施。

Camel 组件目前作为 ESME （外部消息实体）运行，而不是作为 SMSC 本身运行。

从*Camel 2.9* 开始，您还可以执行 replaceSm, QuerySm, SubmitMulti, CancelSm 和 DataSm。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-smpp</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

310.1. SMS 限制

SMS 也不可靠或安全。需要可靠和安全交付的用户可能希望考虑使用 XMPP 或 SIP 组件，以及支持所选协议的智能手机应用程序。

- 可靠性：虽然 SMPP 标准提供了一系列反馈机制来指示错误、非发送和确认交付，但移动网络无法隐藏或模拟这些响应。例如，即使目标号无效或未打开，一些网络会自动为每个消息发送发送发送确认。如果某些网络认为是垃圾邮件，则它们会静默丢弃消息。网络中垃圾邮件检测规则可能会非常繁琐，有时来自单个发送方的每天都超过 100 个消息可能会被视为垃圾邮件。

- 安全性：从无线 tower 向下到接收者手动设置的最后一个跃点有基本的加密。SMS 消息在网络的任何其他部分中未加密或验证。某些人员允许零售商专员或致电中心浏览客户的 SMS 消息。可轻松使用消息发送器身份。监管者，甚至移动电话行业本身对在双因素验证方案和其他目的中使用 SMS 非常小心。

虽然 Camel 组件使其尽可能轻松地将消息发送到 SMS 网络，但它无法提供这些问题的简单解决方案。

310.2. 数据编码、字母和国际字符集

数据编码和字母可以根据每个消息指定。可以为端点指定默认值。务必要了解这些选项之间的关系以及设置多个值时组件的行为方式。

数据编码是 SMPP 线格式的 8 位字段。

alphabet 对应于数据编码字段的 0-3 的位。对于某些类型的消息，使用消息类（通过设置数据编码字段的位 5）时，数据编码字段的较低位不会解释为 alphabet，仅位 2 和 3 影响字母。

此外，当前版本的 JSMPP 库仅看似支持位 2 和 3，假设消息类使用 0 和 1。这就是为什么 JSMPP 中的 Alphabet 类不支持值 3 （二进制 0011），这表示 ISO-8859-1。

虽然 JSMPP 提供了消息类参数的表示，但 Camel 组件目前不提供设置它的方式，而不是在数据编码字段中手动设置相应的位。

当在传出消息中设置数据编码字段时，Camel 组件会考虑以下值，并使用它可以找到的第一个值：

- 标头中指定的数据编码
- 在标头中指定的字母
- 端点配置中指定的数据编码(URI 参数)

较旧版本的 Camel 对于国际字符集的支持错误。仅当一个编码用于所有消息时，此功能才可以正常工作，当用户希望根据每个消息更改它时，才会出现问题。需要此工作的用户应该确保其 Camel 版本包括修复

JIRA issues Macro: com.atlassian.sal.api.net.ResponseStatusException: Unexpected response received。状态代码： 404

.

除了尝试向 SMSC 发送数据编码值外，Camel 组件还会尝试分析消息正文，将其转换为 Java 字符串 (Unicode)，并将其转换为对应的字母顺序中的字节数组（决定在字节数组中使用哪个字母，Camel SMPP 组件不考虑数据编码值(header 或 配置)，它只会考虑指定的 alphabet 或端点参数。

如果 String 中的某些字符不能在选定的字母代表，则它们可能被问号(? ?)符号替换。在将其传递给组件之前，API 的用户可能会考虑检查其消息正文是否可转换为 ISO-8859-1，如果不是，将 alphabet 标头设置为请求 UCS-2 编码。如果没有指定 alphabet 和 data 编码选项，则组件可能会尝试检测所需的编码并为您设置数据编码。

alphabet 代码列表在 SMPP 规格 v3.4 部分中指定。19.19。SMPP 规范的一个显著限制是没有 alphabet 代码明确请求使用 GSM 3.38 (7 位)字符集。为 alphabet 选择值 0，选择 SMSC 默认 alphabet，这通常意味着 GSM 3.38，但它不能保证。SMPP 网关 Nexmo 实际上允许默认映射到使用控制面板选项 设置的任何其他字符。建议用户通过其 SMSC operator 检查，以确认将哪个字符集用作默认值。

310.3. 消息分割和节流

在将消息正文从 String 转换为字节数后，Camel 组件还负责将消息拆分为部分（使用 140 字节 SMS 大小限制），然后再将它传递给 JSMPP。这会自动完成。

如果使用 GSM 3.38 alphabet，则组件会将最多 160 个字符打包到 140 字节消息正文中。如果使用 8 位字符集（例如，用于 western Europe 的 ISO-8859-1），则将在 140 字节消息正文中允许 140 个字符。如果使用 16 位 UCS-2 编码，则只有 70 个字符适合每个 140 字节消息。

有些 SMSC 提供程序实施节流规则。已分割消息的每一个部分都可以由提供商的节流机制单独计算。在将 SMPP 路由中的节流发送到 SMSC 前，Camel Throttler 组件非常有用。

310.4. URI 格式

```
smpp://[username@]hostname[:port][?options]
smpps://[username@]hostname[:port][?options]
```

如果没有提供 用户名，则 Camel 将提供默认值 smppclient。
如果没有提供 端口号，则 Camel 将提供默认值 2775。
Camel 2.3： 如果协议名称为 "smpps"，则 camel-smpp 试图使用 SSLSocket 来初始化与服务器的连接。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

310.5. URI 选项

SMPP 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	将共享 SmppConfiguration 用作配置。		SmppConfiguration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

SMPP 端点使用 URI 语法进行配置：

```
smpp:host:port
```

使用以下路径和查询参数：

310.5.1. 路径参数(2 参数)：

Name	描述	默认值	类型
主机	要使用 SMSC 服务器的主机名。	localhost	字符串
port	要使用的 SMSC 服务器的端口号。	2775	整数

310.5.2. 查询参数(38 参数)：

Name	描述	默认值	类型
initialReconnectDelay (common)	在连接丢失后，在 consumer/producer 尝试重新连接到 SMSC 后定义初始延迟（毫秒）。	5000	long
maxReconnect (common)	定义在 SMSC 返回负绑定响应时尝试重新连接到 SMSC 的最大次数	2147483647	int
reconnectDelay (common)	如果与 SMSC 的连接丢失并且之前没有成功，定义重新尝试之间的间隔（毫秒）。	5000	long

Name	描述	默认值	类型
splittingPolicy (common)	您可以为处理长消息指定一个策略：ALLOW - 默认，长消息被分成 140 字节，每个消息 TRUNCATE - 长消息被分割，并将第一个片段发送到 SMSC。有些载体会丢弃随后的片段，因此这可减少 SMPP 连接发送部分的负载，而这些片段永远不会发送。REJECT - 如果需要分割消息，它将被拒绝，并显示 SMPP NegativeResponseException，原因代码表示消息太长。	ALLOW	SmppSplittingPolicy
systemType (common)	这个参数用于对绑定到 SMSC （最大 13 个字符）的 ESME （外部消息实体）的类型进行分类。	cp	字符串
addressRange (consumer)	您可以为 SmppConsumer 指定地址范围，如 SMPP 3.4 规范的 5.2.7 部分中定义。SmppConsumer 仅接收来自此范围内的地址(MSISDN 或 IP 地址)的 SMSC 的消息。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
destAddr (producer)	定义目标 SME 地址。对于移动终止的消息，这是接收者 MS 的目录号。仅适用于 SubmitSm、SubmitMulti、CancelSm 和 DataSm。	1717	字符串
destAddrNpi (producer)	定义在 SME 目标地址参数中使用的数字类型(TON)。仅适用于 SubmitSm、SubmitMulti、CancelSm 和 DataSm。以下 NPI 值被定义：0: Unknown 1: ISDN (E163/E164) 2: Data (X.121) 3: Telex (F.69) 6: Land mobile (E.212) 8: National 9: Private 10: ERMES 13: Internet (IP) 18: WAP Client Id (to be defined by WAPum)		byte

Name	描述	默认值	类型
destAddrTon (producer)	定义在 SME 目标地址参数中使用的数字类型(TON)。仅适用于 SubmitSm、SubmitMulti、CancelSm 和 DataSm。以下 TON 值已定义： 0: Unknown 1: International 2: National 3: National 3: Network Specific 4: Subscriber Number 5: Alphanumeric 6: Abbreviated		byte
lazySessionCreation (producer)	可以在 Camel producer 启动时使用 SMSC，则会话可能会被创建来避免异常。Camel 将检查第一个交换的消息标头 'CamelSmppSystemId' 和 'CamelSmppPassword'。如果存在，Camel 将使用这些数据连接到 SMSC。	false	布尔值
numberingPlanIndicator (producer)	定义要在 SME 中使用的数字计划指示符(NPI)。以下 NPI 值被定义：0: Unknown 1: ISDN (E163/E164) 2: Data (X.121) 3: Telex (F.69) 6: Land mobile (E.212) 8: National 9: Private 10: ERMES 13: Internet (IP) 18: WAP Client Id (to be defined by WAPum)		byte
priorityFlag (producer)	允许原始 SME 将优先级级别分配给简短消息。仅适用于 SubmitSm 和 SubmitMulti。支持四个优先级级别： 0: Level 0 （最低）优先级 1: Level 1 priority 2: Level 2 priority 3: Level 3: Level 3 （最高）优先级		byte
protocolId (producer)	协议 ID		byte
registeredDelivery (producer)	用于请求 SMSC 交付接收和/或 SME 源自确认。定义了以下值： 0: No SMSC delivery receipt.1：SMSC 发送接收请求最终交付结果是成功还是失败。2: SMSC 交付结果请求发送失败。		byte
replaceIfPresentFlag (producer)	用于请求 SMSC 替换之前提交的消息，但仍然处于等待状态的发送。SMSC 将替换源地址、目标地址和服务类型与新消息中的相同字段匹配的现有消息。如果定义了 present 标记值，则以下替换： 0: Don't replace 1: replace		byte
ServiceType (producer)	service type 参数可用于指示与消息关联的 SMS Application 服务。以下通用 service_types 定义了： CMT: Cellular Messaging CPT: Cellular Paging VMN: Voice Mail Notification VMA: Voice Mail Alerting WAP: Wireless Application Protocol USSD: Unstructured Supplementary Services Data	CMT	字符串
sourceAddr (producer)	定义源自此消息的 SME (Short Message Entity)的地址。	1616	字符串

Name	描述	默认值	类型
sourceAddrNpi (producer)	定义在 SME 原始器地址参数中使用的数字计划指示符 (NPI)。以下 NPI 值被定义：0: Unknown 1: ISDN (E163/E164) 2: Data (X.121) 3: Telex (F.69) 6: Land mobile (E.212) 8: National 9: Private 10: ERMES 13: Internet (IP) 18: WAP Client Id (to be defined by WAPum)		byte
sourceAddrTon (producer)	定义在 SME 原始器地址参数中使用的数字类型 (TON)。以下 TON 值已定义： 0: Unknown 1: International 2: National 3: National 3: Network Specific 4: Subscriber Number 5: Alphanumeric 6: Abbreviated		byte
typeOfNumber (producer)	定义要在 SME 中使用的数字类型(TON)。以下 TON 值已定义： 0: Unknown 1: International 2: National 3: National 3: Network Specific 4: Subscriber Number 5: Alphanumeric 6: Abbreviated		byte
enquireLinkTimer (advanced)	定义自信检查之间的间隔（以毫秒为单位）。信任检查用于测试 ESME 和 SMSC 之间的通信路径。	5000	整数
sessionStateListener (advanced)	您可以引用 Registry 中的 org.jsmpp.session.SessionStateListener，以在会话状态更改时接收回调。		SessionStateListener
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
transactionTimer (advanced)	定义在事务后允许的最大不活跃期限，之后 SMPP 实体可能假设会话不再活跃。此计时器可以在通信 SMPP 实体（即 SMSC 或 ESME）上处于活动状态。	10000	整数
alphabet (codec)	根据 SMPP 3.4 规范定义数据的编码，第 5.2.19. 0: SMSC Default Alphabet 4: 8 bit Alphabet 8: UCS2 Alphabet		byte
DataCoding (codec)	根据 SMPP 3.4 规范定义数据编码，第 5.2.19 节。数据编码示例为：0: SMSC Default Alphabet 3:la 1 (ISO-8859-1) 4: Octet unspecified (8-bit binary) 8: UCS2 (ISO/IEC-10646) 13: Extended Kanji JIS (X 0212-1990)		byte
编码 (codec)	定义简短消息用户数据的编码方案。仅适用于 SubmitSm、replaceSm 和 SubmitMulti。	ISO-8859-1	字符串
httpProxyHost (proxy)	如果您需要通过 HTTP 代理隧道 SMPP，请将此属性设置为 HTTP 代理的主机名或 ip 地址。		字符串

Name	描述	默认值	类型
httpProxyPassword (proxy)	如果您的 HTTP 代理需要基本身份验证，请将此属性设置为 HTTP 代理所需的密码。		字符串
httpProxyPort (proxy)	如果您需要通过 HTTP 代理隧道 SMPP，请将此属性设置为 HTTP 代理的端口。	3128	整数
httpProxyUsername (proxy)	如果您的 HTTP 代理需要基本身份验证，请将此属性设置为 HTTP 代理所需的用户名。		字符串
proxyHeaders (proxy)	这些标头将在建立连接时传递给代理服务器。		Map
密码 (security)	连接到 SMSC 服务器的密码。		字符串
systemid (security)	用于连接到 SMSC 服务器的系统 ID （用户名）。	smppclient	字符串
使用SSL (security)	是否使用带有 smpps 协议的 SSL	false	布尔值

310.6. SPRING BOOT AUTO-CONFIGURATION

组件支持 38 选项，如下所列。

Name	描述	默认值	类型
camel.component.smpp.configuration.address-range	您可以为 SmppConsumer 指定地址范围，如 SMPP 3.4 规范的 5.2.7 部分中定义。SmppConsumer 仅接收来自此范围内的地址(MSISDN 或 IP 地址)的 SMSC 的消息。		字符串
camel.component.smpp.configuration.alphabet	根据 SMPP 3.4 规范定义数据的编码，第 5.2.19. 0: SMSC Default Alphabet 4: 8 bit Alphabet 8: UCS2 Alphabet		byte
camel.component.smpp.configuration.data-coding	根据 SMPP 3.4 规范定义数据编码，第 5.2.19 节。数据编码示例为：0: SMSC Default Alphabet 3:la 1 (ISO-8859-1) 4: Octet unspecified (8-bit binary) 8: UCS2 (ISO/IEC-10646) 13: Extended Kanji JIS (X 0212-1990)		byte
camel.component.smpp.configuration.dest-addr	定义目标 SME 地址。对于移动终止的消息，这是接收者 MS 的目录号。仅适用于 SubmitSm、SubmitMulti、CancelSm 和 DataSm。	1717	字符串

Name	描述	默认值	类型
camel.component.smpp.configuration.dest-addr-npi	定义在 SME 目标地址参数中使用的数字类型(TON)。仅适用于 SubmitSm、SubmitMulti、CancelSm 和 DataSm。以下 NPI 值被定义：0: Unknown 1: ISDN (E163/E164) 2: Data (X.121) 3: Telex (F.69) 6: Land mobile (E.212) 8: National 9: Private 10: ERMES 13: Internet (IP) 18: WAP Client Id (to be defined by WAPum)		byte
camel.component.smpp.configuration.dest-addr-ton	定义在 SME 目标地址参数中使用的数字类型(TON)。仅适用于 SubmitSm、SubmitMulti、CancelSm 和 DataSm。以下 TON 值已定义： 0: Unknown 1: International 2: National 3: National 3: Network Specific 4: Subscriber Number 5: Alphanumeric 6: Abbreviated		byte
camel.component.smpp.configuration.encoding	定义简短消息用户数据的编码方案。仅适用于 SubmitSm、replaceSm 和 SubmitMulti。	ISO-8859-1	字符串
camel.component.smpp.configuration.enquire-link-timer	定义自信检查之间的间隔（以毫秒为单位）。信任检查用于测试 ESME 和 SMSC 之间的通信路径。	5000	整数
camel.component.smpp.configuration.host	要使用 SMSC 服务器的主机名。	localhost	字符串
camel.component.smpp.configuration.http-proxy-host	如果您需要通过 HTTP 代理隧道 SMPP，请将此属性设置为 HTTP 代理的主机名或 ip 地址。		字符串
camel.component.smpp.configuration.http-proxy-password	如果您的 HTTP 代理需要基本身份验证，请将此属性设置为 HTTP 代理所需的密码。		字符串
camel.component.smpp.configuration.http-proxy-port	如果您需要通过 HTTP 代理隧道 SMPP，请将此属性设置为 HTTP 代理的端口。	3128	整数
camel.component.smpp.configuration.http-proxy-username	如果您的 HTTP 代理需要基本身份验证，请将此属性设置为 HTTP 代理所需的用户名。		字符串

Name	描述	默认值	类型
camel.component.smpp.configuration.initial-reconnect-delay	在连接丢失后，在 consumer/producer 尝试重新连接到 SMSC 后定义初始延迟（毫秒）。	5000	Long
camel.component.smpp.configuration.lazy-session-creation	可以在 Camel producer 启动时使用 SMSC，则会话可能会被创建来避免异常。Camel 将检查第一个交换的消息标头 'CamelSmppSystemId' 和 'CamelSmppPassword'。如果存在，Camel 将使用这些数据连接到 SMSC。	false	布尔值
camel.component.smpp.configuration.max-reconnect	定义在 SMSC 返回负绑定响应时尝试重新连接到 SMSC 的最大次数	2147483647	整数
camel.component.smpp.configuration.numbering-plan-indicator	定义要在 SME 中使用的数字计划指示符(NPI)。以下 NPI 值被定义：0: Unknown 1: ISDN (E163/E164) 2: Data (X.121) 3: Telex (F.69) 6: Land mobile (E.212) 8: National 9: Private 10: ERMES 13: Internet (IP) 18: WAP Client Id (to be defined by WAPum)		byte
camel.component.smpp.configuration.password	连接到 SMSC 服务器的密码。		字符串
camel.component.smpp.configuration.port	要使用的 SMSC 服务器的端口号。	2775	整数
camel.component.smpp.configuration.priority-flag	允许原始 SME 将优先级级别分配给简短消息。仅适用于 SubmitSm 和 SubmitMulti。支持四个优先级级别：0: Level 0 （最低）优先级 1: Level 1 priority 2: Level 2 priority 3: Level 3: Level 3 （最高）优先级		byte
camel.component.smpp.configuration.protocol-id	协议 ID		byte
camel.component.smpp.configuration.proxy-headers	这些标头将在建立连接时传递给代理服务器。		Map
camel.component.smpp.configuration.reconnect-delay	如果与 SMSC 的连接丢失并且之前没有成功，定义重新尝试之间的间隔（毫秒）。	5000	Long

Name	描述	默认值	类型
camel.component.smpp.configuration.registered-delivery	用于请求 SMSC 交付接收和/或 SME 源自确认。定义了以下值： 0: No SMSC delivery receipt.1：SMSC 发送接收请求最终交付结果是成功还是失败。2: SMSC 交付结果请求发送失败。		byte
camel.component.smpp.configuration.replace-if-present-flag	用于请求 SMSC 替换之前提交的消息，但仍然处于等待状态的发送。SMSC 将替换源地址、目标地址和服务类型与新消息中的相同字段匹配的现有消息。如果定义了 present 标记值，则以下替换： 0: Don't replace 1: replace		byte
camel.component.smpp.configuration.service-type	service type 参数可用于指示与消息关联的 SMS Application 服务。以下通用 service_types 定义了：CMT: Cellular Messaging CPT: Cellular Paging VMN: Voice Mail Notification VMA: Voice Mail Alerting WAP: Wireless Application Protocol USSD: Unstructured Supplementary Services Data	CMT	字符串
camel.component.smpp.configuration.session-state-listener	您可以引用 Registry 中的 org.jsmpp.session.SessionStateListener，以在会话状态更改时接收回调。		SessionStateListener
camel.component.smpp.configuration.source-addr	定义源自此消息的 SME (Short Message Entity)的地址。	1616	字符串
camel.component.smpp.configuration.source-addr-npi	定义在 SME 原始器地址参数中使用的数字计划指示符 (NPI)。以下 NPI 值被定义：0: Unknown 1: ISDN (E163/E164) 2: Data (X.121) 3: Telex (F.69) 6: Land mobile (E.212) 8: National 9: Private 10: ERMES 13: Internet (IP) 18: WAP Client Id (to be defined by WAPum)		byte
camel.component.smpp.configuration.source-addr-ton	定义在 SME 原始器地址参数中使用的数字类型 (TON)。以下 TON 值已定义： 0: Unknown 1: International 2: National 3: National 3: Network Specific 4: Subscriber Number 5: Alphanumeric 6: Abbreviated		byte
camel.component.smpp.configuration.splitting-policy	您可以为处理长消息指定一个策略：ALLOW - 默认，长消息被分成 140 字节，每个消息 TRUNCATE - 长消息被分割，并将第一个片段发送到 SMSC。有些载体会丢弃随后的片段，因此这可减少 SMPP 连接发送部分的负载，而这些片段永远不会发送。REJECT - 如果需要分割消息，它将被拒绝，并显示 SMPP NegativeResponseException，原因代码表示消息太长。		SmppSplittingPolicy

Name	描述	默认值	类型
camel.component.smpp.configuration.system-id	用于连接到 SMSC 服务器的系统 ID （用户名）。	smppclient	字符串
camel.component.smpp.configuration.system-type	这个参数用于对绑定到 SMSC （最大 13 个字符）的 ESME （外部消息实体）的类型进行分类。	cp	字符串
camel.component.smpp.configuration.transaction-timer	定义在事务后允许的最大不活跃期限，之后 SMPP 实体可能假设会话不再活跃。此计时器可以在通信 SMPP 实体（即 SMSC 或 ESME）上处于活动状态。	10000	整数
camel.component.smpp.configuration.type-of-number	定义要在 SME 中使用的数字类型(TON)。以下 TON 值已定义： 0: Unknown 1: International 2: National 3: National 3: Network Specific 4: Subscriber Number 5: Alphanumeric 6: Abbreviated		byte
camel.component.smpp.configuration.using-s-s-l	是否使用带有 smpps 协议的 SSL	false	布尔值
camel.component.smpp.enabled	启用 smpp 组件	true	布尔值
camel.component.smpp.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

您可以根据需要拥有任意数量的选项。

> ***smpp://smppclient@localhost:2775?password=password&enquireLinkTimer=3000&transactionTimer=5000&systemType=consumer***

310.7. 制作者消息标头

以下消息标头可用于影响 SMPP producer 的行为

标头	类型	描述
CamelSmppDestAddr	**列出/字符串**	**仅适用于 SubmitSm、SubmitMulti、CancelSm 和 DataSm** 定义目的地 SME 地址。对于移动终止的消息，这是接收者 MS 的目录号。必须是 SubmitMulti 的 **List< String >**，否则是一个字符串。
CamelSmppDestAddrTon	**byte**	**仅适用于 SubmitSm、SubmitMulti、CancelSm 和 DataSm** 定义要在 SME 目标地址参数中使用的数字类型(TON)。使用上面定义的 **sourceAddrTon** URI 选项值。
CamelSmppDestAddrNpi	**byte**	**仅适用于 SubmitSm、SubmitMulti、CancelSm 和 DataSm** 定义要在 SME 目标地址参数中使用的数字计划指示符(NPI)。使用上面定义的 URI 选项 **sourceAddrNpi** 值。
CamelSmppSourceAddr	**字符串**	定义源自此消息的 SME (Short Message Entity)的地址。
CamelSmppSourceAddrTon	**byte**	定义在 SME 原始器地址参数中使用的数字类型(TON)。使用上面定义的 **sourceAddrTon** URI 选项值。
CamelSmppSourceAddrNpi	**byte**	定义在 SME 原始器地址参数中使用的数字计划指示符(NPI)。使用上面定义的 URI 选项 **sourceAddrNpi** 值。
CamelSmppServiceType	**字符串**	service type 参数可用于指示与消息关联的 SMS Application 服务。使用上面的 URI 选项 **serviceType** 设置。
CamelSmppRegisteredDelivery	**byte**	**仅适用于 SubmitSm、replaceSm、SubmitMulti 和 DataSm** ls 用于请求 SMSC 交付接收和/或 SME 源自确认。使用上述 URI 选项 **registeredDelivery** 设置。

标头	类型	描述
CamelSmppPriorityFlag	**byte**	**仅适用于 SubmitSm 和 SubmitMulti**，允许原始 SME 将优先级级别分配给简短消息。使用上面的 URI 选项 **priorityFlag** 设置。
CamelSmppScheduleDeliveryTime	**Date**	**仅针对 SubmitSm、SubmitMulti 和 replaceSm 替换** 此参数指定应首先尝试消息交付的调度时间。它定义了来自当前 SMSC 时间的绝对日期和时间，或 SMSC 将尝试发送此消息的时间。它可以采用绝对时间格式或相对时间格式指定。时间格式的编码在 smpp 规格 v3.4 的章节 7.1.1. 中指定。
CamelSmppValidityPeriod	**字符串/日期**	**仅适用于 SubmitSm、SubmitMulti 和 replaceSm** The validity period 参数，表示 SMSC 过期时间，之后当未传送到目的地时，应丢弃消息。如果它作为 **Date** 提供，它将被解释为绝对时间。**Camel 2.9.1 以后：** 如果您以绝对时间格式或相对时间格式定义，则可以将其指定为 **String**，如 smpp 规格 v3.4 的章节 7.1.1 中指定的字符串。
CamelSmppReplaceIfPresentFlag	**byte**	**只适用于 SubmitSm 和 SubmitMulti**。如果 present 标志参数用于请求 SMSC 来替换之前提交的消息，仍然处于等待的发送。SMSC 将替换源地址、目标地址和服务类型与新消息中的相同字段匹配的现有消息。定义了以下值：**0**、Don't replace 和 **1**，replace
CamelSmppAlphabet/CamelSmppDataCoding	**byte**	**Camel 2.5 for SubmitSm, SubmitMulti 和 replaceSm** (Prior to **Camel 2.9** 使用 **CamelSmppDataCoding** 而不是 **CamelSmppAlphabet**。)根据 SMPP 3.4 规范部分 5.2.19 的数据编码。使用上面的 URI 选项 **alphabet** 设置。
CamelSmppOptionalParameters	**Map<String, String>**	**弃用并将在 Camel 2.13.0/3.0.0** **Camel 2.10.5 和 2.11.1 开始删除，且仅适用于 SubmitSm、SubmitMulti 和 DataSm**The 可选参数由 SMSC 发回。

标头	类型	描述
CamelSmppOptionalParameter	**map<Short, Object>**	Camel 2.10.7 和 2.11.2 开始，仅适用于 SubmitSm、SubmitMulti 和 DataSm，可选参数发送到 SMSC。该值以以下方式转换： **String → org.jsmpp.bean.OptionalParameter.COctet String, byte [] → org.jsmpp.bean.OptionalParameter.Octet String, Byte → org.jsmpp.bean.OptionalParameter. Byte, Integer → org.jsmpp.bean.OptionalParameter. Int, Short → org.jsmpp.bean.OptionalParameter. Short, null → org.jsmpp.bean.OptionalParameter.Null**
CamelSmppEncoding	字符串	**Camel 2.14.1 和 Camel 2.15.0 以后**为 SubmitSm、SubmitMulti 和 DataSm*。指定消息正文中字节的编码（字符集名称）。如果消息正文是字符串，则这不是相关的，因为 Java 字符串始终是 Unicode。如果正文是一个字节数组，则可以使用此标头来指示它是 ISO-8859-1 或某些其他值。默认值由端点配置参数 *编码*指定
CamelSmppSplittingPolicy	字符串	**Camel 2.14.1 和 Camel 2.15.0 以后**为 SubmitSm、SubmitMulti 和 DataSm*。指定用于此交换的消息分割的策略。端点配置参数 *splittingPolicy*中描述了可能的值

SMPP producer 使用以下消息标头来设置消息标头中的 SMSC 的响应

标头	类型	描述
CamelSmppId	**List<String>; /String**	用于标识提交的简短消息的 id 以供以后使用。**来自 Camel 2.9.0** ：如果是 replaceSm, QuerySm, CancelSm and DataSm this header vaule 是 **String**。如果是 SubmitSm 或 SubmitMultiSm，这个标头 vaule 是一个 **List<String>**。
CamelSmppSentMessageCount	**整数**	**从 Camel 2.9 开始，仅适用于 SubmitSm 和 SubmitMultiSm，**即已发送的消息总数。

标头	类型	描述
CamelSmppError	**Map<String, List<Map<String, Object>>**	**从 Camel 2.9 开始只针对 SubmitMultiSm，**即发送简短消息，格式为 **Map<String, List<Map<String, Object>>**> （messageID : (destAddr : address, error : error : error)。
CamelSmppOptionalParameters	**Map<String, String>**	**弃用并将在 Camel 2.13.0/3.0.0 中删除，仅针对 DataSm 使用从** SMSC 返回的可选参数，通过发送消息来从 SMSC 返回。
CamelSmppOptionalParameter	**map<Short, Object>**	**从 Camel 2.10.7 中，仅针对 DataSm 提供 2.11.2，**通过发送消息从 SMSC 返回的参数。key 是可选参数的 **Short** 代码。该值以以下方式转换：**org.jsmpp.bean.OptionalParameter.COctetString → String**,**org.jsmpp.bean.OptionalParameter.OctetString → byte[]**, **org.jsmpp.bean.OptionalParameter.Byte → Byte**,**org.jsmpp.bean.OptionalParameter.Int → Integer**,**org.jsmpp.bean.OptionalParameter.Short** → Short, **org.jsmpp.bean.OptionalParameter.** Null → null

310.8. 消费者消息标头

SMPP 使用者使用以下消息标头，在消息标头中设置来自 SMSC 的请求数据

标头	类型	描述
CamelSmppSequenceNumber	**整数**	**仅适用于 AlertNotification，ElivementSm 和 DataSm** A 序列号允许响应 PDU 与请求 PDU 关联。关联的 SMPP 响应 PDU 必须保留此字段。
CamelSmppCommandId	**整数**	**只适用于 AlertNotification, deliveringSm and DataSm** The command id 项用于标识特定的 SMPP PDU。有关定义值的完整列表，请参阅 smpp 规格 v3.4 中的 5.1.2.1 章节。

标头	类型	描述
CamelSmppSourceAddr	**字符串**	**仅适用于 AlertNotification，ElivementSm 和 DataSm** 定义源自此消息的 SME 的地址(Short Message Entity)。
CamelSmppSourceAddrNpi	**byte**	**只适用于 AlertNotification 和 DataSm** 定义要在 SME 原始卷地址参数中使用的数字计划指示符(NPI)。使用上面定义的 URI 选项 **sourceAddrNpi** 值。
CamelSmppSourceAddrTon	**byte**	**只适用于 AlertNotification，DataSm** 定义要在 SME 原始卷地址参数中使用的数字类型(TON)。使用上面定义的 **sourceAddrTon** URI 选项值。
CamelSmppEsmeAddr	**字符串**	**仅针对 AlertNotification** 定义目标 ESME 地址。对于移动终止的消息，这是接收者 MS 的目录号。
CamelSmppEsmeAddrNpi	**byte**	**只适用于 AlertNotification** 定义在 ESME 原始器地址参数中使用的数字计划指示符(NPI)。使用上面定义的 URI 选项 **sourceAddrNpi** 值。
CamelSmppEsmeAddrTon	**byte**	**只适用于 AlertNotification** 定义 ESME 原始卷地址参数中使用的数字类型(TON)。使用上面定义的 **sourceAddrTon** URI 选项值。
CamelSmppId	**字符串**	**只适用于 smsc DeliveryReceipt 和 DataSm** The message ID 在最初提交时由 SMSC 分配给消息。
CamelSmppDelivered	**整数**	**只适用于 smsc DeliveryReceipt** Number of short 信息。这仅与原始消息提交至分发列表的位置相关。如果需要，该值会添加前导零。
CamelSmppDoneDate	**Date**	**只适用于 smsc DeliveryReceipt** The time 和 date，其中短消息达到它的最终状态。格式如下：YYMMDDhhmm。

标头	类型	描述
CamelSmppStatus	**DeliveryReceiptState**	**只适用于 smsc DeliveryReceipt ：** 消息的最终状态。定义了以下值： **DELIVR** D : Message 传送到目的地，EXPIRED： Message validity period 已过期，DELETED : **Message 已删除，** UNDELIV： Message is **unlive** rable, ACCEPTD : Message 处于 accepted 状态（例如，**客户服务** 代表订阅者手动读取），UNKNOWN **：** Message 处于无效状态，REJECTD ： Message 处于被拒绝状态
CamelSmppCommandStatus	**整数**	**只适用于 DataSm** The Command status。
CamelSmppError	**字符串**	**只适用于 smsc DeliveryReceipt** 可能保存网络特定错误代码或 SMSC 错误代码用于发送消息的尝试。这些错误是特定于 Network 或 SMSC 的，不包含在其中。
CamelSmppSubmitDate	**Date**	**只适用于 smsc DeliveryReceipt** The time and date that the short 信息被提交。如果是已替换的消息，这是替换原始消息的日期。格式如下：YYMMDDhhmm。
CamelSmppSubmitted	**整数**	**只适用于最初提交** 的短消息的编号。这只有在将原始消息提交到发行版列表时才相关。如果需要，该值会添加前导零。
CamelSmppDestAddr	**字符串**	**仅适用于 deliveringSm 和 DataSm：** 定义目标 SME 地址。对于移动终止的消息，这是接收者 MS 的目录号。
CamelSmppScheduleDeliveryTime	**字符串**	**仅限交付Sm：** 此参数指定应首先尝试消息交付的计划时间。它定义了来自当前 SMSC 时间的绝对日期和时间，或 SMSC 将尝试发送此消息的时间。它可以采用绝对时间格式或相对时间格式指定。时间格式的编码在 smpp 规格 v3.4 的第 7.1.1. 节中指定。
CamelSmppValidityPeriod	**字符串**	**仅对 DeliverSm** The validity period 参数表示 SMSC 过期时间，之后当未传送到目的地时，应丢弃消息。它可以以绝对时间格式或相对时间格式定义。绝对和相对时间格式的编码在 smpp 规格 v3.4 的第 7.1.1 中指定。

标头	类型	描述
CamelSmppServiceType	**字符串**	**仅适用于 deliveringSm 和 DataSm** The service type 参数，指示与消息关联的 SMS 应用程序服务。
CamelSmppRegisteredDelivery	**byte**	仅针对 **DataSm** Is 用于请求发送接收和/或 SME 源自确认。与以上 Producer 标头列表中的值相同。
CamelSmppDestAddrNpi	**byte**	**只适用于 DataSm** 定义目标地址参数中的数字计划指示符(NPI)。使用上面定义的 URI 选项 **sourceAddrNpi** 值。
CamelSmppDestAddrTon	**byte**	仅针对 **DataSm** 定义目标地址参数中的数字类型(TON)。使用上面定义的 **sourceAddrTon** URI 选项值。
CamelSmppMessageType	**字符串**	**Camel 2.6 之后**: 标识传入消息的类型： **AlertNotification**: an SMSC 警报通知, **DataSm**: a SMSC data short message,**DeliveryReceipt**: an SMSC delivery receipt, deliverySm : A SMSC deliver **Sm**: a SMSC data short message
CamelSmppOptionalParameters	**Map<String, Object>**	**弃用并将在 Camel 2.13.0/3.0.0Camel 2.10.5 以后删除，仅适用于** SMSC 发回的可选参数。
CamelSmppOptionalParameter	**map<Short, Object>**	**Camel 2.10.7, 2.11.2 开始，仅适用于** SMSC 发回的可选参数。key 是可选参数的**Short** 代码。该值以以下方式转换： **org.jsmpp.bean.OptionalParameter.COctetString → String**,**org.jsmpp.bean.OptionalParameter.OctetString → byte[]**, **org.jsmpp.bean.OptionalParameter.Byte → Byte**,**org.jsmpp.bean.OptionalParameter.Int → Integer**,**org.jsmpp.bean.OptionalParameter.Short** → Short, **org.jsmpp.bean.OptionalParameter.** Null → null

提示

JSMPP 库 请查看 JSMPP 库 的文档以获取有关底层库的更多详情。

310.9. 异常处理

此组件支持常规 Camel 异常处理功能

当出现错误时，会发送带有 SubmitSm （默认操作）的消息，org.apache.camel.component.smpp.SmppException 会抛出带有嵌套异常 org.jsmpp.extra.NegativeResponseException。调用 NegativeResponseException.getCommandStatus （）以获取确切的 SMPP 负响应代码，这些值在 SMPP 规格 3.4 节中解释。
Camel 2.8 以后 ：当 SMPP 使用者收到 deliveringSm 或 DataSm 短消息时，您可以抛出一个 ProcessRequestException，而不是处理失败。在这种情况下，这个例外被转发到底层 JSMPP 库，该库会将包含的错误代码返回到 SMSC。此功能对于例如指示 SMSC 稍后重新发送简短消息非常有用。这可以通过以下代码行完成：

```
from("smpp://smppclient@localhost:2775?
password=password&enquireLinkTimer=3000&transactionTimer=5000&systemType=consum
er")
  .doTry()
    .to("bean:dao?method=updateSmsState")
  .doCatch(Exception.class)
    .throwException(new ProcessRequestException("update of sms state failed", 100))
  .end();
```

有关错误代码及其含义的完整列表，请参阅 SMPP 规格。

310.10. SAMPLES

使用 Java DSL 发送 SMS 的路由：

```
from("direct:start")
  .to("smpp://smppclient@localhost:2775?

password=password&enquireLinkTimer=3000&transactionTimer=5000&systemType=producer
");
```

使用 Spring XML DSL 发送 SMS 的路由：

```
<route>
```

```
  <from uri="direct:start"/>
  <to uri="smpp://smppclient@localhost:2775?

password=password&amp;enquireLinkTimer=3000&amp;transactionTimer=5000&amp;systemType=prc
ducer"/>
</route>
```

使用 Java DSL 接收 SMS 的路由：

```
from("smpp://smppclient@localhost:2775?
password=password&enquireLinkTimer=3000&transactionTimer=5000&systemType=consum
er")
  .to("bean:foo");
```

使用 Spring XML DSL 接收 SMS 的路由：

```
  <route>
    <from uri="smpp://smppclient@localhost:2775?

password=password&amp;enquireLinkTimer=3000&amp;transactionTimer=5000&amp;systemType=co
nsumer"/>
    <to uri="bean:foo"/>
  </route>
```

提示

如果需要 测试的 SMSC 模拟器，则 SMSC 模拟器可以使用 Logica 提供的 simulator。

310.11. 调试日志记录

此组件具有日志级别 DEBUG，这在调试问题时非常有用。如果使用 log4j，您可以在配置中添加以下行：

```
log4j.logger.org.apache.camel.component.smpp=DEBUG
```

310.12. 另请参阅

- 配置 Camel
- 组件

- *端点*
- *开始使用*

第 311 章 SNMP 组件

从 Camel 版本 2.1 开始提供

snmp: 组件可让您轮询 SNMP 功能的设备或接收陷阱

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-snmp</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

311.1. URI 格式

```
snmp://hostname[:port][?Options]
```

组件支持从 SNMP 启用的设备和接收陷阱轮询 OID 值。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

311.2. SNMP PRODUCER

2.18 发行版提供

它还可用于使用 GET 方法请求信息。

响应正文类型是 org.apache.camel.component.snmp.SnmpMessage

311.3. 选项

SNMP 组件没有选项。

SNMP 端点使用 URI 语法进行配置：

```
snmp:host:port
```

使用以下路径和查询参数：

311.3.1. 路径参数(2 参数)：

Name	描述	默认值	类型
主机	SNMP 启用的设备的主机名		字符串
port	SNMP 启用的设备 **所需的** 端口号		整数

311.3.2. 查询参数(35 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
delay (consumer)	设置更新率（以秒为单位）	60000	long
OIDs (consumer)	定义您感兴趣的值。请仔细查看维基百科以更好地了解。您可以提供一个 OID 或一个以 OID 分开的 OID 列表。示例：oids=1.3.6.1.2.1.1.3.0,1.3.6.1.2.1.25.3.2.1.5.1,1.3.6.1.3.5.1.1.1,1.3.6.1.2.1.43.5.1.1.11.1		字符串
protocol (consumer)	您可以在此处选择要使用的协议。您可以使用 udp 或 tcp。	udp	字符串
retries (consumer)	定义在取消请求前重试的频率。	2	int
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值

Name	描述	默认值	类型
snmpCommunity (consumer)	为 snmp 请求设置 community octet 字符串。	public	字符串
snmpContextEngineId (consumer)	设置范围 PDU 的上下文引擎 ID 字段。		字符串
snmpContextName (consumer)	设置此范围 PDU 的 context name 字段。		字符串
snmpVersion (consumer)	为请求设置 snmp 版本。值 0 表示 SNMPv1，1 表示 SNMPv2c，值 3 表示 SNMPv3	0	int
timeout (consumer)	为 millis 设置请求的超时值。	1500	int
treeList (consumer)	如果其树中有子元素，则设置范围 PDU 是否显示为列表	false	布尔值
type (consumer)	要执行哪个操作，如 poll、trap 等。		SnmpActionType
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int

Name	描述	默认值	类型
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
AuthenticationPassphrase (security)	身份验证密码短语。如果没有为空，则 authenticationProtocol 也不能为空。RFC3414 11.2 要求密码短语最小长度为 8 字节。如果验证Passphrase 的长度小于 8 字节，则会抛出 IllegalArgumentException。		字符串
authenticationProtocol (security)	如果将安全级别设置为启用身份验证，则使用身份验证协议： MD5、SHA1		字符串
隐私密码 （安全）	隐私密码短语。如果没有为空，则隐私协议也不能为空。RFC3414 11.2 要求密码短语最小长度为 8 字节。如果验证Passphrase 的长度小于 8 字节，则会抛出 IllegalArgumentException。		字符串
隐私协议 (security)	与此用户关联的隐私协议 ID。如果设置为 null，则此用户只支持未加密的信息。		字符串

Name	描述	默认值	类型
安全级别(security)	设置此目标的安全级别。提供的安全级别必须受到与为此目标设置的安全名称关联的安全模型依赖信息的支持。值 1 表示：没有身份验证，没有加密。任何人都可以创建并读取具有此安全级别的消息。值 2 表示：身份验证和无加密。只有具有正确身份验证密钥的用户才能创建具有此安全级别的消息，但任何人都可以读取消息的内容。值 3 表示：身份验证和加密。只有具有正确身份验证密钥的用户才能创建具有此安全级别的消息，并且只有具有正确加密/解密密钥的消息才能读取消息的内容。	3	int
securityName (security)	设置用于此目标的安全名称。		字符串

311.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component .snmp.enabled	启用 snmp 组件	true	布尔值
camel.component .snmp.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

311.5. 轮询的结果

在这种情况下，我轮询以下 OID：

OID

```
1.3.6.1.2.1.1.3.0
1.3.6.1.2.1.25.3.2.1.5.1
1.3.6.1.2.1.25.3.5.1.1.1
1.3.6.1.2.1.43.5.1.1.11.1
```

结果将是以下内容：

toString 转换的结果

```
<?xml version="1.0" encoding="UTF-8"?>
<snmp>
  <entry>
    <oid>1.3.6.1.2.1.1.3.0</oid>
    <value>6 days, 21:14:28.00</value>
  </entry>
  <entry>
    <oid>1.3.6.1.2.1.25.3.2.1.5.1</oid>
    <value>2</value>
  </entry>
  <entry>
    <oid>1.3.6.1.2.1.25.3.5.1.1.1</oid>
    <value>3</value>
  </entry>
  <entry>
    <oid>1.3.6.1.2.1.43.5.1.1.11.1</oid>
    <value>6</value>
  </entry>
  <entry>
    <oid>1.3.6.1.2.1.1.1.0</oid>
    <value>My Very Special Printer Of Brand Unknown</value>
  </entry>
</snmp>
```

因为您可能会发现，结果比 requested....1.3.6.1.2.1.1.1.0 有多个结果。在这个特殊情况下，该设备会自动填写这个设备。因此，可能会完全发生，您会收到超过您请求的...be 准备。

OID 以点表示开始

```
.1.3.6.1.4.1.6527.3.1.2.21.2.1.50
```

您可能注意到，默认的 snmpVersion 为 0，这意味着端点中的 version1 （如果未明确设置）。在可以使用不同版本查询 SNMP 表时，请确保明确设置了不是默认值的 snmpVersion。其他可能的值有 version2c 和 version3。

311.6. 例子

轮询远程设备：

```
snmp:192.168.178.23:161?protocol=udp&type=POLL&oids=1.3.6.1.2.1.1.5.0
```

设置陷阱接收器(请注意，此处不需要 OID 信息！)：

```
snmp:127.0.0.1:162?protocol=udp&type=TRAP
```

在 Camel 2.10.0 中，您可以使用消息标头 'securityName' 获取 SNMP TRAP 社区，SNMP TRAP 的对等地址，并带有消息标头 'peerAddress'。

Java 中的路由示例：（将 SNMP PDU 转换为 XML 字符串）

```
from("snmp:192.168.178.23:161?protocol=udp&type=POLL&oids=1.3.6.1.2.1.1.5.0").
convertBodyTo(String.class).
to("activemq:snmp.states");
```

311.7. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 312 章 SOAP DATAFORMAT

从 Camel 版本 2.3 开始提供

SOAP 是一种数据格式，使用 JAXB2 和 JAX-WS 注释来整理和 unmarshal SOAP 载荷。它提供了 Apache CXF 的基本功能，而无需 CXF 堆栈。

支持的 SOAP 版本

默认情况下支持 SOAP 1.1。从 Camel 2.11 开始支持 SOAP 1.2。

命名空间前缀映射

如需了解在使用 SOAP 数据格式的 marshalling 时如何控制命名空间前缀映射的详情，请参阅 JAXB。

312.1. SOAP 选项

SOAP dataformat 支持 7 个选项，如下所列。

Name	默认值	Java 类型	描述
contextPath		字符串	您的 JAXB 类所在的软件包名称。
编码		字符串	overrule 并使用特定的编码

Name	默认值	Java 类型	描述
elementNameStrategyRef		字符串	引用 registry 中要查找的元素策略。元素名称策略用于两个目的。第一个方法是查找给定对象的 xml 元素名称，并在将对象放入 SOAP 消息时进行 soap 操作。第二种是查找给定 soap 错误名称的 Exception 类。以下三元素策略类名称开箱即用。QNameStrategy - 使用在实例化时配置的固定 qName。不支持异常查找 - 使用给定类型的 XMLType 注解中的名称和命名空间。如果没有设置命名空间，则使用 package-info。不支持 ServiceInterfaceStrategy - 使用 webservice 接口的信息来确定类型名称，并查找 SOAP 错误的异常类，所有三种类都位于软件包名称 org.apache.camel.dataformat.soap.dataformat.soap.name 中，如果您生成了 cxf-codegen 或类似工具的 Web 服务 stub 代码，则您可能想使用 ServiceInterfaceStrategy。如果您没有注解的服务接口，您应该使用 QNameStrategy 或 TypeNameStrategy。
version	1.1	字符串	SOAP 版本应为 1.1 或 1.2。默认是 1.1
namespacePrefixRef		字符串	使用 JAXB 或 SOAP 进行 marshalling 时，JAXB 实施将自动分配命名空间前缀，如 ns2、ns3、ns4 等。要控制此映射，Camel 允许您引用包含所需映射的映射。
schema		字符串	对现有模式进行验证。您可以使用前缀 classpath:、file: 或 http: 指定资源应如何解析。您可以使用 ',' 字符分隔多个架构文件。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

312.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 8 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.soapjaxb.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.soapjaxb.context-path	您的 JAXB 类所在的软件包名称。		字符串

Name	描述	默认值	类型
camel.dataformat.soapjaxb.element-name-strategy-ref	引用 registry 中要查找的元素策略。元素名称策略用于两个目的。第一个方法是查找给定对象的 xml 元素名称，并在将对象放入 SOAP 消息时进行 soap 操作。第二种是查找给定 soap 错误名称的 Exception 类。以下三元素策略类名称开箱即用。QNameStrategy - 使用在实例化时配置的固定 qName。不支持异常查找 - 使用给定类型的 XMLType 注解中的名称和命名空间。如果没有设置命名空间，则使用 package-info。不支持 ServiceInterfaceStrategy - 使用 webservice 接口的信息来确定类型名称，并查找 SOAP 错误的异常类，所有三种类都位于软件包名称 org.apache.camel.dataformat.soap.dataformat.soap.name 中，如果您生成了 cxf-codegen 或类似工具的 Web 服务 stub 代码，则您可能想使用 ServiceInterfaceStrategy。如果您没有注解的服务接口，您应该使用 QNameStrategy 或 TypeNameStrategy。		字符串
camel.dataformat.soapjaxb.enabled	启用 soapjaxb dataformat	true	布尔值
camel.dataformat.soapjaxb.encoding	overrule 并使用特定的编码		字符串
camel.dataformat.soapjaxb.namespace-prefix-ref	使用 JAXB 或 SOAP 进行 marshalling 时，JAXB 实施将自动分配命名空间前缀，如 ns2、ns3、ns4 等。要控制此映射，Camel 允许您引用包含所需映射的映射。		字符串
camel.dataformat.soapjaxb.schema	对现有模式进行验证。您可以使用前缀 classpath:、file: 或 http: 指定资源应如何解析。您可以使用 ',' 字符分隔多个架构文件。		字符串
camel.dataformat.soapjaxb.version	SOAP 版本应为 1.1 或 1.2。默认是 1.1	1.1	字符串

ND

312.3. ELEMENTNAMESTRATEGY

元素名称策略用于两个目的。第一个方法是查找给定对象的 xml 元素名称，并在将对象放入 SOAP 消息时进行 soap 操作。第二种是查找给定 soap 错误名称的 Exception 类。

策略	使用方法
QName Strategy	使用在实例化时配置的固定 qName。不支持异常查找
TypeNameStrategy	使用给定类型的 @XMLType 注释中的名称和命名空间。如果没有设置命名空间，则使用 package-info。不支持异常查找
ServiceInterfaceStrategy	使用 webservice 界面中的信息来确定类型名称，并查找 SOAP 错误的异常类

如果您使用 cxf-codegen 或类似的工具生成 web 服务存根代码，那么您可能想使用 ServiceInterfaceStrategy。如果您没有注解的服务接口，您应该使用 QNameStrategy 或 TypeNameStrategy。

312.4. 使用 JAVA DSL

以下示例使用名为 DataFormat 为 soap，它配置有软件包 com.example.customerservice 来初始化 JAXBContext。第二个参数是 ElementNameStrategy。路由可以 marshal 常规对象和例外。（请注意以下仅向队列发送 SOAP Envelope。Web 服务提供商实际上需要侦听实际发生 SOAP 调用的队列，在这种情况下，它将是 SOAP 请求的一种方式。如果您需要请求回复，您应该查看下一个示例。）

```
SoapJaxbDataFormat soap = new SoapJaxbDataFormat("com.example.customerservice", new ServiceInterfaceStrategy(CustomerService.class));
from("direct:start")
  .marshal(soap)
  .to("jms:myQueue");
```

提示

另请参阅 as the SOAP dataformat inherits from the JAXB dataformat大多数设置在此也适用于

312.4.1. 使用 SOAP 1.2

从 Camel 2.11 开始提供

```
SoapJaxbDataFormat soap = new SoapJaxbDataFormat("com.example.customerservice", new ServiceInterfaceStrategy(CustomerService.class));
soap.setVersion("1.2");
```

```
from("direct:start")
  .marshal(soap)
  .to("jms:myQueue");
```

当使用 XML DSL 时，您可以在 <soapjaxb> 元素上设置一个 version 属性。

```
    <!-- Defining a ServiceInterfaceStrategy for retrieving the element name when marshalling --
>
    <bean id="myNameStrategy"
class="org.apache.camel.dataformat.soap.name.ServiceInterfaceStrategy">
        <constructor-arg value="com.example.customerservice.CustomerService"/>
    <constructor-arg value="true"/>
    </bean>
```

在 Camel 路由中

```
<route>
  <from uri="direct:start"/>
  <marshal>
    <soapjaxb contentPath="com.example.customerservice" version="1.2"
elementNameStrategyRef="myNameStrategy"/>
  </marshal>
  <to uri="jms:myQueue"/>
</route>
```

312.5. 多部分消息

从 Camel 2.8.1 开始提供

ServiceInterfaceStrategy 支持多部分 SOAP 消息。ServiceInterfaceStrategy 必须初始化，它根据 JAX-WS 2.2 注解的服务接口定义，并满足 Document Bare 风格的要求。目标方法需要满足以下条件，遵循 JAX-WS 规格：1) 它最多有一个 in 或 in/out 非标头的参数，2) 如果它有一个非 void 的返回类型，它必须没有 in/out 或 out 非标题的参数，3) 如果它有一个返回类型 void，它需要最多有一个 in/out 或 out 非标头的参数。

ServiceInterfaceStrategy 应该使用布尔值参数初始化，以指示映射策略是否应用到请求参数或响应参数。

```
ServiceInterfaceStrategy strat =  new
ServiceInterfaceStrategy(com.example.customerservice.multipart.MultiPartCustomerService.c
lass, true);
SoapJaxbDataFormat soapDataFormat = new
SoapJaxbDataFormat("com.example.customerservice.multipart", strat);
```

312.5.1. 多部分请求

多部分请求的有效负载参数是 initiazlied 使用代表目标操作的签名的 BeanInvocation 对象。在调用 marshal （） 处理器时，camel-soap DataFormat 将 BeanInvocation 中的内容映射到 SOAP 标头中的字段，并遵循 JAX-WS 映射。

```
BeanInvocation beanInvocation = new BeanInvocation();

// Identify the target method
beanInvocation.setMethod(MultiPartCustomerService.class.getMethod("getCustomersByNam
e",
    GetCustomersByName.class, com.example.customerservice.multipart.Product.class));

// Populate the method arguments
GetCustomersByName getCustomersByName = new GetCustomersByName();
getCustomersByName.setName("Dr. Multipart");

Product product = new Product();
product.setName("Multiuse Product");
product.setDescription("Useful for lots of things.");

Object[] args = new Object[] {getCustomersByName, product};

// Add the arguments to the bean invocation
beanInvocation.setArgs(args);

// Set the bean invocation object as the message body
exchange.getIn().setBody(beanInvocation);
```

312.5.2. 多部分响应

多部分 soap 响应可能会在 soap 正文中包含元素，并在 soap 标头中包含一个或多个元素。camel-soap DataFormat 将 unmarshall 位于 soap 正文中的元素（如果存在），并将它放到交换中 out 消息的正文。标头元素 不会 放入其 JAXB 映射的对象类型中。相反，这些元素被放入 camel out 消息标头 org.apache.camel.dataformat.soap.UNMARSHALLED_HEADER_LIST 中。元素将显示为元素实例值，或是 JAXBElement 值，具体取决于 ignoreJAXBElement 属性的设置。此属性从 camel-jaxb 继承。

您还可以通过将 ignoreUnmarshalledHeaders 值设置为 true 来让 camel-soap DataFormate 忽略标头内容。

312.5.3. 拥有者对象映射

JAX-WS 指定将类型参数化 javax.xml.ws.Holder 对象用于 In/Out 和 Out 参数。构建 BeanInvocation 时可以使用 Holder 对象，或者您可以直接使用参数化类型的实例。camel-soap

DataFormat marshals Holder 值根据 Holder 的值的类的 JAXB 映射决定。unmarshalled 响应中没有为 'Holder 对象提供映射。

312.6. 例子

312.6.1. WebService 客户端

以下路由支持对请求进行编译和处理响应或错误。

```
String WS_URI = "cxf://http://myserver/customerservice?
serviceClass=com.example.customerservice&dataFormat=MESSAGE";
SoapJaxbDataFormat soapDF = new SoapJaxbDataFormat("com.example.customerservice",
new ServiceInterfaceStrategy(CustomerService.class));
from("direct:customerServiceClient")
  .onException(Exception.class)
    .handled(true)
    .unmarshal(soapDF)
  .end()
  .marshal(soapDF)
  .to(WS_URI)
  .unmarshal(soapDF);
```

以下片段为服务接口创建代理，并对上述路由发出 SOAP 调用。

```
import org.apache.camel.Endpoint;
import org.apache.camel.component.bean.ProxyHelper;
...

Endpoint startEndpoint = context.getEndpoint("direct:customerServiceClient");
ClassLoader classLoader = Thread.currentThread().getContextClassLoader();
// CustomerService below is the service endpoint interface, *not* the javax.xml.ws.Service
subclass
CustomerService proxy = ProxyHelper.createProxy(startEndpoint, classLoader,
CustomerService.class);
GetCustomersByNameResponse response = proxy.getCustomersByName(new
GetCustomersByName());
```

312.6.2. WebService Server

使用以下路由设置 webservice 服务器，它使用类 CustomerServiceImpl 侦听 jms 队列 customerServiceQueue 和 processes 请求。customerServiceImpl 方法应实施接口 CustomerService。它可以在 spring 上下文中定义，而不是直接实例化服务器类作为常规 Bean。

```
SoapJaxbDataFormat soapDF = new SoapJaxbDataFormat("com.example.customerservice",
new ServiceInterfaceStrategy(CustomerService.class));
CustomerService serverBean = new CustomerServiceImpl();
```

```
from("jms://queue:customerServiceQueue")
  .onException(Exception.class)
    .handled(true)
    .marshal(soapDF)
  .end()
  .unmarshal(soapDF)
  .bean(serverBean)
  .marshal(soapDF);
```

312.7. 依赖项

要在 camel 路由中使用 SOAP dataformat，您需要将以下依赖项添加到 pom 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-soap</artifactId>
  <version>2.3.0</version>
</dependency>
```

第 313 章 SOLR 组件

从 Camel 版本 2.9 开始提供

Solr 组件允许您与 Apache Lucene Solr 服务器（基于 SolrJ 3.5.0）进行接口。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-solr</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

313.1. URI 格式

注意： 自 Camel 2.14 起，新增加了 solrs 和 solrCloud。

```
solr://host[:port]/solr?[options]
solrs://host[:port]/solr?[options]
solrCloud://host[:port]/solr?[options]
```

313.2. SOLR 选项

Solr 组件没有选项。

Solr 端点使用 URI 语法进行配置：

```
solr:url
```

使用以下路径和查询参数：

313.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
url	solr 服务器所需的主机名和端口		字符串

313.2.2. 查询参数(13 参数)：

Name	描述	默认值	类型
allowCompression (producer)	服务器端必须支持 gzip 或防御才能生效		布尔值
connectionTimeout (producer)	底层 HttpConnectionManager 上的 connectionTimeout		整数
defaultMaxConnectionsPer Host (producer)	底层 HttpConnectionManager 上的 maxConnectionsPerHost		整数
followRedirects (producer)	指明重定向是否用于进入 Solr 服务器		布尔值
maxRetries (producer)	在临时错误时尝试尝试的最大重试次数		整数
maxTotalConnections (producer)	底层 HttpConnectionManager 上的 maxTotalConnection		整数
requestHandler (producer)	设置要使用的请求处理程序		字符串
soTimeout (producer)	读取底层 HttpConnectionManager 上的超时。这是查询的理想选择，但可能不适用于索引		整数
streamingQueueSize (producer)	设置 StreamingUpdateSolrServer 的队列大小	10	int
streamingThreadCount (producer)	设置 StreamingUpdateSolrServer 的线程数量	2	int
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
集合 (solrCloud)	设置 solrCloud 服务器可以使用的集合名称		字符串
zkHost (solrCloud)	设置可以使用 solrCloud 的 ZooKeeper 主机信息，如 zkhost=localhost:8123。		字符串

313.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.solr.enabled	启用 solr 组件	true	布尔值
camel.component.solr.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

313.4. 消息操作

目前支持以下 Solr 操作：只需设置带有键为"SolrOperation"的 Exchange 标头，并将值设为以下之一：有些操作还需要设置消息正文。

- *INSERT 操作使用 CommonsHttpSolrServer*
- *INSERT_STREAMING 操作使用 StreamingUpdateSolrServer (Camel 2.9.2)*

操作	消息正文	描述
INSERT/INSERT_STREAMING	不适用	使用消息标头添加索引（必须带有 "SolrField." 前缀）
INSERT/INSERT_STREAMING	File	使用给定文件添加索引（使用 ContentStreamUpdateRequest）
INSERT/INSERT_STREAMING	SolrInputDocument	Camel 2.9.2 根据给定的 SolrInputDocument 更新索引

操作	消息正文	描述
INSERT/INSERT_STREAMING	字符串 XML	Camel 2.9.2 根据给定的 XML 更新索引（必须采用 SolrInputDocument 格式）
ADD_BEAN	Bean 实例	根据注解中的值 添加索引
ADD_BEANS	collection<bean>	Camel 2.15 根据 注解的 bean集合添加索引
DELETE_BY_ID	要删除的索引 ID	通过 ID 删除记录
DELETE_BY_QUERY	查询字符串	通过查询删除记录
COMMIT	不适用	对任何待处理的索引更改执行提交
回滚	不适用	对任何待处理的索引更改执行回滚
OPTIMIZE	不适用	对任何待处理的索引更改执行提交，然后运行 optimize 命令

313.5. EXAMPLE

以下是简单的 INSERT、DELETE 和 COMMIT 示例

```
from("direct:insert")
    .setHeader(SolrConstants.OPERATION, constant(SolrConstants.OPERATION_INSERT))
    .setHeader(SolrConstants.FIELD + "id", body())
    .to("solr://localhost:8983/solr");

from("direct:delete")
    .setHeader(SolrConstants.OPERATION,
constant(SolrConstants.OPERATION_DELETE_BY_ID))
    .to("solr://localhost:8983/solr");

from("direct:commit")
    .setHeader(SolrConstants.OPERATION, constant(SolrConstants.OPERATION_COMMIT))
    .to("solr://localhost:8983/solr");
```

```
<route>
   <from uri="direct:insert"/>
   <setHeader headerName="SolrOperation">
      <constant>INSERT</constant>
   </setHeader>
   <setHeader headerName="SolrField.id">
      <simple>${body}</simple>
   </setHeader>
   <to uri="solr://localhost:8983/solr"/>
</route>
<route>
   <from uri="direct:delete"/>
   <setHeader headerName="SolrOperation">
      <constant>DELETE_BY_ID</constant>
   </setHeader>
   <to uri="solr://localhost:8983/solr"/>
</route>
<route>
   <from uri="direct:commit"/>
   <setHeader headerName="SolrOperation">
      <constant>COMMIT</constant>
   </setHeader>
   <to uri="solr://localhost:8983/solr"/>
</route>
```

客户端只需将正文消息传递给插入或删除路由，然后调用提交路由。

```
   template.sendBody("direct:insert", "1234");
   template.sendBody("direct:commit", null);
   template.sendBody("direct:delete", "1234");
   template.sendBody("direct:commit", null);
```

313.6. 查询 SOLR

目前，这个组件不支持原生查询数据（稍后添加）。现在，您可以使用 HTTP 查询 Solr，如下所示：

```
//define the route to perform a basic query
from("direct:query")
   .recipientList(simple("http://localhost:8983/solr/select/?q=${body}"))
   .convertBodyTo(String.class);
...
//query for an id of '1234' (url encoded)
String responseXml = (String) template.requestBody("direct:query", "id%3A1234");
```

如需更多信息，请参阅这些资源...

Solr Query Tutorial

Solr 查询语法

313.7. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

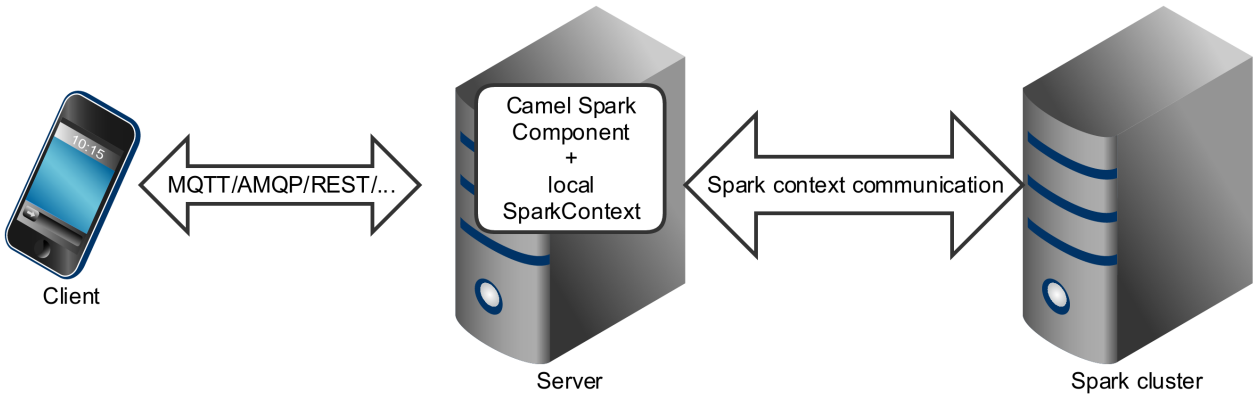
第 314 章 APACHE SPARK 组件

从 Camel 版本 2.17 开始提供

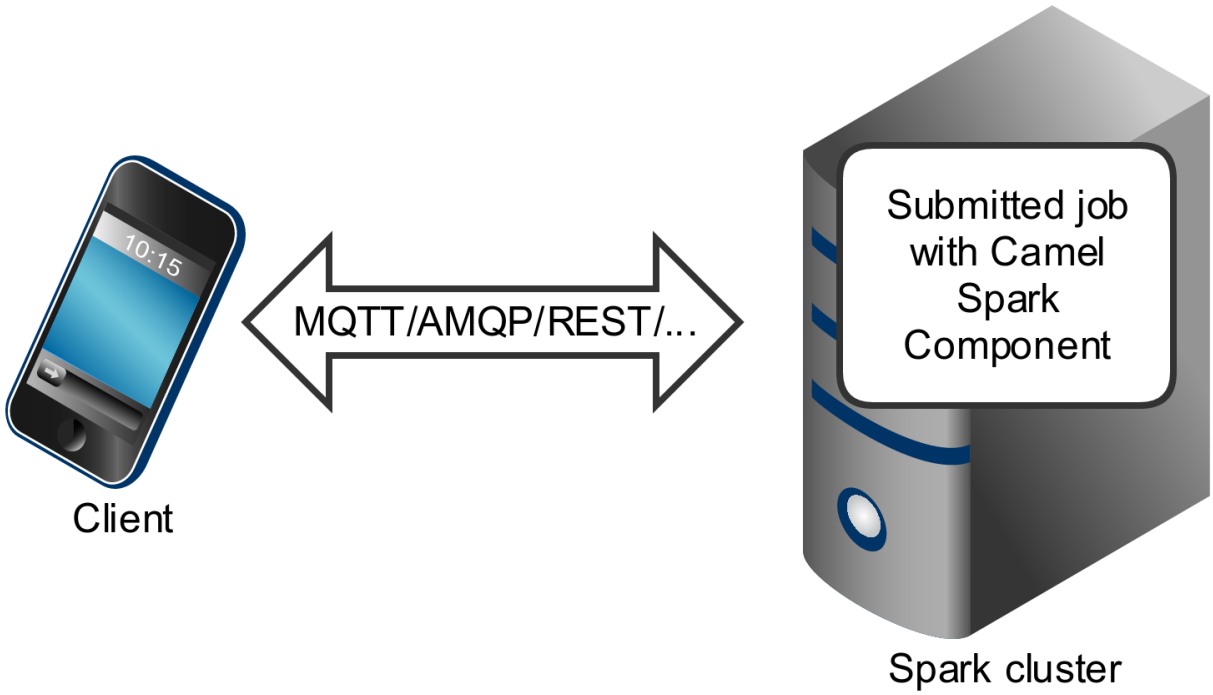
本文档页面涵盖了 Apache Camel 的 Apache Spark 组件。Spark 与 Camel 集成的主要目的是，在 Camel 连接器和 Spark 任务之间提供桥接。特别是 Camel 连接器提供了一种方式，将消息从各种传输路由、动态选择要执行的任务，使用传入消息作为该任务的输入数据，最后将执行结果传送回 Camel 管道。

314.1. 支持的架构样式

spark 组件可以用作部署到应用服务器中的驱动程序应用（或作为 fat jar 执行）。



spark 组件也可以作为作业直接提交到 Spark 集群。



虽然 Spark 组件是主要的，用来 作为 Spark 集群和其他端点之间的桥接来工作，但您也可以将其用作 触发 的简短作业。

314.2. 在 OSGI 服务器中运行 SPARK

目前，Spark 组件不支持在 OSGi 容器中执行。spark 设计为作为 fat jar 执行，通常作为作业提交到集群。由于在 OSGi 服务器中运行 Spark 的原因至少具有挑战性，而且还不支持 Camel。

314.3. URI 格式

目前，Spark 组件只支持 producers - 它旨在调用 Spark 作业和返回结果。您可以调用 RDD、数据帧或 Hive SQL 作业。

spark URI 格式

```
spark:{rdd|dataframe|hive}
```

314.3.1. spark 选项

Apache Spark 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
rdd (producer)	RDD 到计算.		JavaRDDLike
rddCallback (producer)	对 RDD 执行操作的功能。		RddCallback
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Apache Spark 端点使用 URI 语法进行配置：

```
spark:endpointType
```

使用以下路径和查询参数：

314.3.2. 路径参数(1 参数)：

Name	描述	默认值	类型
endpointType	端点 **所需的** 类型(rdd、dataframe、hive)。		EndpointType

314.3.3. 查询参数(6 参数)：

Name	描述	默认值	类型
collect (producer)	指明是否应收集或计算结果。	true	布尔值
dataFrame (producer)	DataFrame 以计算.		Dataset
dataFrameCallback (producer)	对 DataFrame 执行操作的功能。		DataFrameCallback
rdd (producer)	RDD 到计算.		JavaRDDLike
rddCallback (producer)	对 RDD 执行操作的功能。		RddCallback

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

314.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.spark.enabled	启用 spark 组件	true	布尔值
camel.component.spark.rdd	RDD 到计算.选项是 org.apache.spark.api.java.JavaRDDLike 类型。		字符串
camel.component.spark.rdd-callback	对 RDD 执行操作的功能。选项是 org.apache.camel.component.spark.RddCallback 类型。		字符串
camel.component.spark.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

314.5. RDD 作业

要调用 RDD 作业，请使用以下 URI：

spark RDD producer

> ***spark:rdd?rdd=#testFileRdd&rddCallback=#transformation***

> ***Where `rdd` option refers to the name of an RDD instance (subclass of `org.apache.spark.api.java.JavaRDDLike`) from a Camel registry, while `rddCallback` refers to the implementation of `org.apache.camel.component.spark.RddCallback` interface (also from a registry). RDD callback provides a single method used to apply incoming messages against the given RDD. Results of callback computations are saved as a body to an exchange.***

spark RDD 回调

```
public interface RddCallback<T> {
   T onRdd(JavaRDDLike rdd, Object... payloads);
}
```

以下片段演示了如何将消息作为输入发送到作业并返回结果：

调用 spark 作业

```
String pattern = "job input";
long linesCount = producerTemplate.requestBody("spark:rdd?
rdd=#myRdd&rddCallback=#countLinesContaining", pattern, long.class);
```

上面注册的代码片段的 RDD 回调，如下所示：

spark RDD 回调

```
@Bean
RddCallback<Long> countLinesContaining() {
   return new RddCallback<Long>() {
      Long onRdd(JavaRDDLike rdd, Object... payloads) {
         String pattern = (String) payloads[0];
         return rdd.filter({line -> line.contains(pattern)}).count();
      }
   }
}
```

Spring 中的 RDD 定义可能如下所示：

spark RDD 定义

```
@Bean
JavaRDDLike myRdd(JavaSparkContext sparkContext) {
  return sparkContext.textFile("testrdd.txt");
}
```

314.5.1. void RDD 回调

如果您的 RDD 回调没有返回任何值回 Camel 管道，您可以返回 null 值或使用 VoidRddCallback 基础类：

spark RDD 定义

```
@Bean
RddCallback<Void> rddCallback() {
  return new VoidRddCallback() {
        @Override
        public void doOnRdd(JavaRDDLike rdd, Object... payloads) {
            rdd.saveAsTextFile(output.getAbsolutePath());
        }
    };
}
```

314.5.2. 转换 RDD 回调

如果您知道将哪些类型的输入数据发送到 RDD 回调，您可以使用 ConvertingRddCallback，并让 Camel 在将它们插入到回调前自动转换传入的信息：

spark RDD 定义

```
@Bean
RddCallback<Long> rddCallback(CamelContext context) {
  return new ConvertingRddCallback<Long>(context, int.class, int.class) {
          @Override
          public Long doOnRdd(JavaRDDLike rdd, Object... payloads) {
              return rdd.count() * (int) payloads[0] * (int) payloads[1];
          }
      };
    };
}
```

314.5.3. 注解的 RDD 回调

使用 RDD 回调的最简单方法是为类提供标记为 @RddCallback 注释的方法：

注解的 RDD 回调定义

```
import static
org.apache.camel.component.spark.annotations.AnnotatedRddCallback.annotatedRddCallbac
k;
```

```
@Bean
RddCallback<Long> rddCallback() {
   return annotatedRddCallback(new MyTransformation());
}

...

import org.apache.camel.component.spark.annotation.RddCallback;

public class MyTransformation {

   @RddCallback
   long countLines(JavaRDD<String> textFile, int first, int second) {
      return textFile.count() * first * second;
   }

}
```

如果您将 CamelContext 传递给注解的 RDD 回调工厂方法，创建的回调将能够转换传入的有效负载以匹配注解的方法的参数：

注解的 RDD 回调的正文转换

```
import static
org.apache.camel.component.spark.annotations.AnnotatedRddCallback.annotatedRddCallbac
k;

@Bean
RddCallback<Long> rddCallback(CamelContext camelContext) {
   return annotatedRddCallback(new MyTransformation(), camelContext);
}

...


import org.apache.camel.component.spark.annotation.RddCallback;

public class MyTransformation {

   @RddCallback
   long countLines(JavaRDD<String> textFile, int first, int second) {
      return textFile.count() * first * second;
   }

}

...

// Convert String "10" to integer
long result = producerTemplate.requestBody("spark:rdd?
rdd=#rdd&rddCallback=#rddCallback" Arrays.asList(10, "10"), long.class);
```

314.6. DATAFRAME 作业

除了使用 RDDs Spark 组件外，也可以使用 DataFrames。

要调用 DataFrame 作业，请使用以下 URI：

spark RDD producer

```
spark:dataframe?dataFrame=#testDataFrame&dataFrameCallback=#transformation
```

```
 Where `dataFrame` option refers to the name of an DataFrame instance
(`instances of org.apache.spark.sql.Dataset and org.apache.spark.sql.Row`) from a Camel
registry,
while `dataFrameCallback` refers to the implementation
of `org.apache.camel.component.spark.DataFrameCallback` interface (also
from a registry). DataFrame callback provides a single method used to
apply incoming messages against the given DataFrame. Results of callback
computations are saved as a body to an exchange.
```

spark RDD 回调

```
public interface DataFrameCallback<T> {
    T onDataFrame(Dataset<Row> dataFrame, Object... payloads);
}
```

以下片段演示了如何将消息作为输入发送到作业并返回结果：

调用 spark 作业

```
String model = "Micra";
long linesCount = producerTemplate.requestBody("spark:dataFrame?
dataFrame=#cars&dataFrameCallback=#findCarWithModel", model, long.class);
```

上面注册的代码片段的 DataFrame 回调，如下所示：

spark RDD 回调

```
@Bean
```

```
RddCallback<Long> findCarWithModel() {
    return new DataFrameCallback<Long>() {
        @Override
        public Long onDataFrame(Dataset<Row> dataFrame, Object... payloads) {
            String model = (String) payloads[0];
            return dataFrame.where(dataFrame.col("model").eqNullSafe(model)).count();
        }
    };
}
```

Spring 中的 DataFrame 定义可能如下所示：

spark RDD 定义

```
@Bean
Dataset<Row> cars(HiveContext hiveContext) {
    Dataset<Row> jsonCars = hiveContext.read().json("/var/data/cars.json");
    jsonCars.registerTempTable("cars");
    return jsonCars;
}
```

314.7. HIVE 作业

Instead of working with RDDs or DataFrame Spark component can also receive Hive SQL queries as payloads. To send Hive query to Spark component, use the following URI:

spark RDD producer

```
spark:hive
```

以下片段演示了如何将消息作为输入发送到作业并返回结果：

调用 spark 作业

```
long carsCount = template.requestBody("spark:hive?collect=false", "SELECT * FROM cars",
Long.class);
List<Row> cars = template.requestBody("spark:hive", "SELECT * FROM cars", List.class);
```

在查询之前，我们想对其执行查询的表应在 HiveContext 中注册。例如，在 Spring such registration 中可以如下所示：

spark RDD 定义

```
@Bean
Dataset<Row> cars(HiveContext hiveContext) {
    jsonCars = hiveContext.read().json("/var/data/cars.json");
   jsonCars.registerTempTable("cars");
   return jsonCars;
}
```

314.8. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 315 章 SPARK REST 组件

从 Camel 版本 2.14 开始提供

Spark-rest 组件允许使用 Spark REST Java 库（使用 Rest DSL）定义 REST 端点。

INFO: Spark Java 需要 Java 8 运行时。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-spark-rest</artifactId>
    <version>${camel-version}</version>
</dependency>
```

315.1. URI 格式

```
spark-rest://verb:path?[options]
```

315.2. URI 选项

Spark Rest 组件支持 12 个选项，如下所列。

Name	描述	默认值	类型
port (consumer)	端口号。默认使用 4567	4567	int
ipaddress (consumer)	设置 Spark 应该侦听的 IP 地址。如果没有调用默认地址，则为 '0.0.0.0'。	0.0.0.0	字符串
MinThreads (advanced)	Spark thread-pool 中的最小线程数（全局共享）		int
maxThreads (advanced)	Spark thread-pool 中的最大线程数（全局共享）		int
timeOutMillis (advanced)	线程闲置超时(millis)，在较长时间内闲置线程将从线程池中终止		int

Name	描述	默认值	类型
keystoreFile (security)	配置连接以使用密钥存储文件		字符串
keyStorePassword (security)	配置连接以使用密钥存储密码		字符串
truststoreFile (security)	配置连接以使用 truststore 文件		字符串
trustStorePassword (security)	配置连接以使用 truststore 密码		字符串
sparkConfiguration (advanced)	使用共享 SparkConfiguration		SparkConfiguration
sparkBinding (advanced)	使用自定义 SparkBinding 映射到/来自 Camel 消息。		SparkBinding
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Spark Rest 端点使用 URI 语法进行配置：

```
spark-rest:verb:path
```

使用以下路径和查询参数：

315.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
verb	**需要** get、post、put、patch、delete、head、trace、connect 或 options。		字符串
path	**必需** 支持 Spark 语法的内容路径。		字符串

315.2.2. 查询参数(11 参数)：

Name	描述	默认值	类型
accept (consumer)	接受类型，如 'text/xml' 或 'application/json'。默认情况下，我们接受所有类型。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
disableStreamCache (consumer)	确定 Spark HttpRequest"getContent （）中的原始输入流是否被缓存(Camel 会将流读取到基于轻量级内存的流缓存)缓存中。默认情况下，Camel 将缓存 Netty 输入流，以支持多次读取，以确保 Camel 可以从流检索所有数据。但是，当您需要访问原始流时，您可以将此选项设置为 true，例如将其直接流传输到文件或其他持久性存储。请注意，如果您启用了这个选项，则无法多次读取 Netty 流，您需要手动重置 Spark 原始流上的 reader 索引。	false	布尔值
mapHeaders (consumer)	如果启用了这个选项，那么在从 Spark 绑定到 Camel Message 的过程中，标头也会被映射（如作为标头添加到 Camel 消息）。您可以关闭这个选项来禁用这个选项。标头仍然可以从 org.apache.camel.component.sparkrest.SparkMessage 消息访问，该消息返回 Spark HTTP 请求实例。	true	布尔值
transferException (consumer)	如果在消费者端启用并交换失败处理，如果原因 Exception 在响应中作为 application/x-java-serialized-object 内容类型发送序列化，则结果为 application/x-java-serialized-object 内容类型。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
urlDecodeHeaders (consumer)	如果启用了这个选项，那么在从 Spark 绑定到 Camel Message 的过程中，标头值将被解码（例如 %20 将是一个空格字符）。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
matchOnUriPrefix (advanced)	如果找不到完全匹配，消费者是否应该尝试通过匹配 URI 前缀来查找目标消费者。	false	布尔值

Name	描述	默认值	类型
sparkBinding (advanced)	使用自定义 SparkBinding 映射到/来自 Camel 消息。		SparkBinding
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

315.3. 使用 SPARK 语法的路径

path 选项使用 Spark REST 语法来定义，您可以在其中使用支持参数和 splat 定义 REST 上下文路径。请参阅 Spark Java Route 文档。

以下是使用固定路径的 Camel 路由

```
from("spark-rest:get:hello")
  .transform().constant("Bye World");
```

以下路由使用参数，该参数映射到带有键"me"的 Camel 标头。

```
from("spark-rest:get:hello/:me")
  .transform().simple("Bye ${header.me}");
```

315.4. 映射到 CAMEL 消息

Spark Request 对象映射到 Camel Message 作为 org.apache.camel.component.sparkrest.SparkMessage，它可以使用 getRequest 方法访问原始 Spark 请求。默认情况下，Spark 正文映射到 Camel 消息正文，任何 HTTP 标头 / Spark 参数都会映射到 Camel Message 标头。Spark splat 语法有特殊支持，它通过键 splat 映射到 Camel 消息标头。

例如，以下给定路由在上下文路径中使用 Spark splat （星号符号）来作为标头形成简单语言来构造响应消息。

```
from("spark-rest:get:/hello/*/to/*")
  .transform().simple("Bye big ${header.splat[1]} from ${header.splat[0]}");
```

315.5. REST DSL

Apache Camel 提供了一个新的 Rest DSL，它允许以 nice REST 样式定义 REST 服务。例如，我们

可以定义 REST hello 服务，如下所示：

```
return new RouteBuilder() {
   @Override
   public void configure() throws Exception {
        rest("/hello/{me}").get()
           .route().transform().simple("Bye ${header.me}");
    }
 };
```

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <rest uri="/hello/{me}">
    <get>
      <route>
        <transform>
          <simple>Bye ${header.me}</simple>
        </transform>
      </route>
    </get>
  </rest>
</camelContext>
```

请参阅 Rest DSL 的更多详细信息。

315.6. 更多示例

Apache Camel 分发中有一个 camel-example-spark-rest-tomcat 示例，它演示了如何在 Apache Tomcat 或类似 web 容器上部署的 web 应用程序中使用 camel-spark-rest。

第 316 章 SPEL 语言

从 Camel 版本 2.7 开始提供

Camel 允许 Spring Expression Language (SpEL) 用作 DSL 或 XML 配置中的 Expression 或 Predicate。

注意

建议您在 Spring 运行时中使用 SpEL。但是，从 Camel 2.21 开始，您可以在其他运行时中使用 SpEL （当 Spring 运行时没有运行时，功能 SpEL 无法这样做）

316.1. 变量

以下变量在表达式和在 SpEL 中编写的 predicates 中提供：

变量	类型	描述
这	Exchange	Exchange 是根对象
交换	Exchange	Exchange 对象
例外	Throwable	Exchange 异常（如果有）
exchangeId	字符串	交换 ID
Fulting	消息	Fault 消息（如果有）
正文（body）	对象	IN 消息正文。
request	消息	exchange.in 消息
response	消息	exchange.out 消息（如果有）
属性	Map	交换属性

变量	类型	描述
property (name)	对象	给定名称的属性
property (name, type)	类型	给定名称中的属性作为给定类型

316.2. 选项

SpEL 语言支持 1 选项，如下所列。

Name	默认值	Java 类型	描述
trim	**true**	**布尔值**	是否修剪值以移除前导和结尾的空格和换行符

316.3. SAMPLES

316.3.1. 表达式模板

因为启用了表达式模板，SpEL 表达式需要被 #{ } delimiters 括起。这可让您将 SpEL 表达式与常规文本组合，并把它用作非常轻量级的模板语言。

例如，如果您构建以下路由：

```
from("direct:example")
   .setBody(spel("Hello #{request.body}! What a beautiful #{request.headers['dayOrNight']}"))
   .to("mock:result");
```

在上面的路由中，注意 spel 是一个静态方法，我们需要从 org.apache.camel.language.spel.SpelExpression.spel 导入，因为我们将 spel 用作传递至 setBody 方法的参数。虽然我们使用流畅的 API，我们可以这样做：

```
from("direct:example")
   .setBody().spel("Hello #{request.body}! What a beautiful #{request.headers['dayOrNight']}")
   .to("mock:result");
```

注意我们现在使用 setBody （） 方法中的 spel 方法。这不要求我们静态导入 org.apache.camel.language.spel.SpelExpression.spel 的 spel 方法。

并在正文中发送一条字符串 "World" 的消息，并发送标题 "dayOrNight"，值为 "day"：

```
template.sendBodyAndHeader("direct:example", "World", "dayOrNight", "day");
```

mock:result 的输出为 "Hello World!这一天是什么？"

316.3.2. Bean 集成

您可以在 SpEL 表达式中引用 Registry （最有可能为 ApplicationContext）中定义的 Bean。例如，如果您在 ApplicationContext 中有一个名为 "foo" 的 bean，您可以在这个 bean 上调用 "bar" 方法，如下所示：

```
#{@foo.bar == 'xyz'}
```

316.3.3. 企业级集成模式中的 SpEL

您可以使用 SpEL 作为 Recipient List 的表达式，或作为 Message Filter 中的 predicate ：

```
<route>
  <from uri="direct:foo"/>
  <filter>
    <spel>#{request.headers['foo'] == 'bar'}</spel>
    <to uri="direct:bar"/>
  </filter>
</route>
```

以及 Java DSL 中的等效功能：

```
from("direct:foo")
    .filter().spel("#{request.headers['foo'] == 'bar'}")
    .to("direct:bar");
```

316.4. 从外部资源载入脚本

从 Camel 2.11 开始提供

您可以对脚本进行外部化，并让 Camel 从资源（如 "classpath:"、"file:" 或 "http:" ）加载它。这可以通过以下语法完成：" resource:scheme:location"，例如引用您可以进行的类路径上的文件：

```
.setHeader("myHeader").spel("resource:classpath:myspel.txt")
```

第 317 章 SPLUNK 组件

从 Camel 版本 2.13 开始提供

Splunk 组件使用 Splunk 提供的 客户端 api 提供对 Splunk 的访问，它可让您在 Splunk 中发布和搜索事件。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-splunk</artifactId>
    <version>${camel-version}</version>
</dependency>
```

317.1. URI 格式

```
splunk://[endpoint]?[options]
```

317.2. 生产者端点：

端点	描述
流	如果没有指定，将数据流传输到命名索引或默认值。使用流模式时，Splun 在事件进入索引前有一些内部缓冲区（关于 1MB 等等）。如果您需要实时，请更好地使用 submit 或 tcp 模式。
提交	提交模式.使用 Splunk rest api 将事件发布到命名的索引；如果没有指定，则使用默认值。
tcp	TCP 模式.将数据流到 tcp 端口，并在 Splunk 中需要一个开放的接收器端口。

在发布事件时，消息正文应包含 SplunkEvent。请参阅消息正文下的注释。

Example

```
from("direct:start").convertBodyTo(SplunkEvent.class)
    .to("splunk://submit?
username=user&password=123&index=myindex&sourceType=someSourceType&source=mySource")...
```

在本例中，需要一个转换器来转换为 SplunkEvent 类。

317.3. 消费者端点：

端点	描述
normal	执行正常搜索，并在搜索选项中要求搜索查询。
savedsearch	根据 mvapich 中保存的搜索查询执行搜索，且需要在 savedSearch 选项中查询名称。

Example

```
    from("splunk://normal?delay=5s&username=user&password=123&initEarliestTime=-10s&search=search index=myindex sourcetype=someSourcetype")
        .to("direct:search-result");
```

camel-splunk 使用正文中的 SplunkEvent 为每个搜索结果创建一个路由交换。

317.4. URI 选项

Splunk 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
mvapichConfiguration Factory (advanced)	使用 SplunkConfigurationFactory		SplunkConfiguration Factory
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Splunk 端点使用 URI 语法进行配置：

```
splunk:name
```

使用以下路径和查询参数：

317.4.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	**必需** 的名称没有目的		字符串

317.4.2. 查询参数(42 参数)：

Name	描述	默认值	类型
app (common)	Splunk 应用程序		字符串
connectionTimeout (common)	连接到 Splunk 服务器时的 MS 中的超时	5000	int
host (common)	Splunk 主机.	localhost	字符串
owner (common)	Splunk 所有者		字符串
port (common)	Splunk 端口	8089	int
scheme (common)	Splunk 方案	https	字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
count (consumer)	表示要返回的最大实体数的数字。		int
earliestTime (consumer)	搜索时间窗的最早时间。		字符串
initEarliestTime (consumer)	第一个搜索的初始开始偏移		字符串
latestTime (consumer)	搜索时间窗的最新时间。		字符串

Name	描述	默认值	类型
savedSearch (consumer)	Splunk 中保存的查询名称		字符串
search (consumer)	要运行的 Splunk 查询		字符串
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
streaming (consumer)	设置流模式。流模式在收到交换时发送交换，而不是批量发送。		布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
eventHost (producer)	覆盖默认的 Splunk 事件主机字段		字符串
index (producer)	要写入的 Splunk 索引		字符串
raw (producer)	如果有效负载插入原始数据	false	布尔值
source (producer)	Splunk source 参数		字符串
sourceType (producer)	Splunk sourcetype 参数		字符串
tcpReceiverPort (producer)	Splunk tcp 接收器端口		int
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int

Name	描述	默认值	类型
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
密码 (security)	Splunk 的密码		字符串

Name	描述	默认值	类型
SSLProtocol (security)	设置要使用的 ssl 协议	TLSv1.2	SSLSecurityProtocol
用户名 (security)	Splunk 的用户名		字符串
useSunHttpsHandler (security)	使用 sun.net.www.protocol.https.Handler Https 处理程序建立 Splunk 连接。在应用服务器中运行时，可以使用 app. server https 处理。	false	布尔值

317.5. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.splunk.enabled	启用 mvapich 组件	true	布尔值
camel.component.splunk.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.splunk.splunk-configuration-factory	使用 SplunkConfigurationFactory。选项是 org.apache.camel.component.splunk.SplunkConfigurationFactory 类型。		字符串

317.6. 消息正文

Splunk 在键/值对中针对数据进行操作。SplunkEvent 类是此类数据的占位符，应位于制作者的消息正文中。同样，它将在每个搜索结果的每个搜索结果中返回。

自 Camel 2.16.0 起，您可以通过在生成者端点上设置 raw 选项，将原始数据发送到 Splunk。这对 eg. json/xml 和其他有效负载(Splunk)支持很有用。

317.7. 使用案例

使用 music 中搜索 tweets，并将事件发布到 Splunk

```
    from("twitter://search?
type=polling&keywords=music&delay=10&consumerKey=abc&consumerSecret=def&accessT
oken=hij&accessTokenSecret=xxx")
        .convertBodyTo(SplunkEvent.class)
        .to("splunk://submit?username=foo&password=bar&index=camel-
tweets&sourceType=twitter&source=music-tweets");
```

要将 Tweet 转换为 SplunkEvent，您可以使用如下转换器

```
@Converter
public class Tweet2SplunkEvent {
   @Converter
   public static SplunkEvent convertTweet(Status status) {
      SplunkEvent data = new SplunkEvent("twitter-message", null);
      //data.addPair("source", status.getSource());
      data.addPair("from_user", status.getUser().getScreenName());
      data.addPair("in_reply_to", status.getInReplyToScreenName());
      data.addPair(SplunkEvent.COMMON_START_TIME, status.getCreatedAt());
      data.addPair(SplunkEvent.COMMON_EVENT_ID, status.getId());
      data.addPair("text", status.getText());
      data.addPair("retweet_count", status.getRetweetCount());
      if (status.getPlace() != null) {
         data.addPair("place_country", status.getPlace().getCountry());
         data.addPair("place_name", status.getPlace().getName());
         data.addPair("place_street", status.getPlace().getStreetAddress());
      }
      if (status.getGeoLocation() != null) {
         data.addPair("geo_latitude", status.getGeoLocation().getLatitude());
         data.addPair("geo_longitude", status.getGeoLocation().getLongitude());
      }
      return data;
   }
}
```

搜索 Splunk for tweets

```
    from("splunk://normal?username=foo&password=bar&initEarliestTime=-
2m&search=search index=camel-tweets sourcetype=twitter")
        .log("${body}");
```

317.8. 其他评论

Splunk 附带各种选项，用于利用机器生成的数据与预构建的应用程序进行分析和显示。例如，jmx app. 可用于发布 jmx 属性，如路由和 jvm 指标到 Splunk，并在仪表板上显示它。

317.9. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 318 章 SPRING SUPPORT

Apache Camel 旨在以多种方式与 Spring Framework 配合工作。

- Camel 使用 Spring Transactions 作为 JMS 和 JPA等组件中的默认事务处理
- Camel 与 Spring 2 XML 处理与 Xml 配置一起工作
- Camel Spring XML Schema 在 Xml 参考中定义
- Camel 支持强大的 Spring Remoting 版本，可在客户端和服务器端使用强大的路由，并使用所有可用的组件进行传输
- Camel 提供强大的 Bean 集成，与 Spring ApplicationContext 中定义的任何 Bean 集成
- Camel 与各种 Spring 帮助程序类集成，如为 Spring Resources 提供 Type Converter 支持。
- 允许 Spring 依赖项组件实例或 CamelContext 实例本身，以及自动公开 Spring Bean 作为组件和端点。
- 允许您重复使用 Spring Testing 框架，以简化使用 企业集成模式 和 Camel 强大的 Mock 和 Test 端点的单元和集成测试
- 在 Camel 2.15 中，Camel 支持使用 camel-spring-boot 组件进行 Spring Boot。

318.1. 使用 SPRING 配置 CAMELCONTEXT

您可以使用 CamelContextFactoryBean 在任何 spring.xml 中配置 CamelContext。这将自动启动 CamelContext 以及任何引用的路由，以及任何引用的组件和 Endpoint 实例。

- 添加 Camel 模式

- 以两种方式配置路由：
  - 使用 Java Code
  - 使用 Spring XML

318.2. 添加 CAMEL 架构

对于 Camel 1.x，您需要使用以下命名空间：

```
http://activemq.apache.org/camel/schema/spring
```

使用以下模式位置：

```
http://activemq.apache.org/camel/schema/spring/camel-spring.xsd
```

您需要将 Camel 添加到 schemaLocation 声明中

```
http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd
```

因此 XML 文件类似如下：

```
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
        http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd">
```

318.2.1. 使用 camel: namespace

或者您可以在 XML 声明中引用 camel XSD ：

```
xmlns:camel="http://camel.apache.org/schema/spring"
```

i.
因此，声明是：

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:camel="http://camel.apache.org/schema/spring"
    xsi:schemaLocation="
      http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
      http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-
spring.xsd">
```

i.
然后，使用 camel: namespace 前缀，您可以省略内联命名空间声明：

```
<camel:camelContext id="camel5">
  <camel:package>org.apache.camel.spring.example</camel:package>
</camel:camelContext>
```

318.2.2. 使用 Spring 进行高级配置

请参阅 使用 Spring 的 CamelContext 高级配置中更多详情

$# using Java Code

您可以使用 Java Code 定义 RouteBuilder 实现。它们可以在 spring 中定义 Bean，然后在您的 camel 上下文中引用，例如：

318.2.3. 使用 <package>

Camel 还提供了强大的功能，允许在给定软件包中自动发现和初始化路由。这通过将标签添加到 spring 上下文定义中的 camel 上下文，指定要递归搜索 RouteBuilder 实施的软件包。要在 1.X 中使用此功能，需要一个 <package></package> 标签，指定应搜索的以逗号分隔的软件包列表，例如：

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <package>org.apache.camel.spring.config.scan.route</package>
</camelContext>
```

WARNING: 在将软件包名称指定为 org.apache.camel 或其子软件包时要小心。这会导致 Camel 在自己的软件包中搜索可能会导致问题的路由。

INFO: will ignore already instantiated classes*.<package> 和 <packageScan> 将跳过由 Spring 等创建的任何类。因此，如果您将路由构建器定义为 spring bean 标签，则会跳过该类。您可以使用 < routeBuilder ref="theBeanId"/> 或 < contextScan> 功能包含这些 Bean。*

318.2.4. 使用 <packageScan>

在 Camel 2.0 中，这已扩展为允许使用 Ant （如路径匹配）选择和排除发现的路由类。在 spring 中，这通过添加 <packageScan/> 标签来指定。该标签必须包含一个或多个 'package' 元素（类似于 1.x），可选一个或多个 'includes' 或 'excludes' 元素指定要应用到发现类的完全限定域名的模式。

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <packageScan>
    <package>org.example.routes</package>
    <excludes>**.*Excluded*</excludes>
    <includes>**.*</includes>
  </packageScan>
</camelContext>
```

在 include 模式之前应用排除模式。如果没有定义 include 或 exclude 模式，则返回软件包中发现的所有 Route 类。

在上例中，camel 将扫描所有 'org.example.routes' 软件包以及 RouteBuilder 类的任何子软件包。表明扫描找到两个 RouteBuilders，一个用于 org.example.routes 中，一个名为 'MyRoute"，另一个名为 'MyExcludedRoute'（子软件包 'excluded'）。每个类的完全限定名称都会被提取 (org.example.routes.MyRoute, org.example.routes.excluded.MyExcluded.MyExcludedRoute)，并应用 include 和 exclude 模式。

排除模式是，为为 fqcn 'org.example.routes.excluded.MyExcludedRoute' 和 veto camel 匹配，from 初始化它。

在覆盖范围内，这使用 Spring 的 AntPatternMatcher 实现，它匹配，如下所示

```
? matches one character
* matches zero or more characters
** matches zero or more segments of a fully qualified name
```

例如：

** unsetexcluded 将与 org.simple.Excluded, org.apache.camel.SomeExcludedRoute 或 org.example.RouteWhichIsExcluded 匹配*

dc?cluded 将与 org.simple.IncludedRoute 匹配 org.simple.IncludedRoute，但不匹配 org.simple.PrecludedRoute

318.2.5. 使用 contextScan

从 Camel 2.4 开始提供

您可以允许 Camel 扫描容器上下文，例如，用于路由构建器实例的 Spring ApplicationContext。这可让您使用 Spring < component-scan> 功能，并让 Camel 选择其扫描过程中由 Spring 创建的任何 RouteBuilder 实例。

这可让您使用 Spring @Component 注解路由，并包含这些路由。

```
@Component
public class MyRoute extends SpringRouteBuilder {

    @Override
    public void configure() throws Exception {
        from("direct:start").to("mock:result");
    }
}
```

您还可以使用 ANT 风格来包含和排除，如 < packageScan> 文档中所述。

318.3. 如何从其他 XML 文件导入路由

从 Camel 2.3 开始提供

当使用 Xml 配置 在 Camel 中定义路由时，您可能希望在其他 XML 文件中定义一些路由。例如，您可能有多个路由，如果某些路由位于单独的 XML 文件中，则可能有助于维护应用程序。您也可以将通用路由和可重复使用的路由存储在其他 XML 文件中，您可以根据需要简单地导入。

在 Camel 2.3 中，您现在可以在新的 < route Context/> 标签中定义在 <camel Context/> 以外的路由。

注意：当您使用 <routeContext> 后，它们会被分离，且无法重复使用现有的 <onException>、<intercept>、<dataFormats> 和类似在 <camelContext> 中定义的跨组合功能。换句话

说，<routeContext> 当前被隔离。这可能会在 Camel 3.x 中有所变化。

例如，我们可以有一个名为 myCoolRoutes.xml 的文件，其中包含几个路由，如下所示：

myCoolRoutes.xml

然后，在包含 CamelContext 的 XML 文件中，您可以使用 Spring 导入 myCoolRoute.xml 文件。然后，在 <camelContext/& gt; 中，您可以使用其 id 引用 & lt;routeContext />，如下所示：

另请注意，您可以混合和匹配，在 CamelContext 中具有路由，也可以在 RouteContext 中进行外部化。

您可以根据需要有许多 &lt ;routeContextRef/& gt;。

可重复使用的路由

< routeContext/& gt; 中定义的路由可以被多个 < camelContext/> 重复使用。但是，它只是被重复使用的定义。在运行时，每个 CamelContext 将基于定义创建自己的路由实例。

318.3.1. 测试时间排除。

在测试时，通常需要选择性地将匹配的路由排除在 initalized 中，这些路由不适用于测试场景或很有用。例如，您可能会在 'org.example.routes' 软件包中有一个 spring 上下文文件 routes-context.xml 和三个 Route builders RouteA、RouteB 和 RouteC。packageScan 定义将发现所有三个路由并初始化它们。

假设 RouteC 不适用于我们的测试场景，并在测试期间生成很多 noise。从这个特定测试中排除此路由会很好。SpringTestSupport 类已修改为允许这一点。它提供两种方法(excludedRoute 和 excludedRoutes)，它们可以被覆盖来排除单个类或类数组。

```
public class RouteAandRouteBOnlyTest extends SpringTestSupport {
    @Override
    protected Class excludeRoute() {
        return RouteC.class;
    }
}
```

为了让 hook 到 camelContext 初始化中，spring 会排除 MyExcludedRouteBuilder.class，我们需要截获 spring 上下文创建。当覆盖 createApplicationContext 以创建 spring 上下文时，我们调用 getRouteExcludingApplicationContext （）方法，以提供处理排除的特殊父 spring 上下文。

```
@Override
protected AbstractXmlApplicationContext createApplicationContext() {
    return new ClassPathXmlApplicationContext(new String[] {"routes-context.xml"},
getRouteExcludingApplicationContext());
}
```

现在，RouteC 将不包括在初始化中。同样，在只测试 RouteC 的另一个测试中，我们可以通过覆盖来排除 RouteB 和 RouteA

```
@Override
protected Class[] excludeRoutes() {
    return new Class[]{RouteA.class, RouteB.class};
}
```

318.4. 使用 SPRING XML

您可以使用 Spring 2.0 XML 配置为路由指定 Xml 配置，如下例所示。

318.5. 配置组件和端点

您可以在 Spring XML 中配置组件或 Endpoint 实例，如下例所示。

允许您使用某些名称（上例中的activemq）配置组件，然后使用 activemq:[queue:|topic:]destinationName 来引用组件。这由 SpringCamelContext lazily 从 spring 上下文获取用于 Endpoint URI 的方案名称的组件。

如需了解更多详细信息，请参阅配置端点和组件。

318.6. CAMELCONTEXTAWARE

如果您要使用 POJO 中的 CamelContext 注入，仅实施 CamelContextAware 接口 ；然后，当 Spring 创建 POJO 时，CamelContext 将注入到您的 POJO 中。另外，请参阅 Bean 集成 以进一步注入。

318.7. 集成测试

为了避免在使用 Spring Transactions 测试时挂起的路由，请参阅有关 Transactional Client 下的 Spring Integration Testing 的备注。

318.8. 另请参阅

- Spring JMS 教程
- 创建基于 Spring 的新 Camel 路由
- Spring 示例
- XML 参考
- 使用 Spring 进行 CamelContext 高级配置
- 如何从其他 XML 文件导入路由

第 319 章 SPRING BATCH 组件

从 Camel 版本 2.10 开始提供

spring-batch: 组件和支持类在 Camel 和 Spring Batch 基础架构之间提供集成网桥。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-spring-batch</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

319.1. URI 格式

```
spring-batch:jobName[?options]
```

其中 jobName 代表位于 Camel registry 中的 Spring 批处理作业的名称。或者，如果提供了 JobRegistry，它将用于查找该作业。

WARNING：此组件只能用于定义制作者端点，这意味着您无法在 from （） 语句中使用 Spring Batch 组件。

319.2. 选项

Spring Batch 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
jobLauncher (producer)	明确指定要使用的 JobLauncher。		JobLauncher
jobRegistry (producer)	明确指定要使用的 JobRegistry。		JobRegistry

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Spring Batch 端点使用 URI 语法进行配置：

```
spring-batch:jobName
```

使用以下路径和查询参数：

319.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
jobName	***必需*** 位于 registry 中的 Spring 批处理作业的名称。		字符串

319.2.2. 查询参数(4 参数)：

Name	描述	默认值	类型
jobFromHeader (producer)	明确定义 jobName 是否应该从标头获取，而不是 URI。	false	布尔值
jobLauncher (producer)	明确指定要使用的 JobLauncher。		JobLauncher
jobRegistry (producer)	明确指定要使用的 JobRegistry。		JobRegistry
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

319.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.spring-batch.enabled	启用 spring-batch 组件	true	布尔值
camel.component.spring-batch.job-launcher	明确指定要使用的 JobLauncher。选项是一个 org.springframework.batch.core.launch.JobLauncher 类型。		字符串
camel.component.spring-batch.job-registry	明确指定要使用的 JobRegistry。选项是一个 org.springframework.batch.core.configuration.JobRegistry 类型。		字符串
camel.component.spring-batch.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

319.4. 使用方法

当 Spring Batch 组件收到消息时，它会触发作业执行。该作业将使用 org.springframework.batch.core.launch.JobLaucher 实例按照以下算法解析：

- *如果在组件上手动设置 JobLauncher，则使用它。*
- *如果在组件上设置了 jobLauncherRef 选项，则使用给定名称搜索 Camel Registry for the JobLauncher。弃用并将在 Camel 3.0 中删除！*
- *如果在 Camel Registry 中注册了 Job Launcher 名称，则使用它。*
- *如果以上步骤都不允许解析 JobLauncher，且 Camel Registry 中只有一个 JobLauncher 实例，则使用它。*

消息中找到的所有标头都作为作业参数传递给 JobLauncher。字符串,Long,Double 和 java.util.Date 值被复制到 org.springframework.batch.core.JobParametersBuilder - other 数据类型转换为 Strings。

319.5. 例子

触发 Spring Batch 作业执行：

```
from("direct:startBatch").to("spring-batch:myJob");
```

使用 JobLauncher 明确设置来触发 Spring Batch 作业执行。

```
from("direct:startBatch").to("spring-batch:myJob?jobLauncherRef=myJobLauncher");
```

从 JobLauncher 返回的 Camel 2.11.1 JobExecution 实例开始由 SpringBatchProducer 作为输出消息转发。您可以使用 JobExecution 实例直接通过 Spring Batch API 执行一些操作。

```
from("direct:startBatch").to("spring-batch:myJob").to("mock:JobExecutions");
...
MockEndpoint mockEndpoint = ...;
JobExecution jobExecution =
mockEndpoint.getExchanges().get(0).getIn().getBody(JobExecution.class);
BatchStatus currentJobStatus = jobExecution.getStatus();
```

319.6. 支持类

除了组件外，Camel Spring Batch 还支持类，它们可用于 hook 到 Spring Batch 基础架构中。

319.6.1. CamelItemReader

CamelItemReader 可用于直接从 Camel 基础架构读取批处理数据。

例如，以下代码片段将 Spring Batch 配置为从 JMS 队列读取数据。

```
<bean id="camelReader"
class="org.apache.camel.component.spring.batch.support.CamelItemReader">
  <constructor-arg ref="consumerTemplate"/>
  <constructor-arg value="jms:dataQueue"/>
</bean>

<batch:job id="myJob">
  <batch:step id="step">
    <batch:tasklet>
      <batch:chunk reader="camelReader" writer="someWriter" commit-interval="100"/>
```

```
    </batch:tasklet>
  </batch:step>
</batch:job>
```

319.6.2. CamelItemWriter

CamelItemWriter 具有类似 CamelItemReader 的目的，但它专用于写入处理的数据的块。

例如，以下代码片段将 Spring Batch 配置为从 JMS 队列读取数据。

```
<bean id="camelwriter" class="org.apache.camel.component.spring.batch.support.CamelItemWriter">
  <constructor-arg ref="producerTemplate"/>
  <constructor-arg value="jms:dataQueue"/>
</bean>

<batch:job id="myJob">
  <batch:step id="step">
    <batch:tasklet>
      <batch:chunk reader="someReader" writer="camelwriter" commit-interval="100"/>
    </batch:tasklet>
  </batch:step>
</batch:job>
```

319.6.3. CamelItemProcessor

CamelItemProcessor 是 Spring Batch org.springframework.batch.item.ItemProcessor 接口的实现。在 Request Reply 模式 上的后一种实施中继，将批处理项目的处理委派给 Camel 基础架构。要处理的项目作为消息的正文发送到 Camel 端点。

例如，以下代码片段使用 Direct 端点和 Simple 表达式语言 执行批处理项目的简单处理。

```
<camel:camelContext>
  <camel:route>
    <camel:from uri="direct:processor"/>
    <camel:setExchangePattern pattern="InOut"/>
    <camel:setBody>
      <camel:simple>Processed ${body}</camel:simple>
    </camel:setBody>
  </camel:route>
</camel:camelContext>

<bean id="camelProcessor"
class="org.apache.camel.component.spring.batch.support.CamelItemProcessor">
  <constructor-arg ref="producerTemplate"/>
  <constructor-arg value="direct:processor"/>
</bean>
```

```
<batch:job id="myJob">
  <batch:step id="step">
    <batch:tasklet>
      <batch:chunk reader="someReader" writer="someWriter" processor="camelProcessor" commit-
interval="100"/>
    </batch:tasklet>
  </batch:step>
</batch:job>
```

319.6.4. CamelJobExecutionListener

CamelJobExecutionListener 是 org.springframework.batch.core.JobExecutionListener 接口将作业执行事件发送到 Camel 端点的实现。

由 Spring Batch 生成的 org.springframework.batch.core.JobExecution 实例作为消息的正文发送。为了区分 before- 和 after-callbacks SPRING_BATCH_JOB_EVENT_TYPE 标头被设置为 BEFORE 或 AFTER 值。

以下示例片断将 Spring Batch 任务执行事件发送到 JMS 队列。

```
<bean id="camelJobExecutionListener"
class="org.apache.camel.component.spring.batch.support.CamelJobExecutionListener">
  <constructor-arg ref="producerTemplate"/>
  <constructor-arg value="jms:batchEventsBus"/>
</bean>

<batch:job id="myJob">
  <batch:step id="step">
    <batch:tasklet>
      <batch:chunk reader="someReader" writer="someWriter" commit-interval="100"/>
    </batch:tasklet>
  </batch:step>
  <batch:listeners>
    <batch:listener ref="camelJobExecutionListener"/>
  </batch:listeners>
</batch:job>
```

319.7. SPRING CLOUD

从 Camel 2.19 开始提供

Spring Cloud 组件

Maven 用户需要将以下依赖项添加到其 pom.xml 中，以便使用此组件：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-spring-cloud</artifactId>
  <version>${camel.version}</version> <!-- use the same version as your Camel core version --
>
</dependency>
```

camel-spring-cloud jar 附带 spring.factories 文件，因此当您将该依赖项添加到类路径时，Spring Boot 将自动为您配置 Camel。

319.7.1. Camel Spring Cloud Starter

从 Camel 2.19 开始提供

要使用启动程序，将以下内容添加到 spring boot pom.xml 文件中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-spring-cloud-starter</artifactId>
  <version>${camel.version}</version> <!-- use the same version as your Camel core version --
>
</dependency>
```

319.8. SPRING CLOUD CONSUL

从 Camel 2.22 开始提供

319.9. SPRING CLOUD ZOOKEEPER

从 Camel 2.22 开始提供

319.10. SPRING CLOUD NETFLIX

从 Camel 2.19 开始提供

Spring Cloud Netflix 组件桥接 Camel Cloud 和 Spring Cloud Netflix，因此您可以利用 Spring

Cloud Netflix 服务发现服务发现服务发现功能，并使用 Camel Service Discovery 实现作为 Spring Cloud Netflix 的 Ribbon 负载 balabncer 的 ServerList 源。

Maven 用户需要将以下依赖项添加到其 pom.xml 中，以便使用此组件：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-spring-cloud-netflix</artifactId>
  <version>${camel.version}</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

camel-spring-cloud-netflix jar 附带 spring.factories 文件，因此当您将该依赖项添加到 classpath 时，Spring Boot 将自动为您配置 Camel。

您可以使用以下属性禁用 Camel Spring Cloud Netflix：

```
# Enable/Disable the whole integration, default true
camel.cloud.netflix = true

# Enable/Disable the integration with Ribbon, default true
camel.cloud.netflix.ribbon = true
```

319.11. SPRING CLOUD NETFLIX STARTER

从 Camel 2.19 开始提供

要使用启动程序，将以下内容添加到 spring boot pom.xml 文件中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-spring-cloud-netflix-starter</artifactId>
  <version>${camel.version}</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

第 320 章 SPRING EVENT 组件

从 Camel 版本 1.4 开始提供

spring-event: 组件提供对 Spring ApplicationEvent 对象的访问。这可让您将 ApplicationEvent 对象发布到 Spring ApplicationContext 或使用它们。然后，您可以使用 企业集成模式 来处理它们，如 Message Filter。

320.1. URI 格式

```
spring-event://default[?options]
```

请注意，目前没有此组件的选项。这可以在将来的版本中轻松更改，因此请重新检查。

320.2. SPRING 事件选项

Spring 事件组件没有选项。

Spring Event 端点使用 URI 语法进行配置：

```
spring-event:name
```

使用以下路径和查询参数：

320.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	端点名称		字符串

320.2.2. 查询参数(4 参数)：

Name	描述	默认值	类型

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

320.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.spring-event.enabled	启用 spring-event 组件	true	布尔值
camel.component.spring-event.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.language.spel.enabled	启用 spel 语言	true	布尔值
camel.language.spel.trim	是否修剪值以移除前导和结尾的空格和换行符	true	布尔值

320.4. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 321 章 SPRING INTEGRATION 组件

从 Camel 版本 1.4 开始提供

spring-integration: 组件为 Camel 组件提供与 spring 集成端点 通信的网桥。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-spring-integration</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

321.1. URI 格式

```
spring-integration:defaultChannelName[?options]
```

其中 defaultChannelName 代表 Spring Integration Spring 上下文使用的默认频道名称。它等同于 Spring Integration consumer 和 Spring Integration 供应商的 outputChannel 名称。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

321.2. 选项

Spring Integration 组件没有选项。

Spring Integration 端点使用 URI 语法进行配置：

```
spring-integration:defaultChannel
```

使用以下路径和查询参数：

321.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
defaultChannel	**必需** Spring Integration Spring 上下文使用的默认频道名称。它等同于 Spring Integration consumer 和 Spring Integration 供应商的 outputChannel 名称。		字符串

321.2.2. 查询参数(7 参数)：

Name	描述	默认值	类型
inOut (common)	Spring 集成端点应使用的交换模式。如果预期为 inOut=true，则可从 Spring Integration Message 标头或端点上配置回复频道。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
inputChannel (consumer)	此端点希望使用来自 Spring 集成使用的 Spring 集成输入频道名称。		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
outputChannel (producer)	用于向 Spring 集成发送消息的 Spring 集成输出频道名称。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

321.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.spring-integration.enabled	启用 spring-integration 组件	true	布尔值
camel.component.spring-integration.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

321.4. 使用方法

Spring 集成组件是一个桥接，它通过 Spring 集成的输入频道和输出频道将 Camel 端点与 Spring 集成端点连接。使用这个组件，我们可以将 Camel 消息发送到 Spring Integration 端点，或者在 Camel 路由上下文中接收来自 Spring 集成端点的消息。

321.5. 例子

321.5.1. 使用 Spring 集成端点

您可以使用 URI 设置 Spring 集成端点，如下所示：

或者直接使用 Spring integration 频道名称：

321.5.2. 源和目标适配器

Spring 集成还提供 Spring 集成的源和目标适配器，它可以将消息从 Spring 集成频道路由到 Camel 端点，或者从 Camel 端点路由到 Spring 集成频道。

这个示例使用以下命名空间：

您可以将源或目标绑定到 Camel 端点，如下所示：

321.6. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

321.7. SPRING JAVA CONFIG

Spring started Life using XML Config to wire Bean 在一起。但是，某些人不像使用 XML，而是使用导致创建 Guice 与 Spring JavaConfig 项目的 Java 代码。

您可以将 XML 或 Java 配置方法与 Camel 一起使用；其选择实际上是您喜欢的。

321.7.1. 使用 Spring Java Config

要在 Camel 项目中使用 Spring Java Config，最简单的操作是将以下内容添加到 pom.xml 中

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-spring-javaconfig</artifactId>
  <version>${camel-version}</version>
</dependency>
```

然后，这将添加 Spring JavaConfig 库的依赖项以及用于在 Spring 中配置 Camel 的一些帮助程序类。

请注意，这个库完全是可选的；您可以只将 Camel 与 Java 配置连接在一起。

321.7.2. 配置

将 JavaConfig 与 Camel 搭配使用的最常见情况是，使用由路由器使用的路由列表来创建配置。

```
@Configuration
public class MyRouteConfiguration extends CamelConfiguration {

    @Autowire
    private MyRouteBuilder myRouteBuilder;

    @Autowire
    private MyAnotherRouteBuilder myAnotherRouteBuilder;

    @Override
    public List<RouteBuilder> routes() {
        return Arrays.asList(myRouteBuilder, myAnotherRouteBuilder);
    }

}
```

从 Camel 2.13.0 开始，您可以跳过 routes （） 定义，再回退到 Spring 上下文中的 RouteBuilder 实例。

```
@Configuration
@ComponentScan("com.example.routes")
public class MyRouteConfiguration extends CamelConfiguration {
}
```

321.7.3. 测试

自 Camel 2.11.0 起，您可以使用 CamelSpringJUnit4ClassRunner 和 CamelSpringDelegatingTestContextLoader。这是测试 Java 配置和 Camel 集成的建议方法。

如果要创建 RouteBuilder 实例的集合，则从 CamelConfiguration 帮助程序类派生并实施 routes （） 方法。请记住，如果没有覆盖 routes （） 方法，则（从 Camel 2.13.0 开始），则 CamelConfiguration 将使用 Spring 上下文中提供的所有 RouteBuilder 实例。

以下示例使用 Java 配置示例 演示了如何测试与 Camel 2.10 和更低的 Java 配置集成。请记住，JavaConfigContextLoader 已被弃用，可代表 Camel SpringDelegatingTestContextLoader 的未来版本删除。

@ContextConfiguration 注释告知 Spring Testing 框架来加载 ContextConfig 类，作为要使用的配置。此类从 SingleRouteCamelConfiguration 派生而来，它是一个帮助程序 Spring Java Config 类，它将为我们配置 CamelContext，然后注册我们创建的 RouteBuilder。

第 322 章 SPRING LDAP 组件

从 Camel 版本 2.11 开始提供

spring-ldap: 组件为 Spring LDAP 提供 Camel 包装器。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-spring-ldap</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

322.1. URI 格式

```
spring-ldap:springLdapTemplate[?options]
```

其中 springLdapTemplate 是 Spring LDAP Template bean 的名称。在这个 bean 中，您要为 LDAP 访问配置 URL 和凭证。

322.2. 选项

Spring LDAP 组件没有选项。

Spring LDAP 端点使用 URI 语法进行配置：

```
spring-ldap:templateName
```

使用以下路径和查询参数：

322.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
templateName	Spring LDAP 模板 Bean **所需的** 名称		字符串

322.2.2. 查询参数(3 参数)：

Name	描述	默认值	类型
operation (producer)	需要执行 LDAP 操作。		LdapOperation
scope (producer)	搜索操作的范围。	subtree	字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

322.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.spring-ldap.enabled	启用 spring-ldap 组件	true	布尔值
camel.component.spring-ldap.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

322.4. 使用方法

组件仅支持 producer 端点。尝试创建消费者端点将导致 UnsupportedOperationException。消息的正文必须是映射(java.util.Map实例)。除非在 ContextSource 配置中指定基本 DN，否则此映射必须至少包含一个带有键 dn （对 function_driven 操作不需要）的条目，用来指定执行 LDAP 操作的 root 节点。映射的其他条目是特定于操作的（请参阅以下）。

对于 bind 和 unbind 操作，消息的正文保持不变。有关 search 和 function_driven 操作，正文被设置为搜索的结果，请参阅 http://static.springsource.org/spring-ldap/site/apidocs/org/springframework/ldap/core/LdapTemplate.html#search%28java.lang.String,%20java.lang.String,%20int,%20org.springframework.ldap.core.AttributesMapper%29。

322.4.1. 搜索

消息正文必须具有一个带有密钥 过滤器 的条目。该值必须是代表有效 LDAP 过滤器的 String，请参阅 http://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol#Search_and_Compare。

322.4.2. 绑定

消息正文必须具有一个带有关键 属性 的条目。该值必须是 javax.naming.directory.Attributes 此条目的实例，指定要创建的 LDAP 节点。

322.4.3. unbind

不需要其他条目，会删除带有指定 dn 的节点。

322.4.4. 身份验证

消息正文必须具有带有键 过滤器 和密码 的条目。该值必须是 String 实例，分别代表有效的 LDAP 过滤器和用户密码。

322.4.5. 修改属性

消息正文必须具有带有密钥 modify Items 的条目。该值必须是任何类型为 javax.naming.directory.ModificationItem的数组的实例

322.4.6. function-Driven

消息正文必须具有带有密钥函数和 请求 的条目。功能 值必须是 java.util.function.BiFunction<L, Q, S > 类型。L type 参数必须是 org.springframework.ldap.core.LdapOperations。request 值必须与函数中的 Q type 参数相同，它必须封装 函数 中调用的 LdapTemplate 方法所预期的参数。 S type 参数代表被调用的 LdapTemplate 方法返回的响应类型。此操作允许动态调用上述操作未涵盖的 LdapTemplate 方法。

键定义

为避免拼写错误，在 org.apache.camel.springldap.springLdapProducer 中定义以下常量：

- 公共静态最终字符串 DN = "dn"

- *公共静态最终字符串 FILTER = "filter"*
- *公共静态最终字符串 ATTRIBUTES = "attributes"*
- *公共静态最终字符串 PASSWORD = "password";*
- *public static final String MODIFICATION_ITEMS = "modificationItems";*
- *公共静态最终字符串 FUNCTION = "function";*
- *公共静态最终字符串 REQUEST = "request";*

第 323 章 SPRING REDIS 组件

从 Camel 版本 2.11 开始提供

此组件允许从 Redis 发送和接收信息。Redis 是高级键值存储，其中键可以包含字符串、散列、列表、集合和排序的集合。此外，它还为应用间通信提供 pub/sub 功能。
Camel 提供了一个制作者，用于执行命令，使用者用于订阅 pub/sub 消息，这是一个幂等存储库，用于过滤重复的消息。

INFO:*Prerequisites* 用来使用这个组件，您必须有一个 Redis 服务器正在运行。

323.1. URI 格式

```
spring-redis://host:port[?options]
```

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

323.2. URI 选项

Spring Redis 组件没有选项。

Spring Redis 端点使用 URI 语法进行配置：

```
spring-redis:host:port
```

使用以下路径和查询参数：

323.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
主机	**必需** 运行 Redis 服务器的主机。		字符串
port	**所需的** Redis 服务器端口号		整数

323.2.2. 查询参数(10 parameters):

Name	描述	默认值	类型
channels (common)	要订阅的主题名称或名称模式列表。可以使用逗号分隔多个名称。		字符串
command (common)	默认命令，可以被消息标头覆盖。请注意，消费者只支持以下命令： PSUBSCRIBE 和 SUBSCRIBE	SET	命令
ConnectionFactory (common)	引用要使用的预配置 RedisConnectionFactory 实例。		RedisConnectionFactory
redisTemplate (common)	引用要使用的预先配置的 RedisTemplate 实例。		RedisTemplate
serializer (common)	引用要使用的预先配置的 RedisSerializer 实例。		RedisSerializer
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
listenerContainer (consumer)	引用要使用的预配置 RedisMessageListenerContainer 实例。		RedisMessageListener Container
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

323.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.spring-redis.enabled	启用 spring-redis 组件	true	布尔值
camel.component.spring-redis.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

323.4. 使用方法

另请参阅位于 https://github.com/apache/camel/tree/master/components/camel-spring-redis/src/test/java/org/apache/camel/component/redis 的单元测试。

323.4.1. 由 Redis producer 评估的消息标头

生产者向服务器发出命令，每个命令具有特定类型的不同参数集合。命令执行的结果在消息正文中返回。

hash 命令	描述	参数	结果
HSET	设置散列字段的字符串值	CamelRedis.Key (String), CamelRedis.Field (String), CamelRedis.Value (Object)	void
HGET	获取 hash 字段的值	CamelRedis.Key (String), CamelRedis.Field (String)	字符串
HSETNX	设置 hash 字段的值，只有在字段不存在时	CamelRedis.Key (String), CamelRedis.Field (String), CamelRedis.Value (Object)	void

hash 命令	描述	参数	结果
HMSET	将多个散列字段设置为多个值	CamelRedis.Key (String), CamelRedis.Values (Map<String, Object>)	void
HMGET	获取所有给定哈希字段的值	CamelRedis.Key (String), CamelRedis.Fields (Collection<String>)	collection<Object>
HINCRBY	按给定数字递增 hash 字段的整数值	CamelRedis.Key (String), CamelRedis.Field (String), CamelRedis.Value (Long)	Long
HEXISTS	确定是否存在哈希字段	CamelRedis.Key (String), CamelRedis.Field (String)	布尔值
HDEL	删除一个或多个哈希字段	CamelRedis.Key (String), CamelRedis.Field (String)	void
HLEN	获取哈希中的字段数	CamelRedis.Key (String)	Long
HKEYS	获取哈希中的所有字段	CamelRedis.Key (String)	set<String>
HVALS	获取哈希中的所有值	CamelRedis.Key (String)	collection<Object>
HGETALL	获取哈希中的所有字段和值	CamelRedis.Key (String)	Map<String, Object>

列出命令	描述	参数	结果

列出命令	描述	参数	结果
RPUSH	在列表中添加一个或多个值	CamelRedis.Key (String), CamelRedis.Value (Object)	Long
RPUSHX	只有在列表存在时，才会将值附加到列表	CamelRedis.Key (String), CamelRedis.Value (Object)	Long
LPUSH	为列表添加一个或多个值	CamelRedis.Key (String), CamelRedis.Value (Object)	Long
LLEN	获取列表的长度	CamelRedis.Key (String)	Long
LRANGE	从列表获取一系列元素	CamelRedis.Key (String), CamelRedis.Start (Long), CamelRedis.End (Long)	List<Object>
LTRIM	将列表修剪到指定的范围	CamelRedis.Key (String), CamelRedis.Start (Long), CamelRedis.End (Long)	void
LINDEX	通过索引从列表获取元素	CamelRedis.Key (String), CamelRedis.Index (Long)	字符串
LINSERT	在列表中的另一个元素之前或之后插入元素	CamelRedis.Key (String), CamelRedis.Value (Object), CamelRedis.Pivot (String), CamelRedis.Position (String)	Long

列出命令	描述	参数	结果
LSET	根据索引在列表中设置元素值	CamelRedis.Key (String), CamelRedis.Value (Object), CamelRedis.Index (Long)	void
LREM	从列表中选择元素	CamelRedis.Key (String), CamelRedis.Value (Object), CamelRedis.Count (Long)	Long
LPOP	删除并获取列表中的第一个元素	CamelRedis.Key (String)	对象
RPOP	删除并获取列表中的最后一个元素	CamelRedis.Key (String)	字符串
RPOPLPUSH	删除列表中的最后一个元素，将其附加到另一个列表并返回它	CamelRedis.Key (String), CamelRedis.Destination (String)	对象
BRPOPLPUSH	从列表中弹出一个值，将其推送到另一个列表并返回它；或 block，直到有可用状态	CamelRedis.Key (String), CamelRedis.Destination (String), CamelRedis.Timeout (Long)	对象
BLPOP	删除并获取列表中的第一个元素，或 block 直到可用为止	CamelRedis.Key (String), CamelRedis.Timeout (Long)	对象
BRPOP	删除并获取列表中的最后一个元素，或 block 直到可用为止	CamelRedis.Key (String), CamelRedis.Timeout (Long)	字符串

设置命令	描述	参数	结果
SADD	为集合添加一个或多个成员	CamelRedis.Key (String), CamelRedis.Value (Object)	布尔值
SMEMBERS	获取集合中的所有成员	CamelRedis.Key (String)	set<Object>
SREM	从集合中删除一个或多个成员	CamelRedis.Key (String), CamelRedis.Value (Object)	布尔值
SPOP	从集合中删除并返回随机成员	CamelRedis.Key (String)	字符串
SMOVE	将成员从一个集合移到另一个集合	CamelRedis.Key (String), CamelRedis.Value (Object), CamelRedis.Destination (String)	布尔值
SCARD	获取集合中成员数量	CamelRedis.Key (String)	Long
SISMEMBER	确定给定值是否为集合的成员	CamelRedis.Key (String), CamelRedis.Value (Object)	布尔值
SINTER	交集多个集合	CamelRedis.Key (String), CamelRedis.Keys (String)	set<Object>
SINTERSTORE	交集多个集合，并将结果集存储在密钥中	CamelRedis.Key (String), CamelRedis.Keys (String), CamelRedis.Destination (String)	void

设置命令	描述	参数	结果
SUNION	添加多个集合	CamelRedis.Key (String), CamelRedis.Keys (String)	set<Object>
SUNIONSTORE	添加多个集合并将生成的集合存储在密钥中	CamelRedis.Key (String), CamelRedis.Keys (String), CamelRedis.Destination (String)	void
SDIFF	减去多个集合	CamelRedis.Key (String), CamelRedis.Keys (String)	set<Object>
SDIFFSTORE	减去多个集合，并将生成的集合存储在键中	CamelRedis.Key (String), CamelRedis.Keys (String), CamelRedis.Destination (String)	void
SRANDMEMBER	从集合中获取一个或多个随机成员	CamelRedis.Key (String)	字符串

排序的设置命令	描述	参数	结果
ZADD	为排序集添加一个或多个成员，或者更新其分数（如果已存在）	CamelRedis.Key (String), CamelRedis.Value (Object), CamelRedis.Score (Double)	布尔值
ZRANGE	按索引返回排序集合中的成员范围	CamelRedis.Key (String), CamelRedis.Start (Long), CamelRedis.End (Long), CamelRedis.WithScore (Boolean)	对象

排序的设置命令	描述	参数	结果
ZREM	从排序的集合中删除一个或多个成员	CamelRedis.Key (String), CamelRedis.Value (Object)	布尔值
ZINCRBY	以排序集递增成员分数	CamelRedis.Key (String), CamelRedis.Value (Object), CamelRedis.Increment (Double)	å
ZRANK	确定排序集合中成员的索引	CamelRedis.Key (String), CamelRedis.Value (Object)	Long
ZREVRANK	确定排序集合中成员的索引，分数从高到低	CamelRedis.Key (String), CamelRedis.Value (Object)	Long
ZREVRANGE	按索引返回排序集合中的一系列成员，分数从高到低	CamelRedis.Key (String), CamelRedis.Start (Long), CamelRedis.End (Long), CamelRedis.WithScore (Boolean)	对象
ZCARD	获取排序集合中的成员数量	CamelRedis.Key (String)	Long
ZCOUNT	计算在给定值中使用分数设置的排序中的成员	CamelRedis.Key (String), CamelRedis.Min (Double), CamelRedis.Max (Double)	Long
ZRANGEBYSCORE	以 score 形式返回排序集合中的成员范围	CamelRedis.Key (String), CamelRedis.Min (Double), CamelRedis.Max (Double)	set<Object>

排序的设置命令	描述	参数	结果
ZREVRANGEBYSCORE	返回按分数排序集合中的一系列成员，分数从高到低	CamelRedis.Key (String), CamelRedis.Min (Double), CamelRedis.Max (Double)	set<Object>
ZREMRANGEBYRANK	删除给定索引中的排序集合中的所有成员	CamelRedis.Key (String), CamelRedis.Start (Long), CamelRedis.End (Long)	void
ZREMRANGEBYSCORE	删除给定分数内排序集合中的所有成员	CamelRedis.Key (String), CamelRedis.Start (Long), CamelRedis.End (Long)	void
ZUNIONSTORE	添加多个排序的集合，并将生成的排序集存储在新键中	CamelRedis.Key (String), CamelRedis.Keys (String), CamelRedis.Destination (String)	void
ZINTERSTORE	交集多个排序的集合，并将生成的排序集存储在新键中	CamelRedis.Key (String), CamelRedis.Keys (String), CamelRedis.Destination (String)	void

字符串命令	描述	参数	结果
SET	设置键的字符串值	CamelRedis.Key (String), CamelRedis.Value (Object)	void
GET	获取键的值	CamelRedis.Key (String)	对象

字符串命令	描述	参数	结果
STRLEN	获取存储在键中的值的长度	CamelRedis.Key (String)	Long
附加	将值附加到键	CamelRedis.Key (String), CamelRedis.Value (String)	整数
SETBIT	在 key 中存储的字符串值中的偏移量设置或清除位	CamelRedis.Key (String), CamelRedis.Offset (Long), CamelRedis.Value (Boolean)	void
GETBIT	返回存储在键的字符串值中的偏移值	CamelRedis.Key (String), CamelRedis.Offset (Long)	布尔值
SETRANGE	覆盖从指定偏移开始的键的一部分	CamelRedis.Key (String), CamelRedis.Value (Object), CamelRedis.Offset (Long)	void
GETRANGE	获取存储在键的字符串的子字符串	CamelRedis.Key (String), CamelRedis.Start (Long), CamelRedis.End (Long)	字符串
SETNX	设置键的值，只有在键不存在时才	CamelRedis.Key (String), CamelRedis.Value (Object)	布尔值
SETEX	设置键的值和过期	CamelRedis.Key (String), CamelRedis.Value (Object), CamelRedis.Timeout (Long), SECONDS	void

字符串命令	描述	参数	结果
DECRBY	按给定数字减少键的值	CamelRedis.Key (String), CamelRedis.Value (Long)	Long
DECR	逐一减少键的整数值	CamelRedis.Key (String),	Long
INCRBY	按给定数量递增键的整数值	CamelRedis.Key (String), CamelRedis.Value (Long)	Long
INCR	逐个递增键的整数值	CamelRedis.Key (String)	Long
MGET	获取所有给定键的值	camelRedis.Fields (Collection<String>)	List<Object>
MSET	将多个键设置为多个值	CamelRedis.Values (Map<String, Object>)	void
MSETNX	将多个键设置为多个值，只有在所有键都不存在时才	CamelRedis.Key (String), CamelRedis.Value (Object)	void
GETSET	设置键的字符串值并返回其旧值	CamelRedis.Key (String), CamelRedis.Value (Object)	对象

键命令	描述	参数	结果
EXISTS	确定是否存在密钥	CamelRedis.Key (String)	布尔值
DEL	删除密钥	CamelRedis.Keys （字符串）	void
TYPE	确定保存在密钥中的类型	CamelRedis.Key (String)	DataType

键命令	描述	参数	结果
KEYS	查找与给定模式匹配的所有键	camelRedis.Pattern (String)	collection<String>
RANDOMKEY	从键空间中返回一个随机键	CamelRedis.Pattern (String), CamelRedis.Value (String)	字符串
RENAME	重命名密钥	CamelRedis.Key (String)	void
RENAMENX	仅在新密钥不存在时重命名密钥	CamelRedis.Key (String), CamelRedis.Value (String)	布尔值
EXPIRE	将密钥的时间设置为 live （以秒为单位）	CamelRedis.Key (String), CamelRedis.Timeout (Long)	布尔值
SORT	对列表中的元素、设置或排序集进行排序	CamelRedis.Key (String)	List<Object>
PERSIST	从密钥中删除过期时间	CamelRedis.Key (String)	布尔值
EXPIREAT	将密钥的过期时间设置为 UNIX 时间戳	CamelRedis.Key (String), CamelRedis.Timestamp (Long)	布尔值
PEXPIRE	以毫秒为单位将密钥时间设置为 live	CamelRedis.Key (String), CamelRedis.Timeout (Long)	布尔值
PEXPIREAT	将密钥的过期时间设置为以毫秒为单位指定的 UNIX 时间戳	CamelRedis.Key (String), CamelRedis.Timestamp (Long)	布尔值
TTL	获取密钥生存时间	CamelRedis.Key (String)	Long

键命令	描述	参数	结果
MOVE	将密钥移动到另一个数据库	CamelRedis.Key (String), CamelRedis.Db (Integer)	布尔值

其他命令	描述	参数	结果
MULTI	标记事务块的开头	none	void
DISCARD	丢弃 MULTI 后发出的所有命令	none	void
EXEC	执行 MULTI 后发布的所有命令	none	void
WATCH	观察给定键以确定 MULTI/EXEC 块的执行	CamelRedis.Keys （字符串）	void
UNWATCH	忘记所有监视的密钥	none	void
ECHO	回显给定字符串	CamelRedis.Value （字符串）	字符串
PING	Ping 服务器	none	字符串
QUIT	关闭连接	none	void
PUBLISH	将消息发布到频道	CamelRedis.Channel (String), CamelRedis.Message (Object)	void

323.5. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-spring-redis</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 ${camel-version} 必须替换为 Camel 的实际版本(2.11 或更高版本)。

323.6. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 324 章 SPRING SECURITY

从 Camel 2.3 开始提供

camel-spring-security 组件为 Camel 路由提供基于角色的授权。它利用 Spring Security （以前称为 Acegi Security）提供的身份验证和用户服务，并添加声明性、基于角色的策略系统来控制路由是否可以被给定主体执行。

如果您不熟悉 Spring Security 身份验证和授权系统，请查看上面链接的 SpringSource 网站中的当前参考文档。

324.1. 创建授权策略

对路由的访问由 SpringSecurityAuthorizationPolicy 对象的实例控制。策略对象包含运行一组端点和对 Spring Security AuthenticationManager 和 AccessDecisionManager 对象的引用所需的 Spring Security authority (role)的名称，用于确定当前主体是否已分配该角色。策略对象可以配置为 Spring Bean，也可以使用 Spring XML 中的 &lt ;authorizationPolicy& gt; 元素进行配置。

& lt;authorizationPolicy > 元素可能包含以下属性：

Name	默认值	描述
id	**null**	唯一的 Spring bean 标识符，用于在路由（必需）中引用策略
access	**null**	传递给访问决策管理器的 Spring Security authority 名称（必需）
authenticationManager	**authenticationManager**	上下文中 Spring Security **AuthenticationManager** 对象的名称
accessDecisionManager	**accessDecisionManager**	上下文中 Spring Security **AccessDecisionManager** 对象的名称

Name	默认值	描述
authenticationAdapter	Default Authentication Adapter	Camel 2.4 上下文中 camel-spring-security**AuthenticationAdapter** 对象的名称，用于将 **javax.security.auth.Subject** 转换为 Spring Security **Authentication** 实例。
useThreadSecurityContext	**true**	如果 Exchange.AUTHENTICATION 下的 In 消息标头中无法找到 **javax.security.auth.Subject**，请检查 **Authentication** 对象的 Spring Security **SecurityContextHolder**。
alwaysReauthenticate	**false**	如果设置为 true，则每次 **访问策略** 时都会调用 **AuthenticationManager.authenticate （）**。

324.2. 控制对 CAMEL 路由的访问

需要 Spring Security AuthenticationManager 和 AccessDecisionManager 来使用此组件。以下是如何使用 Spring Security 命名空间在 Spring XML 中配置这些对象的示例：

现在，已设置了底层的安全对象，我们可以使用它们来配置授权策略，并使用该策略来控制对路由的访问：

在本例中，不会执行端点 mock:end，除非经过验证或可以进行身份验证的 Spring Security Authentication 对象，并且包含 ROLE_ADMIN 授权可由 admin SpringSecurityAuthorizationPolicy 找到。

324.3. 身份验证

此组件没有指定用于授权的安全凭证的过程。您可以编写自己的处理器或组件，这些处理器或组件从交换中获取身份验证信息，具体取决于您的需要。例如，您可以创建一个处理器，从来自 Jetty 组件的 HTTP 请求标头获取凭证。无论如何收集凭证，都需要将它们放在 In 消息或 SecurityContextHolder 中，以便 Camel Spring Security 组件可以访问它们：

```
import javax.security.auth.Subject;
import org.apache.camel.*;
import org.apache.commons.codec.binary.Base64;
import org.springframework.security.authentication.*;
```

```
public class MyAuthService implements Processor {
   public void process(Exchange exchange) throws Exception {
      // get the username and password from the HTTP header
      // http://en.wikipedia.org/wiki/Basic_access_authentication
      String userpass = new
String(Base64.decodeBase64(exchange.getIn().getHeader("Authorization", String.class)));
      String[] tokens = userpass.split(":");

      // create an Authentication object
      UsernamePasswordAuthenticationToken authToken = new
UsernamePasswordAuthenticationToken(tokens[0], tokens[1]);

      // wrap it in a Subject
      Subject subject = new Subject();
      subject.getPrincipals().add(authToken);

      // place the Subject in the In message
      exchange.getIn().setHeader(Exchange.AUTHENTICATION, subject);

      // you could also do this if useThreadSecurityContext is set to true
      // SecurityContextHolder.getContext().setAuthentication(authToken);
   }
}
```

如果需要，SpringSecurityAuthorizationPolicy 会自动验证 Authentication 对象。

在使用 SecurityContextHolder 而不是 Exchange.AUTHENTICATION 标头时，需要注意两个问题。首先，上下文拥有者使用线程本地变量来存放 Authentication 对象。任何跨线程边界（如 seda 或 jms ）的路由都将丢失 身份验证 对象。其次，Spring Security 系统似乎预期上下文中的 Authentication 对象已经通过身份验证，并具有角色（更多详情请参阅 Technical Overview 部分 5.3.1 ）。

camel-spring-security 的默认行为是在 Exchange.AUTHENTICATION 标头中查找 Subject。此主题必须至少包含一个主体，它必须是 org.springframework.security.core.Authentication 的子类。您可以通过向 < authorizationPolicy & gt; bean 提供 org.apache.camel.component.spring.security.AuthenticationAdapter 的实现来自定义 Subject 到 Authentication 对象的映射。如果您使用不使用 Spring Security 的组件，但确实会提供 Subject，这非常有用。目前，只有 CXF 组件会填充 Exchange.AUTHENTICATION 标头。

324.4. 处理身份验证和授权错误

如果在 SpringSecurityAuthorizationPolicy 中进行身份验证或授权失败，则会抛出 CamelAuthorizationException。这可以通过使用 Camel 的标准异常处理方法（如 Exception Clause ）进行处理。CamelAuthorizationException 将具有对策略 ID 的引用，该策略重写异常，以便您可以根据策略以及异常类型处理错误：

```
<onException>
  <exception>org.springframework.security.authentication.AccessDeniedException</exception>
```

```
  <choice>
    <when>
      <simple>${exception.policyId} == 'user'</simple>
      <transform>
        <constant>You do not have ROLE_USER access!</constant>
      </transform>
    </when>
    <when>
      <simple>${exception.policyId} == 'admin'</simple>
      <transform>
        <constant>You do not have ROLE_ADMIN access!</constant>
      </transform>
    </when>
  </choice>
</onException>
```

324.5. 依赖项

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-spring-security</artifactId>
  <version>2.4.0</version>
</dependency>
```

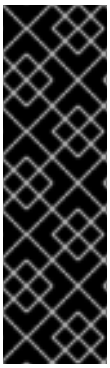
此依赖项也会在 org.springframework.security:spring-security-core:3.0.3.RELEASE 和 org.springframework.security:spring-security-config:3.0.3.RELEASE 中拉取。

324.6. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- 组件

第 325 章 SPRING WEBSERVICE 组件

从 Camel 版本 2.6 开始提供



重要

Red Hat JBoss Enterprise Application Platform (JBoss EAP)容器 不支持 Camel WebService (camel-spring-ws)组件。如果您需要在 JBoss EAP 容器上实施 JAX-WS 端点，建议的方法是使用 JBoss EAP 容器中内置的 webservice 子系统。这种方法会自动为您提供企业级的服务等级（包括管理、安全性等等）。如果您随后需要将 JAX-WS 端点与 Camel 路由集成，您可以在初始有效负载处理后将生成的 EndpointImpl bean 传递给 Camel 路由。

spring-ws: 组件允许您与 Spring Web Services 集成。它提供 客户端支持，用于访问 Web 服务和服务器端支持，以创建自己的合同优先 Web 服务。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-spring-ws</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

INFO:*Dependencies* As of Camel 2.8 this components shipped with Spring-WS 2.0.x which（如 Camel rest of Camel）需要 Spring 3.0.x。较早的 Camel 版本提供了 Spring-WS 1.5.9，它与 Spring 2.5.x 和 3.0.x 兼容。要在 Spring 2.5.x 上运行 camel-spring-ws 的早期版本，您需要从 Spring 2.5.x 添加 spring-webmvc 模块。要在 Spring 3.0.x 上运行 Spring-WS 1.5.9，您需要从 Spring 3.0.x 中排除 OXM 模块，因为此模块也包含在 Spring-WS 1.5.9 中（请参阅 此后文）

325.1. URI 格式

此组件的 URI 方案如下

```
spring-ws:[mapping-type:]address[?options]
```

要公开 web 服务 映射类型，需要设置为以下任意一种：

映射类型	描述
rootqname	提供根据消息中包含的 root 元素的合格名称映射 Web 服务请求的选项。
SOAP Action	用于根据邮件标题中指定的 SOAP 操作来映射 Web 服务请求。
uri	为了映射以特定 URI 为目标的 Web 服务请求。
xpathresult	用于根据 XPath **表达式评估** 针对传入消息映射 Web 服务请求。评估的结果应与端点 URI 中指定的 XPath 结果匹配。
Bean Name	允许您引用 **org.apache.camel.component.spring.ws.bean.CamelEndpointDispatcher** 对象，以便与现有的（传统）端点映射（传统） 端点映射 （如 **PayloadRootQNameEndpointMapping**、S**soapActionEndpointMapping** 等）集成

作为消费者，地址 应包含与指定映射类型相关的值（例如 SOAP 操作、XPath 表达式）。作为制作者，地址应设置为您调用的 Web 服务的 URI。

您可以在 URI 中附加 查询选项，格式为 ?option=value&option=value&...

325.2. 选项

Spring WebService 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Spring WebService 端点使用 URI 语法进行配置：

```
spring-ws:type:lookupKey:webServiceEndpointUri
```

使用以下路径和查询参数：

325.2.1. 路径参数(3 参数)：

Name	描述	默认值	类型
type	如果使用端点映射，则端点映射类型。rootqname - 提供根据消息中包含的根元素限定名称映射 Web 服务请求的选项。soapaction - 用于根据消息标题中指定的 SOAP 操作映射 Web 服务请求。uri - 顺序映射目标目标的 Web 服务请求。xpathresult - 用于根据消息标头中指定的 SOAP 操作来映射 Web 服务请求。评估的结果应与端点 URI. beanname - 允许您引用 org.apache.camel.component.spring.ws.bean.CamelEndpointDispatcher 对象，以便与现有（传统）端点映射（如 PayloadRootQNameEndpointMapping, SoapActionEndpointMapping, SoapActionEndpointMapping 等）集成		EndpointMappingType
lookupKey	如果使用端点映射，端点映射键		字符串
webServiceEndpointUri	用于生成者的默认 Web 服务端点 uri。		字符串

325.2.2. 查询参数(22 参数)：

Name	描述	默认值	类型
messageFilter (common)	提供自定义 MessageFilter 的选项。例如，当您要自行处理标头或附件时。		MessageFilter
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
endpointDispatcher (consumer)	Spring org.springframework.ws.server.endpoint.MessageEndpoint 用于分配 Spring-WS 收到的消息，以便与现有的（传统）端点映射（传统）端点映射（如 PayloadRootQNameEndpointMapping, SoapActionEndpointMapping 等）集成。		CamelEndpointDispatcher

Name	描述	默认值	类型
endpointMapping (consumer)	引用 Registry/ApplicationContext 中的 org.apache.camel.component.spring.ws.bean.CamelEndpointMapping 实例。registry 中只需要一个 bean 来提供所有 Camel/Spring-WS 端点。此 bean 由 MessageDispatcher 自动发现，用于根据端点上指定的特征（如根 QName、SOAP 操作等）将请求映射到 Camel 端点。		CamelSpringWSEndpoint 映射
expression (consumer)	使用 when 选项 type=xpathresult 的 XPath 表达式。然后，需要配置这个选项。		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
allowResponseAttachment Override (producer)	使用实际服务层中的附件覆盖 soap 响应附加选项。如果调用的服务附加或重写 soap attachments，当设为 true 时，允许修改的 soap 附加功能在/out 消息附加中被覆盖	false	布尔值
allowResponseHeaderOverride (producer)	使用实际服务层中的标头覆盖 soap 响应标头的选项。如果调用的服务在设置为 true 时附加或重写 soap 标头此选项，允许在/out 消息标头中覆盖修改后的 soap 标头	false	布尔值
faultAction (producer)	表示方法提供的 faultAction 响应 WS-Addressing Fault Action 标头的值。		URI
faultTo (producer)	表示方法提供的 faultAction 响应 WS-Addressing FaultTo 标头的值。		URI
messageFactory (producer)	提供自定义 WebServiceMessageFactory 的选项。例如，当您希望 Apache Axiom 处理 Web 服务消息而不是 SAAJ 时。		WebServiceMessage Factory
messageIdStrategy (producer)	提供自定义 MessageIdStrategy 来控制生成唯一消息 ID 的选项。		MessageIdStrategy
messageSender (producer)	提供自定义 WebServiceMessageSender 的选项。例如，要执行身份验证或使用其它传输		WebServiceMessage Sender
outputAction (producer)	表示方法提供的响应 WS-Addressing Action 标头的值。		URI

Name	描述	默认值	类型
replyTo (producer)	表示方法提供的 replyTo 响应 WS-Addressing ReplyTo 标头的值。		URI
SOAPAction (producer)	在访问远程 Web 服务时包括在 SOAP 请求中的 SOAP 操作		字符串
timeout (producer)	在使用 producer 调用 webservice 时设置套接字读取超时（以毫秒为单位），请参阅 URLConnection.setReadTimeout （）和 CommonsHttpMessageSender.setReadTimeout （）。在使用内置消息发送器实现时此选项可以正常工作： CommonsHttpMessageSender 和 HttpUrlConnectionMessageSender。除非自定义提供给组件的 Spring WS 配置选项，否则这些实施将默认使用 HTTP 基于 HTTP 的服务。如果您使用非标准发送方，则假设您将处理自己的超时配置。内置消息 sender HttpComponentsMessageSender 被考虑，而不是 CommonsHttpMessageSender （已弃用），请参阅 HttpComponentsMessageSender.setReadTimeout （）。		int
webServiceTemplate (producer)	提供自定义 WebServiceTemplate 的选项。这可以完全控制客户端侧 Web 服务处理；例如添加自定义拦截器或指定错误解析器、消息发送者或消息工厂。		WebServiceTemplate
wsAddressingAction (producer)	WS-Addressing 1.0 操作标头，包括在访问 Web 服务时。To 标头设置为端点 URI （默认的 Spring-WS 行为）中指定的 Web 服务的地址。		URI
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters

325.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.spring-ws.enabled	启用 spring-ws 组件	true	布尔值

Name	描述	默认值	类型
camel.component.spring-ws.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.spring-ws.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

325.3.1. 消息标头

Name	类型	描述
CamelSpringWebserviceEndpointUri	字符串	您作为客户端访问的 Web 服务的 URI，覆盖端点 URI 的地址部分
CamelSpringWebserviceSoapAction	字符串	标头指定消息的 SOAP 操作，如果存在，则覆盖 **soapAction** 选项
CamelSpringWebserviceSoapHeader	Source	**Camel 2.11.1：** 使用此标头指定/访问消息的 SOAP 标头。
CamelSpringWebserviceAddressingAction	URI	使用此标头指定消息的 WS-Addressing 操作，如果存在则覆盖 **wsAddressingAction** 选项

Name	类型	描述
CamelSpringWebserviceAddressingFaultTo	URI	使用此标头指定 WS-Addressing FaultTo，如果存在则覆盖 faultTo 选项
CamelSpringWebserviceAddressingReplyTo	URI	使用此标头指定 WS-Addressing ReplyTo，覆盖 replyTo 选项（如果存在）
CamelSpringWebserviceAddressingOutputAction	URI	使用此标头指定 WS-Addressing Action，如果存在则覆盖 outputAction 选项
CamelSpringWebserviceAddressingFaultAction	URI	使用此标头指定 WS-Addressing Fault Action，如果存在则覆盖 faultAction 选项

325.4. 访问 WEB 服务

要调用 http://foo.com/bar 的 Web 服务，只需定义一个路由：

```
from("direct:example").to("spring-ws:http://foo.com/bar")
```

发送一条信息：

```
template.requestBody("direct:example", "<foobar xmlns=\"http://foo.com\"><msg>test message</msg></foobar>");
```

请记住，如果这是您调用的 SOAP 服务，则不必包含 SOAP 标签。Spring-WS 将执行 XML-to-SOAP marshaling。

325.5. 发送 SOAP 和 WS-ADDRESSING 操作标头

当远程 Web 服务需要 SOAP 操作或使用 WS-Addressing 标准时，您可以将路由定义为：

```
from("direct:example")
.to("spring-ws:http://foo.com/bar?
soapAction=http://foo.com&wsAddressingAction=http://bar.com")
```

另外，您还可以使用标头值覆盖端点选项：

```
template.requestBodyAndHeader("direct:example",
"<foobar xmlns=\"http://foo.com\"><msg>test message</msg></foobar>",
SpringWebserviceConstants.SPRING_WS_SOAP_ACTION, "http://baz.com");
```

325.6. 使用 SOAP 标头

从 Camel 2.11.1 开始提供

在向 spring-ws 端点发送消息时，您可以提供 SOAP 标头作为 Camel Message 标头，例如在 String 中提供以下 SOAP 标头

```
String body = ...
String soapHeader = "<h:Header xmlns:h=\"http://www.webserviceX.NET/\">
<h:MessageID>1234567890</h:MessageID><h:Nested><h:NestedID>1111</h:NestedID>
</h:Nested></h:Header>";
```

我们可以在 Camel 消息上设置正文和标头，如下所示：

```
exchange.getIn().setBody(body);
exchange.getIn().setHeader(SpringWebserviceConstants.SPRING_WS_SOAP_HEADER,
soapHeader);
```

然后，将 Exchange 发送到 spring-ws 端点来调用 Web 服务。

同样，spring-ws consumer 也将使用 SOAP 标头增强 Camel 消息。

有关示例，请参见此 单元测试。

325.7. 标头和附加传播

自版本 2.10.3 起，Spring WS Camel 支持将标头和附件传播到 Spring-WS WebServiceMessage 响应中。端点将使用名为"hook"（默认实施由 BasicMessageFilter 提供）来传播交换标头和附件到 WebServiceMessage 响应。现在，您可以使用

```
exchange.getOut().getHeaders().put("myCustom","myHeaderValue")
exchange.getIn().addAttachment("myAttachment", new DataHandler(...))
```

注：如果管道中的交换标头包含文本，它会在 soap 标头中生成 Qname (key)=value 属性。建议直接创建一个 QName 类，并将其放在标头中。

325.8. 如何使用风格表转换 SOAP 标头

标头转换过滤器(HeaderTransformationMessageFilter.java)可用于转换 soap 请求的 soap 标头。如果要使用标头转换过滤器，请参阅以下示例：

```
<bean id="headerTransformationFilter"
class="org.apache.camel.component.spring.ws.filter.impl.HeaderTransformationMessageFilter">
   <constructor-arg index="0" value="org/apache/camel/component/spring/ws/soap-header-transform.xslt"/>
</bean
```

使用上述在 camel 端点中定义的 bean

```
<route>
   <from uri="direct:stockQuoteWebserviceHeaderTransformation"/>
   <to uri="spring-ws:http://localhost?webServiceTemplate=#webServiceTemplate&amp;soapAction=http://www.stockquotes.edu/GetQuote&amp;messageFilter=#headerTransformationFilter"/>
</route>
```

325.9. 如何使用 MTOM ATTACHMENTS

BasicMessageFilter 提供 Apache Axiom 的所有必要信息，以便生成 MTOM 消息。如果要在 Apache Axiom 中使用 Apache Camel Spring WS，以下是一个示例：- 简单地将 messageFactory 定义为 bellow，Spring-WS 将使用 MTOM 策略来填充带有优化附件的 SOAP 消息。

-

```
<bean id="axiomMessageFactory"
class="org.springframework.ws.soap.axiom.AxiomSoapMessageFactory">
<property name="payloadCaching" value="false" />
<property name="attachmentCaching" value="true" />
<property name="attachmentCacheThreshold" value="1024" />
</bean>
```

- 将以下依赖项添加到 pom.xml 中

```
<dependency>
<groupId>org.apache.ws.commons.axiom</groupId>
<artifactId>axiom-api</artifactId>
<version>1.2.13</version>
</dependency>
<dependency>
<groupId>org.apache.ws.commons.axiom</groupId>
<artifactId>axiom-impl</artifactId>
<version>1.2.13</version>
<scope>runtime</scope>
</dependency>
```

- 将您的附件添加到管道中，例如使用 Processor 实现。

```
private class Attachement implements Processor {
public void process(Exchange exchange) throws Exception
{ exchange.getOut().copyFrom(exchange.getIn()); File file = new File("testAttachment.txt");
exchange.getOut().addAttachment("test", new DataHandler(new FileDataSource(file)));  }
}
```

- 将 endpoint (producer)定义为 ussual，例如：

```
from("direct:send")
.process(new Attachement())
.to("spring-ws:http://localhost:8089/mySoapService?
soapAction=mySoap&messageFactory=axiomMessageFactory");
```

- 现在，您的生成者将使用 otpmized attachments 生成 MTOM 消息。

325.10. 自定义标头和附加过滤

如果您需要提供标头或附件的自定义处理，请扩展现有的 BasicMessageFilter，并覆盖适当的方法，或编写一个 MessageFilter 接口的新实现。
要使用您的自定义过滤器，请将以下内容添加到 spring 上下文中：

您可以按如下方式指定全局或本地消息过滤器： a）为所有 Spring-WS 端点提供全局配置的全局自定义过滤器

```
<bean id="messageFilter" class="your.domain.myMessageFiler" scope="singleton" />
```

或 b）本地 messageFilter 直接在端点上，如下所示：

```
to("spring-ws:http://yourdomain.com?messageFilter=#myEndpointSpecificMessageFilter");
```

如需更多信息，请参阅 CAMEL-5724

如果要创建自己的 MessageFilter，请考虑在类 BasicMessageFilter 中覆盖默认的 MessageFilter 实现中的以下方法：

```
protected void doProcessSoapHeader(Message inOrOut, SoapMessage soapMessage)
{your code /*no need to call super*/ }

protected void doProcessSoapAttachements(Message inOrOut, SoapMessage response)
{ your code /*no need to call super*/ }
```

325.11. 使用自定义 MESSAGESENDER 和 MESSAGEFACTORY

可以在 registry 中引用自定义消息发送者或工厂，如下所示：

```
from("direct:example")
.to("spring-ws:http://foo.com/bar?
messageFactory=#messageFactory&messageSender=#messageSender")
```

Spring 配置 ：

```
<!-- authenticate using HTTP Basic Authentication -->
<bean id="messageSender"
class="org.springframework.ws.transport.http.HttpComponentsMessageSender">
    <property name="credentials">
        <bean class="org.apache.commons.httpclient.UsernamePasswordCredentials">
            <constructor-arg index="0" value="admin"/>
            <constructor-arg index="1" value="secret"/>
        </bean>
    </property>
</bean>
```

```
<!-- force use of Sun SAAJ implementation, http://static.springsource.org/spring-ws/sites/1.5/faq.html#saaj-jboss -->
<bean id="messageFactory" class="org.springframework.ws.soap.saaj.SaajSoapMessageFactory">
    <property name="messageFactory">
        <bean
class="com.sun.xml.messaging.saaj.soap.ver1_1.SOAPMessageFactory1_1Impl"></bean>
    </property>
</bean>
```

325.12. 公开 WEB 服务

要使用此组件公开 Web 服务，您首先需要设置 MessageDispatcher 在 Spring XML 文件中查找端点映射。如果您计划在 servlet 容器中运行，您可能需要使用 web.xml 中配置的 MessageDispatcherServlet。

默认情况下，MessageDispatcherServlet 会查找名为 /WEB-INF/spring-ws-servlet.xml 的 Spring XML。要将 Camel 与 Spring-WS 搭配使用，该 XML 文件中唯一强制 bean 是 CamelEndpointMapping。此 bean 允许 MessageDispatcher 将 Web 服务请求分配给您的路由。

web.xml

```
<web-app>
    <servlet>
        <servlet-name>spring-ws</servlet-name>
        <servlet-class>org.springframework.ws.transport.http.MessageDispatcherServlet</servlet-
class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>spring-ws</servlet-name>
        <url-pattern>/*</url-pattern>
    </servlet-mapping>
</web-app>
```

spring-ws-servlet.xml

```
<bean id="endpointMapping"
class="org.apache.camel.component.spring.ws.bean.CamelEndpointMapping" />

<bean id="wsdl" class="org.springframework.ws.wsdl.wsdl11.DefaultWsdl11Definition">
    <property name="schema">
        <bean class="org.springframework.xml.xsd.SimpleXsdSchema">
            <property name="xsd" value="/WEB-INF/foobar.xsd"/>
        </bean>
    </property>
    <property name="portTypeName" value="FooBar"/>
```

```
    <property name="locationUri" value="/"/>
    <property name="targetNamespace" value="http://example.com/"/>
</bean>
```

有关设置 Spring-WS 的更多信息，请参阅 编写 Contract-First Web Services。基本上，3.6 "Implementing Endpoint"由此组件处理（特别是段落为 3.6.2"对 Endpoint 的消息有关）是处于 CamelEndpointMapping 所在的位置。另外，不要忘记检查 Camel 分发中包含的 Spring Web Services 示例。

325.13. 路由中的端点映射

通过 XML 配置，您现在可以使用 Camel 的 DSL 定义端点处理的 Web 服务请求：

以下路由将收到 http://example.com/ 命名空间中带有名为"GetFoo"的根元素的所有 Web 服务请求。

```
from("spring-ws:rootqname:{http://example.com/}GetFoo?
endpointMapping=#endpointMapping")
.convertBodyTo(String.class).to(mock:example)
```

以下路由将收到包含 http://example.com/GetFoo SOAP 操作的 Web 服务请求。

```
from("spring-ws:soapaction:http://example.com/GetFoo?
endpointMapping=#endpointMapping")
.convertBodyTo(String.class).to(mock:example)
```

以下路由将接收发送到 http://example.com/foobar 的所有请求。

```
from("spring-ws:uri:http://example.com/foobar?endpointMapping=#endpointMapping")
.convertBodyTo(String.class).to(mock:example)
```

以下路由将接收在消息（和 default 命名空间）内包含元素 <foobar>abc </foobar> 的请求。

```
from("spring-ws:xpathresult:abc?
expression=//foobar&endpointMapping=#endpointMapping")
.convertBodyTo(String.class).to(mock:example)
```

325.14. 使用现有端点映射的替代配置

对于带有 mapping-type beanname 的每个端点，Registry/ApplicationContext 中需要一个类型为

CamelEndpointDispatcher 的 bean。此 bean 充当 Camel 端点和现有 端点映射 （如 PayloadRootQNameEndpointMapping ）之间的桥接。

注意：使用 beanname mapping-type 主要用于（传统）您已使用 Spring-WS 的情况，并在 Spring XML 文件中定义端点映射。beanname mapping-type 允许您使 Camel 路由进入现有的端点映射。当您从头开始，建议将端点映射定义为 Camel URI （如上所示，带有 endpointMapping），因为它需要较少的配置，且更具表达性。或者，您可以将 vanilla Spring-WS 与注解帮助一起使用。

使用 beanname 的路由示例：

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="spring-ws:beanname:QuoteEndpointDispatcher" />
    <to uri="mock:example" />
  </route>
</camelContext>

<bean id="legacyEndpointMapping"
class="org.springframework.ws.server.endpoint.mapping.PayloadRootQNameEndpointMapping">
  <property name="mappings">
    <props>
      <prop key="{http://example.com/}GetFuture">FutureEndpointDispatcher</prop>
      <prop key="{http://example.com/}GetQuote">QuoteEndpointDispatcher</prop>
    </props>
  </property>
</bean>

<bean id="QuoteEndpointDispatcher"
class="org.apache.camel.component.spring.ws.bean.CamelEndpointDispatcher" />
<bean id="FutureEndpointDispatcher"
class="org.apache.camel.component.spring.ws.bean.CamelEndpointDispatcher" />
```

325.15. POJO (UN) MARSHALLING

Camel 的可插拔数据格式支持使用 JAXB、XStream、JibX、Harstor 和 XMLBeans 等库支持 pojo/xml marshalling。您可以在路由中使用这些数据格式来发送和接收到 Web 服务的 pojo、并接收。

在 访问 Web 服务时，您可以对请求进行 marshal 和 unmarshal the response 信息：

```
JaxbDataFormat jaxb = new JaxbDataFormat(false);
jaxb.setContextPath("com.example.model");

from("direct:example").marshal(jaxb).to("spring-ws:http://foo.com/bar").unmarshal(jaxb);
```

同样，在提供 Web 服务时，您可以对 POJO 的请求，将响应消息返回回 XML：

```
from("spring-ws:rootqname:{http://example.com/}GetFoo?
endpointMapping=#endpointMapping").unmarshal(jaxb)
.to("mock:example").marshal(jaxb);
```

325.16. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 326 章 SQL 组件

从 Camel 版本 1.4 开始提供

sql: 组件允许您使用 JDBC 查询来处理数据库。此组件和 JDBC 组件之间的区别在于，如果 SQL 查询是端点的属性，它将消息有效负载用作传递给查询的参数。

此组件在后台使用 spring-jdbc 进行实际的 SQL 处理。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-sql</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

SQL 组件还支持：

- 基于 JDBC 的存储库用于 Idempotent Consumer EIP 模式。请参阅 第 326.13 节 “使用基于 JDBC 的幂等存储库”。
- 基于 JDBC 的存储库用于聚合器 EIP 模式。请参阅 第 326.14 节 “使用基于 JDBC 的聚合存储库”。

326.1. URI 格式

WARNING:From Camel 2.11 开始此组件可以创建消费者（如 from （））和生成者端点（如 to （））。在以前的版本中，它只能充当生成者。

INFO：此组件可用作 事务客户端。

SQL 组件使用以下端点 URI 表示法：

```
sql:select * from table where id=# order by name[?options]
```

从 Camel 2.11 开始，您可以使用命名的参数，使用 :' instname_of_the_parameter' 风格，如下所示：

```
sql:select * from table where id=:#myId order by name[?options]
```

使用命名参数时，Camel 将在给定优先级：
1 中查找名称。如果它有一个 java.util.Map
2，则来自消息正文。来自消息标头

如果无法解析指定参数，则抛出异常。

从 Camel 2.14 开始，您可以使用 Simple 表达式作为参数，如下所示：

```
sql:select * from table where id=:#${property.myId} order by name[?options]
```

请注意，表示 SQL 查询参数的标准 ？符号被 @ 符号替换，因为 ？符号用于指定端点的选项。？符号替换可根据端点进行配置。

从 Camel 2.17 开始，您可以将 SQL 查询外部化到 classpath 或文件系统中的文件，如下所示：

```
sql:classpath:sql/myquery.sql[?options]
```

myquery.sql 文件位于 classpath 中，且只是纯文本

```
-- this is a comment
select *
from table
where
  id = :#${property.myId}
order by
  name
```

在文件中，您可以根据需要使用多行并格式化 SQL。和 也会使用诸如 - dash 行等注释。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

326.2. 选项

SQL 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
DataSource (common)	设置 DataSource 用于与数据库通信。		DataSource
usePlaceholder (advanced)	设置是否使用占位符，并将所有占位符字符替换为 SQL 查询中的 符号。这个选项是默认的 true	true	布尔值
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

SQL 端点使用 URI 语法进行配置：

```
sql:query
```

使用以下路径和查询参数：

326.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
query	**必需** 设置要执行的 SQL 查询。您可以使用 file: 或 classpath: 作为前缀来指定查询的位置进行外部化。		字符串

326.2.2. 查询参数(45 参数)：

Name	描述	默认值	类型
allowNamedParameters (common)	是否允许在查询中使用命名参数。	true	布尔值
DataSource (common)	设置 DataSource 用于与数据库通信。		DataSource

Name	描述	默认值	类型
dataSourceRef (common)	**弃用** 将引用设置为从 registry 中查询的 DataSource，以用于与数据库通信。		字符串
outputClass (common)	指定在 outputType=SelectOne 或 SelectList 时使用的完整软件包和类名称。		字符串
outputHeader (common)	将查询结果存储在标头中而不是消息正文。默认情况下，outputHeader == null，查询结果存储在消息正文中，消息正文中的任何现有内容都会被丢弃。如果设置了 outputHeader，则值将用作标头名称来存储查询结果，并保留原始消息正文。		字符串
outputType (common)	将使用者或制作者的输出设置为 SelectList 作为 Map 列表，或者选择One 作为单个 Java 对象：a）如果查询只有一个列，则返回 JDBC Column 对象。（如 SELECT COUNT （）FROM PROJECT 将会返回 Long 对象。b）如果查询具有多个列，则它将返回该结果的映射。如果 outputClass 被设置，则它会返回 Long object. b。然后，它将调用与列名称匹配的所有集合，将查询结果转换为 Java bean 对象。它将假定您的类具有创建实例的默认构造器。d）如果查询导致多个行，它会抛出一个非唯一结果异常。	SelectList	SqlOutputType
分隔符 (common)	当从消息正文（如果正文是一个 String 类型）获取参数值时使用分隔符，要在 # 占位符中插入。请注意，如果您使用命名的参数，则改为使用 Map 类型。默认值为 comma。	,	char
breakBatchOnConsumeFail (consumer)	如果 onConsume 失败，则设置为是否中断批处理。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
expectedUpdateCount (consumer)	设置预期的更新计数，以便在使用 onConsume 时进行验证。	-1	int
maxMessagesPerPoll (consumer)	设置要轮询的最大消息数		int
onConsume (consumer)	处理每行后，可以执行此查询，如果 Exchange 成功处理，例如将行标记为已处理。查询可以有参数。		字符串

Name	描述	默认值	类型
onConsumeBatchComplete (consumer)	处理整个批处理后，可以执行此查询以批量更新行等。查询不能有参数。		字符串
onConsumeFailed (consumer)	处理每行后，可以执行此查询，如果 Exchange 失败，例如将行标记为失败。查询可以有参数。		字符串
routeEmptyResultSet (consumer)	设置是否允许空 resultset 发送到下一跃点。默认为 false。因此，空结果集将被过滤掉。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
transacted (consumer)	启用或禁用事务。如果启用，如果处理交换失败，则消费者无法处理任何进一步的交换，从而导致回滚操作	false	布尔值
useIterator (consumer)	设置如何将 resultset 传送到 route。将交付表示为列表或单个对象。默认为 true。	true	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
processingStrategy (consumer)	当消费者处理 rows/batch 时，允许插件使用自定义 org.apache.camel.component.sql.SqlProcessingStrategy 执行查询。		SqlProcessingStrategy
batch (producer)	启用或禁用批处理模式	false	布尔值
noop (producer)	如果设置，将忽略 SQL 查询的结果，并使用现有 IN 消息作为继续处理的 OUT 消息	false	布尔值
useMessageBodyForSql (producer)	是否将消息正文用作 SQL，然后是参数的标头。如果启用了这个选项，则不会使用 uri 中的 SQL。	false	布尔值

Name	描述	默认值	类型
alwaysPopulateStatement (advanced)	如果启用了，则始终调用 org.apache.camel.component.sql.SqlPrepareStatementStrategy 的 populateStatement 方法，也如果没有准备预期的参数。当这是 false 时，只有在设置了 1 个或多个预期参数时，才会调用 populateStatement；例如，这样可避免读取没有参数的 SQL 查询的消息正文/标题。	false	布尔值
parametersCount (advanced)	如果设置大于零，则 Camel 将使用此计数的参数值替换，而不是通过 JDBC 元数据 API 查询。如果 JDBC 供应商无法返回正确的参数数，则这非常有用，用户可能会改为覆盖。		int
占位符 （高级）	指定在 SQL 查询中替换到的字符。请注意，它只是一个简单的 String.replaceAll （）操作，且不会涉及 SQL 解析（加引号的字符串也会改变）。	#	字符串
prepareStatementStrategy (advanced)	允许插件使用自定义 org.apache.camel.component.sql.SqlPrepareStatementStrategy 来控制查询和准备语句的准备。		SqlPrepareStatement 策略
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
templateOptions (advanced)	使用映射中的键/值配置 Spring JdbcTemplate		Map
usePlaceholder (advanced)	设置是否使用占位符，并将所有占位符字符替换为 SQL 查询中的 符号。这个选项是默认的 true	true	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值

Name	描述	默认值	类型
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

326.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.sql.data-source	设置 DataSource 用于与数据库通信。选项是 javax.sql.DataSource 类型。		字符串
camel.component.sql.enabled	启用 sql 组件	true	布尔值
camel.component.sql.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Name	描述	默认值	类型
camel.component.sql.use-placeholder	设置是否使用占位符，并将所有占位符字符替换为 SQL 查询中的 符号。这个选项是默认的 true	true	布尔值

326.4. 消息正文处理

SQL 组件尝试将消息正文转换为 java.util.Iterator 类型的对象，然后使用此迭代器填充查询参数（每个查询参数都由端点 URI 中的 # 符号（或配置的占位符）表示。如果消息正文不是数组或集合，则转换会导致迭代一个对象，这是正文本身。

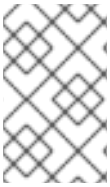
例如，如果消息正文是 java.util.List 的实例，则列表中的第一项将被替换为 SQL 查询中的第一个出现的 #，则列表中的第二个项目将被替换为 # 的第二个位置，以此类推。

如果 batch 设为 true，则入站消息正文更改的解释稍有 - 而非参数器的迭代器，则组件需要包含参数迭代器的迭代器；外部器的大小决定了批处理大小。

从 Camel 2.16 开始，您可以使用选项 useMessageBodyForSql，允许使用消息正文作为 SQL 语句，然后 SQL 参数必须在带有键 SqlConstants.SQL_PARAMETERS 的标头中提供。这允许 SQL 组件更有效地运行，因为 SQL 查询来自消息正文。

326.5. 查询的结果

对于 选择 操作，结果是 List<Map<String, Object>& gt; 类型的实例，如 JdbcTemplate.queryForList （） 方法返回。对于 更新操作，NULL 正文会返回，因为 更新操作 仅被设置为标头，永远不会设置为正文。



注意

有关更新操作的更多信息，请参阅 标题值。

默认情况下，结果放置在消息正文中。如果设置了 outputHeader 参数，则结果将放在标头中。这是使用完整消息增强模式来添加标头的替代选择，它为查询序列或其它小值提供了一个简洁的语法。最好一起使用 outputHeader 和 outputType ：

```
from("jms:order.inbox")
    .to("sql:select order_seq.nextval from dual?
```

```
outputHeader=OrderId&outputType=SelectOne")
    .to("jms:order.booking");
```

326.6. 使用 STREAMLIST

在生成者上的*Camel 2.18* 中支持 outputType=StreamList，它使用 iterator 流传输查询的输出。这允许以流传输方式处理数据，例如，Splitter EIP 可用于一次处理每行，并根据需要从数据库加载数据。

```
from("direct:withSplitModel")
        .to("sql:select * from projects order by id?
outputType=StreamList&outputClass=org.apache.camel.component.sql.ProjectModel")
        .to("log:stream")
        .split(body()).streaming()
            .to("log:row")
            .to("mock:result")
        .end();
```

326.7. 标头值

在 执行更新 操作时，SQL 组件将更新计数存储在以下消息标头中：

标头	描述
CamelSqlUpdateCount	更新更新操作的行数，返回为 **Integer** 对象。**使用 outputType=StreamList 时不提供此标头。**
CamelSqlRowCount	选择 操作返回的行数，返回为 **Integer** 对象。使用 outputType=StreamList 时不提供此标头。
CamelSqlQuery	Camel 2.8： 要执行的 Query。此查询优先于端点 URI 中指定的查询。请注意，标头中 *的查询参数* 由 **?** 而不是一个 **#** 符号表示

在执行 插入 操作时，SQL 组件将生成的密钥和这些行数的行存储在以下消息标头中(从 Camel 2.12.4, 2.13.1 开始提供)：

标头	描述
CamelSqlGeneratedKeysRowCount	标头中包含生成的密钥的行数。
CamelSqlGeneratedKeyRows	包含生成的密钥的行（密钥映射列表）。

326.8. 生成的密钥

** 从 Camel 2.12.4, 2.13.1 和 2.14 * 开始提供*

如果您使用 SQL INSERT 插入数据，则 RDBMS 可能会支持自动生成的密钥。您可以指示 SQL producer 在标头中返回生成的密钥。
为此，请设置标头 CamelSqlRetrieveGeneratedKeys=true。然后，生成的密钥将作为标头提供，其中包含上表中列出的键。

您可以在此 单元测试 中看到更多详细信息。

326.9. DATASOURCE

现在，您可以直接设置对 URI 中的 DataSource 的引用：

```
sql:select * from table where id=# order by name?dataSource=myDS
```

326.10. 示例

在以下示例中，我们执行查询并作为行 列表 检索结果，其中每行是一个 Map<String, Object& gt;, 键是列名称。

首先，我们设置一个表，用于我们的示例。由于这基于单元测试，因此我们在 java 中进行：

我们执行的 SQL 脚本 createAndPopulateDatabase.sql 如下所示：

然后，我们配置路由和我们的 sql 组件。请注意，我们在 sql 端点前面使用 直接 端点。这样，我们可以使用 URI direct:simple 向直接端点发送交换，这比长 sql: URI 更容易使用。请注意，DataSource 在 registry 中查找，因此我们可以使用标准的 Spring XML 来配置我们的 数据源。

然后，我们将触发消息进入 直接 端点，该端点会将它路由到查询数据库的 sql 组件。

我们可以在 Spring XML 中配置数据源，如下所示：

```
<jee:jndi-lookup id="myDS" jndi-name="jdbc/myDataSource"/>
```

326.10.1. 使用命名参数

从 Camel 2.11 开始提供

在下面的给定路由中，我们想要从 projects 表中获取所有项目。请注意，SQL 查询具有 2 个命名参数，：#lic 和 :#min。
然后，Camel 将从消息正文或消息标头中查找这些参数。请注意，在上面的示例中，我们为命名参数设置两个带有恒定值
的标头：

```
from("direct:projects")
  .setHeader("lic", constant("ASF"))
  .setHeader("min", constant(123))
  .to("sql:select * from projects where license = :#lic and id > :#min order by id")
```

尽管消息正文是 java.util.Map，但命名的参数将从正文中获取。

```
from("direct:projects")
  .to("sql:select * from projects where license = :#lic and id > :#min order by id")
```

326.11. 在生成者中使用表达式参数

从 Camel 2.14 开始提供

在以下给定路由中，我们想要从数据库获取所有项目。它使用交换的正文来定义许可证，并将 属性的

值用作第二个参数。

```
from("direct:projects")
  .setBody(constant("ASF"))
  .setProperty("min", constant(123))
  .to("sql:select * from projects where license = :#${body} and id > :#${property.min} order by
id")
```

326.11.1. 在消费者中使用表达式参数

从 Camel 2.23 开始提供

当使用 SQL 组件作为消费者时，您现在可以使用表达式参数（简单语言）构建动态查询参数，如在 bean 上调用方法来检索 id、date 或 something。

例如，在以下示例中，我们调用 bean myIdGenerator 中的 nextId 方法：

```
from("sql:select * from projects where id = :#${bean:myIdGenerator.nextId}")
    .to("mock:result");
```

和 bean 具有以下方法：

```
public static class MyIdGenerator {

    private int id = 1;

    public int nextId() {
        return id++;
    }
```

请注意，没有带有消息正文和标头的现有 Exchange，因此您可以在消费者中使用的简单表达式最可用于调用 bean 方法，如本例中所示。

326.12. 使用带有动态值的 IN 查询

从 Camel 2.17 开始提供

从 SQL 生成者上的 Camel 2.17 中，可以将 SQL 查询与 IN 值动态计算的 IN 语句一起使用。例如，来自消息正文或标头等。

要使用 IN，您需要：

- 使用以下内容为参数名称 加上前缀：
- 在参数外添加 ( )

一个示例解释了这一点。使用以下查询：

```
-- this is a comment
select *
from projects
where project in (:#in:names)
order by id
```

在以下路由中：

```
from("direct:query")
   .to("sql:classpath:sql/selectProjectsIn.sql")
   .to("log:query")
   .to("mock:query");
```

然后，IN 查询可以使用键名称的标头以及动态值，例如：

```
// use an array
template.requestBodyAndHeader("direct:query", "Hi there!", "names", new String[]{"Camel",
"AMQ"});

// use a list
List<String> names = new ArrayList<String>();
names.add("Camel");
names.add("AMQ");

template.requestBodyAndHeader("direct:query", "Hi there!", "names", names);

// use a string separated values with comma
template.requestBodyAndHeader("direct:query", "Hi there!", "names", "Camel,AMQ");
```

查询也可以在端点中指定，而不是外部化（代表外部大小使维护 SQL 查询变得更加容易）

```
from("direct:query")
   .to("sql:select * from projects where project in (:#in:names) order by id")
```

```
    .to("log:query")
    .to("mock:query");
```

326.13. 使用基于 JDBC 的幂等存储库

从 Camel 2.7 开始提供：在本节中，我们将使用基于 JDBC 的幂等存储库。

TIP:*Abstract class* From Camel 2.9 之后，有一个抽象类 org.apache.camel.processor.idempotent.jdbc.AbstractJdbcMessageIdRepository，您可以扩展到构建自定义 JDBC 幂等存储库。

首先，我们必须创建将由幂等存储库使用的数据库表。对于 Camel 2.7，我们使用以下模式：

```
CREATE TABLE CAMEL_MESSAGEPROCESSED ( processorName VARCHAR(255),
messageId VARCHAR(100) )
```

在 Camel 2.8 中，我们添加了 createdAt 列：

```
CREATE TABLE CAMEL_MESSAGEPROCESSED ( processorName VARCHAR(255),
messageId VARCHAR(100), createdAt TIMESTAMP )
```

警告：SQL Server TIMESTAMP 类型是一个固定长度的二进制字符串类型。它没有映射到任何 JDBC 时间类型： DATE、 TIME 或 TIMESTAMP。

自定义 JdbcMessageIdRepository

从 Camel 2.9.1 开始，您有几个选项可根据您的需要调整 org.apache.camel.processor.idempotent.jdbc.JdbcMessageIdRepository：

参数	默认值	描述
createTableIfNotExists	true	定义 Camel 是否应该尝试创建表（如果不存在）。

参数	默认值	描述
tableExistsString	SELECT 1 FROM CAMEL_MESSAGEPROCESSED WHERE 1 = 0	此查询用于找出表是否已存在。必须抛出异常以指示表不存在。
createString	CREATE TABLE CAMEL_MESSAGEPROCESSED (processorName VARCHAR(255), messageId VARCHAR(100), createdAt TIMESTAMP)	用于创建表的 语句。

参数	默认值	描述
queryString	SELECT COUNT(*) FROM CAMEL_MESSAGEPROCESSED WHERE processorName = ? AND messageId = ?	用于找出消息是否已存在于存储库中的查询（结果不等于 '0'）。它采用两个参数。第一个是处理器名称(**字符串**)，第二个则是消息 ID (**字符串**)。
insertString	INSERT INTO CAMEL_MESSAGEPROCESSED (processorName, messageId, createdAt) VALUES (?, ?, ?)	用于将该条目添加到表中的语句。它取三个参数。第一个是处理器名称(**String**)，第二个是消息 ID (**String**)，第三个则是当此条目添加到存储库时的时间戳 (**java.sql.Timestamp**)。
deleteString	DELETE FROM CAMEL_MESSAGEPROCESSED WHERE processorName = ? AND messageId = ?	用于从数据库中删除条目的语句。它取两个参数。第一个是处理器名称(**字符串**)，第二个则是消息 ID (**字符串**)。

326.14. 使用基于 JDBC 的聚合存储库

从 Camel 2.6 开始提供

INFO: 在 Camel 2.6 中使用 JdbcAggregationRepository

在 Camel 2.6 中，JdbcAggregationRepository 在 camel-jdbc-aggregator 组件中提供。从 Camel 2.7 开始，J dbcAggregationRepository 在 camel-sql 组件中提供。

JdbcAggregationRepository 是一个 AggregationRepository，它实时保留聚合的消息。这样可确保您不会松散消息，因为默认聚合器将使用仅在内存中的 AggregationRepository。JdbcAggregationRepository 允许 Camel 与 Camel 共同提供对聚合器的持久支持。

只有在成功处理交换时，才会将其标记为 complete，只有在 AggregationRepository 上调用 confirm 方法时才会发生这种情况。这意味着，如果同一交换再次失败，它将被重试，直到成功为止。

您可以使用选项 maximumRedeliveries 来限制给定恢复的交换的最大重新发送尝试次数。您还必须设置 deadLetterUri 选项，以便 Camel 知道在 max Redeliveries 正在命中时要发送交换的位置。

您可以在 camel-sql 的单元测试中看到一些示例，例如 这个测试。

326.14.1. 数据库

为正常工作，每个聚合器使用两个表：聚合和完成一个。按照惯例，完成的名称与后缀为 "_COMPLETED" 的聚合名称相同。名称必须在带有 RepositoryName 属性的 Spring bean 中配置。在以下示例中，将使用聚合。

两个表的表结构定义相同：如果 String 值用作键(id)，而 Blob 则包含字节数组中的交换序列化。但是，应该记住一个区别： id 字段没有相同的内容，具体取决于表。
在聚合表 id 中，保存组件用于聚合消息的关联 Id。在完成的表中，id 包含存储在对应 blob 字段中的交换 ID。

以下是用于创建表的 SQL 查询，只需将 "聚合" 替换为您的聚合器存储库名称。

```
CREATE TABLE aggregation (
```

```
 id varchar(255) NOT NULL,
 exchange blob NOT NULL,
 constraint aggregation_pk PRIMARY KEY (id)
);
CREATE TABLE aggregation_completed (
 id varchar(255) NOT NULL,
 exchange blob NOT NULL,
 constraint aggregation_completed_pk PRIMARY KEY (id)
);
```

326.15. 将正文和标头存储为文本

从 Camel 2.11 开始提供

您可以配置 JdbcAggregationRepository 以存储消息正文，并选择(ed)标头作为 String 在单独的列中选择。例如，要存储正文，并且以下两个标头 companyName 和 accountName 使用以下 SQL：

```
CREATE TABLE aggregationRepo3 (
 id varchar(255) NOT NULL,
 exchange blob NOT NULL,
 body varchar(1000),
 companyName varchar(1000),
 accountName varchar(1000),
 constraint aggregationRepo3_pk PRIMARY KEY (id)
);
CREATE TABLE aggregationRepo3_completed (
 id varchar(255) NOT NULL,
 exchange blob NOT NULL,
 body varchar(1000),
 companyName varchar(1000),
 accountName varchar(1000),
 constraint aggregationRepo3_completed_pk PRIMARY KEY (id)
);
```

然后，将存储库配置为启用此行为，如下所示：

```
<bean id="repo3"
  class="org.apache.camel.processor.aggregate.jdbc.JdbcAggregationRepository">
  <property name="repositoryName" value="aggregationRepo3"/>
  <property name="transactionManager" ref="txManager3"/>
  <property name="dataSource" ref="dataSource3"/>
  <!-- configure to store the message body and following headers as text in the repo -->
  <property name="storeBodyAsText" value="true"/>
  <property name="headersToStoreAsText">
    <list>
      <value>companyName</value>
      <value>accountName</value>
```

```
    </list>
  </property>
</bean>
```

326.15.1. codec (Serialization)

由于它们可以包含任何类型的有效载荷，因此交换不能按照设计序列化。它将转换为一个字节数组，以存储在数据库 BLOB 字段中。所有这些转换都由 JdbcCodec 类处理。代码的详情需要注意：ClassLoadingAwareObjectInputStream。

ClassLoadingAwareObjectInputStream 已从 Apache ActiveMQ 项目中重复使用。它嵌套 ObjectInputStream，并将其与 ContextClassLoader 一起使用，而不是 当前的Thread。其好处在于能够加载由其他捆绑包公开的类。这允许交换正文和标头具有自定义类型对象引用。

326.15.2. transaction

需要 Spring PlatformTransactionManager 来编配事务。

326.15.2.1. 服务(Start/Stop)

start 方法验证数据库和存在所需的表。如果出现错误，它将在启动期间失败。

326.15.3. 聚合器配置

根据目标环境，聚合器可能需要一些配置。如您已经了解，每个聚合器应具有自己的存储库（在数据库中创建的对应表对）和数据源。如果默认的 lobHandler 没有适应您的数据库系统，它可以与 lobHandler 属性注入。

以下是 Oracle 的声明：

```
<bean id="lobHandler" class="org.springframework.jdbc.support.lob.OracleLobHandler">
  <property name="nativeJdbcExtractor" ref="nativeJdbcExtractor"/>
</bean>
<bean id="nativeJdbcExtractor"
  class="org.springframework.jdbc.support.nativejdbc.CommonsDbcpNativeJdbcExtractor"/>
<bean id="repo"
  class="org.apache.camel.processor.aggregate.jdbc.JdbcAggregationRepository">
  <property name="transactionManager" ref="transactionManager"/>
  <property name="repositoryName" value="aggregation"/>
  <property name="dataSource" ref="dataSource"/>
```

```
  <!-- Only with Oracle, else use default -->
  <property name="lobHandler" ref="lobHandler"/>
</bean>
```

326.15.4. 最佳锁定

从 Camel 2.12 开始，您可以打开 optimisticLocking，并在集群环境中使用此基于 JDBC 的聚合存储库，其中多个 Camel 应用程序为聚合存储库共享同一数据库。如果存在竞争条件，则 JDBC 驱动程序将抛出一个供应商特定异常，Jdb cAggregationRepository 可以响应。要了解导致 JDBC 驱动程序的例外情况被视为一个 optimistick locking 错误，我们需要一个映射程序来执行此操作。因此，有 org.apache.camel.processor.aggregate.jdbc.JdbcOptimisticLockingExceptionMapper 允许您实施您的自定义逻辑（如果需要）。有默认的实现 org.apache.camel.processor.aggregate.jdbc.DefaultJdbcOptimisticLockingExceptionMapper，它的工作方式如下：

完成以下检查：

- 如果导致异常是 SQLException，则如果以 23 开始，则会检查 SQLState。
- 如果原因的例外是 DataIntegrityViolationException
- 如果导致异常类名称的名称的名称中包含 "ConstraintViolation"。
- 如果已经配置了任何类名称，则对 FQN 类名称进行可选检查

您可以添加 FQN 类名，如果任何原因异常（或任何嵌套）等于任何 FQN 类名称，然后是最佳锁定错误。

下面是一个示例，我们从 JDBC 供应商定义 2 个额外的 FQN 类名称：

```
<bean id="repo"
class="org.apache.camel.processor.aggregate.jdbc.JdbcAggregationRepository">
  <property name="transactionManager" ref="transactionManager"/>
  <property name="repositoryName" value="aggregation"/>
  <property name="dataSource" ref="dataSource"/>
  <property name="jdbcOptimisticLockingExceptionMapper" ref="myExceptionMapper"/>
</bean>
<!-- use the default mapper with extraFQN class names from our JDBC driver -->
<bean id="myExceptionMapper"
class="org.apache.camel.processor.aggregate.jdbc.DefaultJdbcOptimisticLockingExceptionMapper">
```

```
  <property name="classNames">
    <util:set>
      <value>com.foo.sql.MyViolationExceptoion</value>
      <value>com.foo.sql.MyOtherViolationExceptoion</value>
    </util:set>
  </property>
</bean>
```

326.16. 传播行为

JdbcAggregationRepository 使用 Spring-TX 的两个不同 事务模板。一个是只读的，一个用于读写操作。

但是，当在自身使用 < transacted /> 的路由中使用 JdbcAggregationRepository 并且使用了通用平台 TransactionManager 时，可能需要配置 JdbcAggregationRepository 中使用的 传播行为。

以下是实现这个目标的方法：

```
<bean id="repo"
class="org.apache.camel.processor.aggregate.jdbc.JdbcAggregationRepository">
  <property name="propagationBehaviorName" value="PROPAGATION_NESTED" />
</bean>
```

传播由 org.springframework.transaction.TransactionDefinition 接口的常量指定，因此 propagationBehaviorName 非常方便，允许使用常量名称。

326.17. POSTGRESQL 问题单

存在特殊的数据库，可能会导致 JdbcAggregationRepository 使用的最佳锁定问题。如果数据完整性违反异常(SQLState 23505)，PostgreSQL 会将连接标记为无效。这使得连接有效地在嵌套事务中不可用。可在 本文档中找到 详细信息。

org.apache.camel.processor.aggregate.jdbc.PostgresAggregationRepository 扩展 JdbcAggregationRepository，并使用特殊的 INSERT .ON CONFLICT .. 语句提供最佳锁定行为。

此声明是 （使用默认聚合表定义）：

```
INSERT INTO aggregation (id, exchange) values (?, ?) ON CONFLICT DO NOTHING
```

详情可在 PostgreSQL 文档 中找到。

使用此子句时，java.sql.PreparedStatement.executeUpdate （） 调用返回 0， 而不是抛出带有 SQLState=23505 的 SQLException。进一步的处理与通用的 JdbcAggregationRepository 完全相同，但没有将 PostgreSQL 连接标记为无效。

326.18. CAMEL SQL STARTER

spring-boot 用户提供了一个初学者模块。使用初学者时， 可以使用 spring-boot 属性直接配置 DataSource。

```
# Example for a mysql datasource
spring.datasource.url=jdbc:mysql://localhost/test
spring.datasource.username=dbuser
spring.datasource.password=dbpass
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
```

要使用这个功能， 请在 spring boot pom.xml 文件中添加以下依赖项：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-sql-starter</artifactId>
  <version>${camel.version}</version> <!-- use the same version as your Camel core version --
>
</dependency>

<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-jdbc</artifactId>
  <version>${spring-boot-version}</version>
</dependency>
```

如果需要，您还应包含特定的数据库驱动程序。

第 327 章 SQL STORED PROCEDURE 组件

从 Camel 版本 2.17 开始提供

sql-stored: 组件允许您使用 JDBC 存储的流程查询来处理数据库。此组件是 SQL 组件的扩展，但专门用于调用存储的流程。

此组件在后台使用 spring-jdbc 进行实际的 SQL 处理。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-sql</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

327.1. URI 格式

SQL 组件使用以下端点 URI 表示法：

```
sql-stored:template[?options]
```

其中 template 是存储的步骤模板，您可以在其中声明存储的步骤名称和 IN、INOUT 和 OUT 参数。

您还可以引用文件系统上的外部文件中的模板或类路径，例如：

```
sql-stored:classpath:sql/myprocedure.sql[?options]
```

其中 sql/myprocedure.sql 是带有模板的类路径中的纯文本文件，如下所示：

```
SUBNUMBERS(
  INTEGER ${headers.num1},
  INTEGER ${headers.num2},
```

```
  INOUT INTEGER ${headers.num3} out1,
  OUT INTEGER out2
)
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

327.2. 选项

SQL Stored procedure 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
DataSource (producer)	设置 DataSource 用于与数据库通信。		DataSource
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

SQL Stored 流程端点使用 URI 语法进行配置：

```
sql-stored:template
```

使用以下路径和查询参数：

327.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
模板	**必需** 设置要执行的 StoredProcedure 模板		字符串

327.2.2. 查询参数(7 参数)：

Name	描述	默认值	类型
batch (producer)	启用或禁用批处理模式	false	布尔值
DataSource (producer)	设置 DataSource 用于与数据库通信。		DataSource

Name	描述	默认值	类型
function (producer)	此调用是否为函数。	false	布尔值
noop (producer)	如果设置，将忽略模板的结果，并使用现有 IN 消息作为继续处理的 OUT 消息	false	布尔值
outputHeader (producer)	将模板结果存储在标头中，而不是消息正文。默认情况下，outputHeader == null，模板结果存储在消息正文中，消息正文中的任何现有内容都会被丢弃。如果设置了 outputHeader，则该值将用作标头的名称，以存储模板结果，并保留原始消息正文。		字符串
useMessageBodyForTemplate (producer)	是否将消息正文用作模板，然后是参数的标头。如果启用了这个选项，则不会使用 uri 中的模板。	false	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

327.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.sql-stored.data-source	设置 DataSource 用于与数据库通信。选项是 javax.sql.DataSource 类型。		字符串
camel.component.sql-stored.enabled	启用 sql-stored 组件	true	布尔值
camel.component.sql-stored.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

327.4. 声明存储的步骤模板

模板通过与 Java 方法签名类似的语法进行声明。存储的步骤的名称，然后以括号括起的参数。解释了这个示例：

```
<to uri="sql-stored:STOREDSAMPLE(INTEGER ${headers.num1},INTEGER
${headers.num2},INOUT INTEGER ${headers.num3} result1,OUT INTEGER result2)"/>
```

参数由类型声明，然后使用简单表达式映射到 Camel 消息。因此，在本示例中，前两个参数是 INTEGER 类型的 IN 值，映射到消息标头。第三个参数是 INOUT，表示它接受 INTEGER，然后返回不同的 INTEGER 结果。last 参数是 OUT 值，也是 INTEGER 类型。

在 SQL 术语中，存储的步骤可以声明为：

```
CREATE PROCEDURE STOREDSAMPLE(VALUE1 INTEGER, VALUE2 INTEGER, INOUT
RESULT1 INTEGER, OUT RESULT2 INTEGER)
```

327.4.1. IN 参数

IN 参数使用由空格分开的四个部分：参数名称、SQL 类型（具有 scale）、类型名称和值源。

参数名称是可选的，如果未提供，则会自动生成。它必须在 quotes (')之间指定。

SQL 类型是必需的，可以是整数（正数或负数）或引用某些类中的整数字段。如果 SQL 类型包含一个点，则组件会尝试解析该类并读取给定字段。例如，SQL 类型 com.Foo.INTEGER 从类 com.Foo 的 INTEGER 字段读取。如果类型不包含逗号，则用于解析整数值的类将是 java.sql.Types。类型可以通过缩放来 postfixed，如 DECIMAL (10)意味着 java.sql.Types.DECIMAL 带有 scale 10。

类型名称是可选的，必须在 quotes (')之间指定。

值源是必需的。值源从 Exchange 填充参数值。它可以是简单表达式或标头位置，例如 ：#<header name& gt;。例如 Simple expression ${header.val} 表示，参数值将从标题"val"中读取。标头位置表达式 :#val 将有相同的效果。

```
<to uri="sql-stored:MYFUNC('param1' org.example.Types.INTEGER(10) ${header.srcValue})"/>
```

URI 表示，使用参数名称"param1"调用存储的流程，它的 SQL 类型是从类 org.example.Types 的 INTEGER 的 INTEGER 读取的，扩展将设置为 10。参数的输入值从标头 "srcValue" 传递。

```
<to uri="sql-stored:MYFUNC('param1' 100 'mytypename' ${header.srcValue})"/>
```

URI 与之前在 上相同，但 SQL 类型为 100，类型为 "mytypename"。

实际调用将使用 org.springframework.jdbc.core.SqlParameter 来完成。

327.4.2. OUT 参数

OUT 参数的工作方式类似于 IN 参数，并包含三个部分：SQL 类型（具有 scale）、类型名称和输出参数名称。

SQL 类型的工作方式与 IN 参数相同。

类型名称是可选的，还可与 IN 参数相同。

输出参数名称用于 OUT 参数名称，以及存储结果的标头名称。

```
<to uri="sql-stored:MYFUNC(OUT org.example.Types.DECIMAL(10) outheader1)"/>
```

URI 表示 OUT 参数的名称为 "outheader1"，结果将是标题为 "outheader1"。

```
<to uri="sql-stored:MYFUNC(OUT org.example.Types.NUMERIC(10) 'mytype' outheader1)"/>
```

这与前一个相同，但类型名称将是"mytype"。

实际调用将使用 org.springframework.jdbc.core.SqlOutParameter 来完成。

327.4.3. INOUT 参数

INOUT 参数是以上所有参数的组合。它们从交换接收值，并存储结果作为消息标头。唯一的注意事项是跳过 IN 参数的"name"。相反，OUT 参数的"name"定义了 SQL 参数的名称，以及结果标头名称。

```
<to uri="sql-stored:MYFUNC(INOUT DECIMAL(10) ${headers.inheader} outheader)"/>
```

实际调用将使用 org.springframework.jdbc.core.SqlInOutParameter 来完成。

327.5. CAMEL SQL STARTER

spring-boot 用户提供了一个初学者模块。使用初学者时，可以使用 spring-boot 属性直接配置 DataSource。

```
# Example for a mysql datasource
spring.datasource.url=jdbc:mysql://localhost/test
spring.datasource.username=dbuser
spring.datasource.password=dbpass
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
```

要使用这个功能，请在 spring boot pom.xml 文件中添加以下依赖项：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-sql-starter</artifactId>
  <version>${camel.version}</version> <!-- use the same version as your Camel core version --
>
</dependency>

<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-jdbc</artifactId>
  <version>${spring-boot-version}</version>
</dependency>
```

如果需要，您还应包含特定的数据库驱动程序。

327.6. 另请参阅

- SQL 组件

第 328 章 SSH 组件

从 Camel 版本 2.10 开始提供

SSH 组件支持访问 SSH 服务器，以便您可以发送 SSH 命令并处理响应。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-ssh</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

328.1. URI 格式

```
ssh:[username[:password]@]host[:port][?options]
```

328.2. 选项

SSH 组件支持 15 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	使用共享的 SSH 配置		SshConfiguration
host (common)	设置远程 SSH 服务器的主机名。		字符串
port (common)	设置远程 SSH 服务器的端口号。		int
用户名 (security)	设置登录远程 SSH 服务器的用户名。		字符串
密码 (security)	设置用于连接远程 SSH 服务器的密码。需要 keyPairProvider 设置为 null。		字符串
pollCommand (common)	设置在每次轮询循环期间要发送到远程 SSH 服务器的命令字符串。只适用于用作消费者的 camel-ssh 组件，即 from (ssh://...)。您可能需要使用换行符来结束您的命令，并且必须采用 URL 编码 %0A		字符串

Name	描述	默认值	类型
keyPairProvider (security)	设置在使用证书连接到远程 SSH 服务器时使用的 KeyPairProvider 引用。		KeyPairProvider
keyType (security)	设置要传递给 KeyPairProvider 的密钥类型，作为身份验证的一部分。KeyPairProvider.loadKey (...)将传递这个值。默认为 ssh-rsa。		字符串
timeout (common)	设置在建立远程 SSH 服务器连接时等待的超时时间（毫秒）。默认值为 30000 毫秒。		long
certFilename (security)	**弃用** 设置用于身份验证的证书的资源路径。		字符串
certResource (security)	设置用于身份验证的证书的资源路径。将使用 ResourceHelperKeyPairProvider 解析基于文件的证书，并依赖于 keyType 设置。		字符串
channelType (advanced)	将频道类型设置为在命令执行过程中传递给频道。默认为 exec。		字符串
shellPrompt (advanced)	在命令执行后读取响应时，将 shell 提示符设置为丢弃		字符串
sleepForShellPrompt (advanced)	设置睡眠周期（以毫秒为单位），以等待从 shell 提示符读取响应。默认值为 100 毫秒。		long
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

SSH 端点使用 URI 语法进行配置：

```
ssh:host:port
```

使用以下路径和查询参数：

328.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
主机	**必需** 设置远程 SSH 服务器的主机名。		字符串

Name	描述	默认值	类型
port	设置远程 SSH 服务器的端口号。	22	int

328.2.2. 查询参数(31 参数)：

Name	描述	默认值	类型
failOnUnknownHost (common)	指定到未知主机的连接是否应该失败。只有在设置了属性 knownHosts 时，才会检查这个值。	false	布尔值
knownHostsResource (common)	设置 known_hosts 文件的资源路径		字符串
timeout (common)	设置在建立远程 SSH 服务器连接时等待的超时时间（毫秒）。默认值为 30000 毫秒。	30000	long
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
pollCommand (consumer)	设置在每次轮询循环期间要发送到远程 SSH 服务器的命令字符串。只适用于用作消费者的 camel-ssh 组件，例如 from.e. from (ssh://...)，您可能需要使用换行符结束您的命令，且必须采用 URL 编码的 %0A		字符串
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
channelType (advanced)	将频道类型设置为在命令执行过程中传递给频道。默认为 exec。	exec	字符串

Name	描述	默认值	类型
shellPrompt (advanced)	在命令执行后读取响应时，将 shell 提示符设置为丢弃		字符串
sleepForShellPrompt (advanced)	设置睡眠周期（以毫秒为单位），以等待从 shell 提示符读取响应。默认值为 100 毫秒。	100	long
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值

Name	描述	默认值	类型
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
certResource (security)	设置用于身份验证的证书的资源路径。将使用 ResourceHelperKeyPairProvider 解析基于文件的证书，并依赖于 keyType 设置。		字符串
keyPairProvider (security)	设置在使用证书连接到远程 SSH 服务器时使用的 KeyPairProvider 引用。		KeyPairProvider
keyType (security)	设置要传递给 KeyPairProvider 的密钥类型，作为身份验证的一部分。KeyPairProvider.loadKey (...)将传递这个值。默认为 ssh-rsa。	ssh-rsa	字符串
密码 (security)	设置用于连接远程 SSH 服务器的密码。需要 keyPairProvider 设置为 null。		字符串
用户名 (security)	设置登录远程 SSH 服务器的用户名。		字符串

328.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 30 个选项，如下所列。

Name	描述	默认值	类型
camel.component.ssh.cert-resource	设置用于身份验证的证书的资源路径。将使用 ResourceHelperKeyPairProvider 解析基于文件的证书，并依赖于 keyType 设置。		字符串
camel.component.ssh.channel-type	将频道类型设置为在命令执行过程中传递给频道。默认为 exec。		字符串
camel.component.ssh.configuration.cert-resource	设置用于身份验证的证书的资源路径。将使用 ResourceHelperKeyPairProvider 解析基于文件的证书，并依赖于 keyType 设置。		字符串
camel.component.ssh.configuration.channel-type	将频道类型设置为在命令执行过程中传递给频道。默认为 exec。	exec	字符串

Name	描述	默认值	类型
camel.component.ssh.configuration.fail-on-unknown-host	指定到未知主机的连接是否应该失败。只有在设置了属性 knownHosts 时，才会检查这个值。	false	布尔值
camel.component.ssh.configuration.host	设置远程 SSH 服务器的主机名。		字符串
camel.component.ssh.configuration.key-pair-provider	设置在使用证书连接到远程 SSH 服务器时使用的 KeyPairProvider 引用。		KeyPairProvider
camel.component.ssh.configuration.key-type	设置要传递给 KeyPairProvider 的密钥类型，作为身份验证的一部分。KeyPairProvider.loadKey (...)将传递这个值。默认为 ssh-rsa。	ssh-rsa	字符串
camel.component.ssh.configuration.known-hosts-resource	设置 known_hosts 文件的资源路径		字符串
camel.component.ssh.configuration.password	设置用于连接远程 SSH 服务器的密码。需要 keyPairProvider 设置为 null。		字符串
camel.component.ssh.configuration.poll-command	设置在每次轮询循环期间要发送到远程 SSH 服务器的命令字符串。只适用于用作消费者的 camel-ssh 组件，例如 from.e. from (ssh://...)，您可能需要使用换行符结束您的命令，且必须采用 URL 编码的 %0A		字符串
camel.component.ssh.configuration.port	设置远程 SSH 服务器的端口号。	22	整数
camel.component.ssh.configuration.shell-prompt	在命令执行后读取响应时，将 shell 提示符设置为丢弃		字符串
camel.component.ssh.configuration.sleep-for-shell-prompt	设置睡眠周期（以毫秒为单位），以等待从 shell 提示符读取响应。默认值为 100 毫秒。	100	Long

Name	描述	默认值	类型
camel.component.ssh.configuration.timeout	设置在建立远程 SSH 服务器连接时等待的超时时间（毫秒）。默认值为 30000 毫秒。	30000	Long
camel.component.ssh.configuration.username	设置登录远程 SSH 服务器的用户名。		字符串
camel.component.ssh.enabled	启用 ssh 组件	true	布尔值
camel.component.ssh.host	设置远程 SSH 服务器的主机名。		字符串
camel.component.ssh.key-pair-provider	设置在使用证书连接到远程 SSH 服务器时使用的 KeyPairProvider 引用。选项是一个 org.apache.sshd.common.keyprovider.KeyPairProvider 类型。		字符串
camel.component.ssh.key-type	设置要传递给 KeyPairProvider 的密钥类型，作为身份验证的一部分。KeyPairProvider.loadKey (...)将传递这个值。默认为 ssh-rsa。		字符串
camel.component.ssh.password	设置用于连接远程 SSH 服务器的密码。需要 keyPairProvider 设置为 null。		字符串
camel.component.ssh.poll-command	设置在每次轮询循环期间要发送到远程 SSH 服务器的命令字符串。只适用于用作消费者的 camel-ssh 组件，即 from (ssh://...)。您可能需要使用换行符来结束您的命令，并且必须采用 URL 编码 %0A		字符串
camel.component.ssh.port	设置远程 SSH 服务器的端口号。		整数
camel.component.ssh.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.ssh.shell-prompt	在命令执行后读取响应时，将 shell 提示符设置为丢弃		字符串
camel.component.ssh.sleep-for-shell-prompt	设置睡眠周期（以毫秒为单位），以等待从 shell 提示符读取响应。默认值为 100 毫秒。		Long

Name	描述	默认值	类型
camel.component.ssh.timeout	设置在建立远程 SSH 服务器连接时等待的超时时间（毫秒）。默认值为 30000 毫秒。		Long
camel.component.ssh.username	设置登录远程 SSH 服务器的用户名。		字符串
camel.component.ssh.cert-filename	设置用于身份验证的证书的资源路径。		字符串
camel.component.ssh.configuration.cert-filename	@deprecated as of 2.11，使用 {@link #setCertResource (String)} 替代		字符串

328.4. 使用 作为 PRODUCER 端点

当 SSH 组件用作 Producer （.to ("ssh://...")时，它将发送消息正文作为在远程 SSH 服务器上执行的命令。

以下是 XML DSL 中的示例：请注意，该命令具有 XML 编码的新行（）。

```
<route id="camel-example-ssh-producer">
  <from uri="direct:exampleSshProducer"/>
  <setBody>
    <constant>features:list&amp;#10;</constant>
  </setBody>
  <to uri="ssh://karaf:karaf@localhost:8101"/>
  <log message="${body}"/>
</route>
```

328.5. 身份验证

SSH 组件可以使用两个机制之一对远程 SSH 服务器进行身份验证：公钥证书或用户名/密码。根据如何和设定选项，配置 SSH 组件如何进行身份验证。

1. *首先，它将查看是否设置了 certResource 选项，如果已设置，则使用它来查找引用的公钥证书，并使用该证书进行身份验证。*

2. *如果没有设置 certResource，它将查看是否设置了 keyPairProvider，如果已设置，它将使用它来进行基于证书的身份验证。*

3. 如果没有设置 certResource 或 keyPairProvider，它将使用 用户名和密码 选项进行身份验证。即使在端点配置中提供了 用户名和密码，但使用 SshConstants.USERNAME_HEADER (CamelSshUsername)和 SshConstants.PASSWORD_HEADER (CamelSshPassword)设置的标头设置的端点配置会被使用。

以下路由片段演示了使用 classpath 中的证书进行 SSH 轮询消费者。

在 XML DSL 中，

```
<route>
  <from uri="ssh://scott@localhost:8101?
certResource=classpath:test_rsa&amp;useFixedDelay=true&amp;delay=5000&amp;pollCommand=features:list%0A"/>
  <log message="${body}"/>
</route>
```

在 Java DSL 中，

```
from("ssh://scott@localhost:8101?
certResource=classpath:test_rsa&useFixedDelay=true&delay=5000&pollCommand=features:list%0A")
    .log("${body}");
```

例如，/camel-example-ssh-security 中提供了使用公钥身份验证的示例。

证书依赖项

如果您使用基于证书的身份验证，则需要添加一些额外的运行时依赖项。显示的依赖项版本显示为 Camel 2.11，您可能需要根据您使用的 Camel 版本使用更新的版本。

```
<dependency>
  <groupId>org.apache.sshd</groupId>
  <artifactId>sshd-core</artifactId>
  <version>0.8.0</version>
</dependency>
<dependency>
  <groupId>org.bouncycastle</groupId>
  <artifactId>bcpg-jdk18on</artifactId>
  <version>1.72</version>
</dependency>
<dependency>
```

```
  <groupId>org.bouncycastle</groupId>
  <artifactId>bcpkix-jdk18on</artifactId>
  <version>1.72</version>
</dependency>
```

328.6. EXAMPLE

请参阅 Camel 发行版中的 examples/camel-example-ssh 和 example/camel-example-ssh-security。

328.7. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 329 章 STAX 组件

从 Camel 版本 2.9 开始提供

StAX 组件允许通过 SAX 内容处理程序处理消息。
此组件的另一项功能是使用 StAX 迭代 JAXB 记录，例如使用 Splitter EIP。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-stax</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

329.1. URI 格式

```
stax:content-handler-class
```

示例：

```
stax:org.superbiz.FooContentHandler
```

从 Camel 2.11.1 开始，您可以使用 # 语法从 Registry 查找 org.xml.sax.ContentHandler bean，如下所示：

```
stax:#myHandler
```

329.2. 选项

StAX 组件没有选项。

StAX 端点使用 URI 语法进行配置：

```
stax:contentHandlerClass
```

使用以下路径和查询参数：

329.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
contentHandlerClass	**必需** 要使用的 ContentHandler 实施的 The FQN 类名称。		字符串

329.2.2. 查询参数(1 参数)：

Name	描述	默认值	类型
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

329.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.stax.enabled	启用 stax 组件	true	布尔值
camel.component.stax.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

329.4. 使用内容处理程序作为 STAX 解析器

处理后的消息正文是处理程序本身。

下面是一个示例：

```
from("file:target/in")
  .to("stax:org.superbiz.handler.CountingHandler")
  // CountingHandler implements org.xml.sax.ContentHandler or extends
org.xml.sax.helpers.DefaultHandler
```

```
.process(new Processor() {
  @Override
  public void process(Exchange exchange) throws Exception {
     CountingHandler handler = exchange.getIn().getBody(CountingHandler.class);
     // do some great work with the handler
  }
});
```

329.5. 使用 JAXB 和 STAX 迭代集合

首先，我们假设您有 JAXB 对象。

例如，在 wrapper 对象中记录列表：

```
import java.util.ArrayList;
import java.util.List;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlAccessorType(XmlAccessType.FIELD)
@XmlRootElement(name = "records")
public class Records {
   @XmlElement(required = true)
   protected List<Record> record;

   public List<Record> getRecord() {
      if (record == null) {
         record = new ArrayList<Record>();
      }
      return record;
   }
}
```

and

```
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;

@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "record", propOrder = { "key", "value" })
public class Record {
   @XmlAttribute(required = true)
   protected String key;
```

```
    @XmlAttribute(required = true)
    protected String value;

    public String getKey() {
        return key;
    }

    public void setKey(String key) {
        this.key = key;
    }

    public String getValue() {
        return value;
    }

    public void setValue(String value) {
        this.value = value;
    }
}
```

然后，您可以获得一个 XML 文件来处理：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<records>
  <record value="v0" key="0"/>
  <record value="v1" key="1"/>
  <record value="v2" key="2"/>
  <record value="v3" key="3"/>
  <record value="v4" key="4"/>
  <record value="v5" key="5"/>
</record>
```

StAX 组件提供 StAXBuilder，可在通过 Camel Splitter 迭代 XML 元素时使用

```
from("file:target/in")
    .split(stax(Record.class)).streaming()
        .to("mock:records");
```

其中 stax 是 org.apache.camel.component.stax.StAXBuilder 上的静态方法，您可以在 Java 代码中静态导入。stax 构建器默认知道它所使用的 XMLReader。从 Camel 2.11.1 开始，您可以通过将布尔值参数设置为 false 来关闭此关闭，如下所示：

```
from("file:target/in")
    .split(stax(Record.class, false)).streaming()
        .to("mock:records");
```

329.5.1. 前面的带有 XML DSL 的示例

以上示例可以在 XML DSL 中实现

329.6. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 330 章 STOMP 组件

从 Camel 版本 2.12 开始提供

stomp: 组件用于与 Stomp 兼容消息代理通信，如 Apache ActiveMQ 或 ActiveMQ Apollo

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-stomp</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

330.1. URI 格式

```
stomp:queue:destination[?options]
```

其中 destination 是队列的名称。

330.2. 选项

Stomp 组件支持 8 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	使用共享的 stomp 配置		StompConfiguration
brokerURL (common)	要连接的 Stomp 代理的 URI		字符串
登录 （安全）	用户名		字符串
passcode (security)	密码		字符串
host (common)	虚拟主机		字符串

Name	描述	默认值	类型
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Stomp 端点使用 URI 语法进行配置：

```
stomp:destination
```

使用以下路径和查询参数：

330.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
目的地	队列 **必需** 名称		字符串

330.2.2. 查询参数(10 parameters):

Name	描述	默认值	类型
brokerURL (common)	**需要** 要连接的 Stomp 代理的 URI	tcp://localhost:61613	字符串
host (common)	虚拟主机名称		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值

Name	描述	默认值	类型
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
headerFilterStrategy (advanced)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
登录 （安全）	用户名		字符串
passcode (security)	密码		字符串
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters

330.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 13 个选项，如下所列。

Name	描述	默认值	类型
camel.component.stomp.broker-u-r-l	要连接的 Stomp 代理的 URI		字符串
camel.component.stomp.configuration.broker-u-r-l	要连接的 Stomp 代理的 URI	tcp://localhost:61613	字符串

Name	描述	默认值	类型
camel.component.stomp.configuration.host	虚拟主机名称		字符串
camel.component.stomp.configuration.login	用户名		字符串
camel.component.stomp.configuration.passcode	密码		字符串
camel.component.stomp.configuration.ssl-context-parameters	使用 SSLContextParameters 配置安全性		SSLContextParameters
camel.component.stomp.enabled	启用 stomp 组件	true	布尔值
camel.component.stomp.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤到 Camel 消息的标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		字符串
camel.component.stomp.host	虚拟主机		字符串
camel.component.stomp.login	用户名		字符串
camel.component.stomp.passcode	密码		字符串
camel.component.stomp.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.stomp.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

330.4. SAMPLES

发送消息：

```
from("direct:foo").to("stomp:queue:test");
```

使用消息：

```
from("stomp:queue:test").transform(body().convertToString()).to("mock:result")
```

330.5. ENDPOINTS

Camel 支持使用 Endpoint 接口的消息端点模式。端点通常由组件创建，端点通常通过其 URI 在 DSL 中引用。

从端点中，您可以使用以下方法

* createProducer （） 将创建一个 Producer 来向端点发送消息交换，它实现了 Event Driven Consumer 模式，以便在创建 Consumer * createPollingConsumer （） 时通过 Processor 使用来自端点的消息交换。 http://camel.apache.org/maven/current/camel-core/apidocs/org/apache/camel/Endpoint.html#createConsumer(org.apache.camel.Processor) http://camel.apache.org/maven/current/camel-core/apidocs/org/apache/camel/PollingConsumer.html

330.6. 另请参阅

- 配置 Camel
- 消息端点模式
- URI

- *编写组件*

第 331 章 流组件

从 Camel 版本 1.3 开始提供

stream: 组件提供对 System.in、system.out 和 System.err 流的访问，并允许处理文件和 URL。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-stream</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

331.1. URI 格式

```
stream:in[?options]
stream:out[?options]
stream:err[?options]
stream:header[?options]
```

另外，支持 文件和 url 端点 URI：

```
stream:file?fileName=/foo/bar.txt
stream:url[?options]
```

如果指定了 stream:header URI，则使用 流 标头来查找要写入的流。此选项仅适用于流制作者（即，它不能出现在 from （）中）。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

331.2. 选项

Stream 组件没有选项。

Stream 端点使用 URI 语法进行配置：

stream:kind

使用以下路径和查询参数：

331.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
kind	**必须使用** System.in 或 System.out 等流的 Kind。		字符串

331.2.2. 查询参数(22 参数)：

Name	描述	默认值	类型
encoding (common)	您可以将编码（是 charset 名称）配置为使用基于文本的流（例如，消息正文是一个 String 对象）。如果没有提供，Camel 将使用 JVM 默认 Charset。		字符串
filename (common)	使用 stream:file URI 格式时，这个选项指定要流传输到/from 的文件名。		字符串
url (common)	使用 stream:url URI 格式时，此选项指定要流到/来自的 URL。输入/输出流将使用 JDK URLConnection 工具打开。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
fileWatcher (consumer)	要使用 JVM 文件监视器侦听文件更改事件，以支持重新加载可能被覆盖的文件，比如 tail --retry	false	布尔值
groupLines (consumer)	要对消费者中的 X 行进行分组。例如，对 10 行进行分组，因此只会通过 10 行而不是每行 1 Exchange。		int
groupStrategy (consumer)	允许使用自定义 GroupStrategy 控制如何分组行。		GroupStrategy
initialPromptDelay (consumer)	显示消息提示前的初始延迟（毫秒）。这个延迟只进行一次。可以在系统启动期间使用，以避免在完成其他日志记录到系统外写入消息提示。	2000	long

Name	描述	默认值	类型
promptDelay (consumer)	显示消息提示前的可选延迟（以毫秒为单位）。		long
promptMessage (consumer)	从 stream:in 读取时使用的消息提示；例如，您可以将它设置为 Enter a 命令：		字符串
retry (consumer)	如果流被覆盖，它将重试打开，很像 tail --retry 一样从文件中读取，那么您还应启用 fileWatcher 选项，使它能够可靠。	false	布尔值
scanStream (consumer)	用于持续读取流，如 unix tail 命令。	false	布尔值
scanStreamDelay (consumer)	使用 scanStream 时读取尝试之间的延迟（毫秒）。		long
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
autoCloseCount (producer)	在 Producer 端关闭流前要处理的消息数量。默认情况下不关闭流（仅在 Producer 停止时）。如果发送了更多消息，则会为另一个 autoCloseCount 批处理重新打开流。		int
closeOnDone (producer)	此选项与 Splitter 和流化到同一文件结合使用。其理念是使流保持开放，且仅在执行 Splitter 时关闭，以提高性能。请注意，这需要您只流到同一文件，而不需要 2 个或更多文件。	false	布尔值
delay (producer)	生成流前的初始延迟（毫秒）。		long
connectTimeout (advanced)	设置指定超时值（以毫秒为单位），在打开与此 URLConnection 引用的资源的通信链接时使用。如果在建立连接前过期超时，则会引发 java.net.SocketTimeoutException。超时为零被解释为无限超时。		int
httpHeaders (advanced)	使用 HTTP URL 时请求中使用的可选 http 标头。		Map

Name	描述	默认值	类型
readTimeout (advanced)	将读取超时设置为指定的超时，以毫秒为单位。非零值指定连接与资源建立时从 Input 流读取时的超时。如果超时在有可供读取的数据前过期，则会引发 java.net.SocketTimeoutException。超时为零被解释为无限超时。		int
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

331.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component .stream.enabled	启用流组件	true	布尔值
camel.component .stream.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

331.4. 消息内容

***stream:** 组件支持 String 或 byte[] 来写入流。只需将 String 或 byte[] 内容添加到 message.in.body。在二进制模式中发送到 stream: producer 的消息不会跟随换行符（而不是 String 消息）。带有 null body 的消息不会附加到输出流中。*
特殊的 stream:header URI 用于自定义输出流。只需将 java.io.OutputStream 对象添加到密钥标头中的 message.in. header 中。
请参阅示例。

331.5. SAMPLES

在以下示例中，我们将信息从 direct:in 端点路由到 System.out 流：

```
// Route messages to the standard output.
from("direct:in").to("stream:out");

// Send String payload to the standard output.
// Message will be followed by the newline.
template.sendBody("direct:in", "Hello Text World");
```

```
// Send byte[] payload to the standard output.
// No newline will be added after the message.
template.sendBody("direct:in", "Hello Bytes World".getBytes());
```

以下示例演示了如何使用标头类型来确定要使用的流。在示例中，我们使用自己的输出流 MyOutputStream。

以下示例演示了如何持续读取文件流（类似于 UNIX tail 命令）：

```
from("stream:file?
fileName=/server/logs/server.log&scanStream=true&scanStreamDelay=1000")
  .to("bean:logService?method=parseLogLine");
```

一个带有 scanStream (pre Camel 2.7)或 scanStream + retry 的 getcha，文件将在每次 scanStreamDelay 迭代被重新打开并扫描。在 NIO2 可用之前，我们无法可靠地检测文件何时被删除/创建。

如果要重新加载文件，如果文件滚动/重写，则您还应打开 fileWatcher 和 retry 选项。

```
from("stream:file?
fileName=/server/logs/server.log&scanStream=true&scanStreamDelay=1000&retry=true&fileWa
tcher=true")
  .to("bean:logService?method=parseLogLine");
```

第 332 章 字符串编码数据格式

从 Camel 版本 2.12 开始提供

String Data Format 是一个基于文本的格式，它支持编码。

332.1. 选项

String Encoding dataformat 支持 2 个选项，如下所列。

Name	默认值	Java 类型	描述
charset		字符串	设置要使用的编码。默认情况下，将使用 JVM 平台默认 charset。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

332.2. MARSHAL

在本例中，我们以 UTF-8 编码将文件内容汇集为 String 对象。

```
from("file://data.csv").marshal().string("UTF-8").to("jms://myqueue");
```

332.3. UNMARSHAL

在本例中，我们在由 newOrder 处理器处理前，使用 UTF-8 编码从 JMS 队列到 String 对象的有效负载。

```
from("jms://queue/order").unmarshal().string("UTF-8").processRef("newOrder");
```

332.4. 依赖项

此数据格式以 camel-core 提供，因此不需要额外的依赖项。

第 333 章 字符串模板组件

从 Camel 版本 1.2 开始提供

string-template: 组件允许您使用 String Template 来处理消息。这在使用 Templating 生成请求的响应时是理想的选择。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-stringtemplate</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

333.1. URI 格式

```
string-template:templateName[?options]
```

其中 templateName 是要调用的模板的 classpath-local URI；或远程模板的完整 URL。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

333.2. 选项

String Template 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
allowContextMapAll (producer)	设置上下文映射是否应该允许访问所有详情。默认情况下，只能访问消息正文和标头。可以启用此选项以完全访问当前 Exchange 和 CamelContext。这样做会带来潜在的安全风险，因为这会打开对 CamelContext API 的完整功能的访问。	false	布尔值

Name	描述	默认值	类型
allowTemplateFromHeader (producer)	是否允许使用来自标头的资源模板（默认 false）。**启用此选项具有安全原因。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。**	false	布尔值

String Template 端点使用 URI 语法进行配置：

```
string-template:resourceUri
```

使用以下路径和查询参数：

333.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceUri	资源 **所需的** 路径。您可以使用这些协议(classpath 为 default)使用： classpath、file、http、ref 或 bean. classpath、file 和 http 加载资源的前缀。ref 将查找 registry 中的资源。bean 将调用 bean 以供用作资源的方法。对于 bean，您可以在点后指定方法名称，如 bean:myBean.myMethod。		字符串

333.2.2. 查询参数(6 参数)：

Name	描述	默认值	类型
allowContextMapAll (producer)	设置上下文映射是否应该允许访问所有详情。默认情况下，只能访问消息正文和标头。可以启用此选项以完全访问当前 Exchange 和 CamelContext。这样做会带来潜在的安全风险，因为这会打开对 CamelContext API 的完整功能的访问。	false	布尔值
allowTemplateFromHeader (producer)	是否允许使用来自标头的资源模板（默认 false）。**启用此选项具有安全原因。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。**	false	布尔值
contentCache (producer)	设置是否使用资源内容缓存	false	布尔值
delimiterStart (producer)	变量启动分隔符	<	char

Name	描述	默认值	类型
delimiterStop (producer)	变量结束分隔符	>	char
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

333.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.string-template.enabled	启用 string-template 组件	true	布尔值
camel.component.string-template.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

333.4. HEADERS

Camel 将在消息标头中存储对资源的引用，其键为 org.apache.camel.stringtemplate.resource。Resource 是一个 org.springframework.core.io.Resource 对象。

333.5. 热重新载入

对于文件和类路径资源(expanded jar)，字符串模板资源默认为 hot-reload。如果设置了 contentCache=true，Camel 只加载一次资源，且无法加载热重新加载。当资源永不更改时，此场景可用于生产环境。

333.6. STRINGTEMPLATE 属性

自 Camel 2.14 起，您可以通过设置消息标头 "CamelStringTemplateVariableMap" 来定义自定义上下文映射，就像以下代码一样。

```
Map<String, Object> variableMap = new HashMap<String, Object>();
```

```
Map<String, Object> headersMap = new HashMap<String, Object>();
headersMap.put("name", "Willem");
variableMap.put("headers", headersMap);
variableMap.put("body", "Monday");
variableMap.put("exchange", exchange);
exchange.getIn().setHeader("CamelStringTemplateVariableMap", variableMap);
```

333.7. SAMPLES

例如，您可以按照以下方式使用字符串模板来制定对消息的响应：

```
from("activemq:My.Queue").
  to("string-template:com/acme/MyResponse.tm");
```

333.8. 电子邮件示例

在本例中，我们希望使用字符串模板来发送订单确认电子邮件。电子邮件模板在 StringTemplate 中布局为：本例适用于 camel 2.11.0。如果您的 camel 版本小于 2.11.0，则变量应启动并以 $ 结尾。

```
Dear <headers.lastName>, <headers.firstName>

Thanks for the order of <headers.item>.

Regards Camel Riders Bookstore
<body>
```

java 代码如下：

333.9. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 334 章 STUB 组件

从 Camel 版本 2.10 开始提供

stub: 组件提供了一种在开发或测试过程中存出任何物理端点的简单方法，允许您运行路由，而无需实际连接到特定的 SMTP 或 Http 端点。只需将任何端点 URI 前面的 stub: 添加到存存端点。

Stub 组件内部 创建虚拟机 端点。Stub 和 VM 之间的主要区别在于，虚拟机将 验证您提供的 URI 和参数，因此将 vm: 置于带有查询参数的典型 URI 前通常会失败。然而，stub 不会完全忽略所有的查询参数，以便让您暂时避免路由中的一个或多个端点。

334.1. URI 格式

```
stub:someUri
```

其中 someUri 可以是具有任何查询参数的任何 URI。

334.2. 选项

Stub 组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
queueSize (advanced)	设置 SEDA 队列的默认最大容量（例如，它可以保存的消息数）。	1000	int
concurrentConsumers (consumer)	设置默认的并发线程处理交换数。	1	int
defaultQueueFactory (advanced)	设置默认队列工厂。		BlockingQueueFactory
defaultBlockWhenFull (producer)	将消息发送到完整 SEDA 队列的线程将阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出一个异常，表示队列已满。通过启用这个选项，调用线程将阻止并等待消息被接受。	false	布尔值

Name	描述	默认值	类型
defaultOfferTimeout (producer)	将消息发送到完整 SEDA 队列的线程将阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出一个异常，表示队列已满。通过启用这个选项，可以将配置超时添加到块问题单中。使用lining java 队列的 .offer (timeout)方法		long
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Stub 端点使用 URI 语法进行配置：

```
stub:name
```

使用以下路径和查询参数：

334.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	**所需** 队列的名称		字符串

334.2.2. 查询参数(17 参数)：

Name	描述	默认值	类型
size (common)	SEDA 队列的最大容量（例如，它可以保存的消息数）。默认情况下，将使用 SEDA 组件上设置的 defaultSize。	1000	int
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外，该处理程序记录在 WARN/ERROR 级别并忽略。	false	布尔值
concurrentConsumers (consumer)	并发线程处理交换的数量。	1	int

Name	描述	默认值	类型
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，该例外记录在 WARN/ERROR 级别并忽略。		ExceptionHandler
exchangePattern (consumer)	在创建交换时设置默认交换模式。		ExchangePattern
limitConcurrentConsumers (consumer)	是否将 concurrentConsumers 的数量限制为最多 500。默认情况下，如果端点配置了更大的数字，则会抛出异常。您可以通过关闭这个选项来禁用该检查。	true	布尔值
multipleConsumers (consumer)	指定是否允许多个消费者。如果启用，您可以使用 SEDA 进行 Publish-Subscribe 消息。也就是说，您可以向 SEDA 队列发送消息，并让每个消费者收到邮件的副本。启用后，应在每个消费者端点上指定此选项。	false	布尔值
pollTimeout (consumer)	轮询时使用的超时。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让使用者在关闭时更快地做出反应。	1000	int
purgeWhenStopping (consumer)	在停止 consumer/route 时是否清除任务队列。这允许更快地停止，因为队列中的任何待处理消息都会被丢弃。	false	布尔值
blockWhenFull (producer)	将消息发送到完整 SEDA 队列的线程将阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出一个异常，表示队列已满。通过启用这个选项，调用线程将阻止并等待消息被接受。	false	布尔值
discardIfNoConsumers (producer)	当发送到没有活跃消费者的队列时，生成者是否应该丢弃消息（不要将消息添加到队列）。可以同时启用其中一个选项 discardIfNoConsumers 和 failIfNoConsumers。	false	布尔值
failIfNoConsumers (producer)	当发送到没有活跃用户的队列时，生成者是否应该通过抛出异常。可以同时启用其中一个选项 discardIfNoConsumers 和 failIfNoConsumers。	false	布尔值
offerTimeout (producer)	当队列已满时，可以使用 offerTimeout （毫秒）添加到块问题单中。您可以使用 0 或负值禁用超时。		long
timeout (producer)	SEDA 生成者停止等待异步任务完成前的超时（以毫秒为单位）。您可以使用 0 或负值禁用超时。	30000	long

Name	描述	默认值	类型
waitForTaskToComplete (producer)	选项指定调用者是否应该等待 async 任务在继续前等待。支持以下三个选项： Always、Never 或 IfReplyExpected。前两个值为 self-explanatory。最后的值如果为 Request ReplyExpected，只有在消息是 Request Reply based 时等待。默认选项为 IfReplyExpected。	IfReplyExpected	WaitForTaskToComplete
队列 (advanced)	定义供端点使用的队列实例。这个选项只适用于您要使用自定义队列实例的个别用例。		BlockingQueue
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

334.3. 例子

以下是几个存存端点 uris 的示例

```
stub:smtp://somehost.foo.com?user=whatnot&something=else
stub:http://somehost.bar.com/something
```

部分 II. OPENAPI JAVA COMPONENT

从 Camel 3.1.0 开始提供

Rest DSL 可以与 camel-openapi-java 模块集成，该模块用于使用 OpenApi 公开 REST 服务及其 API。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-openapi-java</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

camel-openapi-java 模块可从 REST 组件中使用（不需要 servlet）

第 335 章 在 REST-DSL 中使用 OPENAPI

您可以通过配置 apiContextPath dsl 来启用 rest-dsl 中的 OpenApi api，如下所示：

```
public class UserRouteBuilder extends RouteBuilder {
    @Override
    public void configure() throws Exception {
        // configure we want to use servlet as the component for the rest DSL
        // and we enable json binding mode
        restConfiguration().component("netty-http").bindingMode(RestBindingMode.json)
            // and output using pretty print
            .dataFormatProperty("prettyPrint", "true")
            // setup context path and port number that netty will use
            .contextPath("/").port(8080)
            // add OpenApi api-doc out of the box
            .apiContextPath("/api-doc")
                .apiProperty("api.title", "User API").apiProperty("api.version", "1.2.3")
                // and enable CORS
                .apiProperty("cors", "true");

        // this user REST service is json only
        rest("/user").description("User rest service")
            .consumes("application/json").produces("application/json")
            .get("/{id}").description("Find user by id").outType(User.class)
                .param().name("id").type(path).description("The id of the user to
get").dataType("int").endParam()
                .to("bean:userService?method=getUser(${header.id})")
            .put().description("Updates or create a user").type(User.class)
                .param().name("body").type(body).description("The user to update or
create").endParam()
                .to("bean:userService?method=updateUser")
            .get("/findAll").description("Find all users").outType(User[].class)
                .to("bean:userService?method=listUsers");
    }
}
```

第 336 章 选项

可以使用以下选项配置 OpenApi 模块。要使用您如上所示的 init-param 使用 servlet 来配置。在 rest-dsl 中直接配置时，您可以使用适当的方法，如 enableCORS,host,contextPath, dsl。带有 api.xxx 的选项使用 apiProperty dsl 进行配置。

选项	类型	描述
CORS	布尔值	是否启用 CORS。请注意，这只为 api 浏览器启用 CORS，而不启用对 REST 服务的实际访问权限。默认为 false。
openapi.version	字符串	OpenAPI spec 版本。默认为 3.0。
主机	字符串	设置主机名：如果没有配置 camel-openapi-java，则会计算名称为 localhost based。
schemes	字符串	要使用的协议方案。可以使用逗号分隔多个值，如 "http,https"。默认值为 "http"。
base.path	字符串	**必需** ：设置 REST 服务可用的基本路径。该路径是 relative （例如不以 http/https 开头），camel-openapi-java 将在运行时计算绝对路径，其为 **protocol://host:port/context-path/base.path**
api.path	字符串	要设置 API 可用的路径（如 /api-docs）。该路径是 relative （例如，没有以 http/https 开头的），camel-openapi-java 将在运行时计算绝对路径，即 **protocol://host:port/context-path/api.path** So 使用相对路径。请参阅以上示例。
api.version	字符串	api 的版本。默认为 0.0.0。
api.title	字符串	应用程序的标题。
api.description	字符串	应用程序的简短描述。
api.termsOfService	字符串	API 条款的 URL。
api.contact.name	字符串	要联系的人员或机构的名称
api.contact.email	字符串	用于与 API 相关的相应电子邮件。

选项	类型	描述
api.contact.url	字符串	网站的 URL 获取更多信息。
api.license.name	字符串	用于 API 的许可证名称。
api.license.url	字符串	用于 API 的许可证的 URL。
apiContextIdListing	布尔值	是否允许列出具有 REST 服务的 JVM 中的所有 CamelContext 名称。启用后，api-doc 的根路径将列出所有上下文。当禁用时，没有列出上下文 id，api-doc 的根路径会列出当前的 CamelContext。默认为 false。
apiContextIdPattern	字符串	允许过滤在上下文列表中显示哪个 CamelContext 名称的模式。该模式使用正则表达式，作为通配符。其与 Intercept 使用的模式匹配

第 337 章 在 API 文档中添加安全定义

从 Camel 3.1.0 开始提供

Rest DSL 现在支持在生成的 API 文档中声明 OpenApi securityDefinitions。例如：

```
rest("/user").tag("dude").description("User rest service")
    // setup security definitions
    .securityDefinitions()
        .oauth2("petstore_auth").authorizationUrl("http://petstore.swagger.io/oauth/dialog").end()
        .apiKey("api_key").withHeader("myHeader").end()
    .end()
    .consumes("application/json").produces("application/json")
```

此处我们设置了两个安全定义

- OAuth2 - 带有提供的 url 的隐式授权
- API Key - 使用来自名为 myHeader的 HTTP 标头的 api 键

然后，您需要通过引用其密钥(petstore_auth 或 api_key)指定安全性使用的其他操作。

```
.get("/{id}/{date}").description("Find user by id and date").outType(User.class)
    .security("api_key")

...

.put().description("Updates or create a user").type(User.class)
    .security("petstore_auth", "write:pets,read:pets")
```

此处 get 操作使用的是 Api Key 安全性，而 put 操作则使用带有允许的读取和写入 pet 范围的 OAuth 安全性。

第 338 章 CONTEXTIDLISTING ENABLED

启用 contextIdListing 后，它会检测同一 JVM 中所有正在运行的 CamelContexts。这些上下文以 root 路径的形式列出，例如 /api-docs 作为 json 格式的简单名称列表。要访问 OpenApi 文档，则必须将 context-path 附加到 Camel 上下文 ID，如 api-docs/myCamel。选项 apiContextIdPattern 可用于过滤此列表中的名称。

第 339 章 JSON 或 YAML

* 从 Camel 3.1.0 开始提供

camel-openapi-java 模块支持开箱即用的 JSon 和 Yaml。您可以在请求 url 中指定您要使用 /openapi.json 或 /openapi.yaml 返回的内容。如果没有指定，则使用 HTTP Accept 标头来检测是否可以接受 json 或 yaml。如果两者都是 accepted 或 none，则 json 将返回默认格式。

第 340 章 例子

在 Apache Camel 发行版中，我们提供 camel-example-openapi-cdi 和 camel-example-openapi-java，它演示了使用此 OpenApi 组件。

第 341 章 SWAGGER JAVA 组件

从 Camel 2.16 开始提供

Rest DSL 可以与 camel-swagger-java 模块集成，该模块使用 Swagger 来公开 REST 服务及其 API。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-swagger-java</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

camel-swagger-java 模块可从 REST 组件中使用（不需要 servlet）

341.1. 在 REST-DSL 中使用 SWAGGER

您可以通过配置 apiContextPath dsl 来启用 rest-dsl 中的 swagger api，如下所示：

```
public class UserRouteBuilder extends RouteBuilder {
  @Override
  public void configure() throws Exception {
    // configure we want to use servlet as the component for the rest DSL
    // and we enable json binding mode
    restConfiguration().component("netty4-http").bindingMode(RestBindingMode.json)
      // and output using pretty print
      .dataFormatProperty("prettyPrint", "true")
      // setup context path and port number that netty will use
      .contextPath("/").port(8080)
      // add swagger api-doc out of the box
      .apiContextPath("/api-doc")
        .apiProperty("api.title", "User API").apiProperty("api.version", "1.2.3")
        // and enable CORS
        .apiProperty("cors", "true");

    // this user REST service is json only
    rest("/user").description("User rest service")
      .consumes("application/json").produces("application/json")
      .get("/{id}").description("Find user by id").outType(User.class)
        .param().name("id").type(path).description("The id of the user to
get").dataType("int").endParam()
```

```
            .to("bean:userService?method=getUser(${header.id})")
        .put().description("Updates or create a user").type(User.class)
            .param().name("body").type(body).description("The user to update or
create").endParam()
            .to("bean:userService?method=updateUser")
        .get("/findAll").description("Find all users").outTypeList(User.class)
            .to("bean:userService?method=listUsers");
    }
}
```

341.2. 选项

可以使用以下选项配置 swagger 模块：要使用您如上所示的 init-param 使用 servlet 来配置。在 rest-dsl 中直接配置时，您可以使用适当的方法，如 enableCORS,host,contextPath, dsl。带有 api.xxx 的选项使用 apiProperty dsl 进行配置。

选项	类型	描述
CORS	布尔值	是否启用 CORS。请注意，这只为 api 浏览器启用 CORS，而不启用对 REST 服务的实际访问权限。默认为 false。
swagger.version	字符串	Swagger spec 版本。默认 2.0。
主机	字符串	设置主机名：如果没有配置 camel-swagger-java，则会计算名称为 localhost。
模式	字符串	要使用的协议方案。可以使用逗号分隔多个值，如 "http,https"。默认值为 "http"。这个选项在 Camel 2.17 以后 **被弃用**，因为它应该被命名为 schemes。
schemes	字符串	**Camel 2.17：** 要使用的协议方案。可以使用逗号分隔多个值，如 "http,https"。默认值为 "http"。
base.path	字符串	**必需** ：设置 REST 服务可用的基本路径。该路径是 relative （例如不以 http/https 开头），camel-swagger-java 将在运行时计算绝对路径，其为 **protocol://host:port/context-path/base.path**
api.path	字符串	要设置 API 可用的路径（如 /api-docs）。该路径是 relative （例如，没有以 http/https 开头的），camel-swagger-java 将在运行时计算绝对路径，它将是 **protocol://host:port/context-path/api.path** So，使用相对路径更为容易。请参阅以上示例。
api.version	字符串	api 的版本。默认为 0.0.0。
api.title	字符串	应用程序的标题。

选项	类型	描述
api.description	字符串	应用程序的简短描述。
api.termsOfService	字符串	API 条款的 URL。
api.contact.name	字符串	要联系的人员或机构的名称
api.contact.email	字符串	用于与 API 相关的相应电子邮件。
api.contact.url	字符串	网站的 URL 获取更多信息。
api.license.name	字符串	用于 API 的许可证名称。
api.license.url	字符串	用于 API 的许可证的 URL。
apiContextIdListing	布尔值	是否允许列出具有 REST 服务的 JVM 中的所有 CamelContext 名称。启用后，api-doc 的根路径将列出所有上下文。当禁用时，没有列出上下文 id，api-doc 的根路径会列出当前的 CamelContext。默认为 false。
apiContextIdPattern	字符串	允许过滤在上下文列表中显示哪个 CamelContext 名称的模式。该模式使用正则表达式，作为通配符。其与 Intercept 使用的模式匹配

341.3. 在 API 文档中添加安全定义

从 Camel 2.22.0 开始提供

Rest DSL 现在支持在生成的 API 文档中声明 Swagger securityDefinitions。例如：

```
rest("/user").tag("dude").description("User rest service")
    // setup security definitions
    .securityDefinitions()
        .oauth2("petstore_auth").authorizationUrl("http://petstore.swagger.io/oauth/dialog").end()
```

```
        .apiKey("api_key").withHeader("myHeader").end()
    .end()
    .consumes("application/json").produces("application/json")
```

此处我们设置了两个安全定义

- OAuth2 - 带有提供的 url 的隐式授权
- API Key - 使用来自名为 myHeader的 HTTP 标头的 api 键

然后，您需要通过引用其密钥(petstore_auth 或 api_key)指定安全性使用的其他操作。

```
.get("/{id}/{date}").description("Find user by id and date").outType(User.class)
    .security("api_key")

...

.put().description("Updates or create a user").type(User.class)
    .security("petstore_auth", "write:pets,read:pets")
```

此处 get 操作使用的是 Api Key 安全性，而 put 操作则使用带有允许的读取和写入 pet 范围的 OAuth 安全性。

341.4. CONTEXTIDLISTING ENABLED

启用 contextIdListing 后，它会检测同一 JVM 中所有正在运行的 CamelContexts。这些上下文以 root 路径的形式列出，例如 /api-docs 作为 json 格式的简单名称列表。要访问 swagger 文档，则必须将 context-path 附加到 Camel 上下文 ID，如 api-docs/myCamel。选项 apiContextIdPattern 可用于过滤此列表中的名称。

341.5. JSON 或 YAML

从 Camel 2.17 开始提供

camel-swagger-java 模块支持 JSon 和 Yaml 开箱即用。您可以在请求 url 中指定您希望使用 /swagger.json 或 /swagger.yaml 返回的内容。如果没有指定，则使用 HTTP Accept 标头来检测是否可以接受 json 或 yaml。如果两者都是 accepted 或 none，则 json 将返回默认格式。

341.6. 例子

在 Apache Camel 发行版中，我们提供 camel-example-swagger-cdi 和 camel-example-swagger-java，它演示了使用此 Swagger 组件。

第 342 章 SYSLOG DATAFORMAT

从 Camel 版本 2.6 开始提供

syslog dataformat 用于处理 RFC3164 和 RFC5424 信息。

这个组件支持以下内容：

- syslog 信息的 UDP 消耗
- 使用普通 String 对象或 SyslogMessage 模型对象的数据格式。
- 键入 Converter from/to SyslogMessage 和 String
- 与 camel-mina 组件集成。
- 与 camel-netty 组件集成。
- Camel 2.14： camel-netty 组件的 Encoder 和 decoder。
- Camel 2.14： 对 RFC5424 的支持也.

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-syslog</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

342.1. RFC3164 SYSLOG 协议

syslog 使用用户数据报协议(UDP) 1 作为其底层传输层机制。分配给 syslog 的 UDP 端口为 514。

要公开 Syslog 侦听器服务，我们重复使用现有的 camel-mina 组件或 camel-netty，我们将 Rfc3164SyslogDataFormat 用于 marshal 和 unmarshal 消息。请注意，从 Camel 2.14 开始，syslog dataformat 被重命名为 SyslogDataFormat。

342.2. 选项

Syslog dataformat 支持 1 个选项，如下所列。

Name	默认值	Java 类型	描述
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

342.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.syslog.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.syslog.enabled	启用 syslog dataformat	true	布尔值

ND

342.4. RFC5424 SYSLOG 协议

从 Camel 2.14 开始提供

要公开 Syslog 侦听器服务，我们重复使用现有的 camel-mina 组件或 camel-netty，其中我们仅使用 SyslogDataFormat 进行 marshal 和 unmarshal 消息

342.4.1. 公开 Syslog 侦听器

在 Spring XML 文件中，我们将端点配置为侦听端口 10514 上的 udp 信息，请注意，在 netty 中，禁用 defaultCodec，这个
将允许回退到 NettyTypeConverter，并以 InputStream 形式提供消息：

```
<camelContext id="myCamel" xmlns="http://camel.apache.org/schema/spring">

  <dataFormats>
      <syslog id="mySyslog"/>
  </dataFormats>

  <route>
      <from uri="netty:udp://localhost:10514?sync=false&amp;allowDefaultCodec=false"/>
      <unmarshal ref="mySyslog"/>
      <to uri="mock:stop1"/>
  </route>

</camelContext>
```

使用 camel-mina的同一路由

```
<camelContext id="myCamel" xmlns="http://camel.apache.org/schema/spring">

  <dataFormats>
      <syslog id="mySyslog"/>
  </dataFormats>

  <route>
      <from uri="mina:udp://localhost:10514"/>
      <unmarshal ref="mySyslog"/>
      <to uri="mock:stop1"/>
  </route>

</camelContext>
```

342.4.2. 将 syslog 消息发送到远程目的地

```
<camelContext id="myCamel" xmlns="http://camel.apache.org/schema/spring">

  <dataFormats>
     <syslog id="mySyslog"/>
  </dataFormats>

  <route>
```

```
        <from uri="direct:syslogMessages"/>
        <marshal ref="mySyslog"/>
        <to uri="mina:udp://remotehost:10514"/>
    </route>

</camelContext>
```

342.5. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 343 章 TAR 文件数据格式

从 Camel 版本 2.16 开始提供

Tar File Data Format 是消息压缩和解压缩格式。可将消息合并（解压缩）到包含单个条目的 Tar Files，而包含单个条目的 Tar 文件可以 unmarshalled （解压缩）到原始文件内容。

还有一个聚合策略，可以将多个消息聚合到一个 Tar 文件中。

343.1. TARFILE 选项

Tar File dataformat 支持 4 个选项，如下所列。

Name	默认值	Java 类型	描述
usingIterator	**false**	布尔值	如果 tar 文件有多个条目，则将此选项设置为 true，允许使用 splitter EIP 来在流传输模式中使用迭代数据。
allowEmptyDirectory	**false**	布尔值	如果 tar 文件有多个条目，则将此选项设置为 true，则即使目录为空，也允许获取迭代器
preservePathElements	**false**	布尔值	如果文件名包含路径元素，请将此选项设置为 true，允许在 tar 文件中维护路径。
contentTypeHeader	**false**	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

343.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.tarfile.allow-empty-directory	如果 tar 文件有多个条目，则将此选项设置为 true，则即使目录为空，也允许获取迭代器	false	布尔值

Name	描述	默认值	类型
camel.dataformat.tarfile.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.tarfile.enabled	启用 tarfile dataformat	true	布尔值
camel.dataformat.tarfile.preserve-path-elements	如果文件名包含路径元素，请将此选项设置为 true，允许在 tar 文件中维护路径。	false	布尔值
camel.dataformat.tarfile.using-iterator	如果 tar 文件有多个条目，则将此选项设置为 true，允许使用 splitter EIP 来在流传输模式中使用迭代数据。	false	布尔值

ND

343.3. MARSHAL

在本例中，我们使用 Tar 文件压缩将常规文本/XML 有效负载放入压缩有效负载，并将它发送到名为 MY_QUEUE 的 ActiveMQ 队列。

```
from("direct:start").marshal().tarFile().to("activemq:queue:MY_QUEUE");
```

创建 Tar File 中的 Tar 条目的名称基于传入的 CamelFileName 消息标头，这是文件组件使用的标准消息标头。另外，传出 CamelFileName 消息标头会自动设置为传入的 CamelFileName 消息标头的值，带有 ".tar" 后缀。因此，如果以下路由在输入目录中找到名为 "test.txt" 的文件，则输出将是名为 "test.txt.tar" 的 Tar 文件，其中包含一个名为 "test.txt" 的 Tar 条目：

```
from("file:input/directory?antInclude=*/.txt").marshal().tarFile().to("file:output/directory");
```

如果没有传入的 CamelFileName 消息标头（例如，如果文件组件不是消费者），则默认使用消息 ID，因为消息 ID 通常是唯一的生成的 ID，因此您将以文件名（如 ID-MACHINENAME-2443-1211718892437-1-0.tar ）结尾。如果要覆盖此行为，您可以在路由中显式设置 CamelFileName 标头的值：

```
from("direct:start").setHeader(Exchange.FILE_NAME,
constant("report.txt")).marshal().tarFile().to("file:output/directory");
```

此路由会导致输出目录中名为 "report.txt.tar" 的 Tar 文件，其中包含一个名为 "report.txt" 的 Tar 条目。

343.4. UNMARSHAL

在本例中，我们从名为 MY_QUEUE 的 ActiveMQ 队列到其原始格式，并将它转发到 UnTarpedMessageProcessor，将 Tar File payload 转发到其原始格式。

```
from("activemq:queue:MY_QUEUE").unmarshal().tarFile().process(new
UnTarpedMessageProcessor());
```

如果 Tar File 有多个条目，则 TarFileDataFormat 的 useIterator 选项为 true，您可以使用 splitter 来进一步工作。

```
 TarFileDataFormat tarFile = new TarFileDataFormat();
 tarFile.setUsingIterator(true);
 from("file:src/test/resources/org/apache/camel/dataformat/tarfile/?
consumer.delay=1000&noop=true")
   .unmarshal(tarFile)
   .split(body(Iterator.class))
      .streaming()
       .process(new UnTarpedMessageProcessor())
   .end();
```

或者您可以使用 TarSplitter 作为直接分割器的表达式，如下所示

```
  from("file:src/test/resources/org/apache/camel/dataformat/tarfile?
consumer.delay=1000&noop=true")
    .split(new TarSplitter())
      .streaming()
      .process(new UnTarpedMessageProcessor())
    .end();
```

343.5. 聚合

INFO:Please 注意此聚合策略需要 eager completion 检查才能正常工作。

在这个示例中，我们将在输入目录中找到的所有文本文件聚合到一个存储在输出目录中的 Tar 文件中。

```
  from("file:input/directory?antInclude=*/.txt")
```

```
    .aggregate(new TarAggregationStrategy())
      .constant(true)
      .completionFromBatchConsumer()
      .eagerCheckCompletion()
  .to("file:output/directory");
```

传出 CamelFileName 消息标头使用 java.io.File.createTempFile 创建，后缀为 ".tar"。如果要覆盖此行为，您可以在路由中显式设置 CamelFileName 标头的值：

```
  from("file:input/directory?antInclude=*/.txt")
    .aggregate(new TarAggregationStrategy())
      .constant(true)
      .completionFromBatchConsumer()
      .eagerCheckCompletion()
    .setHeader(Exchange.FILE_NAME, constant("reports.tar"))
  .to("file:output/directory");
```

343.6. 依赖项

要使用 camel 路由中的 Tar Files，您需要添加一个对 camel-tarfile 的依赖关系来实现这个数据格式。

如果使用 Maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-tarfile</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

第 344 章 TELEGRAM 组件

从 Camel 版本 2.18 开始提供

Telegram 组件提供对 Telegram Bot API 的访问。它允许基于 Camel 的应用程序通过充当 Bot 来发送和接收消息，并参与与普通用户、私有和公共组或频道直接对话。

在使用此组件之前，必须按照 Telegram Bot 开发人员主页中的说明创建 Telegram Bot。创建新 Bot 时，B owtFather 提供与 Bot 对应的 授权令牌。授权令牌是 camel-telegram 端点的强制参数。

注意

为了允许 Bot 接收在组或频道内交换的所有消息（不仅仅是以 '/' 字符开头），请 BotFather 使用 /setprivacy 命令禁用隐私模式。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-telegram</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

344.1. URI 格式

```
telegram:type/authorizationToken[?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

344.2. 选项

Telegram 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
authorizationToken (security)	当端点中没有提供信息时，要使用的默认 Telegram 授权令牌。		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Telegram 端点使用 URI 语法进行配置：

```
telegram:type/authorizationToken
```

使用以下路径和查询参数：

344.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
type	**必需** 端点类型。目前，只支持 'bots' 类型。		字符串
authorizationToken	**必需** 使用 bot 的授权令牌（作为 BotFather）、eg. 654321531:HGF_dTra456323dHuOedsE343211fqr3t-H。		字符串

344.2.2. 查询参数(22 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
limit (consumer)	限制在单个轮询请求中接收的更新数量。	100	整数
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值

Name	描述	默认值	类型
timeout (consumer)	长时间轮询的超时时间（以秒为单位）。使用 0 进行简短轮询或更长的数字进行长时间轮询。长轮询会生成较短的响应时间。	30	整数
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
chatId (producer)	接收生成消息的聊天的标识符。chat ids 首先从传入消息获得（例如，当电话用户与 bot 进行对话时，其客户端会自动发送包含 chat id 的"/start"消息）。这是一个可选参数，因为可以为每个传出消息（使用正文或标头）动态设置 chat id。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值

Name	描述	默认值	类型
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

344.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.telegram.authorization-token	当端点中没有提供信息时，要使用的默认 Telegram 授权令牌。		字符串
camel.component.telegram.enabled	启用 telegram 组件	true	布尔值

Name	描述	默认值	类型
camel.component.telegram.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

344.4. 消息标头

Name	描述
CamelTelegramChatId	producer 端点使用此标头来解析将接收消息的 chat id。接收者 chat id 可以在消息正文、**CamelTelegramChatId** 标头或端点配置(**chatId** 选项)中放置。此标头也存在于所有传入的消息中。
CamelTelegramMediaType	当传出消息由纯二进制数据组成时，此标头用于识别介质类型。可能的值有字符串或 enum 值，属于 **org.apache.camel.component.telegram.TelegramMediaType** enumeration。
CamelTelegramMediaTitleCaption	此标头用于为传出二进制消息提供标题或标题。
CamelTelegramParseMode	此标头用于格式化使用 HTML 或 Markdown 的文本消息（请参阅 **org.apache.camel.component.telegram.TelegramParseMode**）。

344.5. 使用方法

Telegram 组件支持使用者和制作者端点。它还可用于 被动 chat-bot 模式 （使用，然后生成消息）。

344.6. 生成者示例

以下是如何通过 Telegram Bot API 向 Telegram chat 发送消息的基本示例。

Java DSL

```
from("direct:start").to("telegram:bots/123456789:insertYourAuthorizationTokenHere");
```

或在 Spring XML 中

```
<route>
```

```
    <from uri="direct:start"/>
    <to uri="telegram:bots/123456789:insertYourAuthorizationTokenHere"/>
<route>
```

代码 123456789:insertYourAuthorizationTokenHere 是与 Bot 对应的 授权令牌。

在不指定 chat id 选项的情况下使用 producer 端点时，将使用消息正文或标头中包含的信息来识别目标聊天。生产者端点允许以下消息正文（类型为 OutgoingXXXMessage 的消息属于软件包 org.apache.camel.component.telegram.model）

Java 类型	描述
OutgoingTextMessage	向聊天发送文本消息
OutgoingPhotoMessage	将照片(JPG, PNG)发送到聊天
OutgoingAudioMessage	将 mp3 音频发送到聊天
OutgoingVideoMessage	向聊天发送 mp4 视频
OutgoingDocumentMessage	要发送文件到聊天（任意介质类型）
byte[]	发送任何受支持的介质类型。它需要 **CamelTelegramMediaType** 标头设置为适当的介质类型
字符串	要发送文本消息到聊天，请执行以下操作：它会自动转换为 **OutgoingTextMessage**

344.7. 消费者示例

以下是如何接收电话用户发送到配置的 Bot 的所有消息的基本示例。In Java DSL

```
from("telegram:bots/123456789:insertYourAuthorizationTokenHere")
.bean(ProcessorBean.class)
```

或在 Spring XML 中

```
<route>
    <from uri="telegram:bots/123456789:insertYourAuthorizationTokenHere"/>
    <bean ref="myBean" />
<route>

<bean id="myBean" class="com.example.MyBean"/>
```

MyBean 是一个将接收消息的简单 bean

```
public class MyBean {

    public void process(String message) {
        // or Exchange, or org.apache.camel.component.telegram.model.IncomingMessage (or
both)

        // do process
    }

}
```

传入消息支持的类型有

Java 类型	描述
IncomingMessage	传入消息的完整对象表示
字符串	消息的内容，仅用于文本消息

344.8. REACTIVE CHAT-BOT 示例

被动 chat-bot 模式是使用 Camel 组件构建一个简单的聊天 bot 的方法，它直接回复从 Telegram 用户收到的聊天消息。

以下是 Java DSL 中 chat-bot 的基本配置

```
from("telegram:bots/123456789:insertYourAuthorizationTokenHere")
.bean(ChatBotLogic.class)
.to("telegram:bots/123456789:insertYourAuthorizationTokenHere");
```

或在 Spring XML 中

```
<route>
    <from uri="telegram:bots/123456789:insertYourAuthorizationTokenHere"/>
    <bean ref="chatBotLogic" />
    <to uri="telegram:bots/123456789:insertYourAuthorizationTokenHere"/>
<route>

<bean id="chatBotLogic" class="com.example.ChatBotLogic"/>
```

ChatBotLogic 是一个简单的 bean，它实现了一个通用的 String-to-String 方法。

```
public class ChatBotLogic {

    public String chatBotProcess(String message) {
        if( "do-not-reply".equals(message) ) {
            return null; // no response in the chat
        }

        return "echo from the bot: " + message; // echoes the message
    }

}
```

chatBotProcess 方法返回的每个非null 字符串都会自动路由到源自请求的聊天（因为 CamelTelegramChatId 标头用于路由消息）。

344.9. 获取聊天 ID

如果要在发生事件时将消息推送到特定的 Telegram chat 中，您需要检索对应的 chat ID。chat ID 目前没有在 telegram 客户端中显示，但您可以使用一个简单的路由来获取它。

首先，将 bot 添加到您要推送消息的聊天中，然后运行类似以下的路由：

```
from("telegram:bots/123456789:insertYourAuthorizationTokenHere")
.to("log:INFO?showHeaders=true");
```

bot 收到的任何消息都将转储到您的日志中，以及有关聊天(CamelTelegramChatId 标头)的信息。

获得 chat ID 后，您可以使用以下示例路由将消息推送到它。

```
from("timer:tick")
.setBody().constant("Hello")
to("telegram:bots/123456789:insertYourAuthorizationTokenHere?chatId=123456")
```

请注意，对应的 URI 参数只是 chatId。

344.10. 自定义键盘

您可以自定义用户键盘，而不是询问他写出一个选项。OutgoingTextMessage 具有属性 ReplyKeyboardMarkup，可用于此类事情。

```
from("telegram:bots/123456789:insertYourAuthorizationTokenHere")
    .process(exchange -> {

        OutgoingTextMessage msg = new OutgoingTextMessage();
        msg.setText("Choose one option!");

        InlineKeyboardButton buttonOptionOneI = InlineKeyboardButton.builder()
                .text("Option One - I").build();

        InlineKeyboardButton buttonOptionOneII = InlineKeyboardButton.builder()
                .text("Option One - II").build();

        InlineKeyboardButton buttonOptionTwoI = InlineKeyboardButton.builder()
                .text("Option Two - I").build();

        ReplyKeyboardMarkup replyMarkup = ReplyKeyboardMarkup.builder()
                .keyboard()
                    .addRow(Arrays.asList(buttonOptionOneI, buttonOptionOneII))
                    .addRow(Arrays.asList(buttonOptionTwoI))
                    .close()
                .oneTimeKeyboard(true)
                .build();

        msg.setReplyKeyboardMarkup(replyMarkup);

        exchange.getIn().setBody(msg);
    })
    .to("telegram:bots/123456789:insertYourAuthorizationTokenHere");
```

如果要禁用它，下一个消息必须具有 ReplyKeyboardMarkup 对象上的属性 removeKeyboard 设置。

```
from("telegram:bots/123456789:insertYourAuthorizationTokenHere")
    .process(exchange -> {

        OutgoingTextMessage msg = new OutgoingTextMessage();
        msg.setText("Your answer was accepted!");

        ReplyKeyboardMarkup replyMarkup = ReplyKeyboardMarkup.builder()
                .removeKeyboard(true)
                .build();

        msg.setReplyKeyboardMarkup(replyMarkup);

        exchange.getIn().setBody(msg);
    })
    .to("telegram:bots/123456789:insertYourAuthorizationTokenHere");
```

第 345 章 测试组件

从 Camel 版本 1.3 开始提供

测试分布式和异步处理非常困难。Mock、Test 和 DataSet 端点与 Camel 测试框架协同工作，从而通过使用 企业集成模式 和 Camel 的大量组件以及强大的 Bean 集成来简化您的单元和集成测试。

测试 组件扩展了 Mock 组件，以支持在启动时从另一个端点拉取信息，以在底层 Mock 端点上设置预期的消息正文。也就是说，您在路由中使用 test 端点，以及到达它的消息将隐式地与从某些其它位置提取的一些预期消息进行比较。

例如，您可以使用一组预期的消息正文作为文件。然后，这将设置一个正确配置的 Mock 端点，该端点只有在收到的消息与预期消息的数量匹配时才有效，并且其消息有效负载相等。

在使用 Camel 2.8 或更早的版本时，Maven 用户需要将以下依赖项添加到此组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-spring</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

从 Camel 2.9 开始，在 camel-core 中直接提供 Test 组件。

345.1. URI 格式

```
test:expectedMessagesEndpointUri
```

其中 expectedMessagesEndpointUri 是指启动测试前从中拉取预期消息正文的一些其他组件 URI。

345.2. URI 选项

Test 组件没有选项。

Test 端点使用 URI 语法进行配置：

test:name

使用以下路径和查询参数：

345.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	**注册表中** 用于轮询消息的端点名称（用于测试）		字符串

345.2.2. 查询参数(14 参数)：

Name	描述	默认值	类型
anyOrder (producer)	预期的消息是否应以相同的顺序到达，还是可以按照任何顺序进行。	false	布尔值
assertPeriod (producer)	设置一个宽限期，之后 mock 端点将重新断言，以确保初始断言仍然有效。例如，这只用于表示多个消息到达的 assert。例如，如果 expectedMessageCount (int)设为 5，则当 5 个或更多消息到达时，会满足断言。为确保准确 5 个消息到达，您需要稍等片刻，以确保进一步的消息到达。这是您可以对此使用 setAssertPeriod （长）方法的内容。默认情况下禁用这个周期。	0	long
delimiter (producer)	启用分割时使用的分割分隔符。默认情况下，delimiter 是基于新行。分隔符可以是正则表达式。		字符串
expectedCount (producer)	指定此端点应接收的消息交换数量。注意：如果要预期 0 个消息，请在测试启动时做额外的小心，以 0 匹配，因此您需要设置一个 assert 周期时间以便让测试在一段时间内运行，以确保仍然没有消息到达；对于 setAssertPeriod （长）。另一种方法是使用 NotifyBuilder，并在调用 mocks 上的 assertIsSatisfied （）方法前，使用 notifier 知道 Camel 何时进行路由。这可让您不使用固定的 assert 周期来加快测试时间。如果您要断言，其中第 n 个消息到达这个 mock 端点，则还要查看 setAssertPeriod (long)方法了解更多详情。	-1	int
reportGroup (producer)	用于根据大小组打开吞吐量日志记录的数字。		int
resultMinimumWaitTime (producer)	设置 assertIsSatisfied （）的最小预期时间（单位为 millis）将等待 latch，直到它满足为止	0	long

Name	描述	默认值	类型
resultWaitTime (producer)	设置 assertIsSatisfied （）将等待的最大时间量，直到它满足为止	0	long
retainFirst (producer)	指定只保留收到的前 n 个接收交换数。这在测试大数据时，通过不存储此模拟端点接收的每一个交换的副本来减少内存消耗。重要： 使用此限制时，getReceivedCounter （）仍然会返回接收的交换的实际数量。例如，如果我们收到 5000 Exchanges，并且配置为仅保留前 10 个交换，那么 getReceivedCounter （） 仍然会返回 5000，但 getExchanges （）中只有前 10 Exchanges （） 和 getReceivedExchanges （）方法。使用此方法时，不支持其他一些预期的方法，例如 expectedBodiesReceived (Object...)在收到的第一个正文数上设置预期。您可以配置 setRetainFirst (int)和 setRetainLast (int)方法，以限制第一个和最后一个收到。	-1	int
retainLast (producer)	指定只保留最后第 n 个接收交换的数量。这在测试大数据时，通过不存储此模拟端点接收的每一个交换的副本来减少内存消耗。重要： 使用此限制时，getReceivedCounter （）仍然会返回接收的交换的实际数量。例如，如果我们收到 5000 Exchanges，并且配置为仅保留最后 20 个交换，那么 getReceivedCounter （） 仍然会返回 5000，但 getExchanges （） 中只有最后 20 个交换程序）和 getReceivedExchanges （）方法。使用此方法时，不支持其他一些预期的方法，例如 expectedBodiesReceived (Object...)在收到的第一个正文数上设置预期。您可以配置 setRetainFirst (int)和 setRetainLast (int)方法，以限制第一个和最后一个收到。	-1	int
sleepForEmptyTest (producer)	使用零调用 expectedMessageCount (int)时，允许指定 sleep 来检查此端点是否确实为空	0	long
split (producer)	如果启用了从测试端点加载的消息将使用新行分隔符分割，因此每行都是预期的消息。例如，要使用 file 端点来加载一行是预期消息的文件。	false	布尔值
timeout (producer)	轮询 URI 中消息正文时使用的超时	2000	long
copyOnExchange (producer)	设置在这个模拟端点收到传入 Exchange 时是否进行深度副本。默认为 true。	true	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

345.3. EXAMPLE

例如，您可以按如下方式编写测试案例：

```
from("seda:someEndpoint").
  to("test:file://data/expectedOutput?noop=true");
```

如果您的测试随后调用 MockEndpoint.assertIsSatisfied (camelContext)方法，您的测试案例将执行必要的断言。

要了解如何在测试端点上设置其他预期，请查看 Mock 组件。

345.4. 另请参阅

- Spring Testing

第 346 章 THRIFT 组件

从 Camel 版本 2.20 开始提供

Thrift 组件允许您使用 Apache Thrift 二进制通信协议和序列化机制调用或公开远程过程调用(RPC)服务。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-thrift</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

346.1. URI 格式

```
thrift://service[?options]
```

346.2. 端点选项

Thrift 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
useGlobalSslContext 参数 (security)	确定 thrift 组件是否使用全局 SSL 上下文参数	false	布尔值
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Thrift 端点使用 URI 语法进行配置：

```
thrift:host:port/service
```

使用以下路径和查询参数：

346.2.1. 路径参数(3 参数)：

Name	描述	默认值	类型
主机	Thrift 服务器主机名。在使用制作者时，这是 localhost 或 0.0.0.0 （如果没有定义）或远程服务器主机名。		字符串
port	**所需的** Thrift 服务器端口		int
service	thrift 描述符文件（软件包点服务定义名称）中需要完全限定的服务名称。		字符串

346.2.2. 查询参数(12 参数)：

Name	描述	默认值	类型
compressionType (common)	协议压缩机制类型	NONE	ThriftCompressionType
exchangeProtocol (common)	Exchange 协议序列化类型	二进制	ThriftExchangeProtocol
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
clientTimeout (consumer)	用户的客户端超时		int
maxPoolSize (consumer)	Thrift 服务器消费者最大线程池大小	10	int
poolSize (consumer)	Thrift 服务器消费者初始线程池大小	1	int
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern

Name	描述	默认值	类型
method (producer)	Thrift 调用的方法名称		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
negotiationType (security)	安全协商类型	PLAIN TEXT	ThriftNegotiationType
sslParameters (security)	SSL/TLS 安全协商的配置参数		SSLContextParameters

346.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 7 个选项，如下所列。

Name	描述	默认值	类型
camel.component.thrift.enabled	是否启用 thrift 组件的自动配置。这默认是启用的。		布尔值
camel.component.thrift.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.thrift.use-global-ssl-context-parameters	确定 thrift 组件是否使用全局 SSL 上下文参数	false	布尔值
camel.dataformat.thrift.content-type-format	定义内容类型格式，其中的 thrift 消息将从 Java 中序列化/反序列化/反序列化。格式可以是 native 或 json，可以是原生二进制 thrift、json 或 simple json 字段表示。默认值为 binary。	二进制	字符串
camel.dataformat.thrift.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.thrift.enabled	是否启用 thrift 数据格式的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.dataformat.thrift.instance-class	unarmshalling 时使用的类名称		字符串

346.4. THRIFT 方法参数映射

调用过程中的参数必须作为消息正文中的对象列表传递。原语从动态上的对象转换。为了正确找到对应的方法，无论值如何，必须传输所有类型。请参阅以下 exmaple，如何将不同的参数传递给带有 Camel 正文的方法

```
List requestBody = new ArrayList();

requestBody.add((boolean)true);
requestBody.add((byte)THRIFT_TEST_NUM1);
requestBody.add((short)THRIFT_TEST_NUM1);
requestBody.add((int)THRIFT_TEST_NUM1);
requestBody.add((long)THRIFT_TEST_NUM1);
requestBody.add((double)THRIFT_TEST_NUM1);
requestBody.add("empty"); // String parameter
requestBody.add(ByteBuffer.allocate(10)); // binary parameter
requestBody.add(new Work(THRIFT_TEST_NUM1, THRIFT_TEST_NUM2,
Operation.MULTIPLY)); // Struct parameter
requestBody.add(new ArrayList<Integer>()); // list paramater
requestBody.add(new HashSet<String>()); // set parameter
requestBody.add(new HashMap<String, Long>()); // map parameter

Object responseBody = template.requestBody("direct:thrift-alltypes", requestBody);
```

服务使用者中的传入参数也会作为对象列表传递给消息正文。

346.5. THRIFT CONSUMER 标头（将在消费者调用后安装）

标头名称	描述	可能的值
CamelThriftMethodName	由消费者服务处理的方法名称	

346.6. 例子

以下是使用主机和端口参数调用的简单同步方法

```
from("direct:thrift-calculate")
.to("thrift://localhost:1101/org.apache.camel.component.thrift.generated.Calculator?
method=calculate&synchronous=true");
```

以下是对 XML DSL 配置调用的简单同步方法

```
<route>
   <from uri="direct:thrift-add" />
   <to uri="thrift://localhost:1101/org.apache.camel.component.thrift.generated.Calculator?
method=add&synchronous=true"/>
</route>
```

带有异步通信的 Thrift 服务消费者

```
from("thrift://localhost:1101/org.apache.camel.component.thrift.generated.Calculator")
.to("direct:thrift-service");
```

可以使用 thrift-maven-plugin 为 .thrift 文件自动生成 Java 代码生成，但在启动 thrift 编译器二进制分发之前，您的操作系统必须存在于运行的主机上。

346.7. 如需更多信息，请参阅这些资源

Thrift 项目 GitHubhttps://thrift.apache.org/tutorial/java [Apache Thrift Java 教程]

346.8. 另请参阅

- 开始使用
- 配置 Camel
- 组件
- 端点

第 347 章 THRIFT DATAFORMAT

从 Camel 版本 2.20 开始提供

Camel 提供了一个数据格式，可在 Java 和 Apache Thrift 之间序列化。该项目详细介绍了您想 https://thrift.apache.org/ 的原因。Apache Thrift 是语言中立且平台中立，因此您的 Camel 路由生成的消息可能会被其他语言实施使用。

Apache Thrift 实现

347.1. THRIFT 选项

Thrift dataformat 支持 3 个选项，如下所列。

Name	默认值	Java 类型	描述
instanceClass		字符串	unarmshalling 时使用的类名称
contentTypeFormat	二进制	字符串	定义内容类型格式，其中的 thrift 消息将从 Java 中序列化/反序列化/反序列化。格式可以是 native 或 json，可以是原生二进制 thrift、json 或 simple json 字段表示。默认值为 binary。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

347.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 7 个选项，如下所列。

Name	描述	默认值	类型
camel.component.thrift.enabled	是否启用 thrift 组件的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.component.thrift.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.thrift.use-global-ssl-context-parameters	确定 thrift 组件是否使用全局 SSL 上下文参数	false	布尔值
camel.dataformat.thrift.content-type-format	定义内容类型格式，其中的 thrift 消息将从 Java 中序列化/反序列化/反序列化。格式可以是 native 或 json，可以是原生二进制 thrift、json 或 simple json 字段表示。默认值为 binary。	二进制	字符串
camel.dataformat.thrift.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.thrift.enabled	是否启用 thrift 数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.thrift.instance-class	unarmshalling 时使用的类名称		字符串

ND

347.3. 内容类型格式

可以解析 JSON 消息，将其转换为 Thrift 格式，并使用原生 util 转换程序将其重新解析。要使用这个选项，请将 contentTypeFormat 值设置为 'json'，或使用第二个参数调用 thrift。如果没有指定默认实例，请始终使用原生二进制 Thrift 格式。简单的 JSON 格式是仅写的(marshal)，并生成适合通过脚本语言解析的简单输出格式。示例代码显示如下：

```
from("direct:marshal")
    .unmarshal()
    .thrift("org.apache.camel.dataformat.thrift.generated.Work", "json")
    .to("mock:reverse");
```

347.4. THRIFT 概述

这一快速概述如何使用 Thrift。详情请查看 完整的教程

347.5. 定义 THRIFT 格式

第一步是定义交换正文的格式。这在 .thrift 文件中定义，如下所示：

tutorial.thrift

```
namespace java org.apache.camel.dataformat.thrift.generated

enum Operation {
  ADD = 1,
  SUBTRACT = 2,
  MULTIPLY = 3,
  DIVIDE = 4
}

struct Work {
  1: i32 num1 = 0,
  2: i32 num2,
  3: Operation op,
  4: optional string comment,
}
```

347.6. 生成 JAVA 类

Apache Thrift 提供编译器，它将为我们在 .thrift 文件中定义的格式生成 Java 类。

您还可以为手动所需的任何其他支持语言运行编译器。

```
thrift -r --gen java -out ../java/ ./tutorial-dataformat.thrift
```

这将为 .thrift 文件中定义的每种类型生成单独的 Java 类，即 struct 或 enum。生成的类实施 org.apache.thrift.TBase，这是序列化机制所需的。因此，您的交换正文中只使用这些类非常重要。如果您试图使用没有实现 org.apache.thrift.TBase 的类，Camel 会在路由创建时抛出异常。

347.7. JAVA DSL

您可以使用创建 ThriftDataFormat 实例，并将其传递给 Camel DataFormat marshal 和 unmarshal

API，如下所示：

```
ThriftDataFormat format = new ThriftDataFormat(new Work());

from("direct:in").marshal(format);
from("direct:back").unmarshal(format).to("mock:reverse");
```

或者，使用 DSL thrift （）传递 unmarshal 默认实例或默认实例类名称，如下所示。

```
// You don't need to specify the default instance for the thrift marshaling
from("direct:marshal").marshal().thrift();
from("direct:unmarshalA").unmarshal()
    .thrift("org.apache.camel.dataformat.thrift.generated.Work")
    .to("mock:reverse");

from("direct:unmarshalB").unmarshal().thrift(new Work()).to("mock:reverse");
```

347.8. SPRING DSL

以下示例演示了如何使用 Thrift 来使用 Spring 配置 thrift 数据类型

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="direct:start"/>
    <unmarshal>
      <thrift instanceClass="org.apache.camel.dataformat.thrift.generated.Work" />
    </unmarshal>
    <to uri="mock:result"/>
  </route>
</camelContext>
```

347.9. 依赖项

要在 camel 路由中使用 Thrift，您需要添加对实现此数据格式的 camel-thrift 的依赖。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-thrift</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

第 348 章 TIDYMARKUP DATAFORMAT

从 Camel 版本 2.0 开始提供

TidyMarkup 是一个数据格式，它使用 TagSoup 解放 HTML。它可用于解析大量 HTML 并返回其格式良好的 HTML。

Camel eats our own -dog food- soap

我们的 pdf Manual 中有一些问题，其中我们有一些奇形符号。因此，Jo nathan 使用这个数据格式来解放 wiki html 页面，用作渲染 pdf 手册的基础。然后是 mysterious 符号。

TidyMarkup 只支持 unmarshal 操作，因为我们不希望将 HTML 正常转换为 HTML。

348.1. TIDYMARKUP 选项

TidyMarkup dataformat 支持 3 个选项，如下所列。

Name	默认值	Java 类型	描述
dataObjectType	org.w3c.dom.Node	字符串	unmarshal 的数据类型是 org.w3c.dom.Node 或 java.lang.String。默认是 org.w3c.dom.Node
omitXmlDeclaration	false	布尔值	返回 String 时，请忽略顶部的 XML 声明。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

348.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.tidymarkup.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.tidymarkup.data-object-type	unmarshal 的数据类型是 org.w3c.dom.Node 或 java.lang.String。默认是 org.w3c.dom.Node	org.w3c.dom.Node	字符串
camel.dataformat.tidymarkup.enabled	启用 tidymarkup dataformat	true	布尔值
camel.dataformat.tidymarkup.omit-xml-declaration	返回 String 时，请忽略顶部的 XML 声明。	false	布尔值

348.3. JAVA DSL 示例

消费者提供一些 HTML 的示例

```
from("file://site/inbox").unmarshal().tidyMarkup().to("file://site/blogs");
```

348.4. SPRING XML 示例

以下示例演示了如何使用 Spring 将 TidyMarkup 用于 unmarshal

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="file://site/inbox"/>
    <unmarshal>
      <tidyMarkup/>
    </unmarshal>
    <to uri="file://site/blogs"/>
  </route>
</camelContext>
```

348.5. 依赖项

要在 camel 路由中使用 TidyMarkup，您需要添加对实现此数据格式的 camel-tagsoup 的依赖。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-tagsoup</artifactId>
  <version>x.x.x</version>
</dependency>
```

第 349 章 TIKA 组件

从 Camel 版本 2.19 开始提供

Tika: 组件提供使用 Apache Tika 检测和解析文档的功能。此组件使用 Apache Tika 作为底层库来使用文档。

要使用 Tika 组件， Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-tika</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

TIKA 组件仅支持生成者端点。

349.1. 选项

Tika 组件没有选项。

Tika 端点使用 URI 语法进行配置：

```
tika:operation
```

使用以下路径和查询参数：

349.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
operation	所需的 Tika Operation. 解析或检测		TikaOperation

349.1.2. 查询参数(5 参数)：

Name	描述	默认值	类型
tikaConfig (producer)	Tika 配置		TikaConfig
tikaConfigUri (producer)	Tika Config Uri ：tika-config.xml 的 URI		字符串
tikaParseOutputEncoding (producer)	Tika Parse Output Encoding - 用来指定解析输出的字符编码。默认为 Charset.defaultCharset （）。		字符串
tikaParseOutputFormat (producer)	Tika 输出格式.支持的输出格式. xml: 返回 Parsed Content as XML. html: returneds Parsed Content as HTML. text: returneds Parsed Content as Text. textMain ：使用 boilerpipe 库从网页自动提取主内容。	xml	TikaParseOutputFormat
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

349.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.tika.enabled	启用 tika 组件	true	布尔值
camel.component.tika.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

349.3. 要检测文件的 MIME 类型

该文件应当放在 Body 中。

```
from("direct:start")
        .to("tika:detect");
```

349.4. 解析文件

该文件应当放在 Body 中。

```
from("direct:start")
        .to("tika:parse");
```

第 350 章 计时器组件

从 Camel 版本 1.0 开始提供

timer: 组件用于在计时器触发时生成消息交换，您只能消耗来自此端点的事件。

350.1. URI 格式

```
timer:name[?options]
```

其中 name 是 Timer 对象的名称，它在端点之间创建和共享。因此，如果您为所有计时器端点使用相同的名称，则只使用一个 Timer 对象和线程。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

注：所生成的交换的 IN 正文为 null。因此，exchange.getIn （）.getBody （） 返回 null。

TIP:*Advanced Scheduler* 另请参阅支持更多高级调度的 Quartz 组件。

TIP:*Specify time in humanfriendly format* In Camel 2.3 之后，您可以 以用户友好的语法 指定时间。

350.2. 选项

Timer 组件没有选项。

Timer 端点使用 URI 语法进行配置：

```
timer:timerName
```

使用以下路径和查询参数：

350.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
timerName	**必需** 计时器的名称		字符串

350.2.2. 查询参数(12 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外，该处理程序记录在 WARN/ERROR 级别并忽略。	false	布尔值
delay (consumer)	生成第一个事件前等待的毫秒数。不应与 time 选项结合使用。默认值为 1000。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
fixedRate (consumer)	事件大约会定期发生，由指定的周期分开。	false	布尔值
period (consumer)	如果大于 0，则每周期毫秒生成定期事件。默认值为 1000。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
repeatCount (consumer)	指定触发的最大数量。因此，如果您将其设置为 1，则计时器将仅触发一次。如果您将其设置为 5，它将只触发五次。值为零或负数表示会永久触发。	0	long
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，该例外记录在 WARN/ERROR 级别并忽略。		ExceptionHandler
exchangePattern (consumer)	在创建交换时设置默认交换模式。		ExchangePattern
守护进程 （高级）	指定与计时器端点关联的线程是否作为守护进程运行。默认值为 true。	true	布尔值
pattern (advanced)	允许您指定用于使用 URI 语法设置时间选项的自定义日期模式。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

Name	描述	默认值	类型
时间 (advanced)	java.util.Date，应生成第一个事件。如果使用 URI，则预期的模式为：yyyy-MM-dd HH:mm:ss 或 yyyy-MM-dd'T'HH:mm:s。		Date
timer (advanced)	使用自定义计时器		计时器

350.3. 交换属性

当触发计时器时，它会将以下信息作为属性添加到 交换 中：

Name	类型	描述
Exchange.TIMER_NAME	**字符串**	**name** 选项的值。
Exchange.TIMER_TIME	**Date**	**time** 选项的值。
Exchange.TIMER_PERIOD	**long**	**period** 选项的值。
Exchange.TIMER_FIRED_TIME	**Date**	消费者触发的时间。
Exchange.TIMER_COUNTER	**Long**	Camel 2.8： 当前触发计数器。从 1 开始。

350.4. 示例

要设置一个路由，该路由每 60 秒生成事件：

-

```
from("timer://foo?fixedRate=true&period=60000").to("bean:myBean?
method=someMethodName");
```

提示

您可以使用 period=60s 而不是 60000s，这更易于阅读。

以上路由将生成一个事件，然后在 Registry （如 JNDI 或 Spring）中调用名为 myBean 的 bean 上的 someMethodName 方法。

以及 Spring DSL 中的路由：

```
<route>
  <from uri="timer://foo?fixedRate=true&amp;period=60000"/>
  <to uri="bean:myBean?method=someMethodName"/>
</route>
```

350.5. 尽快触发

从 Camel 2.17 开始提供

您可能需要尽快触发 Camel 路由中的信息，您可以使用负的延迟：

```
<route>
  <from uri="timer://foo?delay=-1"/>
  <to uri="bean:myBean?method=someMethodName"/>
</route>
```

这样，计时器将立即触发消息。

您还可以同时指定 repeatCount 参数，以及负延迟，以在达到固定数量后停止触发的消息。

如果您没有指定 repeatCount，则计时器将继续触发消息，直到路由将停止为止。

350.6. 只触发一次

从 Camel 2.8 开始提供

您可能希望仅在 Camel 路由中触发一次消息，比如在启动路由时。要使用 repeatCount 选项，如下所示：

```
<route>
  <from uri="timer://foo?repeatCount=1"/>
  <to uri="bean:myBean?method=someMethodName"/>
</route>
```

350.7. 另请参阅

- scheduler
- quartz

第 351 章 TWILIO 组件

从 Camel 版本 2.20 开始提供

Twilio 组件提供对使用 Twilio Java SDK 访问的 Twilio REST API 版本 2010-04-01 的访问权限。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-twilio</artifactId>
  <version>${camel-version}</version>
</dependency>
```

351.1. TWILIO 选项

Twilio 组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	使用共享配置		TwilioConfiguration
RESTClient (advanced)	使用共享 REST 客户端		TwilioRestClient
用户名 (security)	要使用的帐户。		字符串
密码 (security)	帐户的身份验证令牌。		字符串
accountSid (security)	要使用的帐户 SID。		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Twilio 端点使用 URI 语法进行配置：

```
twilio:apiName/methodName
```

使用以下路径和查询参数：

351.1.1. 路径参数(2 参数)：

Name	描述	默认值	类型
apiName	**需要** 执行什么操作		TwilioApiName
methodName	**必需的** 所选操作使用哪些子操作		字符串

351.1.2. 查询参数(5 参数)：

Name	描述	默认值	类型
inBody (common)	设置要在交换 In Body 中传递的参数名称		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

351.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 8 个选项，如下所列。

Name	描述	默认值	类型
camel.component.twilio.account-sid	要使用的帐户 SID。		字符串

Name	描述	默认值	类型
camel.component.twilio.configuration.api-name	要执行的操作类型		TwilioApiName
camel.component.twilio.configuration.method-name	用于所选操作的子操作		字符串
camel.component.twilio.enabled	启用 twilio 组件	true	布尔值
camel.component.twilio.password	帐户的身份验证令牌。		字符串
camel.component.twilio.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.twilio.rest-client	使用共享 REST 客户端。选项是一个 com.twilio.http.TwilioRestClient 类型。		字符串
camel.component.twilio.username	要使用的帐户。		字符串

351.3. URI 格式

> ***twilio://endpoint-prefix/endpoint?[options]***

端点前缀可以是以下之一：

- ***account***
- ***address***
- ***address-dependent-phone-number***
- ***application***

- *available-phone-number-country*
- *available-phone-number-country-local*
- *available-phone-number-country-mobile*
- *available-phone-number-country-toll-free*
- *调用*
- *call-feedback*
- *call-feedback-summary*
- *call-notification*
- *call-recording*
- *会议*
- *conference-participant*
- *connect-app*
- *incoming-phone-number*
- *incoming-phone-number-local*

- ***incoming-phone-number-mobile***
- ***incoming-phone-number-toll-free***
- ***key***
- ***message***
- ***message-feedback***
- ***message-media***
- ***new-key***
- ***new-signing-key***
- ***通知***
- ***outgoing-caller-id***
- ***queue***
- ***queue-member***
- *记录*
- ***recording-add-on-result***

- *recording-add-on-result-payload*
- *recording-transcription*
- *short-code*
- *signing-key*
- *sip-credential-list*
- *sip-credential-list-credential*
- *sip-domain*
- *sip-domain-credential-list-mapping*
- *sip-domain-ip-access-control-list-mapping*
- *sip-ip-access-control-list*
- *sip-ip-access-control-list-ip-address*
- *token*
- *transcription*
- *usage-record*

- *usage-record-all-time*
- *usage-record-daily*
- *usage-record-last-month*
- *usage-record-monthly*
- *usage-record-this-month*
- *usage-record-today*
- *usage-record-yearly*
- *usage-record-yesterday*
- *usage-trigger*
- *validation-request*

351.4. 生产者端点：

生产者端点可以使用端点前缀，后跟端点名称和关联的选项。简写别名可用于所有端点。端点 URI MUST 包含前缀。

任何端点选项都可以在端点 URI 中提供，或者在消息标头中动态提供。消息标头名称的格式必须是 CamelTwilio.<option>。请注意，inBody 选项会覆盖消息标头，即 Body=option 中的端点 选项会覆盖 CamelTwilio.option 标头。

端点可以是以下之一：

端点	简写别名	描述
创建者	create	向 Twilio API 发出请求以执行创建
deleter	delete	向 Twilio API 发出请求以执行删除
fetcher	fetch	向 Twilio API 发出请求来执行获取
读取器	读取	向 Twilio API 发出请求来执行读取
Updater	update	向 Twilio API 发出请求以执行更新

可用的端点根据端点前缀而有所不同。

有关端点和选项的更多信息，请参阅 API 文档，网址为：
https://www.twilio.com/docs/libraries/reference/twilio-java/index.html

351.5. 消费者端点：

任何制作者端点都可以用作消费者端点。消费者端点可以使用 Scheduled Poll Consumer Options 和 consumer. 前缀来调度端点调用。返回数组或集合的消费者端点将为每个元素生成一个交换，它们的路由将为每个交换执行一次。

如果要从 Twilio 接收调用或消息并使用 Camel 使用者端点响应它们，您可以使用其他基于 HTTP 的组件，如 camel-servlet、camel-undertow、camel-jetty 和 camel-netty-http 来响应 TwiML。

351.6. 消息标头

任何选项都可以在带有 CamelTwilio. 前缀的制作者端点的消息标头中提供。

351.7. 消息正文

所有结果消息正文都使用由 Twilio Java SDK 提供的对象。生产者端点可以在 inBody 端点参数中指定传入消息正文的选项名称。

第 352 章 MEWITTER 组件

从 Camel 版本 2.10 开始提供

camel-twitter 由 4 个组件组成：

- Wwitw Direct Message
- author Search
- someonewitter Streaming
- author Timeline

Twitter 组件通过封装了 author 4J，启用 OBC API 的最有用的功能。它允许直接、轮询或事件驱动的时间表、用户、趋势和直接消息消耗。另外，它支持在状态更新或直接消息中生成消息。

author 现在需要使用 OAuth 进行所有客户端应用程序身份验证。要将 camel-twitter 与您的帐户搭配使用，您需要在为 https://dev.twitter.com/apps/new 的 author 中创建一个新应用程序，并授予应用程序对您的帐户的访问权限。最后，生成您的访问令牌和 secret。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-twitter</artifactId>
  <version>${camel-version}</version>
</dependency>
```

352.1. 消费者端点

camel-twitter 而不是端点通过一个路由交换返回列表，而是为每个返回的对象创建一个路由交换。例如，如果"timeline/home"结果为五个状态，则路由将执行五次（每个 Status 的一个）。

端点	Context	正文类型	备注
twitter-directmessage	直接、轮询	twitter4j.DirectMessage	
twitter-search	直接、轮询	twitter4j.Status	
author-streaming	事件、轮询	twitter4j.Status	
author-timeline	直接、轮询	twitter4j.Status	

352.2. 制作者端点

端点	正文类型	备注
twitter-directmessage	字符串	
twitter-search	List<twitter4j.Status>	
author-timeline	字符串	producer 仅支持 'user' timelineType

352.3. 消息标头

Name	描述
CamelTwitterKeywords	搜索制作者使用此标头动态更改搜索关键字。
CamelTwitterSearchLanguage	Camel 2.11.0： 此标头可以覆盖 **lang** 选项，该选项动态设置搜索语言

Name	描述
CamelTwitterCount	Camel 2.11.0 此标头可覆盖 **count** 选项，它会设置将返回的最大 12itters。
CamelTwitterNumberOfPages	Camel 2.11.0 此标头可覆盖 **numberOfPages** 选项，它设定了我们想要 Titter 返回的页面数量。

352.4. 消息正文

所有消息正文都利用了 VMWare4J API 提供的对象。

352.5. 使用案例

注意

API 速率限制： Twitter REST API 封装有 OBC4J 封装的 API 速率限制。您可以在 API Rate Limits 文档中找到每个方法限制。请注意，页面中没有列出的端点/资源默认为每个窗口分配的用户 15 个请求。

352.5.1. 要在 OBC 配置集中创建状态更新，请发送此制作者一个 String 正文：

```
from("direct:foo")
  .to("twitter-timeline://user?consumerKey=[s]&consumerSecret=[s]&accessToken=
[s]&accessTokenSecret=[s]);
```

352.5.2. 要轮询，每 60 sec.，您的主页时间表上的所有状态：

```
from("twitter-timeline://home?type=polling&delay=60&consumerKey=[s]&consumerSecret=
[s]&accessToken=[s]&accessTokenSecret=[s]")
  .to("bean:blah");
```

352.5.3. 仅搜索关键字 'camel' 的所有状态：

```
from("twitter-search://foo?type=polling&keywords=camel&consumerKey=
[s]&consumerSecret=[s]&accessToken=[s]&accessTokenSecret=[s]")
  .to("bean:blah");
```

352.5.4. 使用带有静态关键字的制作者搜索：

```
from("direct:foo")
  .to("twitter-search://foo?keywords=camel&consumerKey=[s]&consumerSecret=
[s]&accessToken=[s]&accessTokenSecret=[s]");
```

352.5.5. 使用带有标头的动态关键字的制作者进行搜索：

在 bar 标头中，我们有要搜索的关键字，因此我们可以将这个值分配给 CamelTwitterKeywords 标头：

```
from("direct:foo")
  .setHeader("CamelTwitterKeywords", header("bar"))
  .to("twitter-search://foo?consumerKey=[s]&consumerSecret=[s]&accessToken=
[s]&accessTokenSecret=[s]");
```

352.6. EXAMPLE

另请参阅 zSystems Websocket 示例。

352.7. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用
- OBC Websocket 示例

第 353 章 AUTHOR DIRECT MESSAGE COMPONENT

从 Camel 版本 2.10 开始提供

Twitter Direct Message 组件消耗/生成用户直接消息。

353.1. 组件选项

Twitter Direct Message 组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
accessToken (security)	访问令牌		字符串
accessTokenSecret (security)	访问令牌 secret		字符串
consumerKey (security)	消费者密钥		字符串
consumerSecret (security)	消费者 secret		字符串
httpProxyHost (proxy)	http 代理主机，可用于 camel-twitter。		字符串
httpProxyUser (proxy)	http 代理用户，可用于 camel-twitter。		字符串
httpProxyPassword (proxy)	http 代理密码，可用于 camel-twitter。		字符串
httpProxyPort (proxy)	可用于 camel-twitter 的 http 代理端口。		int
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

353.2. 端点选项

Twitter Direct Message 端点使用 URI 语法进行配置：

```
twitter-directmessage:user
```

使用以下路径和查询参数：

353.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
user	**必需** The user name to send a direct message。对于消费者，这将会被忽略。		字符串

353.2.2. 查询参数(42 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
type (consumer)	要使用的端点类型。只有流支持事件类型。	轮询	EndpointType
distanceMetric (consumer)	非流域搜索使用，使用配置的指标按 radius 搜索。单位可以是 mi 代表 miles，也可以是 kilometers 的 km。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。	km	字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern

Name	描述	默认值	类型
extendedMode (consumer)	用于启用来自 twitter 的完整文本（例如，接收包含超过 140 个字符的 12 倍）。	true	布尔值
latitude (consumer)	非流地搜索用于按 latitude 搜索。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。		å�Œ
locations (consumer)	绑定框，由 lat/lons 对创建。可用于 streaming/filter。对定义为 lat,lon。和多个 paris 可以通过分号隔开。		字符串
longitude (consumer)	非流地搜索用于按长期搜索进行搜索。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。		å�Œ
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
RADIUS (consumer)	非流地搜索用于按 radius 搜索。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。		å�Œ
twitterStream (consumer)	使用自定义实例 OBCStream		TwitterStream
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
count (filter)	每个页面限制结果数。	5	整数
filterOld (filter)	过滤掉之前轮询的旧 12eets。此状态仅存储在内存中，并且基于最后的 12 个 ID。	true	布尔值
lang (filter)	用于搜索的 lang 字符串 ISO_639-1		字符串
numberOfPages (filter)	您希望 camel-twitter 使用的页结果数。	1	整数
sinceId (filter)	最后的 Teet id，将用于拉取 Teets。当长时间运行后，camel 路由重启时会很有用。	1	long
userIds (filter)	根据用户 ID 过滤的 streaming/filter：可以使用逗号分隔多个值。		字符串
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int

Name	描述	默认值	类型
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。	30000	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
sortById (sort)	按 id 排序，因此最旧的是第一个，最后是最新的。	true	布尔值

Name	描述	默认值	类型
httpProxyHost (proxy)	http 代理主机，可用于 camel-twitter。也可以在 OBCComponent 级别上配置。		字符串
httpProxyPassword (proxy)	http 代理密码，可用于 camel-twitter。也可以在 OBCComponent 级别上配置。		字符串
httpProxyPort (proxy)	可用于 camel-twitter 的 http 代理端口。也可以在 OBCComponent 级别上配置。		整数
httpProxyUser (proxy)	http 代理用户，可用于 camel-twitter。也可以在 OBCComponent 级别上配置。		字符串
accessToken (security)	访问令牌。也可以在 OBCComponent 级别上配置。		字符串
accessTokenSecret (security)	access secret。也可以在 OBCComponent 级别上配置。		字符串
consumerKey (security)	consumer 密钥。也可以在 OBCComponent 级别上配置。		字符串
consumerSecret (security)	consumer 机密。也可以在 OBCComponent 级别上配置。		字符串

353.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
camel.component.twitter-directmessage.access-token	访问令牌		字符串
camel.component.twitter-directmessage.access-token-secret	访问令牌 secret		字符串

Name	描述	默认值	类型
camel.component.twitter-directmessage.consumer-key	消费者密钥		字符串
camel.component.twitter-directmessage.consumer-secret	消费者 secret		字符串
camel.component.twitter-directmessage.enabled	是否启用 twitter-directmessage 组件的自动配置。这默认是启用的。		布尔值
camel.component.twitter-directmessage.http-proxy-host	http 代理主机，可用于 camel-twitter。		字符串
camel.component.twitter-directmessage.http-proxy-password	http 代理密码，可用于 camel-twitter。		字符串
camel.component.twitter-directmessage.http-proxy-port	可用于 camel-twitter 的 http 代理端口。		整数
camel.component.twitter-directmessage.http-proxy-user	http 代理用户，可用于 camel-twitter。		字符串
camel.component.twitter-directmessage.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 354 章 AUTHOR SEARCH COMPONENT

从 Camel 版本 2.10 开始提供

author Search 组件会消耗搜索结果。

354.1. 组件选项

OBC Search 组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
accessToken (security)	访问令牌		字符串
accessTokenSecret (security)	访问令牌 secret		字符串
consumerKey (security)	消费者密钥		字符串
consumerSecret (security)	消费者 secret		字符串
httpProxyHost (proxy)	http 代理主机，可用于 camel-twitter。		字符串
httpProxyUser (proxy)	http 代理用户，可用于 camel-twitter。		字符串
httpProxyPassword (proxy)	http 代理密码，可用于 camel-twitter。		字符串
httpProxyPort (proxy)	可用于 camel-twitter 的 http 代理端口。		int
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

354.2. 端点选项

author Search 端点使用 URI 语法进行配置：

```
twitter-search:keywords
```

使用以下路径和查询参数：

354.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
关键字	**必需的** 搜索关键字。可以使用逗号分隔多个值。		字符串

354.2.2. 查询参数(42 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
type (consumer)	要使用的端点类型。只有流支持事件类型。	轮询	EndpointType
distanceMetric (consumer)	非流域搜索使用，使用配置的指标按 radius 搜索。单位可以是 mi 代表 miles，也可以是 kilometers 的 km。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。	km	字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
extendedMode (consumer)	用于启用来自 twitter 的完整文本（例如，接收包含超过 140 个字符的 12 倍）。	true	布尔值

Name	描述	默认值	类型
latitude (consumer)	非流地搜索用于按 latitude 搜索。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。		å�Œ
locations (consumer)	绑定框，由 lat/lons 对创建。可用于 streaming/filter。对定义为 lat,lon。和多个 paris 可以通过分号隔开。		字符串
longitude (consumer)	非流地搜索用于按长期搜索进行搜索。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。		å�Œ
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
RADIUS (consumer)	非流地搜索用于按 radius 搜索。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。		å�Œ
twitterStream (consumer)	使用自定义实例 OBCStream		TwitterStream
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
count (filter)	每个页面限制结果数。	5	整数
filterOld (filter)	过滤掉之前轮询的旧 12eets。此状态仅存储在内存中，并且基于最后的 12 个 ID。	true	布尔值
lang (filter)	用于搜索的 lang 字符串 ISO_639-1		字符串
numberOfPages (filter)	您希望 camel-twitter 使用的页结果数。	1	整数
sinceId (filter)	最后的 Teet id，将用于拉取 Teets。当长时间运行后，camel 路由重启时会很有用。	1	long
userIds (filter)	根据用户 ID 过滤的 streaming/filter：可以使用逗号分隔多个值。		字符串
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int

Name	描述	默认值	类型
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。	30000	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
sortById (sort)	按 id 排序，因此最旧的是第一个，最后是最新的。	true	布尔值
httpProxyHost (proxy)	http 代理主机，可用于 camel-twitter。也可以在 OBCComponent 级别上配置。		字符串

Name	描述	默认值	类型
httpProxyPassword (proxy)	http 代理密码，可用于 camel-twitter。也可以在 OBCComponent 级别上配置。		字符串
httpProxyPort (proxy)	可用于 camel-twitter 的 http 代理端口。也可以在 OBCComponent 级别上配置。		整数
httpProxyUser (proxy)	http 代理用户，可用于 camel-twitter。也可以在 OBCComponent 级别上配置。		字符串
accessToken (security)	访问令牌。也可以在 OBCComponent 级别上配置。		字符串
accessTokenSecret (security)	access secret。也可以在 OBCComponent 级别上配置。		字符串
consumerKey (security)	consumer 密钥。也可以在 OBCComponent 级别上配置。		字符串
consumerSecret (security)	consumer 机密。也可以在 OBCComponent 级别上配置。		字符串

354.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
camel.component.twitter-search.access-token	访问令牌		字符串
camel.component.twitter-search.access-token-secret	访问令牌 secret		字符串
camel.component.twitter-search.consumer-key	消费者密钥		字符串

Name	描述	默认值	类型
camel.component.twitter-search.consumer-secret	消费者 secret		字符串
camel.component.twitter-search.enabled	是否启用 twitter-search 组件的自动配置。这默认是启用的。		布尔值
camel.component.twitter-search.http-proxy-host	http 代理主机，可用于 camel-twitter。		字符串
camel.component.twitter-search.http-proxy-password	http 代理密码，可用于 camel-twitter。		字符串
camel.component.twitter-search.http-proxy-port	可用于 camel-twitter 的 http 代理端口。		整数
camel.component.twitter-search.http-proxy-user	http 代理用户，可用于 camel-twitter。		字符串
camel.component.twitter-search.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 355 章 AUTHOR STREAMING 组件

从 Camel 版本 2.10 开始提供

使用 Streaming API， Twitter Streaming 组件会消耗 twitter 状态。

355.1. 组件选项

OBC Streaming 组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
accessToken (security)	访问令牌		字符串
accessTokenSecret (security)	访问令牌 secret		字符串
consumerKey (security)	消费者密钥		字符串
consumerSecret (security)	消费者 secret		字符串
httpProxyHost (proxy)	http 代理主机，可用于 camel-twitter。		字符串
httpProxyUser (proxy)	http 代理用户，可用于 camel-twitter。		字符串
httpProxyPassword (proxy)	http 代理密码，可用于 camel-twitter。		字符串
httpProxyPort (proxy)	可用于 camel-twitter 的 http 代理端口。		int
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

355.2. 端点选项

author Streaming 端点使用 URI 语法进行配置：

```
twitter-streaming:streamingType
```

使用以下路径和查询参数：

355.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
streamingType	**需要** 要使用的流类型。		StreamingType

355.2.2. 查询参数(43 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
type (consumer)	要使用的端点类型。只有流支持事件类型。	轮询	EndpointType
distanceMetric (consumer)	非流域搜索使用，使用配置的指标按 radius 搜索。单位可以是 mi 代表 miles，也可以是 kilometers 的 km。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。	km	字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
extendedMode (consumer)	用于启用来自 twitter 的完整文本（例如，接收包含超过 140 个字符的 12 倍）。	true	布尔值

Name	描述	默认值	类型
latitude (consumer)	非流地搜索用于按 latitude 搜索。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。		å�Œ
locations (consumer)	绑定框，由 lat/lons 对创建。可用于 streaming/filter。对定义为 lat,lon。和多个 paris 可以通过分号隔开。		字符串
longitude (consumer)	非流地搜索用于按长期搜索进行搜索。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。		å�Œ
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
RADIUS (consumer)	非流地搜索用于按 radius 搜索。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。		å�Œ
twitterStream (consumer)	使用自定义实例 OBCStream		TwitterStream
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
count (filter)	每个页面限制结果数。	5	整数
filterOld (filter)	过滤掉之前轮询的旧 12eets。此状态仅存储在内存中，并且基于最后的 12 个 ID。	true	布尔值
关键字 (filter)	可用于流过滤器。可以使用逗号分隔多个值。		字符串
lang (filter)	用于搜索的 lang 字符串 ISO_639-1		字符串
numberOfPages (filter)	您希望 camel-twitter 使用的页结果数。	1	整数
sinceId (filter)	最后的 Teet id，将用于拉取 Teets。当长时间运行后，camel 路由重启时会很有用。	1	long
userIds (filter)	根据用户 ID 过滤的 streaming/filter：可以使用逗号分隔多个值。		字符串
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int

Name	描述	默认值	类型
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。	30000	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
sortById (sort)	按 id 排序，因此最旧的是第一个，最后是最新的。	true	布尔值
httpProxyHost (proxy)	http 代理主机，可用于 camel-twitter。也可以在 OBCComponent 级别上配置。		字符串

Name	描述	默认值	类型
httpProxyPassword (proxy)	http 代理密码，可用于 camel-twitter。也可以在 OBCComponent 级别上配置。		字符串
httpProxyPort (proxy)	可用于 camel-twitter 的 http 代理端口。也可以在 OBCComponent 级别上配置。		整数
httpProxyUser (proxy)	http 代理用户，可用于 camel-twitter。也可以在 OBCComponent 级别上配置。		字符串
accessToken (security)	访问令牌。也可以在 OBCComponent 级别上配置。		字符串
accessTokenSecret (security)	access secret。也可以在 OBCComponent 级别上配置。		字符串
consumerKey (security)	consumer 密钥。也可以在 OBCComponent 级别上配置。		字符串
consumerSecret (security)	consumer 机密。也可以在 OBCComponent 级别上配置。		字符串

355.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
camel.component.twitter-streaming.access-token	访问令牌		字符串
camel.component.twitter-streaming.access-token-secret	访问令牌 secret		字符串
camel.component.twitter-streaming.consumer-key	消费者密钥		字符串

Name	描述	默认值	类型
camel.component.twitter-streaming.consumer-secret	消费者 secret		字符串
camel.component.twitter-streaming.enabled	是否启用 twitter-streaming 组件的自动配置。这默认是启用的。		布尔值
camel.component.twitter-streaming.http-proxy-host	http 代理主机，可用于 camel-twitter。		字符串
camel.component.twitter-streaming.http-proxy-password	http 代理密码，可用于 camel-twitter。		字符串
camel.component.twitter-streaming.http-proxy-port	可用于 camel-twitter 的 http 代理端口。		整数
camel.component.twitter-streaming.http-proxy-user	http 代理用户，可用于 camel-twitter。		字符串
camel.component.twitter-streaming.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 356 章 AUTHOR TIMELINE COMPONENT

从 Camel 版本 2.10 开始提供

author Timeline 组件会消耗 twitter 时间表或更新特定用户的状态。

356.1. 组件选项

Twitter Timeline 组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
accessToken (security)	访问令牌		字符串
accessTokenSecret (security)	访问令牌 secret		字符串
consumerKey (security)	消费者密钥		字符串
consumerSecret (security)	消费者 secret		字符串
httpProxyHost (proxy)	http 代理主机，可用于 camel-twitter。		字符串
httpProxyUser (proxy)	http 代理用户，可用于 camel-twitter。		字符串
httpProxyPassword (proxy)	http 代理密码，可用于 camel-twitter。		字符串
httpProxyPort (proxy)	可用于 camel-twitter 的 http 代理端口。		int
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

356.2. 端点选项

author Timeline 端点使用 URI 语法进行配置：

```
twitter-timeline:timelineType
```

使用以下路径和查询参数：

356.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
timelineType	**需要** 时间表类型才能生成/消耗。		TimelineType

356.2.2. 查询参数(43 参数)：

Name	描述	默认值	类型
user (common)	使用 timelineType=user 的用户名		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
type (consumer)	要使用的端点类型。只有流支持事件类型。	轮询	EndpointType
distanceMetric (consumer)	非流域搜索使用，使用配置的指标按 radius 搜索。单位可以是 mi 代表 miles，也可以是 kilometers 的 km。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。	km	字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern

Name	描述	默认值	类型
extendedMode (consumer)	用于启用来自 twitter 的完整文本（例如，接收包含超过 140 个字符的 12 倍）。	true	布尔值
latitude (consumer)	非流地搜索用于按 latitude 搜索。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。		å�Œ
locations (consumer)	绑定框，由 lat/lons 对创建。可用于 streaming/filter。对定义为 lat,lon。和多个 paris 可以通过分号隔开。		字符串
longitude (consumer)	非流地搜索用于按长期搜索进行搜索。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。		å�Œ
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
RADIUS (consumer)	非流地搜索用于按 radius 搜索。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。		å�Œ
twitterStream (consumer)	使用自定义实例 OBCStream		TwitterStream
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
count (filter)	每个页面限制结果数。	5	整数
filterOld (filter)	过滤掉之前轮询的旧 12eets。此状态仅存储在内存中，并且基于最后的 12 个 ID。	true	布尔值
lang (filter)	用于搜索的 lang 字符串 ISO_639-1		字符串
numberOfPages (filter)	您希望 camel-twitter 使用的页结果数。	1	整数
sinceId (filter)	最后的 Teet id，将用于拉取 Teets。当长时间运行后，camel 路由重启时会很有用。	1	long
userIds (filter)	根据用户 ID 过滤的 streaming/filter：可以使用逗号分隔多个值。		字符串

Name	描述	默认值	类型
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。	30000	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
sortById (sort)	按 id 排序，因此最旧的是第一个，最后是最新的。	true	布尔值

Name	描述	默认值	类型
httpProxyHost (proxy)	http 代理主机，可用于 camel-twitter。也可以在 OBCComponent 级别上配置。		字符串
httpProxyPassword (proxy)	http 代理密码，可用于 camel-twitter。也可以在 OBCComponent 级别上配置。		字符串
httpProxyPort (proxy)	可用于 camel-twitter 的 http 代理端口。也可以在 OBCComponent 级别上配置。		整数
httpProxyUser (proxy)	http 代理用户，可用于 camel-twitter。也可以在 OBCComponent 级别上配置。		字符串
accessToken (security)	访问令牌。也可以在 OBCComponent 级别上配置。		字符串
accessTokenSecret (security)	access secret。也可以在 OBCComponent 级别上配置。		字符串
consumerKey (security)	consumer 密钥。也可以在 OBCComponent 级别上配置。		字符串
consumerSecret (security)	consumer 机密。也可以在 OBCComponent 级别上配置。		字符串

356.3. SPRING BOOT AUTO-CONFIGURATION

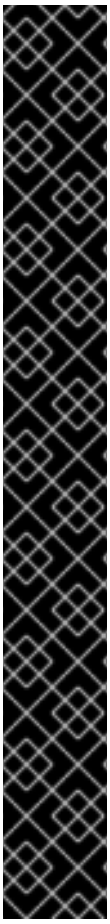
组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
camel.component.twitter-timeline.access-token	访问令牌		字符串
camel.component.twitter-timeline.access-token-secret	访问令牌 secret		字符串
camel.component.twitter-timeline.consumer-key	消费者密钥		字符串

Name	描述	默认值	类型
camel.component.twitter-timeline.consumer-secret	消费者 secret		字符串
camel.component.twitter-timeline.enabled	是否启用 twitter-timeline 组件的自动配置。这默认是启用的。		布尔值
camel.component.twitter-timeline.http-proxy-host	http 代理主机，可用于 camel-twitter。		字符串
camel.component.twitter-timeline.http-proxy-password	http 代理密码，可用于 camel-twitter。		字符串
camel.component.twitter-timeline.http-proxy-port	可用于 camel-twitter 的 http 代理端口。		整数
camel.component.twitter-timeline.http-proxy-user	http 代理用户，可用于 camel-twitter。		字符串
camel.component.twitter-timeline.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

第 357 章 TWITTER COMPONENT （已弃用）

从 Camel 版本 2.10 开始提供



重要

composite twitter 组件已弃用。使用单个组件进行 directmessage、搜索、流和时间表。

- mewitter 组件
  - Wwitw Direct Message
  - author Search
  - someonewitter Streaming
  - author Timeline

Twitter 组件通过封封witwitw 4J，启用witter API 的最有用的功能。它允许直接、轮询或事件驱动的时间表、用户、趋势和直接消息消耗。另外，它支持在状态更新或直接消息中生成消息。

author 现在需要使用 OAuth 进行所有客户端应用程序身份验证。要将 camel-twitter 与您的帐户搭配使用，您需要在为 https://dev.twitter.com/apps/new 的 author 中创建一个新应用程序，并授予应用程序对您的帐户的访问权限。最后，生成您的访问令牌和 secret。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-twitter</artifactId>
  <version>${camel-version}</version>
</dependency>
```

357.1. URI 格式

```
twitter://endpoint[?options]
```

357.2. WWITW 组件

twitter 组件可以使用witter帐户设置进行配置，在使用前必须对其进行配置。

OBC 组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
accessToken (security)	访问令牌		字符串
accessTokenSecret (security)	访问令牌 secret		字符串
consumerKey (security)	消费者密钥		字符串
consumerSecret (security)	消费者 secret		字符串
httpProxyHost (proxy)	http 代理主机，可用于 camel-twitter。		字符串
httpProxyUser (proxy)	http 代理用户，可用于 camel-twitter。		字符串
httpProxyPassword (proxy)	http 代理密码，可用于 camel-twitter。		字符串
httpProxyPort (proxy)	可用于 camel-twitter 的 http 代理端口。		int
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

您还可以直接在端点中配置这些选项。

357.3. 消费者端点

camel-twitter 而不是端点通过一个路由交换返回列表，而是为每个返回的对象创建一个路由交换。例如，如果"timeline/home"结果为五个状态，则路由将执行五次（每个 Status 的一个）。

端点	Context	正文类型	备注
directmessage	直接、轮询	twitter4j.DirectMessage	
search	直接、轮询	twitter4j.Status	
streaming/filter	事件、轮询	twitter4j.Status	
streaming/sample	事件、轮询	twitter4j.Status	
streaming/user	事件、轮询	twitter4j.Status	Camel 2.16: 要接收来自受保护的用户和帐户的十二者。
timeline/home	直接、轮询	twitter4j.Status	
时间表/指导	直接、轮询	twitter4j.Status	
timeline/public	**直接、轮询**	**twitter4j.Status**	@deprecated.改为使用时间表/home 或 direct/home。从 *Camel 2.11 onwards shim 中删除
timeline/retweetsofme	直接、轮询	twitter4j.Status	
timeline/user	直接、轮询	twitter4j.Status	

端点	Context	正文类型	备注
trends/daily	*Camel 2.10.1: direct, polling*	twitter4j.Status	@deprecated.从 Camel 2.11 开始。
趋势/每周	*Camel 2.10.1: direct, polling*	twitter4j.Status	@deprecated.从 Camel 2.11 开始。

357.4. 制作者端点

端点	正文类型
directmessage	字符串
search	List<twitter4j.Status>
timeline/user	字符串

357.5. URI 选项

OBC 端点使用 URI 语法进行配置：

```
twitter:kind
```

使用以下路径和查询参数：

357.5.1. 路径参数(1 参数)：

Name	描述	默认值	类型
kind	**必需** 端点类型		字符串

357.5.2. 查询参数(44 参数)：

Name	描述	默认值	类型
user (common)	用户名，用于用户时间表消耗，直接消息 production 等。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
type (consumer)	要使用的端点类型。只有流支持事件类型。	轮询	EndpointType
distanceMetric (consumer)	非流域搜索使用，使用配置的指标按 radius 搜索。单位可以是 mi 代表 miles，也可以是 kilometers 的 km。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。	km	字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
extendedMode (consumer)	用于启用来自 twitter 的完整文本（例如，接收包含超过 140 个字符的 12 倍）。	true	布尔值
latitude (consumer)	非流地搜索用于按 latitude 搜索。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。		å�Œ
locations (consumer)	绑定框，由 lat/lons 对创建。可用于 streaming/filter。对定义为 lat,lon。和多个 paris 可以通过分号隔开。		字符串
longitude (consumer)	非流地搜索用于按长期搜索进行搜索。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。		å�Œ
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy

Name	描述	默认值	类型
RADIUS (consumer)	非流地搜索用于按 radius 搜索。您需要配置所有以下选项： longitude、latitude、radius 和 distanceMetric。		å‘Œ
twitterStream (consumer)	使用自定义实例 OBCStream		TwitterStream
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
count (filter)	每个页面限制结果数。	5	整数
filterOld (filter)	过滤掉之前轮询的旧 12eets。此状态仅存储在内存中，并且基于最后的 12 个 ID。	true	布尔值
关键字 (filter)	可用于搜索和流/过滤。可以使用逗号分隔多个值。		字符串
lang (filter)	用于搜索的 lang 字符串 ISO_639-1		字符串
numberOfPages (filter)	您希望 camel-twitter 使用的页结果数。	1	整数
sinceId (filter)	最后的 Teet id，将用于拉取 Teets。当长时间运行后，camel 路由重启时会很有用。	1	long
userIds (filter)	根据用户 ID 过滤的 streaming/filter：可以使用逗号分隔多个值。		字符串
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。	30000	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值

Name	描述	默认值	类型
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
sortById (sort)	按 id 排序，因此最旧的是第一个，最后是最新的。	true	布尔值
httpProxyHost (proxy)	http 代理主机，可用于 camel-twitter。也可以在 OBCComponent 级别上配置。		字符串
httpProxyPassword (proxy)	http 代理密码，可用于 camel-twitter。也可以在 OBCComponent 级别上配置。		字符串
httpProxyPort (proxy)	可用于 camel-twitter 的 http 代理端口。也可以在 OBCComponent 级别上配置。		整数
httpProxyUser (proxy)	http 代理用户，可用于 camel-twitter。也可以在 OBCComponent 级别上配置。		字符串
accessToken (security)	访问令牌。也可以在 OBCComponent 级别上配置。		字符串
accessTokenSecret (security)	access secret。也可以在 OBCComponent 级别上配置。		字符串

Name	描述	默认值	类型
consumerKey (security)	consumer 密钥。也可以在 OBCComponent 级别上配置。		字符串
consumerSecret (security)	consumer 机密。也可以在 OBCComponent 级别上配置。		字符串

357.6. SPRING BOOT AUTO-CONFIGURATION

组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
camel.component.twitter.access-token	访问令牌		字符串
camel.component.twitter.access-token-secret	访问令牌 secret		字符串
camel.component.twitter.consumer-key	消费者密钥		字符串
camel.component.twitter.consumer-secret	消费者 secret		字符串
camel.component.twitter.enabled	启用 twitter 组件	true	布尔值
camel.component.twitter.http-proxy-host	http 代理主机，可用于 camel-twitter。		字符串
camel.component.twitter.http-proxy-password	http 代理密码，可用于 camel-twitter。		字符串
camel.component.twitter.http-proxy-port	可用于 camel-twitter 的 http 代理端口。		整数

Name	描述	默认值	类型
camel.component.twitter.http-proxy-user	http 代理用户，可用于 camel-twitter。		字符串
camel.component.twitter.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

357.7. 消息标头

Name	描述
CamelTwitterKeywords	搜索制作者使用此标头动态更改搜索关键字。
CamelTwitterSearchLanguage	Camel 2.11.0： 此标头可以覆盖 **lang** 选项，该选项动态设置搜索语言
CamelTwitterCount	Camel 2.11.0 此标头可覆盖 **count** 选项，它会设置将返回的最大 12itters。
CamelTwitterNumberOfPages	Camel 2.11.0 此标头可覆盖 **numberOfPages** 选项，它设定了我们想要 Titter 返回的页面数量。

357.8. 消息正文

所有消息正文都利用了 VMWare4J API 提供的对象。

357.9. 使用案例

注意

API 速率限制： Twitter REST API 封装有 OBC4J 封装的 API 速率限制。您可以在 API Rate Limits 文档中找到每个方法限制。请注意，页面中没有列出的端点/资源默认为每个窗口分配的用户 15 个请求。

357.9.1. 要在 OBC 配置集中创建状态更新，请发送此制作者一个 String 正文：

```
from("direct:foo")
  .to("twitter://timeline/user?consumerKey=[s]&consumerSecret=[s]&accessToken=
[s]&accessTokenSecret=[s]);
```

-

357.9.2. 要轮询，每 60 sec.，您的主页时间表上的所有状态：

```
from("twitter://timeline/home?type=polling&delay=60&consumerKey=[s]&consumerSecret=
[s]&accessToken=[s]&accessTokenSecret=[s]")
  .to("bean:blah");
```

357.9.3. 仅搜索关键字 'camel' 的所有状态：

```
from("twitter://search?type=polling&keywords=camel&consumerKey=[s]&consumerSecret=
[s]&accessToken=[s]&accessTokenSecret=[s]")
  .to("bean:blah");
```

357.9.4. 使用带有静态关键字的制作者搜索：

```
from("direct:foo")
  .to("twitter://search?keywords=camel&consumerKey=[s]&consumerSecret=
[s]&accessToken=[s]&accessTokenSecret=[s]");
```

357.9.5. 使用带有标头的动态关键字的制作者进行搜索：

在 bar 标头中，我们有要搜索的关键字，因此我们可以将这个值分配给 CamelTwitterKeywords 标头：

```
from("direct:foo")
  .setHeader("CamelTwitterKeywords", header("bar"))
  .to("twitter://search?consumerKey=[s]&consumerSecret=[s]&accessToken=
[s]&accessTokenSecret=[s]");
```

357.10. EXAMPLE

另请参阅 zSystems Websocket 示例。

357.11. 另请参阅

- 配置 Camel
- 组件
- 端点

- *开始使用*

- *OBC Websocket 示例*

第 358 章 UNDERTOW 组件

从 Camel 版本 2.16 开始提供

undertow 组件提供基于 HTTP 和 WebSocket 的端点，用于消耗和生成 HTTP/WebSocket 请求。

也就是说，Undertow 组件的行为是简单的 Web 服务器。Undertow 也可以用作 http 客户端，这意味着您也可以将其用作制作者。

提示

自 Camel 版本 2.21 起，undertow 组件还支持 WebSocket 连接，从而充当 Camel websocket 组件的简易替代品或tmo consist-websocket 组件。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-undertow</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

358.1. URI 格式

```
undertow:http://hostname[:port][/resourceUri][?options]
undertow:https://hostname[:port][/resourceUri][?options]
undertow:ws://hostname[:port][/resourceUri][?options]
undertow:wss://hostname[:port][/resourceUri][?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

358.2. 选项

Undertow 组件支持下列 5 个选项：

Name	描述	默认值	类型
undertowHttpBinding (advanced)	使用自定义 HttpBinding 来控制 Camel 消息和 HttpClient 之间的映射。		UndertowHttpBinding
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
hostOptions (advanced)	配置常见选项，如线程池		UndertowHostOptions
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Undertow 端点使用 URI 语法进行配置：

```
undertow:httpURI
```

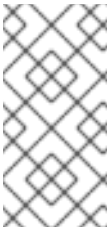
使用以下路径和查询参数：

358.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
httpURI	**必需** 使用的 HTTP 端点的 url。		URI

358.2.2. 查询参数(22 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值

Name	描述	默认值	类型
httpMethodRestrict (consumer)	仅在 HttpMethod 匹配时才允许使用，如 GET/POST/PUT 等。可以使用逗号分隔多个方法。		字符串
matchOnUriPrefix (consumer)	如果找不到完全匹配，消费者是否应该尝试通过匹配 URI 前缀来查找目标消费者。	false	布尔值
optionsEnabled (consumer)	指定是否为这个 Servlet 消费者启用 HTTP OPTIONS。默认情况下关闭 OPTIONS。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
handlers (consumer)	指定 registry 中要查找的以逗号分隔的 **io.undertow.server.HttpHandler** 实例集合。这些处理程序添加到 Undertow 处理程序链中，例如为添加安全性。  **注意** 您不能使用相同的端口号，将不同的处理程序用于不同的 Undertow 端点。处理程序与端口号关联。如果您需要不同的处理程序，则使用不同的端口号。		字符串
cookieHandler (producer)	配置 Cookie 处理程序，以维护 HTTP 会话		CookieHandler
keepalive (producer)	设置以确保因为不活跃而不会关闭套接字	true	布尔值
options (producer)	设置其他频道选项。可以使用的选项在 org.xnio.Options 中定义。要从端点 uri 配置，然后使用选项为每个选项添加前缀，如 option.close-abort=true&option.send-buffer=8192		Map
reuseAddresses (producer)	设置以方便套接字多路	true	布尔值
tcpNoDelay (producer)	设置以提高 TCP 协议性能	true	布尔值

Name	描述	默认值	类型
throwExceptionOnFailure (producer)	如果来自远程服务器的失败响应，用于禁用抛出 HttpOperationFailedException。这样，您可以获取所有响应，而不考虑 HTTP 状态代码。	true	布尔值
transferException (producer)	如果在消费者端启用和交换失败处理，如果原因 Exception 在响应中作为 application/x-java-serialized-object 内容类型发送序列化，且原因为 Exception。在生产者侧，异常将反序列化并丢弃为 HttpOperationFailedException。原因异常需要按顺序处理。默认为关闭。如果您启用它，则 Java 会将传入的数据反序列化 Java 请求，这可能会成为潜在的安全风险。	false	布尔值
headerFilterStrategy (advanced)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
undertowHttpBinding (advanced)	使用自定义 UndertowHttpBinding 来控制 Camel 消息和 undertow 之间的映射。		UndertowHttpBinding
fireWebSocketChannelEvents (websocket)	如果为 true，则消费者将在新的 WebSocket 对等点连接、断开连接等时向路由发布通知。请参阅 UndertowConstants.EVENT_TYPE 和 EventType。	false	布尔值
sendTimeout (websocket)	发送到 websocket 频道时的超时时间（毫秒）。默认超时为 30000 (30 秒)。	30000	整数
sendToAll (websocket)	发送到所有 websocket 订阅者。可用于在端点级别上配置，而不必在消息上使用 UndertowConstants.SEND_TO_ALL 标头。		布尔值
useStreaming (websocket)	如果为 true，则通过 WebSocket 的文本和二进制消息将分别嵌套为 java.io.Reader 和 java.io.InputStream，然后再将它们分别传递为 Exchange；否则它们将分别传递为 String 和字节。	false	布尔值
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters

358.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
camel.component.undertow.enabled	启用 undertow 组件	true	布尔值
camel.component.undertow.host-options.buffer-size	Undertow 主机的缓冲区大小。		整数
camel.component.undertow.host-options.direct-buffers	设置 Undertow 主机是否应该使用直接缓冲区。		布尔值
camel.component.undertow.host-options.http2-enabled	设置 Undertow 主机是否应使用 http2 协议。		布尔值
camel.component.undertow.host-options.io-threads	Undertow 主机中使用的 io 线程数量。		整数
camel.component.undertow.host-options.worker-threads	Undertow 主机中使用的工作线程数量。		整数
camel.component.undertow.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.undertow.ssl-context-parameters	使用 SSLContextParameters 配置安全性。选项是 org.apache.camel.util.jsse.SSLContextParameters 类型。		字符串
camel.component.undertow.undertow-http-binding	使用自定义 HttpBinding 来控制 Camel 消息和 HttpClient 之间的映射。选项是 org.apache.camel.component.undertow.UndertowHttpBinding 类型。		字符串
camel.component.undertow.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

358.4. 消息标头

Camel 使用与 HTTP 组件相同的消息标头。在 Camel 2.2 中，它还使用 Exchange.HTTP_CHUNKED、CamelHttpChunked 标头打开或关闭 camel-undertow 消费者上的 chuched 编码。

Camel 还会填充 所有 request.parameter 和 request.headers。例如，给定一个带有 URL 的客户端请求 http://myserver/myserver?orderid=123，交换将包含名为 orderid 的标头，其值为 123。

358.5. HTTP PRODUCER 示例

以下是如何将 HTTP 请求发送到现有 HTTP 端点的基本示例。

Java DSL

```
from("direct:start")
   .to("undertow:http://www.google.com");
```

或者在 XML 中

```
<route>
   <from uri="direct:start"/>
   <to uri="undertow:http://www.google.com"/>
<route>
```

358.6. HTTP CONSUMER 示例

在这个示例中，我们定义在 http://localhost:8080/myapp/myservice 中公开 HTTP 服务的路由：

```
<route>
  <from uri="undertow:http://localhost:8080/myapp/myservice"/>
  <to uri="bean:myBean"/>
</route>
```

358.7. WEBSOCKET 示例

在这个示例中，我们定义在 http://localhost:8080/myapp/mysocket 中公开 WebSocket 服务的路由，并返回对同一频道的响应：

```
<route>
  <from uri="undertow:ws://localhost:8080/myapp/mysocket"/>
  <transform><simple>Echo ${body}</simple></transform>
  <to uri="undertow:ws://localhost:8080/myapp/mysocket"/>
</route>
```

358.8. HTTP/2 示例

在这个示例中，我们将 camel-undertow 组件配置为支持 HTTP/2 协议，并在 http://localhost:7766/foo/bar 公开 HTTP 服务。

这个示例的目录结构如下：

```
├── pom.xml (1)
├── README.md
└── src
    └── main
        └── resources
            └── META-INF
                └── spring
                    └── camel-context.xml (2)
```

以下文件对于配置 camel-undertow 组件来支持 HTTP/2 协议非常重要：

(1)

pom.xml ：包含以下属性和依赖项：

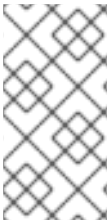
```
<properties>
   <camel.version>2.23.1</camel.version>
</properties>

<dependency>
   <groupId>org.apache.camel</groupId>
   <artifactId>camel-core</artifactId>
   <version>${camel.version}</version>
</dependency>
<dependency>
   <groupId>org.apache.camel</groupId>
   <artifactId>camel-spring</artifactId>
   <version>${camel.version}</version>
</dependency>
<dependency>
   <groupId>org.apache.camel</groupId>
   <artifactId>camel-undertow</artifactId>
   <version>${camel.version}</version>
   </dependency>
```

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-http-common</artifactId>
  <version>${camel.version}</version>
</dependency>

<plugin>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-maven-plugin</artifactId>
  <version>${camel.version}</version>
  <configuration>
    <fileApplicationContextUri>src/main/resources/META-INF/spring/camel-
context.xml</fileApplicationContextUri>
  </configuration>
</plugin>
```

2

camel-context.xml ：配置 camel-undertow 组件，如下所示：

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-
spring.xsd">
  <camelContext id="cbr-example-context" xmlns="http://camel.apache.org/schema/spring">
    <route id="cbr-route" trace="true">
      <from id="_from1" uri="undertow:http://localhost:7766/foo/bar"/>
      <setBody id="_setBody1">
        <constant>Sending Response</constant>
      </setBody>
      <log id="_log5" message="Headers ${in.headers}"/>
      <log id="_log5" message="Done processing ${body}"/>
    </route>
  </camelContext>
  <bean class="org.apache.camel.component.undertow.UndertowComponent" id="undertow">
    <property name="hostOptions" ref="undertowHostOptions"/>
  </bean>
  <bean    class="org.apache.camel.component.undertow.UndertowHostOptions"
id="undertowHostOptions">
    <property name="http2Enabled" value="true"/>
  </bean>
</beans>
```



注意

对于 **UndertowHostOptions** 类，**http2Enabled** 属性默认设置为 **false**。然后，这个类被称为 **UndertowComponent**，它在 **camel route** 中使用。

在本例中，通过使用 pom.xml 文件中的 camel-maven-plugin，当我们执行 Maven 命令 mvn camel:run 时，我们可以在 http://localhost:7766/foo/bar 中公开 HTTP 服务。

我们可以使用 curl 命令测试此示例，如下所示：

```
$ curl -v --http2 http://localhost:7766/foo/bar
* Trying ::1...
* TCP_NODELAY set
* connect to ::1 port 7766 failed: Connection refused
* Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 7766 (#0)
> GET /foo/bar HTTP/1.1
> Host: localhost:7766
> User-Agent: curl/7.53.1
> Accept: */*
> Connection: Upgrade, HTTP2-Settings
> Upgrade: h2c
> HTTP2-Settings: AAMAAABkAARAAAAAAAIAAAAA
>
< HTTP/1.1 101 Switching Protocols
< Connection: Upgrade
< Upgrade: h2c
< Date: Sun, 10 Dec 2017 08:43:58 GMT
* Received 101
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade: len=0
* Connection state changed (MAX_CONCURRENT_STREAMS updated)!
< HTTP/2 200
< accept: */*
< http2-settings: AAMAAABkAARAAAAAAAIAAAAA
< breadcrumbid: ID-dhcppc1-1512886066149-0-25
< content-length: 16
< user-agent: curl/7.53.1
< date: Sun, 10 Dec 2017 08:43:58 GMT
<
* Connection #0 to host localhost left intact
Sending Response
```

358.9. 使用 LOCALHOST 作为主机

当您在 URL 中指定 localhost 时，Camel 仅在本地 TCP/IP 网络接口上公开端点，因此无法从其操作的计算机外部访问。

如果您需要在特定网络接口上公开 Jetty 端点，此接口的数字 IP 地址应用作主机。如果您需要在所有网络接口上公开 Jetty 端点，则应使用 0.0.0.0 地址。

要侦听整个 URI 前缀，请参阅 如何让 Jetty 匹配通配符。

如果您实际希望通过 HTTP 公开路由，且已经有一个 Servlet，您应该引用 Servlet 传输。

358.10. WILDFLY 上的 UNDERTOW 用户

WildFly 上的 camel-undertow 用户配置与独立 Camel 的配置不同。生产者端点可以正常运行。

在 WildFly 上，camel-undertow 用户利用容器提供的默认 Undertow HTTP 服务器。服务器在 undertow 子系统配置中定义。以下是 standalone.xml 中默认配置的摘录：

```
<subsystem xmlns="urn:jboss:domain:undertow:4.0">
    <buffer-cache name="default" />
    <server name="default-server">
        <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true" />
        <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-
http2="true" />
        <host name="default-host" alias="localhost">
            <location name="/" handler="welcome-content" />
            <filter-ref name="server-header" />
            <filter-ref name="x-powered-by-header" />
            <http-invoker security-realm="ApplicationRealm" />
        </host>
    </server>
</subsystem>
```

在本实例中，Undertow 配置为侦听 http 和 https socket-binding 指定的接口/端口。默认情况下，对于 http，对于端口 8080，https 为 8443。

这有以下影响：

- camel-undertow 用户将仅绑定到 localhost:8080 或 localhost:8443
- 有些端点消费者配置选项无效（请参阅以下），因为这些设置由 WildFly 容器管理

例如，如果您使用不同的主机或端口组合配置端点使用者，服务器日志文件中会显示警告。例如，以下主机和端口配置将被忽略：

-

```
from("undertow:http://somehost:1234/path/to/resource")
```

```
[org.wildfly.extension.camel] (pool-2-thread-1) Ignoring configured host:
http://somehost:1234/path/to/resource
```

但是，消费者在默认主机和端口 localhost:8080 或 localhost:8443 上仍然可用。

358.10.1. 配置替代端口

如果要接受替代端口，则必须通过 WildFly 子系统配置它们。这在服务器文档中解释：

https://access.redhat.com/documentation/zh-cn/red_hat_jboss_enterprise_application_platform/7.1/html/configuration_guide/configuring_the_web_server_undertow

358.10.2. 在 WildFly 中忽略了 camel-undertow 消费者配置选项

hostOptions

有关如何配置服务器主机选项，请参阅 WildFly undertow 配置指南：

https://access.redhat.com/documentation/zh-cn/red_hat_jboss_enterprise_application_platform/7.1/html-single/how_to_configure_server_security/#configure_one_way_and_two_way_ssl_tls_for_application

sslContextParameters

要配置 SSL，请参阅 WildFly SSL 配置指南：

https://access.redhat.com/documentation/zh-cn/red_hat_jboss_enterprise_application_platform/7.1/html-single/how_to_configure_server_security/#configure_one_way_and_two_way_ssl_tls_for_application

第 359 章 UNIVOCITY CSV DATAFORMAT

从 Camel 版本 2.15 开始提供

这个 数据格式 使用 uniVocity-parsers 来读取和写入 3 种表格数据文本文件：

- CSV (9 月值)，其中值用符号（通常是逗号）分隔
- 固定宽度，其中值有已知的大小
- TSV (Tabular Separated Values)，其中字段由 tabulation 分隔

因此，根据 uniVocity-parsers 有 3 个数据格式。

如果使用 Maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-univocity-parsers</artifactId>
  <version>x.x.x</version>
</dependency>
```

359.1. 选项

uniVocity-parsers 的大多数配置选项都以数据格式提供。如果您需要特定选项的更多信息，请参阅其文档页面。

3 数据格式共享通用选项并具有专用的选项，本节将将它们全部显示。

359.2. 选项

uniVocity CSV dataformat 支持 18 个选项，如下所列。

Name	默认值	Java 类型	描述
quoteAllFields	**false**	**布尔值**	在编写值时，是否必须引用所有值。
quote	"	**字符串**	quote 符号。
quoteEscape	"	**字符串**	quote 转义符号
delimiter	,	**字符串**	值的分隔符
nullValue		**字符串**	null 值的字符串表示。默认值为 null
skipEmptyLines	**true**	**布尔值**	必须忽略空行。默认值为 true
ignoreTrailingWhitespaces	**true**	**布尔值**	尾随空格是否必须忽略。默认值为 true
ignoreLeadingWhitespaces	**true**	**布尔值**	是否需要忽略前导空格。默认值为 true
headersDisabled	**false**	**布尔值**	是否禁用标头。定义后，此选项将标头明确设置为 null，表示没有标头。默认值为 false
headerExtractionEnabled	**false**	**布尔值**	在测试文档的第一行中是否必须读取标头。默认值为 false
numberOfRecordsToRead		**整数**	要读取的最大记录数。
emptyValue		**字符串**	空值的 String 表示
lineSeparator		**字符串**	文件的行分隔符 默认使用 JVM 平台行分隔符
normalizedLineSeparator		**字符串**	文件的规范化行分隔符。默认值是一个新行字符。
注释	#	**字符串**	注释符号。默认值为 #
lazyLoad	**false**	**布尔值**	unmarshalling 是否应该生成一个迭代器，该实用程序可即时读取行，或者是否必须读取所有行。默认值为 false
asMap	**false**	**布尔值**	unmarshalling 是否应该为行值而不是列表生成映射。它需要有标头（定义或收集）。默认值为 false

Name	默认值	Java 类型	描述
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

359.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 19 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.univocity-csv.as-map	unmarshalling 是否应该为行值而不是列表生成映射。它需要有标头（定义或收集）。默认值为 false	false	布尔值
camel.dataformat.univocity-csv.comment	注释符号。默认值为 #	#	字符串
camel.dataformat.univocity-csv.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.univocity-csv.delimiter	值的分隔符	,	字符串
camel.dataformat.univocity-csv.empty-value	空值的 String 表示		字符串
camel.dataformat.univocity-csv.enabled	启用 univocity-csv dataformat	true	布尔值
camel.dataformat.univocity-csv.header-extraction-enabled	在测试文档的第一行中是否必须读取标头。默认值为 false	false	布尔值

Name	描述	默认值	类型
camel.dataformat.univocity-csv.headers-disabled	是否禁用标头。定义后，此选项将标头明确设置为 null，表示没有标头。默认值为 false	false	布尔值
camel.dataformat.univocity-csv.ignore-leading-whitespaces	是否需要忽略前导空格。默认值为 true	true	布尔值
camel.dataformat.univocity-csv.ignore-trailing-whitespaces	尾随空格是否必须忽略。默认值为 true	true	布尔值
camel.dataformat.univocity-csv.lazy-load	unmarshalling 是否应该生成一个迭代器，该实用程序可即时读取行，或者是否必须读取所有行。默认值为 false	false	布尔值
camel.dataformat.univocity-csv.line-separator	文件的行分隔符 默认使用 JVM 平台行分隔符		字符串
camel.dataformat.univocity-csv.normalized-line-separator	文件的规范化行分隔符。默认值是一个新行字符。		字符串
camel.dataformat.univocity-csv.null-value	null 值的字符串表示。默认值为 null		字符串
camel.dataformat.univocity-csv.number-of-records-to-read	要读取的最大记录数。		整数
camel.dataformat.univocity-csv.quote	quote 符号。	"	字符串
camel.dataformat.univocity-csv.quote-all-fields	在编写值时，是否必须引用所有值。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.univocity-csv.quote-escape	quote 转义符号	"	字符串
camel.dataformat.univocity-csv.skip-empty-lines	必须忽略空行。默认值为 true	true	布尔值

359.4. MARSHALLING USAGES

marshalling 接受：

- ***映射列表(L'ist<Map<String,?>>'), 每行一个***
- ***一行的单个映射(Map<String,?>)***

任何其它正文都会抛出异常。

359.4.1. 使用示例：将映射到 CSV 格式

```
<route>
   <from uri="direct:input"/>
   <marshal>
      <univocity-csv/>
   </marshal>
   <to uri="mock:result"/>
</route>
```

359.4.2. 使用示例：将映射到固定宽度格式

```
<route>
   <from uri="direct:input"/>
   <marshal>
      <univocity-fixed padding="_">
         <univocity-header length="5"/>
         <univocity-header length="5"/>
         <univocity-header length="5"/>
      </univocity-fixed>
```

```
    </marshal>
    <to uri="mock:result"/>
</route>
```

359.4.3. 使用示例：将 map 打包为 TSV 格式

```
<route>
    <from uri="direct:input"/>
    <marshal>
        <univocity-tsv/>
    </marshal>
    <to uri="mock:result"/>
</route>
```

359.5. UNMARSHALLING 用法

unmarshalling 使用 InputStream 来读取数据。

每行生成：

- 包含其中所有值的列表(以Map 选项设为 false)；
- 包含标头索引的所有值的映射(以Map 选项带有 true)。

所有行都可以：

- 一次收集到一个列表（带有 false的lazyLoad 选项）；
- 使用迭代器（带 true的lazyLoad 选项）实时读取。

359.5.1. 使用示例：将 CSV 格式与自动标头映射

```
<route>
    <from uri="direct:input"/>
    <unmarshal>
        <univocity-csv headerExtractionEnabled="true" asMap="true"/>
    </unmarshal>
    <to uri="mock:result"/>
</route>
```

359.5.2. 使用示例：将固定宽度格式成列表

```
<route>
    <from uri="direct:input"/>
    <unmarshal>
        <univocity-fixed>
            <univocity-header length="5"/>
            <univocity-header length="5"/>
            <univocity-header length="5"/>
        </univocity-fixed>
    </unmarshal>
    <to uri="mock:result"/>
</route>
```

第 360 章 UNIVOCITY FIXED LENGTH DATAFORMAT

从 Camel 版本 2.15 开始提供

这个 数据格式 使用 uniVocity-parsers 来读取和写入 3 种表格数据文本文件：

- CSV (9 月值)，其中值用符号（通常是逗号）分隔
- 固定宽度，其中值有已知的大小
- TSV (Tabular Separated Values)，其中字段由 tabulation 分隔

因此，根据 uniVocity-parsers 有 3 个数据格式。

如果使用 Maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-univocity-parsers</artifactId>
  <version>x.x.x</version>
</dependency>
```

360.1. 选项

uniVocity-parsers 的大多数配置选项都以数据格式提供。如果您需要特定选项的更多信息，请参阅其文档页面。

3 数据格式共享通用选项并具有专用的选项，本节将将它们全部显示。

360.2. 选项

uniVocity Fixed Length dataformat 支持 17 个选项，如下所列。

Name	默认值	Java 类型	描述
skipTrailingCharsUntilNewline	false	布尔值	在必须忽略新行之前，是否结尾的字符。默认值为 false
recordEndsOnNewline	false	布尔值	记录是否结束在新行中。默认值为 false
padding		字符串	padding 字符。默认值为空格
nullValue		字符串	null 值的字符串表示。默认值为 null
skipEmptyLines	true	布尔值	必须忽略空行。默认值为 true
ignoreTrailingWhitespaces	true	布尔值	尾随空格是否必须忽略。默认值为 true
ignoreLeadingWhitespaces	true	布尔值	是否需要忽略前导空格。默认值为 true
headersDisabled	false	布尔值	是否禁用标头。定义后，此选项将标头明确设置为 null，表示没有标头。默认值为 false
headerExtractionEnabled	false	布尔值	在测试文档的第一行中是否必须读取标头。默认值为 false
numberOfRecordsToRead		整数	要读取的最大记录数。
emptyValue		字符串	空值的 String 表示
lineSeparator		字符串	文件的行分隔符 默认使用 JVM 平台行分隔符
normalizedLineSeparator		字符串	文件的规范化行分隔符。默认值是一个新行字符。
注释	#	字符串	注释符号。默认值为 #
lazyLoad	false	布尔值	unmarshalling 是否应该生成一个迭代器，该实用程序可即时读取行，或者是否必须读取所有行。默认值为 false
asMap	false	布尔值	unmarshalling 是否应该为行值而不是列表生成映射。它需要有标头（定义或收集）。默认值为 false
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

360.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 18 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.univocity-fixed.as-map	unmarshalling 是否应该为行值而不是列表生成映射。它需要有标头（定义或收集）。默认值为 false	false	布尔值
camel.dataformat.univocity-fixed.comment	注释符号。默认值为 #	#	字符串
camel.dataformat.univocity-fixed.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.univocity-fixed.empty-value	空值的 String 表示		字符串
camel.dataformat.univocity-fixed.enabled	启用 univocity-fixed dataformat	true	布尔值
camel.dataformat.univocity-fixed.header-extraction-enabled	在测试文档的第一行中是否必须读取标头。默认值为 false	false	布尔值
camel.dataformat.univocity-fixed.headers-disabled	是否禁用标头。定义后，此选项将标头明确设置为 null，表示没有标头。默认值为 false	false	布尔值
camel.dataformat.univocity-fixed.ignore-leading-whitespaces	是否需要忽略前导空格。默认值为 true	true	布尔值

Name	描述	默认值	类型
camel.dataformat.univocity-fixed.ignore-trailing-whitespaces	尾随空格是否必须忽略。默认值为 true	true	布尔值
camel.dataformat.univocity-fixed.lazy-load	unmarshalling 是否应该生成一个迭代器，该实用程序可即时读取行，或者是否必须读取所有行。默认值为 false	false	布尔值
camel.dataformat.univocity-fixed.line-separator	文件的行分隔符 默认使用 JVM 平台行分隔符		字符串
camel.dataformat.univocity-fixed.normalized-line-separator	文件的规范化行分隔符。默认值是一个新行字符。		字符串
camel.dataformat.univocity-fixed.null-value	null 值的字符串表示。默认值为 null		字符串
camel.dataformat.univocity-fixed.number-of-records-to-read	要读取的最大记录数。		整数
camel.dataformat.univocity-fixed.padding	padding 字符。默认值为空格		字符串
camel.dataformat.univocity-fixed.record-ends-on-newline	记录是否结束在新行中。默认值为 false	false	布尔值
camel.dataformat.univocity-fixed.skip-empty-lines	必须忽略空行。默认值为 true	true	布尔值
camel.dataformat.univocity-fixed.skip-trailing-chars-until-newline	在必须忽略新行之前，是否结尾的字符。默认值为 false	false	布尔值

360.4. MARSHALLING USAGES

marshalling 接受：

- *映射列表(L'ist<Map<String,?>>'), 每行一个*
- *一行的单个映射(Map<String,?>)*

任何其它正文都会抛出异常。

360.4.1. 使用示例：将映射到 CSV 格式

```
<route>
  <from uri="direct:input"/>
  <marshal>
    <univocity-csv/>
  </marshal>
  <to uri="mock:result"/>
</route>
```

360.4.2. 使用示例：将映射到固定宽度格式

```
<route>
  <from uri="direct:input"/>
  <marshal>
    <univocity-fixed padding="_">
      <univocity-header length="5"/>
      <univocity-header length="5"/>
      <univocity-header length="5"/>
    </univocity-fixed>
  </marshal>
  <to uri="mock:result"/>
</route>
```

360.4.3. 使用示例：将 map 打包为 TSV 格式

```
<route>
  <from uri="direct:input"/>
  <marshal>
    <univocity-tsv/>
  </marshal>
  <to uri="mock:result"/>
</route>
```

360.5. UNMARSHALLING 用法

unmarshalling 使用 InputStream 来读取数据。

每行生成：

- 包含其中所有值的列表(以Map 选项设为 false)；
- 包含标头索引的所有值的映射(以Map 选项带有 true)。

所有行都可以：

- 一次收集到一个列表（带有 false的lazyLoad 选项）；
- 使用迭代器（带 true的lazyLoad 选项）实时读取。

360.5.1. 使用示例：将 CSV 格式与自动标头映射

```
<route>
  <from uri="direct:input"/>
  <unmarshal>
    <univocity-csv headerExtractionEnabled="true" asMap="true"/>
  </unmarshal>
  <to uri="mock:result"/>
</route>
```

360.5.2. 使用示例：将固定宽度格式成列表

```
<route>
  <from uri="direct:input"/>
  <unmarshal>
    <univocity-fixed>
      <univocity-header length="5"/>
      <univocity-header length="5"/>
      <univocity-header length="5"/>
    </univocity-fixed>
  </unmarshal>
  <to uri="mock:result"/>
</route>
```

第 361 章 UNIVOCITY TSV DATAFORMAT

从 Camel 版本 2.15 开始提供

这个 数据格式 使用 uniVocity-parsers 来读取和写入 3 种表格数据文本文件：

- CSV (9 月值)，其中值用符号（通常是逗号）分隔
- 固定宽度，其中值有已知的大小
- TSV (Tabular Separated Values)，其中字段由 tabulation 分隔

因此，根据 uniVocity-parsers 有 3 个数据格式。

如果使用 Maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-univocity-parsers</artifactId>
  <version>x.x.x</version>
</dependency>
```

361.1. 选项

uniVocity-parsers 的大多数配置选项都以数据格式提供。如果您需要特定选项的更多信息，请参阅其文档页面。

3 数据格式共享通用选项并具有专用的选项，本节将将它们全部显示。

361.2. 选项

uniVocity TSV dataformat 支持 15 个选项，如下所列。

Name	默认值	Java 类型	描述
escapeChar	\	字符串	转义字符。
nullValue		字符串	null 值的字符串表示。默认值为 null
skipEmptyLines	true	布尔值	必须忽略空行。默认值为 true
ignoreTrailingWhitespaces	true	布尔值	尾随空格是否必须忽略。默认值为 true
ignoreLeadingWhitespaces	true	布尔值	是否需要忽略前导空格。默认值为 true
headersDisabled	false	布尔值	是否禁用标头。定义后，此选项将标头明确设置为 null，表示没有标头。默认值为 false
headerExtractionEnabled	false	布尔值	在测试文档的第一行中是否必须读取标头。默认值为 false
numberOfRecordsToRead		整数	要读取的最大记录数。
emptyValue		字符串	空值的 String 表示
lineSeparator		字符串	文件的行分隔符 默认使用 JVM 平台行分隔符
normalizedLineSeparator		字符串	文件的规范化行分隔符。默认值是一个新行字符。
注释	#	字符串	注释符号。默认值为 #
lazyLoad	false	布尔值	unmarshalling 是否应该生成一个迭代器，该实用程序可即时读取行，或者是否必须读取所有行。默认值为 false
asMap	false	布尔值	unmarshalling 是否应该为行值而不是列表生成映射。它需要有标头（定义或收集）。默认值为 false
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

361.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 16 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.univocity-tsv.as-map	unmarshalling 是否应该为行值而不是列表生成映射。它需要有标头（定义或收集）。默认值为 false	false	布尔值
camel.dataformat.univocity-tsv.comment	注释符号。默认值为 #	#	字符串
camel.dataformat.univocity-tsv.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.univocity-tsv.empty-value	空值的 String 表示		字符串
camel.dataformat.univocity-tsv.enabled	启用 univocity-tsv dataformat	true	布尔值
camel.dataformat.univocity-tsv.escape-char	转义字符。	\	字符串
camel.dataformat.univocity-tsv.header-extraction-enabled	在测试文档的第一行中是否必须读取标头。默认值为 false	false	布尔值
camel.dataformat.univocity-tsv.headers-disabled	是否禁用标头。定义后，此选项将标头明确设置为 null，表示没有标头。默认值为 false	false	布尔值
camel.dataformat.univocity-tsv.ignore-leading-whitespaces	是否需要忽略前导空格。默认值为 true	true	布尔值
camel.dataformat.univocity-tsv.ignore-trailing-whitespaces	尾随空格是否必须忽略。默认值为 true	true	布尔值

Name	描述	默认值	类型
camel.dataformat.univocity-tsv.lazy-load	unmarshalling 是否应该生成一个迭代器，该实用程序可即时读取行，或者是否必须读取所有行。默认值为 false	false	布尔值
camel.dataformat.univocity-tsv.line-separator	文件的行分隔符 默认使用 JVM 平台行分隔符		字符串
camel.dataformat.univocity-tsv.normalized-line-separator	文件的规范化行分隔符。默认值是一个新行字符。		字符串
camel.dataformat.univocity-tsv.null-value	null 值的字符串表示。默认值为 null		字符串
camel.dataformat.univocity-tsv.number-of-records-to-read	要读取的最大记录数。		整数
camel.dataformat.univocity-tsv.skip-empty-lines	必须忽略空行。默认值为 true	true	布尔值

361.4. MARSHALLING USAGES

marshalling 接受：

- ***映射列表(L'ist<Map<String,?>>'), 每行一个***
- ***一行的单个映射(Map<String,?>)***

任何其它正文都会抛出异常。

361.4.1. 使用示例：将映射到 CSV 格式

```
<route>
  <from uri="direct:input"/>
  <marshal>
    <univocity-csv/>
  </marshal>
  <to uri="mock:result"/>
</route>
```

361.4.2. 使用示例：将映射到固定宽度格式

```
<route>
  <from uri="direct:input"/>
  <marshal>
    <univocity-fixed padding="_">
      <univocity-header length="5"/>
      <univocity-header length="5"/>
      <univocity-header length="5"/>
    </univocity-fixed>
  </marshal>
  <to uri="mock:result"/>
</route>
```

361.4.3. 使用示例：将 map 打包为 TSV 格式

```
<route>
  <from uri="direct:input"/>
  <marshal>
    <univocity-tsv/>
  </marshal>
  <to uri="mock:result"/>
</route>
```

361.5. UNMARSHALLING 用法

unmarshalling 使用 InputStream 来读取数据。

每行生成：

- 包含其中所有值的列表(以Map 选项设为 **false**)；
- 包含标头索引的所有值的映射(以Map 选项带有 **true**)。

所有行都可以：

- 一次收集到一个列表（带有 **false**的**lazyLoad** 选项）；
- 使用迭代器（带 **true**的**lazyLoad** 选项）实时读取。

361.5.1. 使用示例：将 CSV 格式与自动标头映射

```
<route>
  <from uri="direct:input"/>
  <unmarshal>
    <univocity-csv headerExtractionEnabled="true" asMap="true"/>
  </unmarshal>
  <to uri="mock:result"/>
</route>
```

361.5.2. 使用示例：将固定宽度格式成列表

```
<route>
  <from uri="direct:input"/>
  <unmarshal>
    <univocity-fixed>
      <univocity-header length="5"/>
      <univocity-header length="5"/>
      <univocity-header length="5"/>
    </univocity-fixed>
  </unmarshal>
  <to uri="mock:result"/>
</route>
```

第 362 章 验证器组件

从 Camel 版本 1.1 开始提供

Validation 组件使用 JAXP Validation API 并根据任何受支持的 XML 架构语言（默认为 XML Schema）来执行消息正文的 XML 验证

请注意，Jing 组件还支持以下有用的模式语言：

- RelaxNG Compact Syntax
- RelaxNG XML Syntax

MSV 组件还支持 RelaxNG XML 语法。

362.1. URI 格式

```
validator:someLocalOrRemoteResource
```

其中 someLocalOrRemoteResource 是 classpath 上本地资源或指向远程资源或资源的完整 URL（包含要验证的 XSD）的一些 URL。例如：

- msv:org/foo/bar.xsd
- msv:file:../foo/bar.xsd
- msv:http://acme.com/cheese.xsd
- validator:com/mypackage/myschema.xsd

在使用 Camel 2.8 或更早的版本时，Maven 用户需要将以下依赖项添加到此组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-spring</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

从 Camel 2.9 开始，在 camel-core 中直接提供 Validation 组件。

362.2. 选项

Validator 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
resourceResolverFactory (advanced)	使用自定义 LSResourceResolver，它依赖于动态端点资源 URI		ValidatorResourceResolverFactory
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Validator 端点使用 URI 语法进行配置：

```
validator:resourceUri
```

使用以下路径和查询参数：

362.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceUri	**所需** URL 到 classpath 上的本地资源，或对 Registry 中查找 bean 的完整 URL，或包含要验证的 XSD 的文件系统上的远程资源或资源的完整 URL。		字符串

362.2.2. 查询参数(11 参数)：

Name	描述	默认值	类型
failOnNullBody (producer)	如果没有正文，是否失败。	true	布尔值
failOnNullHeader (producer)	在针对标头验证时，是否没有标头。	true	布尔值
headerName (producer)	以针对标头而不是邮件正文进行验证。		字符串
errorHandler (advanced)	使用自定义 org.apache.camel.processor.validation.ValidatorErrorHandler。默认错误处理程序捕获错误并抛出异常。		ValidatorErrorHandler
resourceResolver (advanced)	使用自定义 LSResourceResolver。另请参阅 setResourceResolverFactory (ValidatorResourceResolverFactory)		LSResourceResolver
resourceResolverFactory (advanced)	要创建依赖于端点资源 URI 的资源解析器。不得与方法 setResourceResolver (LSResourceResolver)结合使用。如果没有设置，则使用 DefaultValidatorResourceResolverFactory		ValidatorResourceResolverFactory
schemaFactory (advanced)	使用自定义 javax.xml.validation.SchemaFactory		SchemaFactory
schemaLanguage (advanced)	配置 W3C XML Schema 命名空间 URI。	http://www.w3.org/2001/XMLSchema	字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
useDom (advanced)	验证器是否应该使用 DOMSource/DOMResult 或 SaxResult。	false	布尔值
useSharedSchema (advanced)	Schema 实例是否应共享。引进了这个选项来临时解决 JDK 1.6.x 错误。Xerces 不应出现这个问题。	true	布尔值

362.3. 示例

*以下示例演示了如何 **配置从端点 direct:start 的路由，然后指向两个端点之一，根据 XML 是否与给定的 schema 匹配（在 classpath 上提供） mock:valid 或 mock:invalid。***

362.4. 高级：JMX 方法 CLEARCACHEDSCHEMA

自 Camel 2.17 起，您可以强制验证端点中的缓存模式被清除，并使用 JMX 操作 clearCachedSchema 的下一个进程调用重新读取。您还可使用此方法以编程方式清除缓存。这个方法包括在 'ValidatorEndpoint 'class' 中。

第 363 章 VELOCITY 组件

从 Camel 版本 1.2 开始提供

velocity: 组件允许您使用 Apache Velocity 模板处理消息。这在使用 Templating 生成请求的响应时是理想的选择。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-velocity</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

363.1. URI 格式

```
velocity:templateName[?options]
```

其中 templateName 是要调用的模板的 classpath-local URI，或者远程模板的完整 URL （例如：file://folder/myfile.vm）。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

363.2. 选项

Velocity 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
allowContextMapAll (producer)	设置上下文映射是否应该允许访问所有详情。默认情况下，只能访问消息正文和标头。可以启用此选项以完全访问当前 Exchange 和 CamelContext。这样做会带来潜在的安全风险，因为这会打开对 CamelContext API 的完整功能的访问。	false	布尔值

Name	描述	默认值	类型
allowTemplateFromHeader (producer)	是否允许使用来自标头的资源模板（默认 false）。**启用此选项具有安全原因。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。**	false	布尔值
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
velocityEngine (advanced)	要使用 VelocityEngine，否则创建新引擎		VelocityEngine

Velocity 端点使用 URI 语法进行配置：

```
velocity:resourceUri
```

使用以下路径和查询参数：

363.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceUri	资源 **所需的** 路径。您可以使用这些协议(classpath 为 default)使用： classpath、file、http、ref 或 bean. classpath、file 和 http 加载资源的前缀。ref 将查找 registry 中的资源。bean 将调用 bean 以供用作资源的方法。对于 bean，您可以在点后指定方法名称，如 bean:myBean.myMethod。		字符串

363.2.2. 查询参数(7 参数)：

Name	描述	默认值	类型
allowContextMapAll (producer)	设置上下文映射是否应该允许访问所有详情。默认情况下，只能访问消息正文和标头。可以启用此选项以完全访问当前 Exchange 和 CamelContext。这样做会带来潜在的安全风险，因为这会打开对 CamelContext API 的完整功能的访问。	false	布尔值

Name	描述	默认值	类型
allowTemplateFromHeader (producer)	是否允许使用来自标头的资源模板（默认 false）。**启用此选项具有安全原因。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。**	false	布尔值
contentCache (producer)	设置是否使用资源内容缓存	false	布尔值
encoding (producer)	资源内容的字符编码。		字符串
loaderCache (producer)	启用 / 禁用默认启用的 velocity 资源加载程序缓存	true	布尔值
propertiesFile (producer)	用于 VelocityEngine 初始化的属性文件的 URI。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

363.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.component.velocity.enabled	启用 velocity 组件	true	布尔值
camel.component.velocity.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.velocity.velocity-engine	要使用 VelocityEngine，否则会创建新的引擎。选项是 org.apache.velocity.app.VelocityEngine 类型。		字符串

363.4. 消息标头

***velocity** 组件对消息设置两个标头（您无法自行设置这些标头，从 Camel 2.1 velocity 组件不会设置这些标头，这会对动态模板支持造成一些副作用）：*

标头	描述
CamelVelocityResourceUri	templateName 作为 **String** 对象。
CamelVelocitySupplementalContext	Camel 2.16： 要将额外信息添加到使用的 VelocityContext 中。此标头的值应当是添加的键/值的 **Map** （覆盖具有相同名称的任何现有键）。 这可用于预设要在 velocity 端点中重复使用的一些常用键/值。

在 Velocity 评估期间设置的标头将返回到消息，并作为标头添加。然后，可以将它的值从 Velocity 返回到 Message。

例如，要在 Velocity 模板 .tm 中设置 fruit 的标头值：

```
$in.setHeader("fruit", "Apple")
```

fruit 标头现在可以从 message.out.headers 访问。

363.5. VELOCITY CONTEXT

Camel 将在 Velocity 上下文（只使用 映射）中提供交换信息。Exchange 被传输为：

key	value
交换	**Exchange** 本身。
exchange.properties	**Exchange** 属性.
标头	In 消息的标头。
camelContext	Camel 上下文实例。

key	value
Request（请求）	In 消息。
in	In 消息。
正文（body）	In 消息正文。
out	Out 消息（仅适用于 InOut 消息交换模式）。
响应	Out 消息（仅适用于 InOut 消息交换模式）。

从 Camel-2.14 开始，您可以通过设置消息标头 *CamelVelocityContext *just 等设置自定义 Velocity Context。

```
VelocityContext velocityContext = new VelocityContext(variableMap);
exchange.getIn().setHeader("CamelVelocityContext", velocityContext);
```

363.6. 热重新载入

Velocity 模板资源默认为 file 和 classpath 资源(expanded jar)的热重新加载。如果您设置了 contentCache=true，Camel 将只加载一次资源，因此无法热重新载入。当资源永不更改时，此场景可用于生产环境。

363.7. 动态模板

从 Camel 2.1
Camel 开始，提供了两个标头，您可以在其中为模板或模板内容本身定义不同的资源位置。如果设置了其中任何标头，则 Camel 会通过端点配置的资源使用此标头。这可让您在运行时提供动态模板。

标头	类型	描述
CamelVelocityResourceUri	字符串	**Camel 2.1：** 要使用的模板资源的 URI，而不是配置的端点。

标头	类型	描述
CamelVelocityTemplate	字符串	Camel 2.1：要使用的模板，而不是配置的端点。

363.8. SAMPLES

例如，您可以使用类似如下的内容

```
from("activemq:My.Queue").
  to("velocity:com/acme/MyResponse.vm");
```

要使用 Velocity 模板，对 InOut 消息交换的消息做出响应（其中有一个 JMSReplyTo 标头）。

如果要使用 InOnly 并使用信息并将其发送到另一个目的地，您可以使用以下路由：

```
from("activemq:My.Queue").
  to("velocity:com/acme/MyResponse.vm").
  to("activemq:Another.Queue");
```

和 使用内容缓存，例如要在生产环境中使用，其中 .vm 模板永不更改：

```
from("activemq:My.Queue").
  to("velocity:com/acme/MyResponse.vm?contentCache=true").
  to("activemq:Another.Queue");
```

和基于文件的资源：

```
from("activemq:My.Queue").
  to("velocity:file://myfolder/MyResponse.vm?contentCache=true").
  to("activemq:Another.Queue");
```

在 Camel 2.1 中，可以指定组件应该通过标头动态使用的模板，例如：

```
from("direct:in").
  setHeader("CamelVelocityResourceUri").constant("path/to/my/template.vm").
  to("velocity:dummy?allowTemplateFromHeader=true");
```

在 Camel 2.1 中，可以直接将模板指定为标头，组件应该通过标头动态使用，例如：

```
from("direct:in").
  setHeader("CamelVelocityTemplate").constant("Hi this is a velocity template that can do templating ${body}").
  to("velocity:dummy?allowTemplateFromHeader=true");
```

警告

启用 allowTemplateFromHeader 选项具有安全等级。例如，如果标头包含不受信任的或用户派生的内容，这最终可能会影响您的最终应用的自信性和完整性，因此请谨慎使用这个选项。

363.9. 电子邮件示例

在本例中，我们希望使用 Velocity 模板进行订单确认电子邮件。电子邮件模板已放置在 Velocity 中，因为：

```
Dear ${headers.lastName}, ${headers.firstName}

Thanks for the order of ${headers.item}.

Regards Camel Riders Bookstore
${body}
```

和 java 代码：

363.10. 另请参阅

- 配置 Camel
- 组件
- 端点

- *开始使用*

第 364 章 VERT.X COMPONENT

从 Camel 版本 2.12 开始提供

vertx 组件用于 VertxEventBus。

vertx EventBus 发送和接收 JSON 事件。

INFO:From Camel 2.16 onwards vertx 3 is use which requires Java 1.8 at runtime.

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-vertx</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

364.1. URI 格式

```
vertx:channelName[?options]
```

364.2. 选项

Vert.x 组件支持 7 个选项，如下所列。

Name	描述	默认值	类型
vertxFactory (advanced)	使用自定义 VertxFactory 实现		VertxFactory
host (common)	创建嵌入式集群事件Bus 的主机名		字符串
port (common)	创建嵌入式集群事件Bus 的端口		int
vertxOptions (common)	用于创建 vertx 的选项		VertxOptions

Name	描述	默认值	类型
vertx (common)	使用给定的 vertx EventBus 而不是创建新的嵌入式 EventBus		Vertx
timeout (common)	等待集群 Vertx EventBus 准备好超时（以秒为单位）。默认值为 60。	60	int
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Vert.x 端点使用 URI 语法进行配置：

```
vertx:address
```

使用以下路径和查询参数：

364.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
address	**必需** 设置用于通信的事件总线地址		字符串

364.2.2. 查询参数(5 参数)：

Name	描述	默认值	类型
pubSub (common)	是否使用发布/订阅而不是在发送到 vertx 端点时指向点。		布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler

Name	描述	默认值	类型
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

364.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 8 个选项，如下所列。

Name	描述	默认值	类型
camel.component.vertx.enabled	启用 vertx 组件	true	布尔值
camel.component.vertx.host	创建嵌入式集群事件Bus 的主机名		字符串
camel.component.vertx.port	创建嵌入式集群事件Bus 的端口		整数
camel.component.vertx.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.vertx.timeout	等待集群 Vertx EventBus 准备好超时（以秒为单位）。默认值为 60。	60	整数
camel.component.vertx.vertx	使用给定的 vertx EventBus，而不是创建一个新的嵌入式 EventBus。选项是一个 io.vertx.core.Vertx 类型。		字符串
camel.component.vertx.vertx-factory	使用自定义 VertxFactory 实施。选项是一个 io.vertx.core.spi.VertxFactory 类型。		字符串
camel.component.vertx.vertx-options	用于创建 vertx 的选项。选项是一个 io.vertx.core.VertxOptions 类型。		字符串

Camel 2.12. 3：在发送到 vertx 端点时，是否使用发布/订阅而不是点到点。

> *You can append query options to the URI in the following format, ?option=value&option=value&...*

364.4. 连接到现有的 VERT.X 实例

如果要连接到 JVM 中已存在的 Vert.x 实例，您可以在组件级别上设置实例：

```
Vertx vertx = ...;
VertxComponent vertxComponent = new VertxComponent();
vertxComponent.setVertx(vertx);
camelContext.addComponent("vertx", vertxComponent);
```

364.5. 另请参阅

- *配置 Camel*
- *组件*
- *端点*
- *开始使用*

第 365 章 VM 组件

从 Camel 版本 1.1 开始提供

vm: 组件提供异步 SEDA 行为，在单独的线程池中交换 BlockingQueue 和调用消费者的消息。

此组件与 Seda 组件的不同之处在于，虚拟机支持 CamelContext 实例之间的通信 - 因此，您可以使用此机制在 Web 应用程序之间进行通信（只要 camel-core.jar 位于 system/boot classpath 上）。

VM 是 Seda 组件的扩展。

365.1. URI 格式

```
vm:queueName[?options]
```

其中 queueName 可以是任意字符串，用于唯一标识 JVM 中的端点（或在加载 camel-core.jar 的类 loader 中至少标识端点）

您可以以以下格式将查询选项附加到 URI 中： ?option=value&option=value&...

完全相同 的虚拟机 端点 URI 必须 用于生成者和消费者端点。否则，Camel 将创建一个第二个 虚拟机 端点，尽管 URI 的 queueName 部分相同。例如：

```
from("direct:foo").to("vm:bar?concurrentConsumers=5");

from("vm:bar?concurrentConsumers=5").to("file://output");
```

请注意，我们必须使用完整的 URI，包括生成者和消费者中的选项。

在 Camel 2.4 中，这个问题已被解决，只有队列名称必须匹配。使用队列名称 栏，您可以重写以前的 exmple，如下所示：

```
from("direct:foo").to("vm:bar");

from("vm:bar?concurrentConsumers=5").to("file://output");
```

365.2. 选项

VM 组件支持 6 个选项，如下所列。

Name	描述	默认值	类型
queueSize (advanced)	设置 SEDA 队列的默认最大容量（例如，它可以保存的消息数）。	1000	int
concurrentConsumers (consumer)	设置默认的并发线程处理交换数。	1	int
defaultQueueFactory (advanced)	设置默认队列工厂。		BlockingQueueFactory
defaultBlockWhenFull (producer)	将消息发送到完整 SEDA 队列的线程将阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出一个异常，表示队列已满。通过启用这个选项，调用线程将阻止并等待消息被接受。	false	布尔值
defaultOfferTimeout (producer)	将消息发送到完整 SEDA 队列的线程将阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出一个异常，表示队列已满。通过启用这个选项，可以将配置超时添加到块问题单中。使用lining java 队列的 .offer (timeout)方法		long
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

VM 端点使用 URI 语法进行配置：

```
vm:name
```

使用以下路径和查询参数：

365.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	**所需** 队列的名称		字符串

365.2.2. 查询参数(17 参数)：

Name	描述	默认值	类型
size (common)	SEDA 队列的最大容量（例如，它可以保存的消息数）。默认情况下，将使用 SEDA 组件上设置的 defaultSize。	1000	int
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外，该处理程序记录在 WARN/ERROR 级别并忽略。	false	布尔值
concurrentConsumers (consumer)	并发线程处理交换的数量。	1	int
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，该例外记录在 WARN/ERROR 级别并忽略。		ExceptionHandler
exchangePattern (consumer)	在创建交换时设置默认交换模式。		ExchangePattern
limitConcurrentConsumers (consumer)	是否将 concurrentConsumers 的数量限制为最多 500。默认情况下，如果端点配置了更大的数字，则会抛出异常。您可以通过关闭这个选项来禁用该检查。	true	布尔值
multipleConsumers (consumer)	指定是否允许多个消费者。如果启用，您可以使用 SEDA 进行 Publish-Subscribe 消息。也就是说，您可以向 SEDA 队列发送消息，并让每个消费者收到邮件的副本。启用后，应在每个消费者端点上指定此选项。	false	布尔值
pollTimeout (consumer)	轮询时使用的超时。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让使用者在关闭时更快地做出反应。	1000	int
purgeWhenStopping (consumer)	在停止 consumer/route 时是否清除任务队列。这允许更快地停止，因为队列中的任何待处理消息都会被丢弃。	false	布尔值
blockWhenFull (producer)	将消息发送到完整 SEDA 队列的线程将阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出一个异常，表示队列已满。通过启用这个选项，调用线程将阻止并等待消息被接受。	false	布尔值

Name	描述	默认值	类型
discardIfNoConsumers (producer)	当发送到没有活跃消费者的队列时，生成者是否应该丢弃消息（不要将消息添加到队列）。可以同时启用其中一个选项 discardIfNoConsumers 和 failIfNoConsumers。	false	布尔值
failIfNoConsumers (producer)	当发送到没有活跃用户的队列时，生成者是否应该通过抛出异常。可以同时启用其中一个选项 discardIfNoConsumers 和 failIfNoConsumers。	false	布尔值
offerTimeout (producer)	当队列已满时，可以使用 offerTimeout （毫秒）添加到块问题单中。您可以使用 0 或负值禁用超时。		long
timeout (producer)	SEDA 生成者停止等待异步任务完成前的超时（以毫秒为单位）。您可以使用 0 或负值禁用超时。	30000	long
waitForTaskToComplete (producer)	选项指定调用者是否应该等待 async 任务在继续前等待。支持以下三个选项： Always、Never 或 IfReplyExpected。前两个值为 self-explanatory。最后的值如果为 Request ReplyExpected，只有在消息是 Request Reply based 时等待。默认选项为 IfReplyExpected。	IfReplyExpected	WaitForTaskToComplete
队列 (advanced)	定义供端点使用的队列实例。这个选项只适用于您要使用自定义队列实例的个别用例。		BlockingQueue
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

有关适用于 Vm 组件的同一规则的选项和其他重要用法详情，请参阅 Seda 组件。

365.3. SAMPLES

在以下路由中，我们将 CamelContext 实例之间的交换发送到名为 order.email 的虚拟机队列：

```
from("direct:in").bean(MyOrderBean.class).to("vm:order.email");
```

然后，我们在其它 Camel 上下文中收到交换（比如在另一个 .war 应用程序中部署）：

```
from("vm:order.email").bean(MyOrderEmailSender.class);
```

365.4. 另请参阅

- *SEDA*

第 366 章 WEATHER COMPONENT

从 Camel 版本 2.12 开始提供

天气：组件用于从 Open Weather Map （一个提供免费全球天气和预测信息的站点）轮询天气信息。该信息返回为 json String 对象。

默认情况下，Camel 将轮询当前天天的更新，并按小时进行一次预测。它还可用于根据端点上定义的参数（用作制作者）查询 apps api。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-weather</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

366.1. URI 格式

```
weather://<unused name>[?options]
```

366.2. REMARK

从 10 月 9 日起，需要访问 openweather 服务需要一个 Api Key。此密钥作为参数传递给STORE 端点的 URI 定义，而使用 appid 参数！

366.3. 地理位置供应商

从 2018 年 7 月正式发布(FreegeoIP)不再可用。camel-weather 组件使用此 API。我们切换到 IPstack，因此您需要指定和访问密钥以及您现在使用 API 的 IP。

366.4. 选项

Weather 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
geolocationAccesskey (common)	地理位置服务现在需要一个 accessKey 才能使用		字符串
geolocationRequestHost IP (common)	地理位置服务现在需要指定与您正在使用的 accessKey 关联的 IP		字符串
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Weather 端点使用 URI 语法进行配置：

```
weather:name
```

使用以下路径和查询参数：

366.4.1. 路径参数(1 参数)：

Name	描述	默认值	类型
name	**必需**，不使用 name 值。		字符串

366.4.2. 查询参数(45 参数)：

Name	描述	默认值	类型
appId (common)	用于验证连接到 API 服务器的用户 **所需的** APPID ID		字符串
headerName (common)	要在此标头中存储 apps 结果，而不是邮件正文。如果要按原样保留当前消息正文，可以使用它。		字符串
language (common)	响应语言。	en	WeatherLanguage
mode (common)	Weather 数据的输出格式。	JSON	WeatherMode
period (common)	如果为 null，则返回当前 Weather，否则将使用 5、7、14 天的值。只有预测周期的数字值实际上会被解析，因此拼写，时间周期的大写是您（忽略）		字符串

Name	描述	默认值	类型
units (common)	温度测量的单位。		WeatherUnits
STOREApi (common)	要使用的 API （当前、预测/3 小时、每天预测、站）		WeatherApi
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
httpConnectionManager (advanced)	使用自定义 HttpConnectionManager 管理连接		HttpConnectionManager
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int

Name	描述	默认值	类型
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
cnt (filter)	要找到的结果数		整数
id (filter)	id 的城市/站列表.您可以使用逗号分隔多个 id。		字符串
lat (filter)	定位符.您可以使用 lat 和 lon 选项而不是 location。对于开箱即用的查询，这是底部模板。		字符串

Name	描述	默认值	类型
位置 (filter)	如果 null Camel 将尝试使用 IP 地址的地理位置并确定您的当前位置，否则指定了城市，country。对于已知的城市名称，Open Weather Map 将确定最佳适配，但可能会返回多个结果。因此，指定和国家/地区也会返回更准确的数据。如果您将 current 指定为位置，则组件将尝试获取当前的 latitude 和 longitude，并使用它来获取 Nagios 详情。您可以使用 lat 和 lon 选项而不是 location。		字符串
lon (filter)	位置的 Longitude。您可以使用 lat 和 lon 选项而不是 location。对于方框的查询，这是左侧的长度。		字符串
rightLon (filter)	对于方框的查询，这是正确的长度。需要与 topLat 和 zoom 结合使用。		字符串
topLat (filter)	对于开箱即用的查询，这是顶尖的 latitude。需要与 rightLon 和缩放结合使用。		字符串
zip (filter)	zip-code,如 94040,us		字符串
缩放 (filter)	对于方框的查询，这是缩放。需要与 rightLon 和 topLat 结合使用。		整数
proxyAuthDomain (proxy)	代理 NTLM 身份验证的域		字符串
proxyAuthHost (proxy)	用于代理 NTLM 身份验证的可选主机		字符串
proxyAuthMethod (proxy)	代理的身份验证方法，可以是 Basic、Digest 或 NTLM。		字符串
proxyAuthPassword (proxy)	用于代理身份验证的密码		字符串
proxyAuthUsername (proxy)	用于代理身份验证的用户名		字符串
proxyHost (proxy)	代理主机名		字符串
proxyPort (proxy)	代理端口号		整数
geolocationAccessKey (security)	**必需的** geolocation 服务现在需要一个 accessKey 才能使用		字符串
geolocationRequestHostIP (security)	**必需的** 地理位置 服务现在需要指定与您正在使用的 accessKey 关联的 IP		字符串

Name	描述	默认值	类型

366.5. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
camel.component.weather.enabled	启用天气组件	true	布尔值
camel.component.weather.geolocation-access-key	地理位置服务现在需要一个 accessKey 才能使用		字符串
camel.component.weather.geolocation-request-host-i-p	地理位置服务现在需要指定与您正在使用的 accessKey 关联的 IP		字符串
camel.component.weather.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&…

366.6. 交换数据格式

Camel 将以 json 格式的 java.lang.String 形式提供正文（请参阅上面的 mode 选项）。

366.7. 消息标头

标头	描述
CamelWeatherQuery	发送到 Open Weather Map 站点的原始查询 URL
CamelWeatherLocation	由生成者用于覆盖端点位置，并使用此标头中的位置。

366.8. SAMPLES

在此例中，我们发现了西班牙马德里的 7 天气预报：

```
from("weather:foo?location=Madrid,Spain&period=7
days&appid=APIKEY&geolocationAccessKey=IPSTACK_ACCESS_KEY&geolocationRequest
HostIP=LOCAL_IP").to("jms:queue:weather");
```

要只找到当前位置的当前天子，您可以使用：

```
from("weather:foo?
appid=APIKEY&geolocationAccessKey=IPSTACK_ACCESS_KEY&geolocationRequestHostIP
=LOCAL_IP").to("jms:queue:weather");
```

使用我们执行的生成者查找天气。

```
from("direct:start")
  .to("weather:foo?
location=Madrid,Spain&appid=APIKEY&geolocationAccessKey=IPSTACK_ACCESS_KEY&geo
locationRequestHostIP=LOCAL_IP");
```

我们可以在带有标头的消息中发送，以获取任何位置，如下所示：

```
  String json = template.requestBodyAndHeader("direct:start", "", "CamelWeatherLocation",
"Paris,France&appid=APIKEY", String.class);
```

在当前位置获取天子，然后：

```
String json = template.requestBodyAndHeader("direct:start", "", "CamelWeatherLocation",
"current&appid=APIKEY", String.class);
```

第 367 章 WEB3J ETHEREUM BLOCKCHAIN 组件

从 Camel 版本 2.22 开始提供

Ethereum 块chain 组件使用 web3j 客户端 API，并允许您与 Ethereum 兼容节点（如 Geth、 Parity、 Quorum、 Infura 等）交互。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-web3j</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

367.1. URI 格式

```
web3j://<local/remote host:port or local IPC path>[?options]
```

367.2. WEB3J 选项

Web3j Ethereum Blockchain 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	默认配置		Web3jConfiguration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Web3j Ethereum Blockchain 端点使用 URI 语法进行配置：

```
web3j:nodeAddress
```

使用以下路径和查询参数：

367.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
nodeAddress	***必需*** 设置用于进行通信的节点地址		字符串

367.2.2. 查询参数(38 参数)：

Name	描述	默认值	类型
addresses (common)	合同地址或地址列表。		list
fromAddress (common)	事务发送的地址		字符串
fromBlock (common)	块号，或最后 mined 块或待处理字符串 latest，对于尚未采集的事务，最早。	latest	DefaultBlockParameter
fullTransactionObjects (common)	如果为 true，它将返回完整的事务对象，如果只返回事务的哈希值。	false	布尔值
gasLimit (common)	此块中允许的最大 gas。		BigInteger
privateFor (common)	一个事务 privateFor 节点，在 Quorum 网络中具有公钥		list
quorumAPI (common)	如果为 true，这将支持 Quorum API。	false	布尔值
toAddress (common)	事务定向到的地址。		字符串
toBlock (common)	块号，或最后 mined 块或待处理字符串 latest，对于尚未采集的事务，最早。	latest	DefaultBlockParameter
topics (common)	主题取决于顺序。每个主题也可以是一个主题列表。指定用逗号分开的多个主题。		字符串
web3j (common)	预配置的 Web3j 对象。		Web3j

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
address (producer)	合同地址.		字符串
atBlock (producer)	块号，或最后 mined 块或待处理字符串 latest，对于尚未采集的事务，最早。	latest	DefaultBlockParameter
blockHash (producer)	此事务所在的块的哈希值。		字符串
clientId (producer)	标识客户端的随机十六进制(32 字节) ID。		字符串
data (producer)	已编译合同代码或调用方法签名和编码参数的哈希。		字符串
databaseName (producer)	本地数据库名称。		字符串
filterId (producer)	要使用的过滤器 ID。		BigInteger
gasPrice (producer)	每个付费的天然的Gas 价格。		BigInteger
hashrate (producer)	哈希速率的十六进制字符串表示(32 字节)。		字符串
headerPowHash (producer)	标头的 pow-hash (256 位)用于提交概念验证解决方案。		字符串
index (producer)	块中的事务/无限索引位置。		BigInteger
KeyName (producer)	数据库中的密钥名称。		字符串

Name	描述	默认值	类型
mixDigest (producer)	用于提交概念验证解决方案的组合摘要(256 位)。		字符串
nonce (producer)	尚未发现(64 位)用于提交概念验证解决方案。		字符串
operation (producer)	要使用的操作。	transaction	字符串
position (producer)	使用块的事务索引位置。		BigInteger
priority (producer)	whisper 消息的优先级。		BigInteger
sha3HashOfDataToSign (producer)	通过计算特定 Ethereum 特定签名的消息。		字符串
signedTransactionData (producer)	新消息调用事务的签名事务数据或为签名事务创建合同。		字符串
sourceCode (producer)	要编译的源代码。		字符串
transactionHash (producer)	有关事务哈希请求的事务的信息。		字符串
TTL (producer)	whisper 消息的时间（以秒为单位）。		BigInteger
value (producer)	在事务中发送的值。		BigInteger
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

367.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 33 选项，如下所列。

Name	描述	默认值	类型
camel.component.web3j.configuration.address	合同地址.		字符串
camel.component.web3j.configuration.addresses	合同地址或地址列表。		list
camel.component.web3j.configuration.block-hash	此事务所在的块的哈希值。		字符串
camel.component.web3j.configuration.client-id	标识客户端的随机十六进制(32 字节) ID。		字符串
camel.component.web3j.configuration.data	已编译合同代码或调用方法签名和编码参数的哈希。		字符串
camel.component.web3j.configuration.database-name	本地数据库名称。		字符串
camel.component.web3j.configuration.filter-id	要使用的过滤器 ID。		BigInteger
camel.component.web3j.configuration.from-address	事务发送的地址		字符串
camel.component.web3j.configuration.full-transaction-objects	如果为 true，它将返回完整的事务对象，如果只返回事务的哈希值。	false	布尔值
camel.component.web3j.configuration.gas-limit	此块中允许的最大 gas。		BigInteger
camel.component.web3j.configuration.gas-price	每个付费的天然的Gas 价格。		BigInteger

Name	描述	默认值	类型
camel.component.web3j.configuration.hashrate	哈希速率的十六进制字符串表示(32 字节)。		字符串
camel.component.web3j.configuration.header-pow-hash	标头的 pow-hash (256 位)用于提交概念验证解决方案。		字符串
camel.component.web3j.configuration.index	块中的事务/无限索引位置。		BigInteger
camel.component.web3j.configuration.key-name	数据库中的密钥名称。		字符串
camel.component.web3j.configuration.mix-digest	用于提交概念验证解决方案的组合摘要(256 位)。		字符串
camel.component.web3j.configuration.nonce	尚未发现(64 位)用于提交概念验证解决方案。		字符串
camel.component.web3j.configuration.operation	要使用的操作。	transaction	字符串
camel.component.web3j.configuration.position	使用块的事务索引位置。		BigInteger
camel.component.web3j.configuration.priority	whisper 消息的优先级。		BigInteger
camel.component.web3j.configuration.private-for	一个事务 privateFor 节点，在 Quorum 网络中具有公钥		list
camel.component.web3j.configuration.quorum-a-p-i	如果为 true，这将支持 Quorum API。	false	布尔值
camel.component.web3j.configuration.sha3-hash-of-data-to-sign	通过计算特定 Ethereum 特定签名的消息。		字符串

Name	描述	默认值	类型
camel.component.web3j.configuration.signed-transaction-data	新消息调用事务的签名事务数据或为签名事务创建合同。		字符串
camel.component.web3j.configuration.source-code	要编译的源代码。		字符串
camel.component.web3j.configuration.to-address	事务定向到的地址。		字符串
camel.component.web3j.configuration.topics	主题取决于顺序。每个主题也可以是一个主题列表。指定用逗号分开的多个主题。		list
camel.component.web3j.configuration.transaction-hash	有关事务哈希请求的事务的信息。		字符串
camel.component.web3j.configuration.ttl	whisper 消息的时间（以秒为单位）。		BigInteger
camel.component.web3j.configuration.value	在事务中发送的值。		BigInteger
camel.component.web3j.configuration.web3j	预配置的 Web3j 对象。		Web3j
camel.component.web3j.enabled	启用 web3j 组件	true	布尔值
camel.component.web3j.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

您可以在 URI 中附加查询选项，格式为 ?options=value&option2=value&...

367.4. 消息标头

标头	描述
所有 URI 选项	所有 URI 选项也可以设置为交换标头。

367.5. SAMPLES

侦听新的 mined 块，并将块哈希发送到 jms 队列：

```
from("web3j://http://127.0.0.1:7545?operation=ETH_BLOCK_HASH_OBSERVABLE")
    .to("jms:queue:blocks");
```

使用块哈希代码来检索块和完整的事务详情：

```
from("jms:queue:blocks")
    .setHeader(BLOCK_HASH, body())
    .to("web3j://http://127.0.0.1:7545?
operation=ETH_GET_BLOCK_BY_HASH&fullTransactionObjects=true");
```

在特定块号读取地址的平衡：

```
from("direct:start")
    .to("web3j://http://127.0.0.1:7545?
operation=ETH_GET_BALANCE&address=0xc8CDceCE5d006dAB638029EBCf6Dd666efF5A952&atBlock=10");
```

第 368 章 JETTY WEBSOCKET 组件

从 Camel 版本 2.10 开始提供

websocket 组件提供 websocket 端点，用于使用 websocket 与客户端通信。组件使用 Eclipse Jetty 服务器来实现 IETF 规范(drafts 和 RFC 6455)。它支持协议 ws:// 和 wss://。要使用 wss:// 协议，必须定义 SSLContextParameters。

目前支持的版本

Camel 2.18 使用 Jetty 9

368.1. URI 格式

```
websocket://hostname[:port][/resourceUri][?options]
```

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

368.2. WEBSOCKET 选项

Jetty Websocket 组件支持 14 个选项，如下所列。

Name	描述	默认值	类型
staticResources (consumer)	为静态资源设置资源路径（如 .html 文件等）。如果使用 classpath: 前缀，则可以从 classpath: 加载资源，否则资源是从文件系统或 JAR 文件加载的。例如，若要从 root classpath 加载，请使用 classpath:., 或 classpath:WEB-INF/static if not configured （如 null），但没有使用静态资源。		字符串
host (common)	主机名。默认值为 0.0.0.0	0.0.0.0	字符串
port (common)	端口号。默认值为 9292	9292	整数
sslKeyPassword (security)	使用 SSL 时密钥存储的密码。		字符串

Name	描述	默认值	类型
sslpassword (security)	使用 SSL 时的密码。		字符串
sslKeystore (security)	密钥存储的路径。		字符串
enableJmx (advanced)	如果这个选项为 true，则会为这个端点启用 Jetty JMX 支持。如需了解更多详细信息，请参阅 Jetty JMX 支持。	false	布尔值
MinThreads (advanced)	要为服务器线程池中最少的线程数设置值。由于切换到 Jetty9，需要 MaxThreads/minThreads 或 threadPool 字段。minThreads 的默认值为 1。		整数
maxThreads (advanced)	要为服务器线程池中最大线程数设置值。由于切换到 Jetty9，需要 MaxThreads/minThreads 或 threadPool 字段。maxThreads 的默认值为 12 noCores。		整数
threadPool (advanced)	为服务器使用自定义线程池。由于切换到 Jetty9，需要 MaxThreads/minThreads 或 threadPool 字段。		ThreadPool
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters
useGlobalSslContext 参数 (security)	启用使用全局 SSL 上下文参数。	false	布尔值
socketFactory (common)	配置包含子协议的自定义 WebSocketFactory 的映射。映射中的键是子协议。默认键是为默认实现保留的。		Map
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Jetty Websocket 端点使用 URI 语法进行配置：

```
websocket:host:port/resourceUri
```

使用以下路径和查询参数：

368.2.1. 路径参数(3 参数)：

Name	描述	默认值	类型
主机	主机名。默认值为 0.0.0.0。在组件上设置这个选项将使用组件配置的值作为默认值。	0.0.0.0	字符串
port	端口号。默认值为 9292。在组件上设置这个选项将使用组件配置的值作为默认值。	9292	整数
resourceUri	要使用的 websocket 频道的名称		字符串

368.2.2. 查询参数(18 参数)：

Name	描述	默认值	类型
maxBinaryMessageSize (common)	可用于设置 websocketServlet 创建的 websocket 在关闭前可以接受的大小（以字节为单位）。（默认为 -1 - 或无限）	-1	整数
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
sessionSupport (consumer)	是否启用会话支持，为每个 http 请求启用 HttpSession。	false	布尔值
staticResources (consumer)	为静态资源设置资源路径（如 .html 文件等）。如果使用 classpath: 前缀，则可以从 classpath: 加载资源，否则资源是从文件系统或 JAR 文件加载的。例如，若要从 root classpath 加载，请使用 classpath:., 或 classpath:WEB-INF/static if not configured （如 null），但没有使用静态资源。		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
sendTimeout (producer)	发送到 websocket 频道时的超时（以 millis 为单位）。默认超时为 30000 (30 秒)。	30000	整数

Name	描述	默认值	类型
sendToAll (producer)	发送到所有 websocket 订阅者。可用于在端点级别上配置，而不必在消息上使用 WebsocketConstants.SEND_TO_ALL 标头。		布尔值
bufferSize (advanced)	设置 websocketServlet 的缓冲区大小，它也是最大帧字节大小（默认值 8192）	8192	整数
maxIdleTime (advanced)	设置 websocketServlet 创建的 websocket 在关闭前可能会闲置的时间（默认为 300000）	300000	整数
maxTextMessageSize (advanced)	可用于在关闭前接受 websocketServlet 创建的 websocket 的大小。		整数
minVersion (advanced)	可用于设置 websocketServlet 接受的最低协议版本。（默认为 13 - RFC6455 版本）	13	整数
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
allowedOrigins (cors)	CORS 允许的源。使用 允许所有.		字符串
crossOriginFilterOn (cors)	是否启用 CORS	false	布尔值
filterPath (cors)	过滤 CORS 的上下文路径		字符串
enableJmx (monitoring)	如果这个选项为 true，则会为这个端点启用 Jetty JMX 支持。如需了解更多详细信息，请参阅 Jetty JMX 支持。	false	布尔值
sslContextParameters (security)	使用 SSLContextParameters 配置安全性		SSLContextParameters

368.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 15 个选项，如下所列。

Name	描述	默认值	类型
camel.component.websocket.enable-jmx	如果这个选项为 true，则会为这个端点启用 Jetty JMX 支持。如需了解更多详细信息，请参阅 Jetty JMX 支持。	false	布尔值
camel.component.websocket.enabled	启用 websocket 组件	true	布尔值
camel.component.websocket.host	主机名。默认值为 0.0.0.0	0.0.0.0	字符串
camel.component.websocket.max-threads	要为服务器线程池中最大线程数设置值。由于切换到 Jetty9，需要 MaxThreads/minThreads 或 threadPool 字段。maxThreads 的默认值为 1 2 noCores。		整数
camel.component.websocket.min-threads	要为服务器线程池中最少的线程数设置值。由于切换到 Jetty9，需要 MaxThreads/minThreads 或 threadPool 字段。minThreads 的默认值为 1。		整数
camel.component.websocket.port	端口号。默认值为 9292	9292	整数
camel.component.websocket.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.websocket.socket-factory	配置包含子协议的自定义 WebSocketFactory 的映射。映射中的键是子协议。默认键是为默认实现保留的。		Map
camel.component.websocket.ssl-context-parameters	使用 SSLContextParameters 配置安全性。选项是 org.apache.camel.util.jsse.SSLContextParameters 类型。		字符串
camel.component.websocket.ssl-key-password	使用 SSL 时密钥存储的密码。		字符串
camel.component.websocket.ssl-keystore	密钥存储的路径。		字符串
camel.component.websocket.ssl-password	使用 SSL 时的密码。		字符串

Name	描述	默认值	类型
camel.component.websocket.static-resources	为静态资源设置资源路径（如 .html 文件等）。如果使用 classpath: 前缀，则可以从 classpath: 加载资源，否则资源是从文件系统或 JAR 文件加载的。例如，若要从 root classpath 加载，请使用 classpath:., 或 classpath:WEB-INF/static if not configured （如 null），但没有使用静态资源。		字符串
camel.component.websocket.thread-pool	为服务器使用自定义线程池。由于切换到 Jetty9，需要 MaxThreads/minThreads 或 threadPool 字段。选项是 org.eclipse.jetty.util.thread.ThreadPool 类型。		字符串
camel.component.websocket.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

368.4. 消息标头

websocket 组件使用 2 标头来指示将消息发回到单个/当前客户端或所有客户端。

WebsocketConstants.SEND_TO_ALL	将消息发送到当前连接的所有客户端。您可以在端点上使用 sendToAll 选项，而不使用此标头。
WebsocketConstants.CONNECTION_KEY	使用给定连接密钥向客户端发送消息。
WebsocketConstants.REMOTE_ADDRESS	websocket 会话的远程地址。

368.5. 使用方法

在这个示例中，我们让 Camel 会公开一个客户端可以与之通信的 websocket 服务器。websocket 服务器使用默认主机和端口，即 0.0.0.0:9292。
这个示例将发回输入的 echo。要发回消息，我们需要将转换的消息发送到同一端点 "websocket://echo"。这是需要的
，因为默认情况下消息传递是 InOnly。

这个示例是单元测试的一部分，您可以在其中找到。作为客户端，我们使用 AHC 库，它也支持 Web 套接字。

下面是一个示例，其中定义了 webapp 资源位置，以允许 Jetty Application Server 不注册 WebSocket servlet，同时还为浏览器公开 Web 资源。资源应在 webapp 目录下定义。

```
from("activemq:topic:newsTopic")
  .routeId("fromJMStoWebSocket")
  .to("websocket://localhost:8443/newsTopic?
sendToAll=true&staticResources=classpath:webapp");
```

368.6. 为 WEBSOCKET 组件设置 SSL

368.6.1. 使用 JSSE 配置实用程序

从 Camel 2.10 开始， WebSocket 组件通过 Camel JSSE 配置实用程序支持 SSL/TLS 配置。这个实用程序可大大减少您需要写入的组件特定代码量，并在端点和组件级别进行配置。以下示例演示了如何将实用程序与 Cometd 组件一起使用。

组件的编程配置

```
KeyStoreParameters ksp = new KeyStoreParameters();
ksp.setResource("/users/home/server/keystore.jks");
ksp.setPassword("keystorePassword");

KeyManagersParameters kmp = new KeyManagersParameters();
kmp.setKeyStore(ksp);
kmp.setKeyPassword("keyPassword");

TrustManagersParameters tmp = new TrustManagersParameters();
tmp.setKeyStore(ksp);

SSLContextParameters scp = new SSLContextParameters();
scp.setKeyManagers(kmp);
scp.setTrustManagers(tmp);
```

```
CometdComponent commetdComponent = getContext().getComponent("cometds",
CometdComponent.class);
commetdComponent.setSslContextParameters(scp);
```

基于 Spring DSL 的端点配置

```
...
  <camel:sslContextParameters
     id="sslContextParameters">
    <camel:keyManagers
       keyPassword="keyPassword">
      <camel:keyStore
         resource="/users/home/server/keystore.jks"
         password="keystorePassword"/>
    </camel:keyManagers>
    <camel:trustManagers>
      <camel:keyStore
         resource="/users/home/server/keystore.jks"
         password="keystorePassword"/>
    </camel:trustManagers>
  </camel:sslContextParameters>...
...
  <to uri="websocket://127.0.0.1:8443/test?sslContextParameters=#sslContextParameters"/>...
```

基于 Java DSL 端点配置

```
...
    protected RouteBuilder createRouteBuilder() throws Exception {
        return new RouteBuilder() {
            public void configure() {

                String uri = "websocket://127.0.0.1:8443/test?
sslContextParameters=#sslContextParameters";

                from(uri)
                    .log(">>> Message received from WebSocket Client : ${body}")
                    .to("mock:client")
                    .loop(10)
                        .setBody().constant(">> Welcome on board!")
                        .to(uri);
...
```

368.7. 另请参阅

- 配置 Camel

- *组件*
- *端点*
- *开始使用*
- *AHC*
- *Jetty*
- *Little Websocket 示例演示了如何实时轮询使用 Web 套接字搜索和发布结果的常量源。*

第 369 章 WEKA 组件

Since Camel 3.1

仅支持生成者

Weka 组件提供对 (Weka Data Mining) 工具集的访问。

Weka 是尝试和测试的开源机器学习软件，可以通过图形用户界面、标准终端应用程序或 Java API 进行访问。它被广泛用于指导、研究和工业应用程序，包含用于标准机器学习任务的内置工具的 plethora，同时还提供对已知的 toolboxes （如 scikit-learn、R 和 Deep learning4j）的透明访问。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-weka</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

369.1. URI 格式

```
weka://cmd
```

369.2. 选项

Weka 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
lazyStartProducer (producer)	生成者是否应懒惰启动（在第一个消息中）。通过懒惰启动，您可以使用此选项来允许 CamelContext 和路由在生成者启动期间启动，并导致路由启动失败。通过懒惰启动，启动失败可以在路由信息时通过 Camel 的路由错误处理程序进行处理。请注意，在处理第一个消息时，创建并启动生成者可能需要稍等时间，并延长处理的总处理时间。	false	布尔值

Name	描述	默认值	类型
basicPropertyBinding (advanced)	组件是否应该使用基本属性绑定(Camel 2.x)或较新的属性绑定	false	布尔值

Weka 端点使用 URI 语法进行配置：

```
weka:command
```

使用以下路径和查询参数：

369.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
命令	**必需** 的命令。该值可以是： filter、model、read、write、push、pop、version 之一		命令

369.2.2. 查询参数(12 参数)：

Name	描述	默认值	类型
lazyStartProducer (producer)	生成者是否应懒惰启动 （在第一个消息中）。通过懒惰启动，您可以使用此选项来允许 CamelContext 和路由在生成者启动期间启动，并导致路由启动失败。通过懒惰启动，启动失败可以在路由信息时通过 Camel 的路由错误处理程序进行处理。请注意，在处理第一个消息时，创建并启动生成者可能需要稍等时间，并延长处理的总处理时间。	false	布尔值
basicPropertyBinding (advanced)	端点是否应该使用基本属性绑定(Camel 2.x)或较新的属性绑定	false	布尔值
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
apply (filter)	过滤器 spec （如名称选项）		字符串
build (model)	分类器规格（如名称选项）		字符串
dsname (model)	用于培训分类器的指定数据集		字符串

Name	描述	默认值	类型
折叠 （型号）	用于跨验证的折叠数量	10	int
loadFrom (model)	加载模型的路径		字符串
saveTo (model)	保存模型的路径		字符串
seed (model)	可选的用于随机工具的 seed	1	int
xval (model)	是否在当前数据集中使用跨验证的标志	false	布尔值
路径 （写入）	读/写命令的 in/out 路径		字符串

369.3. KARAF 支持

Karaf 不支持此组件

369.4. 消息标头

369.5. SAMPLES

369.5.1. Read + Filter + Write

第一个示例演示了如何使用文件组件读取 CSV 文件，然后将其传递到 Weka。在 Weka 中，我们对数据集应用几个过滤器，然后将它传递给文件组件以进行编写。

```
@Override
public void configure() throws Exception {

    // Use the file component to read the CSV file
    from("file:src/test/resources/data?fileName=sfny.csv")

    // Convert the 'in_sf' attribute to nominal
    .to("weka:filter?apply=NumericToNominal -R first")

    // Move the 'in_sf' attribute to the end
    .to("weka:filter?apply=Reorder -R 2-last,1")

    // Rename the relation
    .to("weka:filter?apply=RenameRelation -modify sfny")

    // Use the file component to write the Arff file
    .to("file:target/data?fileName=sfny.arff")
}
```

-

在这里，我们在不使用文件组件的情况下执行以上操作。

```
@Override
public void configure() throws Exception {

    // Initiate the route from somewhere
    .from("...")

    // Use Weka to read the CSV file
    .to("weka:read?path=src/test/resources/data/sfny.csv")

    // Convert the 'in_sf' attribute to nominal
    .to("weka:filter?apply=NumericToNominal -R first")

    // Move the 'in_sf' attribute to the end
    .to("weka:filter?apply=Reorder -R 2-last,1")

    // Rename the relation
    .to("weka:filter?apply=RenameRelation -modify sfny")

    // Use Weka to write the Arff file
    .to("weka:write?path=target/data/sfny.arff");
}
```

在本例中，客户端会提供输入路径或其它支持的类型。查看用于一组支持的输入类型的 WekaTypeConverters。

```
@Override
public void configure() throws Exception {

    // Initiate the route from somewhere
    .from("...")

    // Convert the 'in_sf' attribute to nominal
    .to("weka:filter?apply=NumericToNominal -R first")

    // Move the 'in_sf' attribute to the end
    .to("weka:filter?apply=Reorder -R 2-last,1")

    // Rename the relation
    .to("weka:filter?apply=RenameRelation -modify sfny")

    // Use Weka to write the Arff file
    .to("weka:write?path=target/data/sfny.arff");
}
```

369.5.2. 构建模型

在构建模型时，我们首先选择要使用的分类算法，然后将其与某些数据进行培训。结果是经过培训的

模式，稍后我们用来对不可预见的数据进行分类。

在这里，我们培训了 10 倍的跨验证方式的 J48。

```
try (CamelContext camelctx = new DefaultCamelContext()) {

    camelctx.addRoutes(new RouteBuilder() {

        @Override
        public void configure() throws Exception {

            // Use the file component to read the training data
            from("file:src/test/resources/data?fileName=sfny-train.arff")

            // Build a J48 classifier using cross-validation with 10 folds
            .to("weka:model?build=J48&xval=true&folds=10&seed=1")

            // Persist the J48 model
            .to("weka:model?saveTo=src/test/resources/data/sfny-j48.model")
        }
    });
    camelctx.start();
}
```

369.5.3. 预测类

在这里，我们使用 处理器访问不是直接从端点 URI 提供的功能。

如果您直接出现此语法，则可能需要查看有关 Nessus API 概念 的部分。

```
try (CamelContext camelctx = new DefaultCamelContext()) {

    camelctx.addRoutes(new RouteBuilder() {

        @Override
        public void configure() throws Exception {

            // Use the file component to read the test data
            from("file:src/test/resources/data?fileName=sfny-test.arff")

            // Remove the class attribute
            .to("weka:filter?apply=Remove -R last")

            // Add the 'prediction' placeholder attribute
            .to("weka:filter?apply=Add -N predicted -T NOM -L 0,1")

            // Rename the relation
            .to("weka:filter?apply=RenameRelation -modify sfny-predicted")
```

```
            // Load an already existing model
            .to("weka:model?loadFrom=src/test/resources/data/sfny-j48.model")

            // Use a processor to do the prediction
            .process(new Processor() {
                public void process(Exchange exchange) throws Exception {
                    Dataset dataset = exchange.getMessage().getBody(Dataset.class);
                    dataset.applyToInstances(new NominalPredictor());
                }
            })

            // Write the data file
            .to("weka:write?path=src/test/resources/data/sfny-predicted.arff")
        }
    });
    camelctx.start();
}
```

369.6. RESOURCES

- *实用机器学习工具和技术*
- *机器学习课程*
- *Weka 文档*
- *Nessus-Weka*

第 370 章 WORDPRESS 组件

从 Camel 版本 2.21 开始提供

Wordpress API 的 Camel 组件.

目前只支持 Posts 和 Users 操作。

370.1. 选项

Wordpress 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	wordpress 组件配置		WordpressComponent 配置
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Wordpress 端点使用 URI 语法进行配置：

```
wordpress:operationDetail
```

使用以下路径和查询参数：

370.1.1. 路径参数(2 参数)：

Name	描述	默认值	类型
operation	**必需** 端点操作。		字符串
operationDetail	端点操作的第二部分。仅在端点语义不够时才需要，如 wordpress:post:delete		字符串

370.1.2. 查询参数(11 参数)：

Name	描述	默认值	类型
apiVersion (common)	Wordpress REST API 版本	2	字符串
criteria (common)	与复杂搜索一起使用的条件。		Map
force (common)	是否绕过垃圾箱并强制删除。	false	布尔值
id (common)	实体 ID		整数
password (common)	来自授权用户的密码		字符串
url (common)	**需要** 您的站点中的 Wordpress API URL，例如 http://myblog.com/wp-json/		字符串
user (common)	授权用户执行写入操作		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

370.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
camel.component.wordpress.configuration.api-version	Wordpress REST API 版本	2	字符串
camel.component.wordpress.configuration.criteria-properties			Map
camel.component.wordpress.configuration.force	是否绕过垃圾箱并强制删除。	false	布尔值
camel.component.wordpress.configuration.id	实体 ID		整数
camel.component.wordpress.configuration.password	来自授权用户的密码		字符串
camel.component.wordpress.configuration.search-criteria			SearchCriteria
camel.component.wordpress.configuration.url	您的站点的 Wordpress API URL，例如 http://myblog.com/wp-json/		字符串
camel.component.wordpress.configuration.user	授权用户执行写入操作		字符串
camel.component.wordpress.enabled	是否启用 wordpress 组件的自动配置。这默认是启用的。		布尔值
camel.component.wordpress.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

从官方 API 执行读取操作镜像时所需的大多数参数。在执行搜索操作时，需要 conditions. 后缀。使用以下 Consumer 作为示例：

```
wordpress:post?criteria.perPage=10&criteria.orderBy=author&criteria.categories=camel,dozer,json
```

370.2.1. 配置 Wordpress 组件

WordpressConfiguration 类可用于设置初始属性配置到组件，而不是将其作为查询参数传递。以下列表演示了如何设置要在路由中使用的组件。

```
public void configure() {
    final WordpressConfiguration configuration = new WordpressConfiguration();
    final WordpressComponentConfiguration component = new
WordpressComponentConfiguration();
    configuration.setApiVersion("2");
    configuration.setUrl("http://yoursite.com/wp-json/");
    component.setConfiguration(configuration);
    getContext().addComponent("wordpress", component);

    from("wordpress:post?id=1")
      .to("mock:result");
}
```

370.2.2. 消费者示例

消费者从 API 轮询到 Wordpress 中的时间域对象。以下是使用 Post 操作的示例：

- wordpress:post 检索后（默认为 10 后）
- wordpress:post?id=1 搜索特定的 post

370.2.3. 生成者示例

生产者在 Wordpress 上执行写入操作，如添加新用户或更新 post。要能够写入，您必须具有授权用户凭证（请参阅身份验证）。

- wordpress:post 从消息正文中的 org.apache.camel.component.wordpress.api.model.Post 类创建一个新的 post。
- wordpress:post?id=1 根据消息正文中的数据 org.apache.camel.component.wordpress.api.model.Post 更新 post。

- *wordpress:post:delete?id=1 删除特定的 post*

370.3. 身份验证

执行写入操作的生成者（如创建新后） 必须具有经过身份验证的用户 才能这样做。Wordpress 使用的标准身份验证机制是 cookie。不幸的是，在 Wordpress 环境之外不支持这个方法，因为它依赖于 非内部功能。

有些替代方案可以在不使用非ce 的情况下使用 Wordpress API，但需要特定的插件安装。

目前，camel-wordpress 只支持 Basic Authentication （出现更多内容）。要配置它，您必须安装 Basic-Auth Wordpress 插件，并将凭证传递给端点：

```
from ("direct:deletePost").to ("wordpress:post:delete?
id=9&user=ben&password=password123").to ("mock:resultDelete");
```

不建议在没有 TLS 的情况下在生产环境中使用基本身份验证！

第 371 章 XCHANGE 组件

从 Camel 版本 2.21 开始提供

xchange: 组件使用 XChange Java 库来提供对 60+ Bitcoin 和 Altcoin 交换的访问。它提供了一个一致的界面，用于交易和访问市场数据。

Camel 可以获取加密货币市场数据、查询历史数据、放置市场订单等。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-xchange</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

371.1. URI 格式

```
xchange://exchange?options
```

371.2. 选项

XChange 组件没有选项。

XChange 端点使用 URI 语法进行配置：

```
xchange:name
```

使用以下路径和查询参数：

371.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型

Name	描述	默认值	类型
name	**所需的** 交换连接到		字符串

371.2.2. 查询参数(5 参数)：

Name	描述	默认值	类型
currency (producer)	currency		currency
currencyPair (producer)	currency 对		CurrencyPair
method (producer)	**需要执行** 的方法		XChangeMethod
service (producer)	**需要** 调用的服务		XChangeService
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

371.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.xchange.enabled	是否启用 xchange 组件的自动配置。这默认是启用的。		布尔值
camel.component.xchange.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

371.4. 身份验证

此组件通过 REST API 与受支持的加密货币通信。有些 API 请求使用简单的未经身份验证的 GET 请求。然而，对于大多数有趣的操作，您需要一个交换帐户并启用了 API 访问密钥。

这些 API 访问密钥需要严格保护，特别是当它们也允许撤回功能时。在这种情况下，可以拥有您的

API 密钥的任何人都可以轻松地将费用从您的帐户转移到其他地址，例如：窃取您的费用。

您的 API 访问密钥可以在 SSH 目录中的交换特定属性文件中选择。例如，例如：~/.ssh/binance-secret.keys

```
##
# This file MUST NEVER be commited to source control.
# It is therefore added to .gitignore.
#
apiKey = GuRW0*********
secretKey = nKLki************
```

371.5. 消息标头

<TODO><title>Samples</title>

在这个示例中，我们在 USDT 中找到当前的 Bitcoin 市场价格：

```
from("direct:ticker").to("xchange:binance?
service=market&method=ticker&currencyPair=BTC/USDT")
```

</TODO>

第 372 章 XML BEANS DATAFORMAT （已弃用）

从 Camel 版本 1.2 开始提供

XmlBeans 是一种 Data Format，它使用 XmlBeans 库 将 XML 有效负载 unmarshal 到 Java 对象，或将 Java 对象嵌套到 XML 有效负载中。

```
from("activemq:My.Queue").
  unmarshal().xmlBeans().
  to("mqseries:Another.Queue");
```

372.1. 选项

XML Beans dataformat 支持 2 个选项，如下所列。

Name	默认值	Java 类型	描述
prettyPrint	false	布尔值	要启用用户的打印输出，请执行以下操作：默认为 false。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

372.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.xmlbeans.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.xmlbeans.enabled	启用 xmlbeans dataformat	true	布尔值

Name	描述	默认值	类型
camel.dataformat.xmlbeans.pretty-print	要启用用户的打印输出，请执行以下操作：默认为 false。	false	布尔值

372.3. 依赖项

要在 camel 路由中使用 XmlBeans，您需要添加对实现此数据格式的 camel-xmlbeans 的依赖关系。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-xmlbeans</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

第 373 章 XML JSON DATAFORMAT （已弃用）

从 Camel 版本 2.10 开始提供

Camel 已支持多种数据格式来执行 XML 和 JSON 相关的转换，但它们都需要一个 POJO 作为输入（用于 marshalling）或生成 POJO 作为输出（用于 unmarshalling）。此数据格式提供从 XML 转换为 JSON 的功能，反之亦然，而无需逐步完成中间 POJO。

此数据格式利用 Json-lib 库来实现直接转换。在这种情况下，XML 被视为高级别格式，而 JSON 是低级格式。因此，marshal/unmarshal 语义会分配如下：

- Marshalling plugging 从 XML 转换为 JSON
- unmarshalling TOKEN 从 JSON 转换为 XML。

373.1. 选项

XML JSon dataformat 支持 13 个选项，如下所列。

Name	默认值	Java 类型	描述
编码		字符串	设置编码。用于 unmarshalling (JSON 到 XML 转换)。
elementName		字符串	指定代表每个数组元素的 XML 元素的名称。用于 unmarshalling (JSON 到 XML 转换)。
arrayName		字符串	指定顶层 XML 元素的名称。用于 unmarshalling (JSON 到 XML 转换)。例如，在转换 1, 2, 3 时，它默认是 123 的输出。通过设置这个选项或 rootName，您可以更改元素"a"的名称。
forceTopLevelObject	false	布尔值	确定生成的 JSON 是否以与 XML root 元素匹配的最顶层元素开始。用于 marshalling (XML 到 JSon 转换)。如果禁用，XML 字符串 12 会变为 'x: '1', 'y': '2'。否则，它会变为 'a': 'x: '1', 'y': '2'。
namespaceLenient	false	布尔值	可以接受不完整的命名空间前缀的标记。用于 unmarshalling (JSON 到 XML 转换)。在大多数情况下，json-lib 会在运行时自动更改此标志以匹配处理。

Name	默认值	Java 类型	描述
rootName		字符串	指定顶层元素的名称。用于 unmarshalling (JSON 到 XML 转换)。如果没有设置，则 json-lib 将使用 arrayName 或 objectName （默认值：'o'，在当前时间无法以这个数据格式配置）。如果设置为 'root'，则 JSON 字符串 'x': 'value1', 'y' : 'value2' 将转换为 value1value2，否则 'root' 元素将命名为 'o'。
skipWhitespace	false	布尔值	确定 XML 元素之间的空格是否被视为文本值或忽略。用于 marshalling (XML 到 JSon 转换)。
trimSpaces	false	布尔值	确定在 String 值中是否省略前导和尾随空格。用于 marshalling (XML 到 JSon 转换)。
skipNamespaces	false	布尔值	指示是否应忽略命名空间。默认情况下，它们将使用 xmlns 元素添加到 JSON 输出中。用于 marshalling (XML 到 JSon 转换)。
removeNamespacePrefixes	false	布尔值	从 XML 限定元素中删除命名空间前缀，以便生成的 JSON 字符串不包含它们。用于 marshalling (XML 到 JSon 转换)。
expandableProperties		list	使用可扩展的属性，JSON 数组元素转换为 XML，作为类似 JSON 键的本地名称的重复 XML 元素序列，例如： number: 1,2,3，通常转换为 123 （可通过设置 elementName 修改），如果数字设置为用于 unmarshalling (JSON 到 XML 转换)的可扩展属性，则转换为 123。
typeHints		字符串	为生成的 XML 添加类型提示，以帮助转换回 JSON。用于 unmarshalling (JSON 到 XML 转换)。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

373.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 14 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.xmljson.array-name	指定顶层 XML 元素的名称。用于 unmarshalling (JSON 到 XML 转换)。例如，在转换 1, 2, 3 时，它默认是 123 的输出。通过设置这个选项或 rootName，您可以更改元素"a"的名称。		字符串

Name	描述	默认值	类型
camel.dataformat.xmljson.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.xmljson.element-name	指定代表每个数组元素的 XML 元素的名称。用于 unmarshalling (JSON 到 XML 转换)。		字符串
camel.dataformat.xmljson.enabled	启用 xmljson dataformat	true	布尔值
camel.dataformat.xmljson.encoding	设置编码。用于 unmarshalling (JSON 到 XML 转换)。		字符串
camel.dataformat.xmljson.expandable-properties	使用可扩展的属性，JSON 数组元素转换为 XML，作为类似 JSON 键的本地名称的重复 XML 元素序列，例如： number: 1,2,3，通常转换为 123 （可通过设置 elementName 修改），如果数字设置为用于 unmarshalling (JSON 到 XML 转换)的可扩展属性，则转换为 123。		list
camel.dataformat.xmljson.force-top-level-object	确定生成的 JSON 是否以与 XML root 元素匹配的最顶层元素开始。用于 marshalling (XML 到 JSon 转换)。如果禁用，XML 字符串 12 会变为 'x: '1', 'y': '2'。否则，它会变为 'a': 'x: '1', 'y': '2'。	false	布尔值
camel.dataformat.xmljson.namespace-lenient	可以接受不完整的命名空间前缀的标记。用于 unmarshalling (JSON 到 XML 转换)。在大多数情况下，json-lib 会在运行时自动更改此标志以匹配处理。	false	布尔值
camel.dataformat.xmljson.remove-namespace-prefixes	从 XML 限定元素中删除命名空间前缀，以便生成的 JSON 字符串不包含它们。用于 marshalling (XML 到 JSon 转换)。	false	布尔值
camel.dataformat.xmljson.root-name	指定顶层元素的名称。用于 unmarshalling (JSON 到 XML 转换)。如果没有设置，则 json-lib 将使用 arrayName 或 objectName （默认值：'o'，在当前时间无法以这个数据格式配置）。如果设置为 'root'，则 JSON 字符串 'x': 'value1', 'y' : 'value2' 将转换为 value1value2，否则 'root' 元素将命名为 'o'。		字符串
camel.dataformat.xmljson.skip-namespaces	指示是否应忽略命名空间。默认情况下，它们将使用 xmlns 元素添加到 JSON 输出中。用于 marshalling (XML 到 JSon 转换)。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.xmljson.skip-whitespace	确定 XML 元素之间的空格是否被视为文本值或忽略。用于 marshalling (XML 到 JSon 转换)。	false	布尔值
camel.dataformat.xmljson.trim-spaces	确定在 String 值中是否省略前导和尾随空格。用于 marshalling (XML 到 JSon 转换)。	false	布尔值
camel.dataformat.xmljson.type-hints	为生成的 XML 添加类型提示，以帮助转换回 JSON。用于 unmarshalling (JSON 到 XML 转换)。		字符串

ND

373.3. JAVA DSL 的基本用法

373.3.1. 显式实例化数据格式

只需从软件包 org.apache.camel.dataformat.xmljson 实例化 XmlJsonDataFormat。确保您已安装了 camel-xmljson 功能（如果在 OSGi 上运行），或者您已包含 camel-xmljson-7.13.jar 及其在 classpath 中的传输依赖项。使用默认配置初始化示例：

```
XmlJsonDataFormat xmlJsonFormat = new XmlJsonDataFormat();
```

要根据以上选项调整数据格式的行为，请使用适当的集合：

```
XmlJsonDataFormat xmlJsonFormat = new XmlJsonDataFormat();
xmlJsonFormat.setEncoding("UTF-8");
xmlJsonFormat.setForceTopLevelObject(true);
xmlJsonFormat.setTrimSpaces(true);
xmlJsonFormat.setRootName("newRoot");
xmlJsonFormat.setSkipNamespaces(true);
xmlJsonFormat.setRemoveNamespacePrefixes(true);
xmlJsonFormat.setExpandableProperties(Arrays.asList("d", "e"));
```

数据格式实例化后，下一步是从 marshal （） 和 unmarshal （） DSL 元素中实际使用它：

```
// from XML to JSON
from("direct:marshal").marshal(xmlJsonFormat).to("mock:json");
// from JSON to XML
from("direct:unmarshal").unmarshal(xmlJsonFormat).to("mock:xml");
```

373.3.2. 在命令行中定义数据格式

或者，您可以使用 xmljson （） DSL 元素定义内联数据格式：

```
// from XML to JSON - inline dataformat
from("direct:marshalInline").marshal().xmljson().to("mock:jsonInline");
// from JSON to XML - inline dataformat
from("direct:unmarshalInline").unmarshal().xmljson().to("mock:xmlInline");
```

如果需要，您也可以将 Map<String, String> 传递给内联方法，以提供自定义选项：

```
Map<String, String> xmlJsonOptions = new HashMap<String, String>();
xmlJsonOptions.put(org.apache.camel.model.dataformat.XmlJsonDataFormat.ENCODING,
"UTF-8");
xmlJsonOptions.put(org.apache.camel.model.dataformat.XmlJsonDataFormat.ROOT_NAME,
"newRoot");
xmlJsonOptions.put(org.apache.camel.model.dataformat.XmlJsonDataFormat.SKIP_NAMESP
ACES, "true");
xmlJsonOptions.put(org.apache.camel.model.dataformat.XmlJsonDataFormat.REMOVE_NAM
ESPACE_PREFIXES, "true");
xmlJsonOptions.put(org.apache.camel.model.dataformat.XmlJsonDataFormat.EXPANDABLE_
PROPERTIES, "d e");

// from XML to JSON - inline dataformat w/ options
from("direct:marshalInlineOptions").marshal().xmljson(xmlJsonOptions).to("mock:jsonInlineO
ptions");
// form JSON to XML - inline dataformat w/ options
from("direct:unmarshalInlineOptions").unmarshal().xmljson(xmlJsonOptions).to("mock:xmlInl
ineOptions");
```

373.4. SPRING 或 BLUEPRINT DSL 的基本用法

在 &lt ;dataFormats& gt; 块中，只需配置带有唯一 ID 的 xmljson 元素：

```
<dataFormats>
    <xmljson id="xmljson"/>
    <xmljson id="xmljsonWithOptions" forceTopLevelObject="true" trimSpaces="true"
rootName="newRoot" skipNamespaces="true"
            removeNamespacePrefixes="true" expandableProperties="d e"/>
</dataFormats>
```

然后，您只需引用 < marshal /> 和 < unmarshal /> DSLs 中的数据格式对象：

```
<route>
    <from uri="direct:marshal"/>
    <marshal ref="xmljson"/>
```

```
    <to uri="mock:json" />
</route>

<route>
    <from uri="direct:unmarshalWithOptions"/>
    <unmarshal ref="xmljsonWithOptions"/>
    <to uri="mock:xmlWithOptions"/>
</route>
```

为此组件启用 XML DSL 自动完成很简单：仅参考相应的 架构位置，具体取决于您使用的是 Spring 还是 Blueprint DSL。请记住，这个数据格式可从 Camel 2.10 开始获得，因此只有该版本的模式会包括这些新的 XML 元素和属性。

带有 Blueprint 的语法与 Spring DSL 的语法相同。只需确保使用正确的命名空间和 schemaLocations。

373.5. 命名空间映射

XML 具有完全限定元素和属性的命名空间；JSON 不.执行 XML-JSON 转换时，您需要考虑到这一点。

为缩小差距，Json-lib 有一个选项来绑定命名空间声明，格式为 prefixs 和 namespace URIs 到 XML 输出元素，同时 unmarshalling （例如从 JSON 转换为 XML）。例如，提供以下 JSON 字符串：

```
{ "pref1:a": "value1", "pref2:b": "value2" }
```

您可以要求 Json-lib 在元素 pref1:a 和 pref2:b 上输出命名空间声明，将前缀 pref1 和 pref2 绑定到特定的命名空间 URI。

若要使用此功能，只需创建 XmlJsonDataFormat.NamespacesPerElementMapping 对象，并将它们添加到 namespaceMappings 选项（这是 List）。

XmlJsonDataFormat.NamespacesPerElementMapping 包含元素名称和映射 [prefix adjust namespace URI]。为了便于映射多个前缀和命名空间 URI，NamespacesPerElementMapping (String 元素, String pipeSeparatedMappings) 构造器采用基于 String 的 pipe- separated 序列 [prefix, namespaceURI] 对：
|ns2|http://camel.apache.org/personalData|ns3|http://camel.apache.org/personalData2|。

要定义 default 命名空间，只需将对应的 key 字段留空：
|ns1|http://camel.apache.org/test1||http://camel.apache.org/default|。

将命名空间声明绑定到元素 name = 空字符串会将这些命名空间附加到 root 元素。

完整代码类似如下：

```
XmlJsonDataFormat namespacesFormat = new XmlJsonDataFormat();
List<XmlJsonDataFormat.NamespacesPerElementMapping> namespaces = new
ArrayList<XmlJsonDataFormat.NamespacesPerElementMapping>();
namespaces.add(new XmlJsonDataFormat.
                NamespacesPerElementMapping("",
"|ns1|http://camel.apache.org/test1||http://camel.apache.org/default|"));
namespaces.add(new XmlJsonDataFormat.
                NamespacesPerElementMapping("surname",
"|ns2|http://camel.apache.org/personalData|" +
                    "ns3|http://camel.apache.org/personalData2|"));
namespacesFormat.setNamespaceMappings(namespaces);
namespacesFormat.setRootElement("person");
```

您可以在 Spring DSL 中实现相同的操作。

373.5.1. Example

在以下 JSON 字符串中使用以上 Java 片断中的命名空间绑定：

```
{ "name": "Raul", "surname": "Kripalani", "f": true, "g": null}
```

会产生以下 XML：

```
<person xmlns="http://camel.apache.org/default" xmlns:ns1="http://camel.apache.org/test1">
   <f>true</f>
   <g null="true"/>
   <name>Raul</name>
   <surname xmlns:ns2="http://camel.apache.org/personalData"
xmlns:ns3="http://camel.apache.org/personalData2">Kripalani</surname>
</person>
```

请记住，JSON 规格定义了 JSON 对象，如下所示：

> 对象是一组未排序的名称/值对。[...]。

正因如此，元素在输出 XML 中以不同顺序排列。

373.6. 依赖项

要在 camel 路由中使用 XmlJson dataformat，您需要将以下依赖项添加到 pom 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-xmljson</artifactId>
  <version>x.x.x</version>
  <!-- Use the same version as camel-core, but remember that this component is only available
from 2.10 onwards -->
</dependency>

<!-- And also XOM must be included. XOM cannot be included by default due to an
incompatible
license with ASF; so add this manually -->
<dependency>
  <groupId>xom</groupId>
  <artifactId>xom</artifactId>
  <version>1.2.5</version>
</dependency>
```

373.7. 另请参阅

- 数据格式
- json-lib

第 374 章 XML 安全组件

从 Camel 版本 2.12 开始提供

使用这个 Apache Camel 组件，您可以生成并验证 XML 签名，如 W3C 标准 XML 签名语法和处理 中所述，或者在后续版本 1.1 中所述。有关 XML 加密支持，请参阅 XML 安全 数据格式。

您可以在此处找到 XML 签名简介。http://www.oracle.com/technetwork/articles/javase/dig-signatures-141823.html组件的实施基于 JSR 105，即与 W3C 标准对应的 Java API，并支持 Apache Santuario 和 JSR 105 的 JDK 提供程序。该实施将首先尝试使用 Apache Santuario 供应商；如果它找不到 Santuario 供应商，它将使用 JDK 供应商。此外，实施基于 DOM。

自 Camel 2.15.0 起，我们还为签名人端点提供对 XAdES-BES/EPES 的支持；请参见"签名端点的 XAdES-BES/EPES"部分。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-xmlsecurity</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

374.1. XML 签名表示模式

XML 签名在信封、信封和分离的 XML 签名之间有所不同。在 信封 的 XML 签名案例中，XML 签名由签名的 XML 文档打包；这意味着 XML 签名元素是父元素的一个子元素，属于签名的 XML 文档。在 信封 XML 签名案例中，XML 签名包含签名内容。所有其他情况都称为 分离的 XML 签名。从 2.14.0 开始，支持某种形式分离的 XML 签名。

在 信封的 XML 签名 案例中，支持的生成的 XML 签名具有以下结构(Variables 由 []括起)。

```
<[parent element]>
  ... <!-- Signature element is added as last child of the parent element-->
  <Signature Id="generated_unique_signature_id">
    <SignedInfo>
      <Reference URI="">
        <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
        (<Transform>)* <!-- By default "http://www.w3.org/2006/12/xml-c14n11" is added to
```

```
the transforms -->
                <DigestMethod>
                <DigestValue>
            </Reference>
            (<Reference URI="#[keyinfo_Id]">
                <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
                <DigestMethod>
                <DigestValue>
            </Reference>)?
            <!-- further references possible, see option 'properties' below -->
        </SignedInfo>
        <SignatureValue>
        (<KeyInfo Id="[keyinfo_id]">)?
        <!-- Object elements possible, see option 'properties' below -->
    </Signature>
</[parent element]>
```

在 信封 XML 签名 案例中，支持的生成的 XML 签名有结构：

```
<Signature Id="generated_unique_signature_id">
  <SignedInfo>
        <Reference URI="#generated_unique_object_id" type="[optional_type_value]">
            (<Transform>)* <!-- By default "http://www.w3.org/2006/12/xml-c14n11" is added to the
transforms -->
            <DigestMethod>
            <DigestValue>
        </Reference>
        (<Reference URI="#[keyinfo_id]">
            <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
            <DigestMethod>
            <DigestValue>
        </Reference>)?
         <!-- further references possible, see option 'properties' below  -->
  </SignedInfo>
  <SignatureValue>
  (<KeyInfo Id="[keyinfo_id]">)?
  <Object Id="generated_unique_object_id"/> <!-- The Object element contains the in-message
body; the object ID can either be generated or set by the option parameter "contentObjectId" -
->
  <!-- Further Object elements possible, see option 'properties' below -->
</Signature>
```

从 2.14.0 分离 XML 签名 时，支持以下结构（请参阅子章节 XML 签名作为 Signed Elements 的 Sibling）：

```
(<[signed element] Id="[id_value]">
<!-- signed element must have an attribute of type ID -->
    ...

</[signed element]>
<other sibling/>*
```

```
<!-- between the signed element and the corresponding signature element, there can be other
siblings.
 Signature element is added as last sibling. -->
<Signature Id="generated_unique_ID">
  <SignedInfo>
    <CanonicalizationMethod>
    <SignatureMethod>
    <Reference URI="#[id_value]" type="[optional_type_value]">
    <!-- reference URI contains the ID attribute value of the signed element -->
        (<Transform>)* <!-- By default "http://www.w3.org/2006/12/xml-c14n11" is added to the
transforms -->
        <DigestMethod>
        <DigestValue>
    </Reference>
    (<Reference URI="#[generated_keyinfo_Id]">
        <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
        <DigestMethod>
        <DigestValue>
    </Reference>)?
  </SignedInfo>
  <SignatureValue>
  (<KeyInfo Id="[generated_keyinfo_id]">)?
</Signature>)+
```

374.2. URI 格式

camel 组件由两个端点组成，其 URI 格式如下：

```
xmlsecurity:sign:name[?options]
xmlsecurity:verify:name[?options]
```

- 使用 signer 端点，您可以为 in-message 的正文生成 XML 签名，可以是 XML 文档或纯文本。信封、信封或分离（截至 12.14 XML 签名）将设置为 out-message 的正文。
- 使用 verifier 端点，您可以验证信封或信封 XML 签名，甚至多个分离（从 2.14.0 开始）在消息正文中包含的 XML 签名；如果验证成功，则原始内容会从 XML 签名中提取，并设置为 out-message 的正文。
- URI 中的 name 部分可由用户选择，以区分 camel 上下文中的不同 signer/verifier 端点。

374.3. 基本示例

以下示例显示了组件的基本用法。

```
from("direct:enveloping").to("xmlsecurity:sign://enveloping?keyAccessor=#accessor",
                             "xmlsecurity:verify://enveloping?keySelector=#selector",
                             "mock:result")
```

在 Spring XML 中：

```
<from uri="direct:enveloping" />
    <to uri="xmlsecurity:sign://enveloping?keyAccessor=#accessor" />
    <to uri="xmlsecurity:verify://enveloping?keySelector=#selector" />
<to uri="mock:result" />
```

对于签名过程，需要私钥。您可以指定提供此私钥的密钥 accessor bean。对于验证，需要对应的公钥；您可以指定一个提供此公钥的密钥选择器 bean。

密钥访问器 Bean 必须实施 KeyAccessor 接口。软件包 org.apache.camel.component.xmlsecurity.api 包含从 Java 密钥存储读取私钥的默认实施类 DefaultKeyAccessor。

节点选择器 bean 必须实施 javax.xml.crypto.KeySelector 接口。软件包 org.apache.camel.component.xmlsecurity.api 包含从密钥存储读取公钥的默认实施类 DefaultKeySelector。

在这个示例中，使用默认的签名算法 http://www.w3.org/2000/09/xmldsig#rsa-sha1。您可以通过选项 signatureAlgorithm 设置您选择的签名算法（请参阅以下）。签名者端点 会创建一个信封 XML 签名。如果要创建 信封 XML 签名，那么您必须指定 Signature 元素的父元素；有关更多详情，请参阅 parentLocalName 选项。

有关创建 分离的 XML 签名，请参阅子章节 "Detached XML Signatures as Siblings of the Signed Elements"。

374.4. 组件选项

XML 安全组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
signerConfiguration (advanced)	使用共享的 XmlSignerConfiguration 配置，用作配置端点的基础。		XmlSignerConfiguration

Name	描述	默认值	类型
verifierConfiguration (advanced)	使用共享的 XmlVerifierConfiguration 配置，用作配置端点的基础。		XmlVerifierConfiguration
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

374.5. 端点选项

XML 安全端点使用 URI 语法进行配置：

```
xmlsecurity:command:name
```

使用以下路径和查询参数：

374.5.1. 路径参数(2 参数)：

Name	描述	默认值	类型
命令	**必需** 是要签名的还是验证。		XmlCommand
name	**必需** URI 中的 name 部分可由用户选择，以区分 camel 上下文中的不同 signer/verifier 端点。		字符串

374.5.2. 查询参数(35 参数)：

Name	描述	默认值	类型
baseURI (common)	您可以设置一个在 URI 解引用中使用的基本 URI。然后，相对 URI 与基础 URI 进行连接。		字符串
clearHeaders (common)	确定在签名和验证后是否清除特定 XML 签名的标头。默认值为 true。	true	布尔值

Name	描述	默认值	类型
cryptoContextProperties (common)	设置加密策略属性。请参阅 link XMLCryptoContext SerialsetProperty (String, Object)。可能的属性在 XMLSignContext 中定义（请参阅支持的属性）。对于 XML 验证，以下属性默认设置为 XML 验证的值 Boolean#TRUE。如果要关闭这些功能，您必须将属性值设置为 Boolean failingFALSE. org.jcp.xml.dsig.validateManifests javax.xml.crypto.dsig.cacheReference		Map
disallowDoctypeDecl (common)	不允许传入的 XML 文档包含 DTD DOCTYPE 声明。默认值为 boolean#TRUE。	true	布尔值
omitXmlDeclaration (common)	指示是否应省略传出消息正文中的 XML 声明。默认值为 false。可以被标头 XmlSignatureConstants#HEADER_OMIT_XML_DECLARATION 覆盖。	false	布尔值
outputXmlEncoding (common)	生成的签名 XML 文档中的字符编码。如果为 null，则使用原始 XML 文档的编码。		字符串
schemaResourceUri (common)	到 XML 架构的类别路径。必须在分离的 XML 签名案例中指定，以确定 ID 属性，可以在信封和信封情况下设置。如果设置，则使用指定的 XML 模式验证 XML 文档。模式资源 URI 可以被标头 XmlSignatureConstantsAllHEADER_SCHEMA_RESOURCE_URI 覆盖。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
uriDereferencer (advanced)	如果要通过引用 URI 限制远程访问，您可以设置自己的 dereferencer。可选参数。如果没有设置供应商默认解引用器，则会使用它解析 URI 片段、HTTP、文件和 XPpointer URI。注意：该实施依赖于供应商！		URIDereferencer
addKeyInfoReference (sign)	为了防止 KeyInfo 元素被篡改，您可以添加对已签名 info 元素的引用，使其可以通过签名值进行保护。默认值为 true。只有在 KeyAccessor. 和 KeyInfo 不是 null 时，才会相关。	true	布尔值

Name	描述	默认值	类型
canonicalizationMethod (sign)	在计算摘要前，用于规范 SignedInfo 元素的规范方法。您可以使用帮助程序方法 XmlSignatureHelper.getCanonicalizationMethod (String algorithm)或 getCanonicalizationMethod (String algorithm, List inclusiveNamespacePrefixes) 来创建规范方法。	http://www.w3.org/TR/2001/REC-xml-c14n-20010315	AlgorithmMethod
contentObjectId (sign)	设置内容对象 Id 属性值。默认情况下会生成 UUID。如果设置 null 值，则会生成一个新的 UUID。仅在信封问题单中使用。		字符串
contentReferenceType (sign)	内容引用的类型。默认值为 null。此值可以被标头 XmlSignatureConstants#HEADER_CONTENT_REFERENCE_TYPE 覆盖。		字符串
contentReferenceUri (sign)	要签名的内容的引用 URI。仅在信封问题单中使用。如果引用 URI 包含 ID 属性值，则必须设置资源架构 URI （setSchemaResourceUri (String)，因为 schema 验证器将查找哪些属性是 ID 属性。在信封或分离的情况下将被忽略。		字符串
digestAlgorithm (sign)	摘要算法 URI。可选参数。此摘要算法用于计算输入消息摘要。如果没有指定此摘要算法，则会从签名算法计算摘要算法。示例 ： http://www.w3.org/2001/04/xmlenc#sha256		字符串
keyAccessor (sign)	对于签名过程，需要私钥。您可以指定提供此私钥的密钥 accessor bean。密钥访问器 Bean 必须实施 KeyAccessor 接口。软件包 org.apache.camel.component.xmlsecurity.api 包含从 Java 密钥存储读取私钥的默认实施类 DefaultKeyAccessor。		KeyAccessor
parentLocalName (sign)	将添加到 XML 签名元素的父元素的本地名称。仅适用于信封的 XML 签名。或者，您也可以使用 setParentXpath (XPathFilterParameterSpec)。默认值为 null。对于信封和分离的 XML 签名，该值必须是 null。这个参数或参数 setParentXpath (XPathFilterParameterSpec)用于 enveloped 签名，参数 setXpathsToIdAttributes (List)不得在同一配置中设置。如果在同一配置中指定了 parentXpath 和 parentLocalName 参数，则会抛出异常。		字符串
parentNamespace (sign)	将添加到 XML 签名元素的 parent 元素的命名空间。		字符串

Name	描述	默认值	类型
parentXpath (sign)	将 XPath 设置为在信封问题单中查找父节点。您可以通过此方法指定父节点，或使用方法 setParentLocalName (String)和 setParentNamespace (String)指定父节点。默认值为 null。对于信封和分离的 XML 签名，该值必须是 null。如果在同一配置中指定了 parentXpath 和 parentLocalName 参数，则会抛出异常。		XPathFilterParameter Spec
明文(sign)	指示消息正文是否包含纯文本。默认值为 false，表示消息正文包含 XML。该值可以被标头 XmlSignatureConstants#HEADER_MESSAGE_IS_PLAIN_TEXT 覆盖。	false	布尔值
plainTextEncoding (sign)	纯文本编码。仅在消息正文是纯文本时相关（请参阅参数 plainText）。默认值为 UTF-8。	UTF-8	字符串
prefixForXmlSignature Namespace (sign)	XML 签名命名空间 http://www.w3.org/2000/09/xmldsig# 的命名空间前缀。默认值为 ds。如果设置了 null 或空值，则不会将前缀用于 XML 签名命名空间。查看没有命名空间的最佳实践 http://www.w3.org/TR/xmldsig-bestpractices/#signing-xml-	ds	字符串
Properties (sign)	要在包含额外属性的 XML 签名中添加其他参考和对象，您可以提供一个实现 XmlSignatureProperties 接口的 bean。		XmlSignatureProperties
signatureAlgorithm (sign)	签名算法.默认值为 http://www.w3.org/2000/09/xmldsig#rsa-sha1。	http://www.w3.org/2000/09/xmldsig#rsa-sha1	字符串
signatureId (sign)	设置签名 Id。如果没有设置此参数(null 值)，则会为签名 ID （默认）生成唯一的 ID。如果此参数设为 （空字符串），则在 signature 元素中不会创建 Id 属性。		字符串
transformMethods (sign)	在计算摘要之前，在消息正文上执行转换。默认情况下，添加了 C14n，如果信封签名（请参阅选项 parentLocalName），http://www.w3.org/2000/09/xmldsig#enveloped-signature 也添加到列表的位置 0 中。使用 XmlSignatureHelper 中的方法创建转换方法。		list

Name	描述	默认值	类型
xpathsToIdAttributes (sign)	通过 XPATH 表达式将分离问题单中签名的元素定义为 ID 属性（类型为 ID 的属性）。对于通过 XPATH 表达式发现的每个元素，会创建一个分离的签名，其引用 URI 包含对应的属性值（使用 ':'）。签名成为已签名元素的最后同级数据。首先对具有更深层次的元素进行签名。您还可以通过标头 XmlSignatureConstants SerialHEADER_XPATHS_TO_ID_ATTRIBUTES 动态设置 XPATH 列表。用于 enveloped 签名的参数 setParentLocalName (String)或 setParentXpath (XPathFilterParameterSpec)，对于分离签名的此参数不能在同一配置中设置。		list
keySelector (verify)	提供验证 XML 签名的密钥。		KeySelector
outputNodeSearch (verify)	设置输出节点搜索值，以确定 XML 签名文档中应设置为输出消息正文的节点。值的类取决于输出节点搜索的类型。输出节点搜索转发到 XmlSignature2Message。		字符串
outputNodeSearchType (verify)	确定用于确定在输出消息 bodyF 中序列化的输出节点的搜索类型。请参阅 setOutputNodeSearch (Object)。您可以在 DefaultXmlSignature2Message 中找到的默认搜索类型。	default	字符串
removeSignatureElements (verify)	指示 XML 签名元素（带有本地名称签名和名称sapce http://www.w3.org/2000/09/xmldsig#的元素）是否应该从文档设置为输出消息。通常，只有当 XML 签名被接收时，才需要这样做。默认值为 boolean#FALSE。这个参数被转发到 XmlSignature2Message。如果输出节点搜索类型为 DefaultXmlSignature2Message helloOUTPUT_NODE_SEARCH_TYPE_DEFAULT.F，则此指标无效	false	布尔值
secureValidation (verify)	启用安全验证。如果为 true，则启用安全验证。	true	布尔值
validationFailedHandler (verify)	处理不同的验证失败情况。默认实施会针对不同的情况抛出特定的例外（所有例外情况都有软件包名称 org.apache.camel.component.xmlsecurity.api，它是 XmlSignatureInvalidException 的子类。如果签名值验证失败，则会抛出 XmlSignatureInvalidValueException。如果引用验证失败，则会抛出 XmlSignatureInvalidContentHashException。如需更多信息，请参阅 JavaDoc。		ValidationFailedHandler

Name	描述	默认值	类型
xmlSignature2Message (verify)	在验证后将 XML 签名映射到 output-message 的 bean。应该如何通过选项 outputNodeSearchType、outputNodeSearch 和 removeSignatureElements 来配置此映射。默认实现提供了三种与三种输出节点搜索类型 Default, ElementName, 和 XPath 相关的可能性。默认实现决定了一个节点，它被序列化并设置为输出消息的正文（如果搜索类型是 ElementName），则输出节点（在这种情况下，一个元素）由搜索值中定义的本地名称和命名空间决定（请参阅选项 outputNodeSearch）。如果搜索类型是 XPath，则输出节点由搜索值中指定的 XPath 决定（本例中为输出节点可以是 type Element, TextNode 或 Document）。如果输出节点搜索类型是 Default，则应用以下规则： 在信封的 XML 签名案例中（使用 URI= 的引用并转换 http://www.w3.org/2000/09/xmldsig#enveloped-signature），如果没有 Signature 元素将传入的 XML 文档设置为输出消息正文。在非嵌套的 XML 签名案例中，消息正文由引用的对象决定；这在输出节点在 Enveloping XML 签名问题单中进行了更详细的说明。		XmlSignature2Message
xmlSignatureChecker (verify)	此接口允许应用程序在执行验证前检查 XML 签名。建议在 http://www.w3.org/TR/xmldsig-bestpractices/#check-what-is-signed中执行此步骤		XmlSignatureChecker

374.6. SPRING BOOT AUTO-CONFIGURATION

组件支持 63 个选项，如下所列。

Name	描述	默认值	类型
camel.component.xmlsecurity.enabled	启用 xmlsecurity 组件	true	布尔值
camel.component.xmlsecurity.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.xmlsecurity.signer-configuration.add-key-info-reference	为了防止 KeyInfo 元素被篡改，您可以添加对已签名 info 元素的引用，使其可以通过签名值进行保护。默认值为 true。只有在 KeyAccessor. 和 KeyInfo 不是 null 时，才会相关。	true	布尔值

Name	描述	默认值	类型
camel.component.xmlsecurity.signer-configuration.base-uri	您可以设置一个在 URI 解引用中使用的基本 URI。然后，相对 URI 与基础 URI 进行连接。		字符串
camel.component.xmlsecurity.signer-configuration.canonicalization-method	在计算摘要前，用于规范 SignedInfo 元素的规范方法。您可以使用帮助程序方法 XmlSignatureHelper.getCanonicalizationMethod (String algorithm)或 getCanonicalizationMethod (String algorithm, List inclusiveNamespacePrefixes) 来创建规范方法。		AlgorithmMethod
camel.component.xmlsecurity.signer-configuration.canonicalization-method-name			字符串
camel.component.xmlsecurity.signer-configuration.clear-headers	确定在签名和验证后是否清除特定 XML 签名的标头。默认值为 true。	true	布尔值
camel.component.xmlsecurity.signer-configuration.content-object-id	设置内容对象 Id 属性值。默认情况下会生成 UUID。如果设置 null 值，则会生成一个新的 UUID。仅在信封问题单中使用。		字符串
camel.component.xmlsecurity.signer-configuration.content-reference-type	内容引用的类型。默认值为 null。此值可以被标头 XmlSignatureConstants#HEADER_CONTENT_REFERENCE_TYPE 覆盖。		字符串
camel.component.xmlsecurity.signer-configuration.content-reference-uri	要签名的内容的引用 URI。仅在信封问题单中使用。如果引用 URI 包含 ID 属性值，则必须设置资源架构 URI（setSchemaResourceUri (String)，因为 schema 验证器将查找哪些属性是 ID 属性。在信封或分离的情况下将被忽略。		字符串

Name	描述	默认值	类型
camel.component.xmlsecurity.signer-configuration.crypto-context-properties	设置加密策略属性。请参阅 link XMLCryptoContext SerialsetProperty (String, Object)。可能的属性在 XMLSignContext 中定义（请参阅支持的属性）。对于 XML 验证，以下属性默认设置为 XML 验证的值 Boolean#TRUE。如果要关闭这些功能，您必须将属性值设置为 Boolean failingFALSE. org.jcp.xml.dsig.validateManifests javax.xml.crypto.dsig.cacheReference		Map
camel.component.xmlsecurity.signer-configuration.digest-algorithm	摘要算法 URI。可选参数。此摘要算法用于计算输入消息摘要。如果没有指定此摘要算法，则会从签名算法计算摘要算法。示例 ： http://www.w3.org/2001/04/xmlenc#sha256		字符串
camel.component.xmlsecurity.signer-configuration.disallow-doctype-decl	不允许传入的 XML 文档包含 DTD DOCTYPE 声明。默认值为 boolean#TRUE。	true	布尔值
camel.component.xmlsecurity.signer-configuration.key-accessor	对于签名过程，需要私钥。您可以指定提供此私钥的密钥 accessor bean。密钥访问器 Bean 必须实施 KeyAccessor 接口。软件包 org.apache.camel.component.xmlsecurity.api 包含从 Java 密钥存储读取私钥的默认实施类 DefaultKeyAccessor。		KeyAccessor
camel.component.xmlsecurity.signer-configuration.key-accessor-name			字符串
camel.component.xmlsecurity.signer-configuration.omit-xml-declaration	指示是否应省略传出消息正文中的 XML 声明。默认值为 false。可以被标头 XmlSignatureConstants#HEADER_OMIT_XML_DECLARATION 覆盖。	false	布尔值
camel.component.xmlsecurity.signer-configuration.output-xml-encoding	生成的签名 XML 文档中的字符编码。如果为 null，则使用原始 XML 文档的编码。		字符串

Name	描述	默认值	类型
camel.component.xmlsecurity.signer-configuration.parent-local-name	将添加到 XML 签名元素的父元素的本地名称。仅适用于信封的 XML 签名。或者，您也可以使用 setParentXpath (XPathFilterParameterSpec)。默认值为 null。对于信封和分离的 XML 签名，该值必须是 null。这个参数或参数 setParentXpath (XPathFilterParameterSpec)用于 enveloped 签名，参数 setXpathsToIdAttributes (List)不得在同一配置中设置。如果在同一配置中指定了 parentXpath 和 parentLocalName 参数，则会抛出异常。		字符串
camel.component.xmlsecurity.signer-configuration.parent-namespace	将添加到 XML 签名元素的 parent 元素的命名空间。		字符串
camel.component.xmlsecurity.signer-configuration.parent-xpath	将 XPath 设置为在信封问题单中查找父节点。您可以通过此方法指定父节点，或使用方法 setParentLocalName (String)和 setParentNamespace (String)指定父节点。默认值为 null。对于信封和分离的 XML 签名，该值必须是 null。如果在同一配置中指定了 parentXpath 和 parentLocalName 参数，则会抛出异常。		XPathFilterParameter Spec
camel.component.xmlsecurity.signer-configuration.plain-text	指示消息正文是否包含纯文本。默认值为 false，表示消息正文包含 XML。该值可以被标头 XmlSignatureConstants#HEADER_MESSAGE_IS_PLAIN_TEXT 覆盖。	false	布尔值
camel.component.xmlsecurity.signer-configuration.plain-text-encoding	纯文本编码。仅在消息正文是纯文本时相关（请参阅参数 plainText）。默认值为 UTF-8。	UTF-8	字符串
camel.component.xmlsecurity.signer-configuration.prefix-for-xml-signature-namespace	XML 签名命名空间 http://www.w3.org/2000/09/xmldsig# 的命名空间前缀。默认值为 ds。如果设置了 null 或空值，则不会将前缀用于 XML 签名命名空间。查看没有命名空间的最佳实践 http://www.w3.org/TR/xmldsig-bestpractices/#signing-xml-	ds	字符串
camel.component.xmlsecurity.signer-configuration.properties	要在包含额外属性的 XML 签名中添加其他参考和对象，您可以提供一个实现 XmlSignatureProperties 接口的 bean。		XmlSignatureProperties

Name	描述	默认值	类型
camel.component.xmlsecurity.signer-configuration.properties-name			字符串
camel.component.xmlsecurity.signer-configuration.schema-resource-uri	到 XML 架构的类别路径。必须在分离的 XML 签名案例中指定，以确定 ID 属性，可以在信封和信封情况下设置。如果设置，则使用指定的 XML 模式验证 XML 文档。模式资源 URI 可以被标头 XmlSignatureConstantsAllHEADER_SCHEMA_RESOURCE_URI 覆盖。		字符串
camel.component.xmlsecurity.signer-configuration.signature-algorithm	签名算法.默认值为 http://www.w3.org/2000/09/xmldsig#rsa-sha1。	http://www.w3.org/2000/09/xmldsig#rsa-sha1	字符串
camel.component.xmlsecurity.signer-configuration.signature-id	设置签名 Id。如果没有设置此参数(null 值)，则会为签名 ID （默认）生成唯一的 ID。如果此参数设为 （空字符串），则在 signature 元素中不会创建 Id 属性。		字符串
camel.component.xmlsecurity.signer-configuration.transform-methods	在计算摘要之前，在消息正文上执行转换。默认情况下，添加了 C14n，如果信封签名（请参阅选项 parentLocalName），http://www.w3.org/2000/09/xmldsig#enveloped-signature 也添加到列表的位置 0 中。使用 XmlSignatureHelper 中的方法创建转换方法。		list
camel.component.xmlsecurity.signer-configuration.transform-methods-name			字符串
camel.component.xmlsecurity.signer-configuration.uri-dereferencer	如果要通过引用 URI 限制远程访问，您可以设置自己的 dereferencer。可选参数。如果没有设置供应商默认解引用器，则会使用它解析 URI 片段、HTTP、文件和 XPpointer URI。注意：该实施依赖于供应商！		URIDereferencer

Name	描述	默认值	类型
camel.component.xmlsecurity.signer-configuration.xpaths-to-id-attributes	通过 XPATH 表达式将分离问题单中签名的元素定义为 ID 属性（类型为 ID 的属性）。对于通过 XPATH 表达式发现的每个元素，会创建一个分离的签名，其引用 URI 包含对应的属性值（使用 ':'）。签名成为已签名元素的最后同级数据。首先对具有更深层次的元素进行签名。您还可以通过标头 XmlSignatureConstants SerialHEADER_XPATHS_TO_ID_ATTRIBUTES 动态设置 XPATH 列表。用于 enveloped 签名的参数 setParentLocalName (String)或 setParentXpath (XPathFilterParameterSpec)，对于分离签名的此参数不能在同一配置中设置。		list
camel.component.xmlsecurity.verifier-configuration.base-uri	您可以设置一个在 URI 解引用中使用的基本 URI。然后，相对 URI 与基础 URI 进行连接。		字符串
camel.component.xmlsecurity.verifier-configuration.clear-headers	确定在签名和验证后是否清除特定 XML 签名的标头。默认值为 true。	true	布尔值
camel.component.xmlsecurity.verifier-configuration.crypto-context-properties	设置加密策略属性。请参阅 link XMLCryptoContext SerialsetProperty (String, Object)。可能的属性在 XMLSignContext 中定义（请参阅支持的属性）。对于 XML 验证，以下属性默认设置为 XML 验证的值 Boolean#TRUE。如果要关闭这些功能，您必须将属性值设置为 Boolean failingFALSE. org.jcp.xml.dsig.validateManifests javax.xml.crypto.dsig.cacheReference		Map
camel.component.xmlsecurity.verifier-configuration.disallow-doctype-decl	不允许传入的 XML 文档包含 DTD DOCTYPE 声明。默认值为 boolean#TRUE。	true	布尔值
camel.component.xmlsecurity.verifier-configuration.key-selector	提供验证 XML 签名的密钥。		KeySelector

Name	描述	默认值	类型
camel.component.xmlsecurity.verifier-configuration.omit-xml-declaration	指示是否应省略传出消息正文中的 XML 声明。默认值为 false。可以被标头 XmlSignatureConstants#HEADER_OMIT_XML_DECLARATION 覆盖。	false	布尔值
camel.component.xmlsecurity.verifier-configuration.output-node-search	设置输出节点搜索值，以确定 XML 签名文档中应设置为输出消息正文的节点。值的类取决于输出节点搜索的类型。输出节点搜索转发到 XmlSignature2Message。		对象
camel.component.xmlsecurity.verifier-configuration.output-node-search-type	确定用于确定在输出消息 bodyF 中序列化的输出节点的搜索类型。请参阅 setOutputNodeSearch (Object)。您可以在 DefaultXmlSignature2Message 中找到的默认搜索类型。	default	字符串
camel.component.xmlsecurity.verifier-configuration.output-xml-encoding	生成的签名 XML 文档中的字符编码。如果为 null，则使用原始 XML 文档的编码。		字符串
camel.component.xmlsecurity.verifier-configuration.remove-signature-elements	指示 XML 签名元素（带有本地名称签名和名称sapce http://www.w3.org/2000/09/xmldsig#的元素）是否应该从文档设置为输出消息。通常，只有当 XML 签名被接收时，才需要这样做。默认值为 boolean#FALSE。这个参数被转发到 XmlSignature2Message。如果输出节点搜索类型为 DefaultXmlSignature2Message helloOUTPUT_NODE_SEARCH_TYPE_DEFAULT.F，则此指标无效	false	布尔值
camel.component.xmlsecurity.verifier-configuration.schema-resource-uri	到 XML 架构的类别路径。必须在分离的 XML 签名案例中指定，以确定 ID 属性，可以在信封和信封情况下设置。如果设置，则使用指定的 XML 模式验证 XML 文档。模式资源 URI 可以被标头 XmlSignatureConstantsAllHEADER_SCHEMA_RESOURCE_URI 覆盖。		字符串
camel.component.xmlsecurity.verifier-configuration.secure-validation	启用安全验证。如果为 true，则启用安全验证。	true	布尔值

Name	描述	默认值	类型
camel.component.xmlsecurity.verifier-configuration.uri-dereferencer	如果要通过引用 URI 限制远程访问，您可以设置自己的 dereferencer。可选参数。如果没有设置供应商默认解引用器，则会使用它解析 URI 片段、HTTP、文件和 XPpointer URI。注意：该实施依赖于供应商！		URIDereferencer
camel.component.xmlsecurity.verifier-configuration.validation-failed-handler	处理不同的验证失败情况。默认实施会针对不同的情况抛出特定的例外（所有例外情况都有软件包名称 org.apache.camel.component.xmlsecurity.api，它是 XmlSignatureInvalidException 的子类。如果签名值验证失败，则会抛出 XmlSignatureInvalidValueException。如果引用验证失败，则会抛出 XmlSignatureInvalidContentHashException。如需更多信息，请参阅 JavaDoc。		ValidationFailedHandler
camel.component.xmlsecurity.verifier-configuration.validation-failed-handler-name	处理程序的名称为 @param validationFailedHandlerName		字符串
camel.component.xmlsecurity.verifier-configuration.xml-signature-checker	此接口允许应用程序在执行验证前检查 XML 签名。建议在 http://www.w3.org/TR/xmldsig-bestpractices/#check-what-is-signed中执行此步骤		XmlSignatureChecker
camel.component.xmlsecurity.verifier-configuration.xml-signature2-message	在验证后将 XML 签名映射到 output-message 的 bean。应该如何通过选项 outputNodeSearchType、outputNodeSearch 和 removeSignatureElements 来配置此映射。默认实现提供了三种与三种输出节点搜索类型 Default, ElementName, 和 XPath 相关的可能性。默认实现决定了一个节点，它被序列化并设置为输出消息的正文（如果搜索类型是 ElementName），则输出节点（在这种情况下，一个元素）由搜索值中定义的本地名称和命名空间决定（请参阅选项 outputNodeSearch）。如果搜索类型是 XPath，则输出节点由搜索值中指定的 XPath 决定（本例中为输出节点可以是 type Element, TextNode 或 Document）。如果输出节点搜索类型是 Default，则应用以下规则： 在信封的 XML 签名案例中（使用 URI= 的引用并转换 http://www.w3.org/2000/09/xmldsig#enveloped-signature），如果没有 Signature 元素将传入的 XML 文档设置为输出消息正文。在非嵌套的 XML 签名案例中，消息正文由引用的对象决定；这在输出节点在 Enveloping XML 签名问题单中进行了更详细的说明。		XmlSignature2Message

Name	描述	默认值	类型
camel.dataformat.securexml.add-key-value-for-encrypted-key	是否在 EncryptedKey 结构中将用于加密会话密钥的公钥添加为 KeyValue。	true	布尔值
camel.dataformat.securexml.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.securexml.digest-algorithm	与 RSA OAEP 算法一起使用的摘要算法。可用的选择有：XMLCipher.SHA1 XMLCipher.SHA256 XMLCipher.SHA512 默认值是 XMLCipher.SHA1	SHA1	字符串
camel.dataformat.securexml.enabled	启用 securexml dataformat	true	布尔值
camel.dataformat.securexml.key-cipher-algorithm	用于加密/解密非对称密钥的密码算法。可用的选择包括： XMLCipher.RSA_v1dot5 XMLCipher.RSA_OAEP XMLCipher.RSA_OAEP_11，默认值为 XMLCipher.RSA_OAEP	RSA_OAEP	字符串
camel.dataformat.securexml.key-or-trust-store-parameters-id	代表 registry 中的 KeyStore 实例，用于创建和加载代表发件人 trustStore 或接收者的 keyStore 的 KeyStore 实例的配置选项。		字符串
camel.dataformat.securexml.key-password	用于从 KeyStore 检索私钥的密码。此密钥用于非对称解密。		字符串
camel.dataformat.securexml.mgf-algorithm	用于 RSA OAEP 算法的 MGF Algorithm。可用的选择包括： EncryptionConstants.MGF1_SHA1 EncryptionConstants.MGF1_SHA256 EncryptionConstants.MGF1_SHA512，默认值为 EncryptionConstants.MGF1_SHA1	MGF1_SHA1	字符串
camel.dataformat.securexml.pass-phrase	用作 passPhrase 用于加密/解密内容的字符串。必须提供 passPhrase。如果没有指定 passPhrase，则使用默认 passPhrase。passPhrase 需要与适当的加密算法一起使用。例如，使用 TRIPLEDES passPhase 只能是一个 24 个字节键		字符串
camel.dataformat.securexml.pass-phrase-byte	用作 passPhrase 用于加密/解密内容的字节。必须提供 passPhrase。如果没有指定 passPhrase，则使用默认 passPhrase。passPhrase 需要与适当的加密算法一起使用。例如，使用 TRIPLEDES passPhase 只能是一个 24 个字节键		Byte[]

Name	描述	默认值	类型
camel.dataformat.securexml.recipient-key-alias	执行非对称密钥加密或解密时从 KeyStore 检索收件人的公钥或私钥时使用的密钥别名。		字符串
camel.dataformat.securexml.secure-tag	XPath 引用所选的用于加密/解密的 XML 元素。如果没有指定标签，则整个有效负载会被加密/解密。		字符串
camel.dataformat.securexml.secure-tag-contents	一个布尔值，用于指定 XML 元素是否被加密或者 XML Element false 的内容 = Element Level true = Element Content Level	false	布尔值
camel.dataformat.securexml.xml-cipher-algorithm	用于加密/解密 XML 消息内容的密码算法。可用的选择包括： XMLCipher.TRIPLEDES XMLCipher.AES_128 XMLCipher.AES_128_GCM XMLCipher.AES_192 XMLCipher.AES_192_GCM XMLCipher.AES_256 XMLCipher.AES_256_GCM XMLCipher.SEED_128 XMLCipher.CAMELLIA_128 XMLCipher.CAMELLIA_192 XMLCipher.CAMELLIA_256_256_256_ARCipher.SEED_128 XMLCipher.CAMELLIA_192 XMLCipher.CAMELLIA_256_256_256_256_GCM是MLCipher.TRIPLEDES。	TRIPLEDES	字符串

374.6.1. 输出节点在 Enveloping XML 签名问题单中确定

验证节点从 XML 签名文档中提取后，该文档最后返回到 output-message 正文。在信封 XML 签名案例中，XmlSignature2Message 的默认实现 DefaultXmlSignature2Message 以下列方式为节点搜索类型 Default （请参阅选项 xmlSignature2Message）：

- *首先决定对象引用：*
  - *只考虑相同的文档引用(URI 必须以 #开头)*
  - *另外，还考虑通过清单对对象进行间接相同的文档引用。*
  - *生成的对象引用数量必须为 1。*
- *然后，对象被解引用，对象必须只包含一个 XML 元素。这个元素作为输出节点返回。*

这意味着信封 XML 签名必须具有该结构之一：

```
<Signature>
    <SignedInfo>
      <Reference URI="#object"/>
      <!-- further references possible but they must not point to an Object or Manifest
containing an object reference -->
      ...
    </SignedInfo>

    <Object Id="object">
       <!-- contains one XML element which is extracted to the message body -->
    <Object>
    <!-- further object elements possible which are not referenced-->
    ...
    (<KeyInfo>)?
</Signature>
```

或结构：

```
<Signature>
    <SignedInfo>
      <Reference URI="#manifest"/>
      <!-- further references  are possible but they must not point to an Object or other
manifest containing an object reference -->
      ...
    </SignedInfo>

    <Object >
      <Manifest Id="manifest">
        <Reference URI=#object/>
      </Manifest>
    </Objet>
    <Object Id="object">
       <!-- contains the DOM node which is extracted to the message body -->
    </Object>
     <!-- further object elements possible which are not referenced -->
    ...
    (<KeyInfo>)?
</Signature>
```

374.7. 分离的 XML 签名作为 SIGNED ELEMENTS 的 SIBLING

Since 2.14.0

您可以创建分离的签名，其中签名是已签名元素的同级功能。以下示例包含两个分离的签名。第一个签名用于元素 C，第二个签名用于元素 A。签名是 嵌套的 ；第二个签名用于元素 A，它还包含第一个签

名。

分离 XML 签名示例

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
   <A ID="IDforA">
      <B>
         <C ID="IDforC">
            <D>dvalue</D>
         </C>
         <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
            Id="_6bf13099-0568-4d76-8649-faf5dcb313c0">
            <ds:SignedInfo>
               <ds:CanonicalizationMethod
                  Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
               <ds:SignatureMethod
                  Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
               <ds:Reference URI="#IDforC">
                  ...
               </ds:Reference>
            </ds:SignedInfo>
            <ds:SignatureValue>aUDFmiG71</ds:SignatureValue>
         </ds:Signature>
      </B>
   </A>
   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"Id="_6b02fb8a-30df-42c6-ba25-
76eba02c8214">
      <ds:SignedInfo>
         <ds:CanonicalizationMethod
            Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
         <ds:SignatureMethod
            Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
         <ds:Reference URI="#IDforA">
            ...
         </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>q3tvRoGgc8cMUqUSzP6C21zb7tt04riPnDuk=</ds:SignatureValue>
   </ds:Signature>
<root>
```

这个示例显示您可以签署多个元素，并且每个元素都以同级方式创建签名。要签名的元素必须具有类型为 ID 的属性。属性的 ID 类型必须在 XML 模式中定义（请参阅选项 schemaResourceUri）。您可以指定一个指向类型 ID 属性的 XPATH 表达式列表（请参阅 xpathsToIdAttributes）。这些属性确定要签名的元素。元素使用 keyAccessor bean 提供的同一密钥签名。首先对具有更高的元素（例如，更深入的）层次结构级别进行签名。在示例中，元素 C 在元素 A 之前签名。

Java DSL 示例

```
from("direct:detached")
  .to("xmlsecurity:sign://detached?
keyAccessor=#keyAccessorBeant&xpathsToIdAttributes=#xpathsToIdAttributesBean&schem
aResourceUri=Test.xsd")
  .to("xmlsecurity:verify://detached?
keySelector=#keySelectorBean&schemaResourceUri=org/apache/camel/component/xmlsecuri
ty/Test.xsd")
  .to("mock:result");
```

Spring 示例

```
<bean id="xpathsToIdAttributesBean" class="java.util.ArrayList">
    <constructor-arg type="java.util.Collection">
      <list>
        <bean
          class="org.apache.camel.component.xmlsecurity.api.XmlSignatureHelper"
          factory-method="getXpathFilter">
          <constructor-arg type="java.lang.String"
            value="/ns:root/a/@ID" />
          <constructor-arg>
            <map key-type="java.lang.String" value-type="java.lang.String">
              <entry key="ns" value="http://test" />
            </map>
          </constructor-arg>
        </bean>
      </list>
    </constructor-arg>
  </bean>
...
 <from uri="direct:detached" />
    <to
      uri="xmlsecurity:sign://detached?
keyAccessor=#keyAccessorBean&amp;xpathsToIdAttributes=#xpathsToIdAttributesBean&amp;schema
ResourceUri=Test.xsd" />
    <to
      uri="xmlsecurity:verify://detached?
keySelector=#keySelectorBean&amp;schemaResourceUri=Test.xsd" />
    <to uri="mock:result" />
```

374.8. XADES-BES/EPES 用于信号端点

从 Camel 2.15.0 开始提供

XML 高级电子签名(XAdES) 定义 XML 签名的扩展。该标准由 欧洲电信标准研究所 定义，允许您在用于电子签名的社区框架上创建符合欧盟指令(1999/93/EC)的签名。XAdES 定义不同的签名属性集合，称为签名表单。我们在 Signer Endpoint 中支持签名形式的签名形式 Basic Electronic Signature (XAdES-BES)和 Explicit Policy Based Electronic Signature (XAdES-EPES)。不支持使用Validation Data XAdES-T 和 XAdES-C 格式的 Electronic Signature 。

我们支持 XAdES-EPES 表单的以下属性（"?" 表示零次或出现一项：

支持的 XAdES-EPES 属性

```
<QualifyingProperties Target>
   <SignedProperties>
      <SignedSignatureProperties>
         (SigningTime)?
         (SigningCertificate)?
         (SignaturePolicyIdentifier)
         (SignatureProductionPlace)?
         (SignerRole)?
      </SignedSignatureProperties>
      <SignedDataObjectProperties>
         (DataObjectFormat)?
         (CommitmentTypeIndication)?
      </SignedDataObjectProperties>
   </SignedProperties>
</QualifyingProperties>
```

XAdES-BES 表单的属性相同，但 SignaturePolicyIdentifier 属性不是 XAdES-BES 的一部分。

您可以通过 bean org.apache.camel.component.xmlsecurity.api.XAdESSignatureProperties 或 org.apache.camel.component.xmlsecurity.api.DefaultXAdESSignatureProperties 配置 XAdES-BES/EPES 属性。XAdESSignatureProperties 支持上述所有属性，但 SigningCertificate 属性除外。要获取 SigningCertificate 属性，您必须覆盖方法 XAdESSignatureProperties.getSigningCertificate （） 或 XAdESSignatureProperties.getSigningCertificateChain （）。
类 DefaultXAdESSignatureProperties 覆盖方法 getSigningCertificate （），并通过密钥存储和别名指定签名证书。以下示例显示了您可以指定的所有参数。如果您不需要某些参数，您只需省略它们。

Java DSL 中的 XAdES-BES/EPES 示例

```
 Keystore keystore = ... // load a keystore
DefaultKeyAccessor accessor = new DefaultKeyAccessor();
accessor.setKeyStore(keystore);
accessor.setPassword("password");
accessor.setAlias("cert_alias"); // signer key alias

DefaultXAdESSignatureProperties props = new DefaultXAdESSignatureProperties();
props.setNamespace("http://uri.etsi.org/01903/v1.3.2#"); // sets the namespace for the XAdES elements; the namspace is related to the XAdES version, default value is "http://uri.etsi.org/01903/v1.3.2#", other possible values are "http://uri.etsi.org/01903/v1.1.1#" and "http://uri.etsi.org/01903/v1.2.2#"
props.setPrefix("etsi"); // sets the prefix for the XAdES elements, default value is "etsi"

// signing certificate
```

```
props.setKeystore(keystore));
props.setAlias("cert_alias"); // specify the alias of the signing certificate in the keystore =
signer key alias
props.setDigestAlgorithmForSigningCertificate(DigestMethod.SHA256); // possible values for
the algorithm are "http://www.w3.org/2000/09/xmldsig#sha1",
"http://www.w3.org/2001/04/xmlenc#sha256", "http://www.w3.org/2001/04/xmldsig-
more#sha384", "http://www.w3.org/2001/04/xmlenc#sha512", default value is
"http://www.w3.org/2001/04/xmlenc#sha256"
props.setSigningCertificateURIs(Collections.singletonList("http://certuri"));

// signing time
props.setAddSigningTime(true);

// policy
props.setSignaturePolicy(XAdESSignatureProperties.SIG_POLICY_EXPLICIT_ID);
// also the values XAdESSignatureProperties.SIG_POLICY_NONE ("None"), and
XAdESSignatureProperties.SIG_POLICY_IMPLIED ("Implied")are possible, default value is
XAdESSignatureProperties.SIG_POLICY_EXPLICIT_ID ("ExplicitId")
// For "None" and "Implied" you must not specify any further policy parameters
props.setSigPolicyId("urn:oid:1.2.840.113549.1.9.16.6.1");
props.setSigPolicyIdQualifier("OIDAsURN"); //allowed values are empty string, "OIDAsURI",
"OIDAsURN"; default value is empty string
props.setSigPolicyIdDescription("invoice version 3.1");
props.setSignaturePolicyDigestAlgorithm(DigestMethod.SHA256);// possible values for the
algorithm are "http://www.w3.org/2000/09/xmldsig#sha1",
http://www.w3.org/2001/04/xmlenc#sha256", "http://www.w3.org/2001/04/xmldsig-
more#sha384", "http://www.w3.org/2001/04/xmlenc#sha512", default value is
http://www.w3.org/2001/04/xmlenc#sha256"
props.setSignaturePolicyDigestValue("Ohixl6upD6av8N7pEvDABhEL6hM=");
// you can add  qualifiers for the signature policy either by specifying text or an XML fragment
with the root element "SigPolicyQualifier"
props.setSigPolicyQualifiers(Arrays
   .asList(new String[] {
      "<SigPolicyQualifier xmlns=\"http://uri.etsi.org/01903/v1.3.2#\">
<SPURI>http://test.com/sig.policy.pdf</SPURI><SPUserNotice><ExplicitText>display
text</ExplicitText>"
         + "</SPUserNotice></SigPolicyQualifier>", "category B" }));
props.setSigPolicyIdDocumentationReferences(Arrays.asList(new String[]
{"http://test.com/policy.doc.ref1.txt",
   "http://test.com/policy.doc.ref2.txt" }));

// production place
props.setSignatureProductionPlaceCity("Munich");
props.setSignatureProductionPlaceCountryName("Germany");
props.setSignatureProductionPlacePostalCode("80331");
props.setSignatureProductionPlaceStateOrProvince("Bavaria");

//role
// you can add claimed roles either by specifying text or an XML fragment with the root
element "ClaimedRole"
props.setSignerClaimedRoles(Arrays.asList(new String[] {"test",
   "<a:ClaimedRole xmlns:a=\"http://uri.etsi.org/01903/v1.3.2#\">
<TestRole>TestRole</TestRole></a:ClaimedRole>" }));
props.setSignerCertifiedRoles(Collections.singletonList(new
XAdESEncapsulatedPKIData("Ahixl6upD6av8N7pEvDABhEL6hM=",
   "http://uri.etsi.org/01903/v1.2.2#DER", "IdCertifiedRole")));
```

```
// data object format
props.setDataObjectFormatDescription("invoice");
props.setDataObjectFormatMimeType("text/xml");
props.setDataObjectFormatIdentifier("urn:oid:1.2.840.113549.1.9.16.6.2");
props.setDataObjectFormatIdentifierQualifier("OIDAsURN"); //allowed values are empty string,
"OIDAsURI", "OIDAsURN"; default value is empty string
props.setDataObjectFormatIdentifierDescription("identifier desc");
props.setDataObjectFormatIdentifierDocumentationReferences(Arrays.asList(new String[] {
    "http://test.com/dataobject.format.doc.ref1.txt",
"http://test.com/dataobject.format.doc.ref2.txt" }));

//commitment
props.setCommitmentTypeId("urn:oid:1.2.840.113549.1.9.16.6.4");
props.setCommitmentTypeIdQualifier("OIDAsURN"); //allowed values are empty string,
"OIDAsURI", "OIDAsURN"; default value is empty string
props.setCommitmentTypeIdDescription("description for commitment type ID");
props.setCommitmentTypeIdDocumentationReferences(Arrays.asList(new String[]
{"http://test.com/commitment.ref1.txt",
    "http://test.com/commitment.ref2.txt" }));
// you can specify a commitment type qualifier either by simple text or an XML fragment with
root element "CommitmentTypeQualifier"
props.setCommitmentTypeQualifiers(Arrays.asList(new String[] {"commitment qualifier",
    "<c:CommitmentTypeQualifier xmlns:c=\"http://uri.etsi.org/01903/v1.3.2#\"><C>c</C>
</c:CommitmentTypeQualifier>" }));

beanRegistry.bind("xmlSignatureProperties",props);
beanRegistry.bind("keyAccessorDefault",keyAccessor);

// you must reference the properties bean in the "xmlsecurity" URI
from("direct:xades").to("xmlsecurity:sign://xades?
keyAccessor=#keyAccessorDefault&properties=#xmlSignatureProperties")
            .to("mock:result");
```

Spring XML 中的 XAdES-BES/EPES 示例

```
...
<from uri="direct:xades" />
    <to
        uri="xmlsecurity:sign://xades?keyAccessor=#accessorRsa&amp;properties=#xadesProperties"
/>
    <to uri="mock:result" />
...
<bean id="xadesProperties"
    class="org.apache.camel.component.xmlsecurity.api.XAdESSignatureProperties">
    <!-- For more properties see the previous Java DSL example.
        If you want to have a signing certificate then use the bean class
DefaultXAdESSignatureProperties (see the previous Java DSL example). -->
    <property name="signaturePolicy" value="ExplicitId" />
    <property name="sigPolicyId" value="http://www.test.com/policy.pdf" />
    <property name="sigPolicyIdDescription" value="factura" />
    <property name="signaturePolicyDigestAlgorithm"
value="http://www.w3.org/2000/09/xmldsig#sha1" />
    <property name="signaturePolicyDigestValue" value="Ohixl6upD6av8N7pEvDABhEL1hM=" />
```

```
    <property name="signerClaimedRoles" ref="signerClaimedRoles_XMLSigner" />
    <property name="dataObjectFormatDescription" value="Factura electrónica" />
    <property name="dataObjectFormatMimeType" value="text/xml" />
</bean>
<bean class="java.util.ArrayList" id="signerClaimedRoles_XMLSigner">
    <constructor-arg>
        <list>
            <value>Emisor</value>
            <value>&lt;ClaimedRole
                xmlns=&quot;http://uri.etsi.org/01903/v1.3.2#&quot;&gt;&lt;test
                xmlns=&quot;http://test.com/&quot;&gt;test&lt;/test&gt;&lt;/ClaimedRole&gt;</value>
        </list>
    </constructor-arg>
</bean>
```

374.8.1. Headers

标头	类型	描述
CamelXmlSignatureXAdESQualifyingPropertiesId	**字符串**	对于 **QualifyingProperties** 元素的 'Id' 属性值
CamelXmlSignatureXAdESSignedDataObjectPropertiesId	**字符串**	对于 **SignedDataObjectProperties** 元素的 'Id' 属性值
CamelXmlSignatureXAdESSignedSignaturePropertiesId	**字符串**	对于 **SignedSignatureProperties** 元素的 'Id' 属性值
CamelXmlSignatureXAdESDataObjectFormatEncoding	**字符串**	对于 **DataObjectFormat** 元素的 Encoding 元素的值
CamelXmlSignatureXAdESNamespace	**字符串**	覆盖 XAdES 命名空间参数值
CamelXmlSignatureXAdESPrefix	**字符串**	覆盖 XAdES 前缀参数值

374.8.2. XAdES 版本 1.4.2 的限制

- 不支持 XAdES-T 和 XAdES-C 的签名
- 只有签名人部分实施。verifier 部分当前不可用。
- 不支持 QualifyingPropertiesReference 元素（请参阅 spec 的 6.3.2 部分）。
- 不支持 SignaturePolicy Identifier 元素中包含的 SignaturePolicyId 元素中的 Transforms 元素
- 不支持 CounterSignature 元素 → 不支持 UnsignedProperties 元素
- 最多一个 DataObjectFormat 元素。多个 DataObjectFormat 元素没有意义，因为我们只有一个已签名的数据对象（这是 XML 签名者端点的传入消息正文）。
- 最多一个 CommitmentTypeIndication 元素。多个 CommitmentTypeIndication 元素没有意义，因为我们只有一个已签名的数据对象（这是 XML 签名者端点的传入消息正文）。
- CommitmentTypeIndication 元素始终包含 AllSignedDataObjects 元素。不支持 CommitmentTypeIndication 元素中的 ObjectReference 元素。
- 不支持 AllDataObjectsTimeStamp 元素
- 不支持 IndividualDataObjectsTimeStamp 元素

374.9. 另请参阅

- 最佳实践

第 375 章 XMPP 组件

从 Camel 版本 1.0 开始提供

xmpp: 组件实现 XMPP (Jabber)传输。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-xmpp</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

375.1. URI 格式

```
xmpp://[login@]hostname[:port][/participant][?Options]
```

组件支持基于房间和私人人员的对话。
组件支持生成者和消费者（您可以从 XMPP 获取消息或发送消息到 XMPP）。消费者模式支持启动空间。

您可以在 URI 中附加查询选项，格式为 ?option=value&option=value&...

375.2. 选项

XMPP 组件没有选项。

XMPP 端点使用 URI 语法进行配置：

```
xmpp:host:port/participant
```

使用以下路径和查询参数：

375.2.1. 路径参数(3 参数)：

Name	描述	默认值	类型
主机	chat 服务器 **所需的** 主机名		字符串
port	chat 服务器 **所需的** 端口号		int
参与者	用于接收消息的个人 JID (Jabber ID)。s room 参数优先于参与者。		字符串

375.2.2. 查询参数(18 参数)：

Name	描述	默认值	类型
login (common)	是否登录用户。	true	布尔值
nickname (common)	加入房间时使用 nickname。如果指定了 room，且不指定 nickname，则用户将用于 nickname。		字符串
pubsub (common)	接受输入上的 pubsub 数据包，默认为 false	false	布尔值
room (common)	如果指定了这个选项，则组件将连接到 MUC (Multi User Chat)。通常，MUC 的域名与登录域不同。例如，如果您是 supermanjabber.org，并希望加入 krypton room，则房间 URL 为 kryptonconference.jabber.org。请注意会议部分。不需要提供整个空间 JID。如果 room 参数不包含符号，则 Camel 将发现和添加域部分		字符串
serviceName (common)	要连接的服务名称。对于 Google Talk，这将是 gmail.com。		字符串
testConnectionOnStartup (common)	指定是否在启动时测试连接。这用于确保在路由启动时 XMPP 客户端具有与 XMPP 服务器的有效连接。如果无法建立连接，Camel 会在启动时抛出异常。当此选项设为 false 时，Camel 会在生成者需要时尝试建立 lazy 连接，并将轮询消费者连接，直到连接建立为止。默认为 true。	true	布尔值
createAccount (common)	如果为 true，则会尝试创建帐户。默认值为 false。	false	布尔值
resource (common)	XMPP 资源.默认为 Camel。	Camel	字符串

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
connectionPollDelay (consumer)	轮询（以秒为单位）之间的时间（以秒为单位），以验证 XMPP 连接的健康状况，或在尝试建立初始消费者连接之间间隔时间（以秒为单位）。如果连接已不活跃，Camel 将尝试重新建立连接。默认值为 10 秒。	10	int
doc (consumer)	在 IN 消息上设置包含传入数据包的 Document 形式的 doc 标头；如果存在或 pubsub 为 true，则默认为 true，否则为 false	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
connectionConfig (advanced)	使用现有的连接配置。目前只支持 org.jivesoftware.smack.tcp.XMPPTCPConnectionConfiguration （通过 TCP 支持 XMPP）。		ConnectionConfiguration
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
headerFilterStrategy (filter)	使用自定义 HeaderFilterStrategy 过滤到 Camel 消息的标头。		HeaderFilterStrategy
密码 (security)	用于登录的密码		字符串
用户 （安全）	用户名（不带服务器名称）。如果没有指定，则会尝试匿名登录。		字符串

375.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.component.xmpp.enabled	启用 xmpp 组件	true	布尔值
camel.component.xmpp.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

375.4. 标头和设置主题或语言

Camel 将消息 IN 标头设置为 XMPP 消息的属性。如果需要自定义过滤标头，您可以配置 HeaderFilterStategy。如果以 IN 标头形式提供，则也设置 XMPP 消息的主题和语言。

375.5. 例子

用户 Superman 使用密码在 jabber 服务器加入房 krypton，secret:

```
xmpp://superman@jabber.org/?room=krypton@conference.jabber.org&password=secret
```

User superman 用来向 joker 发送消息：

```
xmpp://superman@jabber.org/joker@jabber.org?password=secret
```

Java 中的路由示例：

```
from("timer://kickoff?period=10000").
setBody(constant("I will win!\n Your Superman.")).
to("xmpp://superman@jabber.org/joker@jabber.org?password=secret");
```

消费者配置，它将来自 joker 的所有消息写入队列 evil.talk。

```
from("xmpp://superman@jabber.org/joker@jabber.org?password=secret").
to("activemq:evil.talk");
```

消费者配置，侦听房间信息：

```
from("xmpp://superman@jabber.org/?
password=secret&room=krypton@conference.jabber.org").
to("activemq:krypton.talk");
```

简短表示法（无域部分）：

```
from("xmpp://superman@jabber.org/?password=secret&room=krypton").
to("activemq:krypton.talk");
```

当连接到 Google Chat 服务时，您需要指定 serviceName 以及您的凭证：

```
from("direct:start").
  to("xmpp://talk.google.com:5222/touser@gmail.com?
serviceName=gmail.com&user=fromuser&password=secret").
  to("mock:result");
```

375.6. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 376 章 XPATH 语言

从 Camel 版本 1.1 开始提供

Camel 支持 XPath 以允许在 DSL 或 Xml 配置中使用 Expression 或 Predicate。例如，您可以使用 XPath 在 Message Filter 中创建一个 predicates，或作为 Recipient List 的 Expression。

流

如果消息正文基于流，这表示它收到的输入作为流提交给 Camel。这意味着您只能够读取一次流的内容。因此，当您使用 XPath 作为 Message Filter 或 Content Based Router 时，您需要多次访问数据，您应该使用流缓存，或者在之前将消息正文转换为 字符串，以保证多次重新读取。

```
from("queue:foo").
  filter().xpath("//foo")).
  to("queue:bar")
```

```
from("queue:foo").
  choice().xpath("//foo")).to("queue:bar").
  otherwise().to("queue:others");
```

376.1. XPATH 语言选项

XPath 语言支持 9 个选项，如下所列。

Name	默认值	Java 类型	描述
documentType		**字符串**	文档类型的类名称为 org.w3c.dom.Document
resultType	**NODESET**	**字符串**	设置结果类型的类名称（输出中的类型）默认结果类型是 NodeSet
saxon	**false**	**布尔值**	是否使用 Saxon。
factoryRef		**字符串**	对 registry 中要查找的自定义 XPathFactory 的引用
objectModel		**字符串**	要使用的 XPath 对象模型
logNamespaces	**false**	**布尔值**	是否在出现问题时记录命名空间

Name	默认值	Java 类型	描述
headerName		**字符串**	用作输入的标头名称，而不是消息正文
threadSafety	**false**	**布尔值**	是否为 xpath 表达式的返回结果启用 thread-safety。这在使用 NODESET 作为结果类型时适用，返回的集合具有多个元素。在这种情况下，如果您同时处理 NODESET，比如在并行处理模式中从 Camel Splitter EIP 中处理 NODESET，则可能会出现线程安全问题。这个选项通过对节点进行防御副本来防止并发问题。如果您在应用程序中使用 camel-saxon 或 Saxon，则建议打开这个选项。saxon 具有 thread-safety 问题，可以通过打开这个选项来防止。
trim	**true**	**布尔值**	是否修剪值以移除前导和结尾的空格和换行符

376.2. 命名空间

您可以使用 Namespaces 帮助程序类轻松使用带有 XPath 表达式的命名空间。

376.3. 变量

XPath 中的变量在不同的命名空间中定义。默认命名空间是 http://camel.apache.org/schema/spring。

命名空间 URI	本地部分	类型	描述
http://camel.apache.org/xml/in/	in	消息	exchange.in 消息
http://camel.apache.org/xml/out/	out	消息	exchange.out 消息

命名空间 URI	本地部分	类型	描述
http://camel.apache.org/xml/function/	函数	对象	**Camel 2.5:** 其他功能
http://camel.apache.org/xml/variables/environment-variables	env	对象	OS 环境变量
http://camel.apache.org/xml/variables/system-properties	system	对象	Java 系统属性
http://camel.apache.org/xml/variables/exchange-property		对象	交换属性

Camel 将通过以下方式解析变量：

- *给定的命名空间*
- *没有给定的命名空间*

376.3.1. 给定的命名空间

如果给出了命名空间，则 Camel 会精确指示要返回的内容。但是，当解析 或 out Camel 时，将尝试解析给定本地部分的标头，并首先返回它。如果本地部分具有值 body，则返回正文。

376.3.2. 没有给定的命名空间

如果没有给定命名空间，则 Camel 仅根据本地部分解析。Camel 将在以下步骤中尝试解析变量：

- 来自 variables，它使用 variable(name, value) fluent 构建程序设置。
- 来自 message.in.header，如果有一个带有给定键的标头
- 来自 exchange.properties，如果存在一个带有给定键的属性

376.4. FUNCTIONS

Camel 添加以下 XPath 函数，可用于访问交换：

功能	参数	类型	描述
in:body	none	对象	将返回消息正文。
in:header	标头名称	对象	将返回消息标头 **中的**。
out:body	none	对象	将返回 **out** 消息正文。
out:header	标头名称	对象	将返回 **out** 消息标头。
function:properties	属性的键	字符串	**Camel 2.5：** 使用 Properties 组件(property 占位符)查找属性。
function:simple	简单表达式	对象	**Camel 2.5：** 评估 一个简单 表达式。

小心

当返回类型是 NodeSet 时，不支持 function:properties 和 function:simple，比如与 Splitter EIP 搭配使用时。

下面是一个示例，其中显示了一些正在使用的功能。

Camel 2.5 中引入的新功能：

376.5. 使用 XML 配置

如果要在 Spring XML 文件中配置路由，您可以使用 XPath 表达式，如下所示

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
       http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-
spring.xsd">

  <camelContext id="camel" xmlns="http://activemq.apache.org/camel/schema/spring"
xmlns:foo="http://example.com/person">
    <route>
      <from uri="activemq:MyQueue"/>
      <filter>
        <xpath>/foo:person[@name='James']</xpath>
        <to uri="mqseries:SomeOtherQueue"/>
      </filter>
    </route>
  </camelContext>
</beans>
```

请注意，在 XPath 表达式中，我们如何重复使用命名空间前缀 foo，以便更轻松地使用基于命名空间的 XPath 表达式！

另请参阅有关使用带有 xpath 自己的命名空间的邮件列表 的讨论

376.6. 设置结果类型

XPath 表达式将使用原生 XML 对象（如 org.w3c.dom.NodeList ）返回结果类型。但是，您可能希

望结果类型成为 String。要做到这一点，您必须指示要使用的结果类型的 XPath。

Java DSL：

```
xpath("/foo:person/@id", String.class)
```

在 Spring DSL 中，您可以使用 resultType 属性提供完全限定的 classname:

```
<xpath resultType="java.lang.String">/foo:person/@id</xpath>
```

在 @XPath:
中，作为 Camel 2.1 提供

```
@XPath(value = "concat('foo-',//order/name/)", resultType = String.class) String name)
```

我们使用 xpath 函数 concat 为顺序名称添加 foo- 前缀。在这种情况下，我们必须指定一个 String 作为结果类型，因此 concat 功能可以正常工作。

376.7. 在标头上使用 XPATH

从 Camel 2.11 开始提供

有些用户可能将 XML 存储在标头中。要将 XPath 应用到标头的值，您可以通过定义 'headerName' 属性来实现此目的。

在 Java DSL 中，您可以将 headerName 指定为第 2 个参数，如下所示：

```
xpath("/invoice/@orderType = 'premium'", "invoiceDetails")
```

376.8. 例子

以下是在 Message Filter 中使用 XPath 表达式作为 predicate 的简单示例

如果您有一个标准的命名空间集合，并希望在多个不同的 XPath 表达式之间共享它们，您可以使用 NamespaceBuilder，如下例所示

在这个示例中，我们有一个选择的结构。如果消息有一个带有值 Camel 的标头键 类型，则第一个选择。
第二种选择评估消息正文是否具有名称标签 <name >，其值是 Kong。
如果不是 true，则会在其他块中路由信息：

和与路由等效的 spring XML:

376.9. XPATH 注入

您可以使用 Bean Integration 对 bean 调用方法，并使用 XPath 等各种语言从消息中提取值并将其绑定到方法参数。

默认 XPath 注解有 SOAP 和 XML 命名空间。如果要在 XPath 表达式中使用您自己的命名空间 URI，您可以使用您自己的 XPath 注解 副本来创建您要使用的任何命名空间前缀。

在其他软件包和/或注解名称中使用您的注解时，I.e. cut and paste upper code to own project，然后在对 method 参数使用注解时添加您想要的任何命名空间前缀/uri。然后，当您在方法参数上使用注解时，您希望的所有命名空间都可在 XPath 表达式中使用。

例如：

```
public class Foo {

   @MessageDriven(uri = "activemq:my.queue")
   public void doSomething(@MyXPath("/ns1:foo/ns2:bar/text()") String correlationID, @Body
String body) {
      // process the inbound message here
   }
}
```

376.10. 在没有交换的情况下使用 XPATHBUILDER

从 Camel 2.3 开始提供

现在，您可以使用 org.apache.camel.builder.XPathBuilder，而无需交换。如果您要将其用作自定义 xpath 评估的帮助，这会很方便。

它要求您在 CamelContext 中传递，因为 XPathBuilder 中的许多移动部分需要访问 Camel Type Converter，因此为什么需要 CamelContext。

例如，您可以执行以下操作：

```
boolean matches = XPathBuilder.xpath("/foo/bar/@xyz").matches(context, "<foo><bar
xyz='cheese'/></foo>"));
```

这将与给定的 predicate 匹配。

您还可以评估以下示例，如以下三个示例所示：

```
String name = XPathBuilder.xpath("foo/bar").evaluate(context, "<foo><bar>cheese</bar>
</foo>", String.class);
Integer number = XPathBuilder.xpath("foo/bar").evaluate(context, "<foo><bar>123</bar>
</foo>", Integer.class);
Boolean bool = XPathBuilder.xpath("foo/bar").evaluate(context, "<foo><bar>true</bar></foo>",
Boolean.class);
```

使用字符串结果评估是一个常见要求，因此您可以更简单：

```
String name = XPathBuilder.xpath("foo/bar").evaluate(context, "<foo><bar>cheese</bar>
</foo>");
```

376.11. 使用带有 XPATHBUILDER 的 SAXON

从 Camel 2.3 开始提供

您需要将 camel-saxon 作为依赖项添加到项目中。

现在，它比 XPathBuilder 更容易使用 Saxon，它可以通过多种方式实现，如下所示：
其中，因为后者是最简单的。

- 使用工厂

- 使用 ObjectModel

容易一

376.12. 使用系统属性设置自定义 XPATHFACTORY

从 Camel 2.3 开始提供

Camel 现在支持读取 JVM 系统属性 javax.xml.xpath.XPathFactory，可用于设置要使用的自定义 XPathFactory。

单位测试演示了如何进行此操作来使用 Saxon：

如果 Camel 使用非默认的 XPathFactory，则 Camel 将在 INFO 级别记录，例如：

```
XPathBuilder  INFO  Using system property
javax.xml.xpath.XPathFactory:http://saxon.sf.net/jaxp/xpath/om with value:
                net.sf.saxon.xpath.XPathFactoryImpl when creating XPathFactory
```

要使用 Apache Xerces，您可以配置系统属性

```
-Djavax.xml.xpath.XPathFactory=org.apache.xpath.jaxp.XPathFactoryImpl
```

376.13. 从 SPRING DSL 启用 SAXON

从 Camel 2.10 开始提供

与 Java DSL 类似，要在 Spring DSL 中启用 Saxon，您有三个选项：

指定工厂

```
<xpath factoryRef="saxonFactory" resultType="java.lang.String">current-dateTime()</xpath>
```

指定对象模型

```
<xpath objectModel="http://saxon.sf.net/jaxp/xpath/om" resultType="java.lang.String">current-dateTime()</xpath>
```

快捷键

```
<xpath saxon="true" resultType="java.lang.String">current-dateTime()</xpath>
```

376.14. 命名空间审核以协助调试

从 Camel 2.10 开始提供

用户经常面临的大量与 XPath 相关的问题与命名空间使用相关联。您消息中存在的命名空间与 XPath 表达式了解或引用的命名空间之间可能有一些不正确。XPath predicates 或表达式无法找到 XML 元素和属性，因为命名空间问题可能只是像"它们无法正常工作"，实际上它并没有命名空间定义。

XML 中的命名空间是完全必要的，而我们希望通过自动实施一些魔法或 voodoo 来简化其使用情况，但事实上是，该路径关闭了标准，并会大大阻碍了互操作性。

因此，我们几乎可以通过向 XPath 表达式语言添加两个新功能来协助您调试这些问题，因此可以从 predicates 和 表达式中轻松实现。

#=== 记录您的 XPath 表达式/predicate 的命名空间上下文

每次在内部池中创建新的 XPath 表达式时，Camel 都会记录 org.apache.camel.builder.xml.XPathBuilder 日志记录器下的表达式的命名空间上下文。由于 Camel 以分级方式表示命名空间上下文（父子关系），整个树都是以递归方式的输出，其格式如下：

```
[me: {prefix -> namespace}, {prefix -> namespace}], [parent: [me: {prefix -> namespace}, {prefix -> namespace}], [parent: [me: {prefix -> namespace}]]]
```

任何这些选项都可用于激活此日志：

1. 在 org.apache.camel.builder.xml.XPathBuilder logger 或一些父日志记录器（如 org.apache.camel 或根日志记录器）上启用 TRACE 日志记录

2. 启用 logNamespaces 选项，如 Auditing Namespaces 所示，在这种情况下，日志记录将在 INFO 级别上发生

376.15. 审计命名空间

在评估 XPath 表达式前，Camel 能够发现和转储每个传入消息上存在的所有命名空间，从而为您提供有助于您分析和固定命名空间问题所需的所有丰富的信息。

要达到此目的，它在内部使用另一个特殊定制的 XPath 表达式来提取消息中显示的所有命名空间映射，显示每个单个映射的前缀和完整的命名空间 URI。

需要考虑的一些点：

- 隐式 XML 命名空间(xmlns:xml="http://www.w3.org/XML/1998/namespace")被禁止在输出中，因为它没有添加值
- 默认命名空间在输出中的 DEFAULT 关键字下列出
- 请记住，可以将命名空间重新映射到不同的范围下。认为顶级 'a' 前缀可以分配不同的命名空间，或者默认命名空间在内部范围内更改。对于每个发现的前缀，会列出所有关联的 URI。

您可以在 Java DSL 和 Spring DSL 中启用这个选项。

Java DSL:

```
XPathBuilder.xpath("/foo:person/@id", String.class).logNamespaces()
```

Spring DSL:

```
<xpath logNamespaces="true" resultType="String">/foo:person/@id</xpath>
```

审计的结果将显示在 org.apache.camel.builder.xml.XPathBuilder 日志记录器下的 INFO 级别，如

下所示：

```
2012-01-16 13:23:45,878 [stSaxonWithFlag] INFO  XPathBuilder  - Namespaces discovered in
message:
{xmlns:a=[http://apache.org/camel], DEFAULT=[http://apache.org/default],
xmlns:b=[http://apache.org/camelA, http://apache.org/camelB]}
```

376.16. 从外部资源载入脚本

从 Camel 2.11 开始提供

您可以对脚本进行外部化，并让 Camel 从资源（如 "classpath:"、"file:" 或 "http:" ）加载它。这可以通过以下语法完成：" resource:scheme:location"，例如引用您可以进行的类路径上的文件：

```
.setHeader("myHeader").xpath("resource:classpath:myxpath.txt", String.class)
```

376.17. 依赖项

XPath 语言是 camel-core 的一部分。

第 377 章 XQUERY 组件

从 Camel 版本 1.0 开始提供

Camel 支持 XQuery 允许在 DSL 或 Xml 配置中使用 Expression 或 Predicate。例如，您可以使用 XQuery 在 Message Filter 中创建一个 predicates，或作为 Recipient List 的 Expression。

377.1. 选项

XQuery 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
ModuleURIResolver (advanced)	使用自定义 ModuleURIResolver		ModuleURIResolver
配置 （高级）	使用自定义 Saxon 配置		配置
configurationProperties (advanced)	设置自定义 Saxon 配置属性		Map
resolvePropertyPlaceholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

XQuery 端点使用 URI 语法进行配置：

```
xquery:resourceUri
```

使用以下路径和查询参数：

377.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceUri	需要从 classpath 或 file 载入的模板名称		字符串

377.1.2. 查询参数(31 参数)：

Name	描述	默认值	类型
allowStAX (common)	是否允许使用 StAX 模式	false	布尔值
headerName (common)	使用 Camel Message 标头作为输入源，而不是 Message body。		字符串
namespacePrefixes (common)	允许控制要用于一组命名空间映射的命名空间前缀		Map
resultsFormat (common)	要使用的输出结果	DOM	ResultFormat
resultType (common)	将什么输出结果定义为类		类
stripsAllWhiteSpace (common)	是否剥离所有空格	true	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
配置 （高级）	使用自定义 Saxon 配置		配置

Name	描述	默认值	类型
configurationProperties (advanced)	设置自定义 Saxon 配置属性		Map
ModuleURIResolver (advanced)	使用自定义 ModuleURIResolver		ModuleURIResolver
参数 (advanced)	其他参数		Map
属性 (advanced)	配置序列化参数的属性		Properties
staticQueryContext (advanced)	使用自定义 Saxon StaticQueryContext		StaticQueryContext
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	500	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service

Name	描述	默认值	类型
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

377.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 8 个选项，如下所列。

Name	描述	默认值	类型
camel.component.xquery.configuration	使用自定义 Saxon 配置。选项是一个 net.sf.saxon.Configuration 类型。		字符串
camel.component.xquery.configuration-properties	设置自定义 Saxon 配置属性		Map
camel.component.xquery.enabled	启用 xquery 组件	true	布尔值
camel.component.xquery.module-u-r-i-resolver	使用自定义 ModuleURIResolver。选项是一个 net.sf.saxon.lib.ModuleURIResolver 类型。		字符串
camel.component.xquery.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.language.xquery.enabled	启用 xquery 语言	true	布尔值

Name	描述	默认值	类型
camel.language.xquery.trim	是否修剪值以移除前导和结尾的空格和换行符	true	布尔值
camel.language.xquery.type	设置结果类型的类名称（输出中的类型）默认结果类型是 NodeSet		字符串

377.3. 例子

```
from("queue:foo").filter().
  xquery("//foo").
  to("queue:bar")
```

您还可以使用查询中的功能，在这种情况下，您需要一个明确的类型转换（或者，您需要一个 org.w3c.dom.DOMException: HIERARCHY_REQUEST_ERR）将类作为第二个参数传递给 xquery（） 方法。

```
from("direct:start").
  recipientList().xquery("concat('mock:foo.', /person/@city)", String.class);
```

377.4. 变量

IN 消息正文将设置为 contextItem。除了这些变量外，也添加了参数：

变量	类型	描述
交换	Exchange	当前的交换
in.body	对象	In 消息正文
out.body	对象	OUT 消息的正文（如果有）
in.headers.*	对象	您可以使用名称 in.headers.foo 的变量，使用键 **foo** 来访问 exchange.in.headers 的值
out.headers.*	对象	您可以使用名称为 out.headers.foo 变量，使用键 **foo** 来访问 exchange.out.headers 的值

变量	类型	描述
键名称	对象	任何 exchange.properties 和 Exchange.in.headers，以及使用 **setParameters (Map)** 设置的任何其他参数。这些参数使用它们自己的密钥名称添加，例如，如果存在带有密钥名称 **foo** 的 IN 标头，则其添加为 **foo**。

377.5. 使用 XML 配置

如果要在 Spring XML 文件中配置路由，您可以使用 XPath 表达式，如下所示

```
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:foo="http://example.com/person"
      xsi:schemaLocation="
      http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
      http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-
spring.xsd">

  <camelContext id="camel" xmlns="http://activemq.apache.org/camel/schema/spring">
    <route>
      <from uri="activemq:MyQueue"/>
      <filter>
        <xquery>/foo:person[@name='James']</xquery>
        <to uri="mqseries:SomeOtherQueue"/>
      </filter>
    </route>
  </camelContext>
</beans>
```

请注意，在 XPath 表达式中，我们如何重复使用命名空间前缀 foo，以便更轻松地使用基于命名空间的 XQuery 表达式！

当您在 XQuery 表达式中使用函数时，您需要一个明确的类型转换，该转换是在 xml 配置中通过 @type 属性完成的：

```
    <xquery type="java.lang.String">concat('mock:foo.', /person/@city)</xquery>
```

377.6. 使用 XQUERY 作为转换

我们可以在路由中使用 transform or setBody 来对消息进行转换，如下所示：

```
from("direct:start").
   transform().xquery("/people/person");
```

-

请注意，xquery 将默认使用 DOMResult，因此如果我们希望获取 person 节点的值，则需要使用 text （）告知 xquery 使用 String 作为结果类型，如下所示：

```
from("direct:start").
  transform().xquery("/people/person/text()", String.class);
```

377.7. 使用 XQUERY 作为端点

有时，XQuery 表达式可能会非常大，它可以被用于 Templating。因此，您可能希望使用 XQuery 端点，以便您可以使用 XQuery 模板路由。

以下示例演示了如何获取 ActiveMQ 队列(MyQueue)的消息，并使用 XQuery 将其发送到 MQSeries。

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="activemq:MyQueue"/>
    <to uri="xquery:com/acme/someTransform.xquery"/>
    <to uri="mqseries:SomeOtherQueue"/>
  </route>
</camelContext>
```

目前 XQuery 中的自定义功能可能会导致 NullPointerException (Camel 2.18, 2.19 和 2.20)。这应该在 Camel 2.21 中解决

377.8. 例子

以下是在 Message Filter 中使用 XQuery 表达式作为 predicate 的简单示例

本例在 Message Filter 中使用 XQuery 和 namespace 作为 predicate

377.9. 学习 XQUERY

XQuery 是一个非常强大的语言，用于查询、搜索、排序和返回 XML。如需学习 XQuery，可以使用这些教程

- Mike Kay 的 XQuery Primer
- W3Schools XQuery 教程

您可能还发现 XQuery 功能参考 很有用

377.10. 从外部资源载入脚本

从 Camel 2.11 开始提供

您可以对脚本进行外部化，并让 Camel 从资源（如 "classpath:"、"file:" 或 "http:" ）加载它。这可以通过以下语法完成：" resource:scheme:location"，例如引用您可以进行的类路径上的文件：

```
.setHeader("myHeader").xquery("resource:classpath:myxquery.txt", String.class)
```

377.11. 依赖项

要在 camel 路由中使用 XQuery，您需要添加实现 XQuery 语言的 camel-saxon 的依赖项。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-saxon</artifactId>
  <version>x.x.x</version>
</dependency>
```

第 378 章 XSLT COMPONENT

从 Camel 版本 1.3 开始提供

xslt: 组件允许您使用 XSLT 模板处理消息。这在使用 Templating 生成请求的重新填充时是理想的选择。

378.1. URI 格式

```
xslt:templateName[?options]
```

URI 格式包含 templateName，它可以是以下之一：

- 要调用的模板的 classpath-local URI
- 远程模板的完整 URL。

您可以使用以下格式将查询选项附加到 URI 中：

```
?option=value&option=value&...
```

如需 URI 语法的更多详情，请参阅 Spring 文档。

表 378.1. URI 示例

URI	描述
xslt:com/acme/mytransform.xsl	在 classpath 上引用 com/acme/mytransform.xsl 文件
xslt:file:///foo/bar.xsl	指的是文件 /foo/bar.xsl
xslt:http://acme.com/cheese/foo.xsl	引用远程 http 资源

对于 Camel 2.8 或更早的版本，Maven 用户需要将以下依赖项添加到组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-spring</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

从 Camel 2.9 开始，XSLT 组件直接在 camel-core 中提供。

378.2. 选项

XSLT 组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
xmlConverter (advanced)	使用 org.apache.camel.converter.jaxp.XmlConverter 的自定义实施		XmlConverter
uriResolverFactory (advanced)	使用自定义 UriResolver，它依赖于动态端点资源 URI。不应与选项 'uriResolver' 一起使用。		XsltUriResolverFactory
uriResolver (advanced)	使用自定义 UriResolver。不应与选项 'uriResolverFactory' 一起使用。		URIResolver
contentCache (producer)	加载资源内容（样式表文件）的缓存。如果设置为 false Camel，则每个消息处理都会重新载入风格表文件。这非常适合开发。可使用 clearCachedStylesheet 操作强制通过 JMX 重新加载缓存风格表。	true	布尔值
saxon (producer)	是否使用 Saxon 作为 transformerFactoryClass。如果启用，则类 net.sf.saxon.TransformerFactoryImpl。您需要将 Saxon 添加到 classpath。	false	布尔值
saxonExtensionFunctions (advanced)	允许您使用自定义 net.sf.saxon.lib.ExtensionFunctionDefinition。您需要将 camel-saxon 添加到 classpath。函数在 registry 中查找，您可以用逗号分隔多个要查找的值。		字符串
saxonConfiguration (advanced)	使用自定义 Saxon 配置		对象
saxonConfiguration Properties (advanced)	设置自定义 Saxon 配置属性		Map

Name	描述	默认值	类型
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

XSLT 端点使用 URI 语法进行配置：

```
xslt:resourceUri
```

使用以下路径和查询参数：

378.2.1. 路径参数(1 参数)：

Name	描述	默认值	类型
resourceUri	**必需的**模板的路径。默认 URIResolver 支持以下内容。您可以使用这些协议(classpath 为 default)使用：classpath、file、http、ref 或 bean. classpath、file 和 http 加载资源的前缀。ref 将查找 registry 中的资源。bean 将调用 bean 以供用作资源的方法。对于 bean，您可以在点后指定方法名称，如 bean:myBean.myMethod		字符串

378.2.2. 查询参数(17 参数)：

Name	描述	默认值	类型
allowStAX (producer)	是否允许将 StAX 用作 javax.xml.transform.Source。	true	布尔值
contentCache (producer)	加载资源内容（样式表文件）的缓存。如果设置为 false Camel，则每个消息处理都会重新载入风格表文件。这非常适合开发。可使用 clearCachedStylesheet 操作强制通过 JMX 重新加载缓存风格表。	true	布尔值
deleteOutputFile (producer)	如果您有 output=file，则此选项指明在 Exchange 完成后是否应该删除输出文件。例如，假设输出文件是临时文件，然后在使用后可以将其删除。	false	布尔值
failOnNullBody (producer)	如果输入正文为 null，是否抛出异常。	true	布尔值

Name	描述	默认值	类型
output (producer)	用于指定要使用的输出类型的选项。可能的值有：string, bytes, DOM, file。前三个选项均基于内存中，其中文件将直接流传输到 java.io.File。对于文件，您必须使用 Exchange.XSLT_FILE_NAME （也是 CamelXsltFileName ）在 IN 标头中指定文件名。另外，必须先创建指向文件名的任何路径，否则会在运行时抛出异常。	string	XsltOutput
saxon (producer)	是否使用 Saxon 作为 transformerFactoryClass。如果启用，则类 net.sf.saxon.TransformerFactoryImpl。您需要将 Saxon 添加到 classpath。	false	布尔值
transformerCacheSize (producer)	被缓存用于重复使用的 javax.xml.transform.Transformer 对象数量，以避免调用 Template.newTransformer （）。	0	int
converter (advanced)	使用 org.apache.camel.converter.jaxp.XmlConverter 的自定义实施		XmlConverter
entityResolver (advanced)	使用自定义 org.xml.sax.EntityResolver 和 javax.xml.transform.sax.SAXSource。		EntityResolver
errorListener (advanced)	允许配置 以使用自定义 javax.xml.transform.ErrorListener。这样做时，请注意，当执行此操作时，会捕获任何错误或严重错误并在交换上存储信息，因为属性不在使用中。因此，仅将这个选项用于特殊用例。		ErrorListener
resultHandlerFactory (advanced)	允许您使用自定义 org.apache.camel.builder.xml.ResultHandlerFactory ，它能够使用自定义 org.apache.camel.builder.xml.ResultHandler 类型。		ResultHandlerFactory
saxonConfiguration (advanced)	使用自定义 Saxon 配置		对象
saxonExtensionFunctions (advanced)	允许您使用自定义 net.sf.saxon.lib.ExtensionFunctionDefinition。您需要将 camel-saxon 添加到 classpath。函数在 registry 中查找，您可以用逗号分隔多个要查找的值。		字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
transformerFactory (advanced)	使用自定义 XSLT 转换程序工厂		TransformerFactory

Name	描述	默认值	类型
transformerFactoryClass (advanced)	要使用自定义 XSLT 转换程序工厂，指定为 FQN 类名称		字符串
uriResolver (advanced)	使用自定义 javax.xml.transform.URIResolver		URIResolver

378.3. 使用 XSLT 端点

以下格式是使用 XSLT 模板为 InOut 消息交换发送响应（其中有 JMSReplyTo 标头）的 expample。

```
from("activemq:My.Queue").
  to("xslt:com/acme/mytransform.xsl");
```

如果要使用 InOnly 并使用信息并将其发送到另一个目的地，您可以使用以下路由：

```
from("activemq:My.Queue").
  to("xslt:com/acme/mytransform.xsl").
  to("activemq:Another.Queue");
```

378.4. 在 XSLT 中获取 USEABLE 参数

默认情况下，所有标头都添加为 XSLT 中提供的参数。
要使参数可用，您需要声明这些参数。

```
<setHeader headerName="myParam"><constant>42</constant></setHeader>
<to uri="xslt:MyTransform.xsl"/>
```

该参数还需要在 XSLT 的顶级中声明，以便其可用：

```
<xsl: ..... >

  <xsl:param name="myParam"/>

   <xsl:template ...>
```

378.5. SPRING XML 版本

要使用 Spring XML 中的以上示例，您可以使用类似以下代码的内容：

```
<camelContext xmlns="http://activemq.apache.org/camel/schema/spring">
  <route>
    <from uri="activemq:My.Queue"/>
    <to uri="xslt:org/apache/camel/spring/processor/example.xsl"/>
    <to uri="activemq:Another.Queue"/>
  </route>
</camelContext>
```

要查看示例，请查看 测试案例 及其 Spring XML。

378.6. 使用 XSL:INCLUDE

Camel 2.2 或旧的
如果您在 Camel 2.2 或更早的 XSL 文件中使用 xsl:include，则会使用默认的 javax.xml.transform.URIResolver。文件将相对于 JVM 启动文件夹解析。

例如，以下 include 语句将查找从启动应用的文件夹开始的 staff_template.xsl 文件。

```
<xsl:include href="staff_template.xsl"/>
```

Camel 2.3 或更新版本
for Camel 2.3 或更新版本, Camel 提供了自己的 URIResolver 实施。这允许 Camel 从 classpath 加载包含的文件。

例如，以下代码中的 include 文件将相对于起始端点：

```
<xsl:include href="staff_template.xsl"/>
```

这意味着，Camel 将在 classpath 中查找文件，存为 org/apache/camel/component/xslt/staff_template.xsl

您可以使用 classpath: 或 file: 来指示 Camel 在 classpath 或文件系统中查找。如果省略前缀，则 Camel 将使用端点配置中的前缀。如果在端点配置中没有指定前缀，则默认为 classpath:。

您还可以在 include 路径中引用。在以下示例中，xsl 文件将在 org/apache/camel/component 下解

析。

```
    <xsl:include href="../staff_other_template.xsl"/>
```

378.7. 使用 XSL:INCLUDE 和默认前缀

在 Camel 2.10.3 及更高版本中，classpath: 用作默认前缀。
如果您使用 file 配置启动资源以加载：，则后续所有目标都将必须以 文件作为前缀：。

从 Camel 2.10.4 中，Camel 将使用端点配置的前缀作为默认前缀。

您可以明确指定 file: 或 classpath: loading。如果需要，可以在 XSLT 脚本中混合两种加载类型。

378.8. 使用 SAXON 扩展功能

从 Saxon 9.2 开始，编写扩展功能已被新机制补充，它称为集成 扩展功能，您可以轻松使用 camel，如下例所示：

```
SimpleRegistry registry = new SimpleRegistry();
registry.put("function1", new MyExtensionFunction1());
registry.put("function2", new MyExtensionFunction2());

CamelContext context = new DefaultCamelContext(registry);
context.addRoutes(new RouteBuilder() {
   @Override
   public void configure() throws Exception {
      from("direct:start")
         .to("xslt:org/apache/camel/component/xslt/extensions/extensions.xslt?
saxonExtensionFunctions=#function1,#function2");
   }
});
```

使用 Spring XML：

```
<bean id="function1" class="org.apache.camel.component.xslt.extensions.MyExtensionFunction1"/>
<bean id="function2" class="org.apache.camel.component.xslt.extensions.MyExtensionFunction2"/>

<camelContext xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="direct:extensions"/>
    <to uri="xslt:org/apache/camel/component/xslt/extensions/extensions.xslt?
```

```
saxonExtensionFunctions=#function1,#function2"/>
  </route>
</camelContext>
```

378.9. 动态风格表

要在运行时提供动态风格表，您可以定义动态 URI。如需更多信息，请参阅 如何在 to （）中使用动态 URI。

从 Camel 2.9 开始（在 2.11.4, 2.12.3 和 2.13.0）中删除了）
Camel 提供了 CamelXsltResourceUri 标头，您可以使用它来定义端点 URI 上配置的替代风格表。这可让您在运行时提供动态风格表。

378.10. 从 XSLT ERRORLISTENER 访问警告、错误和严重错误

从 Camel 2.14 开始提供

从 Camel 2.14 中，任何警告/错误或致命错误都存储在当前交换中，作为包含密钥 Exchange.XSLT_ERROR、Exchange.XSLT_FATAL_ERROR 或 Exchange.XSLT_WARNING 的属性存储，允许最终用户在转换过程中获取任何错误。

例如，在下面的风格表中，如果员工有一个空的 dob 字段，我们希望终止。和 使用 xsl:message 包括自定义错误消息。

```
<xsl:template match="/">
  <html>
    <body>
      <xsl:for-each select="staff/programmer">
        <p>Name: <xsl:value-of select="name"/><br />
          <xsl:if test="dob="">
            <xsl:message terminate="yes">Error: DOB is an empty string!</xsl:message>
          </xsl:if>
        </p>
      </xsl:for-each>
    </body>
  </html>
</xsl:template>
```

例外保存在 Exchange 中，作为密钥 Exchange.XSLT_WARNING 的警告。

378.11. 使用 XSLT 和 JAVA 版本的备注

以下是来自 Sameer （一个 Camel 用户）的一些观察结果，该用户完全与我们共享：

如果任何人都遇到 XSLT 端点的问题，请查看这些点。

我试图使用 xslt 端点使用简单的 xsl 从一个 xml 到另一个的转换。输出 xml 保持显示（在路由中的 xslt 处理器后），其中最外部的 xml 标签中没有内容。

DEBUG 日志中没有显示任何解释。但是，在 TRACE 日志中发现一些错误/警告表示 XMLConverter bean 无法初始化。

在经过几小时后，我必须进行以下操作才能正常工作（因为用户论坛会给用户论坛给出一些线索）：

1. 使用路由中的 transformerFactory 选项 ("xslt:my-transformer.xsl?transformerFactory=tFactory") 以及 tFactory bean 为 class="org.apache.xalan.xsltc.trax.TransformerFactoryImpl".

2. 将 Xalan jar 添加到 my maven pom 中。

my guess 是 JDK （我使用 1.6.0_03）提供的默认 xml 解析机制无法在此上下文中正常工作，且不会抛出任何错误。当我通过这种方式切换到 Xalan 时。这不是 Camel 问题，但可能需要在 xslt 组件页面中提到。

另一个备注，jdk 1.6.0_03 附带 JAXB 2.0，而 Camel 需要 2.1。一个临时解决方案是，将 2.1 jar 添加到 jvm 的 jre/lib/endorsed 目录中，或者由容器指定的。

希望此后期可以节省一些时间。

378.12. 另请参阅

- 配置 Camel

- *组件*
- *端点*
- *开始使用*

第 379 章 XSTREAM DATAFORMAT

从 Camel 版本 1.3 开始提供

Xstream 是一个 Data Format，它使用 XStream 库来聚合和来自 XML 的 Java 对象。

要在 camel 路由中使用 XStream，您需要添加对实现此数据格式的 camel-xstream 的依赖关系。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-xstream</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

379.1. 选项

XStream dataformat 支持 10 个选项，如下所列。

Name	默认值	Java 类型	描述
权限		字符串	添加控制在 unmarshal from xml/json 到 Java Bean 期间允许使用哪些 Java 软件包和类 XStream 的权限。必须在此处或全局使用 JVM 系统属性配置权限。权限可以使用加号允许的语法指定，减号是 deny。使用 . 作为前缀来支持通配符。例如，允许 com.foo 和所有子软件包，然后 specfy com.foo.。可以用逗号分隔多个权限，如 com.foo.,-com.foo.bar.MySecretBean。以下默认权限始终被包括：-,java.lang.,java.util.，除非其使用键 org.apache.camel.xstream.permissions 指定 JVM 系统属性来覆盖它。
编码		字符串	设置要使用的编码
driver		字符串	使用自定义 XStream 驱动程序。实例必须是 com. althoughtworks.xstream.io.HierarchicalStreamDriver 的类型
driverRef		字符串	在 registry 中引用要查询的自定义 XStream 驱动程序。实例必须是 com. althoughtworks.xstream.io.HierarchicalStreamDriver 的类型

Name	默认值	Java 类型	描述
模式		**字符串**	处理重复引用的模式有：NO_REFERENCES ID_REFERENCES XPATH_REFERENCES XPATH_REFERENCES XPATH_ABSLUTE_REFERENCES SINGLE_NODE_XPATH_RELATIVE_REFERENCES SINGLE_NODE_XPATH_REFOLUTE_REFERENCES SINGLE_REFERENCES SINGLE_REFERENCES SINGINGLE_REFERENCES SINGLE_REFERENCES XPATH_REFERENCES SINGLE_REFERENCES XPATH_REFERENCES XPATH_REFERENCES XPATH_REFERENCES XPATH_REFERENCES XPATH
转换器		**list**	使用自定义 XStream 转换器的类名称列表。类必须是 com.althoughtworks.xstream.converters.Converter
别名		**Map**	将类别名为在 XML 元素中使用的短名称。
omitFields		**Map**	防止字段被序列化。若要省略一个字段，您必须始终提供声明类型，而不一定是转换的类型。
implicitCollections		**Map**	添加用于任何未映射 XML 标签的默认隐式集合。
contentTypeHeader	**false**	**布尔值**	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

379.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 31 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.json-xstream.allow-jms-type	用于 JMS 用户，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。	false	布尔值
camel.dataformat.json-xstream.allow-unmarshall-type	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。只有在需要使用时，才会启用。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.json-xstream.collection-type-name	引用要在要使用的 registry 中的自定义集合类型。这个选项很少被使用，但允许使用不同的集合类型，而不是基于 java.util.Collection 作为默认值。		字符串
camel.dataformat.json-xstream.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.json-xstream.disable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的一组功能。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称		字符串
camel.dataformat.json-xstream.enable-features	要在 Jackson com.fasterxml.jackson.databind.ObjectMapper 中启用的功能集。这个功能应该是与来自 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature 中的多个功能匹配的名称		字符串
camel.dataformat.json-xstream.enable-jaxb-annotation-module	使用 jackson 时是否启用 JAXB 注释模块。启用后，Jackson 可以使用 JAXB 注释。	false	布尔值
camel.dataformat.json-xstream.enabled	启用 json-xstream dataformat	true	布尔值
camel.dataformat.json-xstream.include	如果您要将 pojo 放入 JSON，并且 pojo 有一些带有 null 值的字段。如果您想要跳过这些 null 值，您可以将这个选项设置为 NON_NULL		字符串
camel.dataformat.json-xstream.json-view	当将 POJO 聚合到 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。此选项是引用具有 JsonView 注释的类		类

Name	描述	默认值	类型
camel.dataformat.json-xstream.library	要使用哪个 json 库。		JsonLibrary
camel.dataformat.json-xstream.module-class-names	要使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String 带有 FQN 类名称。可以使用逗号分隔多个类。		字符串
camel.dataformat.json-xstream.module-refs	使用 Camel registry 引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。		字符串
camel.dataformat.json-xstream.object-mapper	在使用 Jackson 时，查找并使用给定 id 的现有 ObjectMapper。		字符串
camel.dataformat.json-xstream.permissions	添加控制在 unmarshal from xml/json 到 Java Bean 期间允许使用哪些 Java 软件包和类 XStream 的权限。必须在此处或全局使用 JVM 系统属性配置权限。权限可以使用加号允许的语法指定，减号是 deny。使用 . 作为前缀来支持通配符。例如，允许 com.foo 和所有子软件包，然后 specfy com.foo.。可以用逗号分隔多个权限，如 com.foo.,-com.foo.bar.MySecretBean。以下默认权限始终被包括： -,java.lang.,java.util.，除非其使用键 org.apache.camel.xstream.permissions 指定 JVM 系统属性来覆盖它。		字符串
camel.dataformat.json-xstream.pretty-print	要启用用户的打印输出，请执行以下操作：默认为 false。	false	布尔值
camel.dataformat.json-xstream.timezone	如果设置，则 Jackson 会在 marshalling/unmarshalling 时使用 Timezone。此选项对其他 Json DataFormat （如 gson、fastjson 和 xstream）没有影响。		字符串
camel.dataformat.json-xstream.unmarshal-type-name	取消警报时要使用的 java 类型的类名称		字符串

Name	描述	默认值	类型
camel.dataformat.json-xstream.use-default-object-mapper	是否从 registry 中查找和使用默认的 Jackson ObjectMapper。	true	布尔值
camel.dataformat.json-xstream.use-list	要取消警报到映射列表或 Pojo 列表。	false	布尔值
camel.dataformat.xstream.aliases	将类别名为在 XML 元素中使用的短名称。		Map
camel.dataformat.xstream.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.xstream.converters	使用自定义 XStream 转换器的类名称列表。类必须是 com. althoughtworks.xstream.converters.Converter		list
camel.dataformat.xstream.driver	使用自定义 XStream 驱动程序。实例必须是 com. althoughtworks.xstream.io.HierarchicalStreamDriver 的类型		字符串
camel.dataformat.xstream.driver-ref	在 registry 中引用要查询的自定义 XStream 驱动程序。实例必须是 com. althoughtworks.xstream.io.HierarchicalStreamDriver 的类型		字符串
camel.dataformat.xstream.enabled	启用 xstream dataformat	true	布尔值
camel.dataformat.xstream.encoding	设置要使用的编码		字符串
camel.dataformat.xstream.implicit-collections	添加用于任何未映射 XML 标签的默认隐式集合。		Map

Name	描述	默认值	类型
camel.dataformat.xstream.mode	处理重复引用的模式有：NO_REFERENCES ID_REFERENCES XPATH_REFERENCES XPATH_REFERENCES XPATH_ABSLUTE_REFERENCES SINGLE_NODE_XPATH_RELATIVE_REFERENCES SINGLE_NODE_XPATH_REFOLUTE_REFERENCES SINGLE_REFERENCES SINGLE_REFERENCES SINGINGLE_REFERENCES SINGLE_REFERENCES XPATH_REFERENCES SINGLE_REFERENCES XPATH_REFERENCES XPATH_REFERENCES XPATH_REFERENCES XPATH_REFERENCES XPATH		字符串
camel.dataformat.xstream.omit-fields	防止字段被序列化。若要省略一个字段，您必须始终提供声明类型，而不一定是转换的类型。		Map
camel.dataformat.xstream.permissions	添加控制在 unmarshal from xml/json 到 Java Bean 期间允许使用哪些 Java 软件包和类 XStream 的权限。必须在此处或全局使用 JVM 系统属性配置权限。权限可以使用加号允许的语法指定，减号是 deny。使用 . 作为前缀来支持通配符。例如，允许 com.foo 和所有子软件包，然后 specfy com.foo.。可以用逗号分隔多个权限，如 com.foo.,-com.foo.bar.MySecretBean。以下默认权限始终被包括： -,java.lang.,java.util.，除非其使用键 org.apache.camel.xstream.permissions 指定 JVM 系统属性来覆盖它。		字符串

ND

379.3. 使用 JAVA DSL

```
// lets turn Object messages into XML then send to MQSeries
from("activemq:My.Queue").
  marshal().xstream().
  to("mqseries:Another.Queue");
```

如果要配置 Camel 用于消息转换的 XStream 实例，您只需在 DSL 级别传递对该实例的引用。

```
XStream xStream = new XStream();
xStream.aliasField("money", PurchaseOrder.class, "cash");
// new Added setModel option since Camel 2.14
xStream.setModel("NO_REFERENCES");
...
```

```
from("direct:marshal").
  marshal(new XStreamDataFormat(xStream)).
  to("mock:marshaled");
```

379.4. XMLINPUTFACTORY 和 XMLOUTPUTFACTORY

XStream 库使用 javax.xml.stream.XMLInputFactory 和 javax.xml.stream.XMLOutputFactory，您可以控制应该使用哪个工厂实施。

使用这个算法发现 Factory: 1。使用 javax.xml.stream.XMLInputFactory , javax.xml.stream.XMLOutputFactory 系统属性。2.使用 JRE_HOME 目录中的 lib/xml.stream.properties 文件。3.使用 Services API （如果有）来通过查看 META-INF/services/javax.xml.stream.XMLInputFactory,META-INF/services/javax.xml.stream.XMLOutputFactory 文件来决定 classname。4.使用平台默认 XMLInputFactory、XMLOutputFactory 实例。

379.5. 如何在 XSTREAM DATAFORMAT 中设置 XML 编码？

在 Camel 2.2.0 中，您可以使用密钥 Exchange.CHARSET_NAME 设置 Exchange 的属性，或者在 DSL 或 Spring 配置上设置编码属性，在 Xstream DataFormat 中设置 XML 编码。

```
from("activemq:My.Queue").
  marshal().xstream("UTF-8").
  to("mqseries:Another.Queue");
```

379.6. 设置 XSTREAM DATAFORMAT 的类型权限

在 Camel 中，始终使用路由中的自己的处理步骤来过滤和阻止某些 XML 文档路由到 XStream 的 unmarhall 步骤。从 Camel 2.16.1、2.15.5，您可以设置 XStream 的类型权限，以自动允许或拒绝某些类型的实例化。

Camel 使用的默认类型权限设置拒绝所有类型，但 java.lang 和 java.util 软件包中除外。此设置可以通过设置系统属性 org.apache.camel.xstream.permissions 来更改。其值是一个以逗号分隔的权限术语字符串，每个术语代表允许或拒绝的类型，具体取决于术语是前缀为 " （请注意 " 可能会被省略）还是分别带有 '-'。

每个术语都可以包含一个通配符字符 "。例如，值 "-,java.lang.,java.util." 表示拒绝除 java.lang unset 和 java.util targeted 类以外的所有类型。将此值设置为空字符串 ""，会恢复到默认的 XStream 类型权限处理，该处理拒绝某些黑名单类并允许其他用户。

通过设置 type permissions 属性，可以在单独的 XStream DataFormat 实例上扩展 type 权限设置。

```
<dataFormats>
  <xstream id="xstream-default"
           permissions="org.apache.camel.samples.xstream.*"/>
  ...
```

第 380 章 YAML SNAKEYAML DATAFORMAT

从 Camel 版本 2.17 开始提供

YAML 是一个 Data Format to marshal and unmarshal Java objects to and from YAML。

对于对象的 marshalling，Camel 提供了与三个流行的 YAML 库集成：

- SnakeYAML 库

每个库都需要添加特殊的 camel 组件（请参阅"Dependency..."段落进一步停机）。默认情况下，Camel 使用 SnakeYAML 库。

380.1. YAML 选项

YAML SnakeYAML dataformat 支持 11 个选项，如下所列。

Name	默认值	Java 类型	描述
程序库	SnakeYAML	YAMLLibrary	要使用哪个 yaml 库。默认为 SnakeYAML
unmarshalTypeName		字符串	取消警报时要使用的 java 类型的类名称
constructor		字符串	构建传入文档的 BaseConstructor。
representer		字符串	用于发出传出对象的 Representer。
dumperOptions		字符串	DumperOptions，用于配置传出对象。
resolver		字符串	用于检测隐式类型的解析器
useApplicationContextClassLoader	true	布尔值	使用 ApplicationContextClassLoader 作为自定义 ClassLoader
prettyFlow	false	布尔值	强制发出者在使用流风格时生成用户 YAML 文档。

Name	默认值	Java 类型	描述
allowAnyType	false	布尔值	允许任何类变为 un-marshaled
typeFilter		list	将类型 SnakeYAML 设置为 un-marshall
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

380.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 12 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.yaml-snakeyaml.allow-any-type	允许任何类变为 un-marshaled	false	布尔值
camel.dataformat.yaml-snakeyaml.constructor	构建传入文档的 BaseConstructor。		字符串
camel.dataformat.yaml-snakeyaml.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.yaml-snakeyaml.dumper-options	DumperOptions，用于配置传出对象。		字符串
camel.dataformat.yaml-snakeyaml.enabled	启用 yaml-snakeyaml dataformat	true	布尔值
camel.dataformat.yaml-snakeyaml.library	要使用哪个 yaml 库。默认为 SnakeYAML		YAMLLibrary

Name	描述	默认值	类型
camel.dataformat.yaml-snakeyaml.pretty-flow	强制发出者在使用流风格时生成用户 YAML 文档。	false	布尔值
camel.dataformat.yaml-snakeyaml.representer	用于发出传出对象的 Representer。		字符串
camel.dataformat.yaml-snakeyaml.resolver	用于检测隐式类型的解析器		字符串
camel.dataformat.yaml-snakeyaml.type-filter	将类型 SnakeYAML 设置为 un-marshall		list
camel.dataformat.yaml-snakeyaml.unmarshal-type-name	取消警报时要使用的 java 类型的类名称		字符串
camel.dataformat.yaml-snakeyaml.use-application-context-class-loader	使用 ApplicationContextClassLoader 作为自定义 ClassLoader	true	布尔值

警告

SnakeYAML 可以从 YAML 定义加载任何类，这些类可能会导致安全漏洞，SnakeYAML DataForma 将对象限制为标准 Java 对象，如 List 或 Long。如果要加载自定义 POJO，则需要将其类型添加到 SnakeYAML DataFormat type filter 列表中。如果您的源是可信的，您可以将属性 allowAnyType 设置为 true，以便 SnakeYAML DataForma 不会在类型上执行任何过滤器。

380.3. 在 SNAKEYAML 库中使用 YAML 数据格式

- 将对象消息转换为 yaml，然后发送到 MQ 系列

```
from("activemq:My.Queue")
  .marshal().yaml()
  .to("mqseries:Another.Queue");
```

```
from("activemq:My.Queue")
  .marshal().yaml(YAMLLibrary.SnakeYAML)
  .to("mqseries:Another.Queue");
```

- 限制要从 YAML 加载的类

```
// Creat a SnakeYAMLDataFormat instance
SnakeYAMLDataFormat yaml = new SnakeYAMLDataFormat();

// Restrict classes to be loaded from YAML
yaml.addTypeFilters(TypeFilters.types(MyPojo.class, MyOtherPojo.class));

from("activemq:My.Queue")
  .unmarshal(yaml)
  .to("mqseries:Another.Queue");
```

380.4. 在 SPRING DSL 中使用 YAML

在 Spring DSL 中使用 Data Format 时，您需要首先声明数据格式。这是在 DataFormats XML 标签中完成的。

```
<dataFormats>
  <!--
    here we define a YAML data format with the id snake and that it should use
    the TestPojo as the class type when doing unmarshal. The unmarshalTypeName
    is optional
  -->
  <yaml
    id="snake"
    library="SnakeYAML"
    unmarshalTypeName="org.apache.camel.component.yaml.model.TestPojo"/>

  <!--
    here we define a YAML data format with the id snake-safe which restricts the
    classes to be loaded from YAML to TestPojo and those belonging to package
    com.mycompany
  -->
  <yaml id="snake-safe">
    <typeFilter value="org.apache.camel.component.yaml.model.TestPojo"/>
    <typeFilter value="com.mycompany\..*" type="regexp"/>
  </yaml>
</dataFormats>
```

然后您可以在路由中引用这些 id：

```
<route>
  <from uri="direct:unmarshal"/>
  <unmarshal>
    <custom ref="snake"/>
  </unmarshal>
  <to uri="mock:unmarshal"/>
</route>
<route>
  <from uri="direct:unmarshal-safe"/>
  <unmarshal>
    <custom ref="snake-safe"/>
  </unmarshal>
  <to uri="mock:unmarshal-safe"/>
</route>
```

380.5. SNAKEYAML 的依赖项

要在 camel 路由中使用 YAML，您需要添加对实现此数据格式的 camel-snakeyaml 的依赖。

如果您使用 maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-snakeyaml</artifactId>
  <version>${camel-version}</version>
</dependency>
```

第 381 章 YAMMER 组件

从 Camel 版本 2.12 开始提供

Yammer 组件允许您与 Yammer 企业社交网络进行交互。支持消耗消息、用户和用户关系以及创建新消息。

Yammer 使用 OAuth 2 进行所有客户端应用程序身份验证。要将 camel-yammer 与您的帐户搭配使用，您需要在 Yammer 内创建一个新应用程序，并授予应用程序对您的帐户的访问权限。最后，生成您的访问令牌。更多详情请参阅 https://developer.yammer.com/authentication/。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-yammer</artifactId>
  <version>${camel-version}</version>
</dependency>
```

381.1. URI 格式

```
yammer:[function]?[options]
```

381.2. 组件选项

Yammer 组件可以使用 Yammer 帐户设置进行配置，在使用前必须进行配置。

Yammer 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
consumerKey (security)	消费者密钥		字符串
consumerSecret (security)	消费者 secret		字符串
accessToken (security)	访问令牌		字符串

Name	描述	默认值	类型
config (advanced)	使用共享的 yammer 配置		YammerConfiguration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

您还可以直接在端点中配置这些选项。

381.3. 端点选项

Yammer 端点使用 URI 语法进行配置：

```
yammer:function
```

使用以下路径和查询参数：

381.3.1. 路径参数(1 参数)：

Name	描述	默认值	类型
function	所需的功能		YammerFunctionType

381.3.2. 查询参数(28 参数)：

Name	描述	默认值	类型
useJson (common)	如果要使用原始 JSON，而不是转换为 POJO，则设置为 true。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值

Name	描述	默认值	类型
delay (consumer)	millis 中轮询之间的延迟	5000	long
limit (consumer)	仅返回指定数量的消息。适用于 threaded=true 和 threaded=extended。	-1	int
newerThan (consumer)	返回比指定为数字字符串的消息 ID 的消息。这在轮询新消息时应使用。如果您正在查看消息，并且返回的最新消息为 3516，您可以使用参数 newerThan=3516 发出请求，以确保您不会获得页面中已存在的消息的副本。	-1	int
olderThan (consumer)	返回比指定为数字字符串的消息 ID 旧的消息。这对于分页消息非常有用。例如，如果您当前查看了 20 个消息，并且最旧的消息为 2912，您可以在您的请求中附加 olderThan=2912 以获得您看到的 20 个消息。	-1	int
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
threaded (consumer)	threaded=true 将只返回每个线程中的第一个消息。此参数适用于显示消息线程折叠的应用程序。threaded=extended 将按最近激活的线程启动程序消息以及最近两个最新消息的顺序返回，因为它们在 Yammer Web 界面的默认视图中查看。		字符串
userId (consumer)	用户 ID		字符串
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
pollStrategy (consumer)	可插拔 org.apache.camel.PollingConsumerPollingStrategy 允许您提供自定义实施来控制在轮询操作期间通常会发生错误处理，然后再创建交换并在 Camel 中路由。		PollingConsumerPoll Strategy
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int

Name	描述	默认值	类型
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。您还可以使用单位来指定时间值，如 60s (60 秒)、5m30s (5 分钟和 30 秒)，以及 1h (1 小时)。	1000	long
runLoggingLevel (scheduler)	消费者在轮询时记录 start/complete log 行。这个选项允许您为其配置日志级别。	TRACE	LoggingLevel
scheduledExecutorService (scheduler)	允许配置用于消费者的自定义/共享线程池。默认情况下，每个使用者都有自己的单线程线程池。		ScheduledExecutor Service
scheduler (scheduler)	使用 camel-spring 或 camel-quartz2 组件的 cron 调度程序	none	ScheduledPollConsumer Scheduler
schedulerProperties (scheduler)	在使用自定义调度程序或任何基于 Spring 的调度程序时配置附加属性。		Map
startScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。	MILLISECONDS	TimeUnit
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值
accessToken (security)	**所需的** 访问令牌		字符串
consumerKey (security)	**所需的** 消费者密钥		字符串
consumerSecret (security)	**所需的** 消费者 secret		字符串

381.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 18 个选项，如下所列。

Name	描述	默认值	类型
camel.component.yammer.access-token	访问令牌		字符串
camel.component.yammer.config.access-token	访问令牌		字符串
camel.component.yammer.config.consumer-key	消费者密钥		字符串
camel.component.yammer.config.consumer-secret	消费者 secret		字符串
camel.component.yammer.config.delay	millis 中轮询之间的延迟	5000	Long
camel.component.yammer.config.function	要使用的功能		字符串
camel.component.yammer.config.function-type	要使用的功能		YammerFunctionType
camel.component.yammer.config.limit	仅返回指定数量的消息。适用于 threaded=true 和 threaded=extended。	-1	整数
camel.component.yammer.config.newer-than	返回比指定为数字字符串的消息 ID 的消息。这在轮询新消息时应使用。如果您正在查看消息，并且返回的最新消息为 3516，您可以使用参数 newerThan=3516 发出请求，以确保您不会获得页面中已存在的消息的副本。	-1	整数
camel.component.yammer.config.older-than	返回比指定为数字字符串的消息 ID 旧的消息。这对于分页消息非常有用。例如，如果您当前查看了 20 个消息，并且最旧的消息为 2912，您可以在您的请求中附加 olderThan=2912 以获得您看到的 20 个消息。	-1	整数

Name	描述	默认值	类型
camel.component.yammer.config.requestor			ApiRequestor
camel.component.yammer.config.threaded	threaded=true 将只返回每个线程中的第一个消息。此参数适用于显示消息线程折叠的应用程序。threaded=extended 将按最近激活的线程启动程序消息以及最近两个最新消息的顺序返回，因为它们在 Yammer Web 界面的默认视图中查看。		字符串
camel.component.yammer.config.use-json	如果要使用原始 JSON，而不是转换为 POJO，则设置为 true。	false	布尔值
camel.component.yammer.config.user-id	用户 ID		字符串
camel.component.yammer.consumer-key	消费者密钥		字符串
camel.component.yammer.consumer-secret	消费者 secret		字符串
camel.component.yammer.enabled	启用 yammer 组件	true	布尔值
camel.component.yammer.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

381.5. 使用消息

Yammer 组件为使用消息提供了几个端点：

URI	描述
yammer:messages?[options]	用户（其访问令牌被用来发出 API 调用）Yammer 网络中的所有公共消息。对应于 Yammer Web 界面中的"所有"对话。

URI	描述
yammer:my_feed?[options]	用户源，基于在"Follow"和"Top"对话之间所做的选择。
yammer:algo?[options]	与"Top"对话对应的用户的算法源，这是大多数用户将在 Yammer web 界面中看到的。
yammer:following?[options]	"Following"源，对话涉及用户以下的人员、组和主题。
yammer:sent?[options]	用户发送的所有消息。
yammer:private?[options]	用户接收的专用消息。
yammer:received?[options]	Camel 2.12.1 ： 用户收到的所有信息

381.5.1. 消息格式

默认情况下，所有消息都会转换为 org.apache.camel.component.yammer.model 软件包提供的 POJO 模型。来自 yammer 的原始消息采用 JSON。对于所有使用和生成端点的消息，会返回 Messages 对象。以如下方式为路由：

```
from("yammer:messages?
consumerKey=aConsumerKey&consumerSecret=aConsumerSecretKey&accessToken=aAcce
ssToken")
   .to("mock:result");
```

让我们说 yammer 服务器返回：

```
{
   "messages":[
      {
         "replied_to_id":null,
         "network_id":7654,
         "url":"https://www.yammer.com/api/v1/messages/305298242",
         "thread_id":305298242,
         "id":305298242,
         "message_type":"update",
         "chat_client_sequence":null,
         "body":{
```

```
            "parsed":"Testing yammer API...",
            "plain":"Testing yammer API...",
            "rich":"Testing yammer API..."
         },
         "client_url":"https://www.yammer.com/",
         "content_excerpt":"Testing yammer API...",
         "created_at":"2013/06/25 18:14:45 +0000",
         "client_type":"Web",
         "privacy":"public",
         "sender_type":"user",
         "liked_by":{
            "count":1,
            "names":[
               {
                  "permalink":"janstey",
                  "full_name":"Jonathan Anstey",
                  "user_id":1499642294
               }

            ]

         },
         "sender_id":1499642294,
         "language":null,
         "system_message":false,
         "attachments":[

         ],
         "direct_message":false,
         "web_url":"https://www.yammer.com/redhat.com/messages/305298242"
      },
      {
         "replied_to_id":null,
         "network_id":7654,
         "url":"https://www.yammer.com/api/v1/messages/294326302",
         "thread_id":294326302,
         "id":294326302,
         "message_type":"system",
         "chat_client_sequence":null,
         "body":{
            "parsed":"(Principal Software Engineer) has [[tag:14658]] the redhat.com network.
Take a moment to welcome Jonathan.",
            "plain":"(Principal Software Engineer) has #joined the redhat.com network. Take a
moment to welcome Jonathan.",
            "rich":"(Principal Software Engineer) has #joined the redhat.com network. Take a
moment to welcome Jonathan."
         },
         "client_url":"https://www.yammer.com/",
         "content_excerpt":"(Principal Software Engineer) has #joined the redhat.com network.
Take a moment to welcome Jonathan.",
         "created_at":"2013/05/10 19:08:29 +0000",
         "client_type":"Web",
         "sender_type":"user",
         "privacy":"public",
         "liked_by":{
            "count":0,
```

```
            "names":[

            ]

        }
    }
    ]

}
```

Camel 将在包含 2 个 Message 对象的 Messages 对象中放入一个 Messages 对象中。如下所示，有一个丰富的对象模型，可以轻松地获取您需要的任何信息：

```
Exchange exchange = mock.getExchanges().get(0);
Messages messages = exchange.getIn().getBody(Messages.class);

assertEquals(2, messages.getMessages().size());
assertEquals("Testing yammer API...", messages.getMessages().get(0).getBody().getPlain());
assertEquals("(Principal Software Engineer) has #joined the redhat.com network. Take a
moment to welcome Jonathan.", messages.getMessages().get(1).getBody().getPlain());
```

这意味着，将这个数据嵌套到 POJO 中不是空闲的，因此如果您需要您可以通过将 useJson=false 选项添加到 URI 来切回到使用纯 JSON 的 JSON。

381.6. 创建消息

要在当前用户帐户的帐户中创建新消息，您可以使用以下 URI：

```
yammer:messages?[options]
```

当前的 Camel 消息正文用于设置 Yammer 消息的文本。响应正文将包括与使用消息相同的方式（默认为 Messages 对象）的新消息。

为实例获取此路由：

```
from("direct:start")
   .to("yammer:messages?
consumerKey=aConsumerKey&consumerSecret=aConsumerSecretKey&accessToken=aAcce
ssToken")
   .to("mock:result");
```

通过发送到 direct:start 端点 "Hi from Camel!" 消息正文：

```
template.sendBody("direct:start", "Hi from Camel!");
```

将在服务器上的当前用户帐户中创建一个新消息，同时此新消息将返回到 Camel，并转换为 Messages 对象。例如，当您消耗消息时，您可以评估 Messages 对象：

```
Exchange exchange = mock.getExchanges().get(0);
Messages messages = exchange.getIn().getBody(Messages.class);

assertEquals(1, messages.getMessages().size());
assertEquals("Hi from Camel!", messages.getMessages().get(0).getBody().getPlain());
```

381.7. 检索用户关系

Yammer 组件可以检索用户关系：

```
yammer:relationships?[options]
```

381.8. 检索用户

Yammer 组件为检索用户提供多个端点：

URI	描述
yammer:users?[options]	在当前用户的 Yammer 网络中检索用户。
yammer:current?[options]	查看当前用户的数据。

381.9. 使用增强

有时（或者在用户或关系消费者时）使用增强模式而不是以 camel-yammer 为其中一个轮询用户启动的路由会很有用。这是因为用户会重复触发，但通常会为其设置延迟。如果您只想查找用户数据，或者一次获取消息，最好使用路由调用该消费者一次。

假设您有一个在某个时间点上需要退出并获取当前用户的用户数据的路由。使用 pollEnrich DSL 方法，而不是再次轮询此用户：

```
from("direct:start")
```

```
    .pollEnrich("yammer:current?
consumerKey=aConsumerKey&consumerSecret=aConsumerSecretKey&accessToken=aAcce
ssToken")
    .to("mock:result");
```

这将退出并获取当前用户的 User 对象，并将其设置为 Camel 消息正文。

381.10. 另请参阅

- 配置 Camel
- 组件
- 端点
- 开始使用

第 382 章 ZENDESK 组件

从 Camel 版本 2.19 开始提供

Zendesk 组件提供对使用 zendesk-java-client 访问的所有 zendesk.com API 的访问权限。它允许生成消息来管理 Zendesk ticket、用户、机构等。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-zendesk</artifactId>
  <version>${camel-version}</version>
</dependency>
```

382.1. ZENDESK 选项

Zendesk 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	使用共享配置		ZendeskConfiguration
zendesk (advanced)	使用共享 Zendesk 实例。		Zendesk
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

Zendesk 端点使用 URI 语法进行配置：

```
zendesk:methodName
```

使用以下路径和查询参数：

382.1.1. 路径参数(1 参数)：

Name	描述	默认值	类型
methodName	**需要使用** 什么操作		字符串

382.1.2. 查询参数(10 parameters):

Name	描述	默认值	类型
inBody (common)	设置要在交换 In Body 中传递的参数名称		字符串
serverUrl (common)	要连接的服务器 URL。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值
oauthToken (security)	OAuth 令牌。		字符串
密码 (security)	密码。		字符串
令牌 (security)	安全令牌。		字符串
用户名 (security)	用户名。		字符串

382.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
camel.component.zendesk.configuration.method-name	要使用什么操作		字符串
camel.component.zendesk.configuration.oauth-token	OAuth 令牌。		字符串
camel.component.zendesk.configuration.password	密码。		字符串
camel.component.zendesk.configuration.server-url	要连接的服务器 URL。		字符串
camel.component.zendesk.configuration.token	安全令牌。		字符串
camel.component.zendesk.configuration.username	用户名。		字符串
camel.component.zendesk.enabled	启用 zendesk 组件	true	布尔值
camel.component.zendesk.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.zendesk.zendesk	使用共享 Zendesk 实例。选项是一个 org.zendesk.client.v2.Zendesk 类型。		字符串

382.3. URI 格式

```
zendesk://endpoint?[options]
```

382.4. 生产者端点：

生产者端点可以使用端点名称和关联的选项。

382.5. 消费者端点：

任何制作者端点都可以用作消费者端点。消费者端点可以使用 Scheduled Poll Consumer Options 和 consumer. 前缀来调度端点调用。返回数组或集合的消费者端点将为每个元素生成一个交换，它们的路由将为每个交换执行一次。

382.6. 消息标头

任何选项都可以在带有 CamelZendesk. 前缀的制作者端点的消息标头中提供。在 principal 中，参数名称与原始 org.zendesk.client.v2.Zendesk 类中每个 API 方法的 arugument 名称相同。但是，其中一些被重命名为其他名称，以避免 camel API 组件框架中的冲突。要查看实际参数名称，请检查 org.apache.camel.component.zendesk.internal.ZendeskApiMethod。

382.7. 消息正文

所有结果消息正文都使用 Zendesk Java Client 提供的对象。生产者端点可以在 inBody 端点参数中指定传入消息正文的选项名称。

第 383 章 ZIP DEFLATE COMPRESSION DATAFORMAT

从 Camel 版本 2.12 开始提供

Zip Data Format 是消息压缩和解压缩格式。使用 Zip 压缩消息的 marshalled 可以在端点中使用 Zip decompression，然后再使用 Zip decompression。当您处理大量 XML 和基于文本的有效负载时，压缩功能非常有用。它有助于更好地使用网络带宽，同时降低成本，以便在端点压缩和解压缩有效负载。

INFO:*About with with Files* The Zip data format, does not (yet)对文件有特殊支持。这意味着，在使用大型文件时，整个文件内容都会加载到内存中。这可能会在以后改变，以便基于流的解决方案具有较少的内存占用。

383.1. 选项

Zip Deflate Compression dataformat 支持 2 个选项，如下所列。

Name	默认值	Java 类型	描述
compressionLevel	-1	整数	要指定 0-9 之间的特定压缩。-1 是默认的压缩，0 代表压缩，9 是最佳压缩。
contentTypeHeader	false	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

383.2. MARSHAL

在本例中，我们将常规文本/XML 有效负载分成压缩有效负载，使用 zip 压缩 Deflater.BEST_COMPRESSION，并将其发送一个名为 MY_QUEUE 的 ActiveMQ 队列。

```
from("direct:start").marshal().zip(Deflater.BEST_COMPRESSION).to("activemq:queue:MY_QUEUE");
```

或者，如果要使用默认设置，您可以将它发送为

```
from("direct:start").marshal().zip().to("activemq:queue:MY_QUEUE");
```

383.3. UNMARSHAL

在本例中，我们从名为 MY_QUEUE 的 ActiveMQ 队列到其原始格式，并将它转发到 UnZippedMessageProcessor，并将它转发到 UnZippedMessageProcessor。请注意，在 marshalling 期间使用的压缩级别应该与在 unmarshalling 期间使用的相同，以避免错误。

```
from("activemq:queue:MY_QUEUE").unmarshal().zip().process(new
UnZippedMessageProcessor());
```

383.4. 依赖项

此数据格式以 camel-core 提供，因此不需要额外的依赖项。

第 384 章 ZIP 文件数据格式

从 Camel 版本 2.11 开始提供

Zip File Data Format 是消息压缩和解压缩格式。可以将消息嵌套到包含单个条目的 Zip 文件，而包含单个条目的 Zip 文件也可 unmarshalled （解压缩）到原始文件内容。此数据格式支持 ZIP64，只要使用 Java 7 或更高版本]。

384.1. ZIPFILE 选项

Zip File dataformat 支持 4 个选项，如下所列。

Name	默认值	Java 类型	描述
usingIterator	**false**	布尔值	如果 zip 文件有多个条目，则将此选项设置为 true，允许使用 splitter EIP 来分割数据，在流传输模式中使用迭代器。
allowEmptyDirectory	**false**	布尔值	如果 zip 文件有多个条目，则将此选项设置为 true，即使目录为空，也可以获取迭代器
preservePathElements	**false**	布尔值	如果文件名包含路径元素，请将此选项设置为 true，允许在 zip 文件中维护路径。
contentTypeHeader	**false**	布尔值	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。

384.2. SPRING BOOT AUTO-CONFIGURATION

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.zipfile.allow-empty-directory	如果 zip 文件有多个条目，则将此选项设置为 true，即使目录为空，也可以获取迭代器	false	布尔值

Name	描述	默认值	类型
camel.dataformat.zipfile.content-type-header	如果数据格式可以这样做，则数据格式是否应使用 data 格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 放入 XML 或用于数据格式的 application/json，如 JSon 等。	false	布尔值
camel.dataformat.zipfile.enabled	启用 zipfile dataformat	true	布尔值
camel.dataformat.zipfile.preserve-path-elements	如果文件名包含路径元素，请将此选项设置为 true，允许在 zip 文件中维护路径。	false	布尔值
camel.dataformat.zipfile.using-iterator	如果 zip 文件有多个条目，则将此选项设置为 true，允许使用 splitter EIP 来分割数据，在流传输模式中使用迭代器。	false	布尔值

ND

384.3. MARSHAL

在本例中，我们使用 Zip 文件压缩将常规文本/XML 有效负载放入压缩有效负载，并将它发送到名为 MY_QUEUE 的 ActiveMQ 队列。

```
from("direct:start")
    .marshal().zipFile()
    .to("activemq:queue:MY_QUEUE");
```

创建的 Zip 文件中的 Zip 条目的名称基于传入的 CamelFileName 消息标头，这是文件组件使用的标准消息标头。另外，传出 CamelFileName 消息标头会自动设置为传入的 CamelFileName 消息标头的值，带有 ".zip" 后缀。例如，如果以下路由在输入目录中找到名为 "test.txt" 的文件，则输出将是名为 "test.txt.zip" 的 Zip 文件，其中包含一个名为 "test.txt" 的 Zip 条目：

```
from("file:input/directory?antInclude=*/.txt")
    .marshal().zipFile()
    .to("file:output/directory");
```

如果没有传入的 CamelFileName 消息标头（例如，如果文件组件不是消费者），则默认使用消息 ID，因为消息 ID 通常是唯一生成的 ID，因此您将以文件名（如 ID-MACHINENAME-2443-1211718892437-1-0.zip ）。如果要覆盖此行为，您可以在路由中显式设置 CamelFileName 标头的值：

```
from("direct:start")
    .setHeader(Exchange.FILE_NAME, constant("report.txt"))
    .marshal().zipFile()
    .to("file:output/directory");
```

这个路由会导致输出目录中名为 "report.txt.zip" 的 Zip 文件，其中包含一个名为 "report.txt" 的 Zip 条目。

384.4. UNMARSHAL

在本例中，我们将名为 MY_QUEUE 的 ActiveMQ 队列中的 Zip 文件有效负载转发到其原始格式，并将它转发到 UnZippedMessageProcessor。

```
from("activemq:queue:MY_QUEUE")
    .unmarshal().zipFile()
    .process(new UnZippedMessageProcessor());
```

如果 zip 文件有多个条目，则 ZipFileDataFormat 的 useIterator 选项为 true，您可以使用 splitter 来进一步工作。

```
ZipFileDataFormat zipFile = new ZipFileDataFormat();
zipFile.setUsingIterator(true);

from("file:src/test/resources/org/apache/camel/dataformat/zipfile/?
consumer.delay=1000&noop=true")
    .unmarshal(zipFile)
    .split(body(Iterator.class)).streaming()
        .process(new UnZippedMessageProcessor())
    .end();
```

或者，您可以使用 ZipSplitter 作为直接分割器的表达式，如下所示

```
from("file:src/test/resources/org/apache/camel/dataformat/zipfile?
consumer.delay=1000&noop=true")
    .split(new ZipSplitter()).streaming()
        .process(new UnZippedMessageProcessor())
    .end();
```

384.5. 聚合

注意

请注意，此聚合策略需要强制完成检查才能正常工作。

在这个示例中，我们将在输入目录中找到的所有文本文件聚合到一个 Zip 文件中，该文件存储在输出目录中。

```
from("file:input/directory?antInclude=*/.txt")
   .aggregate(constant(true), new ZipAggregationStrategy())
      .completionFromBatchConsumer().eagerCheckCompletion()
      .to("file:output/directory");
```

传出 CamelFileName 消息标头使用 java.io.File.createTempFile 创建，后缀为 ".zip"。如果要覆盖此行为，您可以在路由中显式设置 CamelFileName 标头的值：

```
from("file:input/directory?antInclude=*/.txt")
   .aggregate(constant(true), new ZipAggregationStrategy())
      .completionFromBatchConsumer().eagerCheckCompletion()
      .setHeader(Exchange.FILE_NAME, constant("reports.zip"))
      .to("file:output/directory");
```

384.6. 依赖项

要在 camel 路由中使用 Zip 文件，您需要添加对实现此数据格式的 camel-zipfile 的依赖关系。

如果使用 Maven，您只需在 pom.xml 中添加以下内容，替换最新和最佳发行版本的版本号（请参阅最新版本的下载页面）。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-zipfile</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

384.7. ZIPKIN 组件

从 Camel 2.18 开始提供

camel-zipkin 组件用于使用 zipkin 跟踪和传出 Camel 消息。

为要发送到/来自 Camel 的传入和传出消息捕获事件(span)。

注意

计划在 Camel 2.22.0 中重构 Camel-zipkin，使其不使用 zipkin-scribe，但使用默认的 http 传输。这种工作可能会导致向后兼容性。

这意味着您需要配置哪些 Camel 端点映射到 zipkin 服务名称。

映射可以使用以下方法配置：

- Route id - Camel route id
- 端点 url - Camel 端点 url

对于这两种类型，您可以使用通配符和正则表达式匹配，使用来自 Intercept 的规则。

要匹配所有 Camel 信息，您可以在模式中使用 *，并将这些信息配置为相同的服务名称。

如果没有配置映射，Camel 将回退并使用 endpoint uri's 作为服务名称。
但是，建议配置服务映射，以便您可以在名称中使用人类可读的名称而不是 Camel 端点 uris。

Camel 将自动配置一个 span reporter 没有明确配置，如果主机名和端口已配置为 zipkin 收集器

- ZIPKIN_COLLECTOR_HTTP_SERVICE_HOST - http 主机名
- ZIPKIN_COLLECTOR_HTTP_SERVICE_PORT - 端口号

or

- *ZIPKIN_COLLECTOR_THRIFT_SERVICE_HOST - Scribe (Thrift RPC) hostname*
- *ZIPKIN_COLLECTOR_THRIFT_SERVICE_PORT - 端口号*

这样可轻松在容器平台中使用 camel-zipkin，平台可在 linux 容器中运行应用程序，其中服务配置作为环境变量提供。

384.8. 选项

选项	默认	描述
rate	1.0f	配置一个决定 zipkin 应追踪多少事件的速率。该速率以百分比表示(1.0f = 100%，0.5f 为 50%，0.1f 为 10%)。
spanReporter		**必需：** 用于将 zipkin span 事件发送到 zipkin 服务器的报告程序。
serviceName		使用与所有 Camel 事件匹配的全局服务名称
clientServiceMappings		将与 Camel 事件匹配 **的客户端** 服务映射设置为给定的 zipkin 服务名称。内容是一个 Map<String, String>，其中键是一个模式，值是服务名称。该模式使用来自 Intercept 的规则。
serverServiceMappings		将与 Camel 事件匹配的 **服务器** 服务映射设置为给定的 zipkin 服务名称。内容是一个 Map<String, String>，其中键是一个模式，值是服务名称。该模式使用来自 Intercept 的规则。
excludePatterns		使用 zipkin 为与模式匹配的 Camel 消息禁用追踪的排除模式。内容是一个 Set<String>，其中键是一个模式。该模式使用来自 Intercept 的规则。
includeMessageBody	false	是否在 zipkin 跟踪中包含 Camel 消息正文。不建议在生产环境中使用，或者有大型有效负载。您可以通过配置最大调试日志大小来限制 大小。

选项	默认	描述
includeMessageBodyStreams	false	是否包含基于 zipkin 跟踪的消息正文。这需要在 CamelContext 上或全局启用流缓存。不建议在生产环境中使用，或者有大型有效负载。您可以通过配置最大调试日志大小来限制 大小。

384.9. SPRING BOOT AUTO-CONFIGURATION

组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
camel.zipkin.client-service-mappings	设置将 Camel 事件与给定 zipkin 服务名称匹配的客户端服务映射。键是模式，值是服务名称。		Map
camel.zipkin.endpoint	为 zipkin 的 v2 api 设置 POST URL，通常为 "http://zipkinhost:9411/api/v2/spans"		字符串
camel.zipkin.exclude-patterns	使用 zipkin 为与模式匹配的 Camel 消息禁用追踪的排除模式。		Set
camel.zipkin.host-name	如果向远程 zipkin scribe (thrift RPC)收集器发送 span，则设置主机名。		字符串
camel.zipkin.include-message-body	是否在 zipkin 跟踪中包含 Camel 消息正文。不建议在生产环境中使用，或者有大型有效负载。您可以通过配置 camel.springboot.log-debug-max-chars 选项来限制大小。	false	布尔值
camel.zipkin.include-message-body-streams	是否包含基于 zipkin 跟踪的消息正文。不建议在生产环境中使用，或者有大型有效负载。您可以通过配置 camel.springboot.log-debug-max-chars 选项来限制大小。	false	布尔值
camel.zipkin.port	如果向远程 zipkin scribe (thrift RPC)收集器发送 span，则设置端口。	0	整数
camel.zipkin.rate	配置一个决定 zipkin 应追踪多少事件的速率。该速率以百分比表示(1.0f = 100%，0.5f 为 50%，0.1f 为 10%)。	1	浮点值

Name	描述	默认值	类型
camel.zipkin.server-service-mappings	设置将 Camel 事件与给定 zipkin 服务名称匹配的服务器服务映射。键是模式，值是服务名称。		Map
camel.zipkin.service-name	使用与所有 Camel 事件匹配的全局服务名称		字符串

384.10. EXAMPLE

要启用 camel-zipkin，您需要首先配置

```
ZipkinTracer zipkin = new ZipkinTracer();
// Configure a reporter, which controls how often spans are sent
//   (the dependency is io.zipkin.reporter2:zipkin-sender-okhttp3)
sender = OkHttpSender.create("http://127.0.0.1:9411/api/v2/spans");
zipkin.setSpanReporter(AsyncReporter.create(sender));
// and then add zipkin to the CamelContext
zipkin.init(camelContext);
```

以上配置将跟踪 Camel 路由中的所有传入和传出消息。

要在 XML 中使用 ZipkinTracer，您只需要定义 scribe 和 zipkin tracer Bean。Camel 将自动发现并使用它们。

```
<!-- configure how to reporter spans to a Zipkin collector
       (the dependency is io.zipkin.reporter2:zipkin-reporter-spring-beans) -->
<bean id="http" class="zipkin2.reporter.beans.AsyncReporterFactoryBean">
  <property name="sender">
    <bean id="sender" class="zipkin2.reporter.beans.OkHttpSenderFactoryBean">
      <property name="endpoint" value="http://localhost:9411/api/v2/spans"/>
    </bean>
  </property>
  <!-- wait up to half a second for any in-flight spans on close -->
  <property name="closeTimeout" value="500"/>
</bean>

<!-- setup zipkin tracer -->
<bean id="zipkinTracer" class="org.apache.camel.zipkin.ZipkinTracer">
  <property name="serviceName" value="dude"/>
  <property name="spanReporter" ref="http"/>
</bean>
```

384.10.1. ServiceName

但是，如果要将 Camel 端点映射到人友好的逻辑名称，您可以添加映射

- ServiceName *

您可以配置所有事件都将回退和使用的全局服务名称，例如：

```
zipkin.setServiceName("invoices");
```

这将为所有传入和传出的 zipkin 跟踪使用相同的服务名称。如果您的应用程序使用不同的服务，您应该将它们映射到更精细的客户端/服务器服务映射

384.10.2. 客户端和服务器服务映射

- ClientServiceMappings
- ServerServiceMappings

如果您的应用程序托管了其他人可以调用的服务，您可以将 Camel 路由端点映射到服务器服务映射。例如，假设您的 Camel 应用程序具有以下路由：

```
from("activemq:queue:inbox")
  .to("http:someserver/somepath");
```

您要将其作为服务器服务，您可以添加以下映射：

```
zipkin.addServerServiceMapping("activemq:queue:inbox", "orders");
```

然后，当从该 inbox 队列中使用消息时，它将成为带有服务名称"orders"的 zipkin 服务器事件。

现在假设调用 http:someserver/somepath 也是一个服务，您要映射到客户端服务名称，这可以作为以下内容完成：

```
zipkin.addClientServiceMapping("http:someserver/somepath", "audit");
```

然后，在同一 Camel 应用程序中将传入和传出端点映射到不同的 zipkin 服务名称。

您可以在服务映射中使用通配符。要将所有传出调用与同一 HTTP 服务器匹配，您可以执行以下操作：

```
zipkin.addClientServiceMapping("http:someserver*", "audit");
```

384.11. 映射规则

使用以下规则进行服务器的服务名称映射

1. 是否有与 from 端点的端点 uri 匹配的排除模式？如果是，则跳过。
2. serviceServiceMapping 中是否有匹配项，它与来自端点的端点 uri 匹配？如果为 yes，则使用找到的服务名称
3. serviceServiceMapping 中是否有与当前路由的路由 ID 匹配？如果为 yes，则使用找到的服务名称
4. serviceServiceMapping 中是否有匹配项，与交换启动的原始路由 id 匹配？如果为 yes，则使用找到的服务名称
5. 未找到服务名称，无法按 zipkin 跟踪交换

客户端的服务名称映射使用以下规则

1. 是否有与 from 端点的端点 uri 匹配的排除模式？如果是，则跳过。
2. clientServiceMapping 中是否有匹配项，与消息发送到的端点的 endpoint uri 匹配？如果为 yes，则使用找到的服务名称
3. clientServiceMapping 中是否有与当前路由的路由 ID 匹配？如果为 yes，则使用找到的服务名称

4. clientServiceMapping 中是否有匹配项，与交换启动的原始路由 id 匹配？如果为 yes，则使用找到的服务名称

5. 未找到服务名称，无法按 zipkin 跟踪交换

384.11.1. 没有客户端或服务器映射

如果没有配置客户端或服务器服务映射，CamelZipkin 以回退模式运行，并使用 endpoint uris 作为服务名称。

在上例中，这意味着如果您自己添加以下代码，则会定义服务名称：

```
zipkin.addServerServiceMapping("activemq:queue:inbox", "activemq:queue:inbox");
zipkin.addClientServiceMapping("http:someserver/somepath", "http:someserver/somepath");
```

这不是推荐的方法，但可以在不执行任何服务名称映射的情况下快速启动并运行。但是，当您在多个基础架构中有多个系统时，您应该考虑使用人类可读的服务名称，该服务名称映射到，而不是使用 camel 端点 uris。

384.12. CAMEL-ZIPIN-STARTER

如果您使用 Spring Boot，您可以添加 camel-zipkin-starter 依赖项，并通过使用 @CamelZipkin 注解主类来打开 zipkin。然后您可以在 application.properties 文件中配置 camel-zipkin，您可以在其中为 Zipkin 服务器配置主机名和端口号，以及以上选项表中列出的所有其他选项。

您可以在 camel-example-zipkin 中找到此示例

第 385 章 ZOOKEEPER COMPONENT

从 Camel 版本 2.9 开始提供

ZooKeeper 组件允许与 ZooKeeper 集群交互，并在 Camel 中公开以下功能：

1. 在任何 ZooKeeper 创建模式中创建节点。
2. 获取并设置任意集群节点的数据内容（正在设置的数据必须转换为 byte[]）。
3. 创建并检索附加到特定节点的子节点。
4. 一个分布式 RoutePolicy，它利用 ZooKeeper 协调的领导选举机制来确定是否应处理交换。

Maven 用户需要将以下依赖项添加到这个组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-zookeeper</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

385.1. URI 格式

```
zookeeper://zookeeper-server[:port][/path][?options]
```

URI 的路径指定 ZooKeeper 服务器中的节点（a.k.a。作为端点目标的 znode）：

385.2. 选项

ZooKeeper 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
配置 （高级）	使用共享 ZooKeeperConfiguration		ZooKeeperConfiguration
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

ZooKeeper 端点使用 URI 语法进行配置：

```
zookeeper:serverUrls/path
```

使用以下路径和查询参数：

385.2.1. 路径参数(2 参数)：

Name	描述	默认值	类型
serverUrls	**必需** zookeeper 服务器主机（多个服务器可以使用逗号分隔）		字符串
path	**所需的** ZooKeeper 服务器中的节点（也称为 znode）		字符串

385.2.2. 查询参数(12 参数)：

Name	描述	默认值	类型
awaitExistence (common)	**已弃用** 不使用	true	布尔值
listChildren (common)	是否应该列出节点的子项	false	布尔值
timeout (common)	在超时前，在连接上等待的时间间隔。	5000	int
backoff (consumer)	重试前错误后的 backoff 的时间间隔。	5000	long

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
repeat (consumer)	应该更改 znode 为 'watched' 并重复处理。	false	布尔值
sendEmptyMessageOnDelete (consumer)	删除 znode 后，应该向消费者发送空消息	true	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
create (producer)	如果端点当前不存在，则应该创建该节点。	false	布尔值
createMode (producer)	用于新创建的节点的创建模式	EPHEMERAL	字符串
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

385.3. SPRING BOOT AUTO-CONFIGURATION

组件支持 57 选项，如下所列。

Name	描述	默认值	类型
camel.component.zookeeper.cluster.service.attributes	自定义服务属性。		Map

Name	描述	默认值	类型
camel.component.zookeeper.cluster.service.auth-info-list			list
camel.component.zookeeper.cluster.service.base-path			字符串
camel.component.zookeeper.cluster.service.connection-timeout			Long
camel.component.zookeeper.cluster.service.connection-timeout-unit			TimeUnit
camel.component.zookeeper.cluster.service.curator-framework			CuratorFramework
camel.component.zookeeper.cluster.service.enabled	如果 zookeeper 集群服务应该启用或不启用，则默认为 false。	false	布尔值
camel.component.zookeeper.cluster.service.id	集群服务 ID		字符串
camel.component.zookeeper.cluster.service.max-close-wait			Long
camel.component.zookeeper.cluster.service.max-close-wait-unit			TimeUnit
camel.component.zookeeper.cluster.service.namespace			字符串

Name	描述	默认值	类型
camel.component.zookeeper.cluster.service.nodes			list
camel.component.zookeeper.cluster.service.order	服务查找顺序/优先级.		整数
camel.component.zookeeper.cluster.service.reconnect-base-sleep-time			Long
camel.component.zookeeper.cluster.service.reconnect-base-sleep-time-unit			TimeUnit
camel.component.zookeeper.cluster.service.reconnect-max-retries			整数
camel.component.zookeeper.cluster.service.reconnect-max-sleep-time			Long
camel.component.zookeeper.cluster.service.reconnect-max-sleep-time-unit			TimeUnit
camel.component.zookeeper.cluster.service.retry-policy			RetryPolicy
camel.component.zookeeper.cluster.service.session-timeout			Long

Name	描述	默认值	类型
camel.component.zookeeper.cluster.service.session-timeout-unit			TimeUnit
camel.component.zookeeper.configuration.backoff	重试前错误后的 backoff 的时间间隔。	5000	Long
camel.component.zookeeper.configuration.create	如果端点当前不存在，则应该创建该节点。	false	布尔值
camel.component.zookeeper.configuration.create-mode	用于新创建的节点的创建模式	EPHEMERAL	字符串
camel.component.zookeeper.configuration.list-children	是否应该列出节点的子项	false	布尔值
camel.component.zookeeper.configuration.path	ZooKeeper 服务器中的节点（也称为 znode）		字符串
camel.component.zookeeper.configuration.repeat	应该更改 znode 为 'watched' 并重复处理。	false	布尔值
camel.component.zookeeper.configuration.send-empty-message-on-delete	删除 znode 后，应该向消费者发送空消息	true	布尔值
camel.component.zookeeper.configuration.servers	zookeeper 服务器主机		list
camel.component.zookeeper.configuration.timeout	在超时前，在连接上等待的时间间隔。	5000	整数
camel.component.zookeeper.enabled	启用 zookeeper 组件	true	布尔值

Name	描述	默认值	类型
camel.component.zookeeper.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.zookeeper.service-registry.attributes	自定义服务属性。		Map
camel.component.zookeeper.service-registry.auth-info-list			list
camel.component.zookeeper.service-registry.base-path			字符串
camel.component.zookeeper.service-registry.connection-timeout			Long
camel.component.zookeeper.service-registry.connection-timeout-unit			TimeUnit
camel.component.zookeeper.service-registry.curator-framework			CuratorFramework
camel.component.zookeeper.service-registry.deregister-services-on-stop			布尔值

Name	描述	默认值	类型
camel.component.zookeeper.service-registry.enabled	如果 zookeeper 服务 registry 应该启用或不启用，则默认为 false。	false	布尔值
camel.component.zookeeper.service-registry.id	Service Registry ID		字符串
camel.component.zookeeper.service-registry.max-close-wait			Long
camel.component.zookeeper.service-registry.max-close-wait-unit			TimeUnit
camel.component.zookeeper.service-registry.namespace			字符串
camel.component.zookeeper.service-registry.nodes			list
camel.component.zookeeper.service-registry.order	服务查找顺序/优先级.		整数
camel.component.zookeeper.service-registry.override-service-host			布尔值
camel.component.zookeeper.service-registry.reconnect-base-sleep-time			Long

Name	描述	默认值	类型
camel.component.zookeeper.service-registry.reconnect-base-sleep-time-unit			TimeUnit
camel.component.zookeeper.service-registry.reconnect-max-retries			整数
camel.component.zookeeper.service-registry.reconnect-max-sleep-time			Long
camel.component.zookeeper.service-registry.reconnect-max-sleep-time-unit			TimeUnit
camel.component.zookeeper.service-registry.retry-policy			RetryPolicy
camel.component.zookeeper.service-registry.service-host			字符串
camel.component.zookeeper.service-registry.session-timeout			Long
camel.component.zookeeper.service-registry.session-timeout-unit			TimeUnit

Name	描述	默认值	类型
camel.component.zookeeper.configuration.await-existence	未使用	true	布尔值

385.4. 使用案例

385.4.1. 从 znode读取

以下片段将从 znode/somepath/somenode/ 中读取它已存在的数据。检索的数据将放入交换中，并传递给路由的其余部分：

```
from("zookeeper://localhost:39913/somepath/somenode").to("mock:result");
```

如果节点尚不存在，则可以提供标志来让端点等待其创建：

```
from("zookeeper://localhost:39913/somepath/somenode?
awaitCreation=true").to("mock:result");
```

385.4.2. 从 znode 读取（额外 Camel 2.10 开始）

当因为从 ZooKeeper ensemble 收到的 WatchedEvent 中读取数据时，CamelZookeeperEventType 标头包含来自该 WatchedEvent 的 EventType 值。如果初始读取数据（不是由 WatchedEvent触发），则不会设置 CamelZookeeperEventType 标头。

385.4.3. 写入一个 znode

以下片段会将交换的有效负载写入位于 /somepath/somenode/ 的 znode 中，前提是它已存在：

```
from("direct:write-to-znode")
   .to("zookeeper://localhost:39913/somepath/somenode");
```

为获得灵活性，端点允许将目标 znode 动态指定为消息标头。如果存在由字符串 CamelZooKeeperNode 键的标头，则标头的值将用作服务器上 znode 的路径。例如，使用上述同一路由定义，以下代码片段会将数据写入 /somepath/somenode，而是写入标题 /somepath/someothernode 的路径。

警告

testPayload 必须转换为 byte[]，因为 ZooKeeper 中存储的数据是基于字节的。

```
Object testPayload = ...
template.sendBodyAndHeader("direct:write-to-znode", testPayload, "CamelZooKeeperNode",
"/somepath/someothernode");
```

如果不存在 create 选项，则也会创建该节点。

```
from("direct:create-and-write-to-znode")
   .to("zookeeper://localhost:39913/somepath/somenode?create=true");
```

启动 2.11 版本 也可以使用标头 CamelZookeeperOperation 删除节点，方法是将其设置为 DELETE：

```
from("direct:delete-znode")
   .setHeader(ZooKeeperMessage.ZOOKEEPER_OPERATION, constant("DELETE"))
   .to("zookeeper://localhost:39913/somepath/somenode");
```

或者等效于：

```
<route>
  <from uri="direct:delete-znode" />
  <setHeader headerName="CamelZookeeperOperation">
    <constant>DELETE</constant>
  </setHeader>
  <to uri="zookeeper://localhost:39913/somepath/somenode" />
</route>
```

ZooKeeper 节点可以有不同的类型，可以是 'Ephemeral' 或 'Persistent' 和 'Sequenced' 或 'Unsequenced'。有关每种类型的更多信息，您可以检查 此处。默认情况下，端点将创建未排序的临时节点，但类型可以通过 uri 配置参数或特殊消息标头轻松操作。create 模式预期的值只是来自 CreateMode 枚举的名称：

- 持久性

- 持久性_SEQUENTIAL
- EPHEMERAL
- EPHEMERAL_SEQUENTIAL

例如，通过 URI 配置创建持久性 znode ：

```
from("direct:create-and-write-to-persistent-znode")
    .to("zookeeper://localhost:39913/somepath/somenode?
create=true&createMode=PERSISTENT");
```

或使用标头 CamelZookeeperCreateMode。

警告

testPayload 必须转换为 byte[]，因为 ZooKeeper 中存储的数据是基于字节的。

```
Object testPayload = ...
template.sendBodyAndHeader("direct:create-and-write-to-persistent-znode", testPayload,
"CamelZooKeeperCreateMode", "PERSISTENT");
```

385.5. ZOOKEEPER 启用的路由策略

ZooKeeper 允许非常简单且有效的领导选举功能。此组件在 RoutePolicy 中利用这个选择功能来控制路由的时间和方式。此策略通常在故障切换场景中使用，以控制基于 Camel 的服务器集群中路由相同的实例。一个非常常见的场景是一个简单的"Master-Slave"设置，其中有多个路由在集群中分发，但只有一个实例(master)之一应该一次运行。如果主设备失败，则应从可用的 slaves 选择一个新的 master，并且应启动此新 master 中的路由。

该策略在将涉及选举的 RoutePolicy 的所有实例中使用通用 znode 路径。每个策略将其 id 写入这个节点，Zookeeper 将根据它收到的顺序对写操作进行排序。然后，策略读取节点列表以查看其 id 的位置；此位置用于确定是否应启动路由。该策略在启动时配置有路由实例数，应在集群中启动，如果其在列

表中的位置小于这个值，则其路由将启动。对于主从方案，路由配置有 1 个路由实例，只有列表中的第一个条目才会启动其路由。所有策略都会监视对列表的更新，如果应该启动其路由，则会重新计算它们。有关 Zookeeper 的领导选举功能的更多信息，请参阅 此页面。

以下示例将节点 /someapplication/somepolicy 用于选举机制，并设置为只启动节点列表中的 top '1' 条目，如选择 master：

```
ZooKeeperRoutePolicy policy = new
ZooKeeperRoutePolicy("zookeeper:localhost:39913/someapp/somepolicy", 1);
from("direct:policy-controlled")
   .routePolicy(policy)
   .to("mock:controlled");
```

目前，组件中定义了 3 个策略，不同的 SLA：

- ZooKeeperRoutePolicy
- CuratorLeaderRoutePolicy （自 2.19开始）
- MultiMasterCuratorLeaderRoutePolicy (since 2.19)

ZooKeeperRoutePolicy 支持多个活跃节点，但只有在 Camel 组件及其对应 Consumer 启动后才会启动激活，这会根据您的路由定义，您组件可能已经开始使用事件并生成 'Exchange's 的风险，然后策略可能无法激活该节点。

CuratorLeaderRoutePolicy 只支持一个活跃的节点，但它绑定到不同的 CamelContext 生命周期方法；此策略在启动任何路由或消费者之前启动，因此您可以确保在策略决定之前没有处理任何节点。

MultiMasterCuratorLeaderRoutePolicy 支持多个活跃节点，并绑定到与 CuratorLeaderRoutePolicy 相同的生命周期方法；此策略在启动任何路由或消费者之前启动，因此您可以确保在策略决定之前没有处理。

385.6. 另请参阅

- 配置 Camel

- *组件*
- *端点*
- *开始使用*

第 386 章 ZOOKEEPER MASTER COMPONENT

从 Camel 版本 2.19 开始提供

zookeeper-master: 端点提供了一种方式，以确保集群中的单个消费者只消耗给定端点；如果 JVM 死机，则自动故障转移。

如果您需要从一些不支持并发使用的后端使用，或者因为商业或稳定性原因而需要消耗一些旧的后端，则这非常有用。

386.1. 使用 MASTER 端点

只需为任何 camel 端点加上前缀 zookeeper-master:someName: where someName 是一个逻辑名称，用于获取 master 锁定。例如：

```
from("zookeeper-master:cheese:jms:foo").to("activemq:wine");
```

以上在 ActiveMQ 中模拟 [Exclusive Consumers](http://activemq.apache.org/exclusive-consumer.html)类型功能；但在任何可能不支持专用用户的第三方 JMS 提供程序上。

386.2. URI 格式

```
zookeeper-master:name:endpoint[?options]
```

其中 endpoint 是您要以 master/slave 模式运行的任何 Camel 端点。

386.3. 选项

ZooKeeper Master 组件支持 7 个选项，如下所列。

Name	描述	默认值	类型
containerIdFactory (consumer)	使用自定义 ContainerIdFactory 创建容器 ID。		ContainerIdFactory

Name	描述	默认值	类型
zkRoot (consumer)	在 zookeeper 中使用的根路径，其中信息存储哪些节点是主/从设备等。默认使用：/camel/zookeepermaster/clusters/master	/camel/zookeepermaster/clusters/master	字符串
Curator (advanced)	使用自定义配置的 CuratorFramework 作为到 zookeeper ensemble 的连接。		CuratorFramework
maximumConnection Timeout (consumer)	连接到 zookeeper ensemble 时使用的超时(millis)	10000	int
zooKeeperUrl (consumer)	zookeeper ensemble 的 url	localhost:2181	字符串
zooKeeperPassword (security)	连接到 zookeeper ensemble 时使用的密码		字符串
resolveProperty Placeholders (advanced)	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值

ZooKeeper Master 端点使用 URI 语法进行配置：

```
zookeeper-master:groupName:consumerEndpointUri
```

使用以下路径和查询参数：

386.3.1. 路径参数(2 参数)：

Name	描述	默认值	类型
groupName	**必需的** 集群组的名称		字符串
consumerEndpointUri	**必需的** 在 master/slave 模式中使用的消费者端点		字符串

386.3.2. 查询参数(4 参数)：

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
ExceptionHandler (consumer)	要让使用者使用自定义例外处理程序：请注意，如果启用了 bridgeErrorHandler 选项，则此选项不使用。默认情况下，消费者将处理异常，其记录在 WARN 或 ERROR 级别中，并忽略。		ExceptionHandler
exchangePattern (consumer)	在消费者创建交换时设置交换模式。		ExchangePattern
同步 （高级）	设置是否应严格使用同步处理，还是允许 Camel 使用异步处理（如果支持）。	false	布尔值

386.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 8 个选项，如下所列。

Name	描述	默认值	类型
camel.component.zookeeper-master.container-id-factory	使用自定义 ContainerIdFactory 创建容器 ID。选项是一个 org.apache.camel.component.zookeepermaster.ContainerIdFactory 类型。		字符串
camel.component.zookeeper-master.curator	使用自定义配置的 CuratorFramework 作为到 zookeeper ensemble 的连接。选项是 org.apache.curator.framework.CuratorFramework 类型。		字符串
camel.component.zookeeper-master.enabled	启用 zookeeper-master 组件	true	布尔值
camel.component.zookeeper-master.maximum-connection-timeout	连接到 zookeeper ensemble 时使用的超时(millis)	10000	整数

Name	描述	默认值	类型
camel.component.zookeeper-master.resolve-property-placeholders	组件是否应在启动时解析属性占位符。只有 String 类型的属性可以使用属性占位符。	true	布尔值
camel.component.zookeeper-master.zk-root	在 zookeeper 中使用的根路径，其中信息存储哪些节点是主/从设备等。默认使用：/camel/zookeepermaster/clusters/master	/camel/zookeepermaster/clusters/master	字符串
camel.component.zookeeper-master.zoo-keeper-password	连接到 zookeeper ensemble 时使用的密码		字符串
camel.component.zookeeper-master.zoo-keeper-url	zookeeper ensemble 的 url	localhost:2181	字符串

386.5. EXAMPLE

您可以保护集群的 Camel 应用程序，使其仅消耗来自一个活跃节点的文件。

```
// the file endpoint we want to consume from
String url = "file:target/inbox?delete=true";

// use the zookeeper master component in the clustered group named myGroup
// to run a master/slave mode in the following Camel url
from("zookeeper-master:myGroup:" + url)
    .log(name + " - Received file: ${file:name}")
    .delay(delay)
    .log(name + " - Done file:     ${file:name}")
    .to("file:target/outbox");
```

ZooKeeper 默认连接到 localhost:2181，但您可以在组件级别上配置它。

```
MasterComponent master = new MasterComponent();
master.setZooKeeperUrl("myzookeeper:2181");
```

但是，您也可以使用环境变量配置 ZooKeeper ensemble 的 url。

```
export ZOOKEEPER_URL = "myzookeeper:2181"
```

第 387 章 MASTER ROUTEPOLICY

您还可以使用 RoutePolicy 在 master/slave 模式中控制路由。

当这样做时，您必须使用以下方法配置路由策略

- zookeeper ensemble 的 URL
- 集群组的名称
- 重要 并将路由设置为不自动启动

一些示例

```
MasterRoutePolicy master = new MasterRoutePolicy();
master.setZooKeeperUrl("localhost:2181");
master.setGroupName("myGroup");

// its import to set the route to not auto startup
// as we let the route policy start/stop the routes when it becomes a master/slave etc
from("file:target/inbox?delete=true").noAutoStartup()
    // use the zookeeper master route policy in the clustered group
    // to run this route in master/slave mode
    .routePolicy(master)
    .log(name + " - Received file: ${file:name}")
    .delay(delay)
    .log(name + " - Done file:     ${file:name}")
    .to("file:target/outbox");
```

387.1. 另请参阅

- 配置 Camel
- 组件
- 端点

- *开始使用*