Red Hat build of Apache Camel 4.4

Red Hat build of Apache Camel for Spring Boot 参考

Red Hat build of Apache Camel for Spring Boot 参考

Last Updated: 2024-07-03

Red Hat build of Apache Camel 4.4 Red Hat build of Apache Camel for Spring Boot 参考

Red Hat build of Apache Camel for Spring Boot 参考

法律通告

摘要

本指南描述了红帽构建的 Apache Camel 组件的设置。

目录

前言

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。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

第 1 章 组件启动者

Camel Spring Boot 支持以下 Camel 工件作为 Spring Boot Starters：

- 表 1.1 “Camel 组件”
- 表 1.2 “Camel 数据格式”
- 表 1.3 “Camel 语言”
- 表 1.4 “其它扩展”

表 1.1. Camel 组件

组件	工件	描述	支持 IBM Power 和 IBM Z
AMQP	camel-amqp-starter	使用 Apache QPid 客户端与 AMQP 协议进行消息传递.	是
AWS Cloudwatch	camel-aws2-cw-starter	使用 AWS SDK 版本 2.x 将指标发送到 AWS CloudWatch。	是
AWS DynamoDB	camel-aws2-ddb-starter	使用 AWS SDK 版本 2.x 从 AWS DynamoDB 服务存储和检索数据。	是
AWS Kinesis	camel-aws2-kinesis-starter	使用 AWS SDK 版本 2.x 使用 AWS SDK 版本 2.x 使用和生成 AWS Kinesis Streams 的记录。	是
AWS Lambda	camel-aws2-lambda-starter	使用 AWS SDK 版本 2.x 管理并调用 AWS Lambda 功能。	是
AWS S3 Storage Service	camel-aws2-s3-starter	使用 AWS SDK 版本 2.x 从 AWS S3 Storage Service 存储和检索对象。	是
AWS Simple Notification System (SNS)	camel-aws2-sns-starter	使用 AWS SDK 版本 2.x 将信息发送到 AWS Simple Notification Topic。	是
AWS Simple Queue Service (SQS)	camel-aws2-sqs-starter	使用 AWS SDK 版本 2.x 向 AWS SQS 服务发送和接收信息。	是

组件	工件	描述	支持 IBM Power 和 IBM Z
Azure ServiceBus	camel-azure-servicebus-starter	向 Azure 事件总线发送和接收信息。	是
Azure Storage Blob Service	camel-azure-storage-blob-starter	使用 SDK v12 从 Azure Storage Blob Service 存储和检索 Blob。	是
Azure Storage Queue Service	camel-azure-storage-queue-starter	azure-storage-queue 组件用于使用 Azure SDK v12 存储和检索信息到 Azure Storage Queue。	是
Bean	camel-bean-starter	调用存储在 Camel registry 中的 Java Bean 的方法。	是
Bean Validator	camel-bean-validator-starter	使用 Java Bean Validation API 验证消息正文。	是
浏览	camel-browse-starter	检查在支持 BrowsableEndpoint 的端点上收到的消息。	是
Cassandra CQL	camel-cassandraql-starter	使用 CQL3 API （而不是 Thrift API）与 Cassandra 2.0 集成。基于 DataStax 提供的 Cassandra Java 驱动程序。	是
CICS	camel-cics-starter	与 CICS® 通用事务处理子系统交互。	否
控制总线	camel-controlbus-starter	管理和监控 Camel 路由。	是
cron	camel-cron-starter	通过 Unix cron 语法指定的时间触发事件的通用接口。	是
加密(JCE)	camel-crypto-starter	使用 Java Cryptographic 扩展(JCE)的签名服务签名并验证交换。	是

组件	工件	描述	支持 IBM Power 和 IBM Z
CXF	camel-cxf-soap-starter	使用 Apache CXF 公开 SOAP WebServices 或使用 CXF WS 客户端连接到外部 WebServices。	是
CXF-RS	camel-cxf-rest-starter	使用 Apache CXF 公开 JAX-RS REST 服务，或使用 CXF REST 客户端连接到外部 REST 服务。	是
数据格式	camel-dataformat-starter	使用 Camel 数据格式作为常规 Camel 组件。	是
Dataset	camel-dataset-starter	提供用于 Camel 应用程序的负载和 soak 测试的数据。	是
direct	camel-direct-starter	同步调用来自同一 Camel 上下文的另一个端点。	是
Elastic Search	camel-elasticsearch-starter	通过 Java 客户端 API 将请求发送到 ElasticSearch。	否
FHIR	camel-fhir-starter	使用 FHIR (Fast Healthcare Interoperability Resources)标准交换医疗域中的信息。	否
File	camel-file-starter	读写文件。	是
Flink	camel-flink-starter	将 DataSet 作业发送到 Apache Flink 集群。	是
FTP	camel-ftp-starter	上传文件并将其下载到 FTP 服务器/从 FTP 服务器下载.	是
Google BigQuery	camel-google-bigquery-starter	用于分析的 Google BigQuery 数据仓库。	是

组件	工件	描述	支持 IBM Power 和 IBM Z
Google Pubsub	camel-google-pubsub-starter	向 Google Cloud Platform PubSub Service 发送和接收信息。	是
gRPC	camel-grpc-starter	公开 gRPC 端点并访问外部 gRPC 端点。	是
HTTP	camel-http-starter	使用 Apache HTTP 客户端 4.x 将请求发送到外部 HTTP 服务器。	是
Infinispan	camel-infinispan-starter	从/到 Infinispan 分布式键/值存储和数据网格进行读写。	否
Infinispan Embedded	camel-infinispan-embedded-starter	从/到 Infinispan 分布式键/值存储和数据网格进行读写。	是
JDBC	camel-jdbc-starter	通过 SQL 和 JDBC 访问数据库.	是
Jira	camel-jira-starter	与 JIRA 问题跟踪器交互。	是
JMS	camel-jms-starter	从 JMS Queue 或 Topic 中发送和接收消息。	是
JPA	camel-jpa-starter	使用 Java Persistence API (zFCP)从数据库存储和检索 Java 对象。	是
JSLT	camel-jslt-starter	使用 JSLT 查询或转换 JSON 有效负载。	是
Kafka	camel-kafka-starter	向 Apache Kafka 代理发送和接收信息。	是
kamelet	camel-kamelet-starter	调用 Kamelets	是
Kubernetes ConfigMap	camel-kubernetes-starter	对 Kubernetes ConfigMap 执行操作，并获取有关 ConfigMap 更改的通知。	是

组件	工件	描述	支持 IBM Power 和 IBM Z
Kubernetes 自定义资源	camel-kubernetes-starter	对 Kubernetes 自定义资源执行操作，并获取有关部署更改的通知。	是
Kubernetes Deployments	camel-kubernetes-starter	对 Kubernetes Deployments 执行操作，并获取有关部署更改的通知。	是
Kubernetes 事件	camel-kubernetes-starter	对 Kubernetes 事件执行操作，并获取有关事件更改的通知。	是
Kubernetes HPA	camel-kubernetes-starter	对 Kubernetes Horizontal Pod Autoscaler (HPA)执行操作，并获取有关 HPA 更改的通知。	是
Kubernetes 任务	camel-kubernetes-starter	对 Kubernetes 作业执行操作。	是
Kubernetes 命名空间	camel-kubernetes-starter	对 Kubernetes 命名空间执行操作，并获得对命名空间更改的通知。	是
Kubernetes 节点	camel-kubernetes-starter	在 Kubernetes 节点上执行操作，并获得有关节点更改的通知。	是
Kubernetes 持久性卷	camel-kubernetes-starter	对 Kubernetes 持久性卷执行操作，并获得有关持久性卷更改的通知。	是
Kubernetes 持久性卷声明	camel-kubernetes-starter	对 Kubernetes 持久性卷声明执行操作，并获得有关持久性卷声明更改的通知。	是
Kubernetes Pod	camel-kubernetes-starter	对 Kubernetes Pod 执行操作并获取有关 Pod 更改的通知。	是
Kubernetes 复制控制器	camel-kubernetes-starter	是对 Kubernetes Replication Controller 的执行操作，并获得关于复制控制器更改的通知。	是

组件	工件	描述	支持 IBM Power 和 IBM Z
Kubernetes 资源配额	camel-kubernetes-starter	对 Kubernetes 资源配额执行操作。	是
Kubernetes Secret	camel-kubernetes-starter	对 Kubernetes Secret 执行操作。	是
Kubernetes 服务帐户	camel-kubernetes-starter	对 Kubernetes 服务帐户执行操作。	是
Kubernetes 服务	camel-kubernetes-starter	对 Kubernetes 服务执行操作并获取有关服务更改的通知。	是
Kudu	camel-kudu-starter	与 Apache Kudu 互动，Apache Hadoop 生态系统的免费、开源列导向型数据存储。	否
语言	camel-language-starter	使用 Camel 支持的任何语言执行脚本。	是
LDAP	camel-ldap-starter	在 LDAP 服务器上执行搜索。	是
Log	camel-log-starter	日志消息到底层日志记录机制。	是
LRA	camel-lra-starter	Camel saga 绑定 for Long-Running-Action 框架。	是
Mail	camel-mail-starter	使用 imap、pop3 和 smtp 协议发送和接收电子邮件。	是
邮件 Microsoft OAuth	camel-mail-microsoft-oauth-starter	Camel Mail OAuth2 Authenticator for Microsoft Exchange Online.	是
MapStruct	camel-mapstruct-starter	使用 Mapstruct 键入 Conversion。	是
Master	camel-master-starter	集群中仅消耗来自给定端点的单一使用者；如果 JVM 中断，则进行自动故障转移。	是

组件	工件	描述	支持 IBM Power 和 IBM Z
Micrometer	camel-micrometer-starter	使用 Micrometer 库直接从 Camel 路由收集各种指标。	是
Minio	camel-minio-starter	使用 Minio SDK 从 Minio Storage Service 存储和检索对象。	是
MLLP	camel-mllp-starter	使用 MLLP 协议与外部系统通信。	是
Mock	camel-mock-starter	使用模拟测试路由和调解规则。	是
MongoDB	camel-mongodb-starter	对 MongoDB 文档和集合执行操作。	是
MyBatis	camel-mybatis-starter	使用 MyBatis 在相关数据库中执行查询、轮询、插入、更新或删除。	是
Netty	camel-netty-starter	使用带有 Netty 4.x 的 TCP 或 UDP 的套接字级别网络。	是
Olingo4	camel-olingo4-starter	使用 Apache Olingo OData API 与 OData 4.0 服务通信。	是
OpenShift 构建配置	camel-kubernetes-starter	对 OpenShift 构建配置执行操作。	是
OpenShift 构建	camel-kubernetes-starter	对 OpenShift 构建执行操作。	是
OpenShift 部署配置	camel-kubernetes-starter	对 Openshift Deployment Configs 执行操作，并获得关于部署配置更改的通知。	是
Netty HTTP	camel-netty-http-starter	使用 Netty 4.x 的 Netty HTTP 服务器和客户端。	是
paho	camel-paho-starter	使用 Eclipse Paho MQTT 客户端与 MQTT 消息代理进行通信。	是

组件	工件	描述	支持 IBM Power 和 IBM Z
Paho MQTT 5	camel-paho-mqtt5-starter	使用 Eclipse Paho MQTT v5 客户端与 MQTT 消息代理进行通信。	是
平台 HTTP	camel-platform-http-starter	使用当前平台中的 HTTP 服务器公开 HTTP 端点。	是
quartz	camel-quartz-starter	调度使用 Quartz 2.x 调度程序发送消息。	是
Ref	camel-ref-starter	将消息路由到端点会根据 Camel Registry 中的名称动态查找。	是
REST	camel-rest-starter	公开 REST 服务或调用外部 REST 服务。	是
saga	camel-saga-starter	使用 Saga EIP 在路由中执行自定义操作。	是
Salesforce	camel-salesforce-starter	使用 Java DTO 与 Salesforce.	是
SAP	camel-sap-starter	使用 SAP Java Connector (SAP JCo)库来促进与 SAP 和 SAP IDoc 库的双向通信，以促进 Intermediate Document (IDoc)格式的文档传输。	是
scheduler	camel-scheduler-starter	使用 java.util.concurrent.ScheduledExecutorService 以指定间隔生成消息。	是
SEDA	camel-seda-starter	异步调用同一 JVM 中任何 Camel 上下文的另一个端点。	是
Servlet	camel-servlet-starter	由 Servlet 提供 HTTP 请求。	是
Slack	camel-slack-starter	向 Slack 发送和接收信息。	是

组件	工件	描述	支持 IBM Power 和 IBM Z
SMB	camel-smb-starter	从 SMB （服务器消息块）共享接收文件。	是
SNMP	camel-snmp-starter	接收陷阱和轮询 SNMP （简单网络管理协议）功能的设备。	是
Splunk	camel-splunk-starter	在 Splunk 中发布或搜索事件。	否
Spring Batch	camel-spring-batch-starter	将消息发送到 Spring Batch 以进一步处理。	是
Spring JDBC	camel-spring-jdbc-starter	使用 Spring Transaction 支持通过 SQL 和 JDBC 访问数据库。	是
Spring LDAP	camel-spring-ldap-starter	将过滤器用作消息有效负载，在 LDAP 服务器中执行搜索。	是
Spring RabbitMQ	camel-spring-rabbitmq-starter	使用 Spring RabbitMQ 客户端从 RabbitMQ 发送和接收消息。	是
Spring Redis	camel-spring-redis-starter	从 Redis 发送和接收信息。	是
Spring Webservice	camel-spring-ws-starter	您可以使用此组件与 Spring Web Services 集成。它提供访问 Web 服务和服务器端支持，以便创建您的合同优先 Web 服务。	是
SQL	camel-sql-starter	使用 Spring JDBC 执行 SQL 查询。	是
SQL 存储流程	camel-sql-starter	使用 Spring JDBC 执行 SQL 查询作为 JDBC 存储的流程。	是
SSH	camel-ssh-starter	使用 SSH 在远程主机上执行命令。	是

组件	工件	描述	支持 IBM Power 和 IBM Z
Stub	camel-stub-starter	在开发或测试过程中处理任何物理端点。	是
telegram	camel-telegram-starter	发送和接收充当 Telegram Botram Bot API 的消息。	是
计时器	camel-timer-starter	使用 java.util.Timer 以指定间隔生成消息。	是
validator	camel-validator-starter	使用 XML 架构和 JAXP 验证验证载荷。	是
Velocity	camel-velocity-starter	使用 Velocity 模板转换消息。	是
Vert.x HTTP Client	camel-vertx-http-starter	使用 Vert.x 将请求发送到外部 HTTP 服务器。	是
Vert.x WebSocket	camel-vertx-websocket-starter	公开 WebSocket 端点，并使用 Vert.x 连接到远程 WebSocket 服务器。	是
Webhook	camel-webhook-starter	公开 Webhook 端点以接收其他 Camel 组件的推送通知。	是
XJ	camel-xj-starter	使用 XSLT 转换 JSON 和 XML 消息。	是
XSLT	camel-xslt-starter	使用 XSLT 模板转换 XML 有效负载。	是
XSLT Saxon	camel-xslt-saxon-starter	使用 Saxon 的 XSLT 模板转换 XML 有效负载。	是

表 1.2. Camel 数据格式

组件	工件	描述	支持 IBM Power 和 IBM Z
Avro	camel-avro-starter	使用 Apache Avro 二进制数据格式序列化和反序列化消息。	是

组件	工件	描述	支持 IBM Power 和 IBM Z
avro Jackson	camel-jackson-avro-starter	marshal POJO 到 Avro，并使用 Jackson 返回。	是
bindy	camel-bindy-starter	使用 Camel Bindy 的 POJO 和键值对(KVP)格式之间的 marshal 和 unmarshal。	是
HL7	camel-hl7-starter	使用 HL7 MLLP codec 的 marshal 和 unmarshal HL7 (Health care)模型对象。	是
JacksonXML	camel-jacksonxml-starter	unmarshal 是一个 XML 有效负载到 POJO，并使用 Jackson 的 XMLMapper 扩展来回放。	是
JAXB	camel-jaxb-starter	unmarshal XML 有效负载到 POJO，使用 JAXB2 XML marshalling 标准。	是
JSON Gson	camel-gson-starter	使用 Gson marshal POJO 到 JSON 并返回	是
JSON Jackson	camel-jackson-starter	marshal POJO 到 JSON 并使用 Jackson 返回	是
protobuf Jackson	camel-jackson-protobuf-starter	使用 Jackson 对 Protobuf 和 back 进行 marshal POJO。	是
SOAP	camel-soap-starter	marshal Java 对象到 SOAP 消息和回放。	是
zip 文件	camel-zipfile-starter	使用 java.util.zip.ZipStream 压缩和解压缩流。	是

表 1.3. Camel 语言

语言	工件	描述	支持 IBM Power 和 IBM Z
常数	camel-core-starter	固定值只在路由启动期间设置一次。	是
CSimple	camel-core-starter	评估编译的简单表达式。	是
ExchangeProperty	camel-core-starter	从 Exchange 获取属性。	是
File	camel-core-starter	简单语言的文件相关功能。	是
标头	camel-core-starter	从 Exchange 获取标头。	是
JQ	camel-jq-starter	针对 JSON 消息正文评估 JQ 表达式。	是
jsonpath	camel-jsonpath-starter	针对 JSON 消息正文评估 JSONPath 表达式。	是
Ref	camel-core-starter	使用 registry 中的现有表达式。	是
Simple（简单）	camel-core-starter	评估 Camel 简单表达式。	是
tokenize	camel-core-starter	使用分隔符模式对文本有效负载进行令牌化。	是
XML 令牌化	camel-xml-jaxp-starter	对 XML 有效负载进行令牌化。	是
XPath	camel-xpath-starter	针对 XML 有效负载评估 XPath 表达式。	是
XQuery	camel-saxon-starter	使用 XQuery 和 Saxon 查询和/或转换 XML 有效负载。	是

表 1.4. 其它扩展

扩展	工件	描述	支持 IBM Power 和 IBM Z
Jasypt	camel-jasypt-starter	使用 Jasypt 的安全性	是

扩展	工件	描述	支持 IBM Power 和 IBM Z
kamelet Main	camel-kamelet-main-starter	运行 Kamelet standalone	是
OpenAPI Java	camel-openapi-java-starter	使用 openapi doc 的 rest-dsl 支持	是
OpenTelemetry	camel-opentelemetry-starter	使用 OpenTelemetry 的分布式追踪	是
Spring Security	camel-spring-security-starter	使用 Spring Security 的安全性	是
YAML DSL	camel-yaml-dsl-starter	使用 YAML 的 Camel DSL	是

第 2 章 AMQP

自 Camel 1.2 开始

支持生成者和消费者

AMQP 组件使用 Qpid 项目的 JMS 客户端 API 支持 AMQP 1.0 协议。

2.1. 依赖项

当在 Camel Spring Boot 中使用 **camel-amqp** 时，请将以下 Maven 依赖项添加到 **pom.xml** 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-amqp-starter</artifactId>
</dependency>
```

2.2. URI 格式

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

2.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

2.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

2.3.2. 端点级别选项

在 **Endpoint 级别**，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 **路径和 查询参数**。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

2.4. 组件选项

AMQP 组件支持 100 个选项，如下所列。

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，且只能被单个 JMS 连接实例使用。**clientId** 选项与 JMS 1.1 durable 主题订阅相一致，因为客户端 ID 用于控制必须存储哪些客户端消息。使用 JMS 2.0 客户端时，clientId 可能会省略，这会创建一个 'global' 订阅。		字符串
ConnectionFactory (common)	要使用的连接工厂。连接工厂必须在组件或端点上配置。		ConnectionFactory
disableReplyTo (common)	指定 Camel 是否忽略消息中的 JMSReplyTo 标头。如果为 true，Camel 不会向 JMSReplyTo 标头中指定的目的地发送回复。如果您希望 Camel 消耗路由，且您不想 Camel 自动发送回复消息，您可以使用这个选项，因为代码中的另一个组件处理回复消息。如果要使用 Camel 作为不同消息代理之间的代理，而您想要将消息从一个系统路由到另一个系统，也可以使用此选项。	false	布尔值
durableSubscriptionName (common)	用于指定持久主题订阅的持久订阅者名称。必须为 JMS 1.1 持久订阅配置 **clientId** 选项，并且可以针对 JMS 2.0 配置，以创建私有持久订阅。		字符串
includeAmqpAnnotations (common)	在从 AMQP 到 Camel 消息映射时是否包含 AMQP 注解。把它设置为 true 将包含 JMS_AMQP_MA_ 前缀的 AMQP 消息注解映射到消息标头。由于 Apache Qpid JMS API 的限制，当前交付注释将被忽略。	false	布尔值
jmsMessageType (common)	允许您强制使用特定的 javax.jms.Message 实现来发送 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 body 类型中使用的 JMS 消息类型。这个选项允许您指定它。 Enum 值： • Bytes • Map • 对象 • Stream • 文本		JmsMessageType
replyTo (common)	提供显式 ReplyTo 目的地（会覆盖消费者中的 Message.getJMSReplyTo （）的任何传入值）。		字符串

Name	描述	默认值	类型
testConnectionOnStartup (common)	指定是否在启动时测试连接。这样可确保 Camel 启动所有 JMS 用户具有与 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 没有使用失败的连接启动。JMS 制作者也经过测试。	false	布尔值
confirmmentModeName (consumer)	JMS 确认名称，即： SESSION_TRANSACTED、CLIENT_ACKNOWLEDGE、AUTO_ACKNOWLEDGE、DUPS_OK_ACKNOWLEDGE。 Enum 值： • SESSION_TRANSACTED • CLIENT_ACKNOWLEDGE • AUTO_ACKNOWLEDGE • DUPS_OK_ACKNOWLEDGE	AUTO_ACKNOWLEDGE	字符串
artemisConsumerPriority (consumer)	通过消费者优先级，您可以确保高优先级消费者在激活时收到消息。通常，连接到队列的活动用户以轮循方式从它接收消息。使用消费者优先级时，如果有多个活跃的消费者具有相同的高优先级，则会发送循环消息。只有高优先级消费者没有使用消息的信用时，消息才会受到较低优先级的用户，或者那些高优先级消费者已拒绝接受该消息（例如，因为它不符合与消费者关联的任何选择器的条件）。		int
asyncConsumer (consumer)	JmsConsumer 是否异步处理 Exchange。如果启用，则 JmsConsumer 可能会从 JMS 队列中获取下一个消息，而前面的消息会被异步处理（通过异步路由引擎）。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 从 JMS 队列获取下一个消息前完全处理 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该自动启动。	true	布尔值
cacheLevel (consumer)	根据 ID 为底层 JMS 资源设置缓存级别。如需了解更多详细信息，请参阅 cacheLevelName 选项。		int

Name	描述	默认值	类型
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。 Enum 值： • CACHE_AUTO • CACHE_CONNECTION • CACHE_CONSUMER • CACHE_NONE • CACHE_SESSION	CACHE_AUTO	字符串
concurrentConsumers (consumer)	指定从 JMS 消耗时的默认并发用户数（不适用于 JMS 上的请求/回复）。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。当对 JMS 进行请求/回复时，选项 replyToConcurrentConsumers 用于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（不适用于 JMS 上的请求/回复）。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。当对 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消费者数量。		int
replyToDeliveryPersistent (consumer)	指定是否默认使用持久性发送进行回复。	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化。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。在带有共享或持久订阅的主题(pub-sub domain)时应用。如果配置了客户端 ID，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：每个订阅只允许 1 个并发消费者（这是此消息监听程序容器的默认值），除了一个共享订阅（需要 JMS 2.0）。		字符串

Name	描述	默认值	类型
subscriptionShared (consumer)	设置是否共享订阅。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 来注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以作为订阅名称使用）。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)时有意义，因此此方法也会切换 pubSubDomain 标志。需要 JMS 2.0 兼容消息代理。	false	布尔值
acceptMessagesWhileStopping (consumer (advanced))	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，您可以考虑启用此选项，同时仍然在队列上排队消息。如果此选项为 false，并且您停止了 JMS 路由，则消息可能会被拒绝，而 JMS 代理必须尝试重新设计，这一次可能被拒绝，最终该消息可能会在 JMS 代理上的死信队列中移动。要避免这种情况，建议启用这个选项。	false	布尔值
allowReplyManagerQuickStop (consumer (advanced))	是否在回复管理器中使用 DefaultMessageListenerContainer 用于 request-reply 消息，都允许 DefaultMessageListenerContainer.runningAllowed 标志在启用了 JmsConfiguration#isAcceptMessagesWhileStopping 时快速停止，并且 org.apache.camel.CamelContext 当前已停止。在常规 JMS 消费者中默认启用这种快速停止功能，但为了启用此标志，您必须启用此标志。	false	布尔值
consumerType (consumer (advanced))	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。指定 Custom 时，messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 选项将决定要使用的 org.springframework.jms.listener.AbstractMessageListenerContainer。 Enum 值： • Simple（简单） • 默认值 • Custom	默认值	ConsumerType

Name	描述	默认值	类型
defaultTaskExecutorType (consumer (advanced))	指定 DefaultMessageListenerContainer 中使用哪些默认 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 - 缓存的 threadpool-like）。如果没有设置，则默认为前面的行为，它对消费者使用缓存的线程池，并将 SimpleAsync 用于回复消费者。建议使用 ThreadPool 来减少弹性配置中的线程垃圾箱，并动态增加和减少并发消费者。 Enum 值： • ThreadPool • SimpleAsync		DefaultTaskExecutorType
eagerLoadingOfProperties (consumer (advanced))	加载消息后马上启用对 JMS 属性和有效负载的 eager 加载，这通常效率低下，因为 JMS 属性可能不需要，但有时可以尽早地捕获与底层 JMS 提供程序和使用 JMS 属性相关的问题。另请参阅 eagerPoisonBody 选项。	false	布尔值
eagerPoisonBody (consumer (advanced))	如果启用了 eagerLoadingOfProperties，并且 JMS 消息有效负载(JMS 正文或 JMS 属性)为 poison （不能读取/映射），则将此文本设置为消息正文，以便处理消息正文（导致 poison 的原因在 Exchange 上已存储为例外）。这可以通过设置 eagerPoisonBody=false 来关闭。另请参阅选项 eagerLoadingOfProperties。	因 $\{exception.message} 导致的 Poison JMS 消息	字符串
exposeListenerSession (consumer (advanced))	指定在消耗消息时是否应公开侦听器会话。	false	布尔值

Name	描述	默认值	类型
replyToConsumerType (consumer (advanced))	回复消费者的消费者类型（在执行请求/回复时），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。指定 Custom 时，messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 选项将决定要使用的 org.springframework.jms.listener.AbstractMessageListenerContainer。 Enum 值： ● Simple（简单） ● 默认值 ● Custom	默认值	ConsumerType
replyToSameDestinationAllowed (consumer (advanced))	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发送相同消息到其自身的无端循环。	false	布尔值
taskExecutor (consumer (advanced))	允许您指定自定义任务 executor 以供使用消息。		TaskExecutor
deliveryDelay (producer)	设置用于为 JMS 发送调用的交付延迟。这个选项需要 JMS 2.0 兼容代理。	-1	long
deliveryMode (producer)	指定要使用的交付模式。可能的值有 javax.jms.DeliveryMode 定义的值。NON_PERSISTENT = 1 和 PERSISTENT = 2. Enum 值： ● 1 ● 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)	设定在发送消息时应使用 deliveryMode、priority 或 timeToLiveity 的服务质量。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用于当前端点。这与 preserveMessageQos 选项不同，该选项以消息粒度运行，读取仅来自 Camel In 消息标头的 QoS 属性。	false	布尔值
formatDateHeadersToIso8601 (producer)	设置 JMS 日期属性是否应根据 ISO 8601 标准进行格式化。	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
preserveMessageQos (producer)	如果设置为 true，如果要使用消息中指定的 QoS 设置发送消息，而不是 JMS 端点上的 QoS 设置。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供 all 或 only some them。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 选项将使用端点上设置的选项，而不是来自消息标头的值。	false	布尔值
priority (producer)	大于 1 的值在发送时指定消息优先级（其中 1 是最低优先级，9 为最高）。还必须启用 explicitQosEnabled 选项，以便此选项有任何效果。 Enum 值： • 1 • 2 • 3 • 4 • 5 • 6 • 7 • 8 • 9	4	int

Name	描述	默认值	类型
replyToConcurrentConsumers (producer)	指定在对 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，以控制线程的动态扩展/关闭。	1	int
replyToMaxConcurrentConsumers (producer)	指定在 JMS 上使用请求/回复时的最大并发用户数。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。		int
replyToOnTimeoutMaxcurrentConsumers (producer)	指定在通过 JMS 使用请求/回复时，进行超时时继续路由的并发消费者的最大数量。	1	int
replyToOverride (producer)	在 JMS 消息中提供显式 ReplyTo 目的地，它会覆盖 replyTo 的设置。如果您要将消息转发到远程队列，并从 ReplyTo 目的地接收回复消息，这很有用。		字符串
replyToType (producer)	在通过 JMS 进行 request/reply 时，允许显式指定用于 replyTo 队列的策略类型。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。这个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的影响的信息，以及共享回复队列的性能比其 alternatives Temporary 和 Exclusive 的性能较低。 Enum 值： • 临时 • 共享 • exclusive		ReplyToType
requestTimeout (producer)	使用 InOut Exchange Pattern （毫秒）时等待回复的超时时间。默认值为 20 秒。您可以包含标头 CamelJmsRequestTimeout 来覆盖这个端点配置的超时值，因此每个消息单个超时值。另请参阅 requestTimeoutCheckerInterval 选项。	20000	long
timeToLive (producer)	在发送消息时，指定消息的生存时间（以毫秒为单位）。	-1	long
allowAdditionalHeaders (producer (advanced))	此选项用于允许其他标头，它们可能具有根据 JMS 规范无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 来执行此操作，其中包含带有字节数组或其他无效类型的值。您可以指定多个标头名称，用逗号分开，并使用 作为通配符匹配的后缀。		字符串

Name	描述	默认值	类型
allowNullBody (producer (advanced))	是否允许发送不包含正文的消息。如果此选项为 false，并且消息正文为 null，则会抛出一个 JMSException。	true	布尔值
alwaysCopyMessage (producer (advanced))	如果为 true，Camel 会在传递给生成者发送时始终生成邮件的 JMS 消息副本。在某些情况下需要复制消息，比如当设置了 replyToDestinationSelectorName 时（通常，如果设置了 replyToDestinationSelectorName），则 Camel 会将 alwaysCopyMessage 选项设置为 true。	false	布尔值
correlationProperty (producer (advanced))	使用 InOut 交换模式时，请使用此 JMS 属性而不是 JMSCorrelationID JMS 属性来关联消息。如果设置消息将只与此属性的 JMSCorrelationID 属性的值关联，则忽略且不由 Camel 设置。		字符串
disableTimeToLive (producer (advanced))	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 值作为发送消息的时间。问题是发送方和接收器系统必须同步其时钟，因此它们同步。这并非始终容易存档。因此，您可以使用 disableTimeToLive=true 来在发送的消息上将时间设置为 live 值。然后，该消息不会在接收器系统上过期。如需了解更多详细信息，请参见以下小节中关于 live 的时间。	false	布尔值
forceSendOriginalMessage (producer (advanced))	使用 mapJmsMessage=false Camel 时，如果您在路由过程中涉及标头(get 或 set)，则创建新的 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 (advanced))	仅在使用 InOnly 发送到 JMS 目的地时（例如触发和忘记）。启用此选项将增强 Camel Exchange 与实际的 JMSMessageID，在消息发送到 JMS 目的地时供 JMS 客户端使用。	false	布尔值

Name	描述	默认值	类型
replyToCacheLevelName (producer (advanced))	在通过 JMS 进行请求/回复时，根据回复消费者设置缓存级别。这个选项只适用于使用固定回复队列（而非临时）。默认情况下，Camel 将使用：CACHE_CONSUMER 用于 exclusive 或 shared w/ replyToSelectorName。用于没有 replyToSelectorName 的共享的 CACHE_SESSION。IBM WebSphere 等一些 JMS 代理可能需要设置 replyToCacheLevelName=CACHE_NONE 才能工作。注意：如果使用临时队列，则不允许 CACHE_NONE，您必须使用更高的值，如 CACHE_CONSUMER 或 CACHE_SESSION。 Enum 值： • CACHE_AUTO • CACHE_CONNECTION • CACHE_CONSUMER • CACHE_NONE • CACHE_SESSION		字符串
replyToDestinationSelectorName (producer (advanced))	使用要使用的固定名称设置 JMS Selector，以便在使用共享队列时过滤您自己的回复（即，如果您不使用临时回复队列）。		字符串
streamMessageTypeEnabled (producer (advanced))	设置 StreamMessage 类型是否已启用。通过以 BytesMessage 或 StreamMessage 发送的消息有效负载，如 files、InputStream 等。这个选项控制将使用的 kind。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。通过启用这个选项，消息有效负载以块的形式读取到内存中，每个块都会被写入 StreamMessage，直到没有更多数据。	false	布尔值
allowAutoWiredConnectionFactory (advanced)	如果没有配置连接工厂，是否从 registry 中自动发现 ConnectionFactory。如果只找到了一个 ConnectionFactory 实例，则会使用它。这默认是启用的。	true	布尔值
allowAutoWiredDestinationResolver (advanced)	如果没有配置目标解析器，是否从 registry 中自动发现 DestinationResolver。如果只找到一个 DestinationResolver 实例，则会使用它。这默认是启用的。	true	布尔值
allowSerializedHeaders (advanced)	控制是否包含序列化标头。仅在 transferExchange 为 true 时应用。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值

Name	描述	默认值	类型
artemisStreamingEnabled (advanced)	是否针对 Apache Artemis 流模式进行优化。当将 Artemis 与 JMS StreamMessage 类型一起使用时，这可以减少内存开销。只有在使用 Apache Artemis 时，才必须启用此选项。	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 无法获得与远程 JMS 代理的连接，那么在重试和/或故障转移时可能会阻止它。这会导致 Camel 在启动路由时阻止。通过将这个选项设置为 true，您可以让路由启动，而 JmsConsumer 使用异步模式的专用线程连接到 JMS 代理。如果使用此选项，请注意，如果无法建立连接，则会在 WARN 级别记录异常，消费者将无法接收消息；然后，您可以重启要重试的路由。	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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
configuration (advanced)	使用共享的 JMS 配置。		JmsConfiguration
destinationResolver (advanced)	可插拔 org.springframework.jms.support.destination.DestinationResolver，允许您使用自己的解析器（例如，在 JNDI 注册表中查找实际目的地）。		DestinationResolver
errorHandler (advanced)	指定在处理消息时抛出异常时调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置 errorHandler，则会在 WARN 级别中记录这些例外。您可以配置日志记录级别，以及堆栈跟踪是否应该使用 errorHandlerLoggingLevel 和 errorHandlerLogStackTrace 选项记录。这样可以更容易配置，而不必对自定义 errorHandler 进行编码。		ErrorHandler
exceptionListener (advanced)	指定要收到任何底层 JMS 异常的 JMS Exception Listener。		ExceptionListener
idleConsumerLimit (advanced)	指定允许在任何给定时间闲置的用户数量的限制。	1	int

Name	描述	默认值	类型
idleTaskExecutionLimit (advanced)	指定接收任务的闲置执行的限制，而不是在其执行中收到任何消息。如果达到这个限制，则任务将关闭并离开接收其他执行任务（在动态调度的情况下；请参阅 maxConcurrentConsumers 设置）。Spring 中还有额外的文档。	1	int
includeAllJMSXProperties (advanced)	在从 JMS 到 Camel 消息映射时，是否包含所有 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，则这个选项不适用。	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。默认策略将安全地 marshal 句点和连字符(. 和 -)。passthrough 策略将密钥保留为原样。可用于 JMS 代理，这些代理不关心 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。 Enum 值： • default • passthrough		JmsKeyFormatStrategy
mapJmsMessage (advanced)	指定 Camel 是否将收到的 JMS 消息自动映射到适合的有效负载类型，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（例如 min max），则此选项可用于设置 eg 100 来控制在需要较少工作时消费者缩小的速度。	-1	int
messageConverter (advanced)	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)	使用在 Camel 发送 JMS 消息时调用的 given MessageCreatedStrategy，在 Camel 创建 javax.jms.Message 对象的新实例时调用。		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 供应商接受这个 hint，则这些消息必须将消息 ID 设置为 null；如果供应商忽略 hint，则消息 ID 必须设置为其正常唯一值。	true	布尔值

Name	描述	默认值	类型
messageListenerContainerFactory (advanced)	用于决定使用消息的 org.springframework.jms.listener.AbstractMessageListenerContainer 的 MessageListenerContainerFactory 的 registry ID。设置此选项会自动将 consumerType 设置为 Custom。		MessageListenerContainerFactory
messageTimestampEnabled (advanced)	指定默认情况下，是否应在发送消息时启用时间戳。这只是 JMS 代理的提示。如果 JMS 供应商接受这个 hint，则这些消息必须将时间戳设置为零；如果供应商忽略 hint，则必须将时间戳设置为其正常值。	true	布尔值
pubSubNoLocal (advanced)	指定是否禁止发送其自身连接发布的消息。	false	布尔值
queueBrowseStrategy (advanced)	在浏览队列时使用自定义 QueueBrowseStrategy。		QueueBrowseStrategy
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，即刷新连接时（以毫秒为单位）。默认值为 5000 ms，即 5 秒。	5000	long
requestTimeoutCheckerInterval (advanced)	配置 Camel 在通过 JMS 进行请求/回复时应检查超时交换的频率。默认情况下，Camel 会每秒检查一次。但是，如果您在超时时必须更快地响应，您可以降低这个间隔，以便更频繁地检查。超时由选项 requestTimeout 决定。	1000	long
同步 (advanced)	设置是否应严格使用同步处理。	false	布尔值
transferException (advanced)	如果启用了，并且您在使用 Request Reply messaging (InOut)，且 Exchange 在消费者端失败，则原因例外将作为 javax.jms.ObjectMessage 发回。如果客户端是 Camel，则返回的例外将被重新箭头。这样，您可以在路由中使用 Camel JMS 作为网桥 - 例如，使用持久性队列启用可靠的路由。请注意，如果您也启用了 transferExchange，这个选项将具有优先权。需要 Caught 异常才能按顺序排序。消费者端的原始 Exception 可以嵌套在外部异常中，如返回到制作者时 org.apache.camel.RuntimeCamelException。请小心谨慎，因为数据使用 Java 对象序列化，并且要求收到的能够在类级别上反序列化数据，这会强行生产者和消费者之间的强耦合。	false	布尔值

Name	描述	默认值	类型
transferExchange (advanced)	您可以通过线路传输交换，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 headers, exchange properties, exchange exception。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，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 知道有效负载是一个交换而不是常规有效负载。请小心谨慎，因为数据使用 Java 对象序列化，并且要求接收器能够在类级别上反序列化数据，这会强制生产者与需要使用兼容 Camel 版本的消费者之间强耦耦合！	false	布尔值
useMessageIDAsCorrelationID (advanced)	指定 JMSMessageID 是否始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值
waitForProvisionCorrelationToBeUpdatedCounter (advanced)	在通过 JMS 进行请求/回复时，等待 provisional correlation id 更新为实际关联 ID 的次数，以及启用选项 useMessageIDAsCorrelationID 的时间。	50	int
waitForProvisionCorrelationToBeUpdatedThreadSleepingTime (advanced)	等待更新调配关联 id 期间每次处于睡眠状态的间隔。	100	long
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头到 Camel 消息。		HeaderFilterStrategy
errorHandlerLoggingLevel (logging)	允许为日志记录无法捕获的异常配置默认 errorHandler 日志记录级别。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	WARN	LoggingLevel
errorHandlerLogStackTrace (logging)	允许默认 errorHandler 控制是否应记录 stacktrace。	true	布尔值

Name	描述	默认值	类型
password (security)	用于 ConnectionFactory 的密码。您也可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
用户名 （安全性）	用于 ConnectionFactory 的用户名。您也可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
转换 （事务）	指定是否使用 transacted 模式。	false	布尔值
transactedInOut (transaction)	指定 InOut 操作（请求回复）是否默认使用 transacted 模式（如果此标志设为 true），则 Spring JmsTemplate 会将 sessionTransacted 设置为 true，以及 acknowledgeMode 作为 InOut 操作的 JmsTemplate 的 transacted。注意从 Spring JMS：在 JTA 事务中，传递给 createQueue 的参数，不会考虑 createTopic 方法。根据 Java EE 事务上下文，容器对这些值自行做出决定。类似地，这些参数不会被在本地管理的事务中考虑，因为本例中的 Spring JMS 在现有的 JMS Session 上运行。在受管事务之外运行时，将此标志设置为 true 将使用简短的本地 JMS 事务，并在存在受管事务（并非 XA 事务）的情况下同步的本地 JMS 事务。这将与主事务一起管理本地 JMS 事务（可能是原生 JDBC 事务），而 JMS 事务会在主事务后提交右边。	false	布尔值
lazyCreateTransactionManager (transaction (advanced))	如果为 true，则 Camel 将创建一个 JmsTransactionManager，如果没有在选项 transacted=true 时注入任何 transactionManager。	true	布尔值
transactionManager (transaction (advanced))	要使用的 Spring 事务管理器。		PlatformTransactionManager
transactionName (transaction (advanced))	要使用的事务的名称。		字符串
transactionTimeout (transaction (advanced))	使用 transacted 模式，事务的超时值（以秒为单位）。	-1	int

2.5. 端点选项

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：

2.5.1. 路径参数(2 参数)

Name	描述	默认值	类型
destinationType (common)	要使用的目标类型。 Enum 值： • queue • topic • temp-queue • temp-topic	queue	字符串
destinationName (common)	用作目标的队列或主题 **必需** 名称。		字符串

2.5.2. 查询参数(96 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，且只能被单个 JMS 连接实例使用。它通常只需要使用 JMS 1.1 的持久主题订阅。**clientId** 选项与 JMS 1.1 durable 主题订阅相一致，因为客户端 ID 用于控制必须存储哪些客户端消息。使用 JMS 2.0 客户端时，clientId 可能会省略，这会创建一个 'global' 订阅。		字符串
ConnectionFactory (common)	要使用的连接工厂。连接工厂必须在组件或端点上配置。		ConnectionFactory
disableReplyTo (common)	指定 Camel 是否忽略消息中的 JMSReplyTo 标头。如果为 true，Camel 不会向 JMSReplyTo 标头中指定的目的地发送回复。如果您希望 Camel 消耗路由，且您不想 Camel 自动发送回复消息，您可以使用这个选项，因为代码中的另一个组件处理回复消息。如果要使用 Camel 作为不同消息代理之间的代理，而您想要将消息从一个系统路由到另一个系统，也可以使用此选项。	false	布尔值
durableSubscriptionName (common)	用于指定持久主题订阅的持久订阅者名称。必须为 JMS 1.1 持久订阅配置 clientId 选项，并且可以针对 JMS 2.0 配置，以创建私有持久订阅。		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 实现来发送 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 body 类型中使用的 JMS 消息类型。这个选项允许您指定它。 Enum 值： • Bytes • Map • 对象 • Stream • 文本		JmsMessageType
replyTo (common)	提供显式 ReplyTo 目的地（会覆盖消费者中的 Message.getJMSReplyTo （）的任何传入值）。		字符串
testConnectionOnStartup (common)	指定是否在启动时测试连接。这样可确保 Camel 启动所有 JMS 用户具有与 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 没有使用失败的连接启动。JMS 制作者也经过测试。	false	布尔值
confirmmentModeName (consumer)	JMS 确认名称，即： SESSION_TRANSACTED、CLIENT_ACKNOWLEDGE、AUTO_ACKNOWLEDGE、DUPS_OK_ACKNOWLEDGE。 Enum 值： • SESSION_TRANSACTED • CLIENT_ACKNOWLEDGE • AUTO_ACKNOWLEDGE • DUPS_OK_ACKNOWLEDGE	AUTO_ACKNOWLEDGE	字符串
artemisConsumerPriority (consumer)	通过消费者优先级，您可以确保高优先级消费者在激活时收到消息。通常，连接到队列的活动用户以轮循方式从它接收消息。使用消费者优先级时，如果有多个活跃的消费者具有相同的高优先级，则会发送循环消息。只有高优先级消费者没有使用消息的信用时，消息才会受到较低优先级的用户，或者那些高优先级消费者已拒绝接受该消息（例如，因为它不符合与消费者关联的任何选择器的条件）。		int

Name	描述	默认值	类型
asyncConsumer (consumer)	JmsConsumer 是否异步处理 Exchange。如果启用，则 JmsConsumer 可能会从 JMS 队列中获取下一个消息，而前面的消息会被异步处理（通过异步路由引擎）。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 从 JMS 队列获取下一个消息前完全处理 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该自动启动。	true	布尔值
cacheLevel (consumer)	根据 ID 为底层 JMS 资源设置缓存级别。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。 Enum 值： • CACHE_AUTO • CACHE_CONNECTION • CACHE_CONSUMER • CACHE_NONE • CACHE_SESSION	CACHE_AUTO	字符串
concurrentConsumers (consumer)	指定从 JMS 消耗时的默认并发用户数（不适用于 JMS 上的请求/回复）。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。当对 JMS 进行请求/回复时，选项 replyToConcurrentConsumers 用于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（不适用于 JMS 上的请求/回复）。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。当对 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消费者数量。		int
replyToDeliveryPersistent (consumer)	指定是否默认使用持久性发送进行回复。	true	布尔值

Name	描述	默认值	类型
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化。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。在带有共享或持久订阅的主题(pub-sub domain)时应用。如果配置了客户端 ID，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：每个订阅只允许 1 个并发消费者（这是此消息监听程序容器的默认值），除了一个共享订阅（需要 JMS 2.0）。		字符串
subscriptionShared (consumer)	设置是否共享订阅。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 来注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以作为订阅名称使用）。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)时有意义，因此此方法也会切换 pubSubDomain 标志。需要 JMS 2.0 兼容消息代理。	false	布尔值
acceptMessagesWhileStopping (consumer (advanced))	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，您可以考虑启用此选项，同时仍然在队列上排队消息。如果此选项为 false，并且您停止了 JMS 路由，则消息可能会被拒绝，而 JMS 代理必须尝试重新设计，这一次可能被拒绝，最终该消息可能会在 JMS 代理上的死信队列中移动。要避免这种情况，建议启用这个选项。	false	布尔值
allowReplyManagerQuickStop (consumer (advanced))	是否在回复管理器中使用 DefaultMessageListenerContainer 用于 request-reply 消息，都允许 DefaultMessageListenerContainer.runningAllowed 标志在启用了 JmsConfiguration#isAcceptMessagesWhileStopping 时快速停止，并且 org.apache.camel.CamelContext 当前已停止。在常规 JMS 消费者中默认启用这种快速停止功能，但为了启用此标志，您必须启用此标志。	false	布尔值

Name	描述	默认值	类型
consumerType (consumer (advanced))	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。指定 Custom 时，messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 选项将决定要使用的 org.springframework.jms.listener.AbstractMessageListenerContainer。 Enum 值： ● Simple（简单） ● 默认值 ● Custom	默认值	ConsumerType
defaultTaskExecutorType (consumer (advanced))	指定 DefaultMessageListenerContainer 中使用哪些默认 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 - 缓存的 threadpool-like）。如果没有设置，则默认为前面的行为，它对消费者使用缓存的线程池，并将 SimpleAsync 用于回复消费者。建议使用 ThreadPool 来减少弹性配置中的线程垃圾箱，并动态增加和减少并发消费者。 Enum 值： ● ThreadPool ● SimpleAsync		DefaultTaskExecutorType
eagerLoadingOfProperties (consumer (advanced))	加载消息后马上启用对 JMS 属性和有效负载的 eager 加载，这通常效率低下，因为 JMS 属性可能不需要，但有时可以尽早地捕获与底层 JMS 提供程序和使用 JMS 属性相关的问题。另请参阅 eagerPoisonBody 选项。	false	布尔值

Name	描述	默认值	类型
eagerPoisonBody (consumer (advanced))	如果启用了 eagerLoadingOfProperties，并且 JMS 消息有效负载(JMS 正文或 JMS 属性)为 poison （不能读取/映射），则将此文本设置为消息正文，以便处理消息正文（导致 poison 的原因在 Exchange 上已存储为例外）。这可以通过设置 eagerPoisonBody=false 来关闭。另请参阅选项 eagerLoadingOfProperties。	因 $\{exception.message} 导致的 Poison JMS 消息	字符串
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
exposeListenerSession (consumer (advanced))	指定在消耗消息时是否应公开侦听器会话。	false	布尔值
replyToConsumerType (consumer (advanced))	回复消费者的消费者类型（在执行请求/回复时），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。指定 Custom 时，messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 选项将决定要使用的 org.springframework.jms.listener.AbstractMessageListenerContainer。 Enum 值： • Simple（简单） • 默认值 • Custom	默认值	ConsumerType

Name	描述	默认值	类型
replyToSameDestinationAllowed (consumer (advanced))	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发送相同消息到其自身的无端循环。	false	布尔值
taskExecutor (consumer (advanced))	允许您指定自定义任务 executor 以供使用消息。		TaskExecutor
deliveryDelay (producer)	设置用于为 JMS 发送调用的交付延迟。这个选项需要 JMS 2.0 兼容代理。	-1	long
deliveryMode (producer)	指定要使用的交付模式。可能的值有 javax.jms.DeliveryMode 定义的值。NON_PERSISTENT = 1 和 PERSISTENT = 2. Enum 值： • 1 • 2		整数
deliveryPersistent (producer)	指定是否默认使用持久性交付。	true	布尔值
explicitQosEnabled (producer)	设定在发送消息时应使用 deliveryMode、priority 或 timeToLiveity 的服务质量。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用于当前端点。这与 preserveMessageQos 选项不同，该选项以消息粒度运行，读取仅来自 Camel In 消息标头的 QoS 属性。	false	布尔值
formatDateHeadersToIso8601 (producer)	设置 JMS 日期属性是否应根据 ISO 8601 标准进行格式化。	false	布尔值
preserveMessageQos (producer)	如果设置为 true，如果要使用消息中指定的 QoS 设置发送消息，而不是 JMS 端点上的 QoS 设置。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供 all 或 only some them。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 选项将使用端点上设置的选项，而不是来自消息标头的值。	false	布尔值

Name	描述	默认值	类型
priority (producer)	大于 1 的值在发送时指定消息优先级（其中 1 是最低优先级，9 为最高）。还必须启用 explicitQosEnabled 选项，以便此选项有任何效果。 Enum 值： • 1 • 2 • 3 • 4 • 5 • 6 • 7 • 8 • 9	4	int
replyToConcurrentConsumers (producer)	指定在对 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，以控制线程的动态扩展/关闭。	1	int
replyToMaxConcurrentConsumers (producer)	指定在 JMS 上使用请求/回复时的最大并发用户数。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。		int
replyToOnTimeoutMaxcurrentConsumers (producer)	指定在通过 JMS 使用请求/回复时，进行超时时继续路由的并发消费者的最大数量。	1	int
replyToOverride (producer)	在 JMS 消息中提供显式 ReplyTo 目的地，它会覆盖 replyTo 的设置。如果您要将消息转发到远程队列，并从 ReplyTo 目的地接收回复消息，这很有用。		字符串

Name	描述	默认值	类型
replyToType (producer)	在通过 JMS 进行 request/reply 时，允许显式指定用于 replyTo 队列的策略类型。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。这个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的影响的信息，以及共享回复队列的性能比其 alternatives Temporary 和 Exclusive 的性能较低。 Enum 值： • 临时 • 共享 • exclusive		ReplyToType
requestTimeout (producer)	使用 InOut Exchange Pattern （毫秒）时等待回复的超时时间。默认值为 20 秒。您可以包含标头 CamelJmsRequestTimeout 来覆盖这个端点配置的超时值，因此每个消息单个超时值。另请参阅 requestTimeoutCheckerInterval 选项。	20000	long
timeToLive (producer)	在发送消息时，指定消息的生存时间（以毫秒为单位）。	-1	long
allowAdditionalHeaders (producer (advanced))	此选项用于允许其他标头，它们可能具有根据 JMS 规范无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 来执行此操作，其中包含带有字节数组或其他无效类型的值。您可以指定多个标头名称，用逗号分开，并使用 作为通配符匹配的后缀。		字符串
allowNullBody (producer (advanced))	是否允许发送不包含正文的消息。如果此选项为 false，并且消息正文为 null，则会抛出一个 JMSException。	true	布尔值
alwaysCopyMessage (producer (advanced))	如果为 true，Camel 会在传递给生成者发送时始终生成邮件的 JMS 消息副本。在某些情况下需要复制消息，比如当设置了 replyToDestinationSelectorName 时（通常，如果设置了 replyToDestinationSelectorName），则 Camel 会将 alwaysCopyMessage 选项设置为 true。	false	布尔值
correlationProperty (producer (advanced))	使用 InOut 交换模式时，请使用此 JMS 属性而不是 JMSCorrelationID JMS 属性来关联消息。如果设置消息将只与此属性的 JMSCorrelationID 属性的值关联，则忽略且不由 Camel 设置。		字符串

Name	描述	默认值	类型
disableTimeToLive (producer (advanced))	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 值作为发送消息的时间。问题是发送方和接收器系统必须同步其时钟，因此它们同步。这并非始终容易存档。因此，您可以使用 disableTimeToLive=true 来在发送的消息上将时间设置为 live 值。然后，该消息不会在接收器系统上过期。如需了解更多详细信息，请参见以下小节中关于 live 的时间。	false	布尔值
forceSendOriginalMessage (producer (advanced))	使用 mapJmsMessage=false Camel 时，如果您在路由过程中涉及标头(get 或 set)，则创建新的 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 (advanced))	仅在使用 InOnly 发送到 JMS 目的地时（例如触发和忘记）。启用此选项将增强 Camel Exchange 与实际的 JMSMessageID，在消息发送到 JMS 目的地时供 JMS 客户端使用。	false	布尔值
lazyStartProducer (producer (advanced))	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
replyToCacheLevelName (producer (advanced))	在通过 JMS 进行请求/回复时，根据回复消费者设置缓存级别。这个选项只适用于使用固定回复队列（而非临时）。默认情况下，Camel 将使用：CACHE_CONSUMER 用于 exclusive 或 shared w/ replyToSelectorName。用于没有 replyToSelectorName 的共享的 CACHE_SESSION。IBM WebSphere 等一些 JMS 代理可能需要设置 replyToCacheLevelName=CACHE_NONE 才能工作。注意：如果使用临时队列，则不允许 CACHE_NONE，您必须使用更高的值，如 CACHE_CONSUMER 或 CACHE_SESSION。 Enum 值： • CACHE_AUTO • CACHE_CONNECTION • CACHE_CONSUMER • CACHE_NONE • CACHE_SESSION		字符串

Name	描述	默认值	类型
replyToDestinationSelectorName (producer (advanced))	使用要使用的固定名称设置 JMS Selector，以便在使用共享队列时过滤您自己的回复（即，如果您不使用临时回复队列）。		字符串
streamMessageTypeEnabled (producer (advanced))	设置 StreamMessage 类型是否已启用。通过以 BytesMessage 或 StreamMessage 发送的消息有效负载，如 files、InputStream 等。这个选项控制将使用的 kind。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。通过启用这个选项，消息有效负载以块的形式读取到内存中，每个块都会被写入 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 时应用。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值
artemisStreamingEnabled (advanced)	是否针对 Apache Artemis 流模式进行优化。当将 Artemis 与 JMS StreamMessage 类型一起使用时，这可以减少内存开销。只有在使用 Apache Artemis 时，才必须启用此选项。	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 无法获得与远程 JMS 代理的连接，那么在重试和/或故障转移时可能会阻止它。这会导致 Camel 在启动路由时阻止。通过将这个选项设置为 true，您可以让路由启动，而 JmsConsumer 使用异步模式的专用线程连接到 JMS 代理。如果使用此选项，请注意，如果无法建立连接，则会在 WARN 级别记录异常，消费者将无法接收消息；然后，您可以重启要重试的路由。	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)	指定在处理消息时抛出异常时调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置 errorHandler，则会在 WARN 级别中记录这些例外。您可以配置日志记录级别，以及堆栈跟踪是否应该使用 errorHandlerLoggingLevel 和 errorHandlerLogStackTrace 选项记录。这样可以更容易配置，而不必对自定义 errorHandler 进行编码。		ErrorHandler
exceptionListener (advanced)	指定要收到任何底层 JMS 异常的 JMS Exception Listener。		ExceptionListener

Name	描述	默认值	类型
headerFilterStrategy (advanced)	使用自定义 HeaderFilterStrategy 过滤标头到 Camel 消息。		HeaderFilterStrategy
idleConsumerLimit (advanced)	指定允许在任何给定时间闲置的用户数量的限制。	1	int
idleTaskExecutionLimit (advanced)	指定接收任务的闲置执行的限制，而不是在其执行中收到任何消息。如果达到这个限制，则任务将关闭并离开接收其他执行任务（在动态调度的情况下；请参阅 maxConcurrentConsumers 设置）。Spring 中还有额外的文档。	1	int
includeAllJMSXProperties (advanced)	在从 JMS 到 Camel 消息映射时，是否包含所有 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，则这个选项不适用。	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。默认策略将安全地 marshal 句点和连字符(. 和 -)。passthrough 策略将密钥保留为原样。可用于 JMS 代理，这些代理不关心 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。 Enum 值： • default • passthrough		JmsKeyFormatStrategy
mapJmsMessage (advanced)	指定 Camel 是否将收到的 JMS 消息自动映射到适合的有效负载类型，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（例如 min max），则此选项可用于设置 eg 100 来控制在需要较少工作时消费者缩小的速度。	-1	int
messageConverter (advanced)	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

Name	描述	默认值	类型
messageCreatedStrategy (advanced)	使用在 Camel 发送 JMS 消息时调用的 given MessageCreatedStrategy，在 Camel 创建 javax.jms.Message 对象的新实例时调用。		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 供应商接受这个 hint，则这些消息必须将消息 ID 设置为 null；如果供应商忽略 hint，则消息 ID 必须设置为其正常唯一值。	true	布尔值
messageListenerContainerFactory (advanced)	用于决定使用消息的 org.springframework.jms.listener.AbstractMessageListenerContainer 的 MessageListenerContainerFactory 的 registry ID。设置此选项会自动将 consumerType 设置为 Custom。		MessageListenerContainerFactory
messageTimestampEnabled (advanced)	指定默认情况下，是否应在发送消息时启用时间戳。这只是 JMS 代理的提示。如果 JMS 供应商接受这个 hint，则这些消息必须将时间戳设置为零；如果供应商忽略 hint，则必须将时间戳设置为其正常值。	true	布尔值
pubSubNoLocal (advanced)	指定是否禁止发送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，即刷新连接时（以毫秒为单位）。默认值为 5000 ms，即 5 秒。	5000	long
requestTimeoutCheckerInterval (advanced)	配置 Camel 在通过 JMS 进行请求/回复时应检查超时交换的频率。默认情况下，Camel 会每秒检查一次。但是，如果您在超时时必须更快地响应，您可以降低这个间隔，以便更频繁地检查。超时由选项 requestTimeout 决定。	1000	long
同步 (advanced)	设置是否应严格使用同步处理。	false	布尔值

Name	描述	默认值	类型
transferException (advanced)	如果启用了，并且您在使用 Request Reply messaging (InOut)，且 Exchange 在消费者端失败，则原因例外将作为 javax.jms.ObjectMessage 发回。如果客户端是 Camel，则返回的例外将被重新箭头。这样，您可以在路由中使用 Camel JMS 作为网桥 - 例如，使用持久性队列启用可靠的路由。请注意，如果您也启用了 transferExchange，这个选项将具有优先权。需要 Caught 异常才能按顺序排序。消费者端的原始 Exception 可以嵌套在外部异常中，如返回到制作者时 org.apache.camel.RuntimeCamelException。请小心谨慎，因为数据使用 Java 对象序列化，并且要求收到的能够在类级别上反序列化数据，这会强行生产者和消费者之间的强耦合。	false	布尔值
transferExchange (advanced)	您可以通过线路传输交换，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 headers, exchange properties, exchange exception。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，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 知道有效负载是一个交换而不是常规有效负载。请小心谨慎，因为数据使用 Java 对象序列化，并且要求接收器能够在类级别上反序列化数据，这会强制生产者与需要使用兼容 Camel 版本的消费者之间强耦耦合！	false	布尔值
useMessageIDAsCorrelationID (advanced)	指定 JMSMessageID 是否始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值
waitForProvisionCorrelationToBeUpdatedCounter (advanced)	在通过 JMS 进行请求/回复时，等待 provisional correlation id 更新为实际关联 ID 的次数，以及启用选项 useMessageIDAsCorrelationID 的时间。	50	int
waitForProvisionCorrelationToBeUpdatedThreadSleepingTime (advanced)	等待更新调配关联 id 期间每次处于睡眠状态的间隔。	100	long

Name	描述	默认值	类型
errorHandlerLoggingLevel (logging)	允许为日志记录无法捕获的异常配置默认 errorHandler 日志记录级别。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	WARN	LoggingLevel
errorHandlerLogStackTrace (logging)	允许默认 errorHandler 控制是否应记录 stacktrace。	true	布尔值
password (security)	用于 ConnectionFactory 的密码。您也可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
用户名 （安全性）	用于 ConnectionFactory 的用户名。您也可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
转换 （事务）	指定是否使用 transacted 模式。	false	布尔值
transactedInOut (transaction)	指定 InOut 操作（请求回复）是否默认使用 transacted 模式（如果此标志设为 true），则 Spring JmsTemplate 会将 sessionTransacted 设置为 true，以及 acknowledgeMode 作为 InOut 操作的 JmsTemplate 的 transacted。注意从 Spring JMS：在 JTA 事务中，传递给 createQueue 的参数，不会考虑 createTopic 方法。根据 Java EE 事务上下文，容器对这些值自行做出决定。类似地，这些参数不会被在本地管理的事务中考虑，因为本例中的 Spring JMS 在现有的 JMS Session 上运行。在受管事务之外运行时，将此标志设置为 true 将使用简短的本地 JMS 事务，并在存在受管事务（并非 XA 事务）的情况下同步的本地 JMS 事务。这将与主事务一起管理本地 JMS 事务（可能是原生 JDBC 事务），而 JMS 事务会在主事务后提交右边。	false	布尔值
lazyCreateTransactionManager (transaction (advanced))	如果为 true，则 Camel 将创建一个 JmsTransactionManager，如果没有在选项 transacted=true 时注入任何 transactionManager。	true	布尔值

Name	描述	默认值	类型
transactionManager (transaction (advanced))	要使用的 Spring 事务管理器。		PlatformTransactionManager
transactionName (transaction (advanced))	要使用的事务的名称。		字符串
transactionTimeout (transaction (advanced))	使用 transacted 模式，事务的超时值（以秒为单位）。	-1	int

2.6. 使用方法

由于 AMQP 组件继承自 JMS 组件，因此前者的使用与后者几乎相同：

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");
```

2.7. 配置 AMQP 组件

创建 AMQP 1.0 组件

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

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

您还可以在 registry 中添加 **org.apache.camel.component.amqp.amqpConnectionDetails** 的实例，以自动配置 AMQP 组件。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

AMQP 连接详情自动配置 CDI

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

您还可以依赖 来读取 AMQP 连接详情。工厂方法 **AMQPConnectionDetails.discoverAMQP （）** 试图以类似于 Kubernetes 的惯例读取 Camel 属性，如以下代码片段所示：

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 具体选项

例如，如果您需要启用 **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");
```

请参考 QPID JMS 客户端配置。

2.8. 使用主题

要使用 **camel-amqp** 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** 预配置组件带有主题前缀，因此您不必明确配置它。

2.9. SPRING BOOT AUTO-CONFIGURATION

组件支持 101 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，您可以考虑启用此选项，同时仍然在队列上排队消息。如果此选项为 false，并且您停止了 JMS 路由，则消息可能会被拒绝，而 JMS 代理必须尝试重新设计，这一次可能被拒绝，最终该消息可能会在 JMS 代理上的死信队列中移动。要避免这种情况，建议启用这个选项。	false	布尔值
camel.component.amqp.acknowledgement-mode-name	JMS 确认名称，即： SESSION_TRANSACTED、CLIENT_ACKNOWLEDGE、AUTO_ACKNOWLEDGE、DUPS_OK_ACKNOWLEDGE。	AUTO_ACKNOWLEDGE	字符串
camel.component.amqp.allow-additional-headers	此选项用于允许其他标头，它们可能具有根据 JMS 规范无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 来执行此操作，其中包含带有字节数组或其他无效类型的值。您可以指定多个标头名称，用逗号分开，并使用 作为通配符匹配的后缀。		字符串
camel.component.amqp.allow-auto-wired-connection-factory	如果没有配置连接工厂，是否从 registry 中自动发现 ConnectionFactory。如果只找到了一个 ConnectionFactory 实例，则会使用它。这默认是启用的。	true	布尔值
camel.component.amqp.allow-auto-wired-destination-resolver	如果没有配置目标解析器，是否从 registry 中自动发现 DestinationResolver。如果只找到一个 DestinationResolver 实例，则会使用它。这默认是启用的。	true	布尔值
camel.component.amqp.allow-null-body	是否允许发送不包含正文的消息。如果此选项为 false，并且消息正文为 null，则会抛出一个 JMSException。	true	布尔值

Name	描述	默认值	类型
camel.component.amqp.allow-reply-manager-quick-stop	是否在回复管理器中使用 DefaultMessageListenerContainer 用于 request-reply 消息，都允许 DefaultMessageListenerContainer.runningAllowed 标志在启用了 JmsConfiguration#isAcceptMessagesWhileStopping 时快速停止，并且 org.apache.camel.CamelContext 当前已停止。在常规 JMS 消费者中默认启用这种快速停止功能，但为了启用此标志，您必须启用此标志。	false	布尔值
camel.component.amqp.allow-serialized-headers	控制是否包含序列化标头。仅在 transferExchange 为 true 时应用。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值
camel.component.amqp.always-copy-message	如果为 true，Camel 会在传递给生成者发送时始终生成邮件的 JMS 消息副本。在某些情况下需要复制消息，比如当设置了 replyToDestinationSelectorName 时（通常，如果设置了 replyToDestinationSelectorName），则 Camel 会将 alwaysCopyMessage 选项设置为 true。	false	布尔值
camel.component.amqp.artemis-consumer-priority	通过消费者优先级，您可以确保高优先级消费者在激活时收到消息。通常，连接到队列的活动用户以轮循方式从它接收消息。使用消费者优先级时，如果有多个活跃的消费者具有相同的高优先级，则会发送循环消息。只有高优先级消费者没有使用消息的信用时，消息才会受到较低优先级的用户，或者那些高优先级消费者已拒绝接受该消息（例如，因为它不符合与消费者关联的任何选择器的条件）。		整数
camel.component.amqp.artemis-streaming-enabled	是否针对 Apache Artemis 流模式进行优化。当将 Artemis 与 JMS StreamMessage 类型一起使用时，这可以减少内存开销。只有在使用 Apache Artemis 时，才必须启用此选项。	false	布尔值
camel.component.amqp.async-consumer	JmsConsumer 是否异步处理 Exchange。如果启用，则 JmsConsumer 可能会从 JMS 队列中获取下一个消息，而前面的消息会被异步处理（通过异步路由引擎）。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 从 JMS 队列获取下一个消息前完全处理 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

Name	描述	默认值	类型
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 无法获得与远程 JMS 代理的连接，那么在重试和/或故障转移时可能会阻止它。这会导致 Camel 在启动路由时阻止。通过将这个选项设置为 true，您可以让路由启动，而 JmsConsumer 使用异步模式的专用线程连接到 JMS 代理。如果使用此选项，请注意，如果无法建立连接，则会在 WARN 级别记录异常，消费者将无法接收消息；然后，您可以重启要重试的路由。	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该自动启动。	true	布尔值
camel.component.amqp.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.amqp.cache-level	根据 ID 为底层 JMS 资源设置缓存级别。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，且只能被单个 JMS 连接实例使用。它通常只需要使用 JMS 1.1 的持久主题订阅。		字符串
camel.component.amqp.concurrent-consumers	指定从 JMS 消耗时的默认并发用户数（不适用于 JMS 上的请求/回复）。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。当对 JMS 进行请求/回复时，选项 replyToConcurrentConsumers 用于控制回复消息监听器上的并发用户数量。	1	整数
camel.component.amqp.configuration	使用共享的 JMS 配置。选项是 org.apache.camel.component.jms.JmsConfiguration 类型。		JmsConfiguration

Name	描述	默认值	类型
camel.component.amqp.connection-factory	要使用的连接工厂。连接工厂必须在组件或端点上配置。选项是 javax.jms.ConnectionFactory 类型。		ConnectionFactory
camel.component.amqp.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。指定 Custom 时，messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 选项将决定要使用的 org.springframework.jms.listener.AbstractMessageListenerContainer。		ConsumerType
camel.component.amqp.correlation-property	使用 InOut 交换模式时，请使用此 JMS 属性而不是 JMSCorrelationID JMS 属性来关联消息。如果设置消息将只与此属性的 JMSCorrelationID 属性的值关联，则忽略且不由 Camel 设置。		字符串
camel.component.amqp.default-task-executor-type	指定 DefaultMessageListenerContainer 中使用哪些默认 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 - 缓存的 threadpool-like）。如果没有设置，则默认为前面的行为，它对消费者使用缓存的线程池，并将 SimpleAsync 用于回复消费者。建议使用 ThreadPool 来减少弹性配置中的线程垃圾箱，并动态增加和减少并发消费者。		DefaultTaskExecutorType
camel.component.amqp.delivery-delay	设置用于为 JMS 发送调用的交付延迟。这个选项需要 JMS 2.0 兼容代理。	-1	Long
camel.component.amqp.delivery-mode	指定要使用的交付模式。可能的值有 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

Name	描述	默认值	类型
camel.component.amqp.destination-resolver	可插拔 org.springframework.jms.support.destination.DestinationResolver，允许您使用自己的解析器（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。		DestinationResolver
camel.component.amqp.disable-reply-to	指定 Camel 是否忽略消息中的 JMSReplyTo 标头。如果为 true，Camel 不会向 JMSReplyTo 标头中指定的目的地发送回复。如果您希望 Camel 消耗路由，且您不想 Camel 自动发送回复消息，您可以使用这个选项，因为代码中的另一个组件处理回复消息。如果要使用 Camel 作为不同消息代理之间的代理，而您想要将消息从一个系统路由到另一个系统，也可以使用此选项。	false	布尔值
camel.component.amqp.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 值作为发送消息的时间。问题是发送方和接收器系统必须同步其时钟，因此它们同步。这并非始终容易存档。因此，您可以使用 disableTimeToLive=true 来在发送的消息上将时间设置为 live 值。然后，该消息不会在接收器系统上过期。如需了解更多详细信息，请参见以下小节中关于 live 的时间。	false	布尔值
camel.component.amqp.durable-subscription-name	用于指定持久主题订阅的持久订阅者名称。也必须配置 clientId 选项。		字符串
camel.component.amqp.eager-loading-of-properties	加载消息后马上启用对 JMS 属性和有效负载的 eager 加载，这通常效率低下，因为 JMS 属性可能不需要，但有时可以尽早地捕获与底层 JMS 提供程序和使用 JMS 属性相关的问题。另请参阅 eagerPoisonBody 选项。	false	布尔值
camel.component.amqp.eager-poison-body	如果启用了 eagerLoadingOfProperties，并且 JMS 消息有效负载(JMS 正文或 JMS 属性)为 poison （不能读取/映射），则将此文本设置为消息正文，以便处理消息正文（导致 poison 的原因在 Exchange 上已存储为例外）。这可以通过设置 eagerPoisonBody=false 来关闭。另请参阅选项 eagerLoadingOfProperties。	因 $\{exception.message} 导致的 Poison JMS 消息	字符串
camel.component.amqp.enabled	是否启用 amqp 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.amqp.error-handler	指定在处理消息时抛出异常时调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置 errorHandler，则会在 WARN 级别中记录这些例外。您可以配置日志记录级别，以及堆栈跟踪是否应该使用 errorHandlerLoggingLevel 和 errorHandlerLogStackTrace 选项记录。这样可以更容易配置，而不必对自定义 errorHandler 进行编码。选项是一个 org.springframework.util.ErrorHandler 类型。		ErrorHandler
camel.component.amqp.error-handler-log-stack-trace	允许默认 errorHandler 控制是否应记录 stacktrace。	true	布尔值
camel.component.amqp.error-handler-logging-level	允许为日志记录无法捕获的异常配置默认 errorHandler 日志记录级别。		LoggingLevel
camel.component.amqp.exception-listener	指定要收到任何底层 JMS 异常的 JMS Exception Listener。选项是 javax.jms.ExceptionListener 类型。		ExceptionListener
camel.component.amqp.explicit-qos-enabled	设定在发送消息时应使用 deliveryMode、priority 或 timeToLiveity 的服务质量。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用于当前端点。这与 preserveMessageQos 选项不同，该选项以消息粒度运行，读取仅来自 Camel In 消息标头的 QoS 属性。	false	布尔值
camel.component.amqp.expose-listener-session	指定在消耗消息时是否应公开侦听器会话。	false	布尔值
camel.component.amqp.force-send-original-message	使用 mapJmsMessage=false Camel 时，如果您在路由过程中涉及标头(get 或 set)，则创建新的 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.amqp.format-date-headers-to-iso8601	设置 JMS 日期属性是否应根据 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 过滤标头到 Camel 消息。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		HeaderFilterStrategy

Name	描述	默认值	类型
camel.component.amqp.idle-consumer-limit	指定允许在任何给定时间闲置的用户数量的限制。	1	整数
camel.component.amqp.idle-task-execution-limit	指定接收任务的闲置执行的限制，而不是在其执行中收到任何消息。如果达到这个限制，则任务将关闭并离开接收其他执行任务（在动态调度的情况下；请参阅 maxConcurrentConsumers 设置）。Spring 中还有额外的文档。	1	整数
camel.component.amqp.include-all-jmsx-properties	在从 JMS 到 Camel 消息映射时，是否包含所有 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，则这个选项不适用。	false	布尔值
camel.component.amqp.include-amqp-annotations	在从 AMQP 到 Camel 消息映射时是否包含 AMQP 注解。把它设置为 true 将包含 JMS_AMQP_MA_ 前缀的 AMQP 消息注解映射到消息标头。由于 Apache Qpid JMS API 的限制，当前交付注释将被忽略。	false	布尔值
camel.component.amqp.include-sent-jms-message-id	仅在使用 InOnly 发送到 JMS 目的地时（例如触发和忘记）。启用此选项将增强 Camel Exchange 与实际的 JMSMessageID，在消息发送到 JMS 目的地时供 JMS 客户端使用。	false	布尔值
camel.component.amqp.jms-key-format-strategy	用于编码和解码 JMS 密钥的可插拔策略，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。默认策略将安全地 marshal 句点和连字符(. 和 -)。passthrough 策略将密钥保留为原样。可用于 JMS 代理，这些代理不关心 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.amqp.jms-message-type	允许您强制使用特定的 javax.jms.Message 实现来发送 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 body 类型中使用的 JMS 消息类型。这个选项允许您指定它。		JmsMessageType
camel.component.amqp.lazy-create-transaction-manager	如果为 true，则 Camel 将创建一个 JmsTransactionManager，如果没有在选项 transacted=true 时注入任何 transactionManager。	true	布尔值

Name	描述	默认值	类型
camel.component.amqp.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.amqp.map-jms-message	指定 Camel 是否将收到的 JMS 消息自动映射到适合的有效负载类型，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（不适用于 JMS 上的请求/回复）。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。当对 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消费者数量。		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（例如 min max），则此选项可用于设置 eg 100 来控制在需要较少工作时消费者缩小的速度。	-1	整数
camel.component.amqp.message-converter	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。		MessageConverter
camel.component.amqp.message-created-strategy	使用在 Camel 发送 JMS 消息时调用的 given MessageCreatedStrategy，在 Camel 创建 javax.jms.Message 对象的新实例时调用。选项是一个 org.apache.camel.component.jms.MessageCreatedStrategy 类型。		MessageCreatedStrategy
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 供应商接受这个 hint，则这些消息必须将消息 ID 设置为 null；如果供应商忽略 hint，则消息 ID 必须设置为其正常唯一值。	true	布尔值

Name	描述	默认值	类型
camel.component.amqp.message-listener-container-factory	用于决定使用消息的 org.springframework.jms.listener.AbstractMessageListenerContainer 的 MessageListenerContainerFactory 的 registry ID。设置此选项会自动将 consumerType 设置为 Custom。选项是 org.apache.camel.component.jms.MessageListenerContainerFactory 类型。		MessageListenerContainerFactory
camel.component.amqp.message-timestamp-enabled	指定默认情况下，是否应在发送消息时启用时间戳。这只是 JMS 代理的提示。如果 JMS 供应商接受这个 hint，则这些消息必须将时间戳设置为零；如果供应商忽略 hint，则必须将时间戳设置为其正常值。	true	布尔值
camel.component.amqp.password	用于 ConnectionFactory 的密码。您也可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
camel.component.amqp.preserve-message-qos	如果设置为 true，如果要使用消息中指定的 QoS 设置发送消息，而不是 JMS 端点上的 QoS 设置。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供 all 或 only some them。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 选项将使用端点上设置的选项，而不是来自消息标头的值。	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 1 是最低优先级，9 为最高）。还必须启用 explicitQosEnabled 选项，以便此选项有任何效果。	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。		QueueBrowseStrategy
camel.component.amqp.receive-timeout	接收消息的超时时间（以毫秒为单位）。选项是一个长类型。	1000	Long
camel.component.amqp.recovery-interval	指定恢复尝试之间的间隔，即刷新连接时（以毫秒为单位）。默认值为 5000 ms，即 5 秒。选项是一个长类型。	5000	Long

Name	描述	默认值	类型
camel.component.amqp.reply-to	提供显式 ReplyTo 目的地（会覆盖消费者中的 Message.getJMSReplyTo （）的任何传入值）。		字符串
camel.component.amqp.reply-to-cache-level-name	在通过 JMS 进行请求/回复时，根据回复消费者设置缓存级别。这个选项只适用于使用固定回复队列（而非临时）。默认情况下，Camel 将使用：CACHE_CONSUMER 用于 exclusive 或 shared w/ replyToSelectorName。用于没有 replyToSelectorName 的共享的 CACHE_SESSION。IBM WebSphere 等一些 JMS 代理可能需要设置 replyToCacheLevelName=CACHE_NONE 才能工作。注意：如果使用临时队列，则不允许 CACHE_NONE，您必须使用更高的值，如 CACHE_CONSUMER 或 CACHE_SESSION。		字符串
camel.component.amqp.reply-to-concurrent-consumers	指定在对 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，以控制线程的动态扩展/关闭。	1	整数
camel.component.amqp.reply-to-consumer-type	回复消费者的消费者类型（在执行请求/回复时），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。指定 Custom 时，messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 选项将决定要使用的 org.springframework.jms.listener.AbstractMessageListenerContainer。		ConsumerType
camel.component.amqp.reply-to-delivery-persistent	指定是否默认使用持久性发送进行回复。	true	布尔值
camel.component.amqp.reply-to-destination-selector-name	使用要使用的固定名称设置 JMS Selector，以便在使用共享队列时过滤您自己的回复（即，如果您不使用临时回复队列）。		字符串
camel.component.amqp.reply-to-max-concurrent-consumers	指定在 JMS 上使用请求/回复时的最大并发用户数。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。		整数

Name	描述	默认值	类型
camel.component.amqp.reply-to-on-timeout-max-concurrent-consumers	指定在通过 JMS 使用请求/回复时，进行超时时继续路由的并发消费者的最大数量。	1	整数
camel.component.amqp.reply-to-override	在 JMS 消息中提供显式 ReplyTo 目的地，它会覆盖 replyTo 的设置。如果您要将消息转发到远程队列，并从 ReplyTo 目的地接收回复消息，这很有用。		字符串
camel.component.amqp.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发送相同消息到其自身的无端循环。	false	布尔值
camel.component.amqp.reply-to-type	在通过 JMS 进行 request/reply 时，允许显式指定用于 replyTo 队列的策略类型。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。这个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的影响的信息，以及共享回复队列的性能比其 alternatives Temporary 和 Exclusive 的性能较低。		ReplyToType
camel.component.amqp.request-timeout	使用 InOut Exchange Pattern （毫秒）时等待回复的超时时间。默认值为 20 秒。您可以包含标头 CamelJmsRequestTimeout 来覆盖这个端点配置的超时值，因此每个消息单个超时值。另请参阅 requestTimeoutCheckerInterval 选项。选项是一个长类型。	20000	Long
camel.component.amqp.request-timeout-checker-interval	配置 Camel 在通过 JMS 进行请求/回复时应检查超时交换的频率。默认情况下，Camel 会每秒检查一次。但是，如果您在超时时必须更快地响应，您可以降低这个间隔，以便更频繁地检查。超时由选项 requestTimeout 决定。选项是一个长类型。	1000	Long
camel.component.amqp.selector	设置要使用的 JMS 选择器。		字符串

Name	描述	默认值	类型
camel.component.amqp.stream-message-type-enabled	设置 StreamMessage 类型是否已启用。通过以 BytesMessage 或 StreamMessage 发送的消息有效负载，如 files、InputStream 等。这个选项控制将使用的 kind。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。通过启用这个选项，消息有效负载以块的形式读取到内存中，每个块都会被写入 StreamMessage，直到没有更多数据。	false	布尔值
camel.component.amqp.subscription-durable	设置是否使订阅持久化。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。在带有共享或持久订阅的主题(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：每个订阅只允许 1 个并发消费者（这是此消息监听程序容器的默认值），除了一个共享订阅（需要 JMS 2.0）。		字符串
camel.component.amqp.subscription-shared	设置是否共享订阅。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 来注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以作为订阅名称使用）。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)时有意义，因此此方法也会切换 pubSubDomain 标志。需要 JMS 2.0 兼容消息代理。	false	布尔值
camel.component.amqp.synchronous	设置是否应严格使用同步处理。	false	布尔值
camel.component.amqp.task-executor	允许您指定自定义任务 executor 以供使用消息。选项是一个 org.springframework.core.task.TaskExecutor 类型。		TaskExecutor
camel.component.amqp.test-connection-on-startup	指定是否在启动时测试连接。这样可确保 Camel 启动所有 JMS 用户具有与 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 没有使用失败的连接启动。JMS 制作者也经过测试。	false	布尔值

Name	描述	默认值	类型
camel.component.amqp.time-to-live	在发送消息时，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 transacted 模式。	false	布尔值
camel.component.amqp.transacted-in-out	指定 InOut 操作（请求回复）是否默认使用 transacted 模式（如果此标志设为 true），则 Spring JmsTemplate 会将 sessionTransacted 设置为 true，以及 acknowledgeMode 作为 InOut 操作的 JmsTemplate 的 transacted。注意从 Spring JMS：在 JTA 事务中，传递给 createQueue 的参数，不会考虑 createTopic 方法。根据 Java EE 事务上下文，容器对这些值自行做出决定。类似地，这些参数不会被在本地管理的事务中考虑，因为本例中的 Spring JMS 在现有的 JMS Session 上运行。在受管事务之外运行时，将此标志设置为 true 将使用简短的本地 JMS 事务，并在存在受管事务（并非 XA 事务）的情况下同步的本地 JMS 事务。这将与主事务一起管理本地 JMS 事务（可能是原生 JDBC 事务），而 JMS 事务会在主事务后提交右边。	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。		PlatformTransactionManager
camel.component.amqp.transaction-name	要使用的事务的名称。		字符串
camel.component.amqp.transaction-timeout	使用 transacted 模式，事务的超时值（以秒为单位）。	-1	整数
camel.component.amqp.transfer-exception	如果启用了，并且您在使用 Request Reply messaging (InOut)，且 Exchange 在消费者端失败，则原因例外将作为 javax.jms.ObjectMessage 发回。如果客户端是 Camel，则返回的例外将被重新箭头。这样，您可以在路由中使用 Camel JMS 作为网桥 - 例如，使用持久性队列启用可靠的路由。请注意，如果您也启用了 transferExchange，这个选项将具有优先权。需要 Caught 异常才能按顺序排序。消费者端的原始 Exception 可以嵌套在外部异常中，如返回到制作者时 org.apache.camel.RuntimeCamelException。请小心谨慎，因为数据使用 Java 对象序列化，并且要求收到的能够在类级别上反序列化数据，这会强行生产者和消费者之间的强耦合。	false	布尔值

Name	描述	默认值	类型
camel.component.amqp.transfer-exchange	您可以通过线路传输交换，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 headers, exchange properties, exchange exception。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，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 知道有效负载是一个交换而不是常规有效负载。请小心谨慎，因为数据使用 Java 对象序列化，并且要求接收器能够在类级别上反序列化数据，这会强制生产者与需要使用兼容 Camel 版本的消费者之间强耦耦合！	false	布尔值
camel.component.amqp.use-message-id-as-correlation-id	指定 JMSMessageID 是否始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值
camel.component.amqp.username	用于 ConnectionFactory 的用户名。您也可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
camel.component.amqp.wait-for-provision-correlation-to-be-updated-counter	在通过 JMS 进行请求/回复时，等待 provisional correlation id 更新为实际关联 ID 的次数，以及启用选项 useMessageIDAsCorrelationID 的时间。	50	整数
camel.component.amqp.wait-for-provision-correlation-to-be-updated-thread-sleeping-time	等待更新调配关联 id 期间每次处于睡眠状态的间隔。选项是一个长类型。	100	Long

第 3 章 AVRO

此组件为 avro 提供 dataformat，允许使用 Apache Avro 的二进制 dataformat 来序列化和解序列化消息。由于 Camel 3.2 rpc 功能被移到单独的 **camel-avro-rpc** 组件中。

您可以使用 maven 和 ant 等模式轻松生成类。如需更多详细信息，请参阅 Apache Avro 文档。

3.1. 依赖项

当在 Camel Spring Boot 中使用 **camel-avro** 时，请添加以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-avro-starter</artifactId>
</dependency>
```

3.2. AVRO DATAFORMAT 选项

Avro dataformat 支持 1 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。

3.3. AVRO 数据格式使用

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

```
AvroDataFormat format = new AvroDataFormat(Value.SCHEMA$);

from("direct:in").marshal(format).to("direct:marshal");
from("direct:back").unmarshal(format).to("direct:unmarshal");
```

其中 Value 是 Avro Maven 插件生成类。

或在 XML 中

```
<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><custom ref="avro"/></marshal>
      <to uri="log:out"/>
    </route>
</camelContext>
```

同样，您可以使用 avro 数据格式 umarshal。

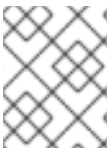
3.4. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 avro 时，请确保添加 Maven 依赖项来支持自动配置。组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.avro.enabled	是否启用 avro 数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.avro.instance-class-name	用于 marshal 和 unmarshalling 的类名称。		字符串

第 4 章 AVRO JACKSON

Jackson Avro 是一个数据格式，它使用带有 Avro 扩展的 Jackson 库，将 Avro 有效负载 unmarshalshalshalshal Java 对象到 Avro 有效负载。



注意

如果您熟悉 Jackson，则此 Avro 数据格式的行为与其 JSON 对应部分相同，因此可用于为 JSON 序列化/序列化/序列化注解的类。

```
from("kafka:topic").
  unmarshal().avro(AvroLibrary.Jackson, JsonNode.class).
  to("log:info");
```

4.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 **avro-jackson** 时，请确保添加 Maven 依赖项来支持自动配置。

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-jackson-avro-starter</artifactId>
</dependency>
```

4.2. 配置 SCHEMARESOLVER

由于 Avro serialization 基于模式，因此此数据格式需要您提供一个 SchemaResolver 对象，该对象能够为将要被处理/unmarshalled 的每个交换查找 schema。

您可以将单个 SchemaResolver 添加到 registry 中，它将被自动查找。或者，您可以明确指定对自定义 SchemaResolver 的引用。

4.3. AVRO JACKSON 选项

Avro Jackson dataformat 支持 18 个选项，如下所列。

Name	默认值	Java 类型	描述
objectMapper		**字符串**	使用 Jackson 时，查找并使用带有给定 id 的现有 ObjectMapper。
useDefaultObjectMapper		**布尔值**	是否从 registry 中查找和使用默认 Jackson ObjectMapper。
unmarshalType		**字符串**	当 unmarshalling 时使用的 java 类型的类名称。

Name	默认值	Java 类型	描述
jsonView		字符串	当 marshalling a POJO to JSON 时，您可能想要从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现此目的。此选项是引用具有 JsonView 注释的类。
Include		字符串	如果您想 marshal a pojo to JSON，并且 pojo 具有一些带有 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		布尔值	用于 JMS 用户，以允许 JMS spec 中的 JMSType 标头指定一个 FQN 类名称来用于 unmarshal。
collectionType		字符串	引用要使用的自定义集合类型，以便在 registry 中查找。这个选项应该很少被使用，但允许使用与基于 java.util.Collection 不同的集合类型。
useList		布尔值	To unmarshal 到 Map 列表或 Pojo 的列表。
moduleClassNames		字符串	使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String with FQN 类名称。可以使用逗号分隔多个类。
moduleRefs		字符串	使用 Camel registry 中引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。
enableFeatures		字符串	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上启用的功能集合。这个功能应该是与 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature multiple features 分开的名称。
disableFeatures		字符串	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的功能集合。这个功能应该是与 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature multiple features 分开的名称。
allowUnmarshallType		布尔值	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。这只有在需要使用时才启用。
timezone		字符串	如果设置，则 Jackson 会在 marshalling/unmarshalling 时使用 Timezone。
autoDiscoverObjectMapper		布尔值	如果设置为 true，则 Jackson 将把 objectMapper 来查找到 registry 中。

Name	默认值	Java 类型	描述
contentTypeHeader		布尔值	数据格式是否应该使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式到 XML 的 application/xml 或用于数据格式的 application/json 发送到 JSON。
schemaResolver		字符串	可选的模式解析器用于查找传输中数据的模式。
autoDiscoverSchemaResolver		布尔值	如果没有禁用，SchemaResolver 将查找到 registry 中。

4.4. 使用自定义 AVROMAPPER

如果需要更多地控制映射配置，您可以将 **JacksonAvroDataFormat** 配置为使用自定义 **AvroMapper**。

如果您在 registry 中设置单个 **AvroMapper**，则 Camel 将自动查找并使用此 **AvroMapper**。

4.5. 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.avro-jackson.allow-jms-type	用于 JMS 用户，以允许 JMS spec 中的 JMSType 标头指定一个 FQN 类名称来用于 unmarshal。	false	布尔值
camel.dataformat.avro-jackson.allow-unmarshall-type	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。这只有在需要使用时才启用。	false	布尔值
camel.dataformat.avro-jackson.auto-discover-object-mapper	如果设置为 true，则 Jackson 将把 objectMapper 来查找到 registry 中。	false	布尔值
camel.dataformat.avro-jackson.auto-discover-schema-resolver	如果没有禁用，SchemaResolver 将查找到 registry 中。	true	布尔值

Name	描述	默认值	类型
camel.dataformat.avro-jackson.collection-type	引用要使用的自定义集合类型，以便在 registry 中查找。这个选项应该很少被使用，但允许使用与基于 java.util.Collection 不同的集合类型。		字符串
camel.dataformat.avro-jackson.content-type-header	数据格式是否应该使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式到 XML 的 application/xml 或用于数据格式的 application/json 发送到 JSON。	true	布尔值
camel.dataformat.avro-jackson.disable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的功能集合。这个功能应该是与 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature multiple features 分开的名称。		字符串
camel.dataformat.avro-jackson.enable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上启用的功能集合。这个功能应该是与 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature multiple features 分开的名称。		字符串
camel.dataformat.avro-jackson.enabled	是否启用 avro-jackson 数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.avro-jackson.include	如果您想 marshal a pojo to JSON，并且 pojo 具有一些带有 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.avro-jackson.json-view	当 marshalling a POJO to JSON 时，您可能想要从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现此目的。此选项是引用具有 JsonView 注释的类。		字符串
camel.dataformat.avro-jackson.module-class-names	使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String with FQN 类名称。可以使用逗号分隔多个类。		字符串
camel.dataformat.avro-jackson.module-refs	使用 Camel registry 中引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。		字符串

Name	描述	默认值	类型
camel.dataformat.avro-jackson.object-mapper	使用 Jackson 时，查找并使用带有给定 id 的现有 ObjectMapper。		字符串
camel.dataformat.avro-jackson.schema-resolver	可选的模式解析器用于查找传输中数据的模式。		字符串
camel.dataformat.avro-jackson.timezone	如果设置，则 Jackson 会在 marshalling/unmarshalling 时使用 Timezone。		字符串
camel.dataformat.avro-jackson.unmarshal-type	当 unmarshalling 时使用的 java 类型的类名称。		字符串
camel.dataformat.avro-jackson.use-default-object-mapper	是否从 registry 中查找和使用默认 Jackson ObjectMapper。	true	布尔值
camel.dataformat.avro-jackson.use-list	To unmarshal 到 Map 列表或 Pojo 的列表。	false	布尔值

第 5 章 AWS CLOUDWATCH

仅支持生成者

AWS2 Cloudwatch 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。

5.1. 依赖项

当在 Camel Spring Boot 中使用 **aws2-cw** 时，请将以下 Maven 依赖项添加到 **pom.xml** 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-aws2-cw-starter</artifactId>
</dependency>
```

5.2. URI 格式

```
aws2-cw://namespace[?options]
```

如果指标不存在，则会创建它们。您可以将查询选项附加到 URI 中，格式为 **?options=value&option2=value&...**

5.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

5.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

5.3.2. 端点级别选项

在 **Endpoint 级别**，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 **路径和** 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

5.4. 组件选项

AWS CloudWatch 组件支持 18 个选项，如下所列。

Name	描述	默认值	类型
amazonCwClient （生成者）	**Autowired** 使用 AmazonCloudWatch 作为客户端。		CloudWatchClient
配置 （生成者）	组件配置。		Cw2Configuration
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 (producer)	指标名称。		字符串
overrideEndpoint (producer)	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
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
proxyProtocol (producer)	在实例化 CW 客户端时定义代理协议。 Enum 值： • HTTP • HTTPS	HTTPS	协议
region (producer)	CW 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串
timestamp (producer)	指标时间戳。		instant

Name	描述	默认值	类型
trustAllCertificates (producer)	如果要在覆盖端点时信任所有证书。	false	布尔值
unit (producer)	指标单元。		字符串
uriEndpointOverride (producer)	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
useDefaultCredentialsProvider (producer)	设置 S3 客户端是否应该希望通过默认凭据提供程序加载凭据，或者希望传递静态凭据。	false	布尔值
value (producer)	指标值。		å�Œ
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	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

5.5. 端点选项

AWS CloudWatch 端点使用 URI 语法进行配置：

```
aws2-cw:namespace
```

使用以下路径和查询参数：

5.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
namespace (producer)	**必需的** 指标命名空间。		字符串

5.5.2. 查询参数 (16 参数)

Name	描述	默认值	类型
amazonCwClient（生成者）	**Autowired** 使用 AmazonCloudWatch 作为客户端。		CloudWatchClient
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 (producer)	指标名称。		字符串
overrideEndpoint (producer)	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
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
proxyProtocol (producer)	在实例化 CW 客户端时定义代理协议。 Enum 值： • HTTP • HTTPS	HTTPS	协议
region (producer)	CW 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串
timestamp (producer)	指标时间戳。		instant
trustAllCertificates (producer)	如果要在覆盖端点时信任所有证书。	false	布尔值
unit (producer)	指标单元。		字符串
uriEndpointOverride (producer)	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
useDefaultCredentialsProvider (producer)	设置 S3 客户端是否应该希望通过默认凭据提供程序加载凭据，或者希望传递静态凭据。	false	布尔值

Name	描述	默认值	类型
value (producer)	指标值。		å�Œ
accessKey (security)	Amazon AWS 访问密钥.		字符串
secretKey (security)	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。

5.6. 使用方法

5.6.1. 静态凭证和默认凭证提供程序

您可以通过指定 useDefaultCredentialsProvider 选项并将其设置为 true 来避免使用显式静态凭证。

- Java 系统属性 - aws.accessKeyId 和 aws.secretKey
- 环境变量 - AWS_ACCESS_KEY_ID 和 AWS_SECRET_ACCESS_KEY。
- AWS STS 的 Web Identity Token。
- 共享凭证和配置文件。
- Amazon ECS 容器凭证 - 如果设置了环境变量 AWS_CONTAINER_CREDENTIALS_RELATIVE_URI，则从 Amazon ECS 加载。
- Amazon EC2 实例配置集凭据。

有关此信息的更多信息，您可以查看 AWS 凭证文档

5.6.2. 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。
CamelAwsCwMetricNamespace	**字符串**	Amazon CW 指标命名空间。

标头	类型	描述
CamelAwsCwMetricTimestamp	**Date**	Amazon CW 指标时间戳。
CamelAwsCwMetricDimensionName	**字符串**	Amazon CW 指标维度名称。
CamelAwsCwMetricDimensionValue	**字符串**	Amazon CW 指标维度值。
CamelAwsCwMetricDimensions	**Map<String, String>**	维度名称和维度值的映射.

5.6.3. 高级 CloudWatchClient 配置

如果您需要对 **CloudWatchClient** 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("aws2-cw://namespace?amazonCwClient=#client");
```

#client 指的是 Registry 中的一个 **CloudWatchClient**。

5.7. 例子

5.7.1. 生成者示例

```
from("direct:start")
  .to("aws2-cw://http://camel.apache.org/aws-cw");
```

发送类似内容

```
exchange.getIn().setHeader(Cw2Constants.METRIC_NAME, "ExchangesCompleted");
exchange.getIn().setHeader(Cw2Constants.METRIC_VALUE, "2.0");
exchange.getIn().setHeader(Cw2Constants.METRIC_UNIT, "Count");
```

5.8. 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.aws2-cw.access-key	Amazon AWS 访问密钥.		字符串

Name	描述	默认值	类型
camel.component.aws2-cw.amazon-cw-client	将 AmazonCloudWatch 用作客户端。选项是一个 software.amazon.awssdk.services.cloudwatch.CloudWatchClient 类型。		CloudWatchClient
camel.component.aws2-cw.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.aws2-cw.configuration	组件配置。选项是 org.apache.camel.component.aws2.cw.Cw2Configuration 类型。		Cw2Configuration
camel.component.aws2-cw.enabled	是否启用 aws2-cw 组件的自动配置。这默认是启用的。		布尔值
camel.component.aws2-cw.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.aws2-cw.name	指标名称。		字符串
camel.component.aws2-cw.override-endpoint	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
camel.component.aws2-cw.proxy-host	在实例化 CW 客户端时定义代理主机。		字符串
camel.component.aws2-cw.proxy-port	在实例化 CW 客户端时定义代理端口。		整数
camel.component.aws2-cw.proxy-protocol	在实例化 CW 客户端时定义代理协议。		协议

Name	描述	默认值	类型
camel.component.aws2-cw.region	CW 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串
camel.component.aws2-cw.secret-key	Amazon AWS Secret 密钥。		字符串
camel.component.aws2-cw.timestamp	指标时间戳。选项是一个 java.time.Instant 类型。		instant
camel.component.aws2-cw.trust-all-certificates	如果要在覆盖端点时信任所有证书。	false	布尔值
camel.component.aws2-cw.unit	指标单元。		字符串
camel.component.aws2-cw.uri-endpoint-override	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
camel.component.aws2-cw.use-default-credentials-provider	设置 S3 客户端是否应该希望通过默认凭据提供程序加载凭据，或者希望传递静态凭据。	false	布尔值
camel.component.aws2-cw.value	指标值。		å�Œ

第 6 章 AWS DYNAMODB

仅支持生成者

AWS2 DynamoDB 组件支持从/向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。

6.1. 依赖项

当使用 **aws2-ddb** 红帽构建的 Camel Spring Boot 时，请将以下 Maven 依赖项添加到 **pom.xml** 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-aws2-ddb-starter</artifactId>
</dependency>
```

6.2. URI 格式

```
aws2-ddb://domainName[?options]
```

您可以将查询选项附加到 URI 中，格式为 **?options=value&option2=value&...**

6.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

6.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

6.3.2. 端点级别选项

在 **Endpoint 级别**，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 **路径和 查询参数**。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

6.4. 组件选项

AWS DynamoDB 组件支持 22 个选项，如下所列。

Name	描述	默认值	类型
amazonDDBClient (producer)	**Autowired** 使用 AmazonDynamoDB 作为客户端。		DynamoDbClient
配置 （生成者）	组件配置。		Ddb2Configuration
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
enabledInitialDescribeTable (producer)	设置 DDB 端点中是否必须完成的初始 Describe 表操作，还是不完成。	true	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
keyScalarType (producer)	key scalar 类型，可以是 S （字符串）、N (数字) 和 B (字节)。		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

Name	描述	默认值	类型
operation (producer)	要执行的操作。 Enum 值： • BatchGetItems • DeleteItem • DeleteTable • DescribeTable • GetItem • PutItem • 查询 • 扫描 • UpdateItem • UpdateTable	PutItem	Ddb2Operations
overrideEndpoint (producer)	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
proxyHost (producer)	在实例化 DDB 客户端时定义代理主机。		字符串
proxyPort (producer)	DynamoDB 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		整数
proxyProtocol (producer)	在实例化 DDB 客户端时定义代理协议。 Enum 值： • HTTP • HTTPS	HTTPS	协议
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
trustAllCertificates (producer)	如果要在覆盖端点时信任所有证书。	false	布尔值

Name	描述	默认值	类型
uriEndpointOverride (producer)	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
useDefaultCredentialsProvider (producer)	设置 S3 客户端是否应该希望通过默认凭据提供程序加载凭据，或者希望传递静态凭据。	false	布尔值
writeCapacity (producer)	为向表写入资源而保留置备的吞吐量。		Long
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	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

6.5. 端点选项

AWS DynamoDB 端点使用 URI 语法进行配置：

```
aws2-ddb:tableName
```

使用以下路径和查询参数：

6.5.1. 路径参数(1 参数)

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

6.5.2. 查询参数 (20 参数)

Name	描述	默认值	类型
amazonDDBClient (producer)	**Autowired** 使用 AmazonDynamoDB 作为客户端。		DynamoDbClient

Name	描述	默认值	类型
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
enabledInitialDescribeTable (producer)	设置 DDB 端点中是否必须完成的初始 Describe 表操作，还是不完成。	true	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
keyScalarType (producer)	key scalar 类型，可以是 S （字符串）、N (数字) 和 B (字节)。		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
operation (producer)	要执行的操作。 Enum 值： • BatchGetItems • DeleteItem • DeleteTable • DescribeTable • GetItem • PutItem • 查询 • 扫描 • UpdateItem • UpdateTable	PutItem	Ddb2Operations
overrideEndpoint (producer)	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值

Name	描述	默认值	类型
proxyHost (producer)	在实例化 DDB 客户端时定义代理主机。		字符串
proxyPort (producer)	DynamoDB 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		整数
proxyProtocol (producer)	在实例化 DDB 客户端时定义代理协议。 Enum 值： • HTTP • HTTPS	HTTPS	协议
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
trustAllCertificates (producer)	如果要在覆盖端点时信任所有证书。	false	布尔值
uriEndpointOverride (producer)	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
useDefaultCredentialsProvider (producer)	设置 S3 客户端是否应该希望通过默认凭据提供程序加载凭据，或者希望传递静态凭据。	false	布尔值
writeCapacity (producer)	为向表写入资源而保留置备的吞吐量。		Long
accessKey (security)	Amazon AWS 访问密钥.		字符串
secretKey (security)	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。

6.6. 使用方法

6.6.1. 静态凭证和默认凭证提供程序

您可以通过指定 useDefaultCredentialsProvider 选项并将其设置为 true 来避免使用显式静态凭证。

- Java 系统属性 - aws.accessKeyId 和 aws.secretKey
- 环境变量 - AWS_ACCESS_KEY_ID 和 AWS_SECRET_ACCESS_KEY。
- AWS STS 的 Web Identity Token。
- 共享凭证和配置文件。
- Amazon ECS 容器凭证 - 如果设置了环境变量 AWS_CONTAINER_CREDENTIALS_RELATIVE_URI，则从 Amazon ECS 加载。
- Amazon EC2 实例配置集凭据。

有关此信息的更多信息，您可以查看 AWS 凭证文档

6.6.2. 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。
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。
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。

标头	类型	描述
CamelAwsDdbKeyConditions	**Map<String, Condition>**	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。
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。
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。

6.6.3. 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。

6.6.4. 在 DeleteItem 操作过程中设置消息标头

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

6.6.5. 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**
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

6.6.6. 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。
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

标头	类型	描述
CamelAwsDdbReadCapacity	**Long**	此表的 ReadCapacityUnits 属性。
CamelAwsDdbWriteCapacity	**Long**	此表的 WriteCapacityUnits 属性。

6.6.7. 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。

6.6.8. 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。

6.6.9. 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。

6.6.10. 扫描操作期间设置的消息标头

标头	类型	描述
CamelAwsDdbItems	**List<java.util.Map<String,AttributeValue>>**	操作返回的属性列表。

标头	类型	描述
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。

6.6.11. 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。

6.6.12. 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("aws2-ddb://domainName?amazonDDBClient=#client");
```

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

6.7. 支持的制作者操作

- BatchGetItems
- DeleteItem
- DeleteTable
- DescribeTable
- GetItem
- PutItem
- 查询
- 扫描
- UpdateItem

- UpdateTable

6.8. 例子

6.8.1. 生成者示例

- PutItem：此操作将在 DynamoDB 中创建一个条目

```
from("direct:start")
  .setHeader(Ddb2Constants.OPERATION, Ddb2Operations.PutItem)
  .setHeader(Ddb2Constants.CONSISTENT_READ, "true")
  .setHeader(Ddb2Constants.RETURN_VALUES, "ALL_OLD")
  .setHeader(Ddb2Constants.ITEM, attributeMap)
  .setHeader(Ddb2Constants.ATTRIBUTE_NAMES, attributeMap.keySet());
  .to("aws2-ddb://" + tableName + "?keyAttributeName=" + attributeName + "&keyAttributeType=" +
KeyType.HASH
  + "&keyScalarType=" + ScalarAttributeType.S
  + "&readCapacity=1&writeCapacity=1");
```

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-aws2-ddb</artifactId>
    <version>${camel-version}</version>
</dependency>
```

其中 **{camel-version}** 必须替换为 Camel 的实际版本。

6.9. SPRING BOOT AUTO-CONFIGURATION

组件支持 40 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws2-ddb.access-key	Amazon AWS 访问密钥.		字符串
camel.component.aws2-ddb.amazon-d-d-b-client	使用 AmazonDynamoDB 作为客户端。选项是一个 software.amazon.awssdk.services.dynamodb.DynamoDbClient 类型。		DynamoDbClient
camel.component.aws2-ddb.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.aws2-ddb.configuration	组件配置。选项是 org.apache.camel.component.aws2.ddb.Ddb2Configuration 类型。		Ddb2Configuration
camel.component.aws2-ddb.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.aws2-ddb.enabled	是否启用 aws2-ddb 组件的自动配置。这默认是启用的。		布尔值
camel.component.aws2-ddb.enabled-initial-describe-table	设置 DDB 端点中是否必须完成的初始 Describe 表操作，还是不完成。	true	布尔值
camel.component.aws2-ddb.key-attribute-name	创建表时的属性名称。		字符串
camel.component.aws2-ddb.key-attribute-type	创建表时的属性类型。		字符串
camel.component.aws2-ddb.key-scalar-type	key scalar 类型，可以是 S （字符串）、N (数字) 和 B (字节)。		字符串
camel.component.aws2-ddb.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.aws2-ddb.operation	要执行的操作。		Ddb2Operations
camel.component.aws2-ddb.override-endpoint	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值

Name	描述	默认值	类型
camel.component.aws2-ddb.proxy-host	在实例化 DDB 客户端时定义代理主机。		字符串
camel.component.aws2-ddb.proxy-port	DynamoDB 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		整数
camel.component.aws2-ddb.proxy-protocol	在实例化 DDB 客户端时定义代理协议。		协议
camel.component.aws2-ddb.read-capacity	要从您的表中读取资源的置备吞吐量。		Long
camel.component.aws2-ddb.region	DDB 客户端需要工作的区域。		字符串
camel.component.aws2-ddb.secret-key	Amazon AWS Secret 密钥。		字符串
camel.component.aws2-ddb.trust-all-certificates	如果要在覆盖端点时信任所有证书。	false	布尔值
camel.component.aws2-ddb.uri-endpoint-override	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
camel.component.aws2-ddb.use-default-credentials-provider	设置 S3 客户端是否应该希望通过默认凭据提供程序加载凭据，或者希望传递静态凭据。	false	布尔值
camel.component.aws2-ddb.write-capacity	为向表写入资源而保留置备的吞吐量。		Long
camel.component.aws2-ddbstream.access-key	Amazon AWS 访问密钥.		字符串

Name	描述	默认值	类型
camel.component.aws2-ddbstream.amazon-dynamo-db-streams-client	用于此端点的所有请求的 Amazon DynamoDB 客户端。选项是一个 software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsClient 类型。		DynamoDbStreamsClient
camel.component.aws2-ddbstream.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.aws2-ddbstream.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.aws2-ddbstream.configuration	组件配置。选项是 org.apache.camel.component.aws2.ddbstream.Ddb2StreamConfiguration 类型。		Ddb2StreamConfiguration
camel.component.aws2-ddbstream.enabled	是否启用 aws2-ddbstream 组件的自动配置。这默认是启用的。		布尔值
camel.component.aws2-ddbstream.max-results-per-request	每次轮询中将获取的最大记录数。		整数
camel.component.aws2-ddbstream.override-endpoint	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
camel.component.aws2-ddbstream.proxy-host	在实例化 DDBStreams 客户端时定义代理主机。		字符串

Name	描述	默认值	类型
camel.component.aws2-ddbstream.proxy-port	在实例化 DDBStreams 客户端时定义代理端口。		整数
camel.component.aws2-ddbstream.proxy-protocol	在实例化 DDBStreams 客户端时定义代理协议。		协议
camel.component.aws2-ddbstream.region	DDBStreams 客户端需要工作的区域。		字符串
camel.component.aws2-ddbstream.secret-key	Amazon AWS Secret 密钥。		字符串
camel.component.aws2-ddbstream.stream-iterator-type	定义 DynamoDB 流中开始获取记录的位置。请注意，使用 FROM_START 可能会导致流及时出现大量延迟。		Ddb2StreamConfiguration$StreamIteratorType
camel.component.aws2-ddbstream.trust-all-certificates	如果要在覆盖端点时信任所有证书。	false	布尔值
camel.component.aws2-ddbstream.uri-endpoint-override	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
camel.component.aws2-ddbstream.use-default-credentials-provider	设置 DynamoDB Streams 客户端是否应该预期通过默认凭据提供商加载凭据，或希望传递静态凭据。	false	布尔值

第 7 章 AWS KINESIS

支持生成者和消费者

AWS2 Kinesis 组件支持接收来自 Amazon Kinesis （不支持 Batch）服务的信息。

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。

7.1. 依赖项

当使用 **aws2-kinesis** 红帽构建的 Camel Spring Boot 时，请将以下 Maven 依赖项添加到 **pom.xml** 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-aws2-kinesis-starter</artifactId>
</dependency>
```

7.2. URI 格式

```
aws2-kinesis://stream-name[?options]
```

需要在使用前创建流。您可以将查询选项附加到 URI 中，格式为 **?options=value&option2=value&...**

7.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

7.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

7.3.2. 端点级别选项

在 **Endpoint 级别**，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 **路径和 查询参数**。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

7.4. 组件选项

AWS Kinesis 组件支持 22 个选项，如下所列。

Name	描述	默认值	类型
amazonKinesisClient (common)	**Autowired** Amazon Kinesis 客户端用于此端点的所有请求。		KinesisClient
cborEnabled (common)	此选项将在执行期间设置 CBOR_ENABLED 属性。	true	布尔值
configuration (common)	组件配置.		Kinesis2Configuration
overrideEndpoint (common)	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
proxyHost (common)	在实例化 Kinesis 客户端时定义代理主机。		字符串
proxyPort (common)	在实例化 Kinesis 客户端时定义代理端口。		整数
proxyProtocol (common)	在实例化 Kinesis 客户端时定义代理协议。 Enum 值： • HTTP • HTTPS	HTTPS	协议
region (common)	Kinesis Firehose 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串
trustAllCertificates (common)	如果要在覆盖端点时信任所有证书。	false	布尔值
uriEndpointOverride (common)	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
useDefaultCredentialsProvider (common)	设置 Kinesis 客户端是否应该希望通过默认凭证提供程序加载凭证，或者希望传递静态凭证。	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
iteratorType (consumer)	定义在 Kinesis 流中开始获取记录的位置。 Enum 值： • AT_SEQUENCE_NUMBER • AFTER_SEQUENCE_NUMBER • TRIM_HORIZON • LATEST • AT_TIMESTAMP • null	TRIM_HORIZON	ShardIteratorType
maxResultsPerRequest (consumer)	每次轮询中将获取的最大记录数。	1	int
resumeStrategy (consumer)	为 AWS Kinesis 定义恢复策略。如果提供，默认策略将读取 sequenceNumber。	KinesisUserConfigurationResumeStrategy	KinesisResumeStrategy
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

Name	描述	默认值	类型
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。如果忽略了消息，并且消费者将从开始重新启动，如果为 silent，则消费者将从开始记录。如果开始，消费者将引发故障关闭状态异常。 Enum 值： • ignore • fail • silent	ignore	Kinesis2ShardClosedStrategyEnum
shardId (consumer)	定义 Kinesis 流中要从哪些分片 ID 获取记录。		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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	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

7.5. 端点选项

AWS Kinesis 端点使用 URI 语法进行配置：

```
aws2-kinesis:streamName
```

使用以下路径和查询参数：

7.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
streamName (common)	流**必需的**名称。		字符串

7.5.2. 查询参数 (38 参数)

Name	描述	默认值	类型
amazonKinesisClient (common)	**Autowired** Amazon Kinesis 客户端用于此端点的所有请求。		KinesisClient
cborEnabled (common)	此选项将在执行期间设置 CBOR_ENABLED 属性。	true	布尔值
overrideEndpoint (common)	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
proxyHost (common)	在实例化 Kinesis 客户端时定义代理主机。		字符串
proxyPort (common)	在实例化 Kinesis 客户端时定义代理端口。		整数
proxyProtocol (common)	在实例化 Kinesis 客户端时定义代理协议。 Enum 值： • HTTP • HTTPS	HTTPS	协议
region (common)	Kinesis Firehose 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串
trustAllCertificates (common)	如果要在覆盖端点时信任所有证书。	false	布尔值
uriEndpointOverride (common)	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
useDefaultCredentialsProvider (common)	设置 Kinesis 客户端是否应该希望通过默认凭证提供程序加载凭证，或者希望传递静态凭证。	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
iteratorType (consumer)	定义在 Kinesis 流中开始获取记录的位置。 Enum 值： • AT_SEQUENCE_NUMBER • AFTER_SEQUENCE_NUMBER • TRIM_HORIZON • LATEST • AT_TIMESTAMP • null	TRIM_HORIZON	ShardIteratorType
maxResultsPerRequest (consumer)	每次轮询中将获取的最大记录数。	1	int
resumeStrategy (consumer)	为 AWS Kinesis 定义恢复策略。如果提供，默认策略将读取 sequenceNumber。	KinesisUserConfigurationResumeStrategy	KinesisResumeStrategy
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

Name	描述	默认值	类型
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。如果忽略了消息，并且消费者将从开始重新启动，如果为 silent，则消费者将从开始记录。如果开始，消费者将引发故障关闭状态异常。 Enum 值： • ignore • fail • silent	ignore	Kinesis2ShardClosedStrategyEnum
shardId (consumer)	定义 Kinesis 流中要从哪些分片 ID 获取记录。		字符串
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
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
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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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
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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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

所需的 Kinesis 组件选项

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

7.6. 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。

7.7. 使用方法

7.7.1. 静态凭证和默认凭证提供程序

您可以通过指定 useDefaultCredentialsProvider 选项并将其设置为 true 来避免使用显式静态凭证。

- Java 系统属性 - aws.accessKeyId 和 aws.secretKey
- 环境变量 - AWS_ACCESS_KEY_ID 和 AWS_SECRET_ACCESS_KEY。
- AWS STS 的 Web Identity Token。
- 共享凭证和配置文件。

- Amazon ECS 容器凭证 - 如果设置了环境变量 AWS_CONTAINER_CREDENTIALS_RELATIVE_URI，则从 Amazon ECS 加载。
- Amazon EC2 实例配置集凭据。

有关此信息的更多信息，您可以查看 AWS 凭证文档

7.7.2. 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
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。

7.7.3. AmazonKinesis 配置

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

```
from("aws2-kinesis://mykinesisstream?amazonKinesisClient=#kinesisClient")
  .to("log:out?showAll=true");
```

7.7.4. 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。

7.7.5. 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。

7.7.6. 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

7.8. 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-aws2-kinesis</artifactId>
    <version>${camel-version}</version>
</dependency>
```

其中 **{camel-version}** 必须替换为 Camel 的实际版本。

7.9. SPRING BOOT AUTO-CONFIGURATION

组件支持 40 个选项，如下所列。

Name	描述	默认值	类型
camel.component.aws2-kinesis-firehose.access-key	Amazon AWS 访问密钥.		字符串
camel.component.aws2-kinesis-firehose.amazon-kinesis-firehose-client	Amazon Kinesis Firehose 客户端用于此端点的所有请求。选项是一个 software.amazon.awssdk.services.firehose.FirehoseClient 类型。		FirehoseClient
camel.component.aws2-kinesis-firehose.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.aws2-kinesis-firehose.cbor-enabled	此选项将在执行期间设置 CBOR_ENABLED 属性。	true	布尔值

Name	描述	默认值	类型
camel.component.aws2-kinesis-firehose.configuration	组件配置.选项是 org.apache.camel.component.aws2.firehose.KinesisFirehose2Configuration 类型。		KinesisFirehose2Configuration
camel.component.aws2-kinesis-firehose.enabled	是否启用 aws2-kinesis-firehose 组件的自动配置。这默认是启用的。		布尔值
camel.component.aws2-kinesis-firehose.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.aws2-kinesis-firehose.operation	当用户不希望只发送记录时，要执行的操作。		KinesisFirehose2Operations
camel.component.aws2-kinesis-firehose.override-endpoint	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
camel.component.aws2-kinesis-firehose.proxy-host	在实例化 Kinesis Firehose 客户端时定义代理主机。		字符串
camel.component.aws2-kinesis-firehose.proxy-port	在实例化 Kinesis Firehose 客户端时定义代理端口。		整数
camel.component.aws2-kinesis-firehose.proxy-protocol	在实例化 Kinesis Firehose 客户端时定义代理协议。		协议
camel.component.aws2-kinesis-firehose.region	Kinesis Firehose 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串

Name	描述	默认值	类型
camel.component.aws2-kinesis-firehose.secret-key	Amazon AWS Secret 密钥。		字符串
camel.component.aws2-kinesis-firehose.trust-all-certificates	如果要在覆盖端点时信任所有证书。	false	布尔值
camel.component.aws2-kinesis-firehose.uri-endpoint-override	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
camel.component.aws2-kinesis-firehose.use-default-credentials-provider	设置 Kinesis Firehose 客户端是否应该通过默认凭据提供商加载凭据，还是希望传递静态凭据。	false	布尔值
camel.component.aws2-kinesis.access-key	Amazon AWS 访问密钥.		字符串
camel.component.aws2-kinesis.amazon-kinesis-client	Amazon Kinesis 客户端用于此端点的所有请求。选项是一个 software.amazon.awssdk.services.kinesis.KinesisClient 类型。		KinesisClient
camel.component.aws2-kinesis.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.aws2-kinesis.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

Name	描述	默认值	类型
camel.component.aws2-kinesis.cbor-enabled	此选项将在执行期间设置 CBOR_ENABLED 属性。	true	布尔值
camel.component.aws2-kinesis.configuration	组件配置.选项是 org.apache.camel.component.aws2.kinesis.Kinesis2Configuration 类型。		Kinesis2Configuration
camel.component.aws2-kinesis.enabled	是否启用 aws2-kinesis 组件的自动配置。这默认是启用的。		布尔值
camel.component.aws2-kinesis.iterator-type	定义在 Kinesis 流中开始获取记录的位置。		ShardIteratorType
camel.component.aws2-kinesis.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.aws2-kinesis.max-results-per-request	每次轮询中将获取的最大记录数。	1	整数
camel.component.aws2-kinesis.override-endpoint	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
camel.component.aws2-kinesis.proxy-host	在实例化 Kinesis 客户端时定义代理主机。		字符串
camel.component.aws2-kinesis.proxy-port	在实例化 Kinesis 客户端时定义代理端口。		整数

Name	描述	默认值	类型
camel.component.aws2-kinesis.proxy-protocol	在实例化 Kinesis 客户端时定义代理协议。		协议
camel.component.aws2-kinesis.region	Kinesis Firehose 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串
camel.component.aws2-kinesis.resume-strategy	为 AWS Kinesis 定义恢复策略。如果提供，默认策略将读取 sequenceNumber。选项是一个 org.apache.camel.component.aws2.kinesis.consumer.KinesisResumeStrategy 类型。		KinesisResumeStrategy
camel.component.aws2-kinesis.secret-key	Amazon AWS Secret 密钥。		字符串
camel.component.aws2-kinesis.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.aws2-kinesis.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。如果忽略了消息，并且消费者将从开始重新启动，如果为 silent，则消费者将从开始记录。如果开始，消费者将引发故障关闭状态异常。		Kinesis2ShardClosedStrategyEnum
camel.component.aws2-kinesis.shard-id	定义 Kinesis 流中要从哪些分片 ID 获取记录。		字符串
camel.component.aws2-kinesis.trust-all-certificates	如果要在覆盖端点时信任所有证书。	false	布尔值
camel.component.aws2-kinesis.uri-endpoint-override	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
camel.component.aws2-kinesis.use-default-credentials-provider	设置 Kinesis 客户端是否应该希望通过默认凭证提供程序加载凭证，或者希望传递静态凭证。	false	布尔值

第 8 章 AWS 2 LAMBDA

仅支持生成者

AWS2 Lambda 组件支持 create, get, list, delete 和 invoke AWS Lambda 函数。

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 AWS 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。

8.1. 依赖项

当在 Camel Spring Boot 中使用 **aws2-lambda** 时，请将以下 Maven 依赖项添加到 **pom.xml** 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-aws2-lambda-starter</artifactId>
</dependency>
```

8.2. URI 格式

```
aws2-lambda://functionName[?options]
```

您可以将查询选项附加到 URI 中，格式为 **options=value&option2=value&...**

8.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

8.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

8.3.2. 端点级别选项

在 **Endpoint 级别**，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 **路径和 查询参数**。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

8.4. 组件选项

AWS Lambda 组件支持 16 个选项，如下所列。

Name	描述	默认值	类型
配置 （生成者）	组件配置.		Lambda2Configuration
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
operation (producer)	要执行的操作。它可以是 listFunctions、getFunction、createFunction、deleteFunction 或 invokeFunction。 Enum 值： • listFunctions • getFunction • createAlias • deleteAlias • getAlias • listAliases • createFunction • deleteFunction • invokeFunction • updateFunction • createEventSourceMapping • deleteEventSourceMapping • listEventSourceMapping • listTags • tagResource • untagResource • publishVersion • listVersions	invokeFunction	Lambda2Operations
overrideEndpoint (producer)	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
pojoRequest (producer)	如果您想要将 POJO 请求用作正文。	false	布尔值
region (producer)	Lambda 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串
trustAllCertificates (producer)	如果要在覆盖端点时信任所有证书。	false	布尔值

Name	描述	默认值	类型
uriEndpointOverride (producer)	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
useDefaultCredentialsProvider (producer)	设置 Lambda 客户端是否应该预期通过默认凭据提供商加载凭证，或者希望传递静态凭证。	false	布尔值
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
awsLambdaClient (advanced)	**Autowired** 使用现有配置的 AwsLambdaClient 作为客户端。		LambdaClient
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
proxyProtocol (proxy)	在实例化 Lambda 客户端时定义代理协议。 Enum 值： • HTTP • HTTPS	HTTPS	协议
accessKey (security)	Amazon AWS 访问密钥.		字符串
secretKey (security)	Amazon AWS Secret 密钥。		字符串

8.5. 端点选项

AWS Lambda 端点使用 URI 语法进行配置：

```
aws2-lambda:function
```

使用以下路径和查询参数：

8.5.1. 路径参数(1 参数)

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

8.5.2. 查询参数 (14 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

Name	描述	默认值	类型
operation (producer)	要执行的操作。它可以是 listFunctions、getFunction、createFunction、deleteFunction 或 invokeFunction。 Enum 值： • listFunctions • getFunction • createAlias • deleteAlias • getAlias • listAliases • createFunction • deleteFunction • invokeFunction • updateFunction • createEventSourceMapping • deleteEventSourceMapping • listEventSourceMapping • listTags • tagResource • untagResource • publishVersion • listVersions	invokeFunction	Lambda2Operations
overrideEndpoint (producer)	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
pojoRequest (producer)	如果您想要将 POJO 请求用作正文。	false	布尔值
region (producer)	Lambda 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串
trustAllCertificates (producer)	如果要在覆盖端点时信任所有证书。	false	布尔值

Name	描述	默认值	类型
uriEndpointOverride (producer)	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
useDefaultCredentialsProvider (producer)	设置 Lambda 客户端是否应该预期通过默认凭据提供商加载凭证，或者希望传递静态凭证。	false	布尔值
awsLambdaClient (advanced)	**Autowired** 使用现有配置的 AwsLambdaClient 作为客户端。		LambdaClient
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
proxyProtocol (proxy)	在实例化 Lambda 客户端时定义代理协议。 Enum 值： • HTTP • HTTPS	HTTPS	协议
accessKey (security)	Amazon AWS 访问密钥.		字符串
secretKey (security)	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，以访问 Amazon Lambda 服务。

8.6. 使用方法

8.6.1. 静态凭证和默认凭证提供程序

您可以通过指定 useDefaultCredentialsProvider 选项并将其设置为 true 来避免使用显式静态凭证。

- Java 系统属性 - aws.accessKeyId 和 aws.secretKey
- 环境变量 - AWS_ACCESS_KEY_ID 和 AWS_SECRET_ACCESS_KEY。
- AWS STS 的 Web Identity Token。
- 共享凭证和配置文件。
- Amazon ECS 容器凭证 - 如果设置了环境变量 AWS_CONTAINER_CREDENTIALS_RELATIVE_URI，则从 Amazon ECS 加载。

- Amazon EC2 实例配置集凭据。

有关此信息的更多信息，您可以查看 AWS 凭证文档

8.6.2. 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
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
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
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
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
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
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
createFunction	**CamelAwsLambdaEnvironmentVariables**	**Map<String, String>**	代表您的环境配置设置的键值对。	否
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
createAlias	**CamelAwsLambdaFunctionVersion**	**字符串**	在别名中设置的功能版本	是
createAlias	**CamelAwsLambdaAliasFunctionName**	**字符串**	在别名中设置的函数名称	是
createAlias	**CamelAwsLambdaAliasFunctionDescription**	**字符串**	在别名中设置的函数描述	否
deleteAlias	**CamelAwsLambdaAliasFunctionName**	**字符串**	别名的功能名称	是
getAlias	**CamelAwsLambdaAliasFunctionName**	**字符串**	别名的功能名称	是
listAliases	**CamelAwsLambdaFunctionVersion**	**字符串**	在别名中设置的功能版本	否

8.7. 可运行的操作列表

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

- tagResource
- untagResource
- publishVersion
- listVersions
- createAlias
- deleteAlias
- getAlias
- listAliases

8.8. 例子

8.8.1. 生成者示例

要完全了解组件的工作方式，您可以参阅这些 集成测试。

8.8.2. 生成者示例

- CreateFunction：此操作将在 AWS Lambda 中为您创建一个功能

```
  from("direct:createFunction").to("aws2-lambda://GetHelloWithName?
operation=createFunction").to("mock:result");
```

并通过发送

```
template.send("direct:createFunction", ExchangePattern.InOut, new Processor() {
    @Override
    public void process(Exchange exchange) throws Exception {
        exchange.getIn().setHeader(Lambda2Constants.RUNTIME, "nodejs6.10");
        exchange.getIn().setHeader(Lambda2Constants.HANDLER, "GetHelloWithName.handler");
        exchange.getIn().setHeader(Lambda2Constants.DESCRIPTION, "Hello with node.js on
Lambda");
        exchange.getIn().setHeader(Lambda2Constants.ROLE,
            "arn:aws:iam::643534317684:role/lambda-execution-role");
        ClassLoader classLoader = getClass().getClassLoader();
        File file = new File(
            classLoader

.getResource("org/apache/camel/component/aws2/lambda/function/node/GetHelloWithName.zip")
                .getFile());
        FileInputStream inputStream = new FileInputStream(file);
        exchange.getIn().setBody(inputStream);
    }
});
```

8.9. 使用 POJO 作为正文

由于多个选项，有时构建 AWS Request 可能会很复杂。我们介绍可能将 POJO 用作正文。在 AWS Lambda 中，您可以提交多个操作，如 Get Function 请求，您可以执行以下操作：

```
from("direct:getFunction")
    .setBody(GetFunctionRequest.builder().functionName("test").build())
    .to("aws2-lambda://GetHelloWithName?
awsLambdaClient=#awsLambdaClient&operation=getFunction&pojoRequest=true")
```

这样，您将直接传递请求，而无需专门传递与此操作相关的标头和选项。

8.10. 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-aws2-lambda</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 **{camel-version}** 必须替换为 Camel 的实际版本。

8.11. 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.aws2-lambda.access-key	Amazon AWS 访问密钥.		字符串
camel.component.aws2-lambda.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.aws2-lambda.aws-lambda-client	使用现有配置的 AwsLambdaClient 作为客户端。选项是一个 software.amazon.awssdk.services.lambda.LambdaClient 类型。		LambdaClient
camel.component.aws2-lambda.configuration	组件配置.选项是 org.apache.camel.component.aws2.lambda.Lambda2Configuration 类型。		Lambda2Configuration

Name	描述	默认值	类型
camel.component.aws2-lambda.enabled	是否启用 aws2-lambda 组件的自动配置。这默认是启用的。		布尔值
camel.component.aws2-lambda.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.aws2-lambda.operation	要执行的操作。它可以是 listFunctions、getFunction、createFunction、deleteFunction 或 invokeFunction。		Lambda2Operations
camel.component.aws2-lambda.override-endpoint	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
camel.component.aws2-lambda.pojo-request	如果您想要将 POJO 请求用作正文。	false	布尔值
camel.component.aws2-lambda.proxy-host	在实例化 Lambda 客户端时定义代理主机。		字符串
camel.component.aws2-lambda.proxy-port	在实例化 Lambda 客户端时定义代理端口。		整数
camel.component.aws2-lambda.proxy-protocol	在实例化 Lambda 客户端时定义代理协议。		协议
camel.component.aws2-lambda.region	Lambda 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串

Name	描述	默认值	类型
camel.component.aws2-lambda.secret-key	Amazon AWS Secret 密钥。		字符串
camel.component.aws2-lambda.trust-all-certificates	如果要在覆盖端点时信任所有证书。	false	布尔值
camel.component.aws2-lambda.uri-endpoint-override	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
camel.component.aws2-lambda.use-default-credentials-provider	设置 Lambda 客户端是否应该预期通过默认凭据提供商加载凭证，或者希望传递静态凭证。	false	布尔值

第 9 章 AWS S3 STORAGE SERVICE

支持生成者和消费者

AWS2 S3 组件支持从/到 Amazon 的 S3 服务存储和检索对象。

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，请访问 link:https://aws.amazon.com/s3 [Amazon S3]。

9.1. 依赖项

当在 Camel Spring Boot 中使用 **aws2-s3** 时，请将以下 Maven 依赖项添加到 **pom.xml** 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-aws2-s3-starter</artifactId>
</dependency>
```

9.2. URI 格式

```
aws2-s3://bucketNameOrArn[?options]
```

如果存储桶不存在，则会创建存储桶。您可以以以下格式将查询选项附加到 URI 中，

options=value&option2=value&...

9.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

9.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

9.3.2. 端点级别选项

在 **Endpoint 级别**，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 **路径和 查询参数**。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

9.4. 组件选项

AWS S3 Storage Service 组件支持 50 个选项，如下所列。

Name	描述	默认值	类型
amazonS3Client (common)	对 registry 中的 com.amazonaws.services.s3.AmazonS3 的 **Autowired** Reference。		S3Client
amazonS3Presigner (common)	**Autowired** 一个用于请求的 S3 Presigner，主要在 createDownloadLink 操作中使用。		S3Presigner
autoCreateBucket (common)	设置 S3 存储桶自动创建的 bucketName。如果启用了 moveAfterRead 选项，则也会应用它，如果尚未存在 moveAfterRead 选项，它将创建 destinationBucket。	false	布尔值
configuration (common)	组件配置。		AWS2S3Configuration
overrideEndpoint (common)	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
pojoRequest (common)	如果您想要将 POJO 请求用作正文。	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
proxyProtocol (common)	在实例化 S3 客户端时定义代理协议。 Enum 值： • HTTP • HTTPS	HTTPS	协议

Name	描述	默认值	类型
region (common)	S3 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串
trustAllCertificates (common)	如果要在覆盖端点时信任所有证书。	false	布尔值
uriEndpointOverride (common)	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
useDefaultCredentialsProvider (common)	设置 S3 客户端是否应该希望通过默认凭据提供程序加载凭据，或者希望传递静态凭据。	false	布尔值
customerAlgorithm (common (advanced))	定义在启用了 CustomerKey 时要使用的客户算法。		字符串
customerKeyId (common (advanced))	定义在启用 CustomerKey 时要使用的 Customer key 的 id。		字符串
customerKeyMD5 (common (advanced))	定义在启用 CustomerKey 时要使用的客户密钥的 MD5。		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
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 AWS2S3Constants#BUCKET_NAME 和 AWS2S3Constants#KEY 标头过滤，或者只过滤 AWS2S3Constants#KEY 标头。	true	布尔值
delimiter (consumer)	com.amazonaws.services.s3.model.ListObjectsRequest 中使用的分隔符仅消耗我们感兴趣的对象。		字符串
destinationBucket (consumer)	定义当 moveAfterRead 设置为 true 时必须移动对象的目标存储桶。		字符串

Name	描述	默认值	类型
destinationBucketPrefix (consumer)	定义在必须移动对象并将 moveAfterRead 设置为 true 时使用的目标存储桶前缀。		字符串
destinationBucketSuffix (consumer)	定义在必须移动对象并将 moveAfterRead 设置为 true 时使用的目标存储桶后缀。		字符串
doneFileName (consumer)	如果提供，Camel 仅在文件存在时使用文件。		字符串
fileName (consumer)	要从具有给定文件名的存储桶获取对象。		字符串
ignoreBody (consumer)	如果为 true，则 S3 对象正文将完全忽略，如果设为 false，则 S3 对象将放入正文中。把它设置为 true，将覆盖 includeBody 选项定义的任何行为。	false	布尔值
includeBody (consumer)	如果为 true，则 S3Object Exchange 将被使用并放入正文和关闭中。如果为 false，S3Object 流将原始放在正文中，标头将使用 S3 对象元数据设置。这个选项与 autocloseBody 选项密切相关。如果将 includeBody 设为 true，因为 S3Object 流将被消耗，然后也会关闭它，而在 includeBody false 时，它将是关闭 S3Object 流的调用者。但是，当 includeBody 为 false 时，将 autocloseBody 设置为 true，它将在交换完成时自动关闭 S3Object 流。	true	布尔值
includeFolders (consumer)	如果为 true，将消费的文件夹/目录。如果是 false，则忽略它们，且不会为那些交换创建。	true	布尔值
moveAfterRead (consumer)	在检索后，将对象从 S3 存储桶移到不同的存储桶。要完成操作，必须设置 destinationBucket 选项。仅当 Exchange 提交时，才会执行复制存储桶操作。如果进行回滚，则对象不会被移动。	false	布尔值
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
autocloseBody (consumer (advanced))	如果此选项为 true，且 includeBody 为 false，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 false，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
batchMessageNumber (producer)	在流传输上传模式中制作批处理的消息数量。	10	int

Name	描述	默认值	类型
batchSize (producer)	流上传模式的批处理大小（以字节为单位）。	1000000	int
deleteAfterWrite (producer)	在 S3 文件上传后删除文件对象。	false	布尔值
KeyName (producer)	通过端点参数在存储桶中设置元素的密钥名称。		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
multiPartUpload (producer)	如果为 true，则 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
namingStrategy (producer)	在流上传模式中使用的命名策略。 Enum 值： • progressive • random	progressive	AWSS3NamingStrategyEnum
operation (producer)	当用户不希望只进行上传时，要执行的操作。 Enum 值： • copyObject • listObjects • deleteObject • deleteBucket • listBuckets • getObject • getObjectRange • createDownloadLink		AWS2S3Operations
partSize (producer)	设置多部分上传中使用的 partSize，默认大小为 25M。	26214400	long

Name	描述	默认值	类型
restartingPolicy (producer)	在流上传模式中使用的重启策略。 Enum 值： • override • lastPart	override	AWSS3RestartingPolicyEnum
storageClass (producer)	在 com.amazonaws.services.s3.model.PutObjectRequest 请求中设置的存储类。		字符串
streamingUploadMode (producer)	当流模式为 true 时，上传到存储桶将以流传输方式进行。	false	布尔值
streamingUploadTimeout (producer)	在流上传模式为 true 时，此选项会将超时设置为完成上传。		long
awsKMSKeyId (producer (advanced))	定义在启用 KMS 时要使用的 KMS 密钥 ID。		字符串
useAwsKMS (producer (advanced))	定义是否必须使用 KMS。	false	布尔值
useCustomerKey (producer (advanced))	定义是否需要使用客户密钥。	false	布尔值
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	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

9.5. 端点选项

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

```
aws2-s3://bucketNameOrArn
```

使用以下路径和查询参数：

9.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
bucketNameOrArn (common)	**所需的** Bucket 名称或 ARN。		字符串

9.5.2. 查询参数 (68 参数)

Name	描述	默认值	类型
amazonS3Client (common)	对 registry 中的 com.amazonaws.services.s3.AmazonS3 的 **Autowired** Reference。		S3Client
amazonS3Presigner (common)	**Autowired** 一个用于请求的 S3 Presigner，主要在 createDownloadLink 操作中使用。		S3Presigner
autoCreateBucket (common)	设置 S3 存储桶自动创建的 bucketName。如果启用了 moveAfterRead 选项，则也会应用它，如果尚未存在 moveAfterRead 选项，它将创建 destinationBucket。	false	布尔值
overrideEndpoint (common)	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
pojoRequest (common)	如果您想要将 POJO 请求用作正文。	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
proxyProtocol (common)	在实例化 S3 客户端时定义代理协议。 Enum 值： • HTTP • HTTPS	HTTPS	协议

Name	描述	默认值	类型
region (common)	S3 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串
trustAllCertificates (common)	如果要在覆盖端点时信任所有证书。	false	布尔值
uriEndpointOverride (common)	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
useDefaultCredentialsProvider (common)	设置 S3 客户端是否应该希望通过默认凭据提供程序加载凭据，或者希望传递静态凭据。	false	布尔值
customerAlgorithm (common (advanced))	定义在启用了 CustomerKey 时要使用的客户算法。		字符串
customerKeyId (common (advanced))	定义在启用 CustomerKey 时要使用的 Customer key 的 id。		字符串
customerKeyMD5 (common (advanced))	定义在启用 CustomerKey 时要使用的客户密钥的 MD5。		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
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 AWS2S3Constants#BUCKET_NAME 和 AWS2S3Constants#KEY 标头过滤，或者只过滤 AWS2S3Constants#KEY 标头。	true	布尔值
delimiter (consumer)	com.amazonaws.services.s3.model.ListObjectsRequest 中使用的分隔符仅消耗我们感兴趣的对象。		字符串

Name	描述	默认值	类型
destinationBucket (consumer)	定义当 moveAfterRead 设置为 true 时必须移动对象的目标存储桶。		字符串
destinationBucketPrefix (consumer)	定义在必须移动对象并将 moveAfterRead 设置为 true 时使用的目标存储桶前缀。		字符串
destinationBucketSuffix (consumer)	定义在必须移动对象并将 moveAfterRead 设置为 true 时使用的目标存储桶后缀。		字符串
doneFileName (consumer)	如果提供，Camel 仅在文件存在时使用文件。		字符串
fileName (consumer)	要从具有给定文件名的存储桶获取对象。		字符串
ignoreBody (consumer)	如果为 true，则 S3 对象正文将完全忽略，如果设为 false，则 S3 对象将放入正文中。把它设置为 true，将覆盖 includeBody 选项定义的任何行为。	false	布尔值
includeBody (consumer)	如果为 true，则 S3Object Exchange 将被使用并放入正文和关闭中。如果为 false，S3Object 流将原始放在正文中，标头将使用 S3 对象元数据设置。这个选项与 autocloseBody 选项密切相关。如果将 includeBody 设为 true，因为 S3Object 流将被消耗，然后也会关闭它，而在 includeBody false 时，它将是关闭 S3Object 流的调用者。但是，当 includeBody 为 false 时，将 autocloseBody 设置为 true，它将在交换完成时自动关闭 S3Object 流。	true	布尔值
includeFolders (consumer)	如果为 true，将消费的文件夹/目录。如果是 false，则忽略它们，且不会为那些交换创建。	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。获取最大消息数，作为每次轮询轮询的限制。默认值为 10。使用 0 或负数设置为无限。	10	int
moveAfterRead (consumer)	在检索后，将对象从 S3 存储桶移到不同的存储桶。要完成操作，必须设置 destinationBucket 选项。仅当 Exchange 提交时，才会执行复制存储桶操作。如果进行回滚，则对象不会被移动。	false	布尔值

Name	描述	默认值	类型
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 (advanced))	如果此选项为 true，且 includeBody 为 false，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 false，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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
batchMessageNumber (producer)	在流传输上传模式中制作批处理的消息数量。	10	int
batchSize (producer)	流上传模式的批处理大小（以字节为单位）。	1000000	int
deleteAfterWrite (producer)	在 S3 文件上传后删除文件对象。	false	布尔值
KeyName (producer)	通过端点参数在存储桶中设置元素的密钥名称。		字符串

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
multiPartUpload (producer)	如果为 true，则 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
namingStrategy (producer)	在流上传模式中使用的命名策略。 Enum 值： • progressive • random	progressive	AWSS3NamingStrategyEnum
operation (producer)	当用户不希望只进行上传时，要执行的操作。 Enum 值： • copyObject • listObjects • deleteObject • deleteBucket • listBuckets • getObject • getObjectRange • createDownloadLink		AWS2S3Operations
partSize (producer)	设置多部分上传中使用的 partSize，默认大小为 25M。	26214400	long
restartingPolicy (producer)	在流上传模式中使用的重启策略。 Enum 值： • override • lastPart	override	AWSS3RestartingPolicyEnum

Name	描述	默认值	类型
storageClass (producer)	在 com.amazonaws.services.s3.model.PutObjectRequest 请求中设置的存储类。		字符串
streamingUploadMode (producer)	当流模式为 true 时，上传到存储桶将以流传输方式进行。	false	布尔值
streamingUploadTimeout (producer)	在流上传模式为 true 时，此选项会将超时设置为完成上传。		long
awsKMSKeyId (producer (advanced))	定义在启用 KMS 时要使用的 KMS 密钥 ID。		字符串
useAwsKMS (producer (advanced))	定义是否必须使用 KMS。	false	布尔值
useCustomerKey (producer (advanced))	定义是否需要使用客户密钥。	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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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

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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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

Name	描述	默认值	类型
accessKey (security)	Amazon AWS 访问密钥.		字符串
secretKey (security)	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。

9.6. 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。

9.7. 使用方法

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

```
from("aws2-s3://helloBucket?
accessKey=yourAccessKey&secretKey=yourSecretKey&prefix=hello.txt")
  .to("file:/var/downloaded");
```

9.7.1. 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**	用于当前操作的存储桶目标名称
CamelAwsS3ContentLength	**Long**	此对象的内容长度。
CamelAwsS3ContentType	**字符串**	此对象的内容类型。
CamelAwsS3ContentControl	**字符串**	此对象的内容控制。
CamelAwsS3ContentDisposition	**字符串**	此对象的内容分布。
CamelAwsS3ContentEncoding	**字符串**	此对象的内容编码。
CamelAwsS3ContentMD5	**字符串**	此对象的 md5 checksum。

标头	类型	描述
CamelAwsS3DestinationKey	**字符串**	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**	此对象的最后修改的时间戳。
CamelAwsS3Operation	**字符串**	要执行的操作。允许的值有 copyObject, deleteObject, listBuckets, deleteBucket, listObjects
CamelAwsS3StorageClass	**字符串**	此对象的存储类。
CamelAwsS3CannedAcl	**字符串**	将应用于对象的 canned acl。请参阅 **software.amazon.awssdk.services.s3.model.ObjectCannedACL** 以了解允许的值。
CamelAwsS3Acl	**software.amazon.awssdk.services.s3.model.BucketCannedACL**	一个精心构建的 Amazon S3 Access Control List 对象。请参阅 **software.amazon.awssdk.services.s3.model.BucketCannedACL**。
CamelAwsS3ServerSideEncryption	字符串	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
CamelAwsS3Metadata	**Map<String, String>**	要与 S3 中对象存储的元数据映射。有关元数据的更多详细信息。

9.7.2. 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。

9.7.3. 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。

标头	类型	描述
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。
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	使用 AWS 管理的密钥加密对象时的服务器端加密算法。
CamelAwsS3Metadata	**Map<String, String>**	与 S3 中对象存储的元数据映射。有关元数据的更多详细信息。

9.7.4. S3 Producer 操作

Camel-AWS2-S3 组件在生成者端提供以下操作：

- copyObject
- deleteObject

- listBuckets
- deleteBucket
- listObjects
- GetObject （这将返回 S3Object 实例）
- getObjectRange （这将返回 S3Object 实例）
- createDownloadLink

如果您没有显式指定生成者将执行的操作： - 单个文件上传 - 如果启用了 multiPartUpload 选项，则多部分上传。

9.7.5. 高级 AmazonS3 配置

如果您的 Camel 应用程序在防火墙后面运行，或者需要对 **S3Client** 实例配置拥有更多控制，您可以创建自己的实例，并在 Camel aws2-s3 组件配置中引用它：

```
from("aws2-s3://MyBucket?amazonS3Client=#client&delay=5000&maxMessagesPerPoll=5")
.to("mock:result");
```

9.7.6. 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("aws2-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。

9.7.7. 静态凭证和默认凭证提供程序

您可以通过指定 useDefaultCredentialsProvider 选项并将其设置为 true 来避免使用显式静态凭证。

- Java 系统属性 - aws.accessKeyId 和 aws.secretKey
- 环境变量 - AWS_ACCESS_KEY_ID 和 AWS_SECRET_ACCESS_KEY。
- AWS STS 的 Web Identity Token。
- 共享凭证和配置文件。
- Amazon ECS 容器凭证 - 如果设置了环境变量 AWS_CONTAINER_CREDENTIALS_RELATIVE_URI，则从 Amazon ECS 加载。
- Amazon EC2 实例配置集凭据。

有关此信息的更多信息，您可以查看 AWS 凭证文档

9.7.8. S3 Producer 操作示例

- 单上传：此操作将根据正文内容上传文件到 S3

```
from("direct:start").process(new Processor() {

    @Override
    public void process(Exchange exchange) throws Exception {
        exchange.getIn().setHeader(S3Constants.KEY, "camel.txt");
        exchange.getIn().setBody("Camel rocks!");
    }
})
.to("aws2-s3://mycamelbucket?amazonS3Client=#amazonS3Client")
.to("mock:result");
```

此操作将上传文件 camel.txt，其内容为 mycamelbucket bucket 中的内容 "Camel rocks!"

- 多部分上传：此操作将根据正文内容执行文件的多部分上传到 S3

```
from("direct:start").process(new Processor() {

    @Override
    public void process(Exchange exchange) throws Exception {
        exchange.getIn().setHeader(AWS2S3Constants.KEY, "empty.txt");
        exchange.getIn().setBody(new File("src/empty.txt"));
    }
})
.to("aws2-s3://mycamelbucket?
amazonS3Client=#amazonS3Client&multiPartUpload=true&autoCreateBucket=true&partSize=1048576"
)
.to("mock:result");
```

此操作将执行文件 empty.txt 的多部分上传，它基于 mycamelbucket bucket 中文件 src/empty.txt 的内容

- CopyObject：此操作将对象从一个存储桶复制到不同的存储桶

```
from("direct:start").process(new Processor() {

    @Override
    public void process(Exchange exchange) throws Exception {
        exchange.getIn().setHeader(S3Constants.BUCKET_DESTINATION_NAME,
"camelDestinationBucket");
        exchange.getIn().setHeader(S3Constants.KEY, "camelKey");
        exchange.getIn().setHeader(S3Constants.DESTINATION_KEY, "camelDestinationKey");
    }
})
.to("aws2-s3://mycamelbucket?amazonS3Client=#amazonS3Client&operation=copyObject")
.to("mock:result");
```

此操作会将带有标头 camelDestinationKey 中的名称的对象复制到 Bucket mycamelbucket 中的 camelDestinationBucket 存储桶。

- DeleteObject：此操作从存储桶中删除对象

```
from("direct:start").process(new Processor() {
```

```
    @Override
    public void process(Exchange exchange) throws Exception {
        exchange.getIn().setHeader(S3Constants.KEY, "camelKey");
    }
})
.to("aws2-s3://mycamelbucket?amazonS3Client=#amazonS3Client&operation=deleteObject")
.to("mock:result");
```

此操作将从 bucket mycamelbucket 中删除对象 camelKey。

- ListBuckets：此操作列出了此区域中此帐户的存储桶

```
from("direct:start")
.to("aws2-s3://mycamelbucket?amazonS3Client=#amazonS3Client&operation=listBuckets")
.to("mock:result");
```

此操作将列出此帐户的存储桶

- DeleteBucket：此操作删除指定为 URI 参数或标头的存储桶

```
from("direct:start")
.to("aws2-s3://mycamelbucket?amazonS3Client=#amazonS3Client&operation=deleteBucket")
.to("mock:result");
```

此操作将删除存储桶 mycamelbucket

- ListObjects ：此操作列表在特定存储桶中的对象

```
from("direct:start")
.to("aws2-s3://mycamelbucket?amazonS3Client=#amazonS3Client&operation=listObjects")
.to("mock:result");
```

此操作将列出 mycamelbucket bucket 中的对象

- GetObject：此操作获取特定存储桶中的单个对象

```
from("direct:start").process(new Processor() {

    @Override
    public void process(Exchange exchange) throws Exception {
        exchange.getIn().setHeader(S3Constants.KEY, "camelKey");
    }
})
.to("aws2-s3://mycamelbucket?amazonS3Client=#amazonS3Client&operation=getObject")
.to("mock:result");
```

此操作将返回与 mycamelbucket bucket 中 camelKey 对象相关的 S3Object 实例。

- GetObjectRange：此操作获得特定存储桶中的单个对象范围

```
from("direct:start").process(new Processor() {

    @Override
```

```
        public void process(Exchange exchange) throws Exception {
            exchange.getIn().setHeader(S3Constants.KEY, "camelKey");
            exchange.getIn().setHeader(S3Constants.RANGE_START, "0");
            exchange.getIn().setHeader(S3Constants.RANGE_END, "9");
        }
    })
    .to("aws2-s3://mycamelbucket?amazonS3Client=#amazonS3Client&operation=getObjectRange")
    .to("mock:result");
```

此操作将返回与 mycamelbucket bucket 中 camelKey 对象相关的 S3Object 实例，其中包含从 0 到 9 的字节数。

- CreateDownloadLink：此操作将通过 S3 Presigner 返回下载链接

```
from("direct:start").process(new Processor() {

    @Override
    public void process(Exchange exchange) throws Exception {
        exchange.getIn().setHeader(S3Constants.KEY, "camelKey");
    }
})
.to("aws2-s3://mycamelbucket?
accessKey=xxx&secretKey=yyy&region=region&operation=createDownloadLink")
.to("mock:result");
```

此操作将返回存储桶 mycamelbucket 和 region 区域中的 camel-key 文件的下载链接 url

9.8. 流上传模式

启用流模式后，用户可以通过多部分上传将数据上传到 S3，而无需提前了解数据维度的时间。上传将在完成后完成： batchSize 已完成，或者达到 batchMessageNumber。有两个可能的命名策略：

- progressive
  使用 progressive 策略，每个文件的名称都由 keyName 选项和一个 progressive 计数器组成，最终文件扩展名（若有）

- random。
  使用随机策略时，将在 keyName 后添加 UUID，最终会附加文件扩展名。

例如：

```
from(kafka("topic1").brokers("localhost:9092"))
        .log("Kafka Message is: ${body}")
        .to(aws2S3("camel-
bucket").streamingUploadMode(true).batchMessageNumber(25).namingStrategy(AWS2S3EndpointBu
ilderFactory.AWSS3NamingStrategyEnum.progressive).keyName("
{{kafkaTopic1}}/{{kafkaTopic1}}.txt"));

from(kafka("topic2").brokers("localhost:9092"))
        .log("Kafka Message is: ${body}")
        .to(aws2S3("camel-
bucket").streamingUploadMode(true).batchMessageNumber(25).namingStrategy(AWS2S3EndpointBu
ilderFactory.AWSS3NamingStrategyEnum.progressive).keyName("
{{kafkaTopic2}}/{{kafkaTopic2}}.txt"));
```

批处理的默认大小为 1 Mb，但您可以根据您的要求进行调整。

当您停止生成者路由时，生成者将负责刷新剩余的缓冲消息，并完成上传。

在流上传中，您将能够从离开的时间点重新启动生成者。务必要注意，只有在使用进度命名策略时，此功能才至关重要。

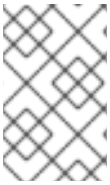
通过将 restartPolicy 设置为 lastPart，您将重启从制作者左侧最后一个部分编号上传文件和内容。

示例

1. 使用 progressive naming strategy 和 keyname 等于 camel.txt 来启动路由，batchMessageNumber 等于 20，restartPolicy 等于 lastPart - Send 70 消息。
2. 停止路由
3. 在您的 S3 存储桶中，您现在应该看到 4 个文件：* camel.txt
   - camel-1.txt
   - camel-2.txt
   - camel-3.txt
     前三个消息将有 20 个消息，而最后一个消息仅有 10 个。
4. 重新启动路由。
5. 发送 25 个消息。
6. 停止路由。
7. 您的存储桶中现在有 2 个其他文件： camel-5.txt 和 camel-6.txt，第一个带有 20 个消息，第二个文件为 5 个信息。
8. 继续

使用随机命名策略时不需要这样做。

相反，您可以指定覆盖重启策略。在这种情况下，您可以覆盖您在存储桶上之前（用于该特定 keyName）写入的任何内容。



注意

在流上传模式中，将考虑的唯一 keyName 选项是端点选项。使用标头将抛出 NPE，这由设计完成。设置标头意味着可能会更改每个交换上的文件名，这针对流上传制作者的动画。keyName 需要修复和静态。所选命名策略将执行其余工作。

另一个可能是使用 batchMessageNumber 和 batchSize 选项指定 streamingUploadTimeout。使用此选项时，用户可以在特定时间通过后完成文件上传。这样，上传完成将在三个层上传递：超时、消息数和批处理大小。

例如：

```
from(kafka("topic1").brokers("localhost:9092"))
        .log("Kafka Message is: ${body}")
        .to(aws2S3("camel-
```

```
bucket").streamingUploadMode(true).batchMessageNumber(25).streamingUploadTimeout(10000).na
mingStrategy(AWS2S3EndpointBuilderFactory.AWSS3NamingStrategyEnum.progressive).keyName(
"{{kafkaTopic1}}/{{kafkaTopic1}}.txt"));
```

在这种情况下，上传将在 10 秒后完成。

9.9. BUCKET 自动创建

使用选项 **autoCreateBucket** 用户可以在 S3 Bucket 不存在时避免自动创建。此选项的默认值是 **true**。如果设置为 false 对 AWS 中不存在的存储桶的操作，则不会成功，并返回错误。

9.10. 在存储桶和其他存储桶间移动操作

有些用户（如从存储桶中消耗大量），并在不同的中移动内容，而无需使用这个组件的 copyObject 功能。如果是这样，请不要忘记从消费者的传入交换中删除 bucketName 标头，否则该文件将始终覆盖同一原始存储桶。

9.11. MOVEAFTERREAD CONSUMER 选项

除了 deleteAfterRead 外，还添加了另一个选项 moveAfterRead。启用此选项后，消耗的对象将移到目标 destinationBucket 中，而不是只被删除。这将需要指定 destinationBucket 选项。例如：

```
  from("aws2-s3://mycamelbucket?
amazonS3Client=#amazonS3Client&moveAfterRead=true&destinationBucket=myothercamelbucket")
  .to("mock:result");
```

在这种情况下，消耗的对象将移到 myothercamelbucket bucket，并从原始存储桶中删除（因为 deleteAfterRead 设置为 true ）。

您还可以在将文件移动到其他存储桶时使用密钥前缀/suffix。这些选项是 destinationBucketPrefix 和 destinationBucketSuffix。

使用以上示例，您可以执行以下操作：

```
  from("aws2-s3://mycamelbucket?
amazonS3Client=#amazonS3Client&moveAfterRead=true&destinationBucket=myothercamelbucket&de
tinationBucketPrefix=RAW(pre-)&destinationBucketSuffix=RAW(-suff)")
  .to("mock:result");
```

在这种情况下，消耗的对象将移到 myothercamelbucket bucket，并从原始存储桶中删除（因为 deleteAfterRead 设置为 true ）。

因此，如果文件名是 test，在 myothercamelbucket 中，您应该会看到一个名为 pre-test-suff 的文件。

9.12. 使用客户密钥加密

我们还引入了客户密钥支持（使用 KMS 的替代方案）。以下代码显示了一个示例。

```
String key = UUID.randomUUID().toString();
byte[] secretKey = generateSecretKey();
String b64Key = Base64.getEncoder().encodeToString(secretKey);
String b64KeyMd5 = Md5Utils.md5AsBase64(secretKey);
```

```
String awsEndpoint = "aws2-s3://mycamel?
autoCreateBucket=false&useCustomerKey=true&customerKeyId=RAW(" + b64Key +
")&customerKeyMD5=RAW(" + b64KeyMd5 + ")&customerAlgorithm=" + AES256.name();

from("direct:putObject")
    .setHeader(AWS2S3Constants.KEY, constant("test.txt"))
    .setBody(constant("Test"))
    .to(awsEndpoint);
```

9.13. 使用 POJO 作为正文

由于多个选项，有时构建 AWS Request 可能会很复杂。我们介绍可能将 POJO 用作正文。在 AWS S3 中，您可以提交多个操作，作为 List 代理请求示例，您可以执行以下操作：

```
from("direct:aws2-s3")
     .setBody(ListObjectsRequest.builder().bucket(bucketName).build())
     .to("aws2-s3://test?
amazonS3Client=#amazonS3Client&operation=listObjects&pojoRequest=true")
```

这样，您将直接传递请求，而无需专门传递与此操作相关的标头和选项。

9.14. 创建 S3 客户端并在 REGISTRY 中添加组件

有时，您要使用 AWS2S3Configuration 执行一些高级配置，这还允许设置 S3 客户端。您可以在组件配置中创建和设置 S3 客户端，如下例所示

```
String awsBucketAccessKey = "your_access_key";
String awsBucketSecretKey = "your_secret_key";

S3Client s3Client =
S3Client.builder().credentialsProvider(StaticCredentialsProvider.create(AwsBasicCredentials.create(aws
BucketAccessKey, awsBucketSecretKey)))
            .region(Region.US_EAST_1).build();

AWS2S3Configuration configuration = new AWS2S3Configuration();
configuration.setAmazonS3Client(s3Client);
configuration.setAutoDiscoverClient(true);
configuration.setBucketName("s3bucket2020");
configuration.setRegion("us-east-1");
```

现在，您可以配置 S3 组件（使用上面创建的配置对象），并在路由初始化前将其添加到配置方法中的 registry 中。

```
AWS2S3Component s3Component = new AWS2S3Component(getContext());
s3Component.setConfiguration(configuration);
s3Component.setLazyStartProducer(true);
camelContext.addComponent("aws2-s3", s3Component);
```

现在，您的组件将用于在 camel 路由中实施的所有操作。

9.15. 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-aws2-s3</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 **{camel-version}** 必须替换为 Camel 的实际版本。

9.16. 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.aws2-s3.access-key	Amazon AWS 访问密钥.		字符串
camel.component.aws2-s3.amazon-s3-client	对 registry 中的 com.amazonaws.services.s3.AmazonS3 的引用。选项是一个 software.amazon.awssdk.services.s3.S3Client 类型。		S3Client
camel.component.aws2-s3.amazon-s3-presigner	用于 Request 的 S3 Presigner 主要在 createDownloadLink 操作中使用。选项是一个 software.amazon.awssdk.services.s3.presigner.S3Presigner 类型。		S3Presigner
camel.component.aws2-s3.auto-create-bucket	设置 S3 存储桶自动创建的 bucketName。如果启用了 moveAfterRead 选项，则也会应用它，如果尚未存在 moveAfterRead 选项，它将创建 destinationBucket。	false	布尔值
camel.component.aws2-s3.autoclose-body	如果此选项为 true，且 includeBody 为 false，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 false，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.aws2-s3.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.aws2-s3.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.aws2-s3.batch-message-number	在流传输上传模式中制作批处理的消息数量。	10	整数
camel.component.aws2-s3.batch-size	流上传模式的批处理大小（以字节为单位）。	1000000	整数
camel.component.aws2-s3.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.aws2-s3.configuration	组件配置。选项是 org.apache.camel.component.aws2.s3.AWS2S3Configuration 类型。		AWS2S3Configuration
camel.component.aws2-s3.customer-algorithm	定义在启用了 CustomerKey 时要使用的客户算法。		字符串
camel.component.aws2-s3.customer-key-id	定义在启用 CustomerKey 时要使用的 Customer key 的 id。		字符串
camel.component.aws2-s3.customer-key-m-d5	定义在启用 CustomerKey 时要使用的客户密钥的 MD5。		字符串
camel.component.aws2-s3.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 AWS2S3Constants#BUCIKET_NAME 和 AWS2S3Constants#KEY 标头过滤，或者只过滤 AWS2S3Constants#KEY 标头。	true	布尔值
camel.component.aws2-s3.delete-after-write	在 S3 文件上传后删除文件对象。	false	布尔值

Name	描述	默认值	类型
camel.component.aws2-s3.delimiter	com.amazonaws.services.s3.model.ListObjectsRequest 中使用的分隔符仅消耗我们感兴趣的对象。		字符串
camel.component.aws2-s3.destination-bucket	定义当 moveAfterRead 设置为 true 时必须移动对象的目标存储桶。		字符串
camel.component.aws2-s3.destination-bucket-prefix	定义在必须移动对象并将 moveAfterRead 设置为 true 时使用的目标存储桶前缀。		字符串
camel.component.aws2-s3.destination-bucket-suffix	定义在必须移动对象并将 moveAfterRead 设置为 true 时使用的目标存储桶后缀。		字符串
camel.component.aws2-s3.done-file-name	如果提供，Camel 仅在文件存在时使用文件。		字符串
camel.component.aws2-s3.enabled	是否启用 aws2-s3 组件的自动配置。这默认是启用的。		布尔值
camel.component.aws2-s3.file-name	要从具有给定文件名的存储桶获取对象。		字符串
camel.component.aws2-s3.ignore-body	如果为 true，则 S3 对象正文将完全忽略，如果设为 false，则 S3 对象将放入正文中。把它设置为 true，将覆盖 includeBody 选项定义的任何行为。	false	布尔值
camel.component.aws2-s3.include-body	如果为 true，则 S3Object Exchange 将被使用并放入正文和关闭中。如果为 false，S3Object 流将原始放在正文中，标头将使用 S3 对象元数据设置。这个选项与 autocloseBody 选项密切相关。如果将 includeBody 设为 true，因为 S3Object 流将被消耗，然后也会关闭它，而在 includeBody false 时，它将是关闭 S3Object 流的调用者。但是，当 includeBody 为 false 时，将 autocloseBody 设置为 true，它将在交换完成时自动关闭 S3Object 流。	true	布尔值
camel.component.aws2-s3.include-folders	如果为 true，将消费的文件夹/目录。如果是 false，则忽略它们，且不会为那些交换创建。	true	布尔值

Name	描述	默认值	类型
camel.component.aws2-s3.key-name	通过端点参数在存储桶中设置元素的密钥名称。		字符串
camel.component.aws2-s3.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.aws2-s3.move-after-read	在检索后，将对象从 S3 存储桶移到不同的存储桶。要完成操作，必须设置 destinationBucket 选项。仅当 Exchange 提交时，才会执行复制存储桶操作。如果进行回滚，则对象不会被移动。	false	布尔值
camel.component.aws2-s3.multi-part-upload	如果为 true，则 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.aws2-s3.naming-strategy	在流上传模式中使用的命名策略。		AWSS3NamingStrategyEnum
camel.component.aws2-s3.operation	当用户不希望只进行上传时，要执行的操作。		AWS2S3Operations
camel.component.aws2-s3.override-endpoint	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
camel.component.aws2-s3.part-size	设置多部分上传中使用的 partSize，默认大小为 25M。	26214400	Long
camel.component.aws2-s3.pojo-request	如果您想要将 POJO 请求用作正文。	false	布尔值
camel.component.aws2-s3.policy	此队列的策略在 com.amazonaws.services.s3.AmazonS3#setBucketPolicy() 方法中设置。		字符串
camel.component.aws2-s3.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.aws2-s3.proxy-host	在实例化 SQS 客户端时定义代理主机。		字符串
camel.component.aws2-s3.proxy-port	指定要在客户端定义中使用的代理端口。		整数
camel.component.aws2-s3.proxy-protocol	在实例化 S3 客户端时定义代理协议。		协议
camel.component.aws2-s3.region	S3 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串
camel.component.aws2-s3.restarting-policy	在流上传模式中使用的重启策略。		AWSS3RestartingPolicyEnum
camel.component.aws2-s3.secret-key	Amazon AWS Secret 密钥。		字符串
camel.component.aws2-s3.storage-class	在 com.amazonaws.services.s3.model.PutObjectRequest 请求中设置的存储类。		字符串
camel.component.aws2-s3.streaming-upload-mode	当流模式为 true 时，上传到存储桶将以流传输方式进行。	false	布尔值
camel.component.aws2-s3.streaming-upload-timeout	在流上传模式为 true 时，此选项会将超时设置为完成上传。		Long
camel.component.aws2-s3.trust-all-certificates	如果要在覆盖端点时信任所有证书。	false	布尔值
camel.component.aws2-s3.uri-endpoint-override	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串

Name	描述	默认值	类型
camel.component.aws2-s3.use-aws-k-m-s	定义是否必须使用 KMS。	false	布尔值
camel.component.aws2-s3.use-customer-key	定义是否需要使用客户密钥。	false	布尔值
camel.component.aws2-s3.use-default-credentials-provider	设置 S3 客户端是否应该希望通过默认凭据提供程序加载凭据，或者希望传递静态凭据。	false	布尔值

第 10 章 AWS SIMPLE NOTIFICATION SYSTEM (SNS)

仅支持生成者

AWS2 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。

10.1. 依赖项

当在 Camel Spring Boot 中使用 **aws2-sns** 时，请将以下 Maven 依赖项添加到 **pom.xml** 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-aws2-sns-starter</artifactId>
</dependency>
```

10.2. URI 格式

```
aws2-sns://topicNameOrArn[?options]
```

如果主题不存在，则会创建它们。您可以将查询选项附加到 URI 中，格式为 **?options=value&option2=value&...**

10.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

10.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

10.3.2. 端点级别选项

在 **Endpoint 级别**，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 **路径和** 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

10.4. 组件选项

AWS Simple Notification System (SNS) 组件支持 24 个选项，如下所列。

Name	描述	默认值	类型
amazonSNSClient (producer)	**Autowired** 使用 AmazonSNS 作为客户端。		SnsClient
autoCreateTopic (producer)	设置主题的自动创建。	false	布尔值
配置 （生成者）	组件配置.		Sns2Configuration
kmsMasterKeyId (producer)	用于 Amazon SNS 或自定义 CMK 的 AWS 管理的客户主密钥 (CMK) 的 ID。		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
messageDeduplicationIdStrategy (producer)	仅适用于 FIFO 主题。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。 Enum 值： • useExchangeId • useContentBasedDeduplication	useExchangeId	字符串

Name	描述	默认值	类型
messageGroupIdStrategy (producer)	仅适用于 FIFO 主题。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。 Enum 值： • useConstant • useExchangeId • usePropertyValue		字符串
messageStructure (producer)	使用 json 的消息结构。		字符串
overrideEndpoint (producer)	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
policy (producer)	本主题的策略。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
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
proxyProtocol (producer)	在实例化 SNS 客户端时定义代理协议。 Enum 值： • HTTP • HTTPS	HTTPS	协议
queueUrl (producer)	要订阅的 queueUrl。		字符串
region (producer)	SNS 客户端需要在其中工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串
serverSideEncryptionEnabled (producer)	定义是否在主题中启用 Server Side Encryption。	false	布尔值

Name	描述	默认值	类型
subject (producer)	如果邮件标头 'CamelAwsSnsSubject' 不存在，则使用主题。		字符串
subscribeSNStoSQS (producer)	定义 SNS 主题和 SQS 之间的订阅是否必须完成。	false	布尔值
trustAllCertificates (producer)	如果要在覆盖端点时信任所有证书。	false	布尔值
uriEndpointOverride (producer)	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
useDefaultCredentialsProvider (producer)	设置 SNS 客户端是否应该预期在 AWS infra 实例上加载凭证，或希望传递静态凭证。	false	布尔值
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	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

10.5. 端点选项

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

```
aws2-sns:topicNameOrArn
```

使用以下路径和查询参数：

10.5.1. 路径参数(1 参数)

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

10.5.2. 查询参数(23 参数)

Name	描述	默认值	类型
amazonSNSClient (producer)	**Autowired** 使用 AmazonSNS 作为客户端。		SnsClient
autoCreateTopic (producer)	设置主题的自动创建。	false	布尔值
headerFilterStrategy (producer)	使用自定义 HeaderFilterStrategy 将标头映射到/来自 Camel。		HeaderFilterStrategy
kmsMasterKeyId (producer)	用于 Amazon SNS 或自定义 CMK 的 AWS 管理的客户主密钥 (CMK) 的 ID。		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
messageDeduplicationIdStrategy (producer)	仅适用于 FIFO 主题。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。 Enum 值： • useExchangeId • useContentBasedDeduplication	useExchangeId	字符串
messageGroupIdStrategy (producer)	仅适用于 FIFO 主题。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。 Enum 值： • useConstant • useExchangeId • usePropertyValue		字符串
messageStructure (producer)	使用 json 的消息结构。		字符串

Name	描述	默认值	类型
overrideEndpoint (producer)	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
policy (producer)	本主题的策略。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
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
proxyProtocol (producer)	在实例化 SNS 客户端时定义代理协议。 Enum 值： • HTTP • HTTPS	HTTPS	协议
queueUrl (producer)	要订阅的 queueUrl。		字符串
region (producer)	SNS 客户端需要在其中工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串
serverSideEncryptionEnabled (producer)	定义是否在主题中启用 Server Side Encryption。	false	布尔值
subject (producer)	如果邮件标头 'CamelAwsSnsSubject' 不存在，则使用主题。		字符串
subscribeSNStoSQS (producer)	定义 SNS 主题和 SQS 之间的订阅是否必须完成。	false	布尔值
trustAllCertificates (producer)	如果要在覆盖端点时信任所有证书。	false	布尔值
uriEndpointOverride (producer)	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
useDefaultCredentialsProvider (producer)	设置 SNS 客户端是否应该预期在 AWS infra 实例上加载凭证，或希望传递静态凭证。	false	布尔值

Name	描述	默认值	类型
accessKey (security)	Amazon AWS 访问密钥.		字符串
secretKey (security)	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。

10.6. 使用方法

10.6.1. 静态凭证和默认凭证提供程序

您可以通过指定 useDefaultCredentialsProvider 选项并将其设置为 true 来避免使用显式静态凭证。

- Java 系统属性 - aws.accessKeyId 和 aws.secretKey
- 环境变量 - AWS_ACCESS_KEY_ID 和 AWS_SECRET_ACCESS_KEY。
- AWS STS 的 Web Identity Token。
- 共享凭证和配置文件。
- Amazon ECS 容器凭证 - 如果设置了环境变量 AWS_CONTAINER_CREDENTIALS_RELATIVE_URI，则从 Amazon ECS 加载。
- Amazon EC2 实例配置集凭据。

有关此信息的更多信息，您可以查看 AWS 凭证文档。

10.6.2. 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。

10.6.3. SNS producer 设置的消息标头

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

10.6.4. 高级 AmazonSNS 配置

如果需要对 **SnsClient** 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("aws2-sns://MyTopic?amazonSNSClient=#client");
```

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

10.6.5. 在 AWS SNS 主题和 AWS SQS Queue 之间创建订阅

您可以创建一个 SQS Queue 订阅到 SNS 主题：

```
from("direct:start")
.to("aws2-sns://test-camel-sns1?
amazonSNSClient=#amazonSNSClient&subscribeSNStoSQS=true&queueUrl=https://sqs.eu-central-
1.amazonaws.com/780410022472/test-camel");
```

#amazonSNSClient 是指 Registry 中的 **SnsClient**。通过将 **subscribeSNStoSQS** 指定为 true，并且指定现有 SQS 队列的 **queueUrl**，您可以将 SQS Queue 订阅到您的 SNS 主题。

此时，您可以通过 SQS Queue 使用来自 SNS 主题的消息

```
from("aws2-sqs://test-camel?
amazonSQSClient=#amazonSQSClient&delay=50&maxMessagesPerPoll=5")
    .to(...);
```

10.7. TOPIC AUTOCREATION

通过选项 **autoCreateTopic** 用户，如果 SNS Topic 不存在，可以避免自动创建它。此选项的默认值是 **true**。如果设置为 false 任何对 AWS 中不存在的主题的操作，则不会成功，并返回错误。

10.8. SNS FIFO

支持 SNS FIFO。在创建 SQS 队列时，您将订阅 SNS 主题，需要记住，您需要让 SNS Topic 发送消息到 SQS Queue。

示例

假设您创建一个名为 **Order.fifo** 的 SNS FIFO 主题，以及一个名为 **QueueSub.fifo** 的 SQS Queue。

在 **QueueSub.fifo** 的访问策略中，您应该提交如下内容：

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__owner_statement",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::780560123482:root"
      },
      "Action": "SQS:*",
      "Resource": "arn:aws:sqs:eu-west-1:780560123482:QueueSub.fifo"
    },
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "SQS:SendMessage",
      "Resource": "arn:aws:sqs:eu-west-1:780560123482:QueueSub.fifo",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:sns:eu-west-1:780410022472:Order.fifo"
        }
      }
    }
  ]
}
```

这是使订阅正常工作的关键步骤。

10.8.1. SNS Fifo Topic 消息组 Id Strategy 和消息 Deduplication Id Strategy

向 FIFO 主题发送一些时，您需要始终设置消息组 Id 策略。

如果 SNS Fifo 主题上启用了基于内容的消息 deduplication，则不需要设置消息 deduplication id 策略，否则您必须对其进行设置。

10.9. 例子

10.9.1. 生成者示例

发送到主题

```
from("direct:start")
  .to("aws2-sns://camel-topic?subject=The+subject+message&autoCreateTopic=true");
```

10.10. 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-aws2-sns</artifactId>
    <version>${camel-version}</version>
</dependency>
```

其中 **{camel-version}** 必须替换为 Camel 的实际版本。

10.11. 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.aws2-sns.access-key	Amazon AWS 访问密钥.		字符串
camel.component.aws2-sns.amazon-s-n-s-client	将 AmazonSNS 用作客户端。选项是一个 software.amazon.awssdk.services.sns.SnsClient 类型。		SnsClient
camel.component.aws2-sns.auto-create-topic	设置主题的自动创建。	false	布尔值
camel.component.aws2-sns.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.aws2-sns.configuration	组件配置.选项是 org.apache.camel.component.aws2.sns.Sns2Configuration 类型。		Sns2Configuration
camel.component.aws2-sns.enabled	是否启用 aws2-sns 组件的自动配置。这默认是启用的。		布尔值
camel.component.aws2-sns.kms-master-key-id	用于 Amazon SNS 或自定义 CMK 的 AWS 管理的客户主密钥 (CMK) 的 ID。		字符串
camel.component.aws2-sns.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.aws2-sns.message-deduplication-id-strategy	仅适用于 FIFO 主题。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	字符串
camel.component.aws2-sns.message-group-id-strategy	仅适用于 FIFO 主题。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。		字符串

Name	描述	默认值	类型
camel.component.aws2-sns.message-structure	使用 json 的消息结构。		字符串
camel.component.aws2-sns.override-endpoint	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
camel.component.aws2-sns.policy	本主题的策略。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.aws2-sns.proxy-host	在实例化 SNS 客户端时定义代理主机。		字符串
camel.component.aws2-sns.proxy-port	在实例化 SNS 客户端时定义代理端口。		整数
camel.component.aws2-sns.proxy-protocol	在实例化 SNS 客户端时定义代理协议。		协议
camel.component.aws2-sns.queue-url	要订阅的 queueUrl。		字符串
camel.component.aws2-sns.region	SNS 客户端需要在其中工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串
camel.component.aws2-sns.secret-key	Amazon AWS Secret 密钥。		字符串
camel.component.aws2-sns.server-side-encryption-enabled	定义是否在主题中启用 Server Side Encryption。	false	布尔值
camel.component.aws2-sns.subject	如果邮件标头 'CamelAwsSnsSubject' 不存在，则使用主题。		字符串

Name	描述	默认值	类型
camel.component.aws2-sns.subscribe-s-n-sto-s-q-s	定义 SNS 主题和 SQS 之间的订阅是否必须完成。	false	布尔值
camel.component.aws2-sns.trust-all-certificates	如果要在覆盖端点时信任所有证书。	false	布尔值
camel.component.aws2-sns.uri-endpoint-override	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
camel.component.aws2-sns.use-default-credentials-provider	设置 SNS 客户端是否应该预期在 AWS infra 实例上加载凭证，或希望传递静态凭证。	false	布尔值

第 11 章 AWS SIMPLE QUEUE SERVICE (SQS)

支持生成者和消费者

AWS2 SQS 组件支持向 Amazon 的 SQS 服务 发送和接收信息。

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。

11.1. 依赖项

当在 Camel Spring Boot 中使用 aws2-sqs 时，请将以下 Maven 依赖项添加到 **pom.xml** 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-aws2-sqs-starter</artifactId>
</dependency>
```

11.2. URI 格式

```
aws2-sqs://queueNameOrArn[?options]
```

如果队列不存在，将创建队列。您可以以以下格式将查询选项附加到 URI 中，

?options=value&option2=value&...

11.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

11.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

11.3.2. 端点级别选项

在 **Endpoint 级别**，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 **路径和** 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

11.4. 组件选项

AWS Simple Queue Service (SQS) 组件支持 43 选项，如下所列。

Name	描述	默认值	类型
amazonAWSHost (common)	Amazon AWS 云的主机名。	amazonaws.com	字符串
amazonSQSClient (common)	**Autowired** 以将 AmazonSQS 用作客户端。		SqsClient
autoCreateQueue (common)	设置队列的自动创建.	false	布尔值
configuration (common)	AWS SQS 默认配置。		Sqs2Configuration
overrideEndpoint (common)	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
protocol (common)	用于与 SQS 通信的底层协议。	https	字符串
proxyProtocol (common)	在实例化 SQS 客户端时定义代理协议。 Enum 值： • HTTP • HTTPS	HTTPS	协议
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)	SQS 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串
trustAllCertificates (common)	如果要在覆盖端点时信任所有证书。	false	布尔值

Name	描述	默认值	类型
uriEndpointOverride (common)	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
useDefaultCredentialsProvider (common)	设置 SQS 客户端是否应该预期在 AWS infra 实例上加载凭证，或希望传递静态凭证。	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
deleteIfFiltered (consumer)	如果交换具有键 Sqs2Constants#SQS_DELETE_FILTERED (CamelAwsSqsDeleteFiltered)，则是否将 DeleteMessage 发送到 SQS 队列。	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。	false	布尔值
kmsDataKeyReusePeriodSeconds (consumer)	Amazon SQS 在再次调用 AWS KMS 之前，以便 Amazon SQS 可以重复使用或解密信息的时间长度（以秒为单位）。一个代表秒的整数，在 60 秒 (1 分钟) 和 86,400 秒 (24 小时)之间。默认：300 (5 分钟)。		整数
kmsMasterKeyId (consumer)	Amazon SQS 或自定义 CMK 的 AWS 管理的客户主密钥 (CMK) 的 ID。		字符串

Name	描述	默认值	类型
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
serverSideEncryptionEnabled (consumer)	定义是否在队列中启用服务器端加密。	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
batchSeparator (producer)	在传递 String 以发送批处理消息操作时，设置分隔符。	,	字符串
delaySeconds (producer)	延迟发送消息的秒数。		整数
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
messageDeduplicationIdStrategy (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。 Enum 值： • useExchangeId • useContentBasedDeduplication	useExchangeId	字符串

Name	描述	默认值	类型
messageGroupIdStrategy (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。 Enum 值： • useConstant • useExchangeId • usePropertyValue		字符串
operation (producer)	当用户不想发送消息时，要执行的操作。 Enum 值： • sendBatchMessage • deleteMessage • listQueues • purgeQueue • deleteQueue		Sqs2Operations
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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
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

Name	描述	默认值	类型
policy (queue)	此队列的策略。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
receiveMessageWaitTimeSeconds (queue)	如果您没有在请求中指定 WaitTimeSeconds，则使用 queue 属性 ReceiveMessageWaitTimeSeconds 来确定等待的时长。		整数
redrivePolicy (queue)	指定发送消息到 DeadLetter 队列的策略。请参阅 Amazon 文档的详情。		字符串
accessKey (security)	Amazon AWS 访问密钥.		字符串
secretKey (security)	Amazon AWS Secret 密钥。		字符串

11.5. 端点选项

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

```
aws2-sqs:queueNameOrArn
```

使用以下路径和查询参数：

11.5.1. 路径参数(1 参数)

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

11.5.2. 查询参数(61 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)	**Autowired** 以将 AmazonSQS 用作客户端。		SqsClient
autoCreateQueue (common)	设置队列的自动创建.	false	布尔值

Name	描述	默认值	类型
headerFilterStrategy (common)	使用自定义 HeaderFilterStrategy 将标头映射到/来自 Camel。		HeaderFilterStrategy
overrideEndpoint (common)	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
protocol (common)	用于与 SQS 通信的底层协议。	https	字符串
proxyProtocol (common)	在实例化 SQS 客户端时定义代理协议。 Enum 值： • HTTP • HTTPS	HTTPS	协议
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)	SQS 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串
trustAllCertificates (common)	如果要在覆盖端点时信任所有证书。	false	布尔值
uriEndpointOverride (common)	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
useDefaultCredentialsProvider (common)	设置 SQS 客户端是否应该预期在 AWS infra 实例上加载凭证，或希望传递静态凭证。	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

Name	描述	默认值	类型
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
deleteIfFiltered (consumer)	如果交换具有键 Sqs2Constants#SQS_DELETE_FILTERED (CamelAwsSqsDeleteFiltered)，则是否将 DeleteMessage 发送到 SQS 队列。	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
kmsDataKeyReusePeriodSeconds (consumer)	Amazon SQS 在再次调用 AWS KMS 之前，以便 Amazon SQS 可以重复使用或解密信息的时间长度（以秒为单位）。一个代表秒的整数，在 60 秒 (1 分钟) 和 86,400 秒 (24 小时)之间。默认：300 (5 分钟)。		整数
kmsMasterKeyId (consumer)	Amazon SQS 或自定义 CMK 的 AWS 管理的客户主密钥 (CMK) 的 ID。		字符串
maxMessagesPerPoll (consumer)	获取最大消息数，作为每次轮询轮询的限制。默认为没有限制，但使用 0 或负数将其禁用为无限。		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
serverSideEncryptionEnabled (consumer)	定义是否在队列中启用服务器端加密。	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

Name	描述	默认值	类型
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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
batchSeparator (producer)	在传递 String 以发送批处理消息操作时，设置分隔符。	,	字符串
delaySeconds (producer)	延迟发送消息的秒数。		整数
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
messageDeduplicationIdStrategy (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。 Enum 值： • useExchangeId • useContentBasedDeduplication	useExchangeId	字符串

Name	描述	默认值	类型
messageGroupIdStrategy (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。 Enum 值： • useConstant • useExchangeId • usePropertyValue		字符串
operation (producer)	当用户不想发送消息时，要执行的操作。 Enum 值： • sendBatchMessage • deleteMessage • listQueues • purgeQueue • deleteQueue		Sqs2Operations
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
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。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

Name	描述	默认值	类型
receiveMessageWaitTimeSeconds (queue)	如果您没有在请求中指定 WaitTimeSeconds，则使用 queue 属性 ReceiveMessageWaitTimeSeconds 来确定等待的时长。		整数
redrivePolicy (queue)	指定发送消息到 DeadLetter 队列的策略。请参阅 Amazon 文档的详情。		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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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
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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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

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。

11.6. 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。

11.7. 使用方法

11.7.1. 静态凭证和默认凭证提供程序

您可以通过指定 **useDefaultCredentialsProvider** 选项并将其设置为 true 来避免使用显式静态凭证。

- Java system properties - **aws.accessKeyId** and **aws.secretKey**
- 环境变量 - **AWS_ACCESS_KEY_ID** 和 **AWS_SECRET_ACCESS_KEY**。
- AWS STS 的 Web Identity Token。
- 共享凭证和配置文件。
- Amazon ECS 容器凭证 - 如果设置了环境变量 **AWS_CONTAINER_CREDENTIALS_RELATIVE_URI**，则从 Amazon ECS 加载。
- Amazon EC2 实例配置集凭据。

有关此信息的更多信息，您可以查看 AWS 凭证文档

11.7.2. 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 校验和。
CamelAwsSqsMessageId	**字符串**	Amazon SQS 消息 ID。
CamelAwsSqsDelaySeconds	**整数**	Amazon SQS 消息可看到的延迟秒数。

11.7.3. SQS 使用者设置的消息标头

标头	类型	描述
CamelAwsSqsMD5OfBody	**字符串**	Amazon SQS 消息的 MD5 校验和。
CamelAwsSqsMessageId	**字符串**	Amazon SQS 消息 ID。
CamelAwsSqsReceiptHandle	**字符串**	Amazon SQS 消息接收处理。
CamelAwsSqsMessageAttributes	**Map<String, String>**	Amazon SQS 消息属性。

11.7.4. 高级 AmazonSQS 配置

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

```
from("aws2-sqs://MyQueue?amazonSQSClient=#client&delay=5000&maxMessagesPerPoll=5")
.to("mock:result");
```

11.7.5. 创建或更新 SQS Queue

在 SQS 组件中，当启动端点时，将执行检查来获取队列是否存在的信息。您可以使用 **SQSConfiguration** 选项通过 **QueueAttributeName** 映射自定义创建。

```
from("aws2-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（并将 **autoCreateQueue** 选项设置为 true），它将使用 SQS 配置中的默认参数创建。如果已在 AWS 上启动，SQS 配置选项将用于覆盖现有的 AWS 配置。

11.7.6. DelayQueue VS Delay 用于单一消息

当选项 **delayQueue** 设为 true 时，SQS Queue 将是一个 **DelayQueue**，且 **DelaySeconds** 选项为 delay。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交换设置，不会影响队列中已存在的消息延迟。
- 对于 FIFO 队列，每个队列延迟设置重新交换设置会影响队列中已存在的消息的延迟。

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，而如果需要向所有消息添加固定延迟，则可以将此选项设置为 true。

11.7.7. 服务器端加密

队列有一组服务器端加密属性。相关选项包括 **serverSideEncryptionEnabled**,**keyMasterKeyId** 和 **kmsDataKeyReusePeriod**。SSE 默认被禁用。您需要明确将选项设置为 true，并将相关的参数设置为 queue 属性。

11.8. 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，它将在尝试将消息发送到其他消费者前等待可见性超时，除非收到 **DeleteMessage**。默认情况下，Camel 始终会在路由末尾发送 **DeleteMessage**，除非路由失败。要进行适当的过滤，即使成功完成路由，也不会发送 **DeleteMessage**，请使用 Filter:

```
from("aws2-sqs://MyQueue?
amazonSQSClient=#client&defaultVisibilityTimeout=5000&deleteIfFiltered=false&deleteAfterRead=false
)
.filter("${header.login} == true")
  .setProperty(Sqs2Constants.SQS_DELETE_FILTERED, constant(true))
  .to("mock:filter");
```

在上面的代码中，如果交换没有适当的标头，则不会通过过滤器 AND 将其从 SQS 队列中删除。在 5000 毫秒后，该消息对其他消费者可见。

请注意，我们必须将属性 **Sqs2Constants.SQS_DELETE_FILTERED** 设置为 **true**，以指示 Camel 发送 **DeleteMessage**（如果被过滤）。

11.9. 可用的 PRODUCER 操作

- 单个消息（默认）
- sendBatchMessage
- deleteMessage
- listQueues

11.10. 发送消息

您可以设置 **SendMessageBatchRequest** 或可 **Iterable**

```
from("direct:start")
  .setBody(constant("Camel rocks!"))
  .to("aws2-sqs://camel-1?accessKey=RAW(xxx)&secretKey=RAW(xxx)&region=eu-west-1");
```

11.11. 发送批处理消息

您可以设置 **SendMessageBatchRequest** 或可 **Iterable**

```
from("direct:start")
  .setHeader(SqsConstants.SQS_OPERATION, constant("sendBatchMessage"))
  .process(new Processor() {
     @Override
     public void process(Exchange exchange) throws Exception {
        Collection c = new ArrayList();
        c.add("team1");
        c.add("team2");
        c.add("team3");
        c.add("team4");
        exchange.getIn().setBody(c);
     }
  })
  .to("aws2-sqs://camel-1?accessKey=RAW(xxx)&secretKey=RAW(xxx)&region=eu-west-1");
```

因此，您将获得一个包含 **SendMessageBatchResponse** 实例的交换，您可以检查哪些消息是成功还是不成功。在批处理的每个消息上设置的 id 将是一个随机 UUID。

11.12. 删除单个消息

使用 **deleteMessage** 操作来删除单个消息。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("aws2-sqs://camel-1?accessKey=RAW(xxx)&secretKey=RAW(xxx)&region=eu-west-1");
```

因此，您将获得包含 **DeleteMessageResponse** 实例的交换，您可以使用它来检查消息是否被删除。

11.13. 列出队列

使用 **listQueues** 操作列出队列。

```
from("direct:start")
  .setHeader(SqsConstants.SQS_OPERATION, constant("listQueues"))
  .to("aws2-sqs://camel-1?accessKey=RAW(xxx)&secretKey=RAW(xxx)&region=eu-west-1");
```

因此，您将获得包含 **ListQueuesResponse** 实例的交换，您可以测试以检查实际队列。

11.14. 清除队列

使用 **purgeQueue** 操作清除队列。

```
from("direct:start")
  .setHeader(SqsConstants.SQS_OPERATION, constant("purgeQueue"))
  .to("aws2-sqs://camel-1?accessKey=RAW(xxx)&secretKey=RAW(xxx)&region=eu-west-1");
```

因此，您将获得包含 **PurgeQueueResponse** 实例的交换。

11.15. 队列自动创建

使用选项 **autoCreateQueue** 用户可以在 SQS Queue 不存在时避免自动创建。此选项的默认值是 **true**。如果设置为 false 任何对 AWS 中不存在的队列的操作，则不会成功，并返回错误。

11.16. 发送批处理消息和消息重复数据删除策略

如果您使用 **SendBatchMessage** 操作，您可以设置两种不同类型的消息重复数据删除策略： - useExchangeId - useContentBasedDeduplication

第一个将使用 **ExchangeIdMessageDeduplicationIdStrategy**，它将使用 Exchange ID 作为参数。另一个使用 **NullMessageDeduplicationIdStrategy**，它将使用正文作为 deduplication 元素。

如果发送批处理消息操作，您需要使用 **useContentBasedDeduplication** 和 Queue，在 Queue 中，您需要 启**用基于内容的 deduplication** 选项。

11.17. 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-aws2-sqs</artifactId>
    <version>${camel-version}</version>
</dependency>
```

其中 **{camel-version}** 必须替换为 Camel 的实际版本。

11.18. SPRING BOOT AUTO-CONFIGURATION

组件支持 44 选项，如下所列。

Name	描述	默认值	类型
camel.component.aws2-sqs.access-key	Amazon AWS 访问密钥.		字符串
camel.component.aws2-sqs.amazon-a-w-s-host	Amazon AWS 云的主机名。	amazonaws.com	字符串
camel.component.aws2-sqs.amazon-s-q-s-client	使用 AmazonSQS 作为客户端。选项是一个 software.amazon.awssdk.services.sqs.SqsClient 类型。		SqsClient
camel.component.aws2-sqs.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.aws2-sqs.auto-create-queue	设置队列的自动创建.	false	布尔值
camel.component.aws2-sqs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.aws2-sqs.batch-separator	在传递 String 以发送批处理消息操作时，设置分隔符。	,	字符串
camel.component.aws2-sqs.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.aws2-sqs.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

Name	描述	默认值	类型
camel.component.aws2-sqs.configuration	AWS SQS 默认配置。选项是 org.apache.camel.component.aws2.sqs.Sqs2Configuration 类型。		Sqs2Configuration
camel.component.aws2-sqs.default-visibility-timeout	默认可见性超时（以秒为单位）。		整数
camel.component.aws2-sqs.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.aws2-sqs.delay-seconds	延迟发送消息的秒数。		整数
camel.component.aws2-sqs.delete-after-read	读取后，从 SQS 删除消息。	true	布尔值
camel.component.aws2-sqs.delete-if-filtered	如果交换具有键 Sqs2Constants#SQS_DELETE_FILTERED (CamelAwsSqsDeleteFiltered)，则是否将 DeleteMessage 发送到 SQS 队列。	true	布尔值
camel.component.aws2-sqs.enabled	是否启用 aws2-sqs 组件的自动配置。这默认是启用的。		布尔值
camel.component.aws2-sqs.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.aws2-sqs.kms-data-key-reuse-period-seconds	Amazon SQS 在再次调用 AWS KMS 之前，以便 Amazon SQS 可以重复使用或解密信息的时间长度（以秒为单位）。一个代表秒的整数，在 60 秒 (1 分钟) 和 86,400 秒 (24 小时)之间。默认：300 (5 分钟)。		整数
camel.component.aws2-sqs.kms-master-key-id	Amazon SQS 或自定义 CMK 的 AWS 管理的客户主密钥 (CMK) 的 ID。		字符串

Name	描述	默认值	类型
camel.component.aws2-sqs.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.aws2-sqs.maximum-message-size	maximumMessageSize （以字节为单位）SQS 消息可以包含此队列。		整数
camel.component.aws2-sqs.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.aws2-sqs.message-deduplication-id-strategy	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	字符串
camel.component.aws2-sqs.message-group-id-strategy	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。		字符串
camel.component.aws2-sqs.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.aws2-sqs.operation	当用户不想发送消息时，要执行的操作。		Sqs2Operations
camel.component.aws2-sqs.override-endpoint	设置覆盖端点的需要。这个选项需要与 uriEndpointOverride 选项结合使用。	false	布尔值
camel.component.aws2-sqs.policy	此队列的策略。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

Name	描述	默认值	类型
camel.component.aws2-sqs.protocol	用于与 SQS 通信的底层协议。	https	字符串
camel.component.aws2-sqs.proxy-host	在实例化 SQS 客户端时定义代理主机。		字符串
camel.component.aws2-sqs.proxy-port	在实例化 SQS 客户端时定义代理端口。		整数
camel.component.aws2-sqs.proxy-protocol	在实例化 SQS 客户端时定义代理协议。		协议
camel.component.aws2-sqs.queue-owner-a-w-s-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.aws2-sqs.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.aws2-sqs.receive-message-wait-time-seconds	如果您没有在请求中指定 WaitTimeSeconds，则使用 queue 属性 ReceiveMessageWaitTimeSeconds 来确定等待的时长。		整数
camel.component.aws2-sqs.redrive-policy	指定发送消息到 DeadLetter 队列的策略。请参阅 Amazon 文档的详情。		字符串
camel.component.aws2-sqs.region	SQS 客户端需要工作的区域。使用此参数时，配置将预期区域（如 ap-east-1）的小写名称，您需要使用名称 Region.EU_WEST_1.id()。		字符串
camel.component.aws2-sqs.secret-key	Amazon AWS Secret 密钥。		字符串
camel.component.aws2-sqs.server-side-encryption-enabled	定义是否在队列中启用服务器端加密。	false	布尔值

Name	描述	默认值	类型
camel.component.aws2-sqs.trust-all-certificates	如果要在覆盖端点时信任所有证书。	false	布尔值
camel.component.aws2-sqs.uri-endpoint-override	设置覆盖 uri 端点。这个选项需要与 overrideEndpoint 选项结合使用。		字符串
camel.component.aws2-sqs.use-default-credentials-provider	设置 SQS 客户端是否应该预期在 AWS infra 实例上加载凭证，或希望传递静态凭证。	false	布尔值
camel.component.aws2-sqs.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.aws2-sqs.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

第 12 章 AZURE SERVICEBUS

自 Camel 3.12 起

支持生成者和消费者

集成 Azure ServiceBus 的 azure-servicebus 组件。Azure ServiceBus 是一个完全托管的企业集成消息代理。服务总线可以分离应用程序和服务。服务总线提供可靠、安全的平台，用于异步传输数据和状态。使用消息在不同的应用程序和服务间传输数据。

先决条件

您必须具有有效的 Windows Azure Storage 帐户。如需更多信息，请参阅 Azure 文档门户。

12.1. 依赖项

当在 Camel Spring Boot 中使用 **azure-servicebus** 时，请将以下 Maven 依赖项添加到 **pom.xml** 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-azure-servicebus-starter</artifactId>
</dependency>
```

12.2. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

12.2.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

12.2.2. 端点级别选项

在 **Endpoint 级别**，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 **路径和 查询参数**。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

12.3. 组件选项

Azure ServiceBus 组件支持 25 个选项，如下所列。

Name	描述	默认值	类型
amqpRetryOptions (common)	为服务总线客户端设置重试选项。如果没有指定，则使用默认重试选项。		AmqpRetryOptions
amqpTransportType (common)	设置与 Azure 服务总线发生的所有通信的传输类型。默认值为 AmqpTransportType#AMQP。 Enum 值： • Amqp • AmqpWebSockets	AMQP	AmqpTransportType
clientOptions (common)	将 ClientOptions 设置为从此构建器构建的客户端发送，启用自定义某些属性，并支持添加自定义标头信息。如需更多信息，请参阅 ClientOptions 文档。		ClientOptions
configuration (common)	组件配置。		ServiceBusConfiguration
proxyOptions (common)	设置用于 ServiceBusSenderAsyncClient 的代理配置。配置代理时，必须使用 AmqpTransportType#AMQP_WEB_SOCKETS 用于传输类型。		ProxyOptions
serviceBusType (common)	**需要** 要执行的服务总线类型。队列用于典型的队列选项，用于基于订阅的模型。 Enum 值： • queue • topic	queue	ServiceBusType
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
consumerOperation (consumer)	设置要在消费者中使用的所需操作。 Enum 值： • receiveMessages • peekMessages	receiveMessages	ServiceBusConsumerOperationDefinition
disableAutoComplete (consumer)	禁用接收消息的自动完成和自动取消。默认情况下，成功处理的消息为 \\{link ServiceBusReceiverAsyncClient#complete (ServiceBusReceivedMessage) completed}。如果在处理消息时出现错误，则为 \\{link ServiceBusReceiverAsyncClient#abandon (ServiceBusReceivedMessage) abandoned}。	false	布尔值
maxAutoLockRenewDuration (consumer)	设置继续自动续订锁定的时间长度。设置 Duration#ZERO 或 null 会禁用自动续订。对于 \\{link ServiceBusReceiveMode#RECEIVE_AND_DELETE_DELETE_DELETE} 模式，禁用自动续订。	5m	Duration
peekNumMaxMessages (consumer)	在 peek 操作过程中，将消息的最大数量设置为 peeked。		整数
prefetchCount (consumer)	设置接收方的预抓取计数。对于 \\{link ServiceBusReceiveMode#PEEK_LOCK PEEK_LOCK} 和 \\{link ServiceBusReceiveMode#RECEIVE_AND_DELETE_RECEIVE_AND_DELETE} 模式，默认值为 1。在应用程序请求使用 ServiceBusReceiverAsyncClient#receiveMessages（）时和之前，预抓取消息流会加快消息流。设置非零值将预先抓取该消息数。将值设为 0 可关闭预先抓取。		int
receiverAsyncClient (consumer)	**Autowired** 设置 receiverAsyncClient，以便供消费者使用消息。		ServiceBusReceiverAsyncClient
serviceBusReceiveMode (consumer)	设置接收方的接收模式。 Enum 值： • PEEK_LOCK • RECEIVE_AND_DELETE	PEEK_LOCK	ServiceBusReceiveMode

Name	描述	默认值	类型
subQueue (consumer)	设置要连接的 SubQueue 的类型。 Enum 值： • NONE • DEAD_LETTER_QUEUE • TRANSFER_DEAD_LETTER_QUEUE		SubQueue
subscriptionName (consumer)	设置主题中的订阅名称，以侦听至主题OrQueueName 和 serviceBusType=topic。如果 serviceBusType=topic 和消费者正在使用，则需要此属性。		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
producerOperation (producer)	设置要在制作者中使用的所需操作。 Enum 值： • sendMessages • scheduleMessages	sendMessages	ServiceBusProducerOperationDefinition
scheduledEnqueueTime (producer)	设置 OffsetDateTime，其中消息应出现在服务总线队列或主题中。		OffsetDateTime
senderAsyncClient (producer)	**Autowired** 设置要在制作者中使用的 SenderAsyncClient。		ServiceBusSenderAsyncClient
serviceBusTransactionContext (producer)	代表服务中的事务。此对象仅包含事务 ID。		ServiceBusTransactionContext
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
connectionString (security)	为服务总线命名空间或特定服务总线资源设置连接字符串。		字符串

Name	描述	默认值	类型
fullyQualifiedNamespace (security)	服务总线的完全限定命名空间。		字符串
tokenCredential (security)	Azure AD 身份验证的 TokenCredential，在 com.azure.identity 中实施。		TokenCredential

12.4. 端点选项

Azure ServiceBus 端点使用 URI 语法进行配置：

```
azure-servicebus:topicOrQueueName
```

使用以下路径和查询参数：

12.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
topicOrQueueName (common)	根据 serviceBusType 配置，所选的主题名称或队列名称。例如，如果 serviceBusType=queue，则这将是队列名称，如果 serviceBusType=topic，则这将是主题名称。		字符串

12.4.2. 查询参数(25 参数)

Name	描述	默认值	类型
amqpRetryOptions (common)	为服务总线客户端设置重试选项。如果没有指定，则使用默认重试选项。		AmqpRetryOptions
amqpTransportType (common)	设置与 Azure 服务总线发生的所有通信的传输类型。默认值为 AmqpTransportType#AMQP。 Enum 值： • Amqp • AmqpWebSockets	AMQP	AmqpTransportType
clientOptions (common)	将 ClientOptions 设置为从此构建器构建的客户端发送，启用自定义某些属性，并支持添加自定义标头信息。如需更多信息，请参阅 ClientOptions 文档。		ClientOptions

Name	描述	默认值	类型
proxyOptions (common)	设置用于 ServiceBusSenderAsyncClient 的代理配置。配置代理时，必须使用 AmqpTransportType#AMQP_WEB_SOCKETS 用于传输类型。		ProxyOptions
serviceBusType (common)	**需要** 要执行的服务总线类型。队列用于典型的队列选项，用于基于订阅的模型。 Enum 值： • queue • topic	queue	ServiceBusType
consumerOperation (consumer)	设置要在消费者中使用的所需操作。 Enum 值： • receiveMessages • peekMessages	receiveMessages	ServiceBusConsumerOperationDefinition
disableAutoComplete (consumer)	禁用接收消息的自动完成和自动取消。默认情况下，成功处理的消息为 \\{link ServiceBusReceiverAsyncClient#complete (ServiceBusReceivedMessage) completed}。如果在处理消息时出现错误，则为 \\{link ServiceBusReceiverAsyncClient#abandon (ServiceBusReceivedMessage) abandoned}。	false	布尔值
maxAutoLockRenewDuration (consumer)	设置继续自动续订锁定的时间长度。设置 Duration#ZERO 或 null 会禁用自动续订。对于 \\{link ServiceBusReceiveMode#RECEIVE_AND_DELETE_DELETE_DELETE} 模式，禁用自动续订。	5m	Duration
peekNumMaxMessages (consumer)	在 peek 操作过程中，将消息的最大数量设置为 peeked。		整数
prefetchCount (consumer)	设置接收方的预抓取计数。对于 \\{link ServiceBusReceiveMode#PEEK_LOCK PEEK_LOCK} 和 \\{link ServiceBusReceiveMode#RECEIVE_AND_DELETE_RECEIVE_AND_DELETE} 模式，默认值为 1。在应用程序请求使用 ServiceBusReceiverAsyncClient#receiveMessages（）时和之前，预抓取消息流会加快消息流。设置非零值将预先抓取该消息数。将值设为 0 可关闭预先抓取。		int

Name	描述	默认值	类型
receiverAsyncClient (consumer)	**Autowired** 设置 receiverAsyncClient，以便供消费者使用消息。		ServiceBusReceiverAsyncClient
serviceBusReceiveMode (consumer)	设置接收方的接收模式。 Enum 值： • PEEK_LOCK • RECEIVE_AND_DELETE	PEEK_LOCK	ServiceBusReceiveMode
subQueue (consumer)	设置要连接的 SubQueue 的类型。 Enum 值： • NONE • DEAD_LETTER_QUEUE • TRANSFER_DEAD_LETTER_QUEUE		SubQueue
subscriptionName (consumer)	设置主题中的订阅名称，以侦听至主题OrQueueName 和 serviceBusType=topic。如果 serviceBusType=topic 和消费者正在使用，则需要此属性。		字符串
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern

Name	描述	默认值	类型
producerOperation (producer)	设置要在制作者中使用的所需操作。 Enum 值： • sendMessages • scheduleMessages	sendMessages	ServiceBusProducerOperationDefinition
scheduledEnqueueTime (producer)	设置 OffsetDateTime，其中消息应出现在服务总线队列或主题中。		OffsetDateTime
senderAsyncClient (producer)	**Autowired** 设置要在制作者中使用的 SenderAsyncClient。		ServiceBusSenderAsyncClient
serviceBusTransactionContext (producer)	代表服务中的事务。此对象仅包含事务 ID。		ServiceBusTransactionContext
lazyStartProducer (producer (advanced))	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
connectionString (security)	为服务总线命名空间或特定服务总线资源设置连接字符串。		字符串
fullyQualifiedNamespace (security)	服务总线的完全限定命名空间。		字符串
tokenCredential (security)	Azure AD 身份验证的 TokenCredential，在 com.azure.identity 中实施。		TokenCredential

12.5. ASYNC CONSUMER 和 PRODUCER

这个组件实现了 async Consumer 和 producer。这允许 camel 路由在不阻塞任何线程的情况下异步使用和生成事件。

12.6. 消息标头

Azure ServiceBus 组件支持 25 个消息标头，如下所列：

Name	描述	默认值	类型
CamelAzureServiceBusApplicationProperties (common) 常量 ：APPLICATION_PROPERTIES	应用属性（也称为自定义属性）分别由生产者和消费者发送和接收的消息。		Map
CamelAzureServiceBusContentType (consumer) 常数：CONTENT_TYPE	获取消息的内容类型。		字符串
CamelAzureServiceBusCorrelationId (consumer) 常数：CORRELATION_ID	获取关联标识符。		字符串
CamelAzureServiceBusDeadLetterErrorDescription (consumer) 常量：DEAD_LETTER_ERROR_DESCRIPTION	获取已死信消息的描述。		字符串
CamelAzureServiceBusDeadLetterReason (consumer) 常数：DEAD_LETTER_REASON	获取消息死信的原因。		字符串

Name	描述	默认值	类型
CamelAzureServiceBusDeadLetterSource (consumer) 常量：DEAD_LETTER_SOURCE	获取此消息在死信之前排队的队列或订阅的名称。		字符串
CamelAzureServiceBusDeliveryCount (consumer) 常量 ：DELIVERY_COUNT	获取此消息发送到客户端的次数。		long
CamelAzureServiceBusEnqueuedSequenceNumber (consumer) constant: ENQUEUED_SEQUENCE_NUMBER	通过服务总线获取分配给邮件的 enqueued 序列号。		long
CamelAzureServiceBusEnqueuedTime (consumer) 恒定：ENQUEUED_TIME	获取此消息在 Azure 服务总线中排队的时间。		OffsetDateTime
CamelAzureServiceBusExpiresAt (consumer) 恒定：EXPIRES_AT	获取此消息将过期的日期。		OffsetDateTime
CamelAzureServiceBusLockToken (consumer) 常数：LOCK_TOKEN	获取当前消息的锁定令牌。		字符串

Name	描述	默认值	类型
CamelAzureServiceBusLockedUntil (consumer) 常数 ：LOCKED_UNTIL	获取此消息锁定的日期时间。		OffsetDateTime
CamelAzureServiceBusMessageId (consumer) 恒定：MESSAGE_ID	获取消息的标识符。		字符串
CamelAzureServiceBusPartitionKey (consumer) 常数：PARTITION_KEY	获取将消息发送到分区实体的分区密钥。		字符串
CamelAzureServiceBusRawAmqpMessage (consumer) 恒定：RAW_AMQP_MESSAGE	AMQP 协议定义的消息的表示。		AmqpAnnotatedMessage
CamelAzureServiceBusReplyTo (consumer) 常数 ：REPLY_TO	获取要发送回复的实体的地址。		字符串
CamelAzureServiceBusReplyToSessionId (consumer) 常量 ：REPLY_TO_SESSION_ID	获取或设置会话标识符，增加 ReplyTo 地址。		字符串
CamelAzureServiceBusSequenceNumber (consumer) constant: SEQUENCE_NUMBER	通过服务总线获取分配给消息的唯一编号。		long

Name	描述	默认值	类型
CamelAzureServiceBusSessionId (consumer) 常量 ：SESSION_ID	获取消息的会话 ID。		字符串
CamelAzureServiceBusSubject (consumer) 常数 ：SUBJ ECT	获取消息的主题。		字符串
CamelAzureServiceBusTimeToLive (consumer) 恒定：TIME_TO_LIVE	在此消息过期前获得持续时间。		Duration
CamelAzureServiceBusTo (consumer) 常数： TO	获取地址。		字符串
CamelAzureServiceBusScheduledEnqueueTime (common) 恒定：SCHEDULED_ENQUEUE_TIME	(producer)覆盖应出现在服务总线队列或主题中的 OffsetDateTime。(consumer)获取此消息调度的 enqueue 时间。		OffsetDateTime
CamelAzureServiceBusServiceBusTransactionContext (producer) 常量 ：SERVICE_BUS_TRANSACTION_CONTEXT	覆盖服务中的事务。此对象仅包含事务 ID。		ServiceBusTransactionContext

Name	描述	默认值	类型
CamelAzureServiceBusProducerOperation (producer) 恒定：PRODUCER_OPERATION	覆盖制作者中使用的所需操作。 Enum 值： • sendMessages • scheduleMessages		ServiceBusProducerOperationDefinition

12.6.1. 消息正文

在制作者中，此组件接受 **String** 类型的消息正文或 **List<String>** 来发送批处理消息。

在消费者中，返回的消息正文将类型为 'String。

12.6.2. Azure ServiceBus Producer 操作

操作	描述
sendMessages	使用批处理的方法将一组消息发送到服务总线队列或主题。
scheduleMessages	将调度的消息发送到此发送者所连接的 Azure 服务总线实体。调度的消息会被排队，并仅在调度的 enqueue 时间上提供给接收器。

12.6.3. Azure ServiceBus Consumer 操作

操作	描述
receiveMessages	从服务总线实体接收 infinite 消息流。
peekMessages	在不更改接收方或消息源的状态的情况下读取下一个活跃消息的批处理。

12.6.3.1. 例子

- **sendMessages**

```
from("direct:start")
  .process(exchange -> {
        final List<Object> inputBatch = new LinkedList<>();
```

```
            inputBatch.add("test batch 1");
            inputBatch.add("test batch 2");
            inputBatch.add("test batch 3");
            inputBatch.add(123456);

            exchange.getIn().setBody(inputBatch);
        })
    .to("azure-servicebus:test//?connectionString=test")
    .to("mock:result");
```

- **scheduleMessages**

```
from("direct:start")
    .process(exchange -> {
        final List<Object> inputBatch = new LinkedList<>();
            inputBatch.add("test batch 1");
            inputBatch.add("test batch 2");
            inputBatch.add("test batch 3");
            inputBatch.add(123456);

            exchange.getIn().setHeader(ServiceBusConstants.SCHEDULED_ENQUEUE_TIME,
OffsetDateTime.now());
            exchange.getIn().setBody(inputBatch);
        })
    .to("azure-servicebus:test//?connectionString=test&producerOperation=scheduleMessages")
    .to("mock:result");
```

- **receiveMessages**

```
from("azure-servicebus:test//?connectionString=test")
    .log("${body}")
    .to("mock:result");
```

- **peekMessages**

```
from("azure-servicebus:test//?
connectionString=test&consumerOperation=peekMessages&peekNumMaxMessages=3")
    .log("${body}")
    .to("mock:result");
```

12.7. 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.azure-servicebus.amqp-retry-options	为服务总线客户端设置重试选项。如果没有指定，则使用默认重试选项。选项是一个 com.azure.core.amqp.AmqpRetryOptions 类型。		AmqpRetryOptions

Name	描述	默认值	类型
camel.component.azure-servicebus.amqp-transport-type	设置与 Azure 服务总线发生的所有通信的传输类型。默认值为 AmqpTransportType#AMQP。		AmqpTransportType
camel.component.azure-servicebus.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.azure-servicebus.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.azure-servicebus.client-options	将 ClientOptions 设置为从此构建器构建的客户端发送，启用自定义某些属性，并支持添加自定义标头信息。如需更多信息，请参阅 ClientOptions 文档。选项是一个 com.azure.core.util.ClientOptions 类型。		ClientOptions
camel.component.azure-servicebus.configuration	组件配置。选项是 org.apache.camel.component.azure.servicebus.ServiceBusConfiguration 类型。		ServiceBusConfiguration
camel.component.azure-servicebus.connection-string	为服务总线命名空间或特定服务总线资源设置连接字符串。		字符串
camel.component.azure-servicebus.consumer-operation	设置要在消费者中使用的所需操作。		ServiceBusConsumerOperationDefinition
camel.component.azure-servicebus.disable-auto-complete	禁用接收消息的自动完成和自动取消。默认情况下，成功处理的消息为 \\{link ServiceBusReceiverAsyncClient#complete (ServiceBusReceivedMessage) completed}。如果在处理消息时出现错误，则为 \\{link ServiceBusReceiverAsyncClient#abandon (ServiceBusReceivedMessage) abandoned}。	false	布尔值

Name	描述	默认值	类型
camel.component.azure-servicebus.enabled	是否启用 azure-servicebus 组件的自动配置。这默认是启用的。		布尔值
camel.component.azure-servicebus.fully-qualified-namespace	服务总线的完全限定命名空间。		字符串
camel.component.azure-servicebus.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.azure-servicebus.max-auto-lock-renew-duration	设置继续自动续订锁定的时间长度。设置 Duration#ZERO 或 null 会禁用自动续订。对于 \\{link ServiceBusReceiveMode#RECEIVE_AND_DELETE_DELETE_DELETE} 模式，禁用自动续订。选项是一个 java.time.Duration 类型。		Duration
camel.component.azure-servicebus.peek-num-max-messages	在 peek 操作过程中，将消息的最大数量设置为 peeked。		整数
camel.component.azure-servicebus.prefetch-count	设置接收方的预抓取计数。对于 \\{link ServiceBusReceiveMode#PEEK_LOCK PEEK_LOCK} 和 \\{link ServiceBusReceiveMode#RECEIVE_AND_DELETE_RECEIVE_AND_DELETE} 模式，默认值为 1。在应用程序请求使用 ServiceBusReceiverAsyncClient#receiveMessages（）时和之前，预抓取消息流会加快消息流。设置非零值将预先抓取该消息数。将值设为 0 可关闭预先抓取。		整数
camel.component.azure-servicebus.producer-operation	设置要在制作者中使用的所需操作。		ServiceBusProducerOperationDefinition

Name	描述	默认值	类型
camel.component.azure-servicebus.proxy-options	设置用于 ServiceBusSenderAsyncClient 的代理配置。配置代理时，必须使用 AmqpTransportType#AMQP_WEB_SOCKETS 用于传输类型。选项是一个 com.azure.core.amqp.ProxyOptions 类型。		ProxyOptions
camel.component.azure-servicebus.receiver-async-client	设置 receiverAsyncClient，以便供消费者使用消息。选项是一个 com.azure.messaging.servicebus.ServiceBusReceiverAsyncClient 类型。		ServiceBusReceiverAsyncClient
camel.component.azure-servicebus.scheduled-enqueue-time	设置 OffsetDateTime，其中消息应出现在服务总线队列或主题中。选项是一个 java.time.OffsetDateTime 类型。		OffsetDateTime
camel.component.azure-servicebus.sender-async-client	设置 SenderAsyncClient，以便在制作者中使用。选项是一个 com.azure.messaging.servicebus.ServiceBusSenderAsyncClient 类型。		ServiceBusSenderAsyncClient
camel.component.azure-servicebus.service-bus-receive-mode	设置接收方的接收模式。		ServiceBusReceiveMode
camel.component.azure-servicebus.service-bus-transaction-context	代表服务中的事务。此对象仅包含事务 ID。选项是一个 com.azure.messaging.servicebus.ServiceBusTransactionContext 类型。		ServiceBusTransactionContext
camel.component.azure-servicebus.service-bus-type	要执行的连接的服务总线类型。队列用于典型的队列选项，用于基于订阅的模型。		ServiceBusType
camel.component.azure-servicebus.sub-queue	设置要连接的 SubQueue 的类型。		SubQueue
camel.component.azure-servicebus.subscription-name	设置主题中的订阅名称，以侦听至主题OrQueueName 和 serviceBusType=topic。如果 serviceBusType=topic 和消费者正在使用，则需要此属性。		字符串

Name	描述	默认值	类型
camel.component.azure-servicebus.token-credential	Azure AD 身份验证的 TokenCredential，在 com.azure.identity 中实施。选项是一个 com.azure.core.credential.TokenCredential 类型。		TokenCredential

第 13 章 AZURE STORAGE BLOB SERVICE

支持生成者和消费者

Azure Storage Blob 组件用于使用 Azure **API v12** 从 Azure Storage Blob Service 存储和检索 Blob。但是，对于 v12 以上版本，我们将了解此组件是否可以采用这些更改，具体取决于造成破坏的变化量。

先决条件

您必须具有有效的 Windows Azure Storage 帐户。如需更多信息，请参阅 Azure 文档门户。

13.1. 依赖项

当在 Camel Spring Boot 中使用 **azure-storage-blob** 时，请将以下 Maven 依赖项添加到 **pom.xml** 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-azure-storage-blob-starter</artifactId>
</dependency>
```

13.2. URI 格式

```
azure-storage-blob://accountName[/containerName][?options]
```

如果是消费者，则需要 **accountName**,**containerName**。对于生成者，它取决于请求的操作，例如，如果操作位于容器级别上，例如，只有 **createContainer**、**accountName** 和 **containerName**，则需要在 blob 级别中请求操作，如 **getBlob**、**accountName**、**containerName** 和 **blobName**。

如果 blob 不存在，则会创建它。您可以以以下格式将查询选项附加到 URI 中，

?options=value&option2=value&...

13.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

13.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

13.3.2. 端点级别选项

在 **Endpoint** 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 **路径和** 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

13.4. 组件选项

Azure Storage Blob Service 组件支持 35 选项，如下所列。

Name	描述	默认值	类型
blobName (common)	blob 名称，使用来自容器的特定 blob。但是，在生成者上，只有 blob 级别的操作才需要。		字符串
blobOffset (common)	为上传或下载操作设置 blob 偏移，默认为 0。	0	long
blobType (common)	blob 类型，为每个 blob 类型启动适当的设置。 Enum 值： • blockblob • appendblob • pageblob	blockblob	BlobType
closeStreamAfterRead (common)	在读取或保持打开后关闭流，默认为 true。	true	布尔值
configuration (common)	组件配置。		BlobConfiguration
credentials (common)	可以注入 StorageSharedKeyCredential 来创建 azure 客户端，其中包含重要的身份验证信息。		StorageSharedKeyCredential
CredentialType (common)	决定要采用的凭证策略。 Enum 值： • SHARED_ACCOUNT_KEY • SHARED_KEY_CREDENTIAL • AZURE_IDENTITY • AZURE_SAS	AZURE_IDENTITY	CredentialType

Name	描述	默认值	类型
dataCount (common)	范围中包含的字节数。如果指定，则必须大于或等于 0。		Long
fileDir (common)	下载的 Blob 将保存到的文件的目录，这可用于生成者和消费者。		字符串
maxResultsPerPage (common)	指定要返回的最大 Blob 数量，包括所有 BlobPrefix 元素。如果请求没有指定 maxResultsPerPage 或指定大于 5,000 个值，则服务器将最多返回 5,000 个项目。		整数
maxRetryRequests (common)	指定在从响应正文读取数据时进行的最大额外 HTTP Get 请求数。	0	int
prefix (common)	过滤结果，以仅返回名称以指定前缀开头的 Blob。可能为空，以返回所有 Blob。		字符串
regex (common)	过滤结果，以仅返回名称与指定正则表达式匹配的 Blob。如果同时设置了前缀和 regex，则可能为空返回，regex 采用优先级，并且忽略前缀。		字符串
sasToken (common)	使用共享访问签名时设置 SAS 令牌		字符串
serviceClient (common)	**Autowired** 客户端到存储帐户。此客户端不保存特定存储帐户的任何状态，而是便捷地将适当的请求发送到服务上的资源。它还可用于构建 URL 到 blob 和容器。此客户端包含服务帐户的操作。BlobContainerClient 到 BlobServiceClient#getBlobContainerClient (String) 上的操作在 BlobClient 上通过 BlobContainerClient=<getBlobClient (String)提供。		BlobServiceClient
timeout (common)	将引发 RuntimeException 之外的可选超时值。		Duration
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
blobSequenceNumber (producer)	用户控制的值可用于跟踪请求。序列号的值必须在 0 到 263 - 1 之间。默认值为 0。	0	Long

Name	描述	默认值	类型
blockListType (producer)	指定要返回的块类型。 Enum 值： • 已提交 • uncommitted • all	已提交	BlockListType
changeFeedContext (producer)	使用 getChangeFeed producer 操作时，这提供了在服务调用期间通过 Http 管道传递的其他上下文。		Context
changeFeedEndTime (producer)	使用 getChangeFeed producer 操作时，这会过滤结果，以大约在结束时间前返回事件。注意：也可以返回属于下一个小时的几个事件。可能缺少几小时属于此小时的几个事件；为了确保返回一小时的所有事件，请按小时向上舍入结束时间。		OffsetDateTime
changeFeedStartTime (producer)	使用 getChangeFeed producer 操作时，这会过滤结果，以大约在开始时间后返回事件。注意：也可以返回属于上一小时的几个事件。可能会缺少几个属于此小时的事件；为了确保返回一小时的所有事件，请按小时舍入开始时间。		OffsetDateTime
closeStreamAfterWrite (producer)	在写入或保持打开后关闭流，默认为 true。	true	布尔值
commitBlockListLater (producer)	当设置为 true 时，不会直接提交暂存块。	true	布尔值
createAppendBlob (producer)	当设置为 true 时，将在提交附加块时创建附加块。	true	布尔值
createPageBlob (producer)	当 设置为 true 时，上传页面 Blob 时将创建页面 blob。	true	布尔值
downloadLinkExpiration (producer)	覆盖 URL 下载链接的默认过期(millis)。		Long
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
operation (producer)	可与生成者上此组件的 Blob 操作。 Enum 值： • listBlobContainers • createBlobContainer • deleteBlobContainer • listBlobs • getBlob • deleteBlob • downloadBlobToFile • downloadLink • uploadBlockBlob • stageBlockBlobList • commitBlobBlockList • getBlobBlockList • createAppendBlob • commitAppendBlob • createPageBlob • uploadPageBlob • resizePageBlob • clearPageBlob • getPageBlobRanges	listBlobContainers	BlobOperationsDefinition
pageBlobSize (producer)	指定页面 blob 的最大大小，最多 8 TB。页面 Blob 大小必须与 512 字节边界一致。	512	Long
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
healthCheckConsumerEnabled (health)	用于从这个组件启用或禁用所有基于消费者的健康检查。	true	布尔值

Name	描述	默认值	类型
healthCheckProducerEnabled (health)	用于从此组件启用或禁用所有基于制作者的健康检查。注意：默认情况下，Camel 禁用了所有基于健康检查的制作者。您可以通过设置 camel.health.producersEnabled=true 来全局打开制作者检查。	true	布尔值
accessKey (security)	要使用 azure blob 服务进行身份验证的相关 azure 帐户名称的访问密钥。		字符串
sourceBlobAccessKey (security)	Source Blob Access Key：对于 copyblob 操作，我们需要为源 blob 有一个 accessKey，我们需要复制 Passing an accessKey 作为标头，因此我们可以设置为 key。		字符串

13.5. 端点选项

Azure Storage Blob Service 端点使用 URI 语法进行配置：

```
azure-storage-blob:accountName/containerName
```

使用以下路径和查询参数：

13.5.1. 路径参数(2 参数)

Name	描述	默认值	类型
accountName (common)	用于通过 azure blob 服务进行身份验证的 Azure 帐户名称。		字符串
containerName (common)	blob 容器名称。		字符串

13.5.2. 查询参数(50 参数)

Name	描述	默认值	类型
blobName (common)	blob 名称，使用来自容器的特定 blob。但是，在生成者上，只有 blob 级别的操作才需要。		字符串
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
blobServiceClient (common)	客户端到存储帐户。此客户端不保存特定存储帐户的任何状态，而是便捷地将适当的请求发送到服务上的资源。它还可用于构建 URL 到 blob 和容器。此客户端包含服务帐户的操作。容器上的操作通过 getBlobContainerClient (String)在 BlobContainerClient (String)上通过 getBlobContainerClient (String).getBlobClient (String)在 BlobClient 上提供操作。		BlobServiceClient
blobType (common)	blob 类型，为每个 blob 类型启动适当的设置。 Enum 值： • blockblob • appendblob • pageblob	blockblob	BlobType
closeStreamAfterRead (common)	在读取或保持打开后关闭流，默认为 true。	true	布尔值
credentials (common)	可以注入 StorageSharedKeyCredential 来创建 azure 客户端，其中包含重要的身份验证信息。		StorageSharedKeyCredential
CredentialType (common)	决定要采用的凭证策略。 Enum 值： • SHARED_ACCOUNT_KEY • SHARED_KEY_CREDENTIAL • AZURE_IDENTITY • AZURE_SAS	AZURE_IDENTITY	CredentialType
dataCount (common)	范围中包含的字节数。如果指定，则必须大于或等于 0。		Long
fileDir (common)	下载的 Blob 将保存到的文件的目录，这可用于生成者和消费者。		字符串
maxResultsPerPage (common)	指定要返回的最大 Blob 数量，包括所有 BlobPrefix 元素。如果请求没有指定 maxResultsPerPage 或指定大于 5,000 个值，则服务器将最多返回 5,000 个项目。		整数
maxRetryRequests (common)	指定在从响应正文读取数据时进行的最大额外 HTTP Get 请求数。	0	int

Name	描述	默认值	类型
prefix (common)	过滤结果，以仅返回名称以指定前缀开头的 Blob。可能为空，以返回所有 Blob。		字符串
regex (common)	过滤结果，以仅返回名称与指定正则表达式匹配的 Blob。如果同时设置了前缀和 regex，则可能为空返回，regex 采用优先级，并且忽略前缀。		字符串
sasToken (common)	使用共享访问签名时设置 SAS 令牌。		字符串
serviceClient (common)	**Autowired** 客户端到存储帐户。此客户端不保存特定存储帐户的任何状态，而是便捷地将适当的请求发送到服务上的资源。它还可用于构建 URL 到 blob 和容器。此客户端包含服务帐户的操作。BlobContainerClient 到 BlobServiceClient#getBlobContainerClient (String) 上的操作在 BlobClient 上通过 BlobContainerClient=<getBlobClient (String)提供。		BlobServiceClient
timeout (common)	将引发 RuntimeException 之外的可选超时值。		Duration
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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern

Name	描述	默认值	类型
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
blobSequenceNumber (producer)	用户控制的值可用于跟踪请求。序列号的值必须在 0 到 263 - 1 之间。默认值为 0。	0	Long
blockListType (producer)	指定要返回的块类型。 Enum 值： • 已提交 • uncommitted • all	已提交	BlockListType
changeFeedContext (producer)	使用 getChangeFeed producer 操作时，这提供了在服务调用期间通过 Http 管道传递的其他上下文。		Context
changeFeedEndTime (producer)	使用 getChangeFeed producer 操作时，这会过滤结果，以大约在结束时间前返回事件。注意：也可以返回属于下一个小时的几个事件。可能缺少几小时属于此小时的几个事件；为了确保返回一小时的所有事件，请按小时向上舍入结束时间。		OffsetDateTime
changeFeedStartTime (producer)	使用 getChangeFeed producer 操作时，这会过滤结果，以大约在开始时间后返回事件。注意：也可以返回属于上一小时的几个事件。可能会缺少几个属于此小时的事件；为了确保返回一小时的所有事件，请按小时舍入开始时间。		OffsetDateTime
closeStreamAfterWrite (producer)	在写入或保持打开后关闭流，默认为 true。	true	布尔值
commitBlockListLater (producer)	当设置为 true 时，不会直接提交暂存块。	true	布尔值
createAppendBlob (producer)	当设置为 true 时，将在提交附加块时创建附加块。	true	布尔值
createPageBlob (producer)	当 设置为 true 时，上传页面 Blob 时将创建页面 blob。	true	布尔值
downloadLinkExpiration (producer)	覆盖 URL 下载链接的默认过期(millis)。		Long

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
operation (producer)	可与生成者上此组件的 Blob 操作。 Enum 值： • listBlobContainers • createBlobContainer • deleteBlobContainer • listBlobs • getBlob • deleteBlob • downloadBlobToFile • downloadLink • uploadBlockBlob • stageBlockBlobList • commitBlobBlockList • getBlobBlockList • createAppendBlob • commitAppendBlob • createPageBlob • uploadPageBlob • resizePageBlob • clearPageBlob • getPageBlobRanges	listBlobContainers	BlobOperationsDefinition
pageBlobSize (producer)	指定页面 blob 的最大大小，最多 8 TB。页面 Blob 大小必须与 512 字节边界一致。	512	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

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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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
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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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)	要使用 azure blob 服务进行身份验证的相关 azure 帐户名称的访问密钥。		字符串
sourceBlobAccessKey (security)	Source Blob Access Key：对于 copyblob 操作，我们需要为源 blob 有一个 accessKey，我们需要复制 Passing an accessKey 作为标头，因此我们可以设置为 key。		字符串

所需信息选项

要使用此组件，您可以 3 个选项来提供所需的 Azure 身份验证信息：

- 提供一个链接： BlobServiceClient 实例，它可以注入到 blobServiceClient 中。注意：不需要创建特定的客户端，例如： **BlockBlobClient**，**BlobServiceClient** 代表可用于检索较低级别的客户端的上级。
- 在指定 **credentialsType=AZURE_IDENTITY** 并提供所需的环境变量 时，提供 Azure 身份。https://github.com/Azure/azure-sdk-for-java/tree/main/sdk/identity/azure-identity#environment-variables这可启用带有 secret/certificate 的服务主体（如应用程序注册）身份验证，以及用户名密码。请注意，这是默认的验证策略。
- 当指定 **credentialsType=SHARED_ACCOUNT_KEY** 并为 Azure 帐户提供 **accountName** 和 **accessKey** 时，提供共享存储帐户密钥，这是启动的最简单方法。**accessKey** 可以通过您的 Azure 门户生成。
- 在指定 **credentialType=SHARED_KEY_CREDENTIAL** 时提供共享存储帐户密钥，并提供 StorageSharedKeyCredential 实例，它可以注入到凭证选项中。

- 通过 Azure SAS，当指定 **credentialType=AZURE_SAS** 时，并通过 sasToken 参数提供 SAS Token 参数。

13.6. 使用方法

例如，若要从 **camelazure** 存储帐户中的 **container1** 上 block blob **hello.txt** 下载 blob 内容，请使用以下片断：

```
from("azure-storage-blob://camelazure/container1?
blobName=hello.txt&credentialType=SHARED_ACCOUNT_KEY&accessKey=RAW(yourAccessKey)"
).to("file://blobdirectory");
```

13.6.1. 消息标头

Azure Storage Blob Service 组件支持 63 个消息标头，如下所列：

标头	变量名称	类型	操作	描述
CamelAzureStorageBlobTimeout	**BlobConstants.TIMEOUT**	**Duration**	All	引发 {@link RuntimeException} 的一个可选超时值。
CamelAzureStorageBlobMetadata	**BlobConstants.METADATA**	**Map<String,String>**	与容器和 blob 相关的操作	与容器或 blob 关联的元数据。
CamelAzureStorageBlobPublicAccessType	**BlobConstants.PUBLIC_ACCESS_TYPE**	**PublicAccessType**	**createContainer**	指定此容器中的数据如何供公共使用。传递 **null**，没有公共访问。
CamelAzureStorageBlobRequestCondition	**BlobConstants.BLOB_REQUEST_CONDITION**	**BlobRequestConditions**	与容器和 blob 相关的操作	它包含将各种请求成功操作的值限制到存在的条件。这些条件完全是可选的。
CamelAzureStorageBlobListDetails	**BlobConstants.BLOB_LIST_DETAILS**	**BlobListDetails**	**listBlobs**	列出特定 blob 的详细信息
CamelAzureStorageBlobPrefix	**BlobConstants.PREFIX**	**字符串**	**listBlobs,getBlob**	过滤结果，以仅返回名称以指定前缀开头的 Blob。可能为空，以返回所有 Blob。

标头	变量名称	类型	操作	描述
CamelAzureStorageBlobMaxResultsPerPage	**BlobConstants.MAX_RESULTS_PER_PAGE**	**整数**	**listBlobs**	指定要返回的最大 Blob 数量，包括所有 BlobPrefix 元素。如果请求没有指定 maxResultsPerPage 或指定大于 5,000 个值，则服务器将最多返回 5,000 个项目。
CamelAzureStorageBlobListBlobOptions	**BlobConstants.LIST_BLOB_OPTIONS**	**ListBlobsOptions**	**listBlobs**	定义可用于在 {@link BlobContainerClient} 对象上配置对 listBlobsFlatSegment 的调用行为的选项。
CamelAzureStorageBlobHttpHeaders	**BlobConstants.BLOB_HTTP_HEADERS**	**BlobHttpHeaders**	**uploadBlockBlob**, **commitBlobBlockList**, **createAppendBlob**, **createPageBlob**	一组操作的附加参数。
CamelAzureStorageBlobAccessTier	**BlobConstants.ACCESS_TIER**	**AccessTier**	**uploadBlockBlob**, **commitBlobBlockList**	定义 AccessTier 的值。
CamelAzureStorageBlobContentMD5	**BlobConstants.CONTENT_MD5**	**byte[]**	与上传 blob 相关的大多数操作	块内容的 MD5 哈希。此哈希用于在传输过程中验证块的完整性。当指定此标头时，存储服务会将到达此标头值的内容哈希进行比较。请注意，这个 MD5 哈希没有存储在 blob 中。如果两个哈希不匹配，则操作将失败。

标头	变量名称	类型	操作	描述
CamelAzureStorageBlobPageBlobRange	**BlobConstants.PAGE_BLOB_RANGE**	**PageRange**	与页面 blob 相关的操作	{@link PageRange} 对象。如果页面必须与 512 字节边界一致，起始偏移必须是 512 的 modulus，并且最终偏移必须是 512 - 1 的 modulus。有效字节范围的示例为 0-511、512-1023 等。
CamelAzureStorageBlobCommitBlobBlockListLater	**BlobConstants.COMMIT_BLOCK_LIST_LATER**	**布尔值**	**stageBlockBlobList**	当设置为 **true** 时，不会直接提交暂存块。
CamelAzureStorageBlobCreateAppendBlob	**BlobConstants.CREATE_APPEND_BLOB**	**布尔值**	**commitAppendBlob**	当设置为 **true** 时，将在提交附加块时创建附加块。
CamelAzureStorageBlobCreatePageBlob	**BlobConstants.CREATE_PAGE_BLOB**	**布尔值**	**uploadPageBlob**	当 设置为 **true** 时，上传页面 Blob 时会创建页面 blob。
CamelAzureStorageBlobBlockListType	**BlobConstants.BLOCK_LIST_TYPE**	**BlockListType**	**getBlobBlockList**	指定要返回的块类型。
CamelAzureStorageBlobPageBlobSize	**BlobConstants.PAGE_BLOB_SIZE**	**Long**	**createPageBlob** , **resizePageBlob**	指定页面 blob 的最大大小，最多 8 TB。页面 Blob 大小必须与 512 字节边界一致。
CamelAzureStorageBlobSequenceNumber	**BlobConstants.BLOB_SEQUENCE_NUMBER**	**Long**	**createPageBlob**	用户控制的值可用于跟踪请求。序列号的值必须在 0 到 2^63 - 1 之间。默认值为 0。

标头	变量名称	类型	操作	描述
CamelAzureStorageBlobDeleteSnapshotsOptionType	**BlobConstants.DELETE_SNAPSHOT_OPTION_TYPE**	**DeleteSnapshotsOptionType**	**deleteBlob**	指定删除此 blob 上的快照的行为。\{@code Include} 将删除基础 blob 和所有快照。\{@code Only} 将只删除快照。如果删除了快照，则必须传递 null。
CamelAzureStorageBlobListBlobContainersOptions	**BlobConstants.LIST_BLOB_CONTAINERS_OPTIONS**	**ListBlobContainersOptions**	**listBlobContainers**	{@link ListBlobContainersOptions}，用于指定服务应返回哪些数据。
CamelAzureStorageBlobParallelTransferOptions	**BlobConstants.PARALLEL_TRANSFER_OPTIONS**	**ParallelTransferOptions**	**downloadBlobToFile**	{@link ParallelTransferOptions} 用于下载文件。忽略并行传输参数的数量。
CamelAzureStorageBlobFileDir	**BlobConstants.FILE_DIR**	**字符串**	**downloadBlobToFile**	下载的 Blob 将保存到的文件的目录。
CamelAzureStorageBlobDownloadLinkExpiration	**BlobConstants.DOWNLOAD_LINK_EXPIRATION**	**Long**	**downloadLink**	覆盖 URL 下载链接的默认过期 (millis)。
CamelAzureStorageBlobBlobName	**BlobConstants.BLOB_NAME**	**字符串**	与 blob 相关的操作	覆盖/设置交换标头上的 blob 名称。
CamelAzureStorageBlobContainerName	**BlobConstants.BLOB_CONTAINER_NAME**	**字符串**	与容器和 blob 相关的操作	覆盖/设置交换标头上的容器名称。
CamelAzureStorageBlobOperation	**BlobConstants.BLOB_OPERATION**	**BlobOperationsDefinition**	All	指定要执行的制作者操作，请参阅此页面上的与制作者操作相关的文档。

标头	变量名称	类型	操作	描述
CamelAzureStorageBlobRegex	**BlobConstants.REGEX**	**字符串**	**listBlobs,getBlob**	过滤结果，以仅返回名称与指定正则表达式匹配的 Blob。可以为空返回所有.如果设置了前缀和正则表达式，则 regex 将采用优先级和前缀，并忽略前缀。
CamelAzureStorageBlobChangeFeedStartTime	**BlobConstants.CHANGE_FEED_START_TIME**	**OffsetDateTime**	**getChangeFeed**	它过滤结果，以大约在开始时间后返回事件。注意：也可以返回属于上一小时的几个事件。可能会缺少几个属于此小时的事件；为了确保返回一小时的所有事件，请按小时舍入开始时间。
CamelAzureStorageBlobChangeFeedEndTime	**BlobConstants.CHANGE_FEED_END_TIME**	**OffsetDateTime**	**getChangeFeed**	它过滤结果，以大约在结束时间前返回事件。注意：也可以返回属于下一个小时的几个事件。可能缺少几小时属于此小时的几个事件；为了确保返回一小时的所有事件，请按小时向上舍入结束时间。
CamelAzureStorageBlobChangeFeedContext	**BlobConstants.CHANGE_FEED_CONTEXT**	**Context**	**getChangeFeed**	这提供了在服务调用期间通过 Http 管道传递的额外上下文。
CamelAzureStorageBlobSourceBlobAccountName	**BlobConstants.SOURCE_BLOB_ACCOUNT_NAME**	**字符串**	**copyBlob**	在复制 blob 操作中用作源帐户名称的源 blob 帐户名称
CamelAzureStorageBlobSourceBlobContainerName	**BlobConstants.SOURCE_BLOB_CONTAINER_NAME**	**字符串**	**copyBlob**	在复制 Blob 操作中用作源容器名称的源 blob 容器名称

13.6.2. 由组件制作者或消费者设置的消息标头

标头	变量名称	类型	描述
CamelAzureStorageBlobAccessTier	**BlobConstants.ACCESS_TIER**	**AccessTier**	blob 的访问层。
CamelAzureStorageBlobAccessTierChangeTime	**BlobConstants.ACCESS_TIER_CHANGE_TIME**	**OffsetDateTime**	当 blob 访问层最后一次更改时，日期时间。
CamelAzureStorageBlobArchiveStatus	**BlobConstants.ARCHIVE_STATUS**	**ArchiveStatus**	blob 的存档状态。
CamelAzureStorageBlobCreationTime	**BlobConstants.CREATION_TIME**	**OffsetDateTime**	blob 的创建时间。
CamelAzureStorageBlobSequenceNumber	**BlobConstants.BLOB_SEQUENCE_NUMBER**	**Long**	页面 blob 的当前序列号。
CamelAzureStorageBlobBlobSize	**BlobConstants.BLOB_SIZE**	**long**	blob 的大小。
CamelAzureStorageBlobBlobType	**BlobConstants.BLOB_TYPE**	**BlobType**	blob 的类型。
CamelAzureStorageBlobCacheControl	**BlobConstants.CACHE_CONTROL**	**字符串**	为 blob 指定的缓存控制。
CamelAzureStorageBlobCommittedBlockCount	**BlobConstants.COMMITTED_BLOCK_COUNT**	**整数**	提交到 append blob 的块数
CamelAzureStorageBlobContentDisposition	**BlobConstants.CONTENT_DISPOSITION**	**字符串**	为 blob 指定的内容分布。
CamelAzureStorageBlobContentEncoding	**BlobConstants.CONTENT_ENCODING**	**字符串**	为 blob 指定的内容编码。
CamelAzureStorageBlobContentLanguage	**BlobConstants.CONTENT_LANGUAGE**	**字符串**	为 blob 指定的内容语言。

标头	变量名称	类型	描述
CamelAzureStorageBlobContentMd5	**BlobConstants.CONTENT_MD5**	**byte[]**	为 blob 指定的内容 MD5。
CamelAzureStorageBlobContentType	**BlobConstants.CONTENT_TYPE**	**字符串**	为 blob 指定的内容类型。
CamelAzureStorageBlobCopyCompletionTime	**BlobConstants.COPY_COMPILATION_TIME**	**OffsetDateTime**	当 blob 完成的最后一个复制操作时，日期时间。
CamelAzureStorageBlobCopyDestinationSnapshot	**BlobConstants.COPY_DESTINATION_SNAPSHOT**	**字符串**	blob 的最后增量复制快照的快照标识符。
CamelAzureStorageBlobCopyId	**BlobConstants.COPY_ID**	**字符串**	在 blob 上执行的最后一个复制操作的标识符。
CamelAzureStorageBlobCopyProgress	**BlobConstants.COPY_PROGRESS**	**字符串**	在 blob 上执行的最后一个复制操作的进度。
CamelAzureStorageBlobCopySource	**BlobConstants.COPY_SOURCE**	**字符串**	在 blob 上执行的最后一个复制操作的来源。
CamelAzureStorageBlobCopyStatus	**BlobConstants.COPY_STATUS**	**CopyStatusType**	在 blob 上执行的最后一个复制操作的状态。
CamelAzureStorageBlobCopyStatusDescription	**BlobConstants.COPY_STATUS_DESCRIPTION**	**字符串**	blob 上最后一次复制操作的描述。
CamelAzureStorageBlobETag	**BlobConstants.E_TAG**	**字符串**	blob 的 E Tag
CamelAzureStorageBlobIsAccessTierInferred	**BlobConstants.IS_ACCESS_TIER_INFRRRED**	**布尔值**	表示 blob 的访问层是否从 blob 属性中推断出来的标志。
CamelAzureStorageBlobIsIncrementalCopy	**BlobConstants.IS_INCREMENTAL_COPY**	**布尔值**	表示 blob 是否已递增复制。

标头	变量名称	类型	描述
CamelAzureStorageBlobIsServerEncrypted	**BlobConstants.IS_SERVER_ENCRYPTED**	**布尔值**	表示在服务器中是否加密了 blob 内容的标志。
CamelAzureStorageBlobLastModified	**BlobConstants.LAST_MODIFIED**	**OffsetDateTime**	最后一次修改 Blob 时的日期。
CamelAzureStorageBlobLeaseDuration	**BlobConstants.LEASE_DURATION**	**LeaseDurationType**	blob 上的租期类型。
CamelAzureStorageBlobLeaseState	**BlobConstants.LEASE_STATE**	**LeaseStateType**	blob 上租期的状态。
CamelAzureStorageBlobLeaseStatus	**BlobConstants.LEASE_STATUS**	**LeaseStatusType**	blob 上租期的状态。
CamelAzureStorageBlobMetadata	**BlobConstants.METADATA**	**Map<String, String>**	与 blob 关联的其他元数据。
CamelAzureStorageBlobAppendOffset	**BlobConstants.APPEND_OFFSET**	**字符串**	块提交到块 blob 的偏移量。
CamelAzureStorageBlobFileName	**BlobConstants.FILE_NAME**	**字符串**	从操作 **downloadBlobToFile** 下载的文件名。
CamelAzureStorageBlobDownloadLink	**BlobConstants.DOWNLOAD_LINK**	**字符串**	由 **downloadLink** 操作生成的下载链接。
CamelAzureStorageBlobRawHttpHeaders	**BlobConstants.RAW_HTTP_HEADERS**	**httpHeaders**	返回可供用户使用的非稀疏 httpHeaders。

13.6.3. 高级 Azure Storage Blob 配置

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

```
StorageSharedKeyCredential credential = new StorageSharedKeyCredential("yourAccountName",
"yourAccessKey");
```

```
String uri = String.format("https://%s.blob.core.windows.net", "yourAccountName");

BlobServiceClient client = new BlobServiceClientBuilder()
                    .endpoint(uri)
                    .credential(credential)
                    .buildClient();
// This is camel context
context.getRegistry().bind("client", client);
```

然后，在 Camel **azure-storage-blob** 组件配置中引用此实例：

```
from("azure-storage-blob://cameldev/container1?blobName=myblob&serviceClient=#client")
.to("mock:result");
```

13.6.4. 在 registry 中自动检测 BlobServiceClient 客户端

组件可以检测在 registry 中存在 BlobServiceClient bean。如果这是该类型的唯一实例，它将用作客户端，您不必将其定义为 uri 参数，如上例所示。这对端点的智能配置非常有用。

13.6.5. Azure Storage Blob Producer 操作

Camel Azure Storage Blob 组件在生成者端提供广泛的操作：

服务级别的操作

对于这些操作，**需要 accountName**。

操作	描述
listBlobContainers	获取 blob 的内容。您可以将此操作的输出限制为 blob 范围。
getChangeFeed	返回发生到 blob 和存储帐户中的 blob 元数据的事务日志。更改源提供有排序、保证、持久、不可变、只读日志的这些更改。

容器级别的操作

对于这些操作，**需要 accountName** 和 **containerName**。

操作	描述
createBlobContainer	在存储帐户中创建新容器。如果容器已存在具有相同名称的容器，则生成者将忽略它。
deleteBlobContainer	删除存储帐户中的指定容器。如果容器不存在，则操作会失败。
listBlobs	返回此容器中的 blob 列表，并扁平化文件夹结构。

blob 级别的操作

对于这些操作，**需要 accountName**、**containerName** 和 **blobName**。

操作	blob Type	描述
getBlob	Common	获取 blob 的内容。您可以将此操作的输出限制为 blob 范围。
deleteBlob	Common	删除 blob。
downloadBlobToFile	Common	将整个 blob 下载到路径指定的文件中。如果文件已存在 {@link FileAlreadyExistsException}，则该文件必须不存在。
downloadLink	Common	使用共享访问签名(SAS)为指定的 blob 生成下载链接。默认情况下，这限制为 1 小时允许访问。但是，您可以通过标头覆盖默认的过期持续时间。
uploadBlockBlob	BlockBlob	创建新的块 blob，或更新现有块 blob 的内容。更新现有块 blob 覆盖 blob 上的任何现有元数据。PutBlob 不支持部分更新；现有 blob 的内容会被新内容覆盖。
stageBlockBlobList	**BlockBlob**	将指定的块上传到块 blob 的 "staging 区域"，以便稍后通过调用 commitBlobBlockList 提交。但是，如果标头 **CamelAzureStorageBlobCommitBlobBlockListLater** 或配置 **commitBlockListLater** 被设置为 false，这将在暂存块后立即提交块。
commitBlobBlockList	**BlockBlob**	通过指定要组成 blob 的块 ID 列表来写入 blob。为了写成 blob 的一部分，块必须已成功写入之前的 **stageBlockBlobList** 操作中的服务器。您可以通过只上传更改的块，然后提交新的和现有块来调用 **commitBlobBlockList** 来更新 blob。未在块列表中指定的任何块，并永久删除。
getBlobBlockList	**BlockBlob**	使用指定的块列表过滤器，返回已作为块 blob 一部分上传的块列表。
createAppendBlob	**AppendBlob**	创建一个 0-length 追加 blob。调用 commitAppendBlo'b 操作，将数据附加到附加 Blob。
commitAppendBlob	**AppendBlob**	将新数据块提交到现有附加 Blob 的末尾。如果标头 **CamelAzureStorageBlobCreateAppendBlob** 或配置 **createAppendBlob** 被设为 true，它将试图在提交前，首先通过内部调用 **createAppendBlob** 操作来创建 appendBlob。
createPageBlob	**PageBlob**	创建指定长度的页面 Blob。调用 **uploadPageBlob** 操作，将数据上传到页面 blob。

操作	blob Type	描述
uploadPageBlob	**PageBlob**	将一个或多个页面写入页面 blob。写入大小必须是 512 的倍数。如果标头 **CamelAzureStorageBlobCreatePageBlob** 或配置 **createPageBlob** 被设为 true，它将在上传前试图通过内部调用 **createPageBlob** 创建 appendBlob。
resizePageBlob	**PageBlob**	将页面 blob 大小调整为指定的大小（必须是 512 的倍数）。
clearPageBlob	**PageBlob**	从页面 blob 中释放指定的页面。范围的大小必须是 512 的倍数。
getPageBlobRanges	**PageBlob**	返回页面 blob 或页面 blob 的有效页面范围列表。
copyBlob	**Common**	将 blob 从一个容器复制到另一个容器，甚至从不同的帐户复制。

请参阅此页面中的示例部分，了解如何在 camel 应用程序中使用这些操作。

13.6.6. 消费者示例

要使用文件组件将 Blob 消耗到文件中，如下所示：

```
from("azure-storage-blob://camelazure/container1?
blobName=hello.txt&accountName=yourAccountName&accessKey=yourAccessKey").
to("file://blobdirectory");
```

但是，您也可以使用 file 组件直接写入文件，您需要指定 **fileDir** 文件夹路径，才能将 blob 保存到机器中。

```
from("azure-storage-blob://camelazure/container1?
blobName=hello.txt&accountName=yourAccountName&accessKey=yourAccessKey&fileDir=/var/to/awe
ome/dir").
to("mock:results");
```

另外，组件支持批处理消费者，因此您可以使用仅指定容器名称的多个 Blob，消费者将根据容器中的 Blob 数量返回多个交换。

示例

```
from("azure-storage-blob://camelazure/container1?
accountName=yourAccountName&accessKey=yourAccessKey&fileDir=/var/to/awesome/dir").
to("mock:results");
```

13.6.7. 制作者操作示例

- **listBlobContainers**

```
from("direct:start")
  .process(exchange -> {
    // set the header you want the producer to evaluate, refer to the previous
    // section to learn about the headers that can be set
    // e.g:
    exchange.getIn().setHeader(BlobConstants.LIST_BLOB_CONTAINERS_OPTIONS, new
ListBlobContainersOptions().setMaxResultsPerPage(10));
  })
  .to("azure-storage-blob://camelazure?operation=listBlobContainers&client&serviceClient=#client")
  .to("mock:result");
```

- **createBlobContainer**

```
from("direct:start")
  .process(exchange -> {
    // set the header you want the producer to evaluate, refer to the previous
    // section to learn about the headers that can be set
    // e.g:
    exchange.getIn().setHeader(BlobConstants.BLOB_CONTAINER_NAME, "newContainerName");
  })
  .to("azure-storage-blob://camelazure/container1?
operation=createBlobContainer&serviceClient=#client")
  .to("mock:result");
```

- **deleteBlobContainer**:

```
from("direct:start")
  .process(exchange -> {
    // set the header you want the producer to evaluate, refer to the previous
    // section to learn about the headers that can be set
    // e.g:
    exchange.getIn().setHeader(BlobConstants.BLOB_CONTAINER_NAME, "overridenName");
  })
  .to("azure-storage-blob://camelazure/container1?
operation=deleteBlobContainer&serviceClient=#client")
  .to("mock:result");
```

- **listBlobs**:

```
from("direct:start")
  .process(exchange -> {
    // set the header you want the producer to evaluate, refer to the previous
    // section to learn about the headers that can be set
    // e.g:
    exchange.getIn().setHeader(BlobConstants.BLOB_CONTAINER_NAME, "overridenName");
  })
  .to("azure-storage-blob://camelazure/container1?operation=listBlobs&serviceClient=#client")
  .to("mock:result");
```

- **getBlob**:

我们可以在交换正文中设置 **outputStream**，并将数据写入它。例如：

```
```

```
from("direct:start")
  .process(exchange -> {
    // set the header you want the producer to evaluate, refer to the previous
    // section to learn about the headers that can be set
    // e.g:
    exchange.getIn().setHeader(BlobConstants.BLOB_CONTAINER_NAME, "overridenName");

    // set our body
    exchange.getIn().setBody(outputStream);
  })
  .to("azure-storage-blob://camelazure/container1?
blobName=blob&operation=getBlob&serviceClient=#client")
  .to("mock:result");
```

如果没有设置正文，则此操作将为我们提供一个 **InputStream** 实例，该实例可以继续进一步的下游实例：

```
from("direct:start")
  .to("azure-storage-blob://camelazure/container1?
blobName=blob&operation=getBlob&serviceClient=#client")
  .process(exchange -> {
      InputStream inputStream = exchange.getMessage().getBody(InputStream.class);
      // We use Apache common IO for simplicity, but you are free to do whatever dealing
      // with inputStream
      System.out.println(IOUtils.toString(inputStream, StandardCharsets.UTF_8.name()));
  })
  .to("mock:result");
```

- **deleteBlob**:

```
from("direct:start")
  .process(exchange -> {
    // set the header you want the producer to evaluate, refer to the previous
    // section to learn about the headers that can be set
    // e.g:
    exchange.getIn().setHeader(BlobConstants.BLOB_NAME, "overridenName");
  })
  .to("azure-storage-blob://camelazure/container1?
blobName=blob&operation=deleteBlob&serviceClient=#client")
  .to("mock:result");
```

- **downloadBlobToFile** ：

```
from("direct:start")
  .process(exchange -> {
    // set the header you want the producer to evaluate, refer to the previous
    // section to learn about the headers that can be set
    // e.g:
    exchange.getIn().setHeader(BlobConstants.BLOB_NAME, "overridenName");
  })
  .to("azure-storage-blob://camelazure/container1?
blobName=blob&operation=downloadBlobToFile&fileDir=/var/mydir&serviceClient=#client")
  .to("mock:result");
```

- **downloadLink**

```
from("direct:start")
  .to("azure-storage-blob://camelazure/container1?
blobName=blob&operation=downloadLink&serviceClient=#client")
  .process(exchange -> {
      String link = exchange.getMessage().getHeader(BlobConstants.DOWNLOAD_LINK,
String.class);
      System.out.println("My link " + link);
  })
  .to("mock:result");
```

- **uploadBlockBlob**

```
from("direct:start")
  .process(exchange -> {
    // set the header you want the producer to evaluate, refer to the previous
    // section to learn about the headers that can be set
    // e.g:
    exchange.getIn().setHeader(BlobConstants.BLOB_NAME, "overridenName");
    exchange.getIn().setBody("Block Blob");
  })
  .to("azure-storage-blob://camelazure/container1?
blobName=blob&operation=uploadBlockBlob&serviceClient=#client")
  .to("mock:result");
```

- **stageBlockBlobList**

```
from("direct:start")
  .process(exchange -> {
      final List<BlobBlock> blocks = new LinkedList<>();
      blocks.add(BlobBlock.createBlobBlock(new ByteArrayInputStream("Hello".getBytes())));
      blocks.add(BlobBlock.createBlobBlock(new ByteArrayInputStream("From".getBytes())));
      blocks.add(BlobBlock.createBlobBlock(new ByteArrayInputStream("Camel".getBytes())));

      exchange.getIn().setBody(blocks);
  })
  .to("azure-storage-blob://camelazure/container1?
blobName=blob&operation=stageBlockBlobList&serviceClient=#client")
  .to("mock:result");
```

- **commitBlockBlobList**

```
from("direct:start")
  .process(exchange -> {
      // We assume here you have the knowledge of these blocks you want to commit
      final List<Block> blocksIds = new LinkedList<>();
      blocksIds.add(new Block().setName("id-1"));
      blocksIds.add(new Block().setName("id-2"));
      blocksIds.add(new Block().setName("id-3"));

      exchange.getIn().setBody(blocksIds);
  })
```

```
  .to("azure-storage-blob://camelazure/container1?
blobName=blob&operation=commitBlockBlobList&serviceClient=#client")
  .to("mock:result");
```

- **getBlobBlockList**

```
from("direct:start")
  .to("azure-storage-blob://camelazure/container1?
blobName=blob&operation=getBlobBlockList&serviceClient=#client")
  .log("${body}")
  .to("mock:result");
```

- **createAppendBlob**

```
from("direct:start")
  .to("azure-storage-blob://camelazure/container1?
blobName=blob&operation=createAppendBlob&serviceClient=#client")
  .to("mock:result");
```

- **commitAppendBlob**

```
from("direct:start")
  .process(exchange -> {
    final String data = "Hello world from my awesome tests!";
    final InputStream dataStream = new
ByteArrayInputStream(data.getBytes(StandardCharsets.UTF_8));

    exchange.getIn().setBody(dataStream);

    // of course you can set whatever headers you like, refer to the headers section to learn more
  })
  .to("azure-storage-blob://camelazure/container1?
blobName=blob&operation=commitAppendBlob&serviceClient=#client")
  .to("mock:result");
```

- **createPageBlob**

```
from("direct:start")
  .to("azure-storage-blob://camelazure/container1?
blobName=blob&operation=createPageBlob&serviceClient=#client")
  .to("mock:result");
```

- **uploadPageBlob**

```
from("direct:start")
  .process(exchange -> {
    byte[] dataBytes = new byte[512]; // we set range for the page from 0-511
    new Random().nextBytes(dataBytes);
    final InputStream dataStream = new ByteArrayInputStream(dataBytes);
    final PageRange pageRange = new PageRange().setStart(0).setEnd(511);

    exchange.getIn().setHeader(BlobConstants.PAGE_BLOB_RANGE, pageRange);
    exchange.getIn().setBody(dataStream);
  })
```

```
  .to("azure-storage-blob://camelazure/container1?
blobName=blob&operation=uploadPageBlob&serviceClient=#client")
  .to("mock:result");
```

- **resizePageBlob**

```
from("direct:start")
  .process(exchange -> {
    final PageRange pageRange = new PageRange().setStart(0).setEnd(511);

    exchange.getIn().setHeader(BlobConstants.PAGE_BLOB_RANGE, pageRange);
  })
  .to("azure-storage-blob://camelazure/container1?
blobName=blob&operation=resizePageBlob&serviceClient=#client")
  .to("mock:result");
```

- **clearPageBlob**

```
from("direct:start")
  .process(exchange -> {
    final PageRange pageRange = new PageRange().setStart(0).setEnd(511);

    exchange.getIn().setHeader(BlobConstants.PAGE_BLOB_RANGE, pageRange);
  })
  .to("azure-storage-blob://camelazure/container1?
blobName=blob&operation=clearPageBlob&serviceClient=#client")
  .to("mock:result");
```

- **getPageBlobRanges**

```
from("direct:start")
  .process(exchange -> {
    final PageRange pageRange = new PageRange().setStart(0).setEnd(511);

    exchange.getIn().setHeader(BlobConstants.PAGE_BLOB_RANGE, pageRange);
  })
  .to("azure-storage-blob://camelazure/container1?
blobName=blob&operation=getPageBlobRanges&serviceClient=#client")
  .log("${body}")
  .to("mock:result");
```

- **copyBlob**

```
from("direct:copyBlob")
  .process(exchange -> {
    exchange.getIn().setHeader(BlobConstants.BLOB_NAME, "file.txt");
    exchange.getMessage().setHeader(BlobConstants.SOURCE_BLOB_CONTAINER_NAME,
"containerblob1");
    exchange.getMessage().setHeader(BlobConstants.SOURCE_BLOB_ACCOUNT_NAME,
"account");
  })
  .to("azure-storage-blob://account/containerblob2?
operation=copyBlob&sourceBlobAccessKey=RAW(accessKey)")
  .to("mock:result");
```

这样，帐户"account"的容器 containerblob1 中的 file.txt 将复制到同一帐户的 containerblob2 中。

13.6.8. SAS Token 生成示例

SAS Blob 容器令牌可以通过编程方式或通过 Azure UI 生成。要使用 java 代码生成令牌，可以执行以下操作：

```
BlobContainerClient blobClient = new BlobContainerClientBuilder()
        .endpoint(String.format("https://%s.blob.core.windows.net/%s", accountName, accessKey))
        .containerName(containerName)
        .credential(new StorageSharedKeyCredential(accountName, accessKey))
        .buildClient();

    // Create a SAS token that's valid for 1 day, as an example
    OffsetDateTime expiryTime = OffsetDateTime.now().plusDays(1);

    // Assign permissions to the SAS token
    BlobContainerSasPermission blobContainerSasPermission = new
BlobContainerSasPermission()
        .setWritePermission(true)
        .setListPermission(true)
        .setCreatePermission(true)
        .setDeletePermission(true)
        .setAddPermission(true)
        .setReadPermission(true);

    BlobServiceSasSignatureValues sasSignatureValues = new
BlobServiceSasSignatureValues(expiryTime, blobContainerSasPermission);

    return blobClient.generateSas(sasSignatureValues);
```

然后，生成的 SAS 令牌可以存储在 **application.properties** 文件中，以便 camel 路由可以加载该文件，例如：

```
camel.component.azure-storage-blob.sas-token=MY_TOKEN_HERE

from("direct:copyBlob")
  .to("azure-storage-blob://account/containerblob2?
operation=uploadBlockBlob&credentialType=AZURE_SAS")
```

13.7. SPRING BOOT AUTO-CONFIGURATION

组件支持 36 个选项，如下所列。

Name	描述	默认值	类型
camel.component.azure-storage-blob.access-key	要使用 azure blob 服务进行身份验证的相关 azure 帐户名称的访问密钥。		字符串

Name	描述	默认值	类型
camel.component.azure-storage-blob.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.azure-storage-blob.blob-name	blob 名称，使用来自容器的特定 blob。但是，在生成者上，只有 blob 级别的操作才需要。		字符串
camel.component.azure-storage-blob.blob-offset	为上传或下载操作设置 blob 偏移，默认为 0。	0	Long
camel.component.azure-storage-blob.blob-sequence-number	用户控制的值可用于跟踪请求。序列号的值必须在 0 到 263 - 1 之间。默认值为 0。	0	Long
camel.component.azure-storage-blob.blob-type	blob 类型，为每个 blob 类型启动适当的设置。		BlobType
camel.component.azure-storage-blob.block-list-type	指定要返回的块类型。		BlockListType
camel.component.azure-storage-blob.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.azure-storage-blob.change-feed-context	使用 getChangeFeed producer 操作时，这提供了在服务调用期间通过 Http 管道传递的其他上下文。选项是一个 com.azure.core.util.Context 类型。		Context
camel.component.azure-storage-blob.change-feed-end-time	使用 getChangeFeed producer 操作时，这会过滤结果，以大约在结束时间前返回事件。注意：也可以返回属于下一个小时的几个事件。可能缺少几小时属于此小时的几个事件；为了确保返回一小时的所有事件，请按小时向上舍入结束时间。选项是一个 java.time.OffsetDateTime 类型。		OffsetDateTime

Name	描述	默认值	类型
camel.component.azure-storage-blob.change-feed-start-time	使用 getChangeFeed producer 操作时，这会过滤结果，以大约在开始时间后返回事件。注意：也可以返回属于上一小时的几个事件。可能会缺少几个属于此小时的事件；为了确保返回一小时的所有事件，请按小时舍入开始时间。选项是一个 java.time.OffsetDateTime 类型。		OffsetDateTime
camel.component.azure-storage-blob.close-stream-after-read	在读取或保持打开后关闭流，默认为 true。	true	布尔值
camel.component.azure-storage-blob.close-stream-after-write	在写入或保持打开后关闭流，默认为 true。	true	布尔值
camel.component.azure-storage-blob.commit-block-list-later	当设置为 true 时，不会直接提交暂存块。	true	布尔值
camel.component.azure-storage-blob.configuration	组件配置。选项是 org.apache.camel.component.azure.storage.blob.BlobConfiguration 类型。		BlobConfiguration
camel.component.azure-storage-blob.create-append-blob	当设置为 true 时，将在提交附加块时创建附加块。	true	布尔值
camel.component.azure-storage-blob.create-page-blob	当 设置为 true 时，上传页面 Blob 时将创建页面 blob。	true	布尔值
camel.component.azure-storage-blob.credential-type	决定要采用的凭证策略。		CredentialType
camel.component.azure-storage-blob.credentials	可以注入 StorageSharedKeyCredential 来创建 azure 客户端，其中包含重要的身份验证信息。选项是一个 com.azure.storage.common.StorageSharedKeyCredential 类型。		StorageSharedKeyCredential

Name	描述	默认值	类型
camel.component.azure-storage-blob.data-count	范围中包含的字节数。如果指定，则必须大于或等于 0。		Long
camel.component.azure-storage-blob.download-link-expiration	覆盖 URL 下载链接的默认过期(millis)。		Long
camel.component.azure-storage-blob.enabled	是否启用 azure-storage-blob 组件的自动配置。这默认是启用的。		布尔值
camel.component.azure-storage-blob.file-dir	下载的 Blob 将保存到的文件的目录，这可用于生成者和消费者。		字符串
camel.component.azure-storage-blob.health-check-consumer-enabled	用于从这个组件启用或禁用所有基于消费者的健康检查。	true	布尔值
camel.component.azure-storage-blob.health-check-producer-enabled	用于从此组件启用或禁用所有基于制作者的健康检查。注意：默认情况下，Camel 禁用了所有基于健康检查的制作者。您可以通过设置 camel.health.producersEnabled=true 来全局打开制作者检查。	true	布尔值
camel.component.azure-storage-blob.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.azure-storage-blob.max-results-per-page	指定要返回的最大 Blob 数量，包括所有 BlobPrefix 元素。如果请求没有指定 maxResultsPerPage 或指定大于 5,000 个值，则服务器将最多返回 5,000 个项目。		整数
camel.component.azure-storage-blob.max-retry-requests	指定在从响应正文读取数据时进行的最大额外 HTTP Get 请求数。	0	整数

Name	描述	默认值	类型
camel.component.azure-storage-blob.operation	可与生成者上此组件的 Blob 操作。		BlobOperationsDefinition
camel.component.azure-storage-blob.page-blob-size	指定页面 blob 的最大大小，最多 8 TB。页面 Blob 大小必须与 512 字节边界一致。	512	Long
camel.component.azure-storage-blob.prefix	过滤结果，以仅返回名称以指定前缀开头的 Blob。可能为空，以返回所有 Blob。		字符串
camel.component.azure-storage-blob.regex	过滤结果，以仅返回名称与指定正则表达式匹配的 Blob。如果同时设置了前缀和 regex，则可能为空返回，regex 采用优先级，并且忽略前缀。		字符串
camel.component.azure-storage-blob.sas-token	使用共享访问签名时设置 SAS 令牌		字符串
camel.component.azure-storage-blob.service-client	客户端到存储帐户。此客户端不保存特定存储帐户的任何状态，而是便捷地将适当的请求发送到服务上的资源。它还可用于构建 URL 到 blob 和容器。此客户端包含服务帐户的操作。BlobContainerClient 到 BlobServiceClient#getBlobContainerClient (String) 上的操作在 BlobClient 上通过 BlobContainerClient=<getBlobClient (String)提供。选项是一个 com.azure.storage.blob.BlobServiceClient 类型。		BlobServiceClient
camel.component.azure-storage-blob.source-blob-access-key	Source Blob Access Key：对于 copyblob 操作，我们需要为源 blob 有一个 accessKey，我们需要复制 Passing an accessKey 作为标头，因此我们可以设置为 key。		字符串
camel.component.azure-storage-blob.timeout	将引发 RuntimeException 之外的可选超时值。选项是一个 java.time.Duration 类型。		Duration

第 14 章 AZURE STORAGE QUEUE SERVICE

支持生成者和消费者

Azure Storage Queue 组件支持使用 **Azure APIs v12**，将信息存储和检索到 Azure Storage Queue 服务。但是，对于 v12 以上版本，我们将了解此组件是否可以采用这些更改，具体取决于造成破坏的变化量。

先决条件

您必须具有有效的 Windows Azure Storage 帐户。如需更多信息，请参阅 Azure 文档门户。

14.1. 依赖项

当在 Camel Spring Boot 中使用 **azure-storage-queue** 时，请将以下 Maven 依赖项添加到 **pom.xml** 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-azure-storage-queue-starter</artifactId>
</dependency>
```

14.2. URI 格式

```
azure-storage-queue://accountName[/queueName][?options]
```

如果是消费者，则需要 accountName 和 queueName。如果生成者不同，它取决于所请求的操作，例如，如果操作位于服务级别上，则仅需要 accountName，但在队列级别上请求操作时（如 createQueue、sendMessage.. 等），则需要 accountName 和 queueName。

如果队列尚不存在，则会创建队列。您可以以以下格式将查询选项附加到 URI 中，

?options=value&option2=value&...

14.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

14.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

14.3.2. 端点级别选项

在 **Endpoint** 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 **路径和** 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

14.4. 组件选项

Azure Storage Queue Service 组件支持 15 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	组件配置。		QueueConfiguration
serviceClient (common)	**Autowired** Service 客户端到存储帐户，以便与队列服务交互。此客户端不保存特定存储帐户的任何状态，而是便捷地将适当的请求发送到服务上的资源。此客户端包含与 Azure Storage 中队列帐户交互的所有操作。客户端允许的操作是创建、列出和删除队列、检索和更新帐户属性，以及检索帐户统计信息。		QueueServiceClient
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
createQueue (producer)	当设置为 true 时，将在发送消息到队列时自动创建队列。	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

Name	描述	默认值	类型
operation (producer)	队列服务操作提示到制作者。 Enum 值： • listQueues • createQueue • deleteQueue • clearQueue • sendMessage • deleteMessage • receiveMessages • peekMessages • updateMessage		QueueOperationDefinition
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
maxMessages (queue)	如果要获取的最大消息数，如果队列中存在的消息数量少于请求的所有消息，则返回。如果仅检索 1 个消息，则允许的范围为 1 到 32 个消息。	1	整数
messageId (queue)	要删除或更新的消息 ID。		字符串
popReceipt (queue)	必须匹配删除或更新消息的唯一标识符。		字符串
Timeout (queue)	应用到操作的可选超时。如果在超时结束前没有返回响应，则将抛出 RuntimeException。		Duration
timeToLive (queue)	消息在队列中保持活动状态的时长。如果未设置该值将默认为 7 天，如果传递 -1，则消息不会过期。生存时间必须是 -1 或任意正数。格式应采用以下形式：PnDTnHnMn.nS.，例如：PT20.345Scriu-osgiparses 为 20.345 秒，P2Dcategories- netobservparses 为 2 天，如果您使用的是 EndpointDsl/ComponentDsl，您可以执行类似于 Duration.ofSeconds （）的操作，因为这些 Java API typesafe。		Duration

Name	描述	默认值	类型
visibilityTimeout (queue)	消息在队列中不可见的超时时间。超时必须在 1 秒和 7 天之间。格式应采用以下形式：PnDTnHnMn.nS.，例如：PT20.345Scriu-osgiparses 为 20.345 秒，P2Dcategories- netobservparses 为 2 天，如果您使用的是 EndpointDsl/ComponentDsl，您可以执行类似于 Duration.ofSeconds （）的操作，因为这些 Java API typesafe。		Duration
accessKey (security)	要使用 azure 队列服务进行身份验证的相关 azure 帐户名称的访问密钥。		字符串
凭证 （安全）	可以注入 StorageSharedKeyCredential 来创建 azure 客户端，其中包含重要的身份验证信息。		StorageSharedKeyCredential

14.5. 端点选项

Azure Storage Queue Service 端点使用 URI 语法进行配置：

```
azure-storage-queue:accountName/queueName
```

使用以下路径和查询参数：

14.5.1. 路径参数(2 参数)

Name	描述	默认值	类型
accountName (common)	用于通过 azure 队列服务进行身份验证的 Azure 帐户名称。		字符串
queueName (common)	队列资源名称。		字符串

14.5.2. 查询参数(31 参数)

Name	描述	默认值	类型
serviceClient (common)	**Autowired** Service 客户端到存储帐户，以便与队列服务交互。此客户端不保存特定存储帐户的任何状态，而是便捷地将适当的请求发送到服务上的资源。此客户端包含与 Azure Storage 中队列帐户交互的所有操作。客户端允许的操作是创建、列出和删除队列、检索和更新帐户属性，以及检索帐户统计信息。		QueueServiceClient

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
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
createQueue (producer)	当设置为 true 时，将在发送消息到队列时自动创建队列。	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

Name	描述	默认值	类型
operation (producer)	队列服务操作提示到制作者。 Enum 值： • listQueues • createQueue • deleteQueue • clearQueue • sendMessage • deleteMessage • receiveMessages • peekMessages • updateMessage		QueueOperationDefinition
maxMessages (queue)	如果要获取的最大消息数，如果队列中存在的消息数量少于请求的所有消息，则返回。如果仅检索 1 个消息，则允许的范围为 1 到 32 个消息。	1	整数
messageId (queue)	要删除或更新的消息 ID。		字符串
popReceipt (queue)	必须匹配删除或更新消息的唯一标识符。		字符串
Timeout (queue)	应用到操作的可选超时。如果在超时结束前没有返回响应，则将抛出 RuntimeException。		Duration
timeToLive (queue)	消息在队列中保持活动状态的时长。如果未设置该值将默认为 7 天，如果传递 -1，则消息不会过期。生存时间必须是 -1 或任意正数。格式应采用以下形式：PnDTnHnMn.nS.，例如：PT20.345Scriu-osgiparses 为 20.345 秒，P2Dcategories- netobservparses 为 2 天，如果您使用的是 EndpointDsl/ComponentDsl，您可以执行类似于 Duration.ofSeconds （）的操作，因为这些 Java API typesafe。		Duration
visibilityTimeout (queue)	消息在队列中不可见的超时时间。超时必须在 1 秒和 7 天之间。格式应采用以下形式：PnDTnHnMn.nS.，例如：PT20.345Scriu-osgiparses 为 20.345 秒，P2Dcategories- netobservparses 为 2 天，如果您使用的是 EndpointDsl/ComponentDsl，您可以执行类似于 Duration.ofSeconds （）的操作，因为这些 Java API typesafe。		Duration

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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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
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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象

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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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)	要使用 azure 队列服务进行身份验证的相关 azure 帐户名称的访问密钥。		字符串
凭证 （安全）	可以注入 StorageSharedKeyCredential 来创建 azure 客户端，其中包含重要的身份验证信息。		StorageSharedKeyCredential

所需信息选项

要使用此组件，您可以 3 个选项来提供所需的 Azure 身份验证信息：

- 为您的 Azure 帐户提供 **accountName** 和 **accessKey**，这是开始的最简单方法。accessKey 可以通过您的 Azure 门户生成。
- 提供 StorageSharedKeyCredential 实例，它可以提供给 **凭证** 选项。
- 提供 QueueServiceClient 实例，它可以提供给 **serviceClient**。注意：您不需要创建特定的客户端，如 QueueClient，QueueServiceClient 代表可用于检索较低级别的客户端。

14.6. 使用方法

例如，若要从 **storageAccount** 存储帐户中的队列 **messageQueue** 获取消息内容，请使用以下代码片段：

```
from("azure-storage-queue://storageAccount/messageQueue?accessKey=yourAccessKey").
to("file://queuedirectory");
```

14.6.1. 由组件制作者评估的消息标头

标头	变量名称	类型	操作	描述
CamelAzureStorageQueueSegmentOptions	**QueueConstants.QUEUES_SEGMENT_OPTIONS**	**QueuesSegmentOptions**	**listQueues**	列出队列的选项
CamelAzureStorageQueueTimeout	**QueueConstants.TIMEOUT**	**Duration**	All	将引发 \{@link RuntimeException} 以外的可选超时值。
CamelAzureStorageQueueMetadata	**QueueConstants.METADATA**	**Map<String,String>**	**createQueue**	与队列关联的元数据
CamelAzureStorageQueueTimeToLive	**QueueConstants.TIME_TO_LIVE**	**Duration**	**sendMessage**	消息在队列中保持活动状态的时长。如果未设置该值将默认为 7 天，如果传递 -1，则消息不会过期。生存时间必须是 -1 或任意正数。
CamelAzureStorageQueueVisibilityTimeout	**QueueConstants.VISIBILITY_TIMEOUT**	**Duration**	**sendMessage**, **receiveMessages**, **updateMessage**	消息在队列中不可见的超时时间。如果未设置该值将默认为 0，且信息将立即可见。超时必须在 0 秒和 7 天之间。
CamelAzureStorageQueueCreateQueue	**QueueConstants.CREATE_QUEUE**	**布尔值**	**sendMessage**	当设置为 **true** 时，将在发送消息到队列时自动创建队列。
CamelAzureStorageQueuePopReceipt	**QueueConstants.POP_RECEIPT**	**字符串**	**deleteMessage**, **updateMessage**	必须匹配删除或更新消息的唯一标识符。
CamelAzureStorageQueueMessageId	**QueueConstants.MESSAGE_ID**	**字符串**	**deleteMessage**, **updateMessage**	要删除或更新的消息 ID。

标头	变量名称	类型	操作	描述
CamelAzureStorageQueueMaxMessages	**QueueConstants.MAX_MESSAGES**	**整数**	**receiveMessages**, **peekMessages**	如果要获取的最大消息数，如果队列中存在的消息数量少于请求的所有消息，则返回。如果仅检索 1 个消息，则允许的范围为 1 到 32 个消息。
CamelAzureStorageQueueOperation	**QueueConstants.QUEUE_OPERATION**	**QueueOperationDefinition**	All	指定要执行的制作者操作，请参阅此页面上的与制作者操作相关的文档。
CamelAzureStorageQueueName	**QueueConstants.QUEUE_NAME**	**字符串**	All	覆盖队列名称。

14.6.2. 由组件制作者或消费者设置的消息标头

标头	变量名称	类型	描述
CamelAzureStorageQueueMessageId	**QueueConstants.MESSAGE_ID**	**字符串**	发送到队列的消息 ID。
CamelAzureStorageQueueInsertionTime	**QueueConstants.INSERTION_TIME**	**OffsetDateTime**	Message 插入到队列中的时间。
CamelAzureStorageQueueExpirationTime	**QueueConstants.EXPIRATION_TIME**	**OffsetDateTime**	消息将过期并自动删除的时间。
CamelAzureStorageQueuePopReceipt	**QueueConstants.POP_RECEIPT**	**字符串**	删除/更新消息需要这个值。如果删除失败，使用这个 popreceipt，则消息已被另一个客户端取消队列。
CamelAzureStorageQueueTimeNextVisible	**QueueConstants.TIME_NEXT_VISIBLE**	**OffsetDateTime**	消息再次在 Queue 中可见的时间。
CamelAzureStorageQueueDequeueCount	**QueueConstants.DEQUEUE_COUNT**	**long**	消息已排队的次数。

标头	变量名称	类型	描述
CamelAzureStorageQueueRawHttpHeaders	**QueueConstants.RAW_HTTP_HEADERS**	**httpHeaders**	返回可供用户使用的非稀疏 httpHeaders。

14.6.3. 高级 Azure Storage Queue 配置

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

```
StorageSharedKeyCredential credential = new StorageSharedKeyCredential("yourAccountName",
"yourAccessKey");
String uri = String.format("https://%s.queue.core.windows.net", "yourAccountName");

QueueServiceClient client = new QueueServiceClientBuilder()
                    .endpoint(uri)
                    .credential(credential)
                    .buildClient();
// This is camel context
context.getRegistry().bind("client", client);
```

然后，在 Camel **azure-storage-queue** 组件配置中引用此实例：

```
from("azure-storage-queue://cameldev/queue1?serviceClient=#client")
.to("file://outputFolder?fileName=output.txt&fileExist=Append");
```

14.6.4. 在 registry 中自动检测 QueueServiceClient 客户端

组件能够检测在 registry 中存在 QueueServiceClient bean。如果这是该类型的唯一实例，它将用作客户端，您不必将其定义为 uri 参数，如上例所示。这对端点的智能配置非常有用。

14.6.5. Azure Storage Queue Producer 操作

Camel Azure Storage Queue 组件在生成者端提供广泛的操作：

服务级别的操作

对于这些操作，**需要 accountName**。

操作	描述
listQueues	列出存储帐户中的队列，该帐户通过指定标记开始的过滤器。

队列级别的操作

对于这些操作，**需要 accountName 和 queueName**。

操作	描述
createQueue	创建新队列。
deleteQueue	永久删除队列。
clearQueue	删除队列中的所有消息。
sendMessage	**默认 Producer Operation** Sends a message with a given time-to-live and a timeout period，其中消息在队列中不可见。消息文本从交换消息正文评估。默认情况下，如果队列不存在，它将首先创建一个空队列。如果要禁用此功能，请将 config **createQueue** 或 header **CamelAzureStorageQueueCreateQueue** 设置为 **false**。
deleteMessage	删除队列中的指定消息。
receiveMessages	最多从队列中检索消息数量，并在超时时间内从其他操作中隐藏它们。但是，由于可靠性的原因，它不会从队列中分离消息。
peekMessages	从队列前到最大消息数的 peek 消息。
updateMessage	使用新消息更新队列中的特定消息，并重置可见性超时。消息文本从交换消息正文评估。

请参阅此页面中的示例部分，了解如何在 camel 应用程序中使用这些操作。

14.6.6. 消费者示例

要将队列消耗到一个批处理中最多 5 个消息的文件组件中，您可以执行以下操作：

```
from("azure-storage-queue://cameldev/queue1?serviceClient=#client&maxMessages=5")
.to("file://outputFolder?fileName=output.txt&fileExist=Append");
```

14.6.7. 制作者操作示例

- **listQueues**:

```
from("direct:start")
    .process(exchange -> {
      // set the header you want the producer to evaluate, refer to the previous
      // section to learn about the headers that can be set
      // e.g, to only returns list of queues with 'awesome' prefix:
      exchange.getIn().setHeader(QueueConstants.QUEUES_SEGMENT_OPTIONS, new
QueuesSegmentOptions().setPrefix("awesome"));
    })
    .to("azure-storage-queue://cameldev?serviceClient=#client&operation=listQueues")
    .log("${body}")
    .to("mock:result");
```

- **createQueue** ：

```
from("direct:start")
   .process(exchange -> {
    // set the header you want the producer to evaluate, refer to the previous
    // section to learn about the headers that can be set
    // e.g:
    exchange.getIn().setHeader(QueueConstants.QUEUE_NAME, "overrideName");
   })
   .to("azure-storage-queue://cameldev/test?serviceClient=#client&operation=createQueue");
```

- **deleteQueue**:

```
from("direct:start")
   .process(exchange -> {
    // set the header you want the producer to evaluate, refer to the previous
    // section to learn about the headers that can be set
    // e.g:
    exchange.getIn().setHeader(QueueConstants.QUEUE_NAME, "overrideName");
   })
   .to("azure-storage-queue://cameldev/test?serviceClient=#client&operation=deleteQueue");
```

- **clearQueue** :

```
from("direct:start")
   .process(exchange -> {
    // set the header you want the producer to evaluate, refer to the previous
    // section to learn about the headers that can be set
    // e.g:
    exchange.getIn().setHeader(QueueConstants.QUEUE_NAME, "overrideName");
   })
   .to("azure-storage-queue://cameldev/test?serviceClient=#client&operation=clearQueue");
```

- **sendMessage**:

```
from("direct:start")
   .process(exchange -> {
    // set the header you want the producer to evaluate, refer to the previous
    // section to learn about the headers that can be set
    // e.g:
    exchange.getIn().setBody("message to send");
    // we set a visibility of 1min
    exchange.getIn().setHeader(QueueConstants.VISIBILITY_TIMEOUT, Duration.ofMinutes(1));
   })
   .to("azure-storage-queue://cameldev/test?serviceClient=#client");
```

- **deleteMessage**:

```
from("direct:start")
   .process(exchange -> {
    // set the header you want the producer to evaluate, refer to the previous
    // section to learn about the headers that can be set
    // e.g:
    // Mandatory header:
    exchange.getIn().setHeader(QueueConstants.MESSAGE_ID, "1");
    // Mandatory header:
```

```
        exchange.getIn().setHeader(QueueConstants.POP_RECEIPT, "PAAAAHEEERXXX-1");
    })
    .to("azure-storage-queue://cameldev/test?serviceClient=#client&operation=deleteMessage");
```

- **receiveMessages**:

```
from("direct:start")
    .to("azure-storage-queue://cameldev/test?serviceClient=#client&operation=receiveMessages")
    .process(exchange -> {
        final List<QueueMessageItem> messageItems = exchange.getMessage().getBody(List.class);
        messageItems.forEach(messageItem -> System.out.println(messageItem.getMessageText()));
    })
    .to("mock:result");
```

- **peekMessages**:

```
from("direct:start")
    .to("azure-storage-queue://cameldev/test?serviceClient=#client&operation=peekMessages")
    .process(exchange -> {
        final List<PeekedMessageItem> messageItems = exchange.getMessage().getBody(List.class);
        messageItems.forEach(messageItem -> System.out.println(messageItem.getMessageText()));
    })
    .to("mock:result");
```

- **updateMessage**:

```
from("direct:start")
    .process(exchange -> {
        // set the header you want the producer to evaluate, refer to the previous
        // section to learn about the headers that can be set
        // e.g:
        exchange.getIn().setBody("new message text");
        // Mandatory header:
        exchange.getIn().setHeader(QueueConstants.MESSAGE_ID, "1");
        // Mandatory header:
        exchange.getIn().setHeader(QueueConstants.POP_RECEIPT, "PAAAAHEEERXXX-1");
        // Mandatory header:
        exchange.getIn().setHeader(QueueConstants.VISIBILITY_TIMEOUT, Duration.ofMinutes(1));
    })
    .to("azure-storage-queue://cameldev/test?serviceClient=#client&operation=updateMessage");
```

14.6.8. 开发注意事项(Important)

当在这个组件上开发时，您需要获取您的 Azure accessKey 来运行集成测试。除了模拟的单元测试外，**还需要在每次进行更改时运行集成测试，甚至进行客户端升级，因为 Azure 客户端也可以在次版本升级时中断操作。**要运行集成测试，在此组件目录中运行以下 maven 命令：

```
mvn verify -PfullTests -DaccountName=myacc -DaccessKey=mykey
```

其中 **accountName** 是您的 Azure 帐户名称，accessKey 是从 Azure 门户生成的访问密钥。

14.7. 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.azure-storage-queue.access-key	要使用 azure 队列服务进行身份验证的相关 azure 帐户名称的访问密钥。		字符串
camel.component.azure-storage-queue.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.azure-storage-queue.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.azure-storage-queue.configuration	组件配置。选项是 org.apache.camel.component.azure.storage.queue.QueueConfiguration 类型。		QueueConfiguration
camel.component.azure-storage-queue.create-queue	当设置为 true 时，将在发送消息到队列时自动创建队列。	false	布尔值
camel.component.azure-storage-queue.credentials	可以注入 StorageSharedKeyCredential 来创建 azure 客户端，其中包含重要的身份验证信息。选项是一个 com.azure.storage.common.StorageSharedKeyCredential 类型。		StorageSharedKeyCredential
camel.component.azure-storage-queue.enabled	是否启用 azure-storage-queue 组件的自动配置。这默认是启用的。		布尔值
camel.component.azure-storage-queue.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

Name	描述	默认值	类型
camel.component.azure-storage-queue.max-messages	如果要获取的最大消息数，如果队列中存在的消息数量少于请求的所有消息，则返回。如果仅检索 1 个消息，则允许的范围为 1 到 32 个消息。	1	整数
camel.component.azure-storage-queue.message-id	要删除或更新的消息 ID。		字符串
camel.component.azure-storage-queue.operation	队列服务操作提示到制作者。		QueueOperationDefinition
camel.component.azure-storage-queue.pop-receipt	必须匹配删除或更新消息的唯一标识符。		字符串
camel.component.azure-storage-queue.service-client	服务客户端到存储帐户，以便与队列服务交互。此客户端不保存特定存储帐户的任何状态，而是便捷地将适当的请求发送到服务上的资源。此客户端包含与 Azure Storage 中队列帐户交互的所有操作。客户端允许的操作是创建、列出和删除队列、检索和更新帐户属性，以及检索帐户统计信息。选项是一个 com.azure.storage.queue.QueueServiceClient 类型。		QueueServiceClient
camel.component.azure-storage-queue.time-to-live	消息在队列中保持活动状态的时长。如果未设置该值将默认为 7 天，如果传递 -1，则消息不会过期。生存时间必须是 -1 或任意正数。格式应采用以下形式：PnDTnHnMn.nS.，例如：PT20.345Scriu-osgiparses 为 20.345 秒，P2Dcategories- netobservparses 为 2 天，如果您使用的是 EndpointDsl/ComponentDsl，您可以执行类似于 Duration.ofSeconds （）的操作，因为这些 Java API typesafe。选项是一个 java.time.Duration 类型。		Duration
camel.component.azure-storage-queue.timeout	应用到操作的可选超时。如果在超时结束前没有返回响应，则将抛出 RuntimeException。选项是一个 java.time.Duration 类型。		Duration

Name	描述	默认值	类型
camel.component.azure-storage-queue.visibility-timeout	消息在队列中不可见的超时时间。超时必须在 1 秒和 7 天之间。格式应采用以下形式：PnDTnHnMn.nS.，例如：PT20.345Scriu-osgiparses 为 20.345 秒，P2Dcategories- netobservparses 为 2 天，如果您使用的是 EndpointDsl/ComponentDsl，您可以执行类似于 Duration.ofSeconds （）的操作，因为这些 Java API typesafe。选项是一个 java.time.Duration 类型。		Duration

第 15 章 BEAN

仅支持生成者

Bean 组件将 Bean 绑定到 Camel 消息交换。

15.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 bean 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-bean-starter</artifactId>
</dependency>
```

15.2. 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

15.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

15.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

15.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

15.4. 组件选项

Bean 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
cache (producer)	**弃用了** Use singleton 选项。	true	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

Name	描述	默认值	类型
scope (producer)	bean 范围。使用单例范围（默认）创建 Bean 时，或在端点生命周期中仅查找一次并重复使用。当并发线程同时调用 bean 时，bean 应该为 thread-safe。使用请求范围时，会创建 bean 或查找每个请求一次（交换）。如果要在处理请求时将状态存储在 bean 上，而您希望在处理请求时多次调用同一 bean 实例，则可以使用它。bean 不必是 thread-safe，因为实例只能从同一请求调用。使用委派范围时，每个调用将查找或创建 bean。但是，如果查找，这被委派给 bean registry，如 Spring 或 CDI （如果使用），这取决于其配置可以充当单例或原型范围。因此，这取决于委派的 registry。 Enum 值： • 单例 • Request（请求） • Prototype	单例	BeanScope
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

15.5. 端点选项

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：

15.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
beanName (common)	**必需** 设置要调用的 bean 的名称。		字符串

15.5.2. 查询参数(5 参数)

Name	描述	默认值	类型
cache (common)	**弃用的** Use scope 选项替代。		布尔值
method (common)	设置要在 bean 上调用的方法名称。		字符串
scope (common)	bean 范围。使用单例范围（默认）创建 Bean 时，或在端点生命周期中仅查找一次并重复使用。当并发线程同时调用 bean 时，bean 应该为 thread-safe。使用请求范围时，会创建 bean 或查找每个请求一次（交换）。如果要在处理请求时将状态存储在 bean 上，而您希望在处理请求时多次调用同一 bean 实例，则可以使用它。bean 不必是 thread-safe，因为实例只能从同一请求调用。使用原型范围时，每个调用将查找或创建 bean。但是，如果查找，这被委派给 bean registry，如 Spring 或 CDI （如果使用），这取决于其配置可以充当单例或原型范围。因此，这取决于委派的 registry。 Enum 值： • 单例 • Request（请求） • Prototype	单例	BeanScope
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
参数 (advanced)	用于在 bean 上配置其他属性。		Map

15.6. 例子

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 XML 文件中定义 bean。

您还可以使用 bind 方法通过 Camel 的 Registry 手动注册 Bean。

注册了端点后，您可以构建使用它来处理交换的 Camel 路由。

bean: 端点无法定义为路由的输入；例如，您无法使用来自它的输入，您只能从一些入站消息 Endpoint 中作为输出路由到 bean 端点。因此，请考虑使用 direct: 或 queue: endpoint 作为输入。

您可以使用 ProxyHelper 上的 createProxy （） 方法来创建将生成交换并将其发送到任何端点的代理：

和使用 XML DSL 的同一路由：

```
<route>
  <from uri="direct:hello"/>
  <to uri="bean:bye"/>
</route>
```

15.7. BEAN 作为端点

Camel 还支持将 Bean 作为端点调用。当交换路由到 myBean Camel 时，会出现什么情况，即 Camel 将使用 Bean Binding 调用 bean。bean 的源只是一个普通 POJO。

Camel 通过将 Exchange 的 In body 转换为 String 类型并在 Exchange Out 正文上存储方法的输出，来使用 Bean Binding 调用 sayHello 方法。

15.8. JAVA DSL BEAN 语法

Java DSL 附带组件的语法化 sugar。您可以使用以下语法，而不是将 bean 明确指定为端点（例如：("bean:beanName")）：

```
// Send message to the bean endpoint
// and invoke method resolved using Bean Binding.
from("direct:start").bean("beanName");

// Send message to the bean endpoint
// and invoke given method.
from("direct:start").bean("beanName", "methodName");
```

您可以指定 bean 本身，而不是将引用传递给 bean （因此 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);
```

15.9. BEAN BINDING

可以选择要调用的 bean 方法（如果未通过 method 参数明确指定），以及如何由 Message 构建参数值，它们都由 Camel 中所有不同 Bean 集成机制定义。

15.10. 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.bean.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.bean.enabled	是否启用 bean 组件的自动配置。这默认是启用的。		布尔值
camel.component.bean.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.bean.scope	bean 范围。使用单例范围（默认）创建 Bean 时，或在端点生命周期中仅查找一次并重复使用。当并发线程同时调用 bean 时，bean 应该为 thread-safe。使用请求范围时，会创建 bean 或查找每个请求一次（交换）。如果要在处理请求时将状态存储在 bean 上，而您希望在处理请求时多次调用同一 bean 实例，则可以使用它。bean 不必是 thread-safe，因为实例只能从同一请求调用。使用委派范围时，每个调用将查找或创建 bean。但是，如果查找，这被委派给 bean registry，如 Spring 或 CDI （如果使用），这取决于其配置可以充当单例或原型范围。因此，这取决于委派的 registry。		BeanScope

Name	描述	默认值	类型
camel.component.class.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.class.enabled	是否启用类组件的自动配置。这默认是启用的。		布尔值
camel.component.class.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.class.scope	bean 范围。使用单例范围（默认）创建 Bean 时，或在端点生命周期中仅查找一次并重复使用。当并发线程同时调用 bean 时，bean 应该为 thread-safe。使用请求范围时，会创建 bean 或查找每个请求一次（交换）。如果要在处理请求时将状态存储在 bean 上，而您希望在处理请求时多次调用同一 bean 实例，则可以使用它。bean 不必是 thread-safe，因为实例只能从同一请求调用。使用委派范围时，每个调用将查找或创建 bean。但是，如果查找，这被委派给 bean registry，如 Spring 或 CDI （如果使用），这取决于其配置可以充当单例或原型范围。因此，这取决于委派的 registry。		BeanScope
camel.language.bean.enabled	是否启用 bean 语言的自动配置。这默认是启用的。		布尔值
camel.language.bean.scope	bean 范围。使用单例范围（默认）创建 Bean 时，或在端点生命周期中仅查找一次并重复使用。当并发线程同时调用 bean 时，bean 应该为 thread-safe。使用请求范围时，会创建 bean 或查找每个请求一次（交换）。如果要在处理请求时将状态存储在 bean 上，而您希望在处理请求时多次调用同一 bean 实例，则可以使用它。bean 不必是 thread-safe，因为实例只能从同一请求调用。使用原型范围时，每个调用将查找或创建 bean。但是，如果查找，这被委派给 bean registry，如 Spring 或 CDI （如果使用），这取决于其配置可以充当单例或原型范围。因此，在使用表格范围时，这取决于 bean 注册表实施。	单例	字符串
camel.language.bean.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

Name	描述	默认值	类型
camel.component.bean.cache	**弃用了** Use singleton 选项。	true	布尔值
camel.component.class.cache	**弃用了** Use singleton 选项。	true	布尔值

第 16 章 BEAN VALIDATOR

仅支持生成者

Validator 组件使用 Java Bean Validation API （）执行消息正文的 bean 验证。Camel 使用参考实现，即 Hibernate Validator。

16.1. 依赖项

当在 Camel Spring Boot 中使用 bean-validator 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-bean-validator-starter</artifactId>
</dependency>
```

16.2. URI 格式

```
bean-validator:label[?options]
```

其中 label 是描述端点的任意文本值。您可以以以下格式将查询选项附加到 URI 中，

?option=value&option=value&…

16.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

16.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

16.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

16.4. 组件选项

Bean Validator 组件支持 8 个选项，如下所列。

Name	描述	默认值	类型
ignoreXmlConfiguration (producer)	是否忽略 META-INF/validation.xml 文件中的数据。	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

Name	描述	默认值	类型
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
constraintValidatorFactory (advanced)	使用自定义 ConstraintValidatorFactory。		ConstraintValidatorFactory
messageInterpolator (advanced)	使用自定义 MessageInterpolator。		MessageInterpolator
TraversableResolver (advanced)	使用自定义的 TraversableResolver。		TraversableResolver
validationProviderResolver (advanced)	使用自定义 ValidationProviderResolver。		ValidationProviderResolver
ValidatorFactory (advanced)	**Autowired** 使用自定义 ValidatorFactory。		ValidatorFactory

16.5. 端点选项

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：

16.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
label (producer)	**必需** where 标签是一个描述端点的任意文本值。		字符串

16.5.2. 查询参数(8 参数)

Name	描述	默认值	类型
group (producer)	使用自定义验证组。	javax.validation.groups.Default	字符串
ignoreXmlConfiguration (producer)	是否忽略 META-INF/validation.xml 文件中的数据。	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
constraintValidatorFactory (advanced)	使用自定义 ConstraintValidatorFactory。		ConstraintValidatorFactory
messageInterpolator (advanced)	使用自定义 MessageInterpolator。		MessageInterpolator
TraversableResolver (advanced)	使用自定义的 TraversableResolver。		TraversableResolver
validationProviderResolver (advanced)	使用自定义 ValidationProviderResolver。		ValidationProviderResolver
ValidatorFactory (advanced)	使用自定义 ValidatorFactory。		ValidatorFactory

16.6. OSGI 部署

要在 OSGi 环境中使用 Hibernate Validator，请使用专用的 ValidationProviderResolver 实现，就像 org.apache.camel.component.bean.validator.HibernateValidationProviderResolver.以下代码片段演示了此方法。您还可以使用 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，并且验证器组件已部署到 OSGi 环境中，则会自动使用 HibernateValidationProviderResolver。

16.7. 示例

假设我们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 路由时，将仅对制造商和 licensePlate 属性的 @NotNull 约束进行验证(Camel 使用默认组 javax.validation.groups.Default)。

```
from("direct:start")
.to("bean-validator://x")
.to("mock:end")
```

如果要从组 OptionalChecks 检查约束，则必须定义路由，如下所示

```
from("direct:start")
.to("bean-validator://x?group=OptionalChecks")
.to("mock:end")
```

如果要检查这两个组的约束，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")
```

如果需要提供自己的消息插入器、遍历解析器和约束验证器，则必须编写如下路由

```
<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=#myConstraintVa
lidatorFactory")
.to("mock:end")
```

也可以将您的约束描述为 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

```
<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

```
<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 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>
```

16.8. 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.bean-validator.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.bean-validator.constraint-validator-factory	使用自定义 ConstraintValidatorFactory。选项是 javax.validation.ConstraintValidatorFactory 类型。		ConstraintValidatorFactory
camel.component.bean-validator.enabled	是否启用 bean-validator 组件的自动配置。这默认是启用的。		布尔值
camel.component.bean-validator.ignore-xml-configuration	是否忽略 META-INF/validation.xml 文件中的数据。	false	布尔值
camel.component.bean-validator.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.bean-validator.message-interpolator	使用自定义 MessageInterpolator。选项是 javax.validation.MessageInterpolator 类型。		MessageInterpolator
camel.component.bean-validator.traversable-resolver	使用自定义的 TraversableResolver。选项是 javax.validation.TraversableResolver 类型。		TraversableResolver
camel.component.bean-validator.validation-provider-resolver	使用自定义 ValidationProviderResolver。选项是 javax.validation.ValidationProviderResolver 类型。		ValidationProviderResolver
camel.component.bean-validator.validator-factory	使用自定义 ValidatorFactory。选项是 javax.validation.ValidatorFactory 类型。		ValidatorFactory

第 17 章 BINDY

此组件的目标是允许将非结构化数据（或更精确的非 XML 数据）解析到/从 Java Beans （带有注解定义的绑定映射）的解析/绑定。使用 Bindy，您可以从源（如 ）绑定数据：

- CSV 记录，
- 固定长度记录，
- FIX 消息、
- 或者几乎是其他任何非结构化数据

到一个或多个旧 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。此外，对于诸如 Date、Double、Float、Integer、Short、Long 和 BigDecimal 等数据类型，您可以提供在属性格式化期间应用的模式。

对于 BigDecimal 数字，您还可以定义精度和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**	https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html
十进制（十进制）	**DecimalFormat**	**##.###.###**	https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/DecimalFormat.html

其中 Decimal = Double, Integer, Float, Short, Long

支持的格式

第一个发行版本只支持以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中定义模型（如 com.acme.model）和每个模型类（如 Order, Client, Instrument, ...）将所需的注解（这里介绍的）添加到 Class 或 字段中。

多个模型

当您使用类名称而不是软件包名称配置 bindy 时，您可以将多个模型放在同一个软件包中。

17.1. 依赖项

当在 Camel Spring Boot 中使用 bindy-csv 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-bindy-starter</artifactId>
</dependency>
```

17.2. 选项

Bindy dataformat 支持 5 个选项，如下所列。

Name	默认值	Java 类型	描述
type		**Enum**	**必需** 是否使用 Csv、修复或 KeyValue。 Enum 值： • csv • 已修复 • KeyValue
classType		**字符串**	要使用的模型类名称。
locale		**字符串**	配置要使用的默认区域设置，如我们获取单元状态。要使用 JVM 平台默认区域设置，请使用名称 default。

Name	默认值	Java 类型	描述
unwrapSingleInstance		布尔值	当 unmarshalling 应该取消和返回的实例时，而不是嵌套在 java.util.List 中。
allowEmptyStream		布尔值	是否在 unmarshal 进程中允许空流。如果为 true，如果没有提供记录的正文，则不会抛出异常。

17.3. 注解

创建的注解允许将您的模型的不同概念映射到 POJO，如下所示：

- 记录类型(CSV、键值对（如 FIX 消息)、固定长度 ...）
- 链接（链接另一个对象中的对象）
- DataField 及其属性(int, type, ...),
- KeyValuePairField （用于 key = FIX 财务消息一样的值格式），
- 部分（要识别标头、正文和页脚部分），
- OneToMany,
- BindyConverter,
- FormatFactories

本节将描述它们。

17.3.1. 1.CsvRecord

CsvRecord 注解用于识别模型的根类。它表示记录 = "CSV 文件的行"，并可链接到多个子模型类。

注解名称	记录类型	级别
CsvRecord	CSV	类

参数名称	类型	必填	默认值	info
分隔符	字符串	✓		用于在令牌中分割记录的分隔符（必需）- 可以是 ',' 或 ';' 或 'anything'。唯一支持的空格字符是 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，例如 '|' 符号，则必须屏蔽它，如 '|'。
allowEmptyStream	布尔值		false	allowEmptyStream 参数将允许 prcoess 的 CSV 文件不可避免的流。
autospanLine	布尔值		false	最后一条记录跨越其余行（可选）- 如果启用，则最后一列会自动跨越到行尾，例如，如果其注释，则允许行包含所有字符，也是分隔符 char。
crlf	字符串		WINDOWS	要在每个记录后添加回车符（可选）- 允许使用回车字符。如果您指定了之前列出的三个值，则您输入的值（自定义）将用作 CRLF 字符。可以使用三个值：WINDOWS、UNIX、MAC 或 custom。
endWithLineBreak	布尔值		true	如果 CSV 文件以换行符或未结束（可选）
generateHeaderColumns	布尔值		false	generateHeaderColumns 参数允许在 CSV 中添加生成的包含列名称的标头
isOrdered	布尔值		false	指明消息是否在输出中排序
name	字符串			描述记录的名称（可选）
quote	字符串		"	是否使用给定的引号字符（可选）- 在生成 CSV 时指定字段的引号字符。此注释与模型的根类关联，必须声明一次。
引用	布尔值		false	指明在 marshaling （可选）时是否必须用引号括起值（和标头）
quotingEscaped	布尔值		false	指明在引用时是否必须转义值（可选）

参数名称	类型	必填	默认值	info
removeQuotes	布尔值		true	如果 unmarshalling 应该尝试删除每个字段的引号，则 remove quotes 参数标志
skipField	布尔值		false	skipField 参数将允许跳过 CSV 文件的字段。如果某些字段不需要，可以跳过它们。
skipFirstLine	布尔值		false	skipFirstLine 参数将允许跳过或不是 CSV 文件的第一行。此行通常包含列定义
trimLine	布尔值		true	在将行解析为数据字段之前，是否修剪每行(stand 和 end)。

case 1 : separator = ','

用于隔离 CSV 记录中字段的分隔符是， ：

> 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 或更早版本

当要解析 CSV 记录的字段包含， 或 也会用作分隔符时，我们应该找到另一个策略来告知 camel bindy 如何处理这个问题单。 要使用逗号定义包含数据的字段，您可以使用单引号或双引号作为分隔符（例如 : '10', 'Street 10, NY', 'USA' 或 "10", "Street 10, NY", "USA"）。

注意

在这种情况下，将通过 bindy 删除行的第一个和最后一个字符，它们是单引号或双引号。

```
"10","J","Pauline"," M","XD12345678","Fortis Dynamic 15,15","2500","USD","08-01-2009"
```

```
@CsvRecord( separator = "\",\"" )
public Class Order {

}
```

bindy 会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，但只需如下所示：

```
"10","J","Pauline"," M","XD12345678","Fortis Dynamic 15,15","2500","USD","08-01-2009"
```

```
@CsvRecord( separator = "," )
public Class Order {

}
```

请注意，如果要从 Object 放入 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，如下所示：

```
@CsvRecord( separator = ",", quote = "\"" )
public Class Order {

}
```

case 5 : separator 和 skipFirstLine

当客户端想在文件的第一行（数据字段的名称）中时，该功能很有趣：

```
order id, client id, first name, last name, isin code, instrument name, quantity, currency, date
```

要通知绑定，必须在解析过程中跳过此第一行，我们使用属性 ：

```
@CsvRecord(separator = ",", skipFirstLine = true)
public Class Order {

}
```

case 6 : generateHeaderColumns

要在生成的 CSV 第一行中添加，必须在注解中将 generateHeaderColumns 属性设置为 true，如下所示：

```
@CsvRecord( generateHeaderColumns = true )
public Class Order {

}
```

因此，在 unmarshaling 过程中 Bindy 将生成类似以下的 CSV：

```
order id, client id, first name, last name, isin code, instrument name, quantity, currency, date
10, J, Pauline, M, XD12345678, Fortis Dynamic 15/15, 2500, USD, 08-01-2009
```

case 7 : carriage 返回

如果 camel-bindy 将运行的平台不是 Windows，但 Macintosh 或 Unix，您可以更改类似如下的 crlf 属性。提供三个值：WINDOWS、UNIX 或 MAC

```
@CsvRecord(separator = ",", crlf="MAC")
public Class Order {

}
```

另外，如果出于某种原因您需要添加不同的行尾字符，您可以选择使用 crlf 参数指定它。在以下示例中，我们可以使用逗号结束行，后跟换行符：

```
@CsvRecord(separator = ",", crlf=",\n")
public Class Order {

}
```

case 8 : isOrdered

有时，在从模型创建 CSV 记录时遵循的顺序与解析过程中使用的顺序不同。然后，在这种情况下，我们可以使用属性 isOrdered = true 来表示这一点与 DataField 注解的属性 位置 相结合。

```
@CsvRecord(isOrdered = true)
public Class Order {

  @DataField(pos = 1, position = 11)
  private int orderNr;

  @DataField(pos = 2, position = 10)
  private String clientNr;

}
```

注意

pos 用于解析文件流，而 位置 则用于生成 CSV。

17.3.2. 2.Link

链接注解将允许将对象链接在一起。

注解名称	记录类型	级别
Link	all	类和属性

参数名称	类型	必填	默认值	info
linkType	LinkType		OneToOne	标识类之间关系的链接类型

从当前版本起，只允许一对一关系。

例如：如果模型类客户端链接到 Order 类，则使用 Order 类中的注解链接，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：

类链接

```
@Link
public class Client {

}
```

17.3.3. 3.DataField

DataField 注解定义字段的属性。每个 datafield 都由记录中的位置、类型（字符串、int、日期、...）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	类型	必填	默认值	info
pos	int	✓		数据在输入记录中的位置必须从 1 开始（必需）。请参阅 position 参数。
align	字符串		R	将文本与右侧或左对齐。使用值 <tt>R</tt> 或 <tt>L</tt>。
Clip	布尔值		false	如果在使用固定长度时超过允许的长度，则指示字段中冲突数据。
columnName	字符串			标头列的名称（可选）。使用属性的名称作为默认值。仅在 **CsvRecord** 有 **generateHeaderColumns = true**时才适用
decimalSeparator	字符串			与 BigDecimal 号码一起使用的十进制 9 月器
defaultValue	字符串			如果没有设置值，则字段的默认值
delimiter	字符串			如果字段有变量长度，要使用的可选分隔符
groupingSeparator	字符串			当我们想通过组 e.g. 123,456.789 格式化/稀疏的数字时，分组了要与 BigDecimal 号码一起使用的 9 个分组。
impliedDecimalSeparator	布尔值		false	指明是否存在代表在指定位置的十进制点
length	int		0	如果记录被设置为固定长度，则数据块的长度（字符数）
lengthPos	int		0	识别记录中定义此字段预期的固定长度的数据字段
方法	字符串			调用的方法名称，以在 DataField 上应用此类自定义。这必须是 datafield 本身上的方法，或者您必须提供类方法的静态完全限定名称，例如：请参阅单元测试 org.apache.camel.dataformat.bindy.csv.BindySimpleCsvFunctionWithExternalMethodTest.replaceToBar
name	字符串			字段的名称（可选）
paddingChar	char			如果记录被设置为固定长度，则为 char to pad

参数名称	类型	必填	默认值	info
pattern	字符串			Java formatter （示例为SimpleDateFormat）的模式将用于转换数据（可选）。如果使用模式，则建议在 bindy 数据格式上设置区域设置。设置为已知区域设置，如 "us"，或使用 "default" 来使用平台默认区域设置。
position	int		0	生成输出消息中的字段位置（从 1 开始）。当 CSV 生成的字段的位置（输出消息）中的字段位置与输入位置(pos)不同时，必须使用。请参阅 pos 参数。
精度	int		0	要创建的 \{@link java.math.BigDecimal} 数字的精度
required	布尔值		false	指明字段是否强制
Rounding	字符串		CEILING	Round mode to round/scale a bigDecimal Values : UP, DOWN, CEILING, FLOOR, HALF_UP, HALF_DOWN,HALF_EVEN, UNNECESSARY e.g : Number = 123456.789, Precision = 2, Rounding = CEILING Result: 123.79.
timezone	字符串			要使用的时区。
trim	布尔值		false	指明值是否应修剪

case 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，但仍然在类 Order 中从 5 开始。2 到 4 的数字在类客户端中定义（请参见此处）。

在另一个模型类中继续位置

```
public class Client {

   @DataField(pos = 2)
   private String clientNr;

   @DataField(pos = 3)
   private String firstName;

   @DataField(pos = 4)
   private String lastName;
}
```

case 2：模式

模式允许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

当您想定义数字的十进制部分时，精度很有用。

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：输出中的位置不同

position 属性将通知 bindy 如何将字段放在生成的 CSV 记录中。默认情况下，使用的位置对应于通过属性 pos 定义的位置。如果位置不同（这意味着我们有一个 asymetric processus marshaling from unmarshaling），则我们可以使用 位置 来指示这一点。

下面是一个示例：

输出中的位置不同

```
@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 的属性 isOrdered = true 结合使用。

case 5 : 必需

如果需要字段，只需使用 所需的 属性设为 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，则解析器将使用以下信息来引发错误：

```
Some fields are missing (optional or mandatory), line :
```

case 6 : trim

如果字段具有前导和/或尾随空格，应在处理前删除，只需使用属性 修剪 设置为 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 属性指示的值。

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;
}
```

case 8 : columnName

仅在 @CsvRecord 具有注解 generateHeaderColumns = true 时指定属性的列名称。

列名称

```
@CsvRecord(separator = ",", generateHeaderColumns = true)
public class Order {

    @DataField(pos = 1)
    private int orderNr;

    @DataField(pos = 5, columnName = "ISIN")
    private String isinCode;

    @DataField(name = "Name", pos = 6)
    private String instrumentName;
}
```

此属性仅适用于可选字段。

17.3.4. 4.FixedLengthRecord

FixedLengthRecord 注解用于识别模型的根类。它代表 record = "文件/message 的行包含数据固定长度（字符数）格式，并可链接到多个子模型类。这个格式是一个具体格式，因为字段的数据可以与右侧或左侧一致。

当数据的大小没有完全填写字段长度时，我们可以添加"本"字符。

注解名称	记录类型	级别
FixedLengthRecord	已修复	类

参数名称	类型	必填	默认值	info
countGrapheme	布尔值		false	指明如何计算收费
crlf	字符串		WINDOWS	要在每个记录后添加回车字符（可选）。可能的值有：WINDOWS、UNIX、MAC 或 custom。这个选项仅在 marshalling 期间使用，而 unmarshalling 使用系统默认 JDK 提供的行分隔符，除非自定义 eol。
EOL	字符串			字符用于在每个记录后考虑行尾（可选 - default: ""），它可以帮助默认 JDK 提供的行分隔符使用，除非提供了任何其他值。）此选项仅在 unmarshalling 期间使用，其中 marshall 使用系统默认提供的行分隔符作为 "WINDOWS"。
footer	类		void	表示此类型的记录可以跟随文件末尾的单个页脚记录
header	类		void	表示此类型的记录前面可以加上文件开头的单个标头记录
ignoreMissingChars	布尔值		false	指明是否忽略太短行
ignoreTrailingChars	布尔值		false	表示在 unmarshalling / 解析时会忽略最后一个映射文件之外的字符。此注释与模型的根类关联，必须声明一次。
length	int		0	记录的固定长度（字符数）。这意味着，记录将始终是使用 \{#paddingChar （）}'s 持续添加的
name	字符串			描述记录的名称（可选）
paddingChar	char			使用 char to pad。
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，不是标头或页脚）。
skipHeader	布尔值		false	配置数据格式，以跳过标头记录的 marshalling / unmarshalling。在主记录上配置此参数（例如，不是标头或页脚）。

记录**不能是**标头/页**脚和主要固定**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 alignment 和 padding 修复长度记录

此更详细的示例演示了如何定义字段的对齐以及如何分配 padding 字符，该字符为 ' ' ：

```
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，其中我们希望使用 **'0'** 而不是 **' ' pad**。在这种情况下，您可以在模型中使用 **@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：带有分隔符的固定长度记录

修复了长度记录有时会在记录内有分隔的内容。firstName 和 lastName 字段使用以下示例中的 ^ 字符分隔：

```
10A9Pauline^M^ISINXD12345678BUYShare000002500.45USD01-08-2009
```

固定分隔

```
@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, sequential 值而不是精确列号来定义固定长度记录中的 pos 值。

case 5 ：带有记录定义字段长度的固定长度记录

有时，固定长度记录可能会包含一个字段，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，instrumentNumber 字段值的长度由记录中的 instrumentNumberLen 字段的值定义。

```
10A9Pauline^M^ISIN10XD12345678BUYShare000002500.45USD01-08-2009
```

固定分隔

```
@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：带有标头和页脚的固定长度记录

bindy 将发现作为模型的一部分配置的固定长度标头和页脚记录 - 只要注解的类存在于与主 @FixedLengthRecord 类相同的软件包中，或者在其中一个配置的扫描软件包中。以下文本说明了两个固定长度记录，它们被标头记录和页脚记录括起来。

```
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，其包含比目标用例所需信息更多的系统。在这种情况下，跳过我们不需要的字段的声明和解析。要接受这一点，Bindy 将跳过记录中的下一个映射字段，如果下一个声明字段的 pos 值不在最后一个解析字段的光标位置。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，可能不关注某些字段之外的内容。在这种情况下，您可以通过设置 @FixedLengthRecord 声明上的 ignoreTrailingChars 属性来告知 Bindy 跳过除最后一个映射字段以外的所有内容解析。

```
@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

}
```

17.3.5. 5.消息

Message 注解用于识别包含键值对字段的模型类。此类格式主要在 financial Exchange Protocol Messages (FIX)中使用。但是，此注解可用于通过密钥标识数据的任何其他格式。密钥对值由分隔符相互分隔，可以是诸如 tab delimitor (unicode representation : \u0009)或标题的开头(unicode representation : \u0001)

注意

要使用 FIX 消息，模型必须包含链接到根消息类的标头和 Trailer 类，可以是 Order 类。这不是强制要求，当您将 camel-bindy 与 camel-fix 结合使用时，这是基于 quickFix 项目的修复网关时，将非常有用。

注解名称	记录类型	级别
消息	键值对	类

注解名称	记录类型	级别

参数名称	类型	必填	默认值	info
keyValuePairSeparator	字符串	✓		键值对分隔符用于从其键中分离值（必需）。可以是 '\u0001', '\u0009', '#' 或 'anything'。
pairSeparator	字符串	✓		用于在令牌中分割键值对的对（必需）。可以是 '=', ';', 或 'anything'。
crlf	字符串		WINDOWS	要在每个记录后添加回车字符（可选）。可能的值 = WINDOWS、UNIX、MAC 或 custom。如果您指定了之前列出的三个值，则您输入的值（自定义）将用作 CRLF 字符。
isOrdered	布尔值		false	指明消息是否在输出中排序。此注释与模型的消息类关联，必须声明一次。
name	字符串			描述消息的名称（可选）
type	字符串		修复	type 用于定义消息的类型（如 FIX、EMX、...）（可选）
version	字符串		4.1	version 定义消息的版本（如 4.1、...）（可选）

case 1 : separator = 'u0001'

FIX 消息中用于隔离键值对字段的分隔符是 ASCII 01 字符或 unicode 格式 \u0001。此字符必须第二次转义，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 - message

```
@Message(keyValuePairSeparator = "=", pairSeparator = "\u0001", type="FIX", version="4.1")
public class Order {
```

```
}
```

查看测试问题单

ASCII 字符，如 tab，…无法显示在 WIKI 页面中。因此，请看一个 camel-bindy 测试案例，来准确查看 FIX 消息是什么样子和 Order,Trailer, Header class (https://github.com/apache/camel/blob/main/components/camel-bindy/src/test/java/org/apache/camel/dataformat/bindy/model/fix/simple/Order.java)。

17.3.6. 6.KeyValuePairField

KeyValuePairField 注解定义键值对字段的属性。每个 KeyValuePairField 都由一个标签(= key)及其关联的值标识，一个类型（字符串、int、date、…）、可选模式以及是否需要字段。

注解名称	记录类型	级别
KeyValuePairField	Key Value Pair - FIX	属性

参数名称	类型	必填	默认值	info
tag	int	✓		标识消息中的字段的标签（必需）- 必须是唯一的
impliedDecimalSeparator	布尔值		false	Camel 2.11: 表示在指定位置是否有十进制点
name	字符串			字段的名称（可选）
pattern	字符串			格式者将用于转换数据的模式（可选）
position	int		0	字段在生成的消息中的位置 - 当 FIX 消息中的键/标签的位置必须不同时，必须使用
精度	int		0	要创建的 BigDecimal 数量的精度
required	布尔值		false	指明字段是否强制
timezone	字符串			要使用的时区。

参数名称	类型	必填	默认值	info

问题单 1：标签

这个参数代表消息中字段的密钥：

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;
}
```

case 2：输出中的不同位置

如果我们将放入 FIX 消息中的标签/密钥按照预定义的顺序排序，则使用注释 @KeyValuePairField 的属性 位置。

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;
}
```

17.3.7. 7.部分

在固定长度记录的 FIX 消息中，通常以信息：header, body 和 section 表示不同的部分。注释的 @Section 的目的是告知绑定模型中哪个类代表标头(= section 1)、body (= section 2)和页脚(= section 3)

此注释仅存在一个属性/参数。

注解名称	记录类型	级别
部分	修复	类

参数名称	类型	必填	默认值	info
number	int	✓		部分数

问题单 1：部分

header 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;
    }
```

17.3.8. 8.OneToMany

注释 @OneToMany 的目的是允许处理 List<?& gt; 字段定义的 POJO 类或包含重复组的记录。

注意

OneToMany 的限制要小心，很多绑定都不允许处理在多个层次上定义的重复。

在以下情况下，关系仅是 WORKS：

- 读取包含重复组的 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	类型	必填	默认值	info
mappedTo	字符串			与 Class> 的 List<Type 类型关联的类名称

问题单 1：生成带有重复数据的 CSV

以下是我们需要的 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
```

注意

重复数据涉及书及其发布日期的标题，第一个、姓氏和年龄很常见，以及用于建模的类。Author 类包含 Book 的列表。

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;
}
```

case 2 : Reading FIX message containing group of tags/keys

以下是我们希望在我们的模型中处理的消息：

```
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;
}
```

17.3.9. 9.BindyConverter

注释的 @BindyConverter 定义了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();
    }
}
```

17.3.10. 10.FormatFactories

注释的 @FormatFactories 的目的是在记录级别定义一组转换器。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);
```

```
        }
      };
    }
}
```

17.4. 支持的数据类型

DefaultFormatFactory 通过根据提供的 FormattingOptions 返回接口 FormatFactoryInterface 实例，从而提供以下 datatype 的格式：

- BigDecimal
- BigInteger
- 布尔值
- byte
- 字符
- Date
- å€Œ
- Enums
- æµ®ç‚¹å€¼
- 整数
- LocalDate

- LocalDateTime
- LocalTime
- Long
- short
- 字符串

可以通过提供 registry 中的 FactoryRegistry 实例（如 spring 或 JNDI）来覆盖 DefaultFormatFactory。

17.5. 使用 JAVA DSL

下一步会实例化与此记录类型关联的 DataFormat *bindy* 类，并提供一个类作为参数。

例如，以下命令使用类 BindyCsvDataFormat （对应于与 CSV 记录类型关联的类），它使用 *com.acme.model.MyModel.class* 来初始化这个软件包中配置的模型对象。

```
DataFormat bindy = new BindyCsvDataFormat(com.acme.model.MyModel.class);
```

17.5.1. 设置区域设置

bindy 支持在 dataformat 中配置区域设置，例如

```
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。

```
BindyCsvDataFormat bindy = new BindyCsvDataFormat(com.acme.model.MyModel.class);

bindy.setLocale("default");
```

17.5.2. Unmarshaling

```
from("file://inbox")
  .unmarshal(bindy)
  .to("direct:handleOrders");
```

另外，您可以使用命名引用到数据格式，然后在 Registry 中定义，如 Spring XML 文件：

```
from("file://inbox")
  .unmarshal("myBindyDataFormat")
  .to("direct:handleOrders");
```

Camel 路由将获取 inbox 目录中的文件，unmarshall CSV 记录成模型对象的集合，并将集合发送到 handleOrders 引用的路由。

返回的集合是 Map 对象列表。列表中的每个映射都包含 CSV 每行的模型对象。其后面的原因是 *每行都与多个对象对应*。当您只期望每行返回一个对象时，这可能会造成混淆。

每个对象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 对象以便在路由中处理，您可以使用 Splitter 和 Processor 的组合，如下所示：

```
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();
```

注意 Bindy 使用 CHARSET_NAME 属性或 CHARSET_NAME 标头的事实，如 Exchange 接口中定义，以执行为 unmarshalling 接收的输入流的字符集转换。在一些制作者（如 file-endpoint）中，您可以定义一个字符集。characterset 转换可由此制作者完成。有时，您需要在将此属性发送到 unmarshal 之前从交换中删除此属性或标头。如果您没有删除它，则转换过程可能会进行两次，这可能会导致不必要的结果。

```
from("file://inbox?charset=Cp922")
  .removeProperty(Exchange.CHARSET_NAME)
  .unmarshal("myBindyDataFormat")
  .to("direct:handleOrders");
```

17.5.3. marshaling

要从模型对象集合生成 CSV 记录，请创建以下路由：

```
from("direct:handleOrders")
  .marshal(bindy)
  .to("file://outbox")
```

17.6. 使用 SPRING XML

这非常容易使用 Spring 作为您喜欢的 DSL 语言来声明用于 camel-bindy 的路由。以下示例显示了两个路由，其中第一个将从文件中获取记录，并把内容绑定到其模型。然后，结果将发送到 pojo （无特殊操作），并将它们放入队列。

第二个路由将从队列中提取 pojos，并汇总内容来生成包含 CSV 记录的文件。

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

请验证您的模型类是否实施了 **serializable**，否则队列管理器将引发错误。

17.7. 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。

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>{CamelSBVersion}</version>
</dependency>
```

17.8. 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.allow-empty-stream	是否在 unmarshal 进程中允许空流。如果为 true，如果没有提供记录的正文，则不会抛出异常。	false	布尔值
camel.dataformat.bindy-csv.class-type	要使用的模型类名称。		字符串
camel.dataformat.bindy-csv.enabled	是否启用 bindy-csv 数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.bindy-csv.locale	配置要使用的默认区域设置，如我们获取单元状态。要使用 JVM 平台默认区域设置，请使用名称 default。		字符串
camel.dataformat.bindy-csv.type	是否使用 Csv、修复或 KeyValue。		字符串
camel.dataformat.bindy-csv.unwrap-single-instance	当 unmarshalling 应该取消和返回的实例时，而不是嵌套在 java.util.List 中。	true	布尔值
camel.dataformat.bindy-fixed.allow-empty-stream	是否在 unmarshal 进程中允许空流。如果为 true，如果没有提供记录的正文，则不会抛出异常。	false	布尔值
camel.dataformat.bindy-fixed.class-type	要使用的模型类名称。		字符串
camel.dataformat.bindy-fixed.enabled	是否启用绑定固定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.bindy-fixed.locale	配置要使用的默认区域设置，如我们获取单元状态。要使用 JVM 平台默认区域设置，请使用名称 default。		字符串
camel.dataformat.bindy-fixed.type	是否使用 Csv、修复或 KeyValue。		字符串
camel.dataformat.bindy-fixed.unwrap-single-instance	当 unmarshalling 应该取消和返回的实例时，而不是嵌套在 java.util.List 中。	true	布尔值
camel.dataformat.bindy-kvp.allow-empty-stream	是否在 unmarshal 进程中允许空流。如果为 true，如果没有提供记录的正文，则不会抛出异常。	false	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.enabled	是否启用 bindy-kvp 数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.bindy-kvp.locale	配置要使用的默认区域设置，如我们获取单元状态。要使用 JVM 平台默认区域设置，请使用名称 default。		字符串
camel.dataformat.bindy-kvp.type	是否使用 Csv、修复或 KeyValue。		字符串
camel.dataformat.bindy-kvp.unwrap-single-instance	当 unmarshalling 应该取消和返回的实例时，而不是嵌套在 java.util.List 中。	true	布尔值

第 18 章 浏览

支持生成者和消费者

Browse 组件提供了一个简单的 BrowsableEndpoint，可用于测试、可视化工具或调试。发送到端点的交换都可以浏览。

18.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 浏览 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-browse-starter</artifactId>
</dependency>
```

18.2. URI 格式

```
browse:someName[?options]
```

其中 someName 可以是唯一标识端点的任何字符串。

18.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

18.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

18.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

18.4. 组件选项

Browse 组件支持 3 个选项，如下所列。

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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

18.5. 端点选项

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：

18.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
name (common)	**必需** A 名称，可以是任意字符串来唯一标识端点。		字符串

18.5.2. 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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		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

18.6. 示例

在以下路由中，我们插入一个 浏览： 组件可以浏览要传递的交换：

```
from("activemq:order.in").to("browse:orderReceived").to("bean:processOrder");
```

现在，我们可以从 Java 代码中检查收到的交换：

```
private CamelContext context;

public void inspectReceivedOrders() {
   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
   }
}
```

18.7. 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.browse.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.browse.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.browse.enabled	是否启用浏览组件的自动配置。这默认是启用的。		布尔值
camel.component.browse.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

第 19 章 CASSANDRA CQL

支持生成者和消费者

Apache Cassandra 是一个开源 NoSQL 数据库，旨在处理商业硬件上的大量数据。与 Amazon 的 DynamoDB 一样，Cassandra 具有对等(peer)和无主架构，以避免单点故障和高可用性。与 Google 的 BigTable 一样，Cassandra 使用列系列进行结构化，这些系列可通过 Thrift RPC API 或称为 CQL 的 SQL API 访问。

注意

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。

19.1. 依赖项

当在 Camel Spring Boot 中使用 cql 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-cassandraql-starter</artifactId>
</dependency>
```

19.2. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

19.2.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

19.2.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

19.3. 组件选项

Cassandra CQL 组件支持 3 个选项，如下所列。

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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

19.4. 端点选项

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：

19.4.1. 路径参数(4 参数)

Name	描述	默认值	类型
beanRef (common)	beanRef 使用 bean:id 定义。		字符串
hosts (common)	主机名 Cassandra 服务器。可以使用逗号分隔多个主机。		字符串
port (common)	Cassandra 服务器的端口号。		整数
keyspace (common)	要使用的键空间。		字符串

19.4.2. 查询参数(30 参数)

Name	描述	默认值	类型
clusterName (common)	集群名称。		字符串

Name	描述	默认值	类型
consistencyLevel (common)	要使用的一致性级别。 Enum 值： • 任何 • ONE • TWO • 三个 • 仲裁 • ALL • LOCAL_ONE • LOCAL_QUORUM • EACH_QUORUM • SERIAL • LOCAL_SERIAL		DefaultConsistencyLevel
cql (common)	要执行的 CQL 查询。可使用键 CamelCqlQuery 的消息标头覆盖。		字符串
datacenter (common)	要使用的数据中心。	datacenter1	字符串
loadBalancingPolicyClass (common)	使用特定的 LoadBalancingPolicyClass。		字符串
password (common)	会话身份验证的密码。		字符串
prepareStatements (common)	是否使用 PreparedStatements 还是常规声明。	true	布尔值
resultSetConversionStrategy (common)	要使用实现将 ResultSet 转换为消息正文 ALL、ONE、LIMIT_10, LIMIT_100... 的自定义类。		ResultSetConversionStrategy
session (common)	要使用 Session 实例（您通常不会使用这个选项）。		CqlSession
用户名 (common)	会话身份验证的用户名。		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
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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
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
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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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
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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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

19.5. 端点连接语法

端点可以启动 Cassandra 连接，或使用现有的端点。

URI	描述
cql:localhost/keyspace	单主机、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	默认端口和密钥空间
cql:bean:sessionRef	提供的会话参考
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 Cluster 实例并将其提供给 Camel 端点。

19.6. 消息

19.6.1. 传入的消息

Camel Cassandra 端点需要一个简单对象(Object 或 Object[] 或 Collection<Object>)，该对象将绑定到 CQL 语句作为查询参数。如果消息正文为空或为空，则将在不绑定参数的情况下执行 CQL 查询。

Headers

- CamelCqlQuery （可选，字符串 或 常规状态）

  CQL 查询可以是普通 String，也可以使用 QueryBuilder 构建。

19.6.2. 传出消息

Camel Cassandra 端点会根据 resultSetConversionStrategy 生成一个或多个 Cassandra Row 对象：

- List<Row > if resultSetConversionStrategy is 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

19.7. 软件仓库

Cassandra 可用于存储用于幂等和聚合 EIP 的消息。

Cassandra 可能不是排队用例的最佳工具，即读取 Cassandra 反模式队列和队列，如 datasets。建议对这些表使用 LeveledCompaction 和 small GC grace 设置，以允许快速删除 tombstoned 行。

19.8. IDEMPOTENT 存储库

NamedCassandraIdempotentRepository 将消息密钥存储在 Cassandra 表中，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，可以扩展以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'	主键列
name		仓库名称，用于 NAME 列的值
ttl		生存密钥时间
writeConsistencyLevel		用于插入/删除密钥的一致性级别：NY、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...

19.9. 聚合存储库

NamedCassandraAggregationRepository 在 Cassandra 表中通过关联密钥存储交换，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，Cas1AggregationRepository 没有 NAME 列，可以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,KEY	主键列
exchangeIdColumn	EXCHANGE_ID	exchange Id 列
exchangeColumn	EXCHANGE	Exchange content 列
name		仓库名称，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：NY、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...

19.10. 例子

要在表上插入一些内容，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，您应该可以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。

19.11. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

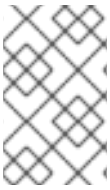
Name	描述	默认值	类型
camel.component.cql.autowired-enabled	是否启用自动关闭。这用于自动关闭选项（选项必须标记为 autowired），方法是在 registry 中查找查找是否有单个匹配类型实例，然后在组件上配置。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.cql.bridge-error-handler	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。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.cql.enabled	是否启用 cql 组件的自动配置。这默认是启用的。		布尔值
camel.component.cql.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

第 20 章 CICS

从 Camel 4.4-redhat

仅支持生成者。

此组件允许您与 IBM CICS® 通用事务处理子系统交互。



注意

仅支持同步模式调用。

20.1. 依赖项

在将 camel-cics 与 Camel Spring Boot 搭配使用时，请将以下 Maven 依赖项添加到 pom.xml 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-cics-starter</artifactId>
</dependency>
```

在使用 camel-cics starer 时，还必须声明 ctgclient.jar 依赖项。

```
<dependency>
  <artifactId>com.ibm</artifactId>
  <groupId>ctgclient</groupId>
  <scope>system</scope>
  <systemPath>${basedir}/lib/ctgclient.jar</systemPath>
</dependency>
```

此 JAR 由 IBM 提供，包含在 cics 系统中。

20.2. URI 格式

```
cics://[interfaceType]/[dataExchangeType][?options]
```

其中 interfaceType 是 camel-cics 调用的外部 API 的 CICS 集。目前，只支持 ECI （外部调用接口）。此组件使用两种类型的 dataExchangeType 与 CICS 服务器通信。

- commarea 是一个存储块，限制为由程序分配的 32763 字节。
- channel 是交换数据的新机制，类似于参数列表。

默认情况下，如果没有指定 dataExchangeType，这个组件会 使用逗号 ：

```
cics://eci?host=xxx&port=xxx...
```

要使用 频道和 容器，您必须在 URI 中明确指定它

```
cics://eci/channel?host=xxx&port=xxx...
```

20.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

20.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

20.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

20.4. 组件选项

CICS 组件支持 17 个选项，如下所列。

Name	描述	默认值	类型
ctgDebug	在底层 IBM CGT 客户端上启用调试模式。	false	java.lang.Boolean
eciBinding	将 Camel Exchange 转换为 EciRequest 的 Binding 实例，反之亦然		com.redhat.camel.component.cics.CICSEciBinding
eciTimeout	与此 ECIRequest 对象关联的 ECI 超时值。ECI 超时值为零表示此 ECIRequest 不会由 CICS 交易网关超时。大于零的 ECI 超时值表示 CICS 事务网关可能会超时 ECIRequest。在从 CICS 收到响应前，ECI 超时可能会过期。这意味着，客户端不会收到来自 CICS 的确认，因为已支持或提交工作单元。	0	short
编码	消息的传输编码。	Cp1145	java.lang.String
gatewayFactory	要使用的连接工厂		com.redhat.camel.component.cics.pool.CICSGatewayFactory
主机	此实例连接的 CICS 事务网关的地址		java.lang.String

Name	描述	默认值	类型
lazyStartProducer	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。		布尔值
port	此实例连接的 CICS 事务网关的端口。	2006	int
protocol	此组件将用于连接到 CICS 事务网关的协议。	tcp	java.lang.String
server	此实例连接的 CICS 服务器的地址。		java.lang.String
sslKeyring	用于客户端加密连接的 SSL 密钥环类或密钥存储文件的完整类名称。		java.lang.String
sslPassword	加密密钥环类或密钥存储的密码		java.lang.String
配置	使用共享的 CICS 配置		com.redhat.camel.component.cics.CICSConfiguration
socketConnectionTimeout	套接字连接超时		int
password	用于身份验证的密码		java.lang.String
userId	用于身份验证的用户 ID		java.lang.String

20.5. 端点选项

CICS 端点使用 URI 语法进行配置：

```
cics://[interfaceType]/[dataExchangeType][?options]
```

使用以下 *路径和 查询参数* ：

20.5.1. 路径参数(2 参数)

Name	描述	默认值	类型
interfaceType	接口类型可以是 eci、esi 或 epi。目前只支持 eci。	eci	java.lang.String
dataExchangeType	使用 Enum 值的数据交换类型： • commarea • channel	commarea	com.redhat.camel.component.cics.support.CICSDataExchangeType

20.5.2. 查询参数(15 参数)

Name	描述	默认值	类型
ctgDebug	在底层 IBM CGT 客户端上启用调试模式。	false	java.lang.Boolean
eciBinding	将 Camel Exchange 转换为 EciRequest 的 Binding 实例，反之亦然		com.redhat.camel.component.cics.CICSEciBinding
eciTimeout	与此 ECIRequest 对象关联的 ECI 超时值。ECI 超时值为零表示此 ECIRequest 不会由 CICS 交易网关超时。大于零的 ECI 超时值表示 CICS 事务网关可能会超时 ECIRequest。在从 CICS 收到响应前，ECI 超时可能会过期。这意味着，客户端不会收到来自 CICS 的确认，因为已支持或提交工作单元。	0	short
编码	在发送前将 COMMAREA 数据转换为编码。	Cp1145	java.lang.String
gatewayFactory	要使用的连接工厂		com.redhat.camel.component.cics.pool.CICSGatewayFactory
主机	此实例连接的 CICS 事务网关的地址	localhost	java.lang.String
port	此实例连接的 CICS 事务网关的端口。	2006	int
protocol	此组件将用于连接到 CICS 事务网关的协议。	tcp	java.lang.String
server	此实例连接的 CICS 服务器的地址		java.lang.String

Name	描述	默认值	类型
lazyStartProducer	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。		布尔值
sslKeyring	用于客户端加密连接的 SSL 密钥环类或密钥存储文件的完整类名称		java.lang.String
sslPassword	加密密钥环类或密钥存储的密码		java.lang.String
socketConnectionTimeout	套接字连接超时		int
password	用于身份验证的密码		java.lang.String
userId	用于身份验证的用户 ID		java.lang.String

20.6. 消息标头

CICS 组件支持 15 个消息标头，如下所列：

Name	描述	默认值	类型
CICS_RETURN_CODE Constant: com.redhat.camel.component.cics.CICSConstants#CICS_RETURN_CODE_HEADER	从此流操作返回代码。		int

Name	描述	默认值	类型
CICS_RETURN_CODE_STRING Constant: com.redhat.camel.component.cics.CICSConstants#CICS_RETURN_CODE_STRING_HEADER	CICS 将代码返回为 String。String 是适当的 Java 常数的名称，例如，如果此标头是 ECI_NO_ERROR，则返回的字符串将是 ECI_NO_ERROR。如果这个标头未知，则返回的字符串将是 ECI_UNKNOWN_CICS_RC。 **注意** 对于 CICS，返回的代码可能有多个意味着返回的字符串是返回代码的串联。唯一的串联字符串是：ECI_ERR_REQUEST_TIMEOUT_OR_ERR_NO_REPLY。		java.ang.String
CICS_EXTEND_MODE Constant: com.redhat.camel.component.cics.CICSConstants#CICS_EXTEND_MODE_HEADER	扩展请求的模式。默认值为 ECI_NO_EXTEND。		int
CICS_LUW_TOKEN Constant: com.redhat.camel.component.cics.CICSConstants#CICS_LUW_TOKEN_HEADER	扩展工作令牌的逻辑单元。默认值为 ECI_LUW_NEW。		int
CICS_PROGRAM_NAME Constant: com.redhat.camel.component.cics.CICSConstants#CICS_PROGRAM_NAME_HEADER	在 CICS 服务器上调用的程序。		java.lang.String

Name	描述	默认值	类型
CICS_TRANSACTION_ID Constant: com.redhat.camel.component.cics.CICSConstants#CICS_TRANSACTION_ID_HEADER	在其下运行 CICS 程序的事务 ID。		java.lang.String
CICS_COMM_AREA_SIZE Constant: com.redhat.camel.component.cics.CICSConstants#CICS_COMM_AREA_SIZE_HEADER	COMMAREA 的长度。默认值为 0。		int
CICS_CHANNEL_NAME Constant: com.redhat.camel.component.cics.CICSConstants#CICS_CHANNEL_NAME_HEADER	创建 com.redhat.camel.component.cics.CICSConstants#CICS_CHANNEL_NAME_HEADER 的频道名称		java.lang.String
CICS_CONTAINER_NAME Constant: com.redhat.camel.component.cics.CICSConstants#CICS_CONTAINER_NAME_HEADER	要创建的容器的名称。		java.lang.String
CICS_CHANNEL_CCSID Constant: com.redhat.camel.component.cics.CICSConstants#CICS_CHANNEL_CCSID_HEADER	频道应设置为其默认值的 CCSID。		int

Name	描述	默认值	类型
CICS_SERVER Constant: com.redhat.camel.component.cics.CICSConstants#CICS_SERVER_HEADER	CICS 服务器将请求定向到.此标头覆盖端点中配置的值。		java.lang.String
CICS_USER_ID Constant: com.redhat.camel.component.cics.CICSConstants#CICS_USER_ID_HEADER	CICS 服务器的用户 ID。此标头覆盖端点中配置的值。		java.lang.String
CICS_PASSWORD Constant: com.redhat.camel.component.cics.CICSConstants#CICS_PASSWORD_HEADER	CICS 服务器的密码或密码短语。此标头覆盖端点中配置的值。		java.lang.String
CICS_ABEND_CODE Constant: com.redhat.camel.component.cics.CICSConstants#CICS_ABEND_CODE_HEADER	CICS 事务处理 abend 代码。		java.lang.String
CICS_ECI_REQUEST_TIMEOUT Constant: com.redhat.camel.component.cics.CICSConstants#CICS_ECI_REQUEST_TIMEOUT_HEADER	当前 ECIRequest 的 ECI 超时的值（以秒为单位）。值为零表示 CICS 事务网关不会超时此 ECIRequest	0	short
CICS_ENCODING Constant: com.redhat.camel.component.cics.CICSConstants#CICS_ENCODING_HEADER	在发送前将 COMMAREA 数据转换为编码。		字符串

20.7. SAMPLES

20.7.1. 使用 Commarea

以下示例演示了如何配置使用 COMMAREA 在 CICS 服务器上运行程序的路由。COMMAREA 大小必须在 CICS_COMM_AREA_SIZE 标头中定义，而 COMMAREA 输入数据在 Camel Exchange 正文中定义。

注意

您必须创建一个足够大的 COMMAREA，使其包含要发送到服务器的所有信息，并足够大，以包含可以从服务器返回的所有信息。

```
//.....
import static
com.redhat.camel.component.cics.CICSConstants.CICS_PROGRAM_NAME_HEADER;
import static
com.redhat.camel.component.cics.CICSConstants.CICS_COMM_AREA_SIZE_HEADER;
//....

from("direct:run").
  setHeader(CICS_PROGRAM_NAME_HEADER, "ECIREADY").
  setHeader(CICS_COMM_AREA_SIZE_HEADER, 18).
  setBody(constant("My input data")).
  to("cics:eci/commarea?
host=192.168.0.23&port=2006&protocol=tcp&userId=foo&password=bar");
```

CICS 程序调用的 Outcome 以这种方式映射到 Camel Exchange：

- 返回代码的数字值保存在 CICS_RETURN_CODE 标头中
- COMMAREA 输出数据存储在 Camel Exchange Body 中。

20.7.2. 使用带有单个输入容器的 Channel

以下示例演示了如何将频道与单个容器一起使用来运行 CICS 程序。频道名称和容器名称来自标头，容器值来自正文：

```
//.....
import static
com.redhat.camel.component.cics.CICSConstants.CICS_PROGRAM_NAME_HEADER;
```

```
import static
com.redhat.camel.component.cics.CICSConstants.CICS_CHANNEL_NAME_HEADER;
import static
com.redhat.camel.component.cics.CICSConstants.CICS_CONTAINER_NAME_HEADER;

//...
from("direct:run").
  setHeader(CICS_PROGRAM_NAME_HEADER, "EC03").
  setHeader(CICS_CHANNEL_NAME_HEADER, "SAMPLECHANNEL").
  setHeader(CICS_CONTAINER_NAME_HEADER, "INPUTDATA").
  setBody(constant("My input data")).
  to("cics:eci/channel?
host=192.168.0.23&port=2006&protocol=tcp&userId=foo&password=bar");
```

返回的容器存储在 java.util.Map<String,Object > 中，键是容器名称，值是容器的输出数据。

20.7.3. 使用带有多个输入容器的 Channel

如果您需要运行一个将多个容器的 CICS 程序作为输入，您可以创建一个 java.util.Map<String,Object& gt;，其中键是容器名称，值是输入数据。在这种情况下，CICS_CONTAINER_NAME 标头将被忽略。

```
//.....
import static
com.redhat.camel.component.cics.CICSConstants.CICS_PROGRAM_NAME_HEADER;
import static
com.redhat.camel.component.cics.CICSConstants.CICS_CHANNEL_NAME_HEADER;

//...
from("direct:run").
  setHeader(CICS_PROGRAM_NAME_HEADER, "EC03").
  setHeader(CICS_CHANNEL_NAME_HEADER, "SAMPLECHANNEL").
  process(exchange->{
    byte[] thirdContainerData =
HexFormat.of().parseHex("e04fd020ea3a6910a2d808002b30309d");
    Map<String,Object> containers = Map.of(
          "firstContainerName", "firstContainerData",
          "secondContainerName", "secondContainerData",
          "thirdContainerName", thirdContainerData
    );
    exchange.getMessage().setBody(containers);
  }).
  to("cics:eci/channel?
host=192.168.0.23&port=2006&protocol=tcp&userId=foo&password=bar");
```

20.8. SPRING BOOT AUTO-CONFIGURATION

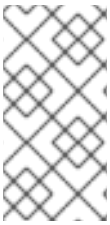
组件支持 17 个选项，如下所列。

Name	描述	默认值	类型
camel.component.cics.binding	将 Camel Exchange 转换为 EciRequest 的 Binding 实例，反之亦然。		com.redhat.camel.component.cics.CICSEciBinding
camel.component.cics.configuration	配置。		com.redhat.camel.component.cics.CICSConfiguration
camel.component.cics.ctg-debug	在底层 IBM CGT 客户端上启用调试模式。		java.lang.Boolean
camel.component.cics.eci-timeout	与此 ECIRequest 对象关联的 ECI 超时值。ECI 超时值为零表示此 ECIRequest 不会由 CICS 交易网关超时。大于零的 ECI 超时值表示 CICS 事务网关可能会超时 ECIRequest。在从 CICS 收到响应前，ECI 超时可能会过期。这意味着，客户端不会收到来自 CICS 的确认，因为已支持或提交工作单元。		java.lang.Short
camel.component.cics.enabled	是否启用 cics 组件的自动配置。这默认是启用的。		java.lang.Boolean
camel.component.cics.encoding	消息的传输编码。	Cp1145	java.lang.String
camel.component.cics.gateway-factory	要使用的连接工厂。选项是一个 com.redhat.camel.component.cics.pool.CICSGatewayFactory 类型。		com.redhat.camel.component.cics.pool.CICSGatewayFactory
camel.component.cics.host	此实例连接的 CICS 事务网关的地址		java.lang.String
camel.component.cics.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。		java.lang.Boolean
camel.component.cics.password	用于身份验证的密码		java.lang.String
camel.component.cics.port	此实例连接的 CICS 事务网关的端口。	2006	java.lang.Integer
camel.component.cics.protocol	此组件将用于连接到 CICS 事务网关的协议。	tcp	java.lang.String

Name	描述	默认值	类型
camel.component.cics.server	此实例连接的 CICS 服务器的地址		java.lang.String
camel.component.cics.socket-connection-timeout	套接字连接超时		java.lang.Integer
camel.component.cics.ssl-keyring	用于客户端加密连接的 SSL 密钥环类或密钥存储文件的完整类名称		java.lang.String
camel.component.cics.ssl-password	加密密钥环类或密钥存储的密码		java.lang.String
camel.component.cics.user-id	用于身份验证的用户 ID		java.lang.String

第 21 章 常数

Constant Expression Language 只是使用恒定值或对象的方法。



注意

这是一个固定的常量值（或对象），只在启动路由时设置一次，如果您希望在路由过程中需要动态值，则不要使用它。

21.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 常量 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-core-starter</artifactId>
</dependency>
```

21.2. 常量选项

Constant 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	设置恒定类型的类名称。
trim		布尔值	是否修剪值以移除前导和结尾的空格和换行符。

21.3. 示例

setHeader EIP 可以使用一个常量表达式，如下所示：

```
<route>
  <from uri="seda:a"/>
  <setHeader name="theHeader">
    <constant>the value</constant>
```

```
  </setHeader>
  <to uri="mock:b"/>
</route>
```

在这种情况下，来自 seda:a 端点的消息将包含标头，并将标头设为 value （字符串 type）。

使用 Java DSL 的同一示例：

```
from("seda:a")
  .setHeader("theHeader", constant("the value"))
  .to("mock:b");
```

21.3.1. 指定值类型

选项 resultType 可以用来指定值类型，当值被指定为 String 值时，会使用 XML 或 YAML DSL 时会出现这种情况：

例如，要设置带有 int 类型的标头，您可以：

```
<route>
  <from uri="seda:a"/>
  <setHeader name="zipCode">
    <constant resultType="int">90210</constant>
  </setHeader>
  <to uri="mock:b"/>
</route>
```

21.4. 从外部资源加载常数

您可以对常量进行外部化，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").constant("resource:classpath:constant.txt")
```

21.5. 依赖项

Constant 语言是 camel-core 的一部分。

21.6. SPRING BOOT AUTO-CONFIGURATION

组件支持 147 选项，如下所列。

Name	描述	默认值	类型
camel.cloud.consul.service-discovery.acl-token	设置用于 Consul 的 ACL 令牌。		字符串
camel.cloud.consul.service-discovery.block-seconds	等待监视事件的秒数，默认为 10 秒。	10	整数
camel.cloud.consul.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.consul.service-discovery.connect-timeout-millis	OkHttpClient 的连接超时。		Long
camel.cloud.consul.service-discovery.datacenter	数据中心。		字符串
camel.cloud.consul.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.consul.service-discovery.password	设置用于基本身份验证的密码。		字符串
camel.cloud.consul.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.consul.service-discovery.read-timeout-millis	OkHttpClient 的读取超时。		Long

Name	描述	默认值	类型
camel.cloud.consul.service-discovery.url	Consul 代理 URL。		字符串
camel.cloud.consul.service-discovery.user-name	设置用于基本身份验证的用户名。		字符串
camel.cloud.consul.service-discovery.write-timeout-millis	OkHttpClient 的写入超时。		Long
camel.cloud.dns.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.dns.service-discovery.domain	域名；		字符串
camel.cloud.dns.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.dns.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.dns.service-discovery.proto	所需服务的传输协议。	_tcp	字符串
camel.cloud.etcd.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.etcd.service-discovery.enabled	启用组件。	true	布尔值

Name	描述	默认值	类型
camel.cloud.etcd.service-discovery.password	用于基本身份验证的密码。		字符串
camel.cloud.etcd.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.etcd.service-discovery.service-path	查找服务发现的路径。	/services/	字符串
camel.cloud.etcd.service-discovery.timeout	要设置操作可以采取的最长时间，请执行以下操作：		Long
camel.cloud.etcd.service-discovery.type	要设置发现类型，有效值为 on-demand 和 watch。	按需	字符串
camel.cloud.etcd.service-discovery.uris	客户端可以连接到的 URI。		字符串
camel.cloud.etcd.service-discovery.user-name	用于基本身份验证的用户名。		字符串
camel.cloud.kubernetes.service-discovery.api-version	使用客户端查找时设置 API 版本。		字符串
camel.cloud.kubernetes.service-discovery.ca-cert-data	使用客户端查找时设置证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.ca-cert-file	在使用客户端查找时，设置从文件加载的证书颁发机构数据。		字符串

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.client-cert-data	使用客户端查找时设置客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-file	在使用客户端查找时，设置从文件加载的客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-algo	设置客户端密钥存储算法，如使用客户端查找时 RSA。		字符串
camel.cloud.kubernetes.service-discovery.client-key-data	使用客户端查找时设置客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-file	在使用客户端查找时，设置从文件加载的客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-passphrase	使用客户端查找时设置客户端密钥存储密码短语。		字符串
camel.cloud.kubernetes.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.kubernetes.service-discovery.dns-domain	设置用于 DNS 查找的 DNS 域。		字符串
camel.cloud.kubernetes.service-discovery.enabled	启用组件。	true	布尔值

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.lookup	如何执行服务查找。可能的值有：client、dns、environment。在使用客户端时，客户端会查询 kubernetes master 来获取提供该服务的活跃 pod 列表，然后随机（或循环）选择一个 pod。当使用 dns 时，服务名称被解析为 name.namespace.svc.dnsDomain。当使用 dnssrv 时，服务名称使用 SRV 查询解析 ....svc... when using environment，环境变量用于查找服务。默认情况下使用环境。	环境	字符串
camel.cloud.kubernetes.service-discovery.master-url	在使用客户端查找时，将 URL 设置为 master。		字符串
camel.cloud.kubernetes.service-discovery.namespace	设置要使用的命名空间。默认情况下，将使用来自 ENV 变量 KUBERNETES_MASTER 的命名空间。		字符串
camel.cloud.kubernetes.service-discovery.oauth-token	在使用客户端查找时，为身份验证设置 OAUTH 令牌（而不是用户名/密码）。		字符串
camel.cloud.kubernetes.service-discovery.password	在使用客户端查找时设置用于身份验证的密码。		字符串
camel.cloud.kubernetes.service-discovery.port-name	设置用于 DNS/DNSSRV 查找的端口名称。		字符串
camel.cloud.kubernetes.service-discovery.port-protocol	设置用于 DNS/DNSSRV 查找的端口协议。		字符串
camel.cloud.kubernetes.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.trust-certs	设置在使用客户端查找时是否打开信任证书检查。	false	布尔值
camel.cloud.kubernetes.service-discovery.username	在使用客户端查找时设置用于身份验证的用户名。		字符串
camel.cloud.ribbon.load-balancer.client-name	设置 Ribbon 客户端名称。		字符串
camel.cloud.ribbon.load-balancer.configurations	定义其他配置定义。		Map
camel.cloud.ribbon.load-balancer.enabled	启用组件。	true	布尔值
camel.cloud.ribbon.load-balancer.namespace	命名空间。		字符串
camel.cloud.ribbon.load-balancer.password	密码。		字符串
camel.cloud.ribbon.load-balancer.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.ribbon.load-balancer.username	用户名。		字符串
camel.hystrix.allow-maximum-size-to-diverge-from-core-size	允许配置使 maximumSize 生效。然后该值可以等于或大于 coreSize。	false	布尔值

Name	描述	默认值	类型
camel.hystrix.circuit-breaker-enabled	是否使用 HystrixCircuitBreaker。如果为 false，则不会使用 断路器逻辑，并且所有允许的请求。这与 circuitBreakerForceClosed （）的影响类似，除非继续跟踪指标，知道它是否应该是 open/closed，此属性即使实例化一个断路器。	true	布尔值
camel.hystrix.circuit-breaker-error-threshold-percentage	错误百分比阈值（如 50 ）指向断路器将打开和拒绝请求。它将在 circuitBreakerSleepWindowInMilliseconds 中定义的持续时间保持出差；与 HystrixCommandMetrics.getHealthCounts （）进行比较的错误百分比。	50	整数
camel.hystrix.circuit-breaker-force-closed	如果为 true，HystrixCircuitBreaker#allowRequest （）将始终返回 true 以允许请求，无论 HystrixCommandMetrics.getHealthCounts （）的错误百分比如何。如果设为 true，则 circuitBreakerForceOpen （）属性具有优先权。	false	布尔值
camel.hystrix.circuit-breaker-force-open	如果为 true，HystrixCircuitBreaker.allowRequest （）将始终返回 false，从而导致电路变为开路（接受），并拒绝所有请求。此属性优先于 circuitBreakerForceClosed （）;。	false	布尔值
camel.hystrix.circuit-breaker-request-volume-threshold	metricsRollingStatisticalWindowInMilliseconds （）中的最少请求数必须存在于 HystrixCircuitBreaker 之前。如果此数字低于这个数字，无论错误百分比如何，电路都不会被出差。	20	整数
camel.hystrix.circuit-breaker-sleep-window-in-milliseconds	HystrixCircuitBreaker trips 之后的时间（以毫秒为单位），它应该在尝试请求前等待。	5000	整数
camel.hystrix.configurations	定义其他配置定义。		Map
camel.hystrix.core-pool-size	传递给 java.util.concurrent.ThreadPoolExecutor#setCorePoolSize (int)的核心 thread-pool 大小。	10	整数
camel.hystrix.enabled	启用组件。	true	布尔值
camel.hystrix.execution-isolation-semaphore-max-concurrent-requests	允许 HystrixCommand.run （）的并发请求数。超过并发限制的请求将被拒绝。仅在执行IsolationStrategy == SEMAPHORE 时使用。	20	整数

Name	描述	默认值	类型
camel.hystrix.execution-isolation-strategy	将通过什么隔离策略 HystrixCommand.run （）执行。如果 THREAD，它将在单独的线程上执行，并且受 thread-pool 中的线程数量限制的并发请求。如果 SEMAPHORE，它将在调用线程上执行，并且受 semaphore 数限制的并发请求。	线程	字符串
camel.hystrix.execution-isolation-thread-interrupt-on-timeout	当线程超时时，执行线程是否应该尝试中断（使用 future#cancel）。仅在执行IsolationStrategy （）== THREAD 时才适用。	true	布尔值
camel.hystrix.execution-timeout-enabled	此命令是否启用了超时机制。	true	布尔值
camel.hystrix.execution-timeout-in-milliseconds	以毫秒为单位，将命令超时和停止执行的时间（以毫秒为单位）。如果 executionIsolationThreadInterruptOnTimeout == true 且命令是线程隔离，则执行线程将中断。如果命令是 semaphore-isolated 和 HystrixObservableCommand，则该命令将被取消订阅。	1000	整数
camel.hystrix.fallback-enabled	出现故障时，是否应尝试 HystrixCommand.getFallback （）。	true	布尔值
camel.hystrix.fallback-isolation-semaphore-max-concurrent-requests	允许 HystrixCommand.getFallback （）的并发请求数。超过并发限制的请求将快速失败，且不会尝试检索回退。	10	整数
camel.hystrix.group-key	设置要使用的 group 键。默认值为 CamelHystrix。	Camel Hystrix	字符串
camel.hystrix.keep-alive-time	更长的时间（以分钟为单位）传递给 ThreadPoolExecutor#setKeepAliveTime (long,TimeUnit)。	1	整数
camel.hystrix.max-queue-size	在 HystrixConcurrencyStrategy.getBlockingQueue (int)中传递给 BlockingQueue 的最大队列大小应该只影响 threadpool 的实例化 - 它不会立即更改队列大小。为此，请使用 queueSizeRejectionThreshold （）。	-1	整数

Name	描述	默认值	类型
camel.hystrix.maximum-size	传递给 ThreadPoolExecutor#setMaximumPoolSize(int)的最大 thread-pool 大小。这是可在不开始拒绝 HystrixCommands 的情况下支持的最大并发数量。请注意，只有在您也设置了 allowMaximumSizeToDivergeFromCoreSize 时，此设置才会生效。	10	整数
camel.hystrix.metrics-health-snapshot-interval-in-milliseconds	在允许计算成功和错误百分比时等待的时间（以毫秒为单位），并影响 HystrixCircuitBreaker.isOpen （）状态。在高容量电路上，错误百分比的连续计算可能会成为 CPU 密集型，从而控制其计算的频率。	500	整数
camel.hystrix.metrics-rolling-percentile-bucket-size	滚动百分比的每个存储桶中存储的最大值数。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	10	整数
camel.hystrix.metrics-rolling-percentile-enabled	是否应该使用 HystrixRollingPercentile 内部 HystrixCommandMetrics 来捕获百分比的指标。	true	布尔值
camel.hystrix.metrics-rolling-percentile-window-buckets	滚动窗口的存储桶数量被分成。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	6	整数
camel.hystrix.metrics-rolling-percentile-window-in-milliseconds	以毫秒为单位的滚动窗口的持续时间。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	10000	整数
camel.hystrix.metrics-rolling-statistical-window-buckets	滚动统计窗口划分为的 bucket 数量。这在 HystrixCommandMetrics 中被传递给 HystrixRollingNumber。	10	整数
camel.hystrix.metrics-rolling-statistical-window-in-milliseconds	此属性设置统计滚动窗口的持续时间，以毫秒为单位。这是为线程池保留指标的时间。窗口被分成 bucket，按这些增量回滚。	10000	整数
camel.hystrix.queue-size-rejection-threshold	队列大小拒绝阈值是 artificial max size，即使尚未达到 maxQueueSize，也会发生拒绝。这是因为 BlockingQueue 的 maxQueueSize 无法动态更改，我们希望动态更改影响拒绝的队列大小。在排队线程以进行执行时，HystrixCommand 会使用它。	5	整数

Name	描述	默认值	类型
camel.hystrix.request-log-enabled	HystrixCommand 执行和事件是否应记录到 HystrixRequestLog。	true	布尔值
camel.hystrix.thread-pool-key	设置要使用的线程池密钥。默认情况下，将使用与 groupKey 配置相同的值。	Camel Hystrix	字符串
camel.hystrix.thread-pool-rolling-number-statistical-window-buckets	滚动统计窗口划分为的 bucket 数量。这会传递到每个 HystrixThreadPoolMetrics 实例内的 HystrixRollingNumber。	10	整数
camel.hystrix.thread-pool-rolling-number-statistical-window-in-milliseconds	统计滚动窗口的持续时间（以毫秒为单位）。这会传递到每个 HystrixThreadPoolMetrics 实例内的 HystrixRollingNumber。	10000	整数
camel.language.constant.enabled	是否启用恒定语言的自动配置。这默认是启用的。		布尔值
camel.language.constant.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.csimple.enabled	是否启用 csimple 语言的自动配置。这默认是启用的。		布尔值
camel.language.csimple.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.exchangeproperty.enabled	是否启用 exchangeProperty 语言的自动配置。这默认是启用的。		布尔值
camel.language.exchangeproperty.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.file.enabled	是否启用文件语言的自动配置。这默认是启用的。		布尔值
camel.language.file.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.header.enabled	是否启用标头语言的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.language.header.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.ref.enabled	是否启用 ref 语言的自动配置。这默认是启用的。		布尔值
camel.language.ref.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.simple.enabled	是否启用简单语言的自动配置。这默认是启用的。		布尔值
camel.language.simple.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.tokenize.enabled	是否启用令牌化语言的自动配置。这默认是启用的。		布尔值
camel.language.tokenize.group-delimiter	设置在分组时要使用的分隔符。如果没有设置，则令牌将用作分隔符。		字符串
camel.language.tokenize.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.resilience4j.automatic-transition-from-open-to-half-open-enabled	在通过 waitDurationInOpenState 后，启用从 OPEN 自动过渡到 HALF_OPEN 状态。	false	布尔值
camel.resilience4j.circuit-breaker-ref	代表现有的 io.github.resilience4j.circuitbreaker.CircuitBreaker 实例从 registry 中查找和使用。使用此选项时，不使用任何其他断路器选项。		字符串
camel.resilience4j.config-ref	指的是现有的 io.github.resilience4j.circuitbreaker.CircuitBreakerConfig 实例，以便从 registry 中查找和使用。		字符串
camel.resilience4j.configurations	定义其他配置定义。		Map
camel.resilience4j.enabled	启用组件。	true	布尔值

Name	描述	默认值	类型
camel.resilience4j.failure-rate-threshold	以百分比为单位配置故障率阈值。如果失败率相等或大于阈值，则 CircuitBreaker 过渡到 open，并启动短路调用。阈值必须大于 0，而不是大于 100。默认值为 50 百分比。		æµ®ç‚¹å€¼
camel.resilience4j.minimum-number-of-calls	在 CircuitBreaker 可以计算错误率之前，配置所需的最少调用数（每个滑动期限）。例如，如果 minimumNumberOfCalls 为 10，则必须至少记录 10 个调用，然后才能计算失败率。如果只记录了 9 个调用，则 CircuitBreaker 不会过渡到 open，即使所有 9 调用都失败。默认 minimumNumberOfCalls 为 100。	100	整数
camel.resilience4j.permitted-number-of-calls-in-half-open-state	配置 CircuitBreaker 为一半打开时允许的调用数量。大小必须大于 0。默认大小为 10。	10	整数
camel.resilience4j.sliding-window-size	配置滑动窗口的大小，该窗口用于在 CircuitBreaker 关闭时记录调用的结果。slidingWindowSize 配置滑动窗口的大小。滑动窗口可以是基于计数或基于时间的窗口。如果 slidingWindowType 是 COUNT_BASED，则最后的 slidingWindowSize 调用会被记录并聚合。如果 slidingWindowType 是 TIME_BASED，则最后 slidingWindowSize 秒的调用会被记录并聚合。slidingWindowSize 必须大于 0。minimumNumberOfCalls 必须大于 0。如果 slidingWindowType 是 COUNT_BASED，则 minimumNumberOfCalls 不能大于 slidingWindowSize。如果 slidingWindowType 是 TIME_BASED，您可以选择任何您需要的。默认 slidingWindowSize 为 100。	100	整数
camel.resilience4j.sliding-window-type	配置滑动窗口的类型，用于记录 CircuitBreaker 关闭时调用的结果。滑动窗口可以是基于计数或基于时间的窗口。如果 slidingWindowType 是 COUNT_BASED，则最后的 slidingWindowSize 调用会被记录并聚合。如果 slidingWindowType 是 TIME_BASED，则最后 slidingWindowSize 秒的调用会被记录并聚合。默认 slidingWindowType 是 COUNT_BASED。	COUNT_BASED	字符串
camel.resilience4j.slow-call-duration-threshold	配置上面的持续时间阈值（秒），调用被视为缓慢，并增加较慢的调用百分比。默认值为 60 秒。	60	整数

Name	描述	默认值	类型
camel.resilience4j.slow-call-rate-threshold	以百分比为单位配置阈值。当调用持续时间大于 slowCallDurationThreshold Duration 时，CircuitBreaker 会将调用视为较慢。当较慢的调用百分比相等或大于阈值时，CircuitBreaker 过渡到 open，并启动短路调用。阈值必须大于 0，而不是大于 100。默认值为 100 百分比，这意味着所有记录的调用都必须比 slowCallDurationThreshold 慢。		æµ®ç‚¹å€¼
camel.resilience4j.wait-duration-in-open-state	配置等待持续时间（以秒为单位），指定 CircuitBreaker 应该保持打开的时间，然后再切换到半次。默认值为 60 秒。	60	整数
camel.resilience4j.writable-stack-trace-enabled	启用可写入堆栈跟踪。当设置为 false 时，Exception.getStackTrace 返回一个零长度数组。当断路器处于开路状态时，这可用于减少日志垃圾邮件，因为存在例外的原因（断路器是短路调用）。	true	布尔值
camel.rest.api-component	用作 REST API 的 Camel 组件名称（如 swagger）如果没有明确配置 API 组件，则 Camel 会查找负责服务并生成 REST API 文档的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestApiProcessorFactory。如果找到其中任何一个，则使用它。		字符串
camel.rest.api-context-path	设置领导的 API 上下文路径将使用的 REST API 服务。这在使用使用 context-path 部署 Web 应用程序的组件（如 camel-servlet ）时使用它。		字符串
camel.rest.api-context-route-id	设置用于服务 REST API 的路由的路由 ID。默认情况下，路由将使用自动分配的路由 ID。		字符串
camel.rest.api-host	要将特定主机名用于 API 文档（如 swagger），这可用于用这个配置的主机名覆盖生成的主机。		字符串
camel.rest.api-property	允许为 api 文档配置任意数量的附加属性(swagger)。例如，将属性 api.title 设置为我的冷却。		Map
camel.rest.api-vendor-extension	是否在 Rest API 中启用供应商扩展。如果启用，Camel 将包含额外信息作为厂商扩展名（例如，以 x- 开头的键），如路由 ID、类名称等。在导入 API 文档时，并非所有第三方 API 网关和工具都支持 vendor-extensions。	false	布尔值
camel.rest.binding-mode	设置要使用的绑定模式。默认值为 off。		RestBindingMode

Name	描述	默认值	类型
camel.rest.client-request-validation	是否启用客户端请求验证，以检查客户端的 Content-Type 和 Accept 标头是否受到其 consume/produces 设置的 Rest-DSL 配置的支持。这可以打开，以启用此检查。如果验证错误，则返回 HTTP Status code 415 或 406。默认值为 false。	false	布尔值
camel.rest.component	用于 REST 传输(consumer)的 Camel Rest 组件，如 netty-http, jetty, servlet, undertow。如果没有明确配置组件，则 Camel 会查找是否有与 Rest DSL 集成的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestConsumerFactory。如果找到其中任何一个，则使用它。		字符串
camel.rest.component-property	允许为正在使用的其他组件配置任意数量的附加属性。		Map
camel.rest.consumer-property	允许为使用中的其他使用者配置任意数量的附加属性。		Map
camel.rest.context-path	设置 REST 服务将使用的前导上下文路径。这在使用使用 context-path 部署 Web 应用程序的组件（如 camel-servlet ）时使用它。或者对于包含 HTTP 服务器的 camel-jetty 或 camel-netty-http 等组件。		字符串
camel.rest.cors-headers	允许配置自定义 CORS 标头。		Map
camel.rest.data-format-property	允许为使用的数据格式配置多个额外属性。例如，将属性 prettyPrint 设置为 true，以便以用户友善模式输出 json。属性可以加上前缀来表示选项仅适用于 JSON 或 XML，以及 IN 或 OUT。前缀为： json.in. json.out. xml.in. xml.out。例如，值为 xml.out.mustBeJAXBElement 的键仅用于传出的 XML 数据格式。没有前缀的密钥是所有情况的通用密钥。		Map
camel.rest.enable-cors	是否在 HTTP 响应中启用 CORS 标头。默认值为 false。	false	布尔值
camel.rest.endpoint-property	允许为使用中的其他端点配置多个额外的属性。		Map
camel.rest.host	用于公开 REST 服务的主机名。		字符串
camel.rest.host-name-resolver	如果没有明确配置的主机名，这个 resolver 会用于计算 REST 服务将要使用的主机名。		RestHostNameResolver

Name	描述	默认值	类型
camel.rest.json-data-format	要使用的特定 json 数据格式的名称。默认将使用 json-jackson。重要：此选项仅用于设置数据格式的自定义名称，而不是引用现有数据格式实例。		字符串
camel.rest.port	用于公开 REST 服务的主机名。请注意，如果您使用 servlet 组件，则此处配置的端口号不适用，因为使用中的端口号是 servlet 组件使用的实际端口号。例如，如果使用 Apache Tomcat，它的 tomcat http 端口，如果使用 Apache Karaf，它的在 Karaf 中的 HTTP 服务，它默认使用端口 8181。虽然在这些情况下，这里设置端口号，但允许工具和 JMX 知道端口号，因此建议将端口号设置为 servlet 引擎使用的数字。		字符串
camel.rest.producer-api-doc	设置 api 文档的位置，REST 生成者将根据这个文档来验证 REST uri 和查询参数是否有效。这需要将 camel-swagger-java 添加到 classpath 中，任何缺失的配置都会导致 Camel 在启动时失败并报告错误。默认情况下从 classpath 加载的 api 文档的位置，但您可以使用 file: 或 http: 引用从文件或 http url 加载的资源。		字符串
camel.rest.producer-component	设置要用作 REST 生成者的 Camel 组件的名称。		字符串
camel.rest.scheme	用于公开 REST 服务的方案。通常支持 http 或 https。默认值为 http。		字符串
camel.rest.skip-binding-on-error-code	如果存在自定义 HTTP 错误代码标头，是否跳过输出绑定。这允许构建没有绑定到 json / xml 等自定义错误消息，否则成功信息会这样做。	false	布尔值
camel.rest.use-x-forward-headers	是否将 X-Forward 标头用于主机和相关设置。默认值为 true。	true	布尔值
camel.rest.xml-data-format	要使用的特定 XML 数据格式的名称。默认情况下将使用 jaxb。重要：此选项仅用于设置数据格式的自定义名称，而不是引用现有数据格式实例。		字符串
camel.rest.api-context-id-pattern	**弃用** 设置 CamelContext id 特征，以只允许 CamelContext 中名称与特征匹配的其他服务的 Rest API。特征 name 指的是 CamelContext 名称，仅匹配当前的 CamelContext。对于任何其他值，特征使用来自 PatternHelper#matchPattern (String,String)的规则。		字符串
camel.rest.api-context-listing	**弃用** 设置是否启用了 JVM 中带有 REST 服务的所有可用 CamelContext 的列表。如果启用，它将允许发现这些上下文，如果为 false，则只使用当前的 CamelContext。	false	布尔值

第 22 章 控制总线

仅支持生成者

来自 EIP 模式的控制总线允许从框架内监控和管理集成系统。

使用控制总线管理企业集成系统。Control Bus 使用与应用程序数据相同的消息传递机制，但使用单独的频道传输与消息流中涉及的组件管理相关的数据。

在 Camel 中，您可以使用 JMX 来管理和监控，或者使用来自 CamelContext 的 Java API，或者从 org.apache.camel.api.management 软件包，或使用此处示例的事件通知程序。

ControlBus 组件根据控制总线 EIP 模式简化 Camel 应用程序管理。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 可以是任意字符串来标识要使用的命令类型。

22.1. 命令

命令	描述
route	使用 **routeId** 和 **action** 参数控制路由。
language	允许您指定 a 用于评估邮件正文。如果评估中有任何结果，则结果将置于邮件正文中。

22.2. 依赖项

当在 Camel Spring Boot 中使用 controlbus 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-controlbus-starter</artifactId>
</dependency>
```

22.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

22.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

22.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

22.4. 组件选项

Control Bus 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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

22.5. 端点选项

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：

22.5.1. 路径参数(2 参数)

Name	描述	默认值	类型
command (producer)	**必需** 的命令可以是 route 或 language。 Enum 值： • route • language		字符串

Name	描述	默认值	类型
language (producer)	允许您指定用于评估邮件正文的语言名称。如果评估中有任何结果，则结果将置于邮件正文中。 Enum 值： • Bean • constant • el • exchangeProperty • file • groovy • header • jsonpath • mvel • OGNL • ref • simple • spel • sql • terser • tokenize • XPath • xquery • xtokenize		语言

22.5.1.1. 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)	表示可以是：start、stop 或 status 的操作。要启动或停止路由，或者以消息正文中的输出形式获取路由的状态。您可以使用 suspend 并从 Camel 2.11.1 恢复，以挂起或恢复路由。从 Camel 2.11.1 开始，您可以使用 stats 获取以 XML 格式返回的性能静态；routeId 选项可用于定义哪个路由来获取性能统计数据，如果未定义 routeId，则获取整个 CamelContext 的统计信息。restart 操作将重启路由。 Enum 值： • 开始 • stop • suspend • resume • restart • status • stats		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，则在 Exchange 上不会设置任务的任何结果。这只有在同步执行任务时才可能。	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
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发生异常。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	INFO	LoggingLevel

Name	描述	默认值	类型
restartDelay (producer)	重启路由时要使用的延迟。	1000	int
routeId (producer)	通过其 id 指定路由。special 关键字 current 表示当前路由。		字符串

22.6. 使用 ROUTE 命令

route 命令允许您在给定路由上非常轻松地执行常见任务，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：

```
String status = template.requestBody("controlbus:route?routeId=foo&action=status", null,
String.class);
```

22.7. 获取性能统计信息

这要求启用 JMX （默认为），然后您可以获取每个路由或 CamelContext 的性能统计信息。例如，要获取名为 foo 的路由的统计信息，我们可以执行以下操作：

```
String xml = template.requestBody("controlbus:route?routeId=foo&action=stats", null,
String.class);
```

返回的统计信息采用 XML 格式。其相同的数据，您可以在 ManagedRouteMBean 上通过 dumpRouteStatsAsXml 操作从 JMX 获取。

要获取整个 CamelContext 的统计信息，只需省略 routeId 参数，如下所示：

```
String xml = template.requestBody("controlbus:route?action=stats", null, String.class);
```

22.8. 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.getRouteController().stopRoute('myRoute')}");
```

由于这是一个 void 操作，因此不会返回任何结果。但是，如果您希望路由状态，您可以：

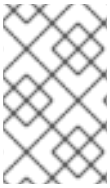
```
String status = template.requestBody("controlbus:language:simple",
"${camelContext.getRouteStatus('myRoute')}", String.class);
```

使用 route 命令控制路由的生命周期更为容易。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：

```
template.sendBody("controlbus:language:simple?async=true", "${camelContext.stop()}");
```

我们使用 async=true 异步停止 Camel，否则我们将试图在处理我们发送到控制总线组件的消息时停止 Camel。



注意

您还可以使用其他语言，如 Groovy 等。

22.9. 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.controlbus.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.controlbus.enabled	是否启用 controlbus 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.controlbus.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

第 23 章 CRON

仅支持消费者

Cron 组件是一个通用接口组件，它允许以特定时间间隔触发事件，使用 Unix cron 语法（如 0/2 * ? ）来每两秒触发事件。

是一个接口组件，Cron 组件不包含默认的实现，而是要求用户插件自己选择的实现。

以下标准 Camel 组件支持 Cron 端点：

- camel-quartz
- camel-spring

Camel K 中也支持 Cron 组件，该组件可以使用 Kubernetes 调度程序在 cron 表达式需要时触发路由。当使用与 Kubernetes cron 语法兼容的 cron 表达式时，Camel K 不需要插入额外的库。

23.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 cron 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-cron-starter</artifactId>
</dependency>
```

为了插入特定实施，可能需要其他库。

23.2. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

23.2.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

23.2.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

23.3. 组件选项

Cron 组件支持 3 个选项，如下所列。

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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
cronService (advanced)	提供多个实现时要使用的 CamelCronService 的 id。		字符串

23.4. 端点选项

Cron 端点使用 URI 语法进行配置：

```
cron:name
```

使用以下路径和查询参数：

23.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
name (consumer)	**必需** cron 触发器的名称。		字符串

23.4.2. 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
schedule (consumer)	**必需** 用于生成事件的 cron 表达式。		字符串
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern

23.5. 使用方法

组件可用于在指定时间触发事件，如下例所示：

```
from("cron:tab?schedule=0/1+*+*+*+*+?")
.setBody().constant("event")
.log("${body}");
```

调度表达式 0/3+10+*+? 也可以写为 0/3 10 * * * ?，并只会在每小时的十分钟内每三秒触发一次事件。

调度表达式中的部分表示（按顺序）：

- **秒（可选）**

- 分钟
- 小时
- 几天
- 月
- 星期几
- 年（可选）

调度表达式可由 5 到 7 个部分组成。当表达式由 6 个部分组成时，第一个项目是"秒"部分（及年份被视为缺失）。

调度表达式的其他有效示例包括：

- 0/2 * * * ?(5 个部分，每两分钟一个事件)
- 0 0/2 * * * MON-FRI 2030 (7 个部分，每两分钟一个事件，仅 2030 年)

路由也可以使用 XML DSL 编写。

```
<route>
  <from uri="cron:tab?schedule=0/1+*+*+*+*+?"/>
  <setBody>
    <constant>event</constant>
  </setBody>
  <to uri="log:info"/>
</route>
```

23.6. 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.cron.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.cron.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.cron.cron-service	提供多个实现时要使用的 CamelCronService 的 id。		字符串
camel.component.cron.enabled	是否启用 cron 组件的自动配置。这默认是启用的。		布尔值

第 24 章 CRYPTO (JCE)

Since Camel 2.3

仅支持生成者

使用 Camel 加密端点和 Java Cryptographic 扩展时，可轻松为交换创建数字签名。Camel 提供了一对灵活的端点，用于 concert 用于在交换工作流的一个部分中为交换创建签名，然后在工作流的后续部分中验证签名。

24.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 crypto 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-crypto-starter</artifactId>
</dependency>
```

24.2. 简介

数字签名使用 Asymmetric Cryptographic 技术为消息签名。从高级别上，算法使用 complimentary 密钥对以及使用一个密钥加密的特殊属性的特殊属性，只能相互解密。私钥 紧密保护并用于"签署"消息，而其他公钥则共享给有兴趣验证签名邮件的任何人。消息使用私钥签名，以加密消息摘要。这个加密摘要会与消息一起传输。在其他一端，verifier 会重新计算消息摘要，并使用公钥解密签名中的摘要。如果两个摘要都匹配，verifier 知道私钥的拥有者只能创建签名。

Camel 使用 Java Cryptographic 扩展中的 Signature 服务来执行创建交换签名所需的所有重度加密机制。以下是解释 Cryptography、Message 摘要和数字签名的机制以及如何通过 JCE 使用它们的资源。

- Bruce Schneier 的 Applied Cryptography
- 开始使用 Java 被 David Hook 开始

- 过时的 Wikipedia Digital_signatures

24.3. URI 格式

Camel 提供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 "CamelDigitalSignature"）的标头中。

- crypto:verify 读取此标头的内容，并执行验证计算。

要正常工作，签名和验证过程需要共享一对密钥，签名需要 PrivateKey 并验证公共密钥（或包含 证书）。使用 JCE 时，生成这些密钥对非常简单，但通常最好使用 KeyStore 来托管和共享您的密钥。DSL 对如何提供密钥并提供了多个机制非常灵活。

crypto:sign 端点通常在一个路由中定义，在另一个路由中包括 complimentary crypto:verify，但对于简单，它们出现在另一个路由后显示的示例。签名和验证应相同配置。

24.4. 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

24.4.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

24.4.2. 配置端点选项

端点有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点直接在端点 URI 中作为路径和查询参数完成。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式 *的安全* 方法。

使用 Property Placeholders 配置不允许硬编码 URL、端口号、敏感信息和其他设置的选项。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

24.5. 组件选项

Crypto (JCE)组件支持以下列出的 21 个选项。

Name	描述	默认值	类型
algorithm (producer)	设置用于签名者的算法的 JCE 名称。	SHA256withRSA	字符串
alias (producer)	设置用于查询键和 \\{link java.security.cert.Certificate Certificates} 的别名，以签名和验证交换。此值可以在运行时通过消息标头 org.apache.camel.component.crypto.DigitalSignatureConstants#KEYSTORE_ALIAS 提供。		字符串

Name	描述	默认值	类型
certificateName (producer)	为 registry 中找到的 PrivateKey 设置引用名称。		字符串
keystore (producer)	设置可包含用于签名和验证交换的密钥和证书的 KeyStore。KeyStore 通常与别名一起使用，可以是在 Route 定义中提供的，或者通过消息标头 CamelSignatureKeyStoreAlias 进行动态使用。如果没有提供别名，且密钥存储中只有一个条目，则将使用此单个条目。		KeyStore
keystoreName (producer)	为 registry 中找到的密钥存储设置引用名称。		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
privateKey* (producer)	设置用于为交换签名的 PrivateKey。		PrivateKey
privateKeyName (producer)	为 registry 中找到的 PrivateKey 设置引用名称。		字符串
provider (producer)	设置提供配置的 Signature 算法的安全供应商的 id。		字符串
publicKeyName (producer)	在上下文更改时应解析的引用。		字符串

Name	描述	默认值	类型
secureRandomName (producer)	为 registry 中找到的 SecureRandom 设置引用名称。		字符串
signatureHeaderName (producer)	设置应该用于存储 base64 编码签名的消息标头名称。默认值为 'CamelDigitalSignature' 。		字符串
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
bufferSize (advanced)	设置用于在 Exchange 有效负载数据中读取的缓冲区的大小。	2048	整数
证书 (advanced)	设置应用来根据其有效负载验证交换中的签名的证书。		证书
clearHeaders (advanced)	确定签名和验证后是否清除了签名特定标头。默认为 true，且应只在每个情况下进行其他影响，因为如果未设置，密钥和密码等重要私有信息可能会转义。	true	布尔值
configuration (advanced)	使用 shared DigitalSignatureConfiguration 作为配置。		DigitalSignatureConfiguration

Name	描述	默认值	类型
keyStoreParameters (advanced)	根据给定的 KeyStoreParameters 设置可包含用于签名和验证交换的密钥和 Certficates 的 KeyStore。KeyStore 通常与别名一起使用，可以是在 Route 定义中提供的，或者通过消息标头 CamelSignatureKeyStoreAlias 进行动态使用。如果没有提供别名，且密钥存储中只有一个条目，则将使用此单个条目。		KeyStoreParameters
Publickey (advanced)	设置用于在交换中验证签名的 PublicKey。		PublicKey
SecureRandom (advanced)	设置用于初始化 Signature 服务的 SecureRandom。		SecureRandom
password (security)	设置用于访问 KeyStore 中别名的 PrivateKey 的密码。		字符串

24.6. 端点选项

Crypto (JCE)端点使用 URI 语法进行配置：

```
crypto:cryptoOperation:name
```

以下是 path 和 查询参数：

24.6.1. 路径参数(2 参数)

Name	描述	默认值	类型

Name	描述	默认值	类型
cryptoOperation (producer)	**必需** 在 endpoint uri e.g. crypto:sign 设置签名为操作中的加密方案后提供的 Crypto 操作。 Enum 值： * 符号 * 验证		CryptoOperation
name (producer)	**需要** 此操作的逻辑名称。		字符串

24.6.2. 查询参数(19 参数)

Name	描述	默认值	类型
algorithm (producer)	设置用于签名者的算法的 JCE 名称。	SHA256withRSA	字符串
alias (producer)	设置用于查询键和 \\{link java.security.cert.Certificate Certificates} 的别名，以签名和验证交换。此值可以在运行时通过消息标头 org.apache.camel.component.crypto.DigitalSignatureConstants#KEYSTORE_ALIAS 提供。		字符串
certificateName (producer)	为 registry 中找到的 PrivateKey 设置引用名称。		字符串
keystore (producer)	设置可包含用于签名和验证交换的密钥和 Certficates 的 KeyStore。KeyStore 通常与别名一起使用，可以是在 Route 定义中提供的，或者通过消息标头 CamelSignatureKeyStoreAlias 进行动态使用。如果没有提供别名，且密钥存储中只有一个条目，则将使用此单个条目。		KeyStore

Name	描述	默认值	类型
keystoreName (producer)	为 registry 中找到的密钥存储设置引用名称。		字符串
privatekey (producer)	设置用于为交换签名的 PrivateKey。		PrivateKey
privateKeyName (producer)	为 registry 中找到的 PrivateKey 设置引用名称。		字符串
provider (producer)	设置提供配置的 Signature 算法的安全供应商的 id。		字符串
publicKeyName (producer)	在上下文更改时应解析的引用。		字符串
secureRandomName (producer)	为 registry 中找到的 SecureRandom 设置引用名称。		字符串
signatureHeaderName (producer)	设置应该用于存储 base64 编码签名的消息标头名称。默认值为 'CamelDigitalSignature' 。		字符串
lazyStartProducer (producer (advanced))	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。通过延迟启动，启动失败可以在路由消息时使用 Camel 路由错误处理程序进行处理。请注意，当处理第一个消息时，创建并启动制作者可能需要稍等片刻，并延长总处理时间。	false	布尔值
bufferSize (advanced)	设置用于在 Exchange 有效负载数据中读取的缓冲区的大小。	2048	整数

Name	描述	默认值	类型
证书 (advanced)	设置应用来根据其有效负载验证交换中的签名的证书。		证书
clearHeaders (advanced)	确定签名和验证后是否清除了签名特定标头。默认为 true，且应只在每个情况下进行其他影响，因为如果未设置，密钥和密码等重要私有信息可能会转义。	true	布尔值
keyStoreParameters (advanced)	根据给定的 KeyStoreParameters 设置可包含用于签名和验证交换的密钥和 Certficates 的 KeyStore。KeyStore 通常与别名一起使用，可以是在 Route 定义中提供的，或者通过消息标头 CamelSignatureKeyStoreAlias 进行动态使用。如果没有提供别名，且密钥存储中只有一个条目，则将使用此单个条目。		KeyStoreParameters
Publickey (advanced)	设置用于在交换中验证签名的 PublicKey。		PublicKey
SecureRandom (advanced)	设置用于初始化 Signature 服务的 SecureRandom。		SecureRandom
password (security)	设置用于访问 KeyStore 中别名的 PrivateKey 的密码。		字符串

24.7. 消息标头

Crypto (JCE)组件支持下面列出的 4 个消息标头。

Name	描述	默认值	类型
[CamelSignaturePrivateKey (producer) 常量 ： SIGNATURE_PRIVATE_KEY	应该用来为消息签名的 PrivateKey。		PrivateKey
[CamelSignaturePublicKeyOrCert (producer) 常数 ： SIGNATURE_PUBLIC_KEY_OR_CERT	应该用于验证签名的证书或公钥。		certificate 或 PublicKey
CamelSignatureKeyStoreAlias (producer) 常量 ： KEYSTORE_ALIAS	用于查询 KeyStore 的别名，用于签名和验证交换时使用的密钥和证书。		字符串
CamelSignatureKeyStorePassword (producer) 常量 ： KEYSTORE_PASSWORD	用于访问 KeyStore 中别名 PrivateKey 的密码。		char[]

24.8. 使用

24.8.1. 原始密钥

签署和验证交换的最基本方法是使用 KeyPair，如下所示：

```
KeyPair keyPair = KeyGenerator.getInstance("RSA").generateKeyPair();

from("direct:sign")
    .setHeader(DigitalSignatureConstants.SIGNATURE_PRIVATE_KEY,
constant(keys.getPrivate()))
    .to("crypto:sign:message")
    .to("direct:verify");

from("direct:verify")
    .setHeader(DigitalSignatureConstants.SIGNATURE_PUBLIC_KEY_OR_CERT,
constant(keys.getPublic()))
    .to("crypto:verify:check");
```

可以通过 Spring XML 扩展 （使用对密钥的引用）来实现相同的操作。

24.8.2. keystores 和 Aliases。

JCE 提供了非常通用的密钥存储概念，用于对私钥和证书进行密钥对，使其加密且受密码保护。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获取到密钥存储中，最常通过外部"keytool"应用程序完成。

以下命令将创建一个包含由 bob 别名的密钥和证书的密钥存储，如下例所示。密钥存储的密码和密钥是 letmein。

```
keytool -genkey -keyalg RSA -keysize 2048 -keystore keystore.jks -storepass letmein -alias
bob -dname "CN=Bob,OU=IT,O=Camel" -noprompt
```

以下路由首先使用与 Camel Registry 绑定的 KeyStore 中的 KeyStore 的别名为交换签名，然后使用同一别名进行验证。

```
from("direct:sign")
    .to("crypto:sign:keystoreSign?alias=bob&keystoreName=myKeystore&password=letmein")
    .log("Signature: ${header.CamelDigitalSignature}")
    .to("crypto:verify:keystoreVerify?
alias=bob&keystoreName=myKeystore&password=letmein")
    .log("Verified: ${body}");
```

以下代码演示了如何加载使用上述 keytool 命令创建的密钥存储，并将它绑定到 registry 中用于上述路由中使用的名称 myKeystore。该示例使用 Camel 3 中引入的 @Configuration 和 @BindToRegistry 注解来实例化 KeyStore，并将它注册到名称 myKeyStore。

```
@Configuration
public class KeystoreConfig {

    @BindToRegistry
    public KeyStore myKeystore() throws Exception {
        KeyStore store = KeyStore.getInstance("JKS");
        try (FileInputStream fis = new FileInputStream("keystore.jks")) {
            store.load(fis, "letmein".toCharArray());
        }
        return store;
    }
}
```

在 Spring a ref 中再次用于查找实际的密钥存储实例。

24.8.3. 更改 JCE Provider 和 Algorithm

更改签名算法或安全供应商是指定名称的简单问题。您还需要使用与您选择的算法兼容的密钥。

24.8.4. 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。路由定义中可以指定不同的标头名称，如下所示

```
from("direct:sign")
   .to("crypto:sign:keystoreSign?
alias=bob&keystoreName=myKeystore&password=letmein&signatureHeaderName=mySignat
ure")
   .log("Signature: ${header.mySignature}")
   .to("crypto:verify:keystoreVerify?
alias=bob&keystoreName=myKeystore&password=letmein&signatureHeaderName=mySignat
ure");
```

===changing the bufferSize

如果需要更新缓冲区的大小。

24.8.5. 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。在每次交换时能够动态指定签名密钥会很有用。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。为方便此目的，签名机制允许通过以下消息标头动态提供密钥。

- DigitalSignatureConstants.SIGNATURE_PRIVATE_KEY, "CamelSignaturePrivateKey"
- DigitalSignatureConstants.SIGNATURE_PUBLIC_KEY_OR_CERT, "CamelSignaturePublicKeyOrCert"

最好是动态提供密钥存储别名。再次在消息标头中提供别名

- DigitalSignatureConstants.KEYSTORE_ALIAS, "CamelSignatureKeyStoreAlias"

标头将设置为如下：

```
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.getMessage());
signed.getIn().setHeader(DigitalSignatureConstants.KEYSTORE_ALIAS, "bob");
template.send("direct:alias-verify", signed);
```

24.9. SPRING BOOT AUTO-CONFIGURATION

组件支持下面列出的 47 选项。

Name	描述	默认值	类型
camel.component.crypto.algorithm	设置用于签名者的算法的 JCE 名称。	SHA256withRSA	字符串
camel.component.crypto.alias	设置用于查询 KeyStore 的别名和 {link java.security.cert.Certificate Certificates}，用于签名和验证交换。此值可以在运行时通过消息标头 org.apache.camel.component.crypto.DigitalSignatureConstants#KEYSTORE_ALIAS 提供。		字符串
camel.component.crypto.autowired-enabled	是否启用自动关闭。这用于自动写入选项（选项必须标记为 autowired），方法是在 registry 中查找是否有单个匹配类型实例，然后在组件上配置。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.crypto.buffer-size	设置用于在 Exchange 有效负载数据中读取的缓冲区的大小。	2048	整数

Name	描述	默认值	类型
camel.component.crypto.certificate	设置应用来根据其有效负载验证交换中的签名的证书。选项是一个 java.security.cert.Certificate 类型。		证书
camel.component.crypto.certificate-name	为 registry 中找到的 PrivateKey 设置引用名称。		字符串
camel.component.crypto.clear-headers	确定签名和验证后是否清除了签名特定标头。默认为 true，且应只在每个情况下进行其他影响，因为如果未设置，密钥和密码等重要私有信息可能会转义。	true	布尔值
camel.component.crypto.configuration	使用 shared DigitalSignatureConfiguration 作为配置。选项是 org.apache.camel.component.crypto.DigitalSignatureConfiguration 类型。		DigitalSignatureConfiguration
camel.component.crypto.enabled	是否启用加密组件的自动配置。这默认是启用的。		布尔值
camel.component.crypto.key-store-parameters	根据给定的 KeyStoreParameters 设置可包含用于签名和验证交换的密钥和 Certficates 的 KeyStore。KeyStore 通常与别名一起使用，可以是在 Route 定义中提供的，或者通过消息标头 CamelSignatureKeyStoreAlias 进行动态使用。如果没有提供别名，且密钥存储中只有一个条目，则将使用此单个条目。选项是 org.apache.camel.support.jsse.KeyStoreParameters 类型。		KeyStoreParameters

Name	描述	默认值	类型
camel.component.crypto.keystore	设置可包含用于签名和验证交换的密钥和 Certficates 的 KeyStore。KeyStore 通常与别名一起使用，可以是在 Route 定义中提供的，或者通过消息标头 CamelSignatureKeyStoreAlias 进行动态使用。如果没有提供别名，且密钥存储中只有一个条目，则将使用此单个条目。选项是一个 java.security.KeyStore 类型。		KeyStore
camel.component.crypto.keystore-name	为 registry 中找到的密钥存储设置引用名称。		字符串
camel.component.crypto.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。通过延迟启动，启动失败可以在路由消息期间通过 Camel 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.crypto.password	设置用于访问 KeyStore 中别名的 PrivateKey 的密码。		字符串
camel.component.crypto.private-key	设置用于为交换签名的 PrivateKey。选项是一个 java.security.PrivateKey 类型。		PrivateKey
camel.component.crypto.private-key-name	为 registry 中找到的 PrivateKey 设置引用名称。		字符串

Name	描述	默认值	类型
camel.component.crypto.provider	设置提供配置的 Signature 算法的安全供应商的 id。		字符串
camel.component.crypto.public-key	设置用于在交换中验证签名的 PublicKey。选项是一个 java.security.PublicKey 类型。		PublicKey
camel.component.crypto.public-key-name	在上下文更改时应解析的引用。		字符串
camel.component.crypto.secure-random	设置用于初始化 Signature 服务的 SecureRandom。选项是一个 java.security.SecureRandom 类型。		SecureRandom
camel.component.crypto.secure-random-name	为 registry 中找到的 SecureRandom 设置引用名称。		字符串
camel.component.crypto.signature-header-name	设置应该用于存储 base64 编码签名的消息标头名称。默认值为 'CamelDigitalSignature' 。		字符串
camel.dataformat.crypto.algorithm	JCE 算法名称表示要使用的加密算法。		字符串
camel.dataformat.crypto.algorithm-parameter-ref	用于初始化 Cipher 的 JCE AlgorithmParameterSpec。将使用指定名称作为 java.security.spec.AlgorithmParameterSpec 类型查找类型。		字符串
camel.dataformat.crypto.buffer-size	签名过程中使用的缓冲区的大小。	4096	整数
camel.dataformat.crypto.crypto-provider	应使用的 JCE 安全提供程序的名称。		字符串

Name	描述	默认值	类型
camel.dataformat.crypto.enabled	是否启用加密数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.crypto.init-vector-ref	指的是包含初始化 Cipher 的初始化向量的字节阵列。		字符串
camel.dataformat.crypto.inline	表示配置的 IV 应该内联到加密的数据流中的标记。默认为 false。	false	布尔值
camel.dataformat.crypto.key-ref	引用要从要使用的注册中查找的 secret 密钥。		字符串
camel.dataformat.crypto.mac-algorithm	JCE 算法名称代表消息身份验证算法。	HmacSHA1	字符串
camel.dataformat.crypto.should-append-h-m-a-c	表示应计算消息身份验证代码并附加到加密数据的标记。	true	布尔值
camel.dataformat.pgp.algorithm	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)。仅与加密相关。		整数
camel.dataformat.pgp.armored	此选项将导致 PGP 对加密文本进行 base64 编码，使它可用于复制/粘贴等。	false	布尔值
camel.dataformat.pgp.compression-algorithm	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)。仅与加密相关。		整数

Name	描述	默认值	类型
camel.dataformat.pgp.enabled	是否启用 pgp 数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.pgp.hash-algorithm	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。		整数
camel.dataformat.pgp.integrity	将完整性检查/签名添加到加密文件中。默认值为 true。	true	布尔值
camel.dataformat.pgp.key-file-name	keyring 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）。		字符串
camel.dataformat.pgp.key-userid	在加密过程中使用的 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。		字符串
camel.dataformat.pgp.password	打开私钥时使用的密码（不用于加密）。		字符串
camel.dataformat.pgp.provider	Java Cryptography 扩展 (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，必须先注册该提供程序，并且必须事先注册 Bouncy Castle 供应商。Sun JCE 供应商无法正常工作。		字符串

Name	描述	默认值	类型
camel.dataformat.pgp.signature-key-file-name	用于签名（确保加密）或签名验证（确保解密）的密钥环的文件名；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）。		字符串
camel.dataformat.pgp.signature-key-ring	用于签名/验证作为字节数组的密钥环。您不能同时设置 signatureKeyFileName 和 signatureKeyRing。		字符串
camel.dataformat.pgp.signature-key-userid	用于签名（确保加密）或签名验证（确保解密）的 PGP 密钥环中的密钥的用户 ID。在签名验证过程中，指定用户 ID 会限制来自公共密钥环的公钥，该密钥环可用于验证。如果没有为签名验证指定用户 ID，则可以使用公钥进行验证。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。		字符串
camel.dataformat.pgp.signature-password	打开用于签名（确保加密）的私钥时使用的密码。		字符串
camel.dataformat.pgp.signature-verification-option	控制在未处理过程中验证签名的行为。有 4 个值：可选：PGP 消息可能或可能不包含签名；如果它包含签名，则执行签名验证。需要：PGP 消息必须至少包含一个签名；如果这不是抛出异常(PGPException)的情况。执行签名验证。忽略：在 PGP 消息中包含签名会被忽略；没有执行签名验证。no_signature_allowed: PGP 消息不得包含签名；否则会抛出异常(PGPException)。		字符串

第 25 章 CSIMPLE

CSimple 语言 编译 了 Simple 语言。

25.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 Csimple 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-core-starter</artifactId>
</dependency>
```

25.2. CSIMPLE 和 SIMPLE 之间的不同

简单语言是一种动态表达式语言，被运行时解析为一组 Camel Expressions 或 Predicates。

Csimple 语言被解析为常规 Java 源代码，并与所有其他源代码一起编译，或者在通过 camel-csimple-joor 模块 bootstrap 期间编译一次。

简单语言通常非常轻便且快，但对于一些通过 OGNL 路径进行动态方法调用的用例，则简单的语言执行运行时内省和反映调用。这对性能有开销，是创建简单原因的原因之一。

csimple 语言需要通过 OGNL 路径是 typesafe 和 method 调用，需要了解解析期间的类型。这意味着，对于 csimple 语言表达式，您需要在脚本中提供类类型，而简单的内省则在运行时是这个类。

换句话说，简单语言使用 *duck typing* （如果看起来像一个 duck ）和 quacks （像 duck），那么它是一个 duck，而 csimple 则正在使用 Java 类型（类型安全）。如果有类型错误，则 simple 会在运行时报告此功能，而 csimple 会出现 Java 编译错误。

25.2.1. 额外的 CSimple 功能

csimple 语言包括一些额外的函数，它们支持使用 Collection, Map 或数组类型的一般用例。以下函数 *bodyAsIndex*、*headerAsIndex* 和 *exchangePropertyAsIndex* 用于这些用例，因为它们被输入。

功能	类型	描述
bodyAsIndex(*type*, *index*)	类型	为了用于从现有的 **Collection**、**Map** 或 数组（按索引查找）收集正文，然后将正文转换为其 classname 确定的给定类型。转换的正文可以是 null。
mandatoryBodyAsIndex(*type*, *index*)	类型	为了用于从现有的 **Collection**、**Map** 或 数组（按索引查找）收集正文，然后将正文转换为其 classname 确定的给定类型。期望正文不是 null。
headerAsIndex (*键,type,index*)	类型	为了用于从现有的 **Collection**、**Map** 或 数组（按索引查找）收集标头，然后将标头值转换为其 classname 确定的给定类型。转换的标头可以是 null。
mandatoryHeaderAsIndex (*key,type,index*)	类型	为了用于从现有的 **Collection**、**Map** 或 数组（按索引查找）收集标头，然后将标头值转换为其 classname 确定的给定类型。期望标头不是 null。
exchangePropertyAsIndex(*key*, *type*, *index*)	类型	为了用于从现有的 **Collection**、**Map** 或 数组（由索引查找）收集交换属性，然后将 Exchange 属性转换为其 classname 确定的给定类型。转换的交换属性可以是 null。
mandatoryExchangePropertyAsIndex(*key*, *type*, *index*)	类型	为了用于从现有的 **Collection**、**Map** 或 数组（由索引查找）收集交换属性，然后将 Exchange 属性转换为其 classname 确定的给定类型。期望 exchange 属性不为空。

例如，给出以下简单表达式：

```
Hello $\{body[0].name}
```

此脚本没有类型信息，简单语言将在运行时解决，方法是内省消息正文，如果基于集合，则查找第一个元素，然后通过反映调用名为 getName 的方法。

在 csimple （编译）中，我们希望预编译它，因此最终用户必须使用 *bodyAsIndex* 功能提供类型信息：

```
Hello $\{bodyAsIndex(com.foo.MyUser, 0).name}
```

25.3. 编译

Csimple 语言被解析为常规 Java 源代码，并与所有其他源代码一起编译，或者在通过 camel-csimple-joor 模块 bootstrap 期间编译一次。

编译简单的方法有两种

- 在构建时使用 camel-csimple-maven-plugin 生成源代码。
- 使用 camel-csimple-joor，它在 Camel bootstrap 过程中执行运行时内存编译。

25.3.1. 使用 camel-csimple-maven-plugin

camel-csimple-maven-plugin Maven 插件用于发现来自源代码的所有 csimple 脚本，然后在 src/generated/java 文件夹中自动生成源代码，然后编译到所有其他源。

maven 插件将对 .java 和 .xml 文件(Java 和 XML DSL)进行源代码扫描。扫描程序限制检测某些代码模式，如果它们以异常/方式使用，则可能会错过发现一些简单脚本。

使用 camel-csimple-joor 的运行时编译没有这个限制。

好处是所有 csimple 脚本都使用常规 Java 编译器编译，因此所有内容都作为应用程序 JAR 文件中的 .class 文件包含在框中，在运行时不需要额外的依赖项。

要使用 camel-csimple-maven-plugin，您需要将其添加到 pom.xml 文件中，如下所示：

```
<plugins>
  <!-- generate source code for csimple languages -->
  <plugin>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-csimple-maven-plugin</artifactId>
    <version>${camel.version}</version>
    <executions>
      <execution>
        <id>generate</id>
        <goals>
          <goal>generate</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
  <!-- include source code generated to maven sources paths -->
  <plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>build-helper-maven-plugin</artifactId>
```

```
        <version>3.1.0</version>
        <executions>
          <execution>
            <phase>generate-sources</phase>
            <goals>
              <goal>add-source</goal>
              <goal>add-resource</goal>
            </goals>
            <configuration>
              <sources>
                <source>src/generated/java</source>
              </sources>
              <resources>
                <resource>
                  <directory>src/generated/resources</directory>
                </resource>
              </resources>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
```

然后，您还必须添加 build-helper-maven-plugin Maven 插件，将 src/generated 包含在 Java 编译器的源文件夹列表中，以确保生成的源代码已编译并包含在应用 JAR 文件中。

请参阅 Camel 示例（使用 maven 插件的 camel-example-csimple 示例）。

25.3.2. 使用 camel-csimple-joor

jOOR 库与 Java 编译器集成，并执行 Java 代码运行时编译。

使用 camel-simple-joor 时支持的运行时适用于 Java 独立、Spring Boot、Camel Quarkus 和其他微服务运行时。在 OSGi、Camel Karaf 或任何类型的 Java Application Server 运行时中不支持它。

joor 不支持使用 *fat jar* 打包(https://github.com/jOOQ/jOOR/issues/69)的运行时编译，它可用于 exploded 类路径。

要使用 camel-simple-joor，只需将其作为依赖项添加到 classpath 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-csimple-joor</artifactId>
```

```
  <version>{CamelSBProjectVersion}</version>
</dependency>
```

不需要将 Maven 插件添加到 pom.xml 文件中。

请参阅 camel-example-csimple-joor 示例（Camel Examples），它使用 jOOR 编译器。

25.4. CSIMPLE LANGUAGE 选项

CSimple 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		布尔值	是否修剪值以移除前导和结尾的空格和换行符。

25.5. 限制

目前，Csimple 语言 不支持 ：

- 嵌套功能（函数中的函数）
- *null 安全运算符* (?)。

例如，以下脚本无法编译：

```
Hello ${bean:greeter(${body}, ${header.counter})}
```

```
${bodyAs(MyUser)?.address?.zip} > 10000
```

25.6. 自动导入

csimple 语言将自动导入：

```
import java.util.*;
import java.util.concurrent.*;
import java.util.stream.*;
import org.apache.camel.*;
import org.apache.camel.util.*;
```

25.7. 配置文件

您可以在 camel-csimple.properties 文件中配置 csimple 语言，该文件是从 root 类路径加载的。

例如，您可以通过添加以下内容在 camel-csimple.properties 文件中添加其他导入：

```
import com.foo.MyUser;
import com.bar.*;
import static com.foo.MyHelper.*;
```

您还可以添加别名(key=value)，其中别名将用作代码中的简写替换。

```
echo()=${bodyAs(String)} ${bodyAs(String)}
```

这允许在 csimple 语言脚本中使用 *echo （）*，例如：

```
from("direct:hello")
    .transform(csimple("Hello echo()"))
    .log("You said ${body}");
```

echo （） 别名将被替换为其值，从而形成如下脚本：

```
    .transform(csimple("Hello ${bodyAs(String)} ${bodyAs(String)}"))
```

25.8. 另请参阅

请参阅 简单 语言，因为 csimple 具有与简单语言相同的功能集合。

25.9. SPRING BOOT AUTO-CONFIGURATION

组件支持 147 选项，如下所列。

Name	描述	默认值	类型
camel.cloud.consul.service-discovery.acl-token	设置用于 Consul 的 ACL 令牌。		字符串
camel.cloud.consul.service-discovery.block-seconds	等待监视事件的秒数，默认为 10 秒。	10	整数
camel.cloud.consul.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.consul.service-discovery.connect-timeout-millis	OkHttpClient 的连接超时。		Long
camel.cloud.consul.service-discovery.datacenter	数据中心。		字符串
camel.cloud.consul.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.consul.service-discovery.password	设置用于基本身份验证的密码。		字符串
camel.cloud.consul.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.consul.service-discovery.read-timeout-millis	OkHttpClient 的读取超时。		Long
camel.cloud.consul.service-discovery.url	Consul 代理 URL。		字符串

Name	描述	默认值	类型
camel.cloud.consul.service-discovery.user-name	设置用于基本身份验证的用户名。		字符串
camel.cloud.consul.service-discovery.write-timeout-millis	OkHttpClient 的写入超时。		Long
camel.cloud.dns.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.dns.service-discovery.domain	域名；		字符串
camel.cloud.dns.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.dns.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.dns.service-discovery.proto	所需服务的传输协议。	_tcp	字符串
camel.cloud.etcd.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.etcd.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.etcd.service-discovery.password	用于基本身份验证的密码。		字符串

Name	描述	默认值	类型
camel.cloud.etcd.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.etcd.service-discovery.service-path	查找服务发现的路径。	/services/	字符串
camel.cloud.etcd.service-discovery.timeout	要设置操作可以采取的最长时间，请执行以下操作：		Long
camel.cloud.etcd.service-discovery.type	要设置发现类型，有效值为 on-demand 和 watch。	按需	字符串
camel.cloud.etcd.service-discovery.uris	客户端可以连接到的 URI。		字符串
camel.cloud.etcd.service-discovery.user-name	用于基本身份验证的用户名。		字符串
camel.cloud.kubernetes.service-discovery.api-version	使用客户端查找时设置 API 版本。		字符串
camel.cloud.kubernetes.service-discovery.ca-cert-data	使用客户端查找时设置证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.ca-cert-file	在使用客户端查找时，设置从文件加载的证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-data	使用客户端查找时设置客户端证书数据。		字符串

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.client-cert-file	在使用客户端查找时，设置从文件加载的客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-algo	设置客户端密钥存储算法，如使用客户端查找时 RSA。		字符串
camel.cloud.kubernetes.service-discovery.client-key-data	使用客户端查找时设置客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-file	在使用客户端查找时，设置从文件加载的客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-passphrase	使用客户端查找时设置客户端密钥存储密码短语。		字符串
camel.cloud.kubernetes.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.kubernetes.service-discovery.dns-domain	设置用于 DNS 查找的 DNS 域。		字符串
camel.cloud.kubernetes.service-discovery.enabled	启用组件。	true	布尔值

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.lookup	如何执行服务查找。可能的值有：client、dns、environment。在使用客户端时，客户端会查询 kubernetes master 来获取提供该服务的活跃 pod 列表，然后随机（或循环）选择一个 pod。当使用 dns 时，服务名称被解析为 name.namespace.svc.dnsDomain。当使用 dnssrv 时，服务名称使用 SRV 查询解析 ....svc... when using environment，环境变量用于查找服务。默认情况下使用环境。	环境	字符串
camel.cloud.kubernetes.service-discovery.master-url	在使用客户端查找时，将 URL 设置为 master。		字符串
camel.cloud.kubernetes.service-discovery.namespace	设置要使用的命名空间。默认情况下，将使用来自 ENV 变量 KUBERNETES_MASTER 的命名空间。		字符串
camel.cloud.kubernetes.service-discovery.oauth-token	在使用客户端查找时，为身份验证设置 OAUTH 令牌（而不是用户名/密码）。		字符串
camel.cloud.kubernetes.service-discovery.password	在使用客户端查找时设置用于身份验证的密码。		字符串
camel.cloud.kubernetes.service-discovery.port-name	设置用于 DNS/DNSSRV 查找的端口名称。		字符串
camel.cloud.kubernetes.service-discovery.port-protocol	设置用于 DNS/DNSSRV 查找的端口协议。		字符串
camel.cloud.kubernetes.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.trust-certs	设置在使用客户端查找时是否打开信任证书检查。	false	布尔值
camel.cloud.kubernetes.service-discovery.username	在使用客户端查找时设置用于身份验证的用户名。		字符串
camel.cloud.ribbon.load-balancer.client-name	设置 Ribbon 客户端名称。		字符串
camel.cloud.ribbon.load-balancer.configurations	定义其他配置定义。		Map
camel.cloud.ribbon.load-balancer.enabled	启用组件。	true	布尔值
camel.cloud.ribbon.load-balancer.namespace	命名空间。		字符串
camel.cloud.ribbon.load-balancer.password	密码。		字符串
camel.cloud.ribbon.load-balancer.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.ribbon.load-balancer.username	用户名。		字符串
camel.hystrix.allow-maximum-size-to-diverge-from-core-size	允许配置使 maximumSize 生效。然后该值可以等于或大于 coreSize。	false	布尔值

Name	描述	默认值	类型
camel.hystrix.circuit-breaker-enabled	是否使用 HystrixCircuitBreaker。如果为 false，则不会使用 断路器逻辑，并且所有允许的请求。这与 circuitBreakerForceClosed （）的影响类似，除非继续跟踪指标，知道它是否应该是 open/closed，此属性即使实例化一个断路器。	true	布尔值
camel.hystrix.circuit-breaker-error-threshold-percentage	错误百分比阈值（如 50 ）指向断路器将打开和拒绝请求。它将在 circuitBreakerSleepWindowInMilliseconds 中定义的持续时间保持出差；与 HystrixCommandMetrics.getHealthCounts （）进行比较的错误百分比。	50	整数
camel.hystrix.circuit-breaker-force-closed	如果为 true，HystrixCircuitBreaker#allowRequest （）将始终返回 true 以允许请求，无论 HystrixCommandMetrics.getHealthCounts （）的错误百分比如何。如果设为 true，则 circuitBreakerForceOpen （）属性具有优先权。	false	布尔值
camel.hystrix.circuit-breaker-force-open	如果为 true，HystrixCircuitBreaker.allowRequest （）将始终返回 false，从而导致电路变为开路（接受），并拒绝所有请求。此属性优先于 circuitBreakerForceClosed （）;。	false	布尔值
camel.hystrix.circuit-breaker-request-volume-threshold	metricsRollingStatisticalWindowInMilliseconds （）中的最少请求数必须存在于 HystrixCircuitBreaker 之前。如果此数字低于这个数字，无论错误百分比如何，电路都不会被出差。	20	整数
camel.hystrix.circuit-breaker-sleep-window-in-milliseconds	HystrixCircuitBreaker trips 之后的时间（以毫秒为单位），它应该在尝试请求前等待。	5000	整数
camel.hystrix.configurations	定义其他配置定义。		Map
camel.hystrix.core-pool-size	传递给 java.util.concurrent.ThreadPoolExecutor#setCorePoolSize (int)的核心 thread-pool 大小。	10	整数
camel.hystrix.enabled	启用组件。	true	布尔值
camel.hystrix.execution-isolation-semaphore-max-concurrent-requests	允许 HystrixCommand.run （）的并发请求数。超过并发限制的请求将被拒绝。仅在执行IsolationStrategy == SEMAPHORE 时使用。	20	整数

Name	描述	默认值	类型
camel.hystrix.execution-isolation-strategy	将通过什么隔离策略 HystrixCommand.run （）执行。如果 THREAD，它将在单独的线程上执行，并且受 thread-pool 中的线程数量限制的并发请求。如果 SEMAPHORE，它将在调用线程上执行，并且受 semaphore 数限制的并发请求。	线程	字符串
camel.hystrix.execution-isolation-thread-interrupt-on-timeout	当线程超时时，执行线程是否应该尝试中断（使用 future#cancel）。仅在执行IsolationStrategy （）== THREAD 时才适用。	true	布尔值
camel.hystrix.execution-timeout-enabled	此命令是否启用了超时机制。	true	布尔值
camel.hystrix.execution-timeout-in-milliseconds	以毫秒为单位，将命令超时和停止执行的时间（以毫秒为单位）。如果 executionIsolationThreadInterruptOnTimeout == true 且命令是线程隔离，则执行线程将中断。如果命令是 semaphore-isolated 和 HystrixObservableCommand，则该命令将被取消订阅。	1000	整数
camel.hystrix.fallback-enabled	出现故障时，是否应尝试 HystrixCommand.getFallback （）。	true	布尔值
camel.hystrix.fallback-isolation-semaphore-max-concurrent-requests	允许 HystrixCommand.getFallback （）的并发请求数。超过并发限制的请求将快速失败，且不会尝试检索回退。	10	整数
camel.hystrix.group-key	设置要使用的 group 键。默认值为 CamelHystrix。	Camel Hystrix	字符串
camel.hystrix.keep-alive-time	更长的时间（以分钟为单位）传递给 ThreadPoolExecutor#setKeepAliveTime (long,TimeUnit)。	1	整数
camel.hystrix.max-queue-size	在 HystrixConcurrencyStrategy.getBlockingQueue (int)中传递给 BlockingQueue 的最大队列大小应该只影响 threadpool 的实例化 - 它不会立即更改队列大小。为此，请使用 queueSizeRejectionThreshold （）。	-1	整数

Name	描述	默认值	类型
camel.hystrix.maximum-size	传递给 ThreadPoolExecutor#setMaximumPoolSize(int)的最大 thread-pool 大小。这是可在不开始拒绝 HystrixCommands 的情况下支持的最大并发数量。请注意，只有在您也设置了 allowMaximumSizeToDivergeFromCoreSize 时，此设置才会生效。	10	整数
camel.hystrix.metrics-health-snapshot-interval-in-milliseconds	在允许计算成功和错误百分比时等待的时间（以毫秒为单位），并影响 HystrixCircuitBreaker.isOpen （）状态。在高容量电路上，错误百分比的连续计算可能会成为 CPU 密集型，从而控制其计算的频率。	500	整数
camel.hystrix.metrics-rolling-percentile-bucket-size	滚动百分比的每个存储桶中存储的最大值数。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	10	整数
camel.hystrix.metrics-rolling-percentile-enabled	是否应该使用 HystrixRollingPercentile 内部 HystrixCommandMetrics 来捕获百分比的指标。	true	布尔值
camel.hystrix.metrics-rolling-percentile-window-buckets	滚动窗口的存储桶数量被分成。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	6	整数
camel.hystrix.metrics-rolling-percentile-window-in-milliseconds	以毫秒为单位的滚动窗口的持续时间。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	10000	整数
camel.hystrix.metrics-rolling-statistical-window-buckets	滚动统计窗口划分为的 bucket 数量。这在 HystrixCommandMetrics 中被传递给 HystrixRollingNumber。	10	整数
camel.hystrix.metrics-rolling-statistical-window-in-milliseconds	此属性设置统计滚动窗口的持续时间，以毫秒为单位。这是为线程池保留指标的时间。窗口被分成 bucket，按这些增量回滚。	10000	整数

Name	描述	默认值	类型
camel.hystrix.queue-size-rejection-threshold	队列大小拒绝阈值是 artificial max size，即使尚未达到 maxQueueSize，也会发生拒绝。这是因为 BlockingQueue 的 maxQueueSize 无法动态更改，我们希望动态更改影响拒绝的队列大小。在排队线程以进行执行时，HystrixCommand 会使用它。	5	整数
camel.hystrix.request-log-enabled	HystrixCommand 执行和事件是否应记录到 HystrixRequestLog。	true	布尔值
camel.hystrix.thread-pool-key	设置要使用的线程池密钥。默认情况下，将使用与 groupKey 配置相同的值。	Camel Hystrix	字符串
camel.hystrix.thread-pool-rolling-number-statistical-window-buckets	滚动统计窗口划分为的 bucket 数量。这会传递到每个 HystrixThreadPoolMetrics 实例内的 HystrixRollingNumber。	10	整数
camel.hystrix.thread-pool-rolling-number-statistical-window-in-milliseconds	统计滚动窗口的持续时间（以毫秒为单位）。这会传递到每个 HystrixThreadPoolMetrics 实例内的 HystrixRollingNumber。	10000	整数
camel.language.constant.enabled	是否启用恒定语言的自动配置。这默认是启用的。		布尔值
camel.language.constant.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.csimple.enabled	是否启用 csimple 语言的自动配置。这默认是启用的。		布尔值
camel.language.csimple.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.exchangeproperty.enabled	是否启用 exchangeProperty 语言的自动配置。这默认是启用的。		布尔值
camel.language.exchangeproperty.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.file.enabled	是否启用文件语言的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.language.file.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.header.enabled	是否启用标头语言的自动配置。这默认是启用的。		布尔值
camel.language.header.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.ref.enabled	是否启用 ref 语言的自动配置。这默认是启用的。		布尔值
camel.language.ref.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.simple.enabled	是否启用简单语言的自动配置。这默认是启用的。		布尔值
camel.language.simple.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.tokenize.enabled	是否启用令牌化语言的自动配置。这默认是启用的。		布尔值
camel.language.tokenize.group-delimiter	设置在分组时要使用的分隔符。如果没有设置，则令牌将用作分隔符。		字符串
camel.language.tokenize.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.resilience4j.automatic-transition-from-open-to-half-open-enabled	在通过 waitDurationInOpenState 后，启用从 OPEN 自动过渡到 HALF_OPEN 状态。	false	布尔值
camel.resilience4j.circuit-breaker-ref	代表现有的 io.github.resilience4j.circuitbreaker.CircuitBreaker 实例从 registry 中查找和使用。使用此选项时，不使用任何其他断路器选项。		字符串
camel.resilience4j.config-ref	指的是现有的 io.github.resilience4j.circuitbreaker.CircuitBreakerConfig 实例，以便从 registry 中查找和使用。		字符串

Name	描述	默认值	类型
camel.resilience4j.configurations	定义其他配置定义。		Map
camel.resilience4j.enabled	启用组件。	true	布尔值
camel.resilience4j.failure-rate-threshold	以百分比为单位配置故障率阈值。如果失败率相等或大于阈值，则 CircuitBreaker 过渡到 open，并启动短路调用。阈值必须大于 0，而不是大于 100。默认值为 50 百分比。		æµ®ç‚¹å€¼
camel.resilience4j.minimum-number-of-calls	在 CircuitBreaker 可以计算错误率之前，配置所需的最少调用数（每个滑动期限）。例如，如果 minimumNumberOfCalls 为 10，则必须至少记录 10 个调用，然后才能计算失败率。如果只记录了 9 个调用，则 CircuitBreaker 不会过渡到 open，即使所有 9 调用都失败。默认 minimumNumberOfCalls 为 100。	100	整数
camel.resilience4j.permitted-number-of-calls-in-half-open-state	配置 CircuitBreaker 为一半打开时允许的调用数量。大小必须大于 0。默认大小为 10。	10	整数
camel.resilience4j.sliding-window-size	配置滑动窗口的大小，该窗口用于在 CircuitBreaker 关闭时记录调用的结果。slidingWindowSize 配置滑动窗口的大小。滑动窗口可以是基于计数或基于时间的窗口。如果 slidingWindowType 是 COUNT_BASED，则最后的 slidingWindowSize 调用会被记录并聚合。如果 slidingWindowType 是 TIME_BASED，则最后 slidingWindowSize 秒的调用会被记录并聚合。slidingWindowSize 必须大于 0。minimumNumberOfCalls 必须大于 0。如果 slidingWindowType 是 COUNT_BASED，则 minimumNumberOfCalls 不能大于 slidingWindowSize。如果 slidingWindowType 是 TIME_BASED，您可以选择任何您需要的。默认 slidingWindowSize 为 100。	100	整数
camel.resilience4j.sliding-window-type	配置滑动窗口的类型，用于记录 CircuitBreaker 关闭时调用的结果。滑动窗口可以是基于计数或基于时间的窗口。如果 slidingWindowType 是 COUNT_BASED，则最后的 slidingWindowSize 调用会被记录并聚合。如果 slidingWindowType 是 TIME_BASED，则最后 slidingWindowSize 秒的调用会被记录并聚合。默认 slidingWindowType 是 COUNT_BASED。	COUNT_BASED	字符串

Name	描述	默认值	类型
camel.resilience4j.slow-call-duration-threshold	配置上面的持续时间阈值（秒），调用被视为缓慢，并增加较慢的调用百分比。默认值为 60 秒。	60	整数
camel.resilience4j.slow-call-rate-threshold	以百分比为单位配置阈值。当调用持续时间大于 slowCallDurationThreshold Duration 时，CircuitBreaker 会将调用视为较慢。当较慢的调用百分比相等或大于阈值时，CircuitBreaker 过渡到 open，并启动短路调用。阈值必须大于 0，而不是大于 100。默认值为 100 百分比，这意味着所有记录的调用都必须比 slowCallDurationThreshold 慢。		æµ®ç‚¹â€¼
camel.resilience4j.wait-duration-in-open-state	配置等待持续时间（以秒为单位），指定 CircuitBreaker 应该保持打开的时间，然后再切换到半次。默认值为 60 秒。	60	整数
camel.resilience4j.writable-stack-trace-enabled	启用可写入堆栈跟踪。当设置为 false 时，Exception.getStackTrace 返回一个零长度数组。当断路器处于开路状态时，这可用于减少日志垃圾邮件，因为存在例外的原因（断路器是短路调用）。	true	布尔值
camel.rest.api-component	用作 REST API 的 Camel 组件名称（如 swagger）如果没有明确配置 API 组件，则 Camel 会查找负责服务并生成 REST API 文档的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestApiProcessorFactory。如果找到其中任何一个，则使用它。		字符串
camel.rest.api-context-path	设置领导的 API 上下文路径将使用的 REST API 服务。这在使用使用 context-path 部署 Web 应用程序的组件（如 camel-servlet ）时使用它。		字符串
camel.rest.api-context-route-id	设置用于服务 REST API 的路由的路由 ID。默认情况下，路由将使用自动分配的路由 ID。		字符串
camel.rest.api-host	要将特定主机名用于 API 文档（如 swagger），这可用于用这个配置的主机名覆盖生成的主机。		字符串
camel.rest.api-property	允许为 api 文档配置任意数量的附加属性(swagger)。例如，将属性 api.title 设置为我的冷却。		Map
camel.rest.api-vendor-extension	是否在 Rest API 中启用供应商扩展。如果启用，Camel 将包含额外信息作为厂商扩展名（例如，以 x- 开头的键），如路由 ID、类名称等。在导入 API 文档时，并非所有第三方 API 网关和工具都支持 vendor-extensions。	false	布尔值

Name	描述	默认值	类型
camel.rest.binding-mode	设置要使用的绑定模式。默认值为 off。		RestBindingMode
camel.rest.client-request-validation	是否启用客户端请求验证，以检查客户端的 Content-Type 和 Accept 标头是否受到其 consume/produces 设置的 Rest-DSL 配置的支持。这可以打开，以启用此检查。如果验证错误，则返回 HTTP Status code 415 或 406。默认值为 false。	false	布尔值
camel.rest.component	用于 REST 传输(consumer)的 Camel Rest 组件，如 netty-http, jetty, servlet, undertow。如果没有明确配置组件，则 Camel 会查找是否有与 Rest DSL 集成的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestConsumerFactory。如果找到其中任何一个，则使用它。		字符串
camel.rest.component-property	允许为正在使用的其他组件配置任意数量的附加属性。		Map
camel.rest.consumer-property	允许为使用中的其他使用者配置任意数量的附加属性。		Map
camel.rest.context-path	设置 REST 服务将使用的前导上下文路径。这在使用使用 context-path 部署 Web 应用程序的组件（如 camel-servlet ）时使用它。或者对于包含 HTTP 服务器的 camel-jetty 或 camel-netty-http 等组件。		字符串
camel.rest.cors-headers	允许配置自定义 CORS 标头。		Map
camel.rest.data-format-property	允许为使用的数据格式配置多个额外属性。例如，将属性 prettyPrint 设置为 true，以便以用户友善模式输出 json。属性可以加上前缀来表示选项仅适用于 JSON 或 XML，以及 IN 或 OUT。前缀为： json.in. json.out. xml.in. xml.out。例如，值为 xml.out.mustBeJAXBElement 的键仅用于传出的 XML 数据格式。没有前缀的密钥是所有情况的通用密钥。		Map
camel.rest.enable-cors	是否在 HTTP 响应中启用 CORS 标头。默认值为 false。	false	布尔值
camel.rest.endpoint-property	允许为使用中的其他端点配置多个额外的属性。		Map
camel.rest.host	用于公开 REST 服务的主机名。		字符串

Name	描述	默认值	类型
camel.rest.host-name-resolver	如果没有明确配置的主机名，这个 resolver 会用于计算 REST 服务将要使用的主机名。		RestHostNameResolver
camel.rest.json-data-format	要使用的特定 json 数据格式的名称。默认将使用 json-jackson。重要：此选项仅用于设置数据格式的自定义名称，而不是引用现有数据格式实例。		字符串
camel.rest.port	用于公开 REST 服务的主机名。请注意，如果您使用 servlet 组件，则此处配置的端口号不适用，因为使用中的端口号是 servlet 组件使用的实际端口号。例如，如果使用 Apache Tomcat，它的 tomcat http 端口，如果使用 Apache Karaf，它的在 Karaf 中的 HTTP 服务，它默认使用端口 8181。虽然在这些情况下，这里设置端口号，但允许工具和 JMX 知道端口号，因此建议将端口号设置为 servlet 引擎使用的数字。		字符串
camel.rest.producer-api-doc	设置 api 文档的位置，REST 生成者将根据这个文档来验证 REST uri 和查询参数是否有效。这需要将 camel-swagger-java 添加到 classpath 中，任何缺失的配置都会导致 Camel 在启动时失败并报告错误。默认情况下从 classpath 加载的 api 文档的位置，但您可以使用 file: 或 http: 引用从文件或 http url 加载的资源。		字符串
camel.rest.producer-component	设置要用作 REST 生成者的 Camel 组件的名称。		字符串
camel.rest.scheme	用于公开 REST 服务的方案。通常支持 http 或 https。默认值为 http。		字符串
camel.rest.skip-binding-on-error-code	如果存在自定义 HTTP 错误代码标头，是否跳过输出绑定。这允许构建没有绑定到 json / xml 等自定义错误消息，否则成功信息会这样做。	false	布尔值
camel.rest.use-x-forward-headers	是否将 X-Forward 标头用于主机和相关设置。默认值为 true。	true	布尔值
camel.rest.xml-data-format	要使用的特定 XML 数据格式的名称。默认情况下将使用 jaxb。重要：此选项仅用于设置数据格式的自定义名称，而不是引用现有数据格式实例。		字符串
camel.rest.api-context-id-pattern	**弃用** 设置 CamelContext id 特征，以只允许 CamelContext 中名称与特征匹配的其他服务的 Rest API。特征 name 指的是 CamelContext 名称，仅匹配当前的 CamelContext。对于任何其他值，特征使用来自 PatternHelper#matchPattern (String,String)的规则。		字符串

Name	描述	默认值	类型
camel.rest.api-context-listing	**弃用** 设置是否启用了 JVM 中带有 REST 服务的所有可用 CamelContext 的列表。如果启用，它将允许发现这些上下文，如果为 false，则只使用当前的 CamelContext。	false	布尔值

第 26 章 CXF

支持生成者和消费者

CXF 组件提供与 Apache CXF 集成，以连接到 CXF 中托管的 JAX-WS 服务。

提示

当在流模式中使用 CXF 时（请参阅 DataFormat 选项），然后阅读有关流缓存的信息。

26.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 cxf 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-cxf-soap-starter</artifactId>
</dependency>
```

26.2. URI 格式

此端点有两个 URI 格式： cxfEndpoint 和 someAddress。

```
cxf:bean:cxfEndpoint[?options]
```

其中 cxfEndpoint 代表一个 bean ID，它引用 Spring bean registry 中的 bean。使用此 URI 格式时，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。使用此 URI 格式时，大多数端点详情都通过选项来指定。

对于以上任一样式，您可以按如下方式在 URI 中附加选项：

```
cxf:bean:cxfEndpoint?wsdlURL=wsdl/hello_world.wsdl&dataFormat=PAYLOAD
```

26.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

26.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

26.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

26.4. 组件选项

CXF 组件支持 6 个选项，如下所列。

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
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
allowStreaming (advanced)	这个选项控制在 PAYLOAD 模式运行时 CXF 组件是否会将传入的消息解析为 DOM Elements，或将有效负载保留为 javax.xml.transform.Source 对象，在某些情况下允许流。		布尔值
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头到 Camel 消息。		HeaderFilterStrategy
useGlobalSslContextParameters (security)	启用使用全局 SSL 上下文参数。	false	布尔值

26.5. 端点选项

CXF 端点使用 URI 语法进行配置：

```
cxf:beanId:address
```

使用以下路径和查询参数：

26.5.1. 路径参数(2 参数)

Name	描述	默认值	类型
beanId (common)	查找现有的已配置的 CxfEndpoint。必须使用 bean: 作为前缀。		字符串
地址 (service)	服务发布地址。		字符串

26.5.2. 查询参数(35 参数)

Name	描述	默认值	类型
dataformat (common)	CXF 端点支持的数据类型消息。 Enum 值： • PAYLOAD • RAW • MESSAGE • CXF_MESSAGE • POJO	POJO	DataFormat
wrapStyle (common)	WSDL 样式，用于描述在 SOAP 正文中如何表示参数。如果值为 false，则 CXF 将选择 document-literal unwrapped 风格，如果值为 true，则 CXF 将选择文档封装样式。		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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
cookieHandler (producer)	配置 Cookie 处理程序来维护 HTTP 会话。		CookieHandler
defaultOperationName (producer)	这个选项将设置默认 operationName，供调用远程服务的 CxfProducer 使用。		字符串
defaultOperationNamespace (producer)	这个选项将设置默认 operationNamespace，供调用远程服务的 CxfProducer 使用。		字符串
hostnameVerifier (producer)	要使用的主机名验证器。使用 # 表示法引用 registry 中的 HostnameVerifier。		HostnameVerifier
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
sslContextParameters (producer)	Camel SSL 设置参考。使用 # 表示法引用 SSL 上下文。		SSLContextParameters
wrap (producer)	CXF 端点制作者将调用哪些操作。	false	布尔值
同步 (producer (advanced))	设置是否应严格使用同步处理。	false	布尔值
allowStreaming (advanced)	这个选项控制在 PAYLOAD 模式运行时 CXF 组件是否会将传入的消息解析为 DOM Elements，或将有效负载保留为 javax.xml.transform.Source 对象，在某些情况下允许流。		布尔值
Bus (advanced)	使用自定义配置的 CXF 总线。		bus

Name	描述	默认值	类型
continuationTimeout (advanced)	这个选项用于设置 CXF 持续超时，在 CXF 服务器使用 Jetty 或 Servlet 传输时，默认可在 CxfConsumer 中使用。	30000	long
cxfBinding (advanced)	使用自定义 CxfBinding 控制 Camel 消息和 CXF 消息之间的绑定。		CxfBinding
cxfConfigurer (advanced)	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。用户可以通过实施 CxfEndpointConfigurer 的 configure{ServerClient} 方法来配置 CXF 服务器和客户端。		CxfConfigurer
defaultBus (advanced)	当 CXF 端点本身创建总线时，将设置默认总线。	false	布尔值
headerFilterStrategy (advanced)	使用自定义 HeaderFilterStrategy 过滤标头到 Camel 消息。		HeaderFilterStrategy
mergeProtocolHeaders (advanced)	是否合并协议标头。如果启用，则在 Camel 和 CXF 之间传播标头变得更为一致且类似。如需了解更多详细信息，请参阅 CAMEL-6393。	false	布尔值
mtomEnabled (advanced)	启用 MTOM (attachments)。这需要使用 POJO 或 PAYLOAD 数据格式模式。	false	布尔值
properties (advanced)	使用 Map 中的键/值对设置额外的 CXF 选项。例如，若要在 SOAP 故障中打开 stacktrace，properties.faultStackTraceEnabled=true。		Map
skipPayloadMessagePartCheck (advanced)	设置是否应禁用 SOAP 消息验证。	false	布尔值
loggingFeatureEnabled (logging)	这个选项启用 CXF Logging 功能，将入站和出站 SOAP 消息写入日志。	false	布尔值
loggingSizeLimit (logging)	要限制在启用了日志记录功能时日志记录器将输出的字节数，并且 -1 代表没有限制。	49152	int
skipFaultLogging (logging)	这个选项控制 PhaseInterceptorChain 是否跳过记录它捕获的 Fault。	false	布尔值
password (security)	这个选项用于为 CXF 客户端设置密码的基本身份验证信息。		字符串

Name	描述	默认值	类型
用户名 （安全性）	这个选项用于为 CXF 客户端设置 username 的基本身份验证信息。		字符串
bindingId (service)	要使用的服务模型的 bindingId。		字符串
portName (service)	此服务所实现的端点名称，它映射到 wsdl:portname。格式为 ns:PORT_NAME，其中 ns 是在这个范围内有效的命名空间前缀。		字符串
publishedEndpointUrl (service)	此选项可以覆盖从 WSDL 发布的 endpointUrl，该Url可以通过服务地址 url 和 wsd 进行访问。		字符串
serviceClass (service)	SEI (Service Endpoint Interface)类的类名称，它们可能具有 JSR181 注释。		类
serviceName (service)	此服务正在实现的服务名称，它映射到 wsdl:servicename。		字符串
wsdlURL (service)	WSDL 的位置。可以位于类路径、文件系统上，也可以远程托管。		字符串

serviceName 和 portName 是 QNames，因此如果您提供它们，请确保它们带有其 {namespace} 前缀，如上例中所示。

26.5.3. dataformats 的描述

在 Apache Camel 中，Camel CXF 组件是将路由与 Web 服务集成的关键。您可以使用 Camel CXF 组件创建 CXF 端点，该端点可通过以下方法之一使用：

- **consumer - （路由开始时）代表 Web 服务实例，它与路由集成。注入路由的有效负载类型取决于端点的 dataFormat 选项的值。**
- **制作者 - （路由中的其他点）代表 WS 客户端代理，它将当前交换对象转换为远程 Web 服务的操作调用。当前交换的格式必须与端点的 dataFormat 设置匹配。**

DataFormat	描述
POJO	POJO （老的 Java 对象）是目标服务器上调用的方法的 Java 参数。支持 Protocol 和 Logical JAX-WS 处理程序。

DataFormat	描述
PAYLOAD	**PAYLOAD** 是应用 CXF 端点的消息配置后的消息有效负载(**soap:body**的内容)。仅支持 Protocol JAX-WS 处理程序。不支持逻辑 JAX-WS 处理程序。
RAW	**RAW** 模式提供从传输层接收的原始消息流。如果您正在使用这种类型的 DataFormat，则无法处理或更改流，则一些 CXF 拦截器会被删除，因此您可以在 camel-cxf 消费者后看到任何 soap 标头。不支持 JAX-WS 处理程序。
CXF_MESSAGE	**CXF_MESSAGE** 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.common.message.CxfConstants.DATA_FORMAT_PROPERTY 中定义。

26.5.4. 如何在 RAW 模式中启用 CXF LoggingOutInterceptor

CXF 的 LoggingOutInterceptor 输出在有线路到日志记录系统(Java Util Logging)的出站消息。由于 LoggingOutInterceptor 位于 PRE_STREAM 阶段（但 PRE_STREAM 阶段在 RAW 模式中被删除），因此您必须配置 LoggingOutInterceptor 以便在 WRITE 阶段运行。以下是一个示例。

```
@Bean
public CxfEndpoint serviceEndpoint(LoggingOutInterceptor loggingOutInterceptor) {
    CxfSpringEndpoint cxfEndpoint = new CxfSpringEndpoint();
    cxfEndpoint.setAddress("http://localhost:" + port
                    + "/services" + SERVICE_ADDRESS);
    cxfEndpoint.setServiceClass(org.apache.camel.component.cxf.HelloService.class);
    Map<String, Object> properties = new HashMap<String, Object>();
    properties.put("dataFormat", "RAW");
    cxfEndpoint.setProperties(properties);
    cxfEndpoint.getOutInterceptors().add(loggingOutInterceptor);
    return cxfEndpoint;
}


@Bean
public LoggingOutInterceptor loggingOutInterceptor() {
    LoggingOutInterceptor logger = new LoggingOutInterceptor("write");
    return logger;
}
```

26.5.5. relayHeaders 选项的描述

有来自一个 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，作为端点的 WSDL 绑定合同的一部分（如 SOAP 标头）明确定义。

带外 标头是通过线上序列化但不是 WSDL 绑定合同的一部分的标头。

标头转发/过滤是双向的。

当路由具有 CXF 端点且开发人员需要具有 on-the-wire 标头（如 SOAP 标头）时，将路由转发给另一个 JAXWS 端点，然后应将 relayHeaders 设置为 true，这是默认值。

26.5.6. 仅适用于 POJO 模式

relayHeaders=true 表示转发标头的意图。关于给定标头是否被中继到实施 MessageHeadersRelay 接口的可插拔实例的实际决定。将参考 MessageHeadersRelay 的具体实施，以确定是否需要转发标头。已有 SoapMessageHeadersRelay 的实现，它将自身绑定到众所周知的 SOAP 命名空间。目前，仅过滤带外标头，当 relayHeaders=true 时，in-band 标头始终会被转发。如果线上有一个标头，其命名空间对运行时未知，则将使用 fall back DefaultMessageHeadersRelay，这只是允许所有标头进行转发。

relayHeaders=false 设置指定所有标头 in-band 和 out-of-band 应该被丢弃。

您可以插件自己的 MessageHeadersRelay 实现覆盖，或向中继列表添加额外的消息。为了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，覆盖中继必须为您要覆盖的所有命名空间提供服务，否则路由启动时的运行时异常将会抛出，因为这会在命名空间中引入一个模糊性来转发实例映射。

```
<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"/>
```

查看显示如何在这里转发/过滤标头的测试：

https://github.com/apache/camel/blob/main/components/camel-cxf/camel-cxf-spring-soap/src/test/java/org/apache/camel/component/cxf/soap/headers/CxfMessageHeadersRelayTest.java

- 支持 POJO 和 PAYLOAD 模式。在 POJO 模式中，只有带外消息标头可用于过滤，因为已由 CXF 处理并从标头列表中删除。in-band 标头会合并到 POJO 模式的 MessageContentList 中。camel-cxf 组件会有任何尝试从 MessageContentList 中删除带内标头。如果需要过滤带内标头，请使用 PAYLOAD 模式或插入 CXF 端点中的 CXF 拦截器/JAXWS 处理程序。

- 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。下面是一个配置它的示例：

```
@Bean
public HeaderFilterStrategy dropAllMessageHeadersStrategy() {
    CxfHeaderFilterStrategy headerFilterStrategy = new CxfHeaderFilterStrategy();
    headerFilterStrategy.setRelayHeaders(false);
    return headerFilterStrategy;
}
```

然后，您的端点可以引用 CxfHeaderFilterStrategy。

```
@Bean
public CxfEndpoint routerNoRelayEndpoint(HeaderFilterStrategy
dropAllMessageHeadersStrategy) {
    CxfSpringEndpoint cxfEndpoint = new CxfSpringEndpoint();

cxfEndpoint.setServiceClass(org.apache.camel.component.cxf.soap.headers.HeaderTester.class);

cxfEndpoint.setAddress("/CxfMessageHeadersRelayTest/HeaderService/routerNoRelayEndpoint");
    cxfEndpoint.setWsdlURL("soap_header.wsdl");
    cxfEndpoint.setEndpointNameAsQName(
        QName.valueOf("
{http://apache.org/camel/component/cxf/soap/headers}SoapPortNoRelay"));
    cxfEndpoint.setServiceNameAsQName(SERVICENAME);
    Map<String, Object> properties = new HashMap<String, Object>();
    properties.put("dataFormat", "PAYLOAD");
    cxfEndpoint.setProperties(properties);
    cxfEndpoint.setHeaderFilterStrategy(dropAllMessageHeadersStrategy);
    return cxfEndpoint;
}
```

```
@Bean
public CxfEndpoint serviceNoRelayEndpoint(HeaderFilterStrategy
dropAllMessageHeadersStrategy) {
    CxfSpringEndpoint cxfEndpoint = new CxfSpringEndpoint();

cxfEndpoint.setServiceClass(org.apache.camel.component.cxf.soap.headers.HeaderTester.cl
ass);
    cxfEndpoint.setAddress("http://localhost:" + port +
"/services/CxfMessageHeadersRelayTest/HeaderService/routerNoRelayEndpointBackend");
    cxfEndpoint.setWsdlURL("soap_header.wsdl");
    cxfEndpoint.setEndpointNameAsQName(
        QName.valueOf("
{http://apache.org/camel/component/cxf/soap/headers}SoapPortNoRelay"));
    cxfEndpoint.setServiceNameAsQName(SERVICENAME);
    Map<String, Object> properties = new HashMap<String, Object>();
    properties.put("dataFormat", "PAYLOAD");
    cxfEndpoint.setProperties(properties);
    cxfEndpoint.setHeaderFilterStrategy(dropAllMessageHeadersStrategy);
    return cxfEndpoint;
}
```

然后，按如下所示配置路由：

```
rom("cxf:bean:routerNoRelayEndpoint")
    .to("cxf:bean:serviceNoRelayEndpoint");
```

- MessageHeadersRelay 接口稍微改变，并被重命名为 MessageHeaderFilter。它是 CxfHeaderFilterStrategy 的一个属性。以下是配置用户定义的消息标头过滤器的示例：

```
@Bean
public HeaderFilterStrategy customMessageFilterStrategy() {
    CxfHeaderFilterStrategy headerFilterStrategy = new CxfHeaderFilterStrategy();
    List<MessageHeaderFilter> headerFilterList = new ArrayList<MessageHeaderFilter>();
    headerFilterList.add(new SoapMessageHeaderFilter());
    headerFilterList.add(new CustomHeaderFilter());
    headerFilterStrategy.setMessageHeaderFilters(headerFilterList);
    return headerFilterStrategy;
}
```

- 除了 relayHeaders 外，还可以在 CxfHeaderFilterStrategy 中配置以下属性。

Name	必填	描述
relayHeaders	否	所有消息标头都将由 Message Header Filters *Type*:**boolean** *Default*:**true**处理

Name	必填	描述
relayAllMessageHeaders	否	所有消息标头都会被传播（不由消息标头过滤器处理）*Type*:**boolean***Default*:**false**
allowFilterNamespaceClash	否	如果激活命名空间中的两个过滤器重叠，则属性控制应如何处理它。如果值为 **true**，则最后一个优先。如果值为 **false**，它将抛出异常 *类型*：**布尔值** *默认值*:**false**

26.6. 使用 SPRING 配置 CXF 端点

您可以使用如下所示的 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 标头设置为显式状态您要调用的操作。

```
<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/jaxws"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
       http://camel.apache.org/schema/cxf/jaxws
http://camel.apache.org/schema/cxf/jaxws/camel-cxf.xsd
       http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-
spring.xsd">
    <cxf:cxfEndpoint id="routerEndpoint"
address="http://localhost:9003/CamelContext/RouterPort"
          serviceClass="org.apache.hello_world_soap_http.GreeterImpl"/>
    <cxf:cxfEndpoint id="serviceEndpoint"
address="http://localhost:9000/SoapContext/SoapPort"
          wsdlURL="testutils/hello_world.wsdl"
          serviceClass="org.apache.hello_world_soap_http.Greeter"
          endpointName="s:SoapPort"
          serviceName="s:SOAPService"
       xmlns:s="http://apache.org/hello_world_soap_http" />
    <camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
      <route>
        <from uri="cxf:bean:routerEndpoint" />
        <to uri="cxf:bean:serviceEndpoint" />
      </route>
    </camelContext>
  </beans>
```

务必包含 root beans 元素中指定的 JAX-WS schemaLocation 属性。这允许 CXF 验证文件并是必需的。另外，请注意 < cxf:cxfEndpoint /> 标签末尾的命名空间声明。这些声明是必需的，因为此标签的属性值目前不支持组合的 {namespace}localName 语法。

cxf:cxfEndpoint 元素支持许多额外的属性：

Name	值
portName	此服务实施的端点名称，它映射到 **wsdl:port@name**。格式为 **ns:PORT_NAME**，其中 **ns** 是在这个范围内有效的命名空间前缀。
serviceName	此服务正在实现的服务名称，它映射到 **wsdl:service@name**。格式为 **ns:SERVICE_NAME**，其中 **ns** 是在这个范围内有效的命名空间前缀。
wsdlURL	WSDL 的位置。可以位于类路径、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	在 JAX-WS 端点中使用的总线名称。
serviceClass	SEI (Service Endpoint Interface)类的类名称，它们可能具有 JSR181 注释。

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	此端点的传入故障拦截器。< **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	此端点的传出故障拦截器。< **bean> 或 &lt**; **ref>** 列表。
cxf:properties	应提供给 JAX-WS 端点的属性映射。请参见以下内容。
cxf:handlers	个 JAX-WS 处理程序列表，应提供给 JAX-WS 端点。请参见以下内容。
cxf:dataBinding	您可以指定端点中使用的 **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。

Name	值
cxf:features	包含此端点的拦截器的功能。Bean 或 refs 列表
cxf:schemaLocations	要使用的端点的 schema 位置。schemaLocations 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

您可以找到更多高级示例，用于显示如何在 CXF JAX-WS Configuration 页面上 提供拦截器、属性和处理程序。

注意

您可以使用 cxf:properties 从 spring 配置文件中设置 camel-cxf 端点的 dataFormat 和 setDefaultBus 属性。

```
<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="RAW"/>
      <entry key="setDefaultBus" value="true"/>
    </cxf:properties>
  </cxf:cxfEndpoint>
```

注意

在 SpringBoot 中，您可以使用 Spring XML 文件来配置 camel-cxf，并使用类似以下示例的代码来创建 XML 配置的 Bean：

```
@ImportResource({
  "classpath:spring-configuration.xml"
})
```

但是，在 SpringBoot 中使用配置了 Java 代码的 Bean （如其他示例中所示）。

26.7. 如何使 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，请按如下所示进行操作。在 classpath 中创建一个名为 META-INF/cxf/org.apache.cxf.logger 的文件。此文件应该在一行中包含类的完全限定名称 org.apache.cxf.common.logging.Log4jLogger，没有注释。

26.8. 如何让 CAMEL-CXF 响应以 XML 处理指令开头

如果您使用一些 SOAP 客户端，如 PHP，您将获得此类错误，因为 CXF 不会添加 XML 处理指令 < ? 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，如下面的 WriteXmlDeclarationInterceptor 中：

```
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);
    }

}
```

作为替代方案，您可以按照 CxfConsumerTest 所示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);
```

26.9. 如何覆盖来自消息标头的 CXF 生成者地址

camel-cxf producer 支持通过设置消息标头 CamelDestinationOverrideUrl 来覆盖目标服务地址。

```
// 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()));
```

26.10. 如何使用 POJO 数据格式的 CAMEL-CXF 端点中的消息

camel-cxf 端点消费者 POJO 数据格式基于 CXF 调用器，因此消息头具有名为 CxfConstants.OPERATION_NAME 的属性，消息正文是 SEI 方法参数的列表。

考虑 PersonProcessor 示例代码：

```
public class PersonProcessor implements Processor {

    private static final Logger LOG = LoggerFactory.getLogger(PersonProcessor.class);

    @Override
    @SuppressWarnings("unchecked")
    public void process(Exchange exchange) throws Exception {
        LOG.info("processing exchange in camel");

        BindingOperationInfo boi = (BindingOperationInfo)
exchange.getProperty(BindingOperationInfo.class.getName());
        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);
            exchange.getMessage().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.getMessage().setBody(new Object[] { null, personId, ssn, name });
    }

}
```

26.11. 如何以 POJO 数据格式为 CAMEL-CXF 端点准备消息

camel-cxf 端点制作者基于 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 method 参数添加到列表中，并使用此参数列表初始化消息。响应消息的正文是一个 messageContentsList，您可以从该列表中获取结果。

如果您没有在消息标头中指定操作名称，CxfProducer 将尝试使用来自 CxfEndpoint 的 defaultOperationName，如果在 CxfEndpoint 上没有设置defaultOperationName，它将从 Operation 列表中选择第一个 operationName。

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 CxfProducerRouterTest.testInvokingSimpleServerWithParams 所示：

```
Exchange senderExchange = new DefaultExchange(context, ExchangePattern.InOut);
final List<String> params = new ArrayList<>();
// 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.getMessage();
// 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 will be filled in the reset of List.
// The result will be 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("UTF-8", responseContext.get(org.apache.cxf.message.Message.ENCODING),
        "We should get the response context here");
assertEquals("echo " + TEST_MESSAGE, result.get(0), "Reply body on Camel is wrong");
```

26.12. 如何处理 PAYLOAD 数据格式的 CAMEL-CXF 端点的消息

PAYLOAD 表示您将来自 SOAP 信封的载荷处理为原生 CxfPayload。message .getBody （） 将返回 org.apache.camel.component.cxf.CxfPayload 对象，带有 SOAP 消息标头和 SOAP 正文的 getters。

请参阅 CxfConsumerPayloadTest:

```
protected RouteBuilder createRouteBuilder() {
    return new RouteBuilder() {
        public void configure() {
            from(simpleEndpointURI + "&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<>();
                    // 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,
exchange);
                    outElements.add(new DOMSource(outDocument.getDocumentElement()));
                    // set the payload header with null
                    CxfPayload<SoapHeader> responsePayload = new CxfPayload<>(null,
outElements, null);
                    exchange.getMessage().setBody(responsePayload);
                }
            });
        }
    };
}
```

26.13. 如何在 POJO 模式中获取和设置 SOAP 标头

POJO 表示当 camel-cxf 端点生成或消耗 Camel 交换时，数据格式是 Java 对象的"列表"。虽然 Camel 在此模式下公开消息正文作为 POJO，但 camel-cxf 仍提供对读取和写入 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中的 SOAP 标头，因此只有 POJO 模式中的 camel-cxf 使用带外 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 Camel-cxf 端点转发到另一个 Camel-cxf 端点。也就是说，SOA Client → Camel → CXF 服务。在请求离开 CXF 服务之前，我们可以附加两个处理器在(1)处获取/插入 SOAP 标头，然后再将响应返回 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：

```
from("cxf:bean:routerRelayEndpointWithInsertion")
    .process(new InsertRequestOutHeaderProcessor())
    .to("cxf:bean:serviceRelayEndpointWithInsertion")
    .process(new InsertResponseOutHeaderProcessor());
```

Bean routerRelayEndpointWithInsertion 和 serviceRelayEndpointWithInsertion 的定义如下：

```
@Bean
public CxfEndpoint routerRelayEndpointWithInsertion() {
    CxfSpringEndpoint cxfEndpoint = new CxfSpringEndpoint();

cxfEndpoint.setServiceClass(org.apache.camel.component.cxf.soap.headers.HeaderTester.cl
ass);

cxfEndpoint.setAddress("/CxfMessageHeadersRelayTest/HeaderService/routerRelayEndpoint
WithInsertion");
    cxfEndpoint.setWsdlURL("soap_header.wsdl");
    cxfEndpoint.setEndpointNameAsQName(
        QName.valueOf("
{http://apache.org/camel/component/cxf/soap/headers}SoapPortRelayWithInsertion"));
    cxfEndpoint.setServiceNameAsQName(SERVICENAME);
    cxfEndpoint.getFeatures().add(new LoggingFeature());
    return cxfEndpoint;
}

@Bean
public CxfEndpoint serviceRelayEndpointWithInsertion() {
    CxfSpringEndpoint cxfEndpoint = new CxfSpringEndpoint();

cxfEndpoint.setServiceClass(org.apache.camel.component.cxf.soap.headers.HeaderTester.cl
ass);
    cxfEndpoint.setAddress("http://localhost:" + port +
"/services/CxfMessageHeadersRelayTest/HeaderService/routerRelayEndpointWithInsertionBa
ckend");
    cxfEndpoint.setWsdlURL("soap_header.wsdl");
    cxfEndpoint.setEndpointNameAsQName(
        QName.valueOf("
{http://apache.org/camel/component/cxf/soap/headers}SoapPortRelayWithInsertion"));
    cxfEndpoint.setServiceNameAsQName(SERVICENAME);
    cxfEndpoint.getFeatures().add(new LoggingFeature());
    return cxfEndpoint;
}
```

SOAP 标头被传播到 Camel 消息标头，或从 Camel 消息标头传播。Camel 消息标头名称为 "org.apache.cxf.headers.Header.list"，它是 CXF (org.apache.cxf.headers.Header.HEADER_LIST) 中定义的常数。标头值是 CXF SoapHeader 对象(org.apache.cxf.binding.soap.SoapHeader)的列表。以下代码片段是 InsertResponseOutHeaderProcessor （在响应消息中插入一个新的 SOAP 标头）。

在 InsertResponseOutHeaderProcessor 和 InsertRequestOutHeaderProcessor 中访问 SOAP 标头的方式实际上相同。两个处理器之间的唯一区别在于设置插入 SOAP 标头的方向。

您可以在 CxfMessageHeadersRelayTest 中找到 InsertResponseOutHeaderProcessor 示例：

```
public static class InsertResponseOutHeaderProcessor implements Processor {

    public void process(Exchange exchange) throws Exception {
        List<SoapHeader> soapHeaders = CastUtils.cast((List<?
>)exchange.getIn().getHeader(Header.HEADER_LIST));

        // 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);

    }

}
```

26.14. 如何在 PAYLOAD 模式中获取和设置 SOAP 标头

我们已显示了如何在 PAYLOAD 模式中以 CxfPayload 对象的形式访问 SOAP 消息，部分将如何处理 PAYLOAD 数据格式的 camel-cxf 端点 的消息。

获取 CxfPayload 对象后，您可以调用返回 DOM Elements (SOAP 标头)的 CxfPayload.getHeaders（）方法。

有关示例，请参阅 CxfPayLoadSoapHeaderTest ：

```
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, "We should get the elements here");
        assertEquals(1, elements.size(), "Get the wrong elements size");
```

```
        Element el = new XmlConverter().toDOMElement(elements.get(0));
        elements.set(0, new DOMSource(el));
        assertEquals("http://camel.apache.org/pizza/types",
                el.getNamespaceURI(), "Get the wrong namespace URI");

        List<SoapHeader> headers = payload.getHeaders();
        assertNotNull(headers, "We should get the headers here");
        assertEquals(1, headers.size(), "Get the wrong headers size");
        assertEquals("http://camel.apache.org/pizza/types",
                ((Element) (headers.get(0).getObject())).getNamespaceURI(), "Get the wrong
namespace URI");
        // alternatively you can also get the SOAP header via the camel header:
        headers = exchange.getIn().getHeader(Header.HEADER_LIST, List.class);
        assertNotNull(headers, "We should get the headers here");
        assertEquals(1, headers.size(), "Get the wrong headers size");
        assertEquals("http://camel.apache.org/pizza/types",
                ((Element) (headers.get(0).getObject())).getNamespaceURI(), "Get the wrong
namespace URI");

    }

})
.to(getServiceEndpointURI());
```

您还可以使用与子章节相同的方法"如何在 POJO 模式中获取和设置 SOAP 标头，以设置或获取 SOAP 标头。因此，您可以使用标头 "org.apache.cxf.headers.Header.list" 获取和设置 SOAP 标头列表。这也意味着，如果您有一个路由，它从一个 Camel-cxf 端点转发到另一个(SOAP Client → Camel → CXF 服务)，现在也转发到 SOAP 客户端发送的 SOAP 标头。如果您不想转发这些标头，则必须在 Camel 标头 "org.apache.cxf.headers.Header.list" 中删除。

26.15. SOAP 标头在 RAW 模式中不可用

SOAP 标头在 RAW 模式中不可用，因为跳过 SOAP 处理。

26.16. 如何从 CAMEL 丢弃 SOAP FAULT

如果您使用 camel-cxf 端点来使用 SOAP 请求，您可能需要从 camel 上下文中丢弃 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。
您可以按照 CxfCustomizedExceptionTest 所示定义 soap 故障：

```
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);
```

然后像您一样丢弃它

```
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，如 CxfMessageStreamExceptionTest所示

```
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));
    }

});
```

相同的使用 POJO 数据格式。您可以在 out 正文上设置 SOAPFault。

26.17. 如何传播 CAMEL-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 camel-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 operation name", "
{http://apache.org/hello_world_soap_http}greetMe",
                     responseContext.get("javax.xml.ws.wsdl.operation").toString());
```

26.18. 附加支持

POJO 模式： 支持带有 Attachment 和 MTOM 的 SOAP （请参阅 Payload Mode for enable MTOM）。但是，没有测试带有 Attachment 的 SOAP。由于附件被放入 POJO 中，因此用户通常不需要自行处理附件。如果没有启用 MTOM，则附件会被传播到 Camel 消息的附件。因此，可以通过 Camel Message API 检索附件

```
DataHandler Message.getAttachment(String id)
```

有效负载模式： 组件支持 MTOM。附件可以通过上述 Camel 消息 API 检索。支持带有 Attachment (SwA)的 SOAP，并可以检索附件。SwA 是默认值（与将 CXF 端点属性"mtom-enabled"设置为 false 的相同）。

要启用 MTOM，将 CXF 端点属性 "mtom-enabled" 设置为 *true*。

```
@Bean
public CxfEndpoint routerEndpoint() {
   CxfSpringEndpoint cxfEndpoint = new CxfSpringEndpoint();
   cxfEndpoint.setServiceNameAsQName(SERVICE_QNAME);
   cxfEndpoint.setEndpointNameAsQName(PORT_QNAME);
   cxfEndpoint.setAddress("/" + getClass().getSimpleName()+ "/jaxws-mtom/hello");
   cxfEndpoint.setWsdlURL("mtom.wsdl");
   Map<String, Object> properties = new HashMap<String, Object>();
   properties.put("dataFormat", "PAYLOAD");
   properties.put("mtom-enabled", true);
   cxfEndpoint.setProperties(properties);
   return cxfEndpoint;
}
```

您可以生成带有附件的 Camel 消息，以发送到 Payload 模式的 CXF 端点。

```
Exchange exchange = context.createProducerTemplate().send("direct:testEndpoint", new
Processor() {

   public void process(Exchange exchange) throws Exception {
      exchange.setPattern(ExchangePattern.InOut);
      List<Source> elements = new ArrayList<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());
```

您还可以使用在 Payload 模式中从 CXF 端点接收的 Camel 消息。CxfMtomConsumerPayloadModeTest 演示了如何工作：

```
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
        Assert.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:"
        Assert.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:"
        Assert.assertEquals(MtomTestHelper.REQ_IMAGE_CID, imageId);

        DataHandler dr = exchange.getIn().getAttachment(photoId);
        Assert.assertEquals("application/octet-stream", dr.getContentType());
        MtomTestHelper.assertEquals(MtomTestHelper.REQ_PHOTO_DATA,
IOUtils.readBytesFromStream(dr.getInputStream()));

        dr = exchange.getIn().getAttachment(imageId);
        Assert.assertEquals("image/jpeg", dr.getContentType());
        MtomTestHelper.assertEquals(MtomTestHelper.requestJpeg,
IOUtils.readBytesFromStream(dr.getInputStream()));

        // create response
        List<Source> elements = new ArrayList<Source>();
        elements.add(new DOMSource(DOMUtils.readXml(new
StringReader(MtomTestHelper.RESP_MESSAGE)).getDocumentElement()));
        CxfPayload<SoapHeader> sbody = new CxfPayload<SoapHeader>(new
ArrayList<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")));

    }
}
```

原始模式： 不支持附件，因为它根本不处理消息。

CXF_RAW 模式 ：支持 MTOM，可以通过上述 Camel 消息 API 检索附件。请注意，当收到多部分

（即 MTOM）消息时，默认 SOAPMessage 到 String converter 将提供正文上的完整多部分有效负载。如果您只需要 SOAP XML 作为字符串，您可以使用 message.getSOAPPart （）设置消息正文，Camel convert 可以为您执行其余工作。

26.19. PAYLOAD 模式中的流支持

camel-cxf 组件现在支持在使用 PAYLOAD 模式时流传输传入的信息。在以前的版本中，传入的信息会完全解析 DOM。对于大型消息，这非常耗时，并且使用了大量内存。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"simple proxy"用例（例如： from ("cxf:...").to ("cxf:...")，这可能会造成非常显著的性能增加，并显著降低了内存要求。

然而，在有些情况下，流可能不合适或需要。由于流性质，在处理链中稍后之前，无效的传入 XML 可能无法被发现。此外，某些操作可能需要通过 DOM 解析消息，如 WS-Security 或消息追踪等，在这种情况下，流的优势有限。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，以打开流 on/off。
- 组件属性：CxfComponent 对象也有一个 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：您可以将 "org.apache.camel.component.cxf.streaming" 的系统属性添加到 "false" 以将其关闭。这会设置全局默认值，但设置上面的 endpoint 属性将覆盖该端点的此值。

26.20. 使用通用 CXF DISPATCH 模式

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，您可以省略指定 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。

26.21. 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.cxf.allow-streaming	这个选项控制在 PAYLOAD 模式运行时 CXF 组件是否会将传入的消息解析为 DOM Elements，或将有效负载保留为 javax.xml.transform.Source 对象，在某些情况下允许流。		布尔值
camel.component.cxf.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.cxf.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.cxf.enabled	是否启用 cxf 组件的自动配置。这默认是启用的。		布尔值
camel.component.cxf.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头到 Camel 消息。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		HeaderFilterStrategy
camel.component.cxf.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.cxf.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.cxfrs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.cxfrs.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.cxfrs.enabled	是否启用 cxfrs 组件的自动配置。这默认是启用的。		布尔值
camel.component.cxfrs.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头到 Camel 消息。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		HeaderFilterStrategy
camel.component.cxfrs.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.cxfrs.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

第 27 章 CXF-RS

支持生成者和消费者

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。

27.1. 依赖项

当在 Camel Spring Boot 中使用 camel-cxf-rest 时，请将以下 Maven 依赖项添加到 pom.xml 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-cxf-rest-starter</artifactId>
</dependency>
```

27.2. URI 格式

```
cxfrs://address?options
```

其中 地址 代表 CXF 端点的地址。

```
cxfrs:bean:rsEndpoint
```

其中 rsEndpoint 代表 spring bean 的名称，它会显示 CXFRS 客户端或服务器。

对于以上格式，您可以将选项附加到 URI 中，如下所示：

```
cxfrs:bean:cxfEndpoint?resourceClasses=org.apache.camel.rs.Example
```

27.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别

- 端点级别

27.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

27.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

27.4. 组件选项

CXF-RS 组件支持 5 个选项，如下所列。

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 Camel 消费者试图获取传入的消息或类似信息时，会出现异常（如果可能），现在将被作为消息进行处理，并由路由 Error Handler 处理。重要：只有在第三方组件允许 Camel 抛出异常时，才能警报这一点。有些组件仅在内部处理，因此无法 bridgeErrorHandler。在其他情况下，我们可能会将 Camel 组件提高到第三方组件中的 hook，并使其可能用于将来的版本。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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头到 Camel 消息。		HeaderFilterStrategy
useGlobalSslContextParameters (security)	启用使用全局 SSL 上下文参数。	false	布尔值

27.5. 端点选项

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* ：

27.5.1. 路径参数(2 参数)

Name	描述	默认值	类型
beanId (common)	查找现有的已配置的 CxfRsEndpoint。必须使用 **bean:** 作为前缀。		字符串
address (common)	服务发布地址。		字符串

27.5.2. 查询参数(31 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)	要限制启用日志记录功能时日志记录器将输出的字节数。		int
modelRef (common)	这个选项用于指定模型文件，该文件对于没有注解的资源类有用。使用此选项时，可以省略服务类，以模拟仅文档端点。		字符串
providers (common)	将自定义 JAX-RS 提供程序列表设置为 CxfRs 端点。您可以使用供应商列表指定字符串，以便在以逗号分开的 registy 中查找。		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)	设置可用于验证传入 XML 或 JAXB 驱动的 JSON 的架构位置。		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

Name	描述	默认值	类型
bindingStyle (consumer)	设置请求和响应如何映射到/来自 Camel。可以有两个值：simpleConsumer ：这种绑定风格进程请求参数、多部分等，并将它们映射到 IN 标头、IN attachments 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单性。仅适用于消费者。默认：默认样式。对于消费者，这会将 MessageContentsList 传递给路由，这需要路由中的低级处理。这是传统的绑定风格，只是将来自 CXF 堆栈的 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。 Enum 值： • SimpleConsumer • 默认值 • Custom	默认值	BindingStyle
publishedEndpointUrl (consumer)	这个选项可以覆盖从 WADL 发布的 endpointUrl，它可以通过资源地址 url 和 _wadl 访问。		字符串
bridgeErrorHandler (consumer (advanced))	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当 Camel 消费者试图获取传入的消息或类似信息时，会出现异常（如果可能），现在将被作为消息进行处理，并由路由 Error Handler 处理。重要：只有在第三方组件允许 Camel 抛出异常时，才能警报这一点。有些组件仅在内部处理，因此无法 bridgeErrorHandler。在其他情况下，我们可能会将 Camel 组件提高到第三方组件中的 hook，并使其可能用于将来的版本。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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut		ExchangePattern
servicebeans (consumer (advanced))	要导出为 REST 服务的 registry 中的服务 Bean （用于查找的 bean id）。可以使用逗号分隔多个 Bean。		字符串
cookieHandler (producer)	配置 Cookie 处理程序来维护 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，如果返回代码大于 207，将生成一个例外。	true	布尔值
httpClientAPI (producer (advanced))	如果为 true，则 CxfRsProducer 将使用 HttpClientAPI 调用该服务。如果是 false，则 CxfRsProducer 将使用 ProxyClientAPI 调用该服务。	true	布尔值
ignoreDeleteMethodMessageBody (producer (advanced))	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
lazyStartProducer (producer (advanced))	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
maxClientCacheSize (producer (advanced))	这个选项允许您配置缓存的最大大小。实施在 CxfProvider 和 CxfRsProvider 中缓存 CXF 客户端或 ClientFactoryBean。	10	int
同步 (producer (advanced))	设置是否应严格使用同步处理。	false	布尔值

Name	描述	默认值	类型
绑定 (advanced)	使用自定义 CxfBinding 控制 Camel 消息和 CXF 消息之间的绑定。		CxfRsBinding
Bus (advanced)	使用自定义配置的 CXF 总线。		bus
continuationTimeout (advanced)	这个选项用于设置 CXF 持续超时，在 CXF 服务器使用 Jetty 或 Servlet 传输时，默认可在 CxfConsumer 中使用。	30000	long
cxfRsConfigurer (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 configure\\{Server/Client} 方法来配置 CXF 服务器和客户端。		CxfRsConfigurer
defaultBus (advanced)	当 CXF 端点本身创建总线时，将设置默认总线。	false	布尔值
headerFilterStrategy (advanced)	使用自定义 HeaderFilterStrategy 过滤标头到 Camel 消息。		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，以进行进一步处理。	false	布尔值
propagateContexts (advanced)	当选项为 true 时，JAXRS UriInfo、HttpHeaders、Request 和 SecurityContext 上下文将作为类型 Camel Exchange 属性提供给自定义 CXFRS 处理器。这些上下文可用于使用 JAX-RS API 分析当前的请求。	false	布尔值

27.6. 消息标头

CXF-RS 组件支持 16 个消息标头，如下所列：

Name	描述	Default（默认）	类型
operationName (common) 恒定：OPERATION_NAME	操作的名称。		字符串

Name	描述	Default（默认）	类型
CamelAuthentication (common) 常数：AUTHENTICATION	身份验证。		主题
CamelHttpMethod (common) 常数：HTTP_METHOD	要使用的 http 方法。		字符串
CamelHttpPath (common) 常数：HTTP_PATH	http 路径。		字符串
content-Type (common) 常数：CONTENT_TYPE	内容类型。		字符串
CamelHttpQuery (common) 常数：HTTP_QUERY	http 查询。		字符串
CamelHttpResponseCode (common) 常量：HTTP_RESPONSE_CODE	http 响应代码。		整数
content-Encoding (common) 常量：CONTENT_ENCODING	内容编码。		字符串

Name	描述	Default（默认）	类型
org.apache.cxf.message.Message.PROTOCOL_HEADERS (common) 常量：PROTOCOL_HEADERS	协议标头。		Map
CamelCxfMessage (common) 常量：CAMEL_CXF_MESSAGE	CXF 消息。		消息
CamelCxfRsUsingHttpAPI (common) 常量：CAMEL_CXF_RS_USING_HTTP_API	如果为 true，则 CxfRsProducer 将使用 HttpClientAPI 调用该服务。如果是 false，则 CxfRsProducer 将使用 ProxyClientAPI 调用该服务。		布尔值
CamelCxfRsVarValues (common) 常量：CAMEL_CXF_RS_VAR_VALUES	路径值。		Object[]
CamelCxfRsResponseClass (common) 恒定：CAMEL_CXF_RS_RESPONSE_CLASS	响应类。		类
CamelCxfRsResponseGenericType (common) 恒定：CAMEL_CXF_RS_RESPONSE_GENERIC_TYPE	响应通用类型。		类型

Name	描述	Default（默认）	类型
CamelCxfRsQueryMap (common) 常量：CAMEL_CXF_RS_QUERY_MAP	查询映射。		Map
CamelCxfRsOperationResourceInfoStack (common) 常量：CAMEL_CXF_RS_OPERATION_RESOURCE_INFO_STACK	当 JAX-RS 调用查找目标时代表资源路径的 MethodInvocationInfo 的堆栈。		OperationResourceInfoStack

您还可以通过 spring 配置配置 CXF REST 端点。

注意

由于 CXF REST 客户端和 CXF REST 服务器之间存在很多差异，我们为它们提供不同的配置。

请检查以下文件以了解更多详情：

- 架构文件.
- CXF JAX-RS 文档.

27.7. 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：

- 用于 REST 使用者的 cxf:rsServer

- 用于 REST 生成者的 cxf:rsClient.

您可以在那里找到 Camel REST 服务路由配置示例。

27.8. 如何覆盖来自消息标头的 CXF 生成者地址

camel-cxfrs producer 支持通过使用 CamelDestinationOverrideUrl 键设置消息来覆盖服务地址。

```
// 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()));
```

27.9. 消耗 REST 请求 - SIMPLE BINDING STYLE

从 Camel 2.11 开始

默认 绑定风格是低级，要求用户手动处理进入路由的 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，这稍微困难且容易出错。

相反，simple Consumer 绑定风格会执行以下映射，以便在 Camel 消息中 更方便地访问请求数据 ：

- JAX-RS 参数(@HeaderParam、@QueryParam 等等)作为 *IN* 消息标头注入。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 MessageContentsList。
- 二进制 @Multipart body parts become *IN* message attachments, support DataHandler,InputStream,DataSource 和 CXF 的 Attachment 类。
- 非二进制 @Multipart body 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 响应)不同，则会创建一个新的 Response，并且消息正文被设置为实体（因此不是 null）。响应状态代码从 Exchange.HTTP_RESPONSE_CODE 标头中获取，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，这表示用户已构建了一个自定义响应，因此被遵守，它将成为最终响应。

- 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。

27.9.1. 启用简单绑定样式

通过将消费者端点中的 bindingStyle 参数设置为 SimpleConsumer，可以激活此绑定风格：

```
from("cxfrs:bean:rsServer?bindingStyle=SimpleConsumer")
  .to("log:TEST?showAll=true");
```

27.9.2. 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简单绑定的预期结果：

- 公共响应 doAction (BusinessObject request);：请求有效负载放置在 tbe *IN* 消息正文中，替换原始 MessageContentsList。

- 公共响应 doAction (BusinessObject request, @HeaderParam ("abcd") String abcd, @QueryParam ("defg") String defg);: 请求有效负载放置在 *IN* 消息正文中，替换原始 MessageContentsList。这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);: 这两个请求参数都映射为 *IN* message 标头，名称为 *"abcd"* 和 *"defg* "。原始 MessageContentsList 被保留，即使它只包含两个参数。

- 公共响应 doAction (@Multipart (value="body1") BusinessObject request, @Multipart (value="body2") BusinessObject request2);: first 参数作为名为 *"body1"* 的标头传输，第二个参数被映射为标题 *"body2* "。原始 MessageContentsList 保留为 *IN* 消息正文。

-

公共响应 doAction (InputStream abcd);: InputStream is unwrapped from the MessageContentsList and reserved as the *IN* message body.

- 公共响应 doAction (DataHandler abcd);: *DataHandler* 从 MessageContentsList 中分离，并保留为 *IN* 消息正文。

27.9.3. Simple Binding Style 的示例

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;
}
```

由以下路由提供服务：

```
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（客户 DTO 为 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>
```

注意

有关如何处理请求和写入响应的更多示例，请参考 https://svn.apache.org/repos/asf/camel/trunk/components/camel-

cxf/src/test/java/org/apache/camel/component/cxf/jaxrs/simplebinding/。

27.10. 消耗 REST 请求 - 默认绑定 STYLE

CXF JAXRS 前端 实施 JAX-RS (concurrency-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。不需要在端点中指定 URI 模板。CXF 根据 JSR-311 规范，负责 REST 请求 URI 到资源类方法映射。您需要在 Camel 中做的只是将此方法请求委派给正确的处理器或端点。

CXFRS 路由示例

```
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);
```

```
        }
    };
}
```

对应的资源类则用于配置端点。

注意

默认情况下，JAX-RS 资源类 仅用于 配置 JAX-RS 属性。在将消息路由到端点时不会执行方法。相反，负责执行所有处理。

对于默认模式，只提供一个接口，而不是 no-op 服务实现类。

如果启用了 performInvocation 选项，将首先调用服务实施，响应将在 Camel 交换上设置，并且路由执行将照常继续。这可用于将现有 JAX-RS 实施集成到 Camel 路由和自定义处理器中的后处理 JAX-RS 响应。

```
@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);
}
```

27.11. 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。使用这个 API，您可以通过代理调用远程 REST 服务。camel-cxfrs producer 基于这个代理 API。您可以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 producer 将为您生成正确的 REST 请求。

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");
        // set up the accepted 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.getMessage().getBody();

assertNotNull(response, "The response should not be null");
assertEquals(123, response.getId(), "Get a wrong customer id");
assertEquals("John", response.getName(), "Get a wrong customer name");
assertEquals(200, exchange.getMessage().getHeader(Exchange.HTTP_RESPONSE_CODE),
"Get a wrong response code");
assertEquals("value", exchange.getMessage().getHeader("key"), "Get a wrong header value");
```

CXF JAXRS 前端 还提供以 HTTP 为中心的客户端 API。您也可以从 camel-cxfrs producer 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。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);
    }
});
```

您还可以为 CXFRS http 以中心客户端指定 cxfrs URI 中的查询参数。

```
Exchange exchange = template.send("cxfrs://http://localhost:9003/testQuery?
httpClientAPI=true&q1=12&q2=13"
```

要支持动态路由，您可以使用 CxfConstants.CAMEL_CXF_RS_QUERY_MAP 标头覆盖 URI 的查询参数，为其设置它的参数映射。

```
Map<String, String> queryMap = new LinkedHashMap<>();
queryMap.put("q1", "new");
queryMap.put("q2", "world");
inMessage.setHeader(CxfConstants.CAMEL_CXF_RS_QUERY_MAP, queryMap);
```

27.12. 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.cxfrs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.cxfrs.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 Camel 消费者试图获取传入的消息或类似信息时，会出现异常（如果可能），现在将被作为消息进行处理，并由路由 Error Handler 处理。重要：只有在第三方组件允许 Camel 抛出异常时，才能警报这一点。有些组件仅在内部处理，因此无法 bridgeErrorHandler。在其他情况下，我们可能会将 Camel 组件提高到第三方组件中的 hook，并使其可能用于将来的版本。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.cxfrs.enabled	是否启用 cxfrs 组件的自动配置。这默认是启用的。		布尔值
camel.component.cxfrs.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头到 Camel 消息。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		HeaderFilterStrategy
camel.component.cxfrs.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.cxfrs.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

第 28 章 数据格式

仅支持生成者

Dataformat 组件允许使用数据格式 作为 Camel 组件。

28.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 dataformat 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-dataformat-starter</artifactId>
</dependency>
```

28.2. 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。选项用于配置正在使用的 数据格式。有关支持哪些选项，请参阅数据格式文档。

28.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

28.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

28.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

28.4. 组件选项

Data Format 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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

28.5. 端点选项

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：

28.5.1. 路径参数(2 参数)

Name	描述	默认值	类型
name (producer)	**数据格式** 的必要名称。		字符串
operation (producer)	**需要的** Operation 来使用 marshal 或 unmarshal。 Enum 值： • marshal • unmarshal		字符串

28.5.2. 查询参数(1 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

28.6. SAMPLES

例如，要使用 JAXB 数据格式，我们可以执行以下操作：

```
from("activemq:My.Queue").
  to("dataformat:jaxb:unmarshal?contextPath=com.acme.model").
  to("mqseries:Another.Queue");
```

在 XML DSL 中：

```
<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>
```

28.7. 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.dataformat.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.dataformat.enabled	是否启用 dataformat 组件的自动配置。这默认是启用的。		布尔值
camel.component.dataformat.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

第 29 章 DATASET

支持生成者和消费者

测试分布式和异步处理并不困难。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 组件提供了一种机制，可以轻松执行您系统的负载和 soak 测试。它的工作原理是，允许您将 DataSet 实例创建 为消息源，并作为接收数据集的方法。

Camel 在发送数据集 时将使用吞吐量日志记录器。

29.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 dataset 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-dataset-starter</artifactId>
</dependency>
```

29.2. URI 格式

```
dataset:name[?options]
```

在 Registry 中查找 DataSet 实例的名称

Camel 附带了 org.apache.camel.component.dataset.DataSet （即 org.apache.camel.component.dataset.DataSetSupport 类）的支持实现，可用于实施您自己的 DataSet。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，它们扩展 DataSetSupport。

29.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

29.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

29.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

29.4. 组件选项

Dataset 组件支持 5 个选项，如下所列。

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
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
log (producer)	在模拟收到传入消息时打开日志记录。这只会记录传入消息的 INFO 级别一次。如需更详细的日志记录，然后将 org.apache.camel.component.mock.MockEndpoint 类的 logger 设置为 DEBUG 级别。	false	布尔值
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
exchangeFormatter (advanced)	**Autowired** 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，我们默认为 DefaultExchangeFormatter。		ExchangeFormatter

29.5. 端点选项

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：

29.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
name (common)	在 registry 中查找的 DataSet **所需的** 名称。		DataSet

29.5.2. 查询参数(21 参数)

Name	描述	默认值	类型
dataSetIndex (common)	控制 CamelDataSetIndex 标头的行为。对于 Consumers: - off = header is not be set - strict/lenient = the header will be set For Producers: - off = header value is not be enabled, will not present = strict = header value be present，并将被验证 = lenient = 标头值（如果不存在），则将设置该标头值。 Enum 值： • strict • lenient • off	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 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
initialDelay (consumer)	开始发送消息前等待的时间周期。	1000	long
minRate (consumer)	等待 DataSet 包含至少此数量的信息。	0	int
preloadSize (consumer)	设置在路由完成其初始化前应预加载(sent)的消息数量。	0	long
produceDelay (consumer)	指定一个延迟，这会在消费者发送消息时造成延迟（模拟缓慢处理）。	3	long
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
assertPeriod (producer)	设置一个宽限期，在此期内，模拟端点将重新开始处理，以确保初始断言仍然有效。这用于 assert，用于准确有多个消息到达的信息。例如，如果 expectedMessageCount (int)被设置为 5，则在 5 或更多消息到达时满足断言。为确保正好 5 个消息到达，您需要稍等片刻，以确保没有进一步的消息到达。这是您可以使用此方法的方法。默认情况下禁用此周期。		long
consumeDelay (producer)	指定一个延迟，这会在生成者消耗消息时造成延迟（模拟较慢的处理）。	0	long
expectedCount (producer)	指定此端点应接收的预期消息交换数量。注意：如果您希望 0 个信息，然后进行额外操作，如测试启动时为 0 匹配，因此您需要设置一个 assert 周期，以便让测试运行一段时间，以确保还没有到达消息；对于使用 setAssertPeriod (long)。另一种方法是使用 NotifyBuilder，并在对模拟调用 assertIsSatisfied （）方法前，使用 notifier 知道 Camel 何时完成一些消息。这可让您不要使用固定的 assert 周期来加快测试时间。如果您要断言正好 n 个消息到达这个 mock 端点，那么还会看到 setAssertPeriod (long)方法了解更多详情。	-1	int
failfast (producer)	设置 assertIsSatisfied （）应该在第一次检测到的失败预期失败时快速失败，否则可能会等待所有预期消息在执行预期验证前到达。默认为 true。设置为 false 以使用 Camel 2.x 中的行为。	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

Name	描述	默认值	类型
log (producer)	在模拟收到传入消息时打开日志记录。这只会记录传入消息的 INFO 级别一次。如需更详细的日志记录，然后将 org.apache.camel.component.mock.MockEndpoint 类的 logger 设置为 DEBUG 级别。	false	布尔值
reportGroup (producer)	用于根据大小组打开吞吐量日志的数字。		int
resultMinimumWaitTime (producer)	设置预期时间（在 millis 中），assertIsSatisfied （）将等待一个白板，直到其满足为止。		long
resultWaitTime (producer)	设置 assertIsSatisfied （）将等待的最长时间（在 millis 中），直到其满足为止。		long
retainFirst (producer)	指定只保留第 n 个接收的交换数。这在使用大数据测试时使用，通过不存储每个交换端点接收的副本来减少内存消耗。重要：使用这个限制时，getReceivedCounter （）仍会返回接收的交换的实际数量。例如，如果我们收到 5000 Exchange，并且已配置为只保留前 10 个交换，则 getReceivedCounter （）仍会返回 5000，但 getExchanges （）和 getReceivedExchanges （）方法中只有前 10 个交换。使用此方法时，不支持一些其他预期方法，例如 expectedBodiesReceived (Object...)在收到第一个数量的正文上设置预期。您可以配置 setRetainFirst (int)和 setRetainLast (int)方法，以限制第一个和最后一个接收的限制。	-1	int
retainLast (producer)	指定只保留最后 n 个接收的交换数。这在使用大数据测试时使用，通过不存储每个交换端点接收的副本来减少内存消耗。重要：使用这个限制时，getReceivedCounter （）仍会返回接收的交换的实际数量。例如，如果我们收到 5000 Exchange，并且已配置为只保留最后 20 个交换，则 getReceivedCounter （）仍会返回 5000，但 getExchanges （）和 getReceivedExchanges （）方法中只有最后 20 个交换。使用此方法时，不支持一些其他预期方法，例如 expectedBodiesReceived (Object...)在收到第一个数量的正文上设置预期。您可以配置 setRetainFirst (int)和 setRetainLast (int)方法，以限制第一个和最后一个接收的限制。	-1	int
sleepForEmptyTest (producer)	当 expectedMessageCount (int)被调用为零时，允许指定 sleep 来检查此端点是否确实为空。		long
copyOnExchange (producer (advanced))	设置在这个模拟端点收到时是否制作传入 Exchange 的深度副本。默认为 true。	true	布尔值

29.6. 配置 DATASET

Camel 将在 Registry 中查找实现 DataSet 接口的 bean。因此，您可以将自己的 DataSet 注册为：

```
<bean id="myDataSet" class="com.mycompany.MyDataSet">
  <property name="size" value="100"/>
</bean>
```

29.7. 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，而不会丢失任何信息：

```
// 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，如配置诸如数据集大小以及消息类似于 etc 的情况等。

29.8. DATASETSUPPORT (ABSTRACT 类)

DataSetSupport 抽象类是新 DataSets 的 nice 起点，它为派生类提供一些有用的功能。

29.8.1. 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 是一个恒定的有效负载；但是，如果要为每个消息创建自定义有效负载，请创建自己的数据Set **Support 派生**。
outputTransformer	**org.apache.camel.Processor**	null	

属性	类型	Default（默认）	描述
size	**long**	**10**	指定要发送/消耗的消息数量。
reportCount	**long**	**-1**	指定在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，如果大小为 0，则设为 **reportCount** 值。

29.9. SIMPLEDATASET

SimpleDataSet 扩展 DataSetSupport，并添加默认的正文。

29.9.1. SimpleDataSet 的额外属性

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** 会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。

29.10. LISTDATASET

List'DataSet' 扩展 DataSetSupport，并添加默认正文列表。

29.10.1. ListDataSet 的额外属性

属性	类型	Default（默认）	描述

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/消耗的消息数量。这个值可以与 **defaultBodies** 列表的大小不同。如果值小于 **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，则将使用 **CamelDataSetIndex** 的 modulus 和 defaultBodies 列表的大小(i.e. **CamelDataSetIndex % defaultBodies.size ()** 来选择交换的有效负载

29.11. FILEDATASET

FileDataSet 扩展 ListDataSet，并添加了对从文件中加载正文的支持。

29.11.1. FileDataSet 的额外属性

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** 用来将文件拆分为多个有效负载的分隔符模式。

29.12. 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.dataset-test.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.dataset-test.enabled	是否启用 dataset-test 组件的自动配置。这默认是启用的。		布尔值
camel.component.dataset-test.exchange-formatter	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，我们默认为 DefaultExchangeFormatter。选项是 org.apache.camel.spi.ExchangeFormatter 类型。		ExchangeFormatter
camel.component.dataset-test.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.dataset-test.log	在模拟收到传入消息时打开日志记录。这只会记录传入消息的 INFO 级别一次。如需更详细的日志记录，然后将 org.apache.camel.component.mock.MockEndpoint 类的 logger 设置为 DEBUG 级别。	false	布尔值
camel.component.dataset.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.dataset.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.dataset.enabled	是否启用 dataset 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.dataset.exchange-formatter	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，我们默认为 DefaultExchangeFormatter。选项是 org.apache.camel.spi.ExchangeFormatter 类型。		ExchangeFormatter
camel.component.dataset.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.dataset.log	在模拟收到传入消息时打开日志记录。这只会记录传入消息的 INFO 级别一次。如需更详细的日志记录，然后将 org.apache.camel.component.mock.MockEndpoint 类的 logger 设置为 DEBUG 级别。	false	布尔值

第 30 章 直接

支持生成者和消费者

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。

注意

异步
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。

30.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 直接 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-direct-starter</artifactId>
</dependency>
```

30.2. URI 格式

```
direct:someName[?options]
```

其中 someName 可以是唯一标识端点的任何字符串

30.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

30.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

30.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

30.4. 组件选项

Direct 组件支持 5 个选项，如下所列。

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
block (producer)	如果向没有活跃消费者的直接端点发送消息，则我们可以告知生成者阻止，并等待消费者变为活动状态。	true	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
timeout (producer)	如果启用了块，要使用的超时值。	30000	long
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

30.5. 端点选项

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：

30.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
name (common)	直接端点 **必需** 名称。		字符串

30.5.2. 查询参数(8 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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
block (producer)	如果向没有活跃消费者的直接端点发送消息，则我们可以告知生成者阻止，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，生成者是否应该通过抛出异常失败。	true	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
timeout (producer)	如果启用了块，要使用的超时值。	30000	long
同步 (advanced)	是否强制同步处理。如果启用，则生成者线程将会被强制等待消息完成，然后同一线程将继续处理。如果禁用（默认），则生成者线程可以被释放，并可以在消息继续由其他线程处理时执行其他工作（重新活跃）。	false	布尔值

30.6. 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。

30.7. 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.direct.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.direct.block	如果向没有活跃消费者的直接端点发送消息，则我们可以告知生成者阻止，并等待消费者变为活动状态。	true	布尔值

Name	描述	默认值	类型
camel.component.direct.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.direct.enabled	是否启用直接组件的自动配置。这默认是启用的。		布尔值
camel.component.direct.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.direct.timeout	如果启用了块，要使用的超时值。	30000	Long

第 31 章 ELASTICSEARCH

Since Camel 3.18.3

仅支持生成者

ElasticSearch 组件允许您使用 Java API 客户端库与 ElasticSearch 8.x API 接口。

31.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 elasticsearch 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-elasticsearch-starter</artifactId>
</dependency>
```

31.2. URI 格式

```
elasticsearch://clusterName[?options]
```

31.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

31.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

31.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

31.4. 组件选项

Elasticsearch 组件支持 14 个选项，如下所列。

Name	描述	默认值	类型
connectionTimeout (producer)	连接超时前等待的时间。	30000	int
hostAddresses (producer)	使用 ip:port 格式的远程传输地址的逗号分隔列表。ip 和 port 选项必须留空，才能考虑 hostAddresses。		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
maxRetryTimeout (producer)	重试前的 ms 时间。	30000	int

Name	描述	默认值	类型
socketTimeout (producer)	套接字超时前等待的超时。	30000	int
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
client (advanced)	**Autowired** 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。		mingw
enableSniffer (advanced)	启用从正在运行的 Elasticsearch 集群自动发现节点。如果将此选项与 Spring Boot 一起使用，则它由 Spring Boot 配置管理（请参阅：在 Spring Boot 中禁用 Sniffer）。	false	布尔值
sniffAfterFailure Delay (advanced)	失败后调度的嗅探执行的延迟（以毫秒为单位）。	60000	int
snifferInterval (advanced)	以毫秒为单位连续普通嗅探执行之间的间隔。当 sniffOnFailure 被禁用或者连续嗅探执行之间没有故障时，将可以发现。	300000	int
certificatePath (security)	用于访问 Elasticsearch 的自签名证书的路径。		字符串
enableSSL (security)	启用 SSL。	false	布尔值
password (security)	用于验证的密码。		字符串
用户 （安全性）	基本身份验证用户。		字符串

31.5. 端点选项

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：

31.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
clusterName (producer)	**集群所需的** 名称。		字符串

31.5.2. 查询参数(19 参数)

Name	描述	默认值	类型
connectionTimeout (producer)	连接超时前等待的时间。	30000	int
disconnect (producer)	在它完成调用制作者后断开连接。	false	布尔值
from (producer)	启动响应的索引。		整数
hostAddresses (producer)	使用 ip:port 格式的远程传输地址的逗号分隔列表。		字符串
indexName (producer)	要操作的索引的名称。		字符串
maxRetryTimeout (producer)	重试前的 ms 时间。	30000	int

Name	描述	默认值	类型
operation (producer)	要执行的操作。 Enum 值： • 索引 • Update（更新） • 批量 • GetById • MultiGet • MultiSearch • 删除 • DeleteIndex • 搜索 • Exists • ping		ElasticsearchOperation
scrollKeepAliveMs (producer)	elasticsearch 将保持搜索上下文处于活动状态的时间 (ms)。	60000	int
size (producer)	响应的大小。		整数
socketTimeout (producer)	套接字超时前等待的超时。	30000	int
useScroll (producer)	启用滚动用法。	false	布尔值
waitForActiveShards (producer)	索引创建会等待分片的写入一致性数量。	1	int
lazyStartProducer (producer (advanced))	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
documentClass (advanced)	取消处理文档时要使用的类。	ObjectNode	类

Name	描述	默认值	类型
enableSniffer (advanced)	启用从正在运行的 Elasticsearch 集群自动发现节点。如果将此选项与 Spring Boot 一起使用，则它由 Spring Boot 配置管理（请参阅：在 Spring Boot 中禁用 Sniffer）。	false	布尔值
sniffAfterFailureDelay (advanced)	失败后调度的嗅探执行的延迟（以毫秒为单位）。	60000	int
snifferInterval (advanced)	以毫秒为单位连续普通嗅探执行之间的间隔。当 sniffOnFailure 被禁用或者连续嗅探执行之间没有故障时，将可以发现。	300000	int
certificatePath (security)	用于访问 Elasticsearch 的自签名证书的路径。		字符串
enableSSL (security)	启用 SSL。	false	布尔值

31.6. 消息标头

Elasticsearch 组件支持 9 个消息标头，如下所列：

Name	描述	默认值	类型
operation (producer) 常数：PARAM_OPERATION	要执行的操作。 Enum 值： • 索引 • Update（更新） • 批量 • GetById • MultiGet • MultiSearch • 删除 • DeleteIndex • 搜索 • Exists • ping		ElasticsearchOperation

Name	描述	默认值	类型
indexId (producer) 常数： PARAM_INDEX_ID	索引文档的 id。		字符串
indexName (producer) 常数： PARAM_INDEX_NAME	要操作的索引的名称。		字符串
documentClass (producer) 常量： PARAM_DOCUMENT_CLASS	unmarshall 文档类的全限定名称。	Object Node	类
waitForActiveShards (producer) 常量 ： PARAM_WAIT_FOR_ACTIVE_SHARDS	索引创建会等待分片的写入一致性数量。		整数
scrollKeepAliveMs (producer) 常量 ： PARAM_SCROLL_KEEP_ALIVE_MS	响应的起始索引。		整数
useScroll (producer) 常量 ： PARAM_SCROLL	设置为 true 以启用滚动用法。		布尔值
size (producer) 常数 ： PARAM_SIZE	响应的大小。		整数
from (producer) 常量 ： PARAM_FROM	响应的起始索引。		整数

31.7. 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，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[], Reader ,InputStream 或 IndexRequest .Builder 内容 **映射到** index	将内容添加到索引中，并返回正文中内容的 indexId。您可以通过使用键 "indexName" 设置消息标头来设置目标索引的名称。您可以通过使用键 "indexId" 设置消息标头来设置 indexId。
GetById	要检索的内容的字符串或 GetRequest. Builder 索引 id	检索与给定索引 id 对应的文档，并在正文中返回 GetResponse 对象。您可以通过使用键 "indexName" 设置消息标头来设置目标索引的名称。您可以通过使用键 "documentClass" 设置消息标头来设置文档类型。
删除	要删除的内容的字符串或 DeleteRequest.Builder 索引 id	删除指定的 indexName，并返回正文中的 Result 对象。您可以通过使用键 "indexName" 设置消息标头来设置目标索引的名称。

operation	消息正文	description
DeleteIndex	要删除的索引的字符串或 **DeleteIndexRequest.Builder** 索引名称	删除指定的 indexName，并在正文中返回状态代码。您可以通过使用键 "indexName" 设置消息标头来设置目标索引的名称。

operation	消息正文	description
批量	已接受的任何类型的**可迭代**或 **BulkRequest.Builder**（用于删除操作的 UpdateOperation.Builder）、用于更新操作的 UpdateOperation.Builder、CreateOperation.Builder 用于创建操作、bytes[]、InputStream、String、Reader、Map 或任何用于索引操作的文档类型）	添加/更新/删除内容从/到索引，并返回正文中的 List<BulkResponseItem> 对象，您可以通过使用键 "indexName" 设置消息标头来设置目标索引的名称。
搜索	**Map,String 或 SearchRequest.Builder**	使用查询字符串映射搜索内容。您可以通过使用键 "indexName" 设置消息标头来设置目标索引的名称。您可以通过使用键 "size" 设置消息标头来设置要返回的点击数。您可以通过使用键 "from" 设置消息标头来设置起始文档偏移。

operation	消息正文	description
MultiSearch	**MsearchRequest.Builder**	在一个中有多个搜索
MultiGet	**Iterable<String>** 或 **MgetRequest.Builder** 用于检索的文档的 id	在一个中有多个信息 您可以通过使用键 "indexName" 设置消息标头来设置目标索引的名称。
Exists	None	检查索引是否存在，并在正文中返回一个布尔值标志。 您必须通过使用键 "indexName" 设置消息标头来设置目标索引的名称。
Update（更新）	**byte[]**, **InputStream**, **String**, **Reader**, **Map**, Map 或任何要更新的文档类型内容	更新内容到索引，并在正文中返回内容的 indexId。您可以通过使用键 "indexName" 设置消息标头来设置目标索引的名称。您可以通过使用键 "indexId" 设置消息标头来设置 indexId。
ping	None	Ping Elasticsearch 集群，如果 ping 成功，则返回 true，否则为 false

31.8. 配置组件并启用基本身份验证

要使用 Elasticsearch 组件，必须为它配置最小配置。

```
ElasticsearchComponent elasticsearchComponent = new ElasticsearchComponent();
elasticsearchComponent.setHostAddresses("myelkhost:9200");
camelContext.addComponent("elasticsearch", elasticsearchComponent);
```

对于使用 elasticsearch 的基本身份验证或使用 elasticsearch 集群前面的反向 http 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);
elasticsearchComponent.setCertificatePath(certPath);

camelContext.addComponent("elasticsearch", elasticsearchComponent);
```

31.9. 索引示例

以下是一个简单的 INDEX 示例

```
from("direct:index")
  .to("elasticsearch://elasticsearch?operation=Index&indexName=twitter");
```

```
<route>
    <from uri="direct:index"/>
    <to uri="elasticsearch://elasticsearch?operation=Index&amp;indexName=twitter"/>
</route>
```

对于此操作，您需要指定 indexId 标头。

客户端只需要将包含 Map 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);
```

31.10. 搜索示例

在特定字段和值中搜索使用 Operation 'Search'。传递查询 JSON 字符串或映射

```
from("direct:search")
  .to("elasticsearch://elasticsearch?operation=Search&indexName=twitter");
```

```
<route>
    <from uri="direct:search"/>
    <to uri="elasticsearch://elasticsearch?operation=Search&amp;indexName=twitter"/>
</route>
```

```
String query = "{\"query\":{\"match\":{\"doc.content\":\"new release of ApacheCamel\"}}}";
HitsMetadata<?> response = template.requestBody("direct:search", query,
HitsMetadata.class);
```

使用 Map 搜索特定字段。

```
Map<String, Object> actualQuery = new HashMap<>();
actualQuery.put("doc.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);
HitsMetadata<?> response = template.requestBody("direct:search", query,
HitsMetadata.class);
```

使用 Elasticsearch 滚动 api 搜索以获取所有结果。

```
from("direct:search")
  .to("elasticsearch://elasticsearch?
operation=Search&indexName=twitter&useScroll=true&scrollKeepAliveMs=30000");
```

```
<route>
    <from uri="direct:search"/>
    <to uri="elasticsearch://elasticsearch?
operation=Search&amp;indexName=twitter&amp;useScroll=true&amp;scrollKeepAliveMs=30000"/>
</route>
```

```
String query = "{\"query\":{\"match\":{\"doc.content\":\"new release of ApacheCamel\"}}}";
try (ElasticsearchScrollRequestIterator response = template.requestBody("direct:search",
query, ElasticsearchScrollRequestIterator.class)) {
    // do something smart with results
}
```

也可以使用.

```
from("direct:search")
  .to("elasticsearch://elasticsearch?
operation=Search&indexName=twitter&useScroll=true&scrollKeepAliveMs=30000")
  .split()
  .body()
  .streaming()
  .to("mock:output")
  .end();
```

31.11. MULTISEARCH 示例

多搜索特定字段和值使用 Operation 'MultiSearch'。传递 MultiSearchRequest 实例

```
from("direct:multiSearch")
  .to("elasticsearch://elasticsearch?operation=MultiSearch");
```

```
<route>
    <from uri="direct:multiSearch"/>
    <to uri="elasticsearch://elasticsearch?operation=MultiSearch"/>
</route>
```

MultiSearch on specific fields）

```
MsearchRequest.Builder builder = new MsearchRequest.Builder().index("twitter").searches(
        new RequestItem.Builder().header(new MultisearchHeader.Builder().build())
                .body(new MultisearchBody.Builder().query(b -> b.matchAll(x -> x)).build()).build(),
        new RequestItem.Builder().header(new MultisearchHeader.Builder().build())
                .body(new MultisearchBody.Builder().query(b -> b.matchAll(x -> x)).build()).build());
List<MultiSearchResponseItem<?>> response = template.requestBody("direct:multiSearch",
builder, List.class);
```

31.12. 文档类型

对于所有搜索操作，可以指示要检索的文档类型，以获得已与预期类型相关的结果。

可使用标头 "documentClass" 或同一名称的 uri 参数来设置文档类型。

31.13. 在 SPRING BOOT 中使用 CAMEL ELASTICSEARCH

当您将 camel-elasticsearch-starter 与 Spring Boot v2 搭配使用时，您必须在您自己的 pom.xml 中声明以下依赖项：

```
<dependency>
  <groupId>jakarta.json</groupId>
  <artifactId>jakarta.json-api</artifactId>
  <version>2.0.2</version>
</dependency>
```

这是必要的，因为 Spring Boot v2 提供 jakarta.json-api:1.1.6，Elasticsearch 需要使用 json-api v2。

31.13.1. 使用 Spring Boot 提供的 RestClient

默认情况下，Spring Boot 将自动配置一个由 camel 使用的 Elasticsearch RestClient，它可以使用以下基本属性自定义客户端：

```
spring.elasticsearch.uris=myelkhost:9200
spring.elasticsearch.username=elkuser
spring.elasticsearch.password=secure!!
```

如需更多信息，请参阅 application-properties.data.spring.elasticsearch.connection-timeout。

31.13.2. 使用 Spring Boot 时禁用 Sniffer

当 Spring Boot 位于类路径上时，默认启用 Elasticsearch 的 Sniffer 客户端。在 Spring Boot 配置中可以禁用这个选项：

```
spring:
  autoconfigure:
    exclude:
org.springframework.boot.autoconfigure.elasticsearch.ElasticsearchRestClientAutoConfiguration
```

31.14. 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.elasticsearch.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.elasticsearch.certificate-path	用于访问 Elasticsearch 的自签名证书的路径。		字符串

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。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。		mingw
camel.component.elasticsearch.connection-timeout	连接超时前等待的时间。	30000	整数
camel.component.elasticsearch.enable-s-s-l	启用 SSL。	false	布尔值
camel.component.elasticsearch.enable-sniffer	启用从正在运行的 Elasticsearch 集群自动发现节点。如果将此选项与 Spring Boot 一起使用，则它由 Spring Boot 配置管理（请参阅：在 Spring Boot 中禁用 Sniffer）。	false	布尔值
camel.component.elasticsearch.enabled	是否启用 elasticsearch 组件的自动配置。这默认是启用的。		布尔值
camel.component.elasticsearch.host-addresses	使用 ip:port 格式的远程传输地址的逗号分隔列表。ip 和 port 选项必须留空，才能考虑 hostAddresses。		字符串
camel.component.elasticsearch.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.elasticsearch.max-retry-timeout	重试前的 ms 时间。	30000	整数
camel.component.elasticsearch.password	用于验证的密码。		字符串
camel.component.elasticsearch.sniff-after-failure-delay	失败后调度的嗅探执行的延迟（以毫秒为单位）。	60000	整数

Name	描述	默认值	类型
camel.component.elasticsearch.sniffer-interval	以毫秒为单位连续普通嗅探执行之间的间隔。当 sniffOnFailure 被禁用或者连续嗅探执行之间没有故障时，将可以发现。	300000	整数
camel.component.elasticsearch.socket-timeout	套接字超时前等待的超时。	30000	整数
camel.component.elasticsearch.user	基本身份验证用户。		字符串

第 32 章 EXCHANGEPROPERTY

ExchangeProperty 表达式语言允许您提取指定交换属性的值。

32.1. 依赖项

ExchangeProperty 语言是 camel-core 的一部分。

当在 Red Hat build of Camel Spring Boot 中使用 exchangeProperty 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-core-starter</artifactId>
</dependency>
```

32.2. EXCHANGE PROPERTY 选项

ExchangeProperty 语言支持 1 个选项，如下所列。

Name	默认值	Java 类型	描述
trim		布尔值	是否修剪值以移除前导和结尾的空格和换行符。

32.3. 示例

recipientList EIP 可以使用类似以下的 ExchangeProperty：

```
<route>
  <from uri="direct:a" />
  <recipientList>
    <exchangeProperty>myProperty</exchangeProperty>
  </recipientList>
</route>
```

在这种情况下，接收者列表包含在属性 'myProperty' 中。

以及 Java DSL 中的相同示例：

```
from("direct:a").recipientList(exchangeProperty("myProperty"));
```

32.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 147 选项，如下所列。

Name	描述	默认值	类型
camel.cloud.consul.service-discovery.acl-token	设置用于 Consul 的 ACL 令牌。		字符串
camel.cloud.consul.service-discovery.block-seconds	等待监视事件的秒数，默认为 10 秒。	10	整数
camel.cloud.consul.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.consul.service-discovery.connect-timeout-millis	OkHttpClient 的连接超时。		Long
camel.cloud.consul.service-discovery.datacenter	数据中心。		字符串
camel.cloud.consul.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.consul.service-discovery.password	设置用于基本身份验证的密码。		字符串

Name	描述	默认值	类型
camel.cloud.consul.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.consul.service-discovery.read-timeout-millis	OkHttpClient 的读取超时。		Long
camel.cloud.consul.service-discovery.url	Consul 代理 URL。		字符串
camel.cloud.consul.service-discovery.user-name	设置用于基本身份验证的用户名。		字符串
camel.cloud.consul.service-discovery.write-timeout-millis	OkHttpClient 的写入超时。		Long
camel.cloud.dns.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.dns.service-discovery.domain	域名；		字符串
camel.cloud.dns.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.dns.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.dns.service-discovery.proto	所需服务的传输协议。	_tcp	字符串

Name	描述	默认值	类型
camel.cloud.etcd.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.etcd.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.etcd.service-discovery.password	用于基本身份验证的密码。		字符串
camel.cloud.etcd.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.etcd.service-discovery.service-path	查找服务发现的路径。	/services/	字符串
camel.cloud.etcd.service-discovery.timeout	要设置操作可以采取的最长时间，请执行以下操作：		Long
camel.cloud.etcd.service-discovery.type	要设置发现类型，有效值为 on-demand 和 watch。	按需	字符串
camel.cloud.etcd.service-discovery.uris	客户端可以连接到的 URI。		字符串
camel.cloud.etcd.service-discovery.username	用于基本身份验证的用户名。		字符串
camel.cloud.kubernetes.service-discovery.api-version	使用客户端查找时设置 API 版本。		字符串

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.ca-cert-data	使用客户端查找时设置证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.ca-cert-file	在使用客户端查找时，设置从文件加载的证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-data	使用客户端查找时设置客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-file	在使用客户端查找时，设置从文件加载的客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-algo	设置客户端密钥存储算法，如使用客户端查找时 RSA。		字符串
camel.cloud.kubernetes.service-discovery.client-key-data	使用客户端查找时设置客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-file	在使用客户端查找时，设置从文件加载的客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-passphrase	使用客户端查找时设置客户端密钥存储密码短语。		字符串
camel.cloud.kubernetes.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.kubernetes.service-discovery.dns-domain	设置用于 DNS 查找的 DNS 域。		字符串

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.kubernetes.service-discovery.lookup	如何执行服务查找。可能的值有：client、dns、environment。在使用客户端时，客户端会查询 kubernetes master 来获取提供该服务的活跃 pod 列表，然后随机（或循环）选择一个 pod。当使用 dns 时，服务名称被解析为 name.namespace.svc.dnsDomain。当使用 dnssrv 时，服务名称使用 SRV 查询解析 ....svc... when using environment，环境变量用于查找服务。默认情况下使用环境。	环境	字符串
camel.cloud.kubernetes.service-discovery.master-url	在使用客户端查找时，将 URL 设置为 master。		字符串
camel.cloud.kubernetes.service-discovery.namespace	设置要使用的命名空间。默认情况下，将使用来自 ENV 变量 KUBERNETES_MASTER 的命名空间。		字符串
camel.cloud.kubernetes.service-discovery.oauth-token	在使用客户端查找时，为身份验证设置 OAUTH 令牌（而不是用户名/密码）。		字符串
camel.cloud.kubernetes.service-discovery.password	在使用客户端查找时设置用于身份验证的密码。		字符串
camel.cloud.kubernetes.service-discovery.port-name	设置用于 DNS/DNSSRV 查找的端口名称。		字符串
camel.cloud.kubernetes.service-discovery.port-protocol	设置用于 DNS/DNSSRV 查找的端口协议。		字符串
camel.cloud.kubernetes.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.trust-certs	设置在使用客户端查找时是否打开信任证书检查。	false	布尔值
camel.cloud.kubernetes.service-discovery.username	在使用客户端查找时设置用于身份验证的用户名。		字符串
camel.cloud.ribbon.load-balancer.client-name	设置 Ribbon 客户端名称。		字符串
camel.cloud.ribbon.load-balancer.configurations	定义其他配置定义。		Map
camel.cloud.ribbon.load-balancer.enabled	启用组件。	true	布尔值
camel.cloud.ribbon.load-balancer.namespace	命名空间。		字符串
camel.cloud.ribbon.load-balancer.password	密码。		字符串
camel.cloud.ribbon.load-balancer.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.ribbon.load-balancer.username	用户名。		字符串

Name	描述	默认值	类型
camel.hystrix.allow-maximum-size-to-diverge-from-core-size	允许配置使 maximumSize 生效。然后该值可以等于或大于 coreSize。	false	布尔值
camel.hystrix.circuit-breaker-enabled	是否使用 HystrixCircuitBreaker。如果为 false，则不会使用 断路器逻辑，并且所有允许的请求。这与 circuitBreakerForceClosed （）的影响类似，除非继续跟踪指标，知道它是否应该是 open/closed，此属性即使实例化一个断路器。	true	布尔值
camel.hystrix.circuit-breaker-error-threshold-percentage	错误百分比阈值（如 50 ）指向断路器将打开和拒绝请求。它将在 circuitBreakerSleepWindowInMilliseconds 中定义的持续时间保持出差；与 HystrixCommandMetrics.getHealthCounts （）进行比较的错误百分比。	50	整数
camel.hystrix.circuit-breaker-force-closed	如果为 true，HystrixCircuitBreaker#allowRequest （）将始终返回 true 以允许请求，无论 HystrixCommandMetrics.getHealthCounts （）的错误百分比如何。如果设为 true，则 circuitBreakerForceOpen （）属性具有优先权。	false	布尔值
camel.hystrix.circuit-breaker-force-open	如果为 true，HystrixCircuitBreaker.allowRequest （）将始终返回 false，从而导致电路变为开路（接受），并拒绝所有请求。此属性优先于 circuitBreakerForceClosed （）;。	false	布尔值
camel.hystrix.circuit-breaker-request-volume-threshold	metricsRollingStatisticalWindowInMilliseconds （）中的最少请求数必须存在于 HystrixCircuitBreaker 之前。如果此数字低于这个数字，无论错误百分比如何，电路都不会被出差。	20	整数
camel.hystrix.circuit-breaker-sleep-window-in-milliseconds	HystrixCircuitBreaker trips 之后的时间（以毫秒为单位），它应该在尝试请求前等待。	5000	整数
camel.hystrix.configurations	定义其他配置定义。		Map
camel.hystrix.core-pool-size	传递给 java.util.concurrent.ThreadPoolExecutor#setCorePoolSize (int)的核心 thread-pool 大小。	10	整数
camel.hystrix.enabled	启用组件。	true	布尔值

Name	描述	默认值	类型
camel.hystrix.execution-isolation-semaphore-max-concurrent-requests	允许 HystrixCommand.run （）的并发请求数。超过并发限制的请求将被拒绝。仅在执行IsolationStrategy == SEMAPHORE 时使用。	20	整数
camel.hystrix.execution-isolation-strategy	将通过什么隔离策略 HystrixCommand.run （）执行。如果 THREAD，它将在单独的线程上执行，并且受 thread-pool 中的线程数量限制的并发请求。如果 SEMAPHORE，它将在调用线程上执行，并且受 semaphore 数限制的并发请求。	线程	字符串
camel.hystrix.execution-isolation-thread-interrupt-on-timeout	当线程超时时，执行线程是否应该尝试中断（使用 future#cancel）。仅在执行IsolationStrategy （）== THREAD 时才适用。	true	布尔值
camel.hystrix.execution-timeout-enabled	此命令是否启用了超时机制。	true	布尔值
camel.hystrix.execution-timeout-in-milliseconds	以毫秒为单位，将命令超时和停止执行的时间（以毫秒为单位）。如果 executionIsolationThreadInterruptOnTimeout == true 且命令是线程隔离，则执行线程将中断。如果命令是 semaphore-isolated 和 HystrixObservableCommand，则该命令将被取消订阅。	1000	整数
camel.hystrix.fallback-enabled	出现故障时，是否应尝试 HystrixCommand.getFallback （）。	true	布尔值
camel.hystrix.fallback-isolation-semaphore-max-concurrent-requests	允许 HystrixCommand.getFallback （）的并发请求数。超过并发限制的请求将快速失败，且不会尝试检索回退。	10	整数
camel.hystrix.group-key	设置要使用的 group 键。默认值为 CamelHystrix。	Camel Hystrix	字符串
camel.hystrix.keep-alive-time	更长的时间（以分钟为单位）传递给 ThreadPoolExecutor#setKeepAliveTime (long,TimeUnit)。	1	整数

Name	描述	默认值	类型
camel.hystrix.max-queue-size	在 HystrixConcurrencyStrategy.getBlockingQueue (int)中传递给 BlockingQueue 的最大队列大小应该只影响 threadpool 的实例化 - 它不会立即更改队列大小。为此，请使用 queueSizeRejectionThreshold （）。	-1	整数
camel.hystrix.maximum-size	传递给 ThreadPoolExecutor#setMaximumPoolSize (int)的最大 thread-pool 大小。这是可在不开始拒绝 HystrixCommands 的情况下支持的最大并发数量。请注意，只有在您也设置了 allowMaximumSizeToDivergeFromCoreSize 时，此设置才会生效。	10	整数
camel.hystrix.metrics-health-snapshot-interval-in-milliseconds	在允许计算成功和错误百分比时等待的时间（以毫秒为单位），并影响 HystrixCircuitBreaker.isOpen （）状态。在高容量电路上，错误百分比的连续计算可能会成为 CPU 密集型，从而控制其计算的频率。	500	整数
camel.hystrix.metrics-rolling-percentile-bucket-size	滚动百分比的每个存储桶中存储的最大值数。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	10	整数
camel.hystrix.metrics-rolling-percentile-enabled	是否应该使用 HystrixRollingPercentile 内部 HystrixCommandMetrics 来捕获百分比的指标。	true	布尔值
camel.hystrix.metrics-rolling-percentile-window-buckets	滚动窗口的存储桶数量被分成。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	6	整数
camel.hystrix.metrics-rolling-percentile-window-in-milliseconds	以毫秒为单位的滚动窗口的持续时间。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	10000	整数
camel.hystrix.metrics-rolling-statistical-window-buckets	滚动统计窗口划分为的 bucket 数量。这在 HystrixCommandMetrics 中被传递给 HystrixRollingNumber。	10	整数

Name	描述	默认值	类型
camel.hystrix.metrics-rolling-statistical-window-in-milliseconds	此属性设置统计滚动窗口的持续时间，以毫秒为单位。这是为线程池保留指标的时间。窗口被分成 bucket，按这些增量回滚。	10000	整数
camel.hystrix.queue-size-rejection-threshold	队列大小拒绝阈值是 artificial max size，即使尚未达到 maxQueueSize，也会发生拒绝。这是因为 BlockingQueue 的 maxQueueSize 无法动态更改，我们希望动态更改影响拒绝的队列大小。在排队线程以进行执行时，HystrixCommand 会使用它。	5	整数
camel.hystrix.request-log-enabled	HystrixCommand 执行和事件是否应记录到 HystrixRequestLog。	true	布尔值
camel.hystrix.thread-pool-key	设置要使用的线程池密钥。默认情况下，将使用与 groupKey 配置相同的值。	Camel Hystrix	字符串
camel.hystrix.thread-pool-rolling-number-statistical-window-buckets	滚动统计窗口划分为的 bucket 数量。这会传递到每个 HystrixThreadPoolMetrics 实例内的 HystrixRollingNumber。	10	整数
camel.hystrix.thread-pool-rolling-number-statistical-window-in-milliseconds	统计滚动窗口的持续时间（以毫秒为单位）。这会传递到每个 HystrixThreadPoolMetrics 实例内的 HystrixRollingNumber。	10000	整数
camel.language.constant.enabled	是否启用恒定语言的自动配置。这默认是启用的。		布尔值
camel.language.constant.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.csimple.enabled	是否启用 csimple 语言的自动配置。这默认是启用的。		布尔值
camel.language.csimple.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.exchangeproperty.enabled	是否启用 exchangeProperty 语言的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.language.exchangeproperty.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.file.enabled	是否启用文件语言的自动配置。这默认是启用的。		布尔值
camel.language.file.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.header.enabled	是否启用标头语言的自动配置。这默认是启用的。		布尔值
camel.language.header.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.ref.enabled	是否启用 ref 语言的自动配置。这默认是启用的。		布尔值
camel.language.ref.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.simple.enabled	是否启用简单语言的自动配置。这默认是启用的。		布尔值
camel.language.simple.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.tokenize.enabled	是否启用令牌化语言的自动配置。这默认是启用的。		布尔值
camel.language.tokenize.group-delimiter	设置在分组时要使用的分隔符。如果没有设置，则令牌将用作分隔符。		字符串
camel.language.tokenize.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.resilience4j.automatic-transition-from-open-to-half-open-enabled	在通过 waitDurationInOpenState 后，启用从 OPEN 自动过渡到 HALF_OPEN 状态。	false	布尔值

Name	描述	默认值	类型
camel.resilience4j.circuit-breaker-ref	代表现有的 io.github.resilience4j.circuitbreaker.CircuitBreaker 实例从 registry 中查找和使用。使用此选项时，不使用任何其他断路器选项。		字符串
camel.resilience4j.config-ref	指的是现有的 io.github.resilience4j.circuitbreaker.CircuitBreakerConfig 实例，以便从 registry 中查找和使用。		字符串
camel.resilience4j.configurations	定义其他配置定义。		Map
camel.resilience4j.enabled	启用组件。	true	布尔值
camel.resilience4j.failure-rate-threshold	以百分比为单位配置故障率阈值。如果失败率相等或大于阈值，则 CircuitBreaker 过渡到 open，并启动短路调用。阈值必须大于 0，而不是大于 100。默认值为 50 百分比。		æµ®ç‚¹â€¼
camel.resilience4j.minimum-number-of-calls	在 CircuitBreaker 可以计算错误率之前，配置所需的最少调用数（每个滑动期限）。例如，如果 minimumNumberOfCalls 为 10，则必须至少记录 10 个调用，然后才能计算失败率。如果只记录了 9 个调用，则 CircuitBreaker 不会过渡到 open，即使所有 9 调用都失败。默认 minimumNumberOfCalls 为 100。	100	整数
camel.resilience4j.permitted-number-of-calls-in-half-open-state	配置 CircuitBreaker 为一半打开时允许的调用数量。大小必须大于 0。默认大小为 10。	10	整数
camel.resilience4j.sliding-window-size	配置滑动窗口的大小，该窗口用于在 CircuitBreaker 关闭时记录调用的结果。slidingWindowSize 配置滑动窗口的大小。滑动窗口可以是基于计数或基于时间的窗口。如果 slidingWindowType 是 COUNT_BASED，则最后的 slidingWindowSize 调用会被记录并聚合。如果 slidingWindowType 是 TIME_BASED，则最后 slidingWindowSize 秒的调用会被记录并聚合。slidingWindowSize 必须大于 0。minimumNumberOfCalls 必须大于 0。如果 slidingWindowType 是 COUNT_BASED，则 minimumNumberOfCalls 不能大于 slidingWindowSize。如果 slidingWindowType 是 TIME_BASED，您可以选择任何您需要的。默认 slidingWindowSize 为 100。	100	整数

Name	描述	默认值	类型
camel.resilience4j.sliding-window-type	配置滑动窗口的类型，用于记录 CircuitBreaker 关闭时调用的结果。滑动窗口可以是基于计数或基于时间的窗口。如果 slidingWindowType 是 COUNT_BASED，则最后的 slidingWindowSize 调用会被记录并聚合。如果 slidingWindowType 是 TIME_BASED，则最后 slidingWindowSize 秒的调用会被记录并聚合。默认 slidingWindowType 是 COUNT_BASED。	COUNT_BASED	字符串
camel.resilience4j.slow-call-duration-threshold	配置上面的持续时间阈值（秒），调用被视为缓慢，并增加较慢的调用百分比。默认值为 60 秒。	60	整数
camel.resilience4j.slow-call-rate-threshold	以百分比为单位配置阈值。当调用持续时间大于 slowCallDurationThreshold Duration 时，CircuitBreaker 会将调用视为较慢。当较慢的调用百分比相等或大于阈值时，CircuitBreaker 过渡到 open，并启动短路调用。阈值必须大于 0，而不是大于 100。默认值为 100 百分比，这意味着所有记录的调用都必须比 slowCallDurationThreshold 慢。		æµ®ç‚¹â€¼
camel.resilience4j.wait-duration-in-open-state	配置等待持续时间（以秒为单位），指定 CircuitBreaker 应该保持打开的时间，然后再切换到半次。默认值为 60 秒。	60	整数
camel.resilience4j.writable-stack-trace-enabled	启用可写入堆栈跟踪。当设置为 false 时，Exception.getStackTrace 返回一个零长度数组。当断路器处于开路状态时，这可用于减少日志垃圾邮件，因为存在例外的原因（断路器是短路调用）。	true	布尔值
camel.rest.api-component	用作 REST API 的 Camel 组件名称（如 swagger）如果没有明确配置 API 组件，则 Camel 会查找负责服务并生成 REST API 文档的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestApiProcessorFactory。如果找到其中任何一个，则使用它。		字符串
camel.rest.api-context-path	设置领导的 API 上下文路径将使用的 REST API 服务。这在使用使用 context-path 部署 Web 应用程序的组件（如 camel-servlet ）时使用它。		字符串
camel.rest.api-context-route-id	设置用于服务 REST API 的路由的路由 ID。默认情况下，路由将使用自动分配的路由 ID。		字符串
camel.rest.api-host	要将特定主机名用于 API 文档（如 swagger），这可用于用这个配置的主机名覆盖生成的主机。		字符串

Name	描述	默认值	类型
camel.rest.api-property	允许为 api 文档配置任意数量的附加属性(swagger)。例如，将属性 api.title 设置为我的冷却。		Map
camel.rest.api-vendor-extension	是否在 Rest API 中启用供应商扩展。如果启用，Camel 将包含额外信息作为厂商扩展名（例如，以 x- 开头的键），如路由 ID、类名称等。在导入 API 文档时，并非所有第三方 API 网关和工具都支持 vendor-extensions。	false	布尔值
camel.rest.binding-mode	设置要使用的绑定模式。默认值为 off。		RestBindingMode
camel.rest.client-request-validation	是否启用客户端请求验证，以检查客户端的 Content-Type 和 Accept 标头是否受到其 consume/produces 设置的 Rest-DSL 配置的支持。这可以打开，以启用此检查。如果验证错误，则返回 HTTP Status code 415 或 406。默认值为 false。	false	布尔值
camel.rest.component	用于 REST 传输(consumer)的 Camel Rest 组件，如 netty-http, jetty, servlet, undertow。如果没有明确配置组件，则 Camel 会查找是否有与 Rest DSL 集成的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestConsumerFactory。如果找到其中任何一个，则使用它。		字符串
camel.rest.component-property	允许为正在使用的其他组件配置任意数量的附加属性。		Map
camel.rest.consumer-property	允许为使用中的其他使用者配置任意数量的附加属性。		Map
camel.rest.context-path	设置 REST 服务将使用的前导上下文路径。这在使用使用 context-path 部署 Web 应用程序的组件（如 camel-servlet ）时使用它。或者对于包含 HTTP 服务器的 camel-jetty 或 camel-netty-http 等组件。		字符串
camel.rest.cors-headers	允许配置自定义 CORS 标头。		Map
camel.rest.data-format-property	允许为使用的数据格式配置多个额外属性。例如，将属性 prettyPrint 设置为 true，以便以用户友善模式输出 json。属性可以加上前缀来表示选项仅适用于 JSON 或 XML，以及 IN 或 OUT。前缀为： json.in. json.out. xml.in. xml.out。例如，值为 xml.out.mustBeJAXBElement 的键仅用于传出的 XML 数据格式。没有前缀的密钥是所有情况的通用密钥。		Map

Name	描述	默认值	类型
camel.rest.enable-cors	是否在 HTTP 响应中启用 CORS 标头。默认值为 false。	false	布尔值
camel.rest.endpoint-property	允许为使用中的其他端点配置多个额外的属性。		Map
camel.rest.host	用于公开 REST 服务的主机名。		字符串
camel.rest.host-name-resolver	如果没有明确配置的主机名，这个 resolver 会用于计算 REST 服务将要使用的主机名。		RestHostNameResolver
camel.rest.json-data-format	要使用的特定 json 数据格式的名称。默认将使用 json-jackson。重要：此选项仅用于设置数据格式的自定义名称，而不是引用现有数据格式实例。		字符串
camel.rest.port	用于公开 REST 服务的主机名。请注意，如果您使用 servlet 组件，则此处配置的端口号不适用，因为使用中的端口号是 servlet 组件使用的实际端口号。例如，如果使用 Apache Tomcat，它的 tomcat http 端口，如果使用 Apache Karaf，它的在 Karaf 中的 HTTP 服务，它默认使用端口 8181。虽然在这些情况下，这里设置端口号，但允许工具和 JMX 知道端口号，因此建议将端口号设置为 servlet 引擎使用的数字。		字符串
camel.rest.producer-api-doc	设置 api 文档的位置，REST 生成者将根据这个文档来验证 REST uri 和查询参数是否有效。这需要将 camel-swagger-java 添加到 classpath 中，任何缺失的配置都会导致 Camel 在启动时失败并报告错误。默认情况下从 classpath 加载的 api 文档的位置，但您可以使用 file: 或 http: 引用从文件或 http url 加载的资源。		字符串
camel.rest.producer-component	设置要用作 REST 生成者的 Camel 组件的名称。		字符串
camel.rest.scheme	用于公开 REST 服务的方案。通常支持 http 或 https。默认值为 http。		字符串
camel.rest.skip-binding-on-error-code	如果存在自定义 HTTP 错误代码标头，是否跳过输出绑定。这允许构建没有绑定到 json / xml 等自定义错误消息，否则成功信息会这样做。	false	布尔值
camel.rest.use-x-forward-headers	是否将 X-Forward 标头用于主机和相关设置。默认值为 true。	true	布尔值
camel.rest.xml-data-format	要使用的特定 XML 数据格式的名称。默认情况下将使用 jaxb。重要：此选项仅用于设置数据格式的自定义名称，而不是引用现有数据格式实例。		字符串

Name	描述	默认值	类型
camel.rest.api-context-id-pattern	**弃用** 设置 CamelContext id 特征，以只允许 CamelContext 中名称与特征匹配的其他服务的 Rest API。特征 name 指的是 CamelContext 名称，仅匹配当前的 CamelContext。对于任何其他值，特征使用来自 PatternHelper#matchPattern (String,String)的规则。		字符串
camel.rest.api-context-listing	**弃用** 设置是否启用了 JVM 中带有 REST 服务的所有可用 CamelContext 的列表。如果启用，它将允许发现这些上下文，如果为 false，则只使用当前的 CamelContext。	false	布尔值

第 33 章 FHIR

支持生成者和消费者

FHIR 组件与 HAPI-FHIR 库集成，它是 Java 中 FHIR (Fast Healthcare Interoperability Resources) 规范的开源实现。

33.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 fhir 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-fhir-starter</artifactId>
</dependency>
```

33.2. 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
- Transactions
- update
- validate

33.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

33.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

33.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

33.4. 组件选项

FHIR 组件支持 27 个选项，如下所列。

Name	描述	默认值	类型
编码 (common)	用于所有请求的编码。 Enum 值： • JSON • XML		字符串

Name	描述	默认值	类型
fhirVersion (common)	要使用的 FHIR 版本。 Enum 值： • DSTU2 • DSTU2_HL7ORG • DSTU2_1 • DSTU3 • R4 • R5	R4	字符串
log (common)	将记录每个请求和响应。	false	布尔值
prettyPrint (common)	用户用户打印所有请求。	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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
client (advanced)	使用自定义客户端。		IGenericClient
clientFactory (advanced)	使用自定义客户端工厂。		IRestfulClientFactory

Name	描述	默认值	类型
compress (advanced)	将传出(POST/PUT)内容压缩为 GZIP 格式。	false	布尔值
configuration (advanced)	使用共享配置。		FhirConfiguration
connectionTimeout (advanced)	尝试和建立初始 TCP 连接（毫秒）。	10000	整数
deferModelScanning (advanced)	当设定此选项时，不会为子对象扫描模型类，直到实际访问给定类型的子列表为止。	false	布尔值
fhirContext (advanced)	FhirContext 是要创建的昂贵的对象。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（单位为 ms）。	10000	整数
Summary (advanced)	请求服务器使用 _summary 参数修改响应。 Enum 值： • 数量 • TEXT • DATA • TRUE • FALSE		字符串
validationMode (advanced)	当 Camel 验证 FHIR 服务器的合规声明时。 Enum 值： • NEVER • ONCE	ONCE	字符串
proxyHost (proxy)	代理主机。		字符串

Name	描述	默认值	类型
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
password (security)	用于基本身份验证的用户名。		字符串
用户名 （安全性）	用于基本身份验证的用户名。		字符串

33.5. 端点选项

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：

33.5.1. 路径参数(2 参数)

Name	描述	默认值	类型

Name	描述	默认值	类型
apiName (common)	**需要** 执行什么操作。 Enum 值： • 功能 • CREATE • DELETE • HISTORY • LOAD_PAGE • META • 操作 • PATCH • READ • 搜索 • TRANSACTIONS • UPDATE （更新） • VALIDATE		FhirApiName
methodName (common)	**必需的**所选操作需要哪些子操作。		字符串

33.5.2. 查询参数(44 参数)

Name	描述	默认值	类型
编码 (common)	用于所有请求的编码。 Enum 值： • JSON • XML		字符串

Name	描述	默认值	类型
fhirVersion (common)	要使用的 FHIR 版本。 Enum 值： • DSTU2 • DSTU2_HL7ORG • DSTU2_1 • DSTU3 • R4 • R5	R4	字符串
inBody (common)	设置要在交换中传递的参数名称。		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
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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
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
client (advanced)	使用自定义客户端。		IGenericClient
clientFactory (advanced)	使用自定义客户端工厂。		IRestfulClientFactory
compress (advanced)	将传出(POST/PUT)内容压缩为 GZIP 格式。	false	布尔值
connectionTimeout (advanced)	尝试和建立初始 TCP 连接（毫秒）。	10000	整数
deferModelScanning (advanced)	当设定此选项时，不会为子对象扫描模型类，直到实际访问给定类型的子列表为止。	false	布尔值
fhirContext (advanced)	FhirContext 是要创建的昂贵的对象。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（单位为 ms）。	10000	整数

Name	描述	默认值	类型
Summary (advanced)	请求服务器使用 _summary 参数修改响应。 Enum 值： • 数量 • TEXT • DATA • TRUE • FALSE		字符串
validationMode (advanced)	当 Camel 验证 FHIR 服务器的合规声明时。 Enum 值： • NEVER • ONCE	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
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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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
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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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)	OAuth 访问令牌。		字符串
password (security)	用于基本身份验证的用户名。		字符串
用户名 （安全性）	用于基本身份验证的用户名。		字符串

33.6. API 参数(13 API)

@FHIR 端点是基于 API 的组件，具有基于哪个 API 名称和 API 方法的额外参数。API 名称和 API 方法位于端点 URI 中，作为 apiName/methodName 路径参数：

```
fhir:apiName/methodName
```

下表列出 13 个 API 名称：

API 名称	类型	描述
功能	两者都	API 用于 Fetch 服务器的能力语句
create	两者都	创建操作的 API，用于在服务器上创建新资源实例
delete	两者都	删除操作的 API，它在服务器资源上执行逻辑删除

API 名称	类型	描述
history	两者都	历史记录方法的 API
load-page	两者都	使用 atom 捆绑包中的 link type=next 标签中指定的链接，从分页集中加载之前/下一步资源捆绑包的 API
meta	两者都	meta 操作的 API，可用于获取、添加和删除资源或跨服务器中的标签和其他 Meta 元素
operation	两者都	扩展 FHIR 操作的 API
patch	两者都	补丁操作的 API，它在服务器资源上执行逻辑补丁
读取	两者都	用于读取操作的 API 方法
search	两者都	API 搜索与给定条件匹配的资源
Transactions	两者都	将事务（资源集合）发送到服务器以作为一个单元执行的 API
update	两者都	更新操作的 API，它在服务器资源上执行逻辑删除
validate	两者都	用于验证资源的 API

每个 API 记录在以下部分中。

33.6.1. API: capabilities

支持生成者和消费者

capabilities API 在语法中定义，如下所示：

```
fhir:capabilities/methodName?[parameters]
```

此方法列在下表中，后跟每种方法的详细语法。(API 方法可以有一个简写别名名称，可用于语法而不是名称)

æ–¹æ³•	描述
ofType	使用给定的模型类型检索合规语句

33.6.1.1. Type 方法

signatures:

- **org.hl7.fhir.instance.model.api.IBaseConformance ofType (Class<org.hl7.fhir.instance.model.api.IBaseConformance> type, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/ofType API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
type	模型类型	类

除了上面的参数外，fhir API 也可以使用任何 Query 参数。

任何参数可以在端点 URI 中提供，也可以在消息标头中动态提供。消息标头名称必须是 CamelFhir.parameter 的格式。inBody 参数覆盖消息标头，即 endpoint 参数 inBody=myParameterNameHere 覆盖 CamelFhir.myParameterNameHere 标头。

33.6.2. API: create

支持生成者和消费者

create API 以语法定义，如下所示：

```
fhir:create/methodName?[parameters]
```

下表中列出了 1 方法，后跟每种方法的详细语法。(API 方法可以有一个简写别名名称，可用于语法而不是名称)

æ–¹æ³•	描述
resource	在服务器上创建一个 IBaseResource

33.6.2.1. 方法资源

signatures:

- **ca.uhn.fhir.rest.api.MethodOutcome resource (String resourceAsString, String url, ca.uhn.fhir.rest.api.PreferReturnEnum preferReturn, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

- **ca.uhn.fhir.rest.api.MethodOutcome resource (org.hl7.fhir.instance.model.api.IBaseResource resource, String url, ca.uhn.fhir.rest.api.PreferReturnEnum preferReturn, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/resource API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
preferReturn	在请求中添加 Prefer 标头，该请求服务器包含或阻止资源正文作为结果的一部分。如果服务器返回资源，它将通过 MethodOutcome#getResource （）解析一个可由客户端访问的资源，则可能是 null	PreferReturnEnum
resource	要创建的资源	IBaseResource
resourceAsString	要创建的资源	字符串
url	要使用的搜索 URL。此 URL 的格式应该是 ResourceTypeParameters，例如：Patientname=Smith&identifier=13.2.4.11.4%7C847366，可以是 null	字符串

除了上面的参数外，fhir API 也可以使用任何 Query 参数。

任何参数可以在端点 URI 中提供，也可以在消息标头中动态提供。消息标头名称必须是 CamelFhir.parameter 的格式。inBody 参数覆盖消息标头，即 endpoint 参数 inBody=myParameterNameHere 覆盖 CamelFhir.myParameterNameHere 标头。

33.6.3. API: delete

支持生成者和消费者

delete API 以语法定义，如下所示：

```
fhir:delete/methodName?[parameters]
```

下表中列出了 3 种方法，后跟每种方法的详细语法。(API 方法可以有一个简写别名名称，可用于语法而不是名称)

æ–¹æ³•	描述
resource	删除给定资源
resourceById	按资源类型 e 删除资源
resourceConditionalByUrl	指定删除应作为一个条件删除对给定的搜索 URL 执行

33.6.3.1. 方法资源

signatures:

- **org.hl7.fhir.instance.model.api.IBaseOperationOutcome resource (org.hl7.fhir.instance.model.api.IBaseResource resource, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/resource API 方法在下表中列出的参数：

参数	描述	类型

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
resource	要删除的 IBaseResource	IBaseResource

33.6.3.2. 方法 resourceById

signatures:

- **org.hl7.fhir.instance.model.api.IBaseOperationOutcome resourceById (String type, String stringId, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**
- **org.hl7.fhir.instance.model.api.IBaseOperationOutcome resourceById (org.hl7.fhir.instance.model.api.IIdType id, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/resourceById API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
id	引用资源的 IIdType	IIdType
stringId	它是 id	字符串
type	资源类型，如 Patient	字符串

33.6.3.3. method resourceConditionalByUrl

signatures:

- **org.hl7.fhir.instance.model.api.IBaseOperationOutcome resourceConditionalByUrl (String url, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object>**

extraParameters);

fhir/resourceConditionalByUrl API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
url	要使用的搜索 URL。此 URL 的格式应该是 ResourceTypeParameters，例如：Patientname=Smith&identifier=13.2.4.11.4%7C847366	字符串

除了上面的参数外，fhir API 也可以使用任何 Query 参数。

任何参数可以在端点 URI 中提供，也可以在消息标头中动态提供。消息标头名称必须是 CamelFhir.parameter 的格式。inBody 参数覆盖消息标头，即 endpoint 参数 inBody=myParameterNameHere 覆盖 CamelFhir.myParameterNameHere 标头。

33.6.4. API: history

支持生成者和消费者

历史 API 以语法定义，如下所示：

```
fhir:history/methodName?[parameters]
```

下表中列出了 3 种方法，后跟每种方法的详细语法。(API 方法可以有一个简写别名名称，可用于语法而不是名称)

æ–¹æ³•	描述
onInstance	在服务器上对特定资源（按 ID 和类型）在所有版本间执行操作
onServer	对服务器上所有类型的所有资源版本执行操作
onType	对服务器上给定类型的所有资源的所有版本执行操作

33.6.4.1. method onInstance

signatures:

- **org.hl7.fhir.instance.model.api.IBaseBundle onInstance（org.hl7.fhir.instance.model.api.IIdType id, Class<org.hl7.fhir.instance.model.api.model.api.IBaseBundle> returnType, Integer count, java.util.Date cutoff, org.hl7.fhir.instance.model.api.IPrimitType<javac.Date cutoff, org.hl7.fhir java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**

fhir/onInstance API 方法在下表中列出的参数：

参数	描述	类型
æ·°é‡	请求服务器只返回资源Count 数量（可能为 NULL）	整数
cutoff	请求服务器仅返回给定时间（包含）后或之后创建的资源版本可能为 NULL	Date
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
iCutoff	请求服务器仅返回给定时间（包含）后或之后创建的资源版本可能为 NULL	IPrimitiveType
id	必须使用资源类型和资源 ID 填充的 IIdType	IIdType
returnType	请求该方法返回 Bundle 资源（如 ca.uhn.fhir.model.dstu2.resource.Bundle）。如果您要访问 DSTU2 服务器，请使用此方法。	类

33.6.4.2. Server 上的方法

signatures:

- **org.hl7.fhir.instance.model.api.IBaseBundle onServer（Class<org.hl7.fhir.instance.model.api.IBaseBundle> returnType, Integer count, java.util.Date cutoff, org.hl7.fhir.instance.model.api.IPrimitiveType<java.util.Date> iCutoff, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**

fhir/onServer API 方法在下表中列出的参数：

参数	描述	类型
æ•°é‡	请求服务器只返回资源Count 数量（可能为 NULL）	整数
cutoff	请求服务器仅返回给定时间（包含）后或之后创建的资源版本可能为 NULL	Date
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
iCutoff	请求服务器仅返回给定时间（包含）后或之后创建的资源版本可能为 NULL	IPrimitiveType
returnType	请求该方法返回 Bundle 资源（如 ca.uhn.fhir.model.dstu2.resource.Bundle）。如果您要访问 DSTU2 服务器，请使用此方法。	类

33.6.4.3. 方法 onType

signatures:

- **org.hl7.fhir.instance.model.api.IBaseBundle onType（Class<org.hl7.fhir.instance.model.api.IBaseResource> resourceType, Class<org.hl7.fhir.instance.model.api.IBaseBundle> returnType, Integer count, java.util.Date cutoff, java.util.Date cutoff, org.hl7.fhir.instance.model.api.IPrimitiveType<java.util.Date> iCutoff, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**

fhir/onType API 方法在下表中列出的参数：

参数	描述	类型
æ•°é‡	请求服务器只返回资源Count 数量（可能为 NULL）	整数
cutoff	请求服务器仅返回给定时间（包含）后或之后创建的资源版本可能为 NULL	Date
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map

参数	描述	类型
iCutoff	请求服务器仅返回给定时间（包含）后或之后创建的资源版本可能为 NULL	IPrimitiveType
resourceType	要搜索的资源类型	类
returnType	请求该方法返回 Bundle 资源（如 ca.uhn.fhir.model.dstu2.resource.Bundle）。如果您要访问 DSTU2 服务器，请使用此方法。	类

除了上面的参数外，fhir API 也可以使用任何 Query 参数。

任何参数可以在端点 URI 中提供，也可以在消息标头中动态提供。消息标头名称必须是 CamelFhir.parameter 的格式。inBody 参数覆盖消息标头，即 endpoint 参数 inBody=myParameterNameHere 覆盖 CamelFhir.myParameterNameHere 标头。

33.6.5. api: load-page

支持生成者和消费者

载入页面 API 以语法中定义，如下所示：

```
fhir:load-page/methodName?[parameters]
```

下表中列出了 3 种方法，后跟每种方法的详细语法。(API 方法可以有一个简写别名名称，可用于语法而不是名称)

æ–¹æ³•	描述
byUrl	使用给定 URL 和捆绑包类型加载结果页面，并返回 DSTU1 Atom 捆绑包
next	使用捆绑包中下一个关系的链接来加载下一页的结果
previous	使用捆绑包中关系 prev 的链接加载前面的结果页面

33.6.5.1. method byUrl

signatures:

- **org.hl7.fhir.instance.model.api.IBaseBundle byUrl (String url, Class<org.hl7.fhir.instance.model.api.IBaseBundle> returnType, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/byUrl API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
returnType	返回类型	类
url	搜索 url	字符串

33.6.5.2. 下一步方法

signatures:

- **org.hl7.fhir.instance.model.api.IBaseBundle next (org.hl7.fhir.instance.model.api.IBaseBundle bundle, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/next API 方法在下表中列出的参数：

参数	描述	类型
bundle	IBaseBundle	IBaseBundle
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map

33.6.5.3. 之前的方法

signatures:

- **org.hl7.fhir.instance.model.api.IBaseBundle previous (org.hl7.fhir.instance.model.api.IBaseBundle bundle, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/previous API 方法在下表中列出的参数：

参数	描述	类型
bundle	IBaseBundle	IBaseBundle
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map

除了上面的参数外，fhir API 也可以使用任何 Query 参数。

任何参数可以在端点 URI 中提供，也可以在消息标头中动态提供。消息标头名称必须是 CamelFhir.parameter 的格式。inBody 参数覆盖消息标头，即 endpoint 参数 inBody=myParameterNameHere 覆盖 CamelFhir.myParameterNameHere 标头。

33.6.6. api: meta

支持生成者和消费者

meta API 以语法定义，如下所示：

```
fhir:meta/methodName?[parameters]
```

下表中列出了 5 方法，后跟每种方法的详细语法。(API 方法可以有一个简写别名名称，可用于语法而不是名称)

æ–¹æ³•	描述
add	将给定元数据中的元素添加到已存在的集合中（请勿删除任何）

æ–¹æ³•	描述
delete	从给定的 ID 中删除给定元数据中的元素
getFromResource	从特定资源获取当前元数据
getFromServer	从整个服务器获取当前元数据
getFromType	从特定类型的中获取当前元数据

33.6.6.1. 方法添加

signatures:

- **org.hl7.fhir.instance.model.api.IBaseMetaType add (org.hl7.fhir.instance.model.api.IBaseMetaType meta, org.hl7.fhir.instance.model.api.IIdType id, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/add API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
id	id	IIdType
meta	IBaseMetaType 类	IBaseMetaType

33.6.6.2. 方法删除

signatures:

- **org.hl7.fhir.instance.model.api.IBaseMetaType delete (org.hl7.fhir.instance.model.api.IBaseMetaType meta, org.hl7.fhir.instance.model.api.IIdType id, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/delete API 方法包含下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
id	id	IIdType
meta	IBaseMetaType 类	IBaseMetaType

33.6.6.3. 方法 getFromResource

signatures:

- **org.hl7.fhir.instance.model.api.IBaseMetaType getFromResource（Class<org.hl7.fhir.instance.model.api.IBaseMetaType> metaType, org.hl7.fhir.instance.model.api.IIdType id, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters;**

fhir/getFromResource API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
id	id	IIdType
metaType	IBaseMetaType 类	类

33.6.6.4. 方法 getFromServer

signatures:

- **org.hl7.fhir.instance.model.api.IBaseMetaType getFromServer (Class<org.hl7.fhir.instance.model.api.IBaseMetaType> metaType,**

java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);

fhir/getFromServer API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
metaType	给定 FHIR 模型版本的 meta 数据类型（应该为 MetaDt.class 或 MetaType.class）	类

33.6.6.5. 方法 getFromType

signatures:

- **org.hl7.fhir.instance.model.api.IBaseMetaType getFromType (Class<org.hl7.fhir.instance.model.api.IBaseMetaType> metaType, String resourceType, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/getFromType API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
metaType	IBaseMetaType 类	类
resourceType	资源类型，如 Patient	字符串

除了上面的参数外，fhir API 也可以使用任何 Query 参数。

任何参数可以在端点 URI 中提供，也可以在消息标头中动态提供。消息标头名称必须是 CamelFhir.parameter 的格式。inBody 参数覆盖消息标头，即 endpoint 参数 inBody=myParameterNameHere 覆盖 CamelFhir.myParameterNameHere 标头。

33.6.7. API: 操作

支持生成者和消费者

操作 API 以语法定义，如下所示：

```
fhir:operation/methodName?[parameters]
```

下表中列出了 5 方法，后跟每种方法的详细语法。(API 方法可以有一个简写别名名称，可用于语法而不是名称)

æ–¹æ³•	描述
onInstance	在服务器上对特定资源（按 ID 和类型）在所有版本间执行操作
onInstanceVersion	此操作在资源的特定版本上运行
onServer	对服务器上所有类型的所有资源版本执行操作
onType	对服务器上给定类型的所有资源的所有版本执行操作
processMessage	此操作称为由 FHIR 规范定义的 $process-message

33.6.7.1. method onInstance

signatures:

- **org.hl7.fhir.instance.model.api.IBaseResource onInstance（org.hl7.fhir.instance.model.api.IIdType id, String name, org.hl7.fhir.instance.model.api.model.api.IBaseParameters parameters, Class<org.hl7.fhir.instance.model.api.IBaseParameters> outputParameterType, 布尔值 useHttpGet, Class<org.hl7.fhir.instance.model.api.IBaseResource> returnType, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**

fhir/onInstance API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
id	资源（版本将被剥离）	IIdType
name	操作名称	字符串
outputParameterType	用于输出参数的类型（这应设置为来自您使用的 FHIR 结构版本的 Parameters.class drawn），可以是 NULL。	类
parameters	用作输入的参数。如果操作不需要任何输入参数，则也可能为 null。	IBaseParameters
returnType	如果此操作返回单个资源正文作为其返回类型而不是 Parameters 资源，请使用此方法指定该资源类型。这对返回捆绑包而不是 Parameters 资源的某些操作（如 Patient/NN/$everything）非常有用。	类
useHttpGet	使用 HTTP GET 动词	布尔值

33.6.7.2. method onInstanceVersion

signatures:

- **org.hl7.fhir.instance.model.api.IBaseResource onInstanceVersion（org.hl7.fhir.instance.model.api.IIdType id, String name, org.hl7.fhir.instance.model.api.model.api.IBaseParameters parameters, Class<org.hl7.fhir.instance.model.api.IBaseParameters> outputParameterType, 布尔值 useHttpGet, Class<org.hl7.fhir.instance.model.api.IBaseResource> returnType, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**

fhir/onInstanceVersion API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
id	资源版本	IIdType

参数	描述	类型
name	操作名称	字符串
outputParameterType	用于输出参数的类型（这应设置为来自您使用的 FHIR 结构版本的 Parameters.class drawn），可以是 NULL。	类
parameters	用作输入的参数。如果操作不需要任何输入参数，则也可能为 null。	IBaseParameters
returnType	如果此操作返回单个资源正文作为其返回类型而不是 Parameters 资源，请使用此方法指定该资源类型。这对返回捆绑包而不是 Parameters 资源的某些操作（如 Patient/NN/$everything）非常有用。	类
useHttpGet	使用 HTTP GET 动词	布尔值

33.6.7.3. Server 上的方法

signatures:

- **org.hl7.fhir.instance.model.api.IBaseResource onServer （String name, org.hl7.fhir.instance.model.api.IBaseParameters 参数, Class<org.hl7.fhir.instance.model.api.IBaseParameters> outputParameterType, boolean useHttpGet, Class<org.hl7.fhir.instance.model.api.IBaseResource>, returnTypeType java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**

fhir/onServer API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
name	操作名称	字符串
outputParameterType	用于输出参数的类型（这应设置为来自您使用的 FHIR 结构版本的 Parameters.class drawn），可以是 NULL。	类
parameters	用作输入的参数。如果操作不需要任何输入参数，则也可能为 null。	IBaseParameters

参数	描述	类型
returnType	如果此操作返回单个资源正文作为其返回类型而不是 Parameters 资源，请使用此方法指定该资源类型。这对返回捆绑包而不是 Parameters 资源的某些操作（如 Patient/NN/$everything）非常有用。	类
useHttpGet	使用 HTTP GET 动词	布尔值

33.6.7.4. 方法 onType

signatures:

- **org.hl7.fhir.instance.model.api.IBaseResource onType（Class<org.hl7.fhir.instance.model.api.IBaseResource> resourceType, String name, org.hl7.fhir.instance.model.api.IBaseParameters parameters, Class<org.hl7.fhir.instance.model.api.IBaseParameters> outputType, outputType, 布尔值 useHttpGet, Class<org.hl7.fhir.instance.model.api.IBaseResource> returnType, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**

fhir/onType API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
name	操作名称	字符串
outputParameterType	用于输出参数的类型（这应设置为来自您使用的 FHIR 结构版本的 Parameters.class drawn），可以是 NULL。	类
parameters	用作输入的参数。如果操作不需要任何输入参数，则也可能为 null。	IBaseParameters
resourceType	要操作的资源类型	类
returnType	如果此操作返回单个资源正文作为其返回类型而不是 Parameters 资源，请使用此方法指定该资源类型。这对返回捆绑包而不是 Parameters 资源的某些操作（如 Patient/NN/$everything）非常有用。	类

参数	描述	类型
useHttpGet	使用 HTTP GET 动词	布尔值

33.6.7.5. method processMessage

signatures:

- **org.hl7.fhir.instance.model.api.IBaseBundle processMessage （String respondToUri, org.hl7.fhir.instance.model.api.IBaseBundle msgBundle, boolean asynchronous, Class<org.hl7.fhir.instance.model.api.IBaseBundle> responseClass, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters>, ObjectParameters; extraParameters; extraParameter**

fhir/processMessage API 方法在下表中列出的参数：

参数	描述	类型
asynchronous	是否异步处理消息，还是同步，默认为同步。	布尔值
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
msgBundle	将 Message Bundle 设置为 POST 到消息传递服务器	IBaseBundle
respondToUri	可选的查询参数（表示来自接收服务器的响应应发送到这个 URI）可以是 NULL	字符串
responseClass	响应类	类

除了上面的参数外，fhir API 也可以使用任何 Query 参数。

任何参数可以在端点 URI 中提供，也可以在消息标头中动态提供。消息标头名称必须是 CamelFhir.parameter 的格式。inBody 参数覆盖消息标头，即 endpoint 参数 inBody=myParameterNameHere 覆盖 CamelFhir.myParameterNameHere 标头。

33.6.8. API: patch

支持生成者和消费者

patch API 以语法定义，如下所示：

```
fhir:patch/methodName?[parameters]
```

下表列出 2 方法，后跟每种方法的详细语法。(API 方法可以有一个简写别名名称，可用于语法而不是名称)

æ–¹æ³•	描述
patchById	将补丁应用到给定的资源 ID
patchByUrl	指定更新应作为针对给定搜索 URL 创建的条件执行

33.6.8.1. 方法 patchById

signatures:

- **ca.uhn.fhir.rest.api.MethodOutcome patchById (String patchBody, String stringId, ca.uhn.fhir.rest.api.PreferReturnEnum preferReturnEnum preferReturn, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**
- **ca.uhn.fhir.rest.api.MethodOutcome patchById (String patchBody, org.hl7.fhir.instance.model.api.IIdType id, ca.uhn.fhir.rest.api.PreferReturnEnum preferReturn, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/patchById API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
id	要修补的资源 ID	IIdType

参数	描述	类型
patchBody	补丁文档在 XML 或 JSON 中序列化的正文	字符串
preferReturn	在请求中添加 Prefer 标头，该请求服务器包含或阻止资源正文作为结果的一部分。如果服务器返回资源，它将通过 MethodOutcome#getResource （）解析可由客户端访问的资源。	PreferReturnEnum
stringId	要修补的资源 ID	字符串

33.6.8.2. 方法 patchByUrl

signatures:

- **ca.uhn.fhir.rest.api.MethodOutcome patchByUrl (String patchBody, String url, ca.uhn.fhir.rest.api.PreferReturnEnum preferReturn, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/patchByUrl API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
patchBody	补丁文档在 XML 或 JSON 中序列化的正文	字符串
preferReturn	在请求中添加 Prefer 标头，该请求服务器包含或阻止资源正文作为结果的一部分。如果服务器返回资源，它将通过 MethodOutcome#getResource （）解析可由客户端访问的资源。	PreferReturnEnum
url	要使用的搜索 URL。此 URL 的格式应该是 ResourceTypeParameters，例如：Patientname=Smith&identifier=13.2.4.11.4%7C847366	字符串

除了上面的参数外，fhir API 也可以使用任何 Query 参数。

任何参数可以在端点 URI 中提供，也可以在消息标头中动态提供。消息标头名称必须是 CamelFhir.parameter 的格式。inBody 参数覆盖消息标头，即 endpoint 参数 inBody=myParameterNameHere 覆盖 CamelFhir.myParameterNameHere 标头。

33.6.9. api: read

支持生成者和消费者

read API 以语法定义，如下所示：

```
fhir:read/methodName?[parameters]
```

下表列出 2 方法，后跟每种方法的详细语法。(API 方法可以有一个简写别名名称，可用于语法而不是名称)

æ–¹æ³•	描述
resourceById	根据 id 读取服务器上的 IBaseResource
resourceByUrl	使用 url 读取服务器上的 IBaseResource

33.6.9.1. 方法 resourceById

signatures:

- **org.hl7.fhir.instance.model.api.IBaseResource resourceById（Class<org.hl7.fhir.instance.model.api.IBaseResource> resource, Long longId, String ifVersionMatches, boolean returnNull, org.hl7.fhir.instance.model.api.IBaseResource returnResource, String ifVersionMatches, boolean returnNull, org.hl7.fhir.instance.model.api.IBaseResource> 布尔值 throwError, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**
- **org.hl7.fhir.instance.model.api.IBaseResource resourceById（Class<org.hl7.fhir.instance.model.api.IBaseResource> resource, String stringId, String version, String ifVersionMatches, boolean returnNull, org.hl7.fhir.instance.model.api.IBaseResource returnResource, 布尔值 throwError, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**

- **org.hl7.fhir.instance.model.api.IBaseResource resourceById（Class<org.hl7.fhir.instance.model.api.IBaseResource> resource, org.hl7.fhir.instance.model.api.IIdType id, String ifVersionMatches, boolean returnNull, org.hl7.fhir.instance.model.api.IBaseResource, returnResource, returnResource 布尔值 throwError, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**

- **org.hl7.fhir.instance.model.api.IBaseResource resourceById （String resourceClass, Long longId, String ifVersionMatches, boolean returnNull, org.hl7.fhir.instance.model.api.IBaseResource returnResource, boolean throwError, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters;**

- **org.hl7.fhir.instance.model.api.IBaseResource resourceById （String resourceClass, String stringId, String ifVersionMatches, String version, Boolean returnNull, org.hl7.fhir.instance.model.api.model.api.IBaseResource returnResource, boolean throwError, java.util.Map<org.apache.camel.component.fhir.api.Parameters, Object> extraParameters;**

- **org.hl7.fhir.instance.model.api.IBaseResource resourceById （String resourceClass, org.hl7.fhir.instance.model.api.IIdType id, String ifVersionMatches, boolean returnNull, org.hl7.fhir.instance.model.api.IBaseResource returnResource, java.util.Map<org.apache.camel.component.fhir.api.Parameters>, ObjectParameters>**

fhir/resourceById API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
id	引用资源的 IIdType	IIdType
ifVersionMatches	与服务器上最新版本匹配的版本	字符串
longId	资源 ID	Long
resource	要读取的资源（如 Patient）	类
resourceClass	要读取的资源（如 Patient）	字符串
returnNull	如果版本匹配，则返回 null	布尔值

参数	描述	类型
returnResource	如果版本匹配，则返回资源	IBaseResource
stringId	资源 ID	字符串
throwError	如果版本匹配，则抛出错误	布尔值
version	资源版本	字符串

33.6.9.2. method resourceByUrl

signatures:

- **org.hl7.fhir.instance.model.api.IBaseResource resourceByUrl（Class<org.hl7.fhir.instance.model.api.IBaseResource> resource, String url, ifVersionMatches, boolean returnNull, org.hl7.fhir.instance.model.api.IBaseResource returnResource, String url, ifVersionMatches, boolean returnNull, org.hl7.fhir.instance.model.api.IBaseResource> 布尔值 throwError, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**
- **org.hl7.fhir.instance.model.api.IBaseResource resourceByUrl（Class<org.hl7.fhir.instance.model.api.IBaseResource> resource, org.hl7.fhir.instance.model.api.IIdType iUrl, String ifVersionMatches, boolean returnNull, org.hl7.fhir.instance.model.api.IBaseResource, returnResourceResource, string ifVersionMatches, organization.hl7.fhir 布尔值 throwError, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**
- **org.hl7.fhir.instance.model.api.IBaseResource resourceByUrl (String resourceClass, String url, String ifVersionMatches, boolean returnNull, org.hl7.fhir.instance.model.api.IBaseResource returnResource, boolean throwError, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**
- **org.hl7.fhir.instance.model.api.IBaseResource resourceByUrl （String resourceClass, org.hl7.fhir.instance.model.api.IIdType iUrl, String ifVersionMatches, boolean returnNull, org.hl7.fhir.instance.model.api.IBaseResource returnResource, String ifVersionMatches, boolean returnNull, org.hl7.fhir.instance.model.api.IBaseResource returnResource, 布尔值 throwError, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**

fhir/resourceByUrl API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
iUrl	通过绝对 url 引用资源的 IIdType	IIdType
ifVersionMatches	与服务器上最新版本匹配的版本	字符串
resource	要读取的资源（如 Patient）	类
resourceClass	要读取的资源（如 Patient.class）	字符串
returnNull	如果版本匹配，则返回 null	布尔值
returnResource	如果版本匹配，则返回资源	IBaseResource
throwError	如果版本匹配，则抛出错误	布尔值
url	通过绝对 url 引用资源	字符串

除了上面的参数外，fhir API 也可以使用任何 Query 参数。

任何参数可以在端点 URI 中提供，也可以在消息标头中动态提供。消息标头名称必须是 CamelFhir.parameter 的格式。inBody 参数覆盖消息标头，即 endpoint 参数 inBody=myParameterNameHere 覆盖 CamelFhir.myParameterNameHere 标头。

33.6.10. api: search

支持生成者和消费者

搜索 API 在语法中定义，如下所示：

```
fhir:search/methodName?[parameters]
```

下表中列出了 1 方法，后跟每种方法的详细语法。(API 方法可以有一个简写别名名称，可用于语法而不是名称)

æ–¹æ³•	描述
searchByUrl	根据 URL 直接执行搜索

33.6.10.1. 方法 searchByUrl

signatures:

- **org.hl7.fhir.instance.model.api.IBaseBundle searchByUrl (String url, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/searchByUrl API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
url	要搜索的 URL。请注意，此 URL 可能已完成（例如 ），在这种情况下，客户端的基本 URL 将被忽略。或者可以是 relative （例如，Patientname=foo），在这种情况下，将使用客户端的基本 URL。	字符串

除了上面的参数外，fhir API 也可以使用任何 Query 参数。

任何参数可以在端点 URI 中提供，也可以在消息标头中动态提供。消息标头名称必须是 CamelFhir.parameter 的格式。inBody 参数覆盖消息标头，即 endpoint 参数 inBody=myParameterNameHere 覆盖 CamelFhir.myParameterNameHere 标头。

33.6.11. API: transaction

支持生成者和消费者

事务 API 以语法定义，如下所示：

```
fhir:transaction/methodName?[parameters]
```

下表列出 2 方法，后跟每种方法的详细语法。(API 方法可以有一个简写别名名称，可用于语法而不是名称)

æ–¹æ³•	描述
withBundle	使用给定的原始文本（应是 Bundle 资源）作为事务输入
withResources	使用资源列表作为事务输入

33.6.11.1. 使用Bundle 的方法

signatures:

- **string withBundle (String stringBundle, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**
- **org.hl7.fhir.instance.model.api.IBaseBundle withBundle (org.hl7.fhir.instance.model.api.IBaseBundle bundle, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/withBundle API 方法在下表中列出参数：

参数	描述	类型
bundle	在事务中使用的捆绑包	IBaseBundle
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
stringBundle	在事务中使用的捆绑包	字符串

33.6.11.2. method withResources

signatures:

- **java.util.List<org.hl7.fhir.instance.model.api.IBaseResource> withResources**

(java.util.List<org.hl7.fhir.instance.model.api.IBaseResource> resources, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);

fhir/withResources API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
资源	在事务中使用的资源	list

除了上面的参数外，fhir API 也可以使用任何 Query 参数。

任何参数可以在端点 URI 中提供，也可以在消息标头中动态提供。消息标头名称必须是 CamelFhir.parameter 的格式。inBody 参数覆盖消息标头，即 endpoint 参数 inBody=myParameterNameHere 覆盖 CamelFhir.myParameterNameHere 标头。

33.6.12. API: update

支持生成者和消费者

更新 API 以语法定义，如下所示：

```
fhir:update/methodName?[parameters]
```

下表列出 2 方法，后跟每种方法的详细语法。(API 方法可以有一个简写别名名称，可用于语法而不是名称)

æ–¹æ³•	描述
resource	按 id 更新服务器上的 IBaseResource
resourceBySearchUrl	通过搜索 url 更新服务器上的 IBaseResource

33.6.12.1. 方法资源

signatures:

- **ca.uhn.fhir.rest.api.MethodOutcome resource (String resourceAsString, String stringId, ca.uhn.fhir.rest.api.PreferReturnEnum preferReturn, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

- **ca.uhn.fhir.rest.api.MethodOutcome resource (String resourceAsString, org.hl7.fhir.instance.model.api.IIdType id, ca.uhn.fhir.rest.api.PreferReturnEnum preferReturnEnum preferReturn, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

- **ca.uhn.fhir.rest.api.MethodOutcome resource (org.hl7.fhir.instance.model.api.IBaseResource resource, String stringId, ca.uhn.fhir.rest.api.PreferReturnEnum preferReturnEnum preferReturn, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

- **ca.uhn.fhir.rest.api.MethodOutcome resource（org.hl7.fhir.instance.model.api.IBaseResource resource, org.hl7.fhir.instance.model.api.IIdType id, ca.uhn.fhir.rest.api.PreferReturnnumE preferReturnnum, java.util.Map<org.apache.camel.component.fhir.api.Parameters>s; extraParameters; extraParameters;**

fhir/resource API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
id	引用资源的 IIdType	IIdType
preferReturn	服务器是否包含或阻止资源正文作为结果的一部分	PreferReturnEnum
resource	要更新的资源（如 Patient）	IBaseResource
resourceAsString	要更新的资源正文	字符串
stringId	引用资源的 ID	字符串

33.6.12.2. method resourceBySearchUrl

signatures:

- **ca.uhn.fhir.rest.api.MethodOutcome resourceBySearchUrl (String resourceAsString, String url, ca.uhn.fhir.rest.api.PreferReturnEnum preferReturnEnum preferReturn, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

- **ca.uhn.fhir.rest.api.MethodOutcome resourceBySearchUrl (org.hl7.fhir.instance.model.api.IBaseResource resource, String url, ca.uhn.fhir.rest.api.PreferReturnEnum preferReturnEnum preferReturnEnum preferReturn, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/resourceBySearchUrl API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
preferReturn	服务器是否包含或阻止资源正文作为结果的一部分	PreferReturnEnum
resource	要更新的资源（如 Patient）	IBaseResource
resourceAsString	要更新的资源正文	字符串
url	指定更新应作为针对给定搜索 URL 创建的条件执行	字符串

除了上面的参数外，fhir API 也可以使用任何 Query 参数。

任何参数可以在端点 URI 中提供，也可以在消息标头中动态提供。消息标头名称必须是 CamelFhir.parameter 的格式。inBody 参数覆盖消息标头，即 endpoint 参数 inBody=myParameterNameHere 覆盖 CamelFhir.myParameterNameHere 标头。

33.6.13. api: validate

支持生成者和消费者

validate API 在语法中定义，如下所示：

```
fhir:validate/methodName?[parameters]
```

下表中列出了 1 方法，后跟每种方法的详细语法。(API 方法可以有一个简写别名名称，可用于语法而不是名称)

æ–¹æ³•	描述
resource	验证资源

33.6.13.1. 方法资源

signatures:

- **ca.uhn.fhir.rest.api.MethodOutcome resource (String resourceAsString, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**
- **ca.uhn.fhir.rest.api.MethodOutcome resource (org.hl7.fhir.instance.model.api.IBaseResource resource, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/resource API 方法在下表中列出的参数：

参数	描述	类型
extraParameters	有关可传递的参数的完整列表，请参阅 ExtraParameters，可以是 NULL	Map
resource	用于验证的 IBaseResource	IBaseResource
resourceAsString	用于验证的原始资源	字符串

除了上面的参数外，fhir API 也可以使用任何 Query 参数。

任何参数可以在端点 URI 中提供，也可以在消息标头中动态提供。消息标头名称必须是 CamelFhir.parameter 的格式。inBody 参数覆盖消息标头，即 endpoint 参数 inBody=myParameterNameHere 覆盖 CamelFhir.myParameterNameHere 标头。

33.7. SPRING BOOT AUTO-CONFIGURATION

组件支持 56 个选项，如下所列。

Name	描述	默认值	类型
camel.component.fhir.access-token	OAuth 访问令牌。		字符串
camel.component.fhir.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.fhir.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.fhir.client	使用自定义客户端。选项是一个 ca.uhn.fhir.rest.client.api.IGenericClient 类型。		IGenericClient
camel.component.fhir.client-factory	使用自定义客户端工厂。选项是一个 ca.uhn.fhir.rest.client.api.IRestfulClientFactory 类型。		IRestfulClientFactory
camel.component.fhir.compress	将传出(POST/PUT)内容压缩为 GZIP 格式。	false	布尔值
camel.component.fhir.configuration	使用共享配置。选项是 org.apache.camel.component.fhir.FhirConfiguration 类型。		FhirConfiguration
camel.component.fhir.connection-timeout	尝试和建立初始 TCP 连接（毫秒）。	10000	整数

Name	描述	默认值	类型
camel.component.fhir.defer-model-scanning	当设定此选项时，不会为子对象扫描模型类，直到实际访问给定类型的子列表为止。	false	布尔值
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
camel.component.fhir.encoding	用于所有请求的编码。		字符串
camel.component.fhir.fhir-context	FhirContext 是要创建的昂贵的对象。为避免创建多个实例，可以直接设置它。选项是一个 ca.uhn.fhir.context.FhirContext 类型。		FhirContext
camel.component.fhir.fhir-version	要使用的 FHIR 版本。	R4	字符串
camel.component.fhir.force-conformance-check	强制检查。	false	布尔值
camel.component.fhir.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.fhir.log	将记录每个请求和响应。	false	布尔值
camel.component.fhir.password	用于基本身份验证的用户名。		字符串
camel.component.fhir.pretty-print	用户用户打印所有请求。	false	布尔值
camel.component.fhir.proxy-host	代理主机。		字符串
camel.component.fhir.proxy-password	代理密码。		字符串
camel.component.fhir.proxy-port	代理端口。		整数

Name	描述	默认值	类型
camel.component.fhir.proxy-user	代理用户名。		字符串
camel.component.fhir.server-url	FHIR 服务器基本 URL。		字符串
camel.component.fhir.session-cookie	要添加到每个请求的 HTTP 会话 Cookie。		字符串
camel.component.fhir.socket-timeout	阻止单个读/写操作的时长（单位为 ms）。	10000	整数
camel.component.fhir.summary	请求服务器使用 _summary 参数修改响应。		字符串
camel.component.fhir.username	用于基本身份验证的用户名。		字符串
camel.component.fhir.validation-mode	当 Camel 验证 FHIR 服务器的合规声明时。	ONCE	字符串
camel.dataformat.fhirjson.content-type-header	数据格式是否应该使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式到 XML 的 application/xml 或用于数据格式的 application/json 发送到 JSON。	true	布尔值
camel.dataformat.fhirjson.dont-encode-elements	如果提供，则指定不应编码的元素。此字段的有效值包括： 病人 - Don't encode 病人及其所有子人的 Patient.name - Don 并不编码病人名 Patient.name.family - Don't encode of the patient's family name .text - Don't en the text element on any resource （只有第一个位置可能包含通配符） DSTU2 备注： 请注意，包括 meta 的值（如 Patient.meta ）将可用于 DSTU2 解析器，但 meta （如 Patient.meta.lastUpdated）的值只能在 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对其资源版本进行编码，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。这个方法提供了一个比 setStripVersionsFromReferences (String)的精细控制级别，即使 setStripVersionsFromReferences (String)被设置为 true （这是默认值）。		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

Name	描述	默认值	类型
camel.dataformat.fhirjson.encode-elements	如果提供，指定应编码的元素，以排除所有其他元素。此字段的有效值包括：Patient - Encode 病人及其所有子 Patient.name - Encode 只包括病人的名称 Patient.name.family - Encode 只包括 .text - Encode - Encode the text element on any resource （只有第一个位置可能包含通配符）.（必需）- 这是一个特殊情况，这会导致任何必需的字段(min)为 0。		Set
camel.dataformat.fhirjson.encode-elements-applies-to-child-resources-only	如果设置为 true （默认为 false），提供给 setEncodeElements (Set)的值不会应用到根资源（通常是 Bundle），但会应用到该捆绑包中包含的任何子资源（例如，搜索结果资源）。	false	布尔值
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）被编码的任何资源的 ID 不会包含在输出中。请注意，这不适用于包含的资源，仅适用于 root 资源。换句话说，如果将其设置为 true，则包含的资源仍具有本地 ID，但外部/包含 ID 将没有 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），则 Bundle.entry.fullUrl 将覆盖 Bundle.entry.resource 的资源 id （如果定义了 fullUrl）。在将源数据解析到 Bundle 对象时，会发生此行为。如果这不是所需的行为（例如，要在 fullUrl 和资源 ID 之间执行其他验证检查，则将其设置为 false）。	false	布尔值
camel.dataformat.fhirjson.pretty-print	设置用户友善的打印标志，这意味着解析器将使用人类可读的空间和新线对资源进行编码，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，如果资源引用作为绝对 URL 提供，则资源引用将转换为相对引用，但具有与给定基础匹配的基础。		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编码时删除版本。这通常很好，因为多数情况下，从一个资源到另一个资源的引用应该通过 ID 指向资源，而不是 ID 和版本。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。如果需要精细的控制，请使用 setDontStripVersionsFromReferencesAtPaths (List)。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.fhirjson.summary-mode	如果设置为 true （默认为 false）则仅包含由 FHIR 规格标记为 is 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），则行为不会包含在编码的值中。	false	布尔值
camel.dataformat.fhirxml.content-type-header	数据格式是否应该使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式到 XML 的 application/xml 或用于数据格式的 application/json 发送到 JSON。	true	布尔值
camel.dataformat.fhirxml.dont-encode-elements	如果提供，则指定不应编码的元素。此字段的有效值包括： 病人 - Don't encode 病人及其所有子人的 Patient.name - Don 并不编码病人名 Patient.name.family - Don't encode of the patient's family name .text - Don't en the text element on any resource （只有第一个位置可能包含通配符） DSTU2 备注： 请注意，包括 meta 的值（如 Patient.meta ）将可用于 DSTU2 解析器，但 meta （如 Patient.meta.lastUpdated）的值只能在 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对其资源版本进行编码，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。这个方法提供了一个比 setStripVersionsFromReferences (String)的精细控制级别，即使 setStripVersionsFromReferences (String)被设置为 true （这是默认值）。		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，指定应编码的元素，以排除所有其他元素。此字段的有效值包括：Patient - Encode 病人及其所有子 Patient.name - Encode 只包括病人的名称 Patient.name.family - Encode 只包括 .text - Encode - Encode the text element on any resource （只有第一个位置可能包含通配符） .（必需）- 这是一个特殊情况，这会导致任何必需的字段(min)为 0。		Set
camel.dataformat.fhirxml.encode-elements-applies-to-child-resources-only	如果设置为 true （默认为 false），提供给 setEncodeElements (Set)的值不会应用到根资源（通常是 Bundle），但会应用到该捆绑包中包含的任何子资源（例如，搜索结果资源）。	false	布尔值

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）被编码的任何资源的 ID 不会包含在输出中。请注意，这不适用于包含的资源，仅适用于 root 资源。换句话说，如果将其设置为 true，则包含的资源仍具有本地 ID，但外部/包含 ID 将没有 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），则 Bundle.entry.fullUrl 将覆盖 Bundle.entry.resource 的资源 id （如果定义了 fullUrl）。在将源数据解析到 Bundle 对象时，会发生此行为。如果这不是所需的行为（例如，要在 fullUrl 和资源 ID 之间执行其他验证检查，则将其设置为 false）。	false	布尔值
camel.dataformat.fhirxml.pretty-print	设置用户友善的打印标志，这意味着解析器将使用人类可读的空间和新线对资源进行编码，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，如果资源引用作为绝对 URL 提供，则资源引用将转换为相对引用，但具有与给定基础匹配的基础。		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编码时删除版本。这通常很好，因为多数情况下，从一个资源到另一个资源的引用应该通过 ID 指向资源，而不是 ID 和版本。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）则仅包含由 FHIR 规格标记为 is 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），则行为不会包含在编码的值中。	false	布尔值

第 34 章 FILE

支持生成者和消费者

File 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。

34.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用文件时，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-file-starter</artifactId>
</dependency>
```

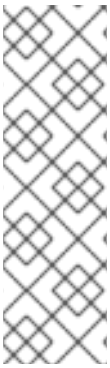
34.2. URI 格式

```
file:directoryName[?options]
```

其中 directoryName 代表底层文件目录。

仅目录

Camel 仅支持使用起始目录配置的端点。因此，catalogName 必须是目录。如果只想消耗单个文件，您可以使用 fileName 选项，例如通过设置 fileName=thefilename。此外，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。



注意

避免读取当前由另一个应用程序写入的文件。注意 JDK File IO API 在检测另一个应用程序当前正在写入/复制文件时有点限制。根据操作系统平台，实施也可以不同。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，您必须自己自行调查您的环境套件。为了帮助此 Camel，可以使用不同的 readLock 选项和 doneFileName 选项。另请参阅 使用来自其他直接丢弃文件的文件夹的文件 部分。

34.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

34.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

34.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

34.4. 组件选项

File 组件支持 3 个选项，如下所列。

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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

34.5. 端点选项

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：

34.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
directoryName (common)	**必需** 起始目录。		File

34.5.2. 查询参数(94 参数)

Name	描述	默认值	类型

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 知道它应在访问文件内容时加载文件内容。在编写文件时，您也可以使用此选项来指定同时写入该文件的 charset。请记住，在编写文件 Camel 时，可能需要将消息内容读到内存中，才能将数据转换为配置的 charset，因此如果您有大量消息，则不要使用它。		字符串
doneFileName (common)	生产者：如果提供，则 Camel 将在写入原始文件时编写完第 2 个文件。完成的文件将为空。这个选项配置要使用的文件名。可以指定固定名称。或者您可以使用动态占位符。完成的文件将始终写在与原始文件相同的文件夹中。消费者：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。可以指定固定名称。或者，您可以使用动态占位符。done 文件始终位于与原始文件相同的文件夹中。仅支持 $\\{file.name} 和 $\\{file.name.next} 作为动态占位符。		字符串
filename (common)	使用文件语言等表达式动态设置文件名。对于消费者，它用作文件名过滤器。对于生成者，它用于评估要写入的文件名。如果设置了表达式，它将优先于 CamelFileName 标头。（注：标题本身也可以是表达式）。表达式选项支持 String 和 Expression 类型。如果表达式是 String 类型，则始终使用 File Language 来评估它。如果表达式是 Expression 类型，则使用指定的 Expression 类型 - 这允许您，使用 OGNL 表达式。对于消费者，您可以使用它来过滤文件名，因此您可以使用 File Language 语法：mydata-$\\{date:now:yyyyMMdd}.txt，实例消耗了现在的文件。生产者支持 CamelOverruleFileName 标头，它优先于任何现有的 CamelFileName 标头；CamelOverruleFileName 是一个仅使用的标头，因此可以更轻松地避免临时存储 CamelFileName，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，则该文件会在成功处理后删除。	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，要将文件移动到 .error 子目录，请使用： .error。注意：当将文件移动到故障位置 Camel 将处理错误时，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。这个选项适用于只读数据，或用于 ETL 类型要求。如果 noop=true，Camel 也会设置 idempotent=true，以避免通过和再次消耗同一文件。	false	布尔值
preMove (consumer)	表达式（如文件语言）用于在处理前动态设置文件名。例如，要将 in-progress 文件移到订购目录中，将此值设置为 order。		字符串
preSort (consumer)	启用 pre-sort 后，消费者将在轮询期间对文件和目录名称进行排序，该名称从文件系统检索。如果您需要按排序的顺序对文件进行操作，您可能需要执行此操作。预排序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，也会在所有子目录中查找文件。	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 (advanced))	与 startDirectoryMustExist 选项类似，但这会在轮询期间应用（在启动消费者后）。	false	布尔值
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
extendedAttributes (consumer (advanced))	定义感兴趣的文件属性。与 posix:permissions,posix:owner,basic:lastAccessTime 一样，它支持基本的通配符，如 posix:, basic:lastAccessTime。		字符串

Name	描述	默认值	类型
inProgressRepository (consumer (advanced))	可插拔式存储库 org.apache.camel.spi.IdempotentRepository.in-progress 存储库用于考虑被消耗的进度文件的当前。默认情况下，使用基于内存的存储库。		IdempotentRepository
localWorkDirectory (consumer (advanced))	使用时，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。这非常有用，如果您使用一个非常大的远程文件，从而可以节省内存。		字符串
onCompletionExceptionHandler (consumer (advanced))	使用自定义 org.apache.camel.spi.ExceptionHandler 处理在完成过程中发生的任何抛出异常，供消费者执行提交或回滚。默认实现将在 WARN 级别记录任何异常并忽略。		ExceptionHandler
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
probeContentType (consumer (advanced))	是否启用内容类型的探测。如果启用，则消费者使用 Files#probeContentType (java.nio.file.Path)来确定文件的内容类型，并将它存储为消息上带有键 Exchange#FILE_CONTENT_TYPE 的标头。	false	布尔值
processStrategy (consumer (advanced))	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，如存在特殊就绪的文件。如果设置了这个选项，则不会应用 readLock 选项。		GenericFileProcessStrategy
resumeStrategy (consumer (advanced))	为文件设置恢复策略。这样，可以在停止应用程序前定义在最后一次点后恢复读取文件的策略。有关实现详情，请参阅 FileConsumerResumeStrategy。		FileConsumerResumeStrategy
startingDirectoryMustExist (consumer (advanced))	启动目录是否必须存在。请记住，autoCreate 选项是默认启用的，这意味着如果启动目录不存在，则启动目录通常会自动创建。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
startingDirectoryMustHaveAccess (consumer (advanced))	起始目录是否具有访问权限。请记住，startDirectoryMustExist 参数必须设置为 true 才能验证目录是否存在。如果目录没有读写权限，则抛出异常。	false	布尔值

Name	描述	默认值	类型
appendChars (producer)	用于在编写文件后附加字符（文本）。例如，这可用于在编写和附加新文件或现有文件时添加新行或其他分隔符。要指定换行符(slash-n 或 slash-r)或 tab (slash-t)字符，然后使用额外的斜杠(eg slash-slash-n)进行转义。		字符串
fileExist (producer)	如果文件名称已存在，则该怎么办。override （默认文件）替换现有文件。- Append - 将内容添加到现有文件。- Fail - 抛出 GenericFileOperationException，表示已有现有文件。- Ignore - 静默忽略问题，且不会覆盖现有文件，但假设所有内容都正常。- Move - 选项需要使用 moveExisting 选项进行配置。选项 eagerDeleteTargetFile 可用于控制移动文件时要执行的操作，并且存在现有文件，否则会导致 move 操作失败。Move 选项将移动任何现有文件，然后再编写目标文件。- 只有使用 tempFileName 选项时才适用 TryRename。这允许尝试将该文件从临时名称重命名为实际名称，而无需进行任何存在的检查。对于某些文件系统，特别是 FTP 服务器上，这个检查可能会更快。 Enum 值： • override • 附加 • Fail • Ignore • Move • TryRename	override	GenericFileExist
flatten (producer)	flatten 用于扁平化文件名路径，以剥离任何前导路径，因此它只是文件名。这样，您便可以将递归地消耗到子目录中，但当您将文件写入另一个目录时，会将文件写入单个目录中。在生成者上将其设置为 true 强制执行 CamelFileName 标头中的任何文件名将被剥离任何前导路径。	false	布尔值
jailStartingDirectory (producer)	用于仅向起始目录（和子）写入文件。这默认是启用的，不允许 Camel 将文件写入外部目录（在开箱即用的情况下更为安全）。您可以关闭此选项，以允许将文件写入起始目录之外的目录，如父目录或根文件夹。	true	布尔值

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
moveExisting (producer)	表达式（如文件语言）用于在配置 fileExist=Move 时使用计算文件名。要将文件移动到备份子目录中，只需输入 backup。这个选项只支持以下文件语言令牌：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动到基于当前 dir 作为基础的相对目录。		字符串
tempFileName (producer)	与 tempPrefix 选项相同，但对临时文件名的命名提供更加精细的控制，因为它使用文件语言。tempFilename 的位置相对于选项 'fileName' 中的最终文件位置，而不是基础 uri 中的目标目录。例如，如果选项 fileName 包含目录前缀：dir/finalFilename，则 tempFileName 相对于该子目录 dir。		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，也可避免消费者（不使用专用读取锁定）读取进度文件。在上传大型文件时，FTP 通常使用 FTP。		字符串
allowNullBody (producer (advanced))	用于指定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，如果设为 false，并尝试将 null 正文发送到文件组件，则将抛出 'Cannot write null body to file.' 的 GenericFileWriteException。如果 fileExist 选项被设置为 'Override'，则该文件将被截断，如果设为 附加该文件，则该文件将保持不变。	false	布尔值
chmod (producer (advanced))	指定制作者发送的文件权限，chmod 值必须在 000 和 777 之间；如果有一个前导数字，如 0755，我们将忽略它。		字符串
chmodDirectory (producer (advanced))	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 0755 中有一个前导数字，我们将忽略它。		字符串

Name	描述	默认值	类型
eagerDeleteTargetFile (producer (advanced))	是否强制删除任何现有目标文件。这个选项仅在使用 fileExists=Override 和 tempFileName 选项时才适用。您可以使用此选项禁用（将其设置为 false）在编写 temp 文件前删除目标文件。例如，您可以编写大文件，并且希望目标文件在 temp 文件被写入期间存在。这样可保证仅删除目标文件，直到最后一次时间之前，只需将 temp 文件重命名为目标文件名之前。这个选项还用于控制是否在启用 fileExist=Move 时删除任何现有的文件，并且存在现有文件。如果此选项 copyAndDeleteOnRenameFails false，则现有文件存在时会抛出异常（如果其为 true），则在移动操作前会删除现有文件。	true	布尔值
forceWrites (producer (advanced))	是否强制同步对文件系统的写操作。如果您不希望此级别保证，例如，如果写入日志/审计日志等，您可以关闭此项。这会产生更好的性能。	true	布尔值
keepLastModified (producer (advanced))	将保留来自源文件的最后修改的时间戳（如果有）。将使用 Exchange.FILE_LAST_MODIFIED 标头来定位时间戳。此标头可以包含 java.util.Date 或 long （时间戳）。如果时间戳存在，且启用了 选项，它将在写入的文件上设置这个时间戳。注意：此选项仅适用于文件制作者。您不能将此选项与任何 ftp producer 一起使用。	false	布尔值
moveExistingFileStrategy (producer (advanced))	策略(Custom Strategy)用于移动带有特殊命名令牌的文件，以便在配置了 fileExist=Move 时使用。默认情况下，如果没有提供自定义策略，则使用实现。		FileMoveExistingStrategy
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。对于文件制作者，这意味着应写入文件的目录。	true	布尔值
bufferSize (advanced)	用于编写文件的缓冲区大小（如果是 FTP 用于下载和上传文件）。	131072	int
copyAndDeleteOnRenameFail (advanced)	如果无法重命名该文件，是否回退到副本和删除文件。这个选项不适用于 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（例如在不同文件系统或网络之间）。这个选项优先于 copyAndDeleteOnRenameFail 参数，该参数会自动回退到复制和删除策略，但仅在额外的延迟后。	false	布尔值
同步 (advanced)	设置是否应严格使用同步处理。	false	布尔值

Name	描述	默认值	类型
antExclude (filter)	Ant 样式过滤器排除。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。		字符串
antFilterCaseSensitive (filter)	在 ant 过滤器中设置问题单敏感标志。	true	布尔值
antInclude (filter)	Ant 样式过滤器包含.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 的限制是否为 eager。如果为 eager，则限制在文件扫描期间。其中为 false 将扫描所有文件，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，这需要较高的内存用量，因为所有文件详情都在内存中执行排序。	true	布尔值
exclude (filter)	用于排除文件，如果文件名与正则表达式模式匹配（匹配区分大小写）。请注意，如果您使用符号（如加号），如果将其配置为 endpoint uri，则需要使用 RAW （）语法进行配置。有关配置 endpoint uris 的更多详细信息。		字符串
excludeExt (filter)	用于排除匹配文件扩展名名称的文件（不区分大小写）。例如，要排除 bak 文件，然后使用 excludeExt=bak。多个扩展可以用逗号分开，例如要排除 bak 和 dat 文件，请使用 excludeExt=bak,dat。请注意，文件扩展名包含所有部分，例如，具有名为 mydata.tar.gz 的文件将扩展为 tar.gz。要获得更大的灵活性，请使用 include/exclude 选项。		字符串
Filter (filter)	可插拔过滤器作为 org.apache.camel.component.file.GenericFileFilter 类。如果过滤器在其 accept （）方法中返回 false，则将跳过文件。		GenericFileFilter
filterDirectory (filter)	根据简单语言过滤目录。例如，要过滤当前日期，您可以使用一个简单的日期模式，如 $\\{date:now:yyyMMdd}。		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 跳过已经处理的文件。默认情况下，将使用基于内存的 LRUCache 来保存 1000 条目。如果 noop=true，则同时启用幂等性，以避免再次消耗同一文件。	false	布尔值

Name	描述	默认值	类型
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。您可以使用文件语言，例如要使用文件名和文件大小，您可以执行： idempotentKey=$\\{file:name}-$\\{file:size}-$\\{file:size}。		字符串
idempotentRepository (filter)	可插拔存储库 org.apache.camel.spi.IdempotentRepository，如果未指定，并且 idempotent 为 true，则默认使用 MemoryIdempotentRepository。		IdempotentRepository
Include (filter)	用于包括文件，如果文件名与正则表达式模式匹配（匹配区分大小写）。请注意，如果您使用符号（如加号），如果将其配置为 endpoint uri，则需要使用 RAW （）语法进行配置。有关配置 endpoint uris 的更多详细信息。		字符串
includeExt (filter)	用于包括匹配文件扩展名名称的文件（不区分大小写）。例如，要包含 txt 文件，然后使用 includeExt=txt。多个扩展可以用逗号分开，例如要包含 txt 和 xml 文件，请使用 includeExt=txt,xml。请注意，文件扩展名包含所有部分，例如，具有名为 mydata.tar.gz 的文件将扩展为 tar.gz。要获得更大的灵活性，请使用 include/exclude 选项。		字符串
maxDepth (filter)	递归处理目录时要遍历的最大深度。	2147483647	int
maxMessagesPerPoll (filter)	定义每个轮询收集的最多消息。默认情况下，没有设置最大值。可用于设置限制，例如 1000 个，以避免启动有数千个文件的服务器。将值设为 0 或负数设置为禁用它。注意：如果此选项正在使用，则文件和 FTP 组件将在任何排序之前进行限制。例如，如果您有 100000 文件并使用 maxMessagesPerPoll=500，则只有前 500 个文件会被提取，然后排序。您可以使用 eagerMaxMessagesPerPoll 选项，并将其设置为 false 以允许首先扫描所有文件，然后在之后排序。		int
minDepth (filter)	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）用于在处理后移动文件名时动态设置文件名。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 实现。		GenericFileExclusiveReadLockStrategy

Name	描述	默认值	类型
readLock (lock)	供消费者使用，仅当文件上有独占的 read-lock （例如，该文件不是 in-progress 或被写入）时轮询文件。Camel 将等待文件锁定被赋予。此选项在策略中提供构建：- none - No read lock is in use - markerFile - Camel 创建一个标记文件(fileName.camelLock)，然后在其上保存锁定。这个选项不适用于 FTP 组件 - 更改 - Changed 使用文件长度/修改时间戳来检测文件当前是否正在复制。至少将使用 1 sec 来确定这一点，因此此选项不能像其他人一样快消耗文件，但可以更可靠，因为 JDK IO API 无法始终确定文件当前是否被其他进程使用。选项 readLockCheckInterval 可用于设置检查频率。- fileLock - 用于使用 java.nio.channels.FileLock。这个选项不适用于 Windows OS 和 FTP 组件。当通过 mount/share 访问远程文件系统时，应避免使用这种方法，除非该文件系统支持分布式文件锁定。- rename 使用尝试将文件重命名为测试（如果我们可以获得独家的 read-lock. - 幂等 - （仅针对文件组件）是使用幂等Repository 作为 read-lock。这允许在幂等存储库实施支持集群时使用读取锁定。- idempotent-changed - （仅适用于文件组件） idempotent-changed 用于使用 idempotentRepository 并更改为组合的 read-lock。这允许在幂等存储库实施支持集群时使用读取锁定。- idempotent-rename - （仅适用于文件组件） idempotent-rename 用于使用 idempotentRepository，并作为组合的 read-lock 重命名。如果幂等存储库实现支持集群，这允许使用支持集群的读取锁定。不同的读取锁定并不适合以集群模式工作，其中不同节点上的并发消费者对共享文件系统上的相同文件竞争。flagsFile 使用接近 atomic 操作来创建空标记文件，但无法保证在集群中工作。fileLock 可以更好地工作，但文件系统需要支持分布式文件锁定，以此类推。如果幂等存储库支持集群（如 Hazelcast 组件或 Infinispan），则使用幂等的读取锁定可以支持集群。 Enum 值： • none • markerFile • fileLock • rename • changed • idempotent • idempotent-changed • idempotent-rename	none	字符串

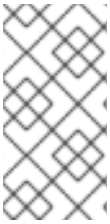
Name	描述	默认值	类型
readLockCheckInterval (lock)	如果读取锁定支持，则 read-lock 中的间隔为 millis。此间隔用于尝试获取读取锁之间的睡眠状态。例如，在使用更改的读取锁定时，您可以为文件设置更高的间隔周期，以便进行速度较慢的写入。默认值 1 sec。如果生成者的写入速度非常慢，则可能太快。注意：对于 FTP，默认的 readLockCheckInterval 是 5000。readLockTimeout 值必须大于 readLockCheckInterval，但 thumb 规则是有一个超时，它至少比 readLockCheckInterval 高 2 倍。这需要确保允许读取锁定进程在达到超时时间前尝试获取锁定。	1000	long
readLockDeleteOrphanLockFiles (lock)	如果 Camel 没有正确关闭（如 JVM 崩溃），在启动后是否应该使用标记文件读取锁定删除任何孤立的读取锁定文件（如 JVM 崩溃）。如果将此选项转换为 false，则任何孤立的锁文件将导致 Camel 不尝试选择该文件，也可能是因为另一个节点同时从同一共享目录读取文件。	true	布尔值
readLockIdempotentReleaseAsync (lock)	延迟发行任务是否应同步或异步。请参阅 readLockIdempotentReleaseDelay 选项的更多详情。	false	布尔值
readLockIdempotentReleaseAsyncPoolSize (lock)	使用异步发行任务时调度的线程池中的线程数量。几乎所有用例中，使用默认的 1 个核心线程应该已经足够了，只有在更新幂等存储库时，或者有大量要处理的文件时，才会将其设置为更高的值。如果您通过配置 readLockIdempotentReleaseExecutorService 选项使用共享线程池，这个选项不会被使用。请参阅 readLockIdempotentReleaseDelay 选项的更多详情。		int
readLockIdempotentReleaseDelay (lock)	是否延迟 millis 期间内发行任务。这可用于在带有共享幂等存储库的主动/主动集群场景中，延迟发行任务以扩展窗口，因为存在竞争条件，因此无法扫描并获取相同的文件。通过扩展发行版本任务的 time-window 有助于防止这种情况。只有在将 readLockRemoveOnCommit 配置为 true 时才需要延迟。		int
readLockIdempotentReleaseExecutorService (lock)	使用自定义和共享线程池进行异步发行任务。请参阅 readLockIdempotentReleaseDelay 选项的更多详情。		ScheduledExecutorService

Name	描述	默认值	类型
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 DEBUG。您可以更改此级别，例如，OFF 没有任何日志记录。这个选项只适用于 readLock 类型：change, fileLock, idempotent, idempotent-changed, idempotent-rename, rename。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	DEBUG	LoggingLevel
readLockMarkerFile (lock)	是否使用更改、重命名或专用读取锁定类型的标记文件。默认情况下，使用标志文件来保护获取同一文件的其他进程。通过将这个选项设置为 false 可关闭此行为。例如，如果您不希望 Camel 应用程序将标记文件写入文件系统。	true	布尔值
readLockMinAge (lock)	这个选项只适用于 readLock=changed。它允许在尝试获取读取锁定前指定文件的最短期限。例如，使用 readLockMinAge=300s 来要求文件在最后 5 分钟内。这可加快更改的读取锁，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 期望文件包含数据，因此默认值为 1。您可以将这个选项设为零，以允许消耗零长度文件。	1	long
readLockRemoveOnCommit (lock)	这个选项只适用于 readLock=idempotent。它允许您指定在处理文件成功时是否从幂等存储库中删除文件名条目，以及提交发生。默认情况下，文件不会被删除，这样可确保不会发生任何竞争条件，因此另一个活动节点可能会试图获取该文件。相反，idempotent 存储库可能支持驱除策略，这些策略可在 X 分钟后驱除文件名条目 - 这确保了竞争条件没有问题。请参阅 readLockIdempotentReleaseDelay 选项的更多详情。	false	布尔值
readLockRemoveOnRollback (lock)	这个选项只适用于 readLock=idempotent。它指定在处理文件失败时是否从幂等存储库中删除文件名条目，以及进行回滚。如果此选项为 false，则确认文件名条目（就像文件执行提交一样）。	true	布尔值

Name	描述	默认值	类型
readLockTimeout (lock)	如果 read-lock 支持，则 read-lock 中的可选超时（如果支持）。如果无法授予 read-lock，并且触发超时，则 Camel 将跳过该文件。在下次轮询 Camel 时，将再次尝试 文件，这一次可能被赋予读锁。使用 0 或更低的值来指示永久。目前 fileLock，更改并重命名支持超时。注意：对于 FTP，默认的 readLockTimeout 值为 20000，而不是 10000。readLockTimeout 值必须大于 readLockCheckInterval，但 thumb 规则是有一个超时，它至少比 readLockCheckInterval 高 2 倍。这需要确保允许读取锁定进程在达到超时时间前尝试获取锁定。	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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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

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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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

Name	描述	默认值	类型
shuffle (sort)	要表示文件列表（按随机顺序排列）。	false	布尔值
sortby (sort)	使用文件语言进行内置排序。支持嵌套排序，因此您可以按文件名排序，并且按修改日期排序第二组。		字符串
sorter (sort)	可插拔排序器作为 java.util.Comparator 类。		比较器



注意

文件生成者的默认行为
默认情况下，它会覆盖任何已存在的、带有相同名称的文件。

34.6. 移动和删除操作

在(post 命令)路由完成后执行任何移动或删除操作；因此，在处理 交换 时，该文件仍然位于 inbox 文件夹中。

我们通过一个示例来说明这一点：

```
from("file://inbox?move=.done").to("bean:handleOrder");
```

当在 inbox 文件夹中丢弃文件时，文件消费者会注意到它，并创建一个新的 FileExchange，它路由到 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，路由完成后，文件消费者将执行 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（尽管您使用 文件 语言等表达式），则表达式评估的结果是要使用的文件名。https://access.redhat.com/documentation/zh-cn/red_hat_build_of_apache_camel/4.4/html-single/red_hat_build_of_apache_camel_for_spring_boot_reference/index#csb-camel-simple-language-starter例如，如果设置了：

```
move=../backup/copy-of-${file:name}
```

然后，使用我们使用的文件语言返回要使用的文件名，可以是相对或绝对的。https://access.redhat.com/documentation/zh-cn/red_hat_build_of_apache_camel/4.4/html-single/red_hat_build_of_apache_camel_for_spring_boot_reference/index#csb-camel-file-language-starter如果相对，则该目录作为子文件夹创建自使用该文件的文件夹内。

默认情况下，Camel 会将消耗的文件移动到相对于文件被消耗的目录的 .camel 子文件夹。

如果要在处理后删除文件，路由应该是：

```
from("file://inbox?delete=true").to("bean:handleOrder");
```

我们引入了一个 预先 移动操作，以便在文件被处理 前 移动文件。这可让您在处理前标记哪些文件已被扫描，因为它们被移动到此子文件夹中。

```
from("file://inbox?preMove=inprogress").to("bean:handleOrder");
```

您可以组合 预先 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，当处理和处理后，文件位于 progress 文件夹中，它会移到 .done 文件夹。

34.7. 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，当我们输入 move=.done Camel 时，会将它转换为 ：${file:parent}/.done/${file:onlyname}。只有 Camel 检测到您在 option 值中未提供 $\{ } 时才完成此操作。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天为模式的备份文件夹中，我们可以：

```
move=backup/${date:now:yyyyMMdd}/${file:name}
```

34.8. 关于 MOVEFAILED

moveFailed 选项允许您将无法成功处理的文件移到其他位置，如您选择的错误文件夹。例如，要移动具有时间戳的错误文件夹中的文件，您可以使用 moveFailed=/error/${file:noext }-${date:now:yyyyMMddHHmmssSSS}.${\'\'file:ext}。

34.9. 消息标头

此组件支持以下标头：

34.9.1. 仅文件制作者

标头	描述
CamelFileName	指定要写入的文件名（相对于端点目录）。此名称可以是 **String** ；一个字符串，带有 File 语言或 Simple 语言表达式；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	已写入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，其目的是向最终用户提供写入的文件的名称。
CamelOverruleFileName	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。

34.9.2. 仅文件消费者

标头	描述
CamelFileName	使用的文件的名称作为相对文件路径，其偏移来自端点上配置的起始目录。
CamelFileNameOnly	仅文件名（没有前导路径的名称）。
CamelFileAbsolute	**布尔值** 选项指定消耗的文件是否表示绝对路径。对于相对路径，通常应为 **false**。通常不应该使用绝对路径，但我们添加到 move 选项中，以允许将文件移动到绝对路径。但是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。
CamelFileRelativePath	相对路径。
CamelFileParent	父路径。
CamelFileLength	包含文件大小 **的长** 值。

标头	描述
CamelFileLastModified	包含文件最后一次修改时间戳的 长 值。

34.10. BATCH CONSUMER

这个组件实现了 Batch Consumer。

34.11. 交换属性，仅限文件消费者

由于文件消费者实施 BatchConsumer，它支持对它轮询的文件进行批处理。通过批处理，我们意味着 Camel 会将以下额外属性添加到交换中，因此您知道轮询的文件数量、当前索引以及批处理是否已完成。

属性	描述
CamelBatchSize	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。对于最后一个条目，仅适用于 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 聚合了这个文件数。

34.12. 使用 CHARSET

charset 选项允许在使用者和制作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：

```
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，但我们希望以 iso-8859-1 格式将文件内容转换为字节数组。然后，让 bean 处理数据。在使用当前 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了 charset，则 Camel 不知道文件的 charset，默认情况下将使用 "UTF-8"。但是，您可以配置 JVM 系统属性，以覆盖并使用键 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，这可能会有问题，这是读取文件的默认编码。在编写文件时，内容已转换为字节数组，因此将内容直接写入原样（没有任何进一步编码）。

```
from("file:inbox")
  .convertBodyTo(byte[].class, "iso-8859-1")
  .to("bean:myBean")
  .to("file:outbox");
```

您还可以在编写文件时覆盖和控制编码动态，方法是使用密钥 Exchange.CHARSET_NAME 在交换上设置属性。例如，在下面的路由中，我们使用消息标头中的值设置属性。

```
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以特定编码写入文件，则优先在端点上使用 charset 选项。

请注意，如果您明确在端点上配置了 charset 选项，则会使用该配置，而不考虑 Exchange.CHARSET_NAME 属性。

如果您有一些问题，那么您可以在 org.apache.camel.component.file 上启用 DEBUG 日志记录，在使用特定 charset 读取/写入文件时 Camel 日志。
例如，以下路由将记录以下内容：

```
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
```

34.13. 常用带有文件夹和文件名的 GETCHAS

当 Camel 生成文件（写文件）时，有几个影响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，由于消息 ID 通常是唯一生成的 ID，因此您会以文件名（如 ID-MACHINENAME-2443-1211718892437-1-0 ）结束。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。也可以使用恒定的 Exchange.FILE_NAME。

以下代码示例使用消息 ID 作为文件名生成文件：

```
from("direct:report").to("file:target/reports");
```

使用 report.txt 作为您需要执行的文件名：

```
from("direct:report").setHeader(Exchange.FILE_NAME, constant("report.txt")).to(
"file:target/reports");
```

- 与以上内容相同，但使用 CamelFileName ：

```
from("direct:report").setHeader("CamelFileName", constant("report.txt")).to(
"file:target/reports");
```

和语法，其中使用 fileName URI 选项在端点上设置文件名。

```
from("direct:report").to("file:target/reports/?fileName=report.txt");
```

34.14. 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 File 语言表达式进行设置。如需语法和样本，请参阅 文件 语言。

34.15. 从其他人直接丢弃文件的文件夹中消耗文件

请注意，如果您消耗来自其他应用程序直接写入文件的文件夹。查看不同的 readLock 选项以查看适合您的用例。但是，最佳方法是写入另一个文件夹，并在 drop 文件夹中的文件移动之后。但是，如果您直接将文件写入 drop 文件夹，则选项更改可以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文件大小/修改在一段时间内的更改。其他 readLock 选项依赖于 Java File API，在检测这一点时并不总是非常好。您可能还想查看 doneFileName 选项，该选项使用标志文件（撤消文件）在文件完成时发出信号并准备好被使用。

34.16. 使用 DONE 文件

另请参阅在下面 *写入 done 文件* 的部分。

如果您只想在文件存在时消耗文件，则可以在端点上使用 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 不会删除 *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 关联。为此，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。消费者只支持 *done 文件名* 的静态部分作为前缀或后缀（不能同时支持两者）。

```
from("file:bar?doneFileName=${file:name}.done");
```

如果存在名称文件名为 .done 的文件，则仅轮询 *本例中的文件*。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

34.17. 编写完成的文件

编写了一个文件后，您可能希望将其他 *done文件* 编写为标记类型，以指示其他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 创建名为 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 关联。为此，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。

34.18. SAMPLES

34.18.1. 从目录读取并写入另一个目录

```
from("file://inputdir/?delete=true").to("file://outputdir")
```

34.18.2. 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}")
```

侦听目录并为丢弃的每个文件创建一个消息。将内容复制到 outputdir，并删除 inputdir 中的 文件。

34.18.3. 递归读取目录并写入另一个

```
from("file://inputdir/?recursive=true&delete=true").to("file://outputdir")
```

侦听目录并为丢弃的每个文件创建一个消息。将内容复制到 outputdir，并删除 inputdir 中的 文件。将递归扫描到子目录。会将文件放置在 outputdir 中与 inputdir 相同的目录结构，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
```

34.19. 使用 FLATTEN

如果要将文件存储在 outputdir 目录中，忽略源目录布局（例如，扁平化路径），请在生成者端添加 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
```

34.20. 从目录和默认移动操作中读取

默认情况下，Camel 将将所有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：
之前

```
inputdir/foo.txt
inputdir/sub/bar.txt
```

after

```
inputdir/.camel/foo.txt
inputdir/sub/.camel/bar.txt
outputdir/foo.txt
outputdir/sub/bar.txt
```

34.21. 从目录读取，并在 JAVA 中处理消息

```
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。

34.22. 写入文件

Camel 还能够写入文件，即生成文件。在以下示例中，我们收到在将 SEDA 队列写入目录之前所处理的 SEDA 队列的一些报告。

34.22.1. 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。

34.22.2. 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，文件将写入 /var/myapp/filesInProgress 目录，在进行数据传输后，它们将会被原子移到 'var/myapp/finalDirectory 'directory 中。

```
from("direct:start").
  to("file:///var/myapp/finalDirectory?tempPrefix=/../filesInProgress/");
```

34.23. 对文件名使用表达式

在这个示例中，我们想将消耗的文件移动到备份文件夹中，并将当前日期用作子文件夹名称：

```
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。

34.24. 避免多次读取同一文件(IDEMPOTENT 消费者)

Camel 直接支持组件内的 Idempotent Consumer，以便跳过已经处理的文件。可以通过设置 idempotent=true 选项来启用此功能。

```
from("file://inbox?idempotent=true").to("...");
```

Camel 使用绝对文件名作为幂等密钥，以检测重复的文件。您可以使用 idempotentKey 选项中的表达式来自定义此密钥。例如，要将名称和文件大小用作密钥

```
<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，它使用最早使用的缓存，最多 1000 个条目。您可以使用值中的 # 符号指示 它引用 带有指定 id 的 Registry 中的 bean，以自行编写此存储实现。

```
 <!-- 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 Camel 跳过了一个文件，则 Camel 会在 DEBUG 级别进行日志，因为它已被使用：

```
DEBUG FileConsumer is idempotent and the file has been consumed before. Will skip this
file: target\idempotent\report.txt
```

34.25. 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，而不是基于默认值的内存。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存活。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。当文件增大较大的 Camel 将截断文件存储时，通过将第一级缓存刷新到新的空文件来重建内容。

我们使用 Spring XML 创建文件幂等存储库来配置我们的存储库，并使用 # 符号定义我们的具有 幂等存储库的存储库 来指示 Registry 查找：

34.26. 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，而不是根据默认值使用的内存。

首先，我们需要 META-INF/persistence.xml 中的 persistence-unit，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，我们也可以在 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 # syntax 选项使用 idempotentRepository 在文件消费者端点中引用 jpaStore bean：

```
<route>
  <from uri="file://inbox?idempotent=true&amp;idempotentRepository=#jpaStore"/>
  <to uri="bean:processInbox"/>
</route>
```

34.27. 使用 ORG.APACHE.CAMEL.COMPONENT.FILE.GENERICFILEFILTER 过滤

Camel 支持可插拔过滤策略。然后，您可以使用此类过滤器配置端点，以跳过正在处理的某些文件。

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 属性配置路由来引用我们在 spring XML 文件中定义的过滤器（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>
```

34.28. 使用 ANT 路径匹配程序进行过滤

ANT 路径匹配器基于 AntPathMatcher。

文件路径与以下规则匹配：

- ? 匹配一个字符

- * 匹配零个或更多字符
- | 匹配路径中的零个或更多目录

antInclude 和 antExclude 选项可以轻松地指定 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。

注意

当将 minDepth/maxDepth 与 recursive=true 组合结合使用时，tExclude=... 和 readLockDeleteOrphanLockFiles=true 会导致扫描所有文件/subfolders 深度，超过 maxDepth 中提到的值。解决办法是配置 readLockDeleteOrphanLockFiles=false。

34.28.1. 使用 Comparator 排序

Camel 支持可插拔排序策略。此策略使用 Java 中的 java.util.Comparator 中的构建。然后，您可以使用此类比较器配置端点，并在处理前 Camel 对文件进行排序。

在示例中，我们构建了自己的比较器，其按文件名排序：

然后，我们可以使用 sorter 选项配置我们的路由来引用我们排序器(我的排序器)，我们在 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>
```

注意

URI 选项可以使用 # 语法来引用 bean
在 Spring DSL 路由中，可以通过在 id 前使用 # 前缀来引用 Registry 中的 beans。因此，编写排序器 =#mySorter，将指示 Camel 来查找 ID 为 mySorter 的 bean 的 Registry。

34.28.2. 使用 sortBy 排序

Camel 支持可插拔排序策略。此策略使用 File 语言来配置排序。sortBy 选项配置如下：

```
sortBy=group 1;group 2;group 3;...
```

其中每个组都用分号隔开。在只使用一个组的简单情况下，一个简单的示例可以是：

```
sortBy=file:name
```

这将按文件名排序，您可以通过向组添加前缀 reverse: 来反向对顺序进行反转，因此排序现在是 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
```

您可以配置来忽略大小写，使用 ignoreCase: 进行字符串比较，因此如果您想要使用文件名排序，但要忽略大小，然后我们这样做：

```
sortBy=ignoreCase:file:name
```

您可以组合忽略问题单和反向，但必须首先指定反向：

```
sortBy=reverse:ignoreCase:file:name
```

在以下示例中，我们想要根据最后修改的文件排序：

```
sortBy=file:modified
```

然后，我们希望按名称对名称进行分组，因此具有相同 modifcation 的文件按名称排序：

```
sortBy=file:modified;file:name
```

现在有一个问题，您可以发现它吗？文件的修改时间戳太大，因为它将以毫秒为单位，但如果我们只希望按日期排序，然后按名称排列子组？
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。因此，这可以解决：

```
sortBy=date:file:yyyyMMdd;file:name
```

Yeah 是非常强大的，通过您可以为每个组群使用反向的方式 oh，因此我们可以撤销文件名：

```
sortBy=date:file:yyyyMMdd;reverse:file:name
```

34.29. 使用 GENERICFILEPROCESSSTRATEGY

选项 processStrategy 可用于使用自定义 GenericFileProcessStrategy，它允许您实施您自己的 *开始、提交 和回滚* 逻辑。
例如，可以假定系统在文件夹中写入文件，您应该使用。但是，在另一个 *就绪* 文件被写入前，您不应该开始使用该文件。

通过实施自己的 GenericFileProcessStrategy，我们可以将其实现：

- 在 begin （） 方法中，我们可以测试是否存在特殊的 *就绪* 文件。begin 方法返回一个 布尔值，以指示我们是否可以消耗该文件。
- 在 abort() 方法中，当 begin 操作返回 false 时，可以执行特殊的逻辑，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 *就绪* 的文件。

34.30. 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。这个接口有一个 accept 方法返回布尔值。返回 true 使其包含该文件，false 可跳过该文件。GenericFile 上有一个 Directory 方法，无论文件是否为目录。这可让您过滤不需要的目录，以避免中断不需要的目录。

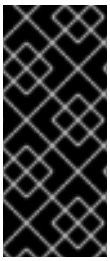
例如，要跳过名称中以 "skip" 开头的任何目录，可按如下方式实施：

34.31. 使用 BRIDGEERRORHANDLER

如果要使用 Camel Error Handler 处理文件消费者中的异常，您可以启用 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?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

当使用 bridgeErrorHandler 时，当使用 bridgeErrorHandler 时，拦截器 不适用。Exchange 由 Camel Error Handler 直接处理，不允许之前操作，如拦截器、完成操作。

34.32. 调试日志记录

此组件具有日志级别 TRACE，在遇到问题时很有用。

34.33. 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.cluster.file.acquire-lock-delay	开始尝试获取锁定前等待的时间。		字符串

Name	描述	默认值	类型
camel.cluster.file.acquire-lock-interval	尝试获取锁定之间的等待时间。		字符串
camel.cluster.file.attributes	自定义服务属性。		Map
camel.cluster.file.enabled	设定是否应启用文件集群服务，默认为 false。	false	布尔值
camel.cluster.file.id	集群服务 ID。		字符串
camel.cluster.file.order	服务查找顺序/优先级.		整数
camel.cluster.file.root	root 路径。		字符串
camel.component.file.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.file.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.file.enabled	是否启用文件组件的自动配置。这默认是启用的。		布尔值
camel.component.file.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

第 35 章 文件语言

文件表达式语言是语言的扩展，可添加与文件相关的功能。这些功能与使用文件路径和名称的常见用例相关。目标是允许表达式用于

用于为消费者和制作者设置动态文件模式的组件。

注意

文件语言与语言合并，这意味着您可以直接使用简单语言中的所有文件语法。

35.1. 依赖项

文件语言是 camel-core 的一部分。

当在 Red Hat build of Camel Spring Boot 中使用文件时，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-core-starter</artifactId>
</dependency>
```

35.2. 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		布尔值	是否修剪值以移除前导和结尾的空格和换行符。

35.3. 语法

这个语言是对语言的扩展，因此也会应用语法。下表仅列出其他与文件相关的功能。

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 ：absolute 指的是 java.io.File.getAbsolute （） 方法。请注意，当前 Exchange 不支持所有表达式。例如，组件支持一些选项，而 File 组件支持所有选项。

èi¨è¾¾å¼	类型	File Consumer	文件 Producer	FTP Consumer	FTP Producer	描述
file:name	字符串	是	否	是	否	文件名（相对于起始目录，请参见以下注释）
file:name.ext	字符串	是	否	是	否	仅引用文件扩展
file:name.ext.single	字符串	是	否	是	否	指的是文件扩展名。如果文件扩展有多个点，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	引用不带扩展名的文件名（相对于起始目录，请参阅下面的备注）
file:name.noext.single	字符串	是	否	是	否	引用不带扩展名的文件名（相对于起始目录，请参阅下面的备注）。如果文件扩展有多个点，则此表达式仅剥离最后一个部分，并保留其他部分。
file:onlyname	字符串	是	否	是	否	仅使用没有前置路径的文件名。
file:onlyname.noext	字符串	是	否	是	否	仅在没有扩展的情况下引用文件名，且没有前导路径。
file:onlyname.noext.single	字符串	是	否	是	否	仅在没有扩展的情况下引用文件名，且没有前导路径。如果文件扩展有多个点，则此表达式仅剥离最后一个部分，并保留其他部分。
file:ext	字符串	是	否	是	否	仅引用文件扩展

èi¨è¾¾å¼	类型	File Consumer	文件 Producer	FTP Consumer	FTP Producer	描述
file:parent	字符串	是	否	是	否	引用文件父项
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
file:absolute.path	字符串	是	否	否	否	指的是绝对文件路径
file:length	Long	是	否	是	否	引用作为 Long 类型返回的文件长度
file:size	Long	是	否	是	否	引用作为 Long 类型返回的文件长度
file:modified	Date	是	否	是	否	引用上次修改的文件作为日期类型
date:_command:pattern_	字符串	是	是	是	是	用于使用 **java.text.SimpleDateFormat** 模式的日期格式。是语言的扩展。**其他命令是：文件（仅限消费者）用于文件的最后修改的时间戳。注意：也可以使用语言中的所有命令。**

35.4. 文件令牌示例

35.4.1. 相对路径

在以下 相对 目录中，我们有一个适用于 hello.txt 文件的 java.io.File 处理： .\filelanguage\test。我们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
file:onlyname.noext	hello
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

35.4.2. 绝对路径

我们在以下 绝对 目录中有一个 java.io.File 处理文件 hello.txt ： \workspace\camel\camel-core\target\filelanguage\test。我们将外部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	hello
file:ext	txt
file:parent	\workspace\camel\camel-core\target\filelanguage\test

èi¨è¾¾å¼	返回
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

35.5. SAMPLES

您可以输入固定的文件名，如 myfile.txt ：

```
fileName="myfile.txt"
```

假设我们假设使用文件使用者读取文件，并希望将读取文件移动到将当前日期作为子文件夹的备份文件夹。这可以通过类似如下的表达式完成：

```
fileName="backup/${date:now:yyyyMMdd}/${file:name.noext}.bak"
```

还要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"
```

由于这是我们可从此语言访问所有好的语言的扩展，因此在这种用例中，我们希望使用 in.header.type 作为动态表达式中的参数：

```
fileName="../backup/${date:now:yyyyMMdd}/type-${in.header.type}/backup-of-${file:name.noext}.bak"
```

如果您在表达式中有一个自定义日期，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，该类生成要使用的字符串输出（或可转换为 String）：

```
fileName="uniquefile-${bean:myguidgenerator.generateid}.txt"
```

-

当然，所有这些都可以在一个表达式中合并，您可以使用一个组合表达式中的 和 语言。这对常见的文件路径模式非常强大。

35.6. SPRING BOOT AUTO-CONFIGURATION

组件支持 147 选项，如下所列。

Name	描述	默认值	类型
camel.cloud.consul.service-discovery.acl-token	设置用于 Consul 的 ACL 令牌。		字符串
camel.cloud.consul.service-discovery.block-seconds	等待监视事件的秒数，默认为 10 秒。	10	整数
camel.cloud.consul.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.consul.service-discovery.connect-timeout-millis	OkHttpClient 的连接超时。		Long
camel.cloud.consul.service-discovery.datacenter	数据中心。		字符串
camel.cloud.consul.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.consul.service-discovery.password	设置用于基本身份验证的密码。		字符串

Name	描述	默认值	类型
camel.cloud.consul.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.consul.service-discovery.read-timeout-millis	OkHttpClient 的读取超时。		Long
camel.cloud.consul.service-discovery.url	Consul 代理 URL。		字符串
camel.cloud.consul.service-discovery.user-name	设置用于基本身份验证的用户名。		字符串
camel.cloud.consul.service-discovery.write-timeout-millis	OkHttpClient 的写入超时。		Long
camel.cloud.dns.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.dns.service-discovery.domain	域名；		字符串
camel.cloud.dns.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.dns.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.dns.service-discovery.proto	所需服务的传输协议。	_tcp	字符串

Name	描述	默认值	类型
camel.cloud.etcd.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.etcd.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.etcd.service-discovery.password	用于基本身份验证的密码。		字符串
camel.cloud.etcd.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.etcd.service-discovery.service-path	查找服务发现的路径。	/services/	字符串
camel.cloud.etcd.service-discovery.timeout	要设置操作可以采取的最长时间，请执行以下操作：		Long
camel.cloud.etcd.service-discovery.type	要设置发现类型，有效值为 on-demand 和 watch。	按需	字符串
camel.cloud.etcd.service-discovery.uris	客户端可以连接到的 URI。		字符串
camel.cloud.etcd.service-discovery.user-name	用于基本身份验证的用户名。		字符串
camel.cloud.kubernetes.service-discovery.api-version	使用客户端查找时设置 API 版本。		字符串

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.ca-cert-data	使用客户端查找时设置证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.ca-cert-file	在使用客户端查找时，设置从文件加载的证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-data	使用客户端查找时设置客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-file	在使用客户端查找时，设置从文件加载的客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-algo	设置客户端密钥存储算法，如使用客户端查找时 RSA。		字符串
camel.cloud.kubernetes.service-discovery.client-key-data	使用客户端查找时设置客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-file	在使用客户端查找时，设置从文件加载的客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-passphrase	使用客户端查找时设置客户端密钥存储密码短语。		字符串
camel.cloud.kubernetes.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.kubernetes.service-discovery.dns-domain	设置用于 DNS 查找的 DNS 域。		字符串

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.kubernetes.service-discovery.lookup	如何执行服务查找。可能的值有：client、dns、environment。在使用客户端时，客户端会查询 kubernetes master 来获取提供该服务的活跃 pod 列表，然后随机（或循环）选择一个 pod。当使用 dns 时，服务名称被解析为 name.namespace.svc.dnsDomain。当使用 dnssrv 时，服务名称使用 SRV 查询解析 ....svc... when using environment，环境变量用于查找服务。默认情况下使用环境。	环境	字符串
camel.cloud.kubernetes.service-discovery.master-url	在使用客户端查找时，将 URL 设置为 master。		字符串
camel.cloud.kubernetes.service-discovery.namespace	设置要使用的命名空间。默认情况下，将使用来自 ENV 变量 KUBERNETES_MASTER 的命名空间。		字符串
camel.cloud.kubernetes.service-discovery.oauth-token	在使用客户端查找时，为身份验证设置 OAUTH 令牌（而不是用户名/密码）。		字符串
camel.cloud.kubernetes.service-discovery.password	在使用客户端查找时设置用于身份验证的密码。		字符串
camel.cloud.kubernetes.service-discovery.port-name	设置用于 DNS/DNSSRV 查找的端口名称。		字符串
camel.cloud.kubernetes.service-discovery.port-protocol	设置用于 DNS/DNSSRV 查找的端口协议。		字符串
camel.cloud.kubernetes.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.trust-certs	设置在使用客户端查找时是否打开信任证书检查。	false	布尔值
camel.cloud.kubernetes.service-discovery.username	在使用客户端查找时设置用于身份验证的用户名。		字符串
camel.cloud.ribbon.load-balancer.client-name	设置 Ribbon 客户端名称。		字符串
camel.cloud.ribbon.load-balancer.configurations	定义其他配置定义。		Map
camel.cloud.ribbon.load-balancer.enabled	启用组件。	true	布尔值
camel.cloud.ribbon.load-balancer.namespace	命名空间。		字符串
camel.cloud.ribbon.load-balancer.password	密码。		字符串
camel.cloud.ribbon.load-balancer.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.ribbon.load-balancer.username	用户名。		字符串

Name	描述	默认值	类型
camel.hystrix.allow-maximum-size-to-diverge-from-core-size	允许配置使 maximumSize 生效。然后该值可以等于或大于 coreSize。	false	布尔值
camel.hystrix.circuit-breaker-enabled	是否使用 HystrixCircuitBreaker。如果为 false，则不会使用 断路器逻辑，并且所有允许的请求。这与 circuitBreakerForceClosed （）的影响类似，除非继续跟踪指标，知道它是否应该是 open/closed，此属性即使实例化一个断路器。	true	布尔值
camel.hystrix.circuit-breaker-error-threshold-percentage	错误百分比阈值（如 50 ）指向断路器将打开和拒绝请求。它将在 circuitBreakerSleepWindowInMilliseconds 中定义的持续时间保持出差；与 HystrixCommandMetrics.getHealthCounts （）进行比较的错误百分比。	50	整数
camel.hystrix.circuit-breaker-force-closed	如果为 true，HystrixCircuitBreaker#allowRequest （）将始终返回 true 以允许请求，无论 HystrixCommandMetrics.getHealthCounts （）的错误百分比如何。如果设为 true，则 circuitBreakerForceOpen （）属性具有优先权。	false	布尔值
camel.hystrix.circuit-breaker-force-open	如果为 true，HystrixCircuitBreaker.allowRequest （）将始终返回 false，从而导致电路变为开路（接受），并拒绝所有请求。此属性优先于 circuitBreakerForceClosed （）;。	false	布尔值
camel.hystrix.circuit-breaker-request-volume-threshold	metricsRollingStatisticalWindowInMilliseconds （）中的最少请求数必须存在于 HystrixCircuitBreaker 之前。如果此数字低于这个数字，无论错误百分比如何，电路都不会被出差。	20	整数
camel.hystrix.circuit-breaker-sleep-window-in-milliseconds	HystrixCircuitBreaker trips 之后的时间（以毫秒为单位），它应该在尝试请求前等待。	5000	整数
camel.hystrix.configurations	定义其他配置定义。		Map
camel.hystrix.core-pool-size	传递给 java.util.concurrent.ThreadPoolExecutor#setCorePoolSize (int)的核心 thread-pool 大小。	10	整数
camel.hystrix.enabled	启用组件。	true	布尔值

Name	描述	默认值	类型
camel.hystrix.execution-isolation-semaphore-max-concurrent-requests	允许 HystrixCommand.run （）的并发请求数。超过并发限制的请求将被拒绝。仅在执行IsolationStrategy == SEMAPHORE 时使用。	20	整数
camel.hystrix.execution-isolation-strategy	将通过什么隔离策略 HystrixCommand.run （）执行。如果 THREAD，它将在单独的线程上执行，并且受 thread-pool 中的线程数量限制的并发请求。如果 SEMAPHORE，它将在调用线程上执行，并且受 semaphore 数限制的并发请求。	线程	字符串
camel.hystrix.execution-isolation-thread-interrupt-on-timeout	当线程超时时，执行线程是否应该尝试中断（使用 future#cancel）。仅在执行IsolationStrategy （）== THREAD 时才适用。	true	布尔值
camel.hystrix.execution-timeout-enabled	此命令是否启用了超时机制。	true	布尔值
camel.hystrix.execution-timeout-in-milliseconds	以毫秒为单位，将命令超时和停止执行的时间（以毫秒为单位）。如果 executionIsolationThreadInterruptOnTimeout == true 且命令是线程隔离，则执行线程将中断。如果命令是 semaphore-isolated 和 HystrixObservableCommand，则该命令将被取消订阅。	1000	整数
camel.hystrix.fallback-enabled	出现故障时，是否应尝试 HystrixCommand.getFallback （）。	true	布尔值
camel.hystrix.fallback-isolation-semaphore-max-concurrent-requests	允许 HystrixCommand.getFallback （）的并发请求数。超过并发限制的请求将快速失败，且不会尝试检索回退。	10	整数
camel.hystrix.group-key	设置要使用的 group 键。默认值为 CamelHystrix。	Camel Hystrix	字符串
camel.hystrix.keep-alive-time	更长的时间（以分钟为单位）传递给 ThreadPoolExecutor#setKeepAliveTime (long,TimeUnit)。	1	整数

Name	描述	默认值	类型
camel.hystrix.max-queue-size	在 HystrixConcurrencyStrategy.getBlockingQueue (int)中传递给 BlockingQueue 的最大队列大小应该只影响 threadpool 的实例化 - 它不会立即更改队列大小。为此，请使用 queueSizeRejectionThreshold （）。	-1	整数
camel.hystrix.maximum-size	传递给 ThreadPoolExecutor#setMaximumPoolSize (int)的最大 thread-pool 大小。这是可在不开始拒绝 HystrixCommands 的情况下支持的最大并发数量。请注意，只有在您也设置了 allowMaximumSizeToDivergeFromCoreSize 时，此设置才会生效。	10	整数
camel.hystrix.metrics-health-snapshot-interval-in-milliseconds	在允许计算成功和错误百分比时等待的时间（以毫秒为单位），并影响 HystrixCircuitBreaker.isOpen （）状态。在高容量电路上，错误百分比的连续计算可能会成为 CPU 密集型，从而控制其计算的频率。	500	整数
camel.hystrix.metrics-rolling-percentile-bucket-size	滚动百分比的每个存储桶中存储的最大值数。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	10	整数
camel.hystrix.metrics-rolling-percentile-enabled	是否应该使用 HystrixRollingPercentile 内部 HystrixCommandMetrics 来捕获百分比的指标。	true	布尔值
camel.hystrix.metrics-rolling-percentile-window-buckets	滚动窗口的存储桶数量被分成。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	6	整数
camel.hystrix.metrics-rolling-percentile-window-in-milliseconds	以毫秒为单位的滚动窗口的持续时间。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	10000	整数
camel.hystrix.metrics-rolling-statistical-window-buckets	滚动统计窗口划分为的 bucket 数量。这在 HystrixCommandMetrics 中被传递给 HystrixRollingNumber。	10	整数

Name	描述	默认值	类型
camel.hystrix.metrics-rolling-statistical-window-in-milliseconds	此属性设置统计滚动窗口的持续时间，以毫秒为单位。这是为线程池保留指标的时间。窗口被分成 bucket，按这些增量回滚。	10000	整数
camel.hystrix.queue-size-rejection-threshold	队列大小拒绝阈值是 artificial max size，即使尚未达到 maxQueueSize，也会发生拒绝。这是因为 BlockingQueue 的 maxQueueSize 无法动态更改，我们希望动态更改影响拒绝的队列大小。在排队线程以进行执行时，HystrixCommand 会使用它。	5	整数
camel.hystrix.request-log-enabled	HystrixCommand 执行和事件是否应记录到 HystrixRequestLog。	true	布尔值
camel.hystrix.thread-pool-key	设置要使用的线程池密钥。默认情况下，将使用与 groupKey 配置相同的值。	Camel Hystrix	字符串
camel.hystrix.thread-pool-rolling-number-statistical-window-buckets	滚动统计窗口划分为的 bucket 数量。这会传递到每个 HystrixThreadPoolMetrics 实例内的 HystrixRollingNumber。	10	整数
camel.hystrix.thread-pool-rolling-number-statistical-window-in-milliseconds	统计滚动窗口的持续时间（以毫秒为单位）。这会传递到每个 HystrixThreadPoolMetrics 实例内的 HystrixRollingNumber。	10000	整数
camel.language.constant.enabled	是否启用恒定语言的自动配置。这默认是启用的。		布尔值
camel.language.constant.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.csimple.enabled	是否启用 csimple 语言的自动配置。这默认是启用的。		布尔值
camel.language.csimple.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.exchangeproperty.enabled	是否启用 exchangeProperty 语言的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.language.exchangeproperty.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.file.enabled	是否启用文件语言的自动配置。这默认是启用的。		布尔值
camel.language.file.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.header.enabled	是否启用标头语言的自动配置。这默认是启用的。		布尔值
camel.language.header.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.ref.enabled	是否启用 ref 语言的自动配置。这默认是启用的。		布尔值
camel.language.ref.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.simple.enabled	是否启用简单语言的自动配置。这默认是启用的。		布尔值
camel.language.simple.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.tokenize.enabled	是否启用令牌化语言的自动配置。这默认是启用的。		布尔值
camel.language.tokenize.group-delimiter	设置在分组时要使用的分隔符。如果没有设置，则令牌将用作分隔符。		字符串
camel.language.tokenize.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.resilience4j.automatic-transition-from-open-to-half-open-enabled	在通过 waitDurationInOpenState 后，启用从 OPEN 自动过渡到 HALF_OPEN 状态。	false	布尔值

Name	描述	默认值	类型
camel.resilience4j.circuit-breaker-ref	代表现有的 io.github.resilience4j.circuitbreaker.CircuitBreaker 实例从 registry 中查找和使用。使用此选项时，不使用任何其他断路器选项。		字符串
camel.resilience4j.config-ref	指的是现有的 io.github.resilience4j.circuitbreaker.CircuitBreakerConfig 实例，以便从 registry 中查找和使用。		字符串
camel.resilience4j.configurations	定义其他配置定义。		Map
camel.resilience4j.enabled	启用组件。	true	布尔值
camel.resilience4j.failure-rate-threshold	以百分比为单位配置故障率阈值。如果失败率相等或大于阈值，则 CircuitBreaker 过渡到 open，并启动短路调用。阈值必须大于 0，而不是大于 100。默认值为 50 百分比。		æµ®ç‚¹å€¼
camel.resilience4j.minimum-number-of-calls	在 CircuitBreaker 可以计算错误率之前，配置所需的最少调用数（每个滑动期限）。例如，如果 minimumNumberOfCalls 为 10，则必须至少记录 10 个调用，然后才能计算失败率。如果只记录了 9 个调用，则 CircuitBreaker 不会过渡到 open，即使所有 9 调用都失败。默认 minimumNumberOfCalls 为 100。	100	整数
camel.resilience4j.permitted-number-of-calls-in-half-open-state	配置 CircuitBreaker 为一半打开时允许的调用数量。大小必须大于 0。默认大小为 10。	10	整数
camel.resilience4j.sliding-window-size	配置滑动窗口的大小，该窗口用于在 CircuitBreaker 关闭时记录调用的结果。slidingWindowSize 配置滑动窗口的大小。滑动窗口可以是基于计数或基于时间的窗口。如果 slidingWindowType 是 COUNT_BASED，则最后的 slidingWindowSize 调用会被记录并聚合。如果 slidingWindowType 是 TIME_BASED，则最后 slidingWindowSize 秒的调用会被记录并聚合。slidingWindowSize 必须大于 0。minimumNumberOfCalls 必须大于 0。如果 slidingWindowType 是 COUNT_BASED，则 minimumNumberOfCalls 不能大于 slidingWindowSize。如果 slidingWindowType 是 TIME_BASED，您可以选择任何您需要的。默认 slidingWindowSize 为 100。	100	整数

Name	描述	默认值	类型
camel.resilience4j.sliding-window-type	配置滑动窗口的类型，用于记录 CircuitBreaker 关闭时调用的结果。滑动窗口可以是基于计数或基于时间的窗口。如果 slidingWindowType 是 COUNT_BASED，则最后的 slidingWindowSize 调用会被记录并聚合。如果 slidingWindowType 是 TIME_BASED，则最后 slidingWindowSize 秒的调用会被记录并聚合。默认 slidingWindowType 是 COUNT_BASED。	COUNT_BASED	字符串
camel.resilience4j.slow-call-duration-threshold	配置上面的持续时间阈值（秒），调用被视为缓慢，并增加较慢的调用百分比。默认值为 60 秒。	60	整数
camel.resilience4j.slow-call-rate-threshold	以百分比为单位配置阈值。当调用持续时间大于 slowCallDurationThreshold Duration 时，CircuitBreaker 会将调用视为较慢。当较慢的调用百分比相等或大于阈值时，CircuitBreaker 过渡到 open，并启动短路调用。阈值必须大于 0，而不是大于 100。默认值为 100 百分比，这意味着所有记录的调用都必须比 slowCallDurationThreshold 慢。		æµ®ç‚¹â€¼
camel.resilience4j.wait-duration-in-open-state	配置等待持续时间（以秒为单位），指定 CircuitBreaker 应该保持打开的时间，然后再切换到半次。默认值为 60 秒。	60	整数
camel.resilience4j.writable-stack-trace-enabled	启用可写入堆栈跟踪。当设置为 false 时，Exception.getStackTrace 返回一个零长度数组。当断路器处于开路状态时，这可用于减少日志垃圾邮件，因为存在例外的原因（断路器是短路调用）。	true	布尔值
camel.rest.api-component	用作 REST API 的 Camel 组件名称（如 swagger）如果没有明确配置 API 组件，则 Camel 会查找负责服务并生成 REST API 文档的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestApiProcessorFactory。如果找到其中任何一个，则使用它。		字符串
camel.rest.api-context-path	设置领导的 API 上下文路径将使用的 REST API 服务。这在使用使用 context-path 部署 Web 应用程序的组件（如 camel-servlet ）时使用它。		字符串
camel.rest.api-context-route-id	设置用于服务 REST API 的路由的路由 ID。默认情况下，路由将使用自动分配的路由 ID。		字符串
camel.rest.api-host	要将特定主机名用于 API 文档（如 swagger），这可用于用这个配置的主机名覆盖生成的主机。		字符串

Name	描述	默认值	类型
camel.rest.api-property	允许为 api 文档配置任意数量的附加属性(swagger)。例如，将属性 api.title 设置为我的冷却。		Map
camel.rest.api-vendor-extension	是否在 Rest API 中启用供应商扩展。如果启用，Camel 将包含额外信息作为厂商扩展名（例如，以 x- 开头的键），如路由 ID、类名称等。在导入 API 文档时，并非所有第三方 API 网关和工具都支持 vendor-extensions。	false	布尔值
camel.rest.binding-mode	设置要使用的绑定模式。默认值为 off。		RestBindingMode
camel.rest.client-request-validation	是否启用客户端请求验证，以检查客户端的 Content-Type 和 Accept 标头是否受到其 consume/produces 设置的 Rest-DSL 配置的支持。这可以打开，以启用此检查。如果验证错误，则返回 HTTP Status code 415 或 406。默认值为 false。	false	布尔值
camel.rest.component	用于 REST 传输(consumer)的 Camel Rest 组件，如 netty-http, jetty, servlet, undertow。如果没有明确配置组件，则 Camel 会查找是否有与 Rest DSL 集成的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestConsumerFactory。如果找到其中任何一个，则使用它。		字符串
camel.rest.component-property	允许为正在使用的其他组件配置任意数量的附加属性。		Map
camel.rest.consumer-property	允许为使用中的其他使用者配置任意数量的附加属性。		Map
camel.rest.context-path	设置 REST 服务将使用的前导上下文路径。这在使用使用 context-path 部署 Web 应用程序的组件（如 camel-servlet ）时使用它。或者对于包含 HTTP 服务器的 camel-jetty 或 camel-netty-http 等组件。		字符串
camel.rest.cors-headers	允许配置自定义 CORS 标头。		Map

Name	描述	默认值	类型
camel.rest.data-format-property	允许为使用的数据格式配置多个额外属性。例如，将属性 prettyPrint 设置为 true，以便以用户友善模式输出 json。属性可以加上前缀来表示选项仅适用于 JSON 或 XML，以及 IN 或 OUT。前缀为： json.in. json.out. xml.in. xml.out。例如，值为 xml.out.mustBeJAXBElement 的键仅用于传出的 XML 数据格式。没有前缀的密钥是所有情况的通用密钥。		Map
camel.rest.enable-cors	是否在 HTTP 响应中启用 CORS 标头。默认值为 false。	false	布尔值
camel.rest.endpoint-property	允许为使用中的其他端点配置多个额外的属性。		Map
camel.rest.host	用于公开 REST 服务的主机名。		字符串
camel.rest.host-name-resolver	如果没有明确配置的主机名，这个 resolver 会用于计算 REST 服务将要使用的主机名。		RestHostNameResolver
camel.rest.json-data-format	要使用的特定 json 数据格式的名称。默认将使用 json-jackson。重要：此选项仅用于设置数据格式的自定义名称，而不是引用现有数据格式实例。		字符串
camel.rest.port	用于公开 REST 服务的主机名。请注意，如果您使用 servlet 组件，则此处配置的端口号不适用，因为使用中的端口号是 servlet 组件使用的实际端口号。例如，如果使用 Apache Tomcat，它的 tomcat http 端口，如果使用 Apache Karaf，它的在 Karaf 中的 HTTP 服务，它默认使用端口 8181。虽然在这些情况下，这里设置端口号，但允许工具和 JMX 知道端口号，因此建议将端口号设置为 servlet 引擎使用的数字。		字符串
camel.rest.producer-api-doc	设置 api 文档的位置，REST 生成者将根据这个文档来验证 REST uri 和查询参数是否有效。这需要将 camel-swagger-java 添加到 classpath 中，任何缺失的配置都会导致 Camel 在启动时失败并报告错误。默认情况下从 classpath 加载的 api 文档的位置，但您可以使用 file: 或 http: 引用从文件或 http url 加载的资源。		字符串
camel.rest.producer-component	设置要用作 REST 生成者的 Camel 组件的名称。		字符串
camel.rest.scheme	用于公开 REST 服务的方案。通常支持 http 或 https。默认值为 http。		字符串

Name	描述	默认值	类型
camel.rest.skip-binding-on-error-code	如果存在自定义 HTTP 错误代码标头，是否跳过输出绑定。这允许构建没有绑定到 json / xml 等自定义错误消息，否则成功信息会这样做。	false	布尔值
camel.rest.use-x-forward-headers	是否将 X-Forward 标头用于主机和相关设置。默认值为 true。	true	布尔值
camel.rest.xml-data-format	要使用的特定 XML 数据格式的名称。默认情况下将使用 jaxb。重要：此选项仅用于设置数据格式的自定义名称，而不是引用现有数据格式实例。		字符串
camel.rest.api-context-id-pattern	**弃用** 设置 CamelContext id 特征，以只允许 CamelContext 中名称与特征匹配的其他服务的 Rest API。特征 name 指的是 CamelContext 名称，仅匹配当前的 CamelContext。对于任何其他值，特征使用来自 PatternHelper#matchPattern (String,String)的规则。		字符串
camel.rest.api-context-listing	**弃用** 设置是否启用了 JVM 中带有 REST 服务的所有可用 CamelContext 的列表。如果启用，它将允许发现这些上下文，如果为 false，则只使用当前的 CamelContext。	false	布尔值

第 36 章 FLINK

从 Camel 2.18 开始

仅支持生成者

本文档页面涵盖了 Apache Camel 的 Flink 组件。camel-flink 组件在 Camel 组件和 Flink 任务之间提供了一个桥接。此组件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。

36.1. 依赖项

当在 Camel Spring Boot 中使用 camel-flink 时，请将以下 Maven 依赖项添加到 pom.xml 中，以支持自动配置：

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.springboot</groupId>
  <artifactId>camel-flink-starter</artifactId>
</dependency>
```

36.2. URI 格式

目前，Flink 组件只支持 Producers。一个可以创建 DataSet、DataStream 作业。

```
flink:dataset?dataset=#myDataSet&dataSetCallback=#dataSetCallback
flink:datastream?datastream=#myDataStream&dataStreamCallback=#dataStreamCallback
```

36.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别

- 端点级别

36.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

36.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

36.4. 组件选项

Flink 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
dataSetCallback (producer)	对 DataSet 执行操作的功能。		DataSetCallback

Name	描述	默认值	类型
datastream (producer)	要计算的数据流。		DataStream
dataStreamCallback (producer)	对 DataStream 执行操作的功能。		DataStreamCallback
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

36.5. 端点选项

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* ：

36.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
endpointType (producer)	端点 **所需的** 类型(dataset、datastream)。 Enum 值： • dataset • datastream		EndpointType

36.5.2. 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)	要计算的数据集。		DataSet
dataSetCallback (producer)	对 DataSet 执行操作的功能。		DataSetCallback
datastream (producer)	要计算的数据流。		DataStream
dataStreamCallback (producer)	对 DataStream 执行操作的功能。		DataStreamCallback
lazyStartProducer (producer (advanced))	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

36.6. 消息标头

Flink 组件支持 4 个消息标头，如下所列：

Name	描述	默认值	类型
CamelFlinkDataSet (producer) 常数 ： FLINK_DATASET_HEADER	dataset。		对象
CamelFlinkDataSetCallback (producer) 常量 ： FLINK_DATASET_CALLBACK_HEADER	dataset 回调。		DataSetCallback

Name	描述	默认值	类型
CamelFlinkDataStream (producer) 常量：FLINK_DATASTREAM_HEADER	数据流。		对象
CamelFlinkDataStreamCallback (producer) 常量：FLINK_DATASTREAM_CALLBACK_HEADER	数据流回调。		DataStreamCallback

36.7. FLINK DATASET 回调

```
@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);
            }
        }
    };
}
```

36.8. 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");
        }
    };
}
```

36.9. 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();
   }
```

36.10. 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.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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。		DataSetCallback
camel.component.flink.data-stream	要计算的数据流。选项是一个 org.apache.flink.streaming.api.datastream.DataStream 类型。		DataStream
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。		DataStreamCallback
camel.component.flink.enabled	是否启用 flink 组件的自动配置。这默认是启用的。		布尔值
camel.component.flink.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

第 37 章 FTP

支持生成者和消费者

这个组件通过 FTP 和 SFTP 协议提供对远程文件系统的访问。

从远程 FTP 服务器消耗时，请确保在更 *消耗文件时阅读标题为 Default* 的部分，以了解与消耗文件相关的详细信息。

不支持 绝对路径。Camel 通过修剪 directoryname 中的所有前斜杠将绝对路径转换为 relative。日志中会显示 WARN 消息。

37.1. 依赖项

当使用带有红帽构建的 Camel Spring Boot 的 ftp 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-ftp-starter</artifactId>
</dependency>
```

37.2. 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。

支持 autoCreate 选项。当消费者启动时，在调度轮询前，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。

如果没有提供 用户名，则尝试使用密码尝试 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。

您可以使用以下格式在 URI 中附加查询选项 ?option=value& amp;option=value&...

此组件为实际 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 （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。

37.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

37.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

37.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

37.4. 组件选项

FTP 组件支持 3 个选项，如下所列。

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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

37.5. 端点选项

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：

37.5.1. 路径参数(3 参数)

Name	描述	默认值	类型
host (common)	**必需** FTP 服务器的主机名。		字符串
port (common)	FTP 服务器的端口。		int
directoryName (common)	起始目录。		字符串

37.5.2. 查询参数(111 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 知道它应在访问文件内容时加载文件内容。在编写文件时，您也可以使用此选项来指定同时写入该文件的 charset。请记住，在编写文件 Camel 时，可能需要将消息内容读到内存中，才能将数据转换为配置的 charset，因此如果您有大量消息，则不要使用它。		字符串
disconnect (common)	使用后是否要与远程 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。如果您有一个要停止的消费者，则需要停止 consumer/route。	false	布尔值
doneFileName (common)	生产者：如果提供，则 Camel 将在写入原始文件时编写完第 2 个文件。完成的文件将为空。这个选项配置要使用的文件名。可以指定固定名称。或者您可以使用动态占位符。完成的文件将始终写在与原始文件相同的文件夹中。消费者：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。可以指定固定名称。或者，您可以使用动态占位符。done 文件始终位于与原始文件相同的文件夹中。仅支持 $\\{file.name} 和 $\\{file.name.next} 作为动态占位符。		字符串

Name	描述	默认值	类型
filename (common)	使用文件语言等表达式动态设置文件名。对于消费者，它用作文件名过滤器。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 Language 来评估它。如果表达式是 Expression 类型，则使用指定的 Expression 类型 - 这允许您，使用 OGNL 表达式。对于消费者，您可以使用它来过滤文件名，因此您可以使用 File Language 语法：mydata-$\\{date:now:yyyyMMdd}.txt，实例消耗了现在的文件。生产者支持 CamelOverruleFileName 标头，它优先于任何现有的 CamelFileName 标头；CamelOverruleFileName 是一个仅使用的标头，因此可以更轻松地避免临时存储 CamelFileName，之后必须恢复它。		字符串
passiveMode (common)	设置被动模式连接。默认为 active 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 Windows 样式路径分隔符 Auto = （默认）在文件名中使用现有路径分隔符. Enum 值： • UNIX • Windows • auto	UNIX	PathSeparator
transferLoggingIntervalSeconds (common)	配置日志记录 in-flight 的上传和下载操作时使用的间隔（以秒为单位）。当操作需要更长的时间时，这用于日志记录进度。	5	int
transferLoggingLevel (common)	配置日志级别，以便在记录上传和下载操作的进度时使用。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	DEBUG	LoggingLevel

Name	描述	默认值	类型
transferLoggingVerbose (common)	配置上传和下载操作进度的执行详细（细粒度）日志记录。	false	布尔值
fastExistsCheck (common (advanced))	如果将此选项设置为 true，则 camel-ftp 将直接使用 list 文件来检查该文件是否存在。由于有些 FTP 服务器可能不支持直接列出文件，如果 选项为 false，则 camel-ftp 将使用旧方法来列出该目录，并检查该文件是否存在。这个选项还影响 readLock=changed 来控制它是否执行快速检查来更新文件信息。如果 FTP 服务器有大量文件，这可用于加快进程速度。	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，则该文件会在成功处理后删除。	false	布尔值
moveFailed (consumer)	根据简单语言设置移动失败表达式。例如，要将文件移动到 .error 子目录，请使用： .error。注意：当将文件移动到故障位置 Camel 将处理错误时，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。这个选项适用于只读数据，或用于 ETL 类型要求。如果 noop=true，Camel 也会设置 idempotent=true，以避免通过和再次消耗同一文件。	false	布尔值
preMove (consumer)	表达式（如文件语言）用于在处理前动态设置文件名。例如，要将 in-progress 文件移到订购目录中，将此值设置为 order。		字符串
preSort (consumer)	启用 pre-sort 后，消费者将在轮询期间对文件和目录名称进行排序，该名称从文件系统检索。如果您需要按排序的顺序对文件进行操作，您可能需要执行此操作。预排序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，也会在所有子目录中查找文件。	false	布尔值
resumeDownload (consumer)	配置是否启用了恢复下载。FTP 服务器必须支持此设置（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，而且必须启用选项二进制文件，这是支持恢复下载所必需的。	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)	设置在不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传输到路由。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。如果启用这个选项，则必须将 stepwise=false 设置为不能同时启用。	false	布尔值
download (consumer (advanced))	FTP 使用者是否应下载该文件。如果此选项设为 false，则消息正文将为空，但消费者仍会触发 Camel Exchange，其中包含文件名称、文件大小等详细信息。只是不会下载该文件。	false	布尔值
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
handleDirectoryParserAbsoluteResult (consumer (advanced))	如果目录解析程序产生了绝对路径，则允许使用者设置在路径中处理子文件夹和文件的方式。因此，一些 FTP 服务器可能会用绝对路径返回文件名，如果是，那么 FTP 组件需要通过将返回的路径转换为相对路径来处理此目的。	false	布尔值
ignoreFileNotFoundOrPermissionError (consumer (advanced))	是否忽略何时（在下载文件时列出目录中的文件）或下载文件，这些文件不存在或因为权限错误。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 (advanced))	可插拔式存储库 org.apache.camel.spi.IdempotentRepository.in-progress 存储库用于考虑被消耗的进度文件的当前。默认情况下，使用基于内存的存储库。		IdempotentRepository

Name	描述	默认值	类型
localWorkDirectory (consumer (advanced))	使用时，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。这非常有用，如果您使用一个非常大的远程文件，从而可以节省内存。		字符串
onCompletionExceptionHandler (consumer (advanced))	使用自定义 org.apache.camel.spi.ExceptionHandler 处理在完成过程中发生的任何抛出异常，供消费者执行提交或回滚。默认实现将在 WARN 级别记录任何异常并忽略。		ExceptionHandler
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
processStrategy (consumer (advanced))	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，如存在特殊就绪的文件。如果设置了这个选项，则不会应用 readLock 选项。		GenericFileProcessStrategy
useList (consumer (advanced))	是否在下载文件时允许使用 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，不允许使用 LIST 命令，因此您可以将这个选项设置为 false。请注意，在使用此选项时，要下载的特定文件不包括元数据信息，如文件大小、时间戳、权限等，因为这些信息只能在使用 LIST 命令时检索。	true	布尔值

Name	描述	默认值	类型
fileExist (producer)	如果文件名称已存在，则该怎么办。override （默认文件）替换现有文件。- Append - 将内容添加到现有文件。- Fail - 抛出 GenericFileOperationException，表示已有现有文件。- Ignore - 静默忽略问题，且不会覆盖现有文件，但假设所有内容都正常。- Move - 选项需要使用 moveExisting 选项进行配置。选项 eagerDeleteTargetFile 可用于控制移动文件时要执行的操作，并且存在现有文件，否则会导致 move 操作失败。Move 选项将移动任何现有文件，然后再编写目标文件。- 只有使用 tempFileName 选项时才适用 TryRename。这允许尝试将该文件从临时名称重命名为实际名称，而无需进行任何存在的检查。对于某些文件系统，特别是 FTP 服务器上，这个检查可能会更快。 Enum 值： • override • 附加 • Fail • Ignore • Move • TryRename	override	GenericFileExist
flatten (producer)	flatten 用于扁平化文件名路径，以剥离任何前导路径，因此它只是文件名。这样，您便可以将递归地消耗到子目录中，但当您将文件写入另一个目录时，会将文件写入单个目录中。在生成者上将其设置为 true 强制执行 CamelFileName 标头中的任何文件名将被剥离任何前导路径。	false	布尔值
jailStartingDirectory (producer)	用于仅向起始目录（和子）写入文件。这默认是启用的，不允许 Camel 将文件写入外部目录（在开箱即用的情况下更为安全）。您可以关闭此选项，以允许将文件写入起始目录之外的目录，如父目录或根文件夹。	true	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

Name	描述	默认值	类型
moveExisting (producer)	表达式（如文件语言）用于在配置 fileExist=Move 时使用计算文件名。要将文件移动到备份子目录中，只需输入 backup。这个选项只支持以下文件语言令牌：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动到基于当前 dir 作为基础的相对目录。		字符串
tempFileName (producer)	与 tempPrefix 选项相同，但对临时文件名的命名提供更加精细的控制，因为它使用文件语言。tempFilename 的位置相对于选项 'fileName' 中的最终文件位置，而不是基础 uri 中的目标目录。例如，如果选项 fileName 包含目录前缀：dir/finalFilename，则 tempFileName 相对于该子目录 dir。		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，也可避免消费者（不使用专用读取锁定）读取进度文件。在上传大型文件时，FTP 通常使用 FTP。		字符串
allowNullBody (producer (advanced))	用于指定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，如果设为 false，并尝试将 null 正文发送到文件组件，则将抛出 'Cannot write null body to file.' 的 GenericFileWriteException。如果 fileExist 选项被设置为 'Override'，则该文件将被截断，如果设为 附加该文件，则该文件将保持不变。	false	布尔值
chmod (producer (advanced))	允许您在存储的文件上设置 chmod。例如 chmod=640。		字符串
disconnectOnBatchComplete (producer (advanced))	批处理上传完成后是否要断开与远程 FTP 服务器的连接。disconnectOnBatchComplete 将仅断开当前与 FTP 服务器的连接。	false	布尔值
eagerDeleteTargetFile (producer (advanced))	是否强制删除任何现有目标文件。这个选项仅在使用 fileExists=Override 和 tempFileName 选项时才适用。您可以使用此选项禁用（将其设置为 false）在编写 temp 文件前删除目标文件。例如，您可以编写大文件，并且希望目标文件在 temp 文件被写入期间存在。这样可保证仅删除目标文件，直到最后一次时间之前，只需将 temp 文件重命名为目标文件名之前。这个选项还用于控制是否在启用 fileExist=Move 时删除任何现有的文件，并且存在现有文件。如果此选项 copyAndDeleteOnRenameFails false，则现有文件存在时会抛出异常（如果其为 true），则在移动操作前会删除现有文件。	true	布尔值

Name	描述	默认值	类型
keepLastModified (producer (advanced))	将保留来自源文件的最后修改的时间戳（如果有）。将使用 Exchange.FILE_LAST_MODIFIED 标头来定位时间戳。此标头可以包含 java.util.Date 或 long （时间戳）。如果时间戳存在，且启用了 选项，它将在写入的文件上设置这个时间戳。注意：此选项仅适用于文件制作者。您不能将此选项与任何 ftp producer 一起使用。	false	布尔值
moveExistingFileStrategy (producer (advanced))	策略(Custom Strategy)用于移动带有特殊命名令牌的文件，以便在配置了 fileExist=Move 时使用。默认情况下，如果没有提供自定义策略，则使用实现。		FileMoveExistingStrategy
sendNoop (producer (advanced))	在上传文件到 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，允许静默重新连接，以便能够上传该文件。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)	在 active 模式中设置客户端侧端口范围。语法为：minPort-maxPort Both 端口号包含，例如 10000-19999 以包含所有 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。对于文件制作者，这意味着应写入文件的目录。	true	布尔值
bufferSize (advanced)	用于编写文件的缓冲区大小（如果是 FTP 用于下载和上传文件）。	131072	int
connectTimeout (advanced)	设置等待由 FTPClient 和 JSCH 使用的连接建立的连接超时。	10000	int
ftpClient (advanced)	要使用自定义 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 用来为 FTPClientConfig 提供其他参数。		Map
ftpClientParameters (advanced)	由 FtpComponent 用来为 FTPClient 提供其他参数。		Map

Name	描述	默认值	类型
maximumReconnectAttempts (advanced)	指定在尝试连接到远程 FTP 服务器时 Camel 执行的最大重新连接尝试。使用 0 禁用此行为。		int
reconnectDelay (advanced)	millis Camel 中的延迟会在执行重新连接尝试前等待。	1000	long
siteCommand (advanced)	设置在成功登录后要执行的可选站点命令。可以使用新行字符分隔多个站点命令。		字符串
soTimeout (advanced)	设置 so timeout FTP 和 FTPS 是 millis 中的 SocketOptions.SO_TIMEOUT 值。推荐的选项是将其设置为 300000，因此没有挂起的连接。在 SFTP 上，此选项在 JSCH Session 实例中被设置为超时。	300000	int
stepwise (advanced)	设置在下载文件时，还是在上传文件到目录时，我们是否应该对目录进行循环更改。例如，如果您因为安全原因无法更改 FTP 服务器上的目录，您可以禁用它。步骤步骤不能与 streamDownload 一起使用。	true	布尔值
同步 (advanced)	设置是否应严格使用同步处理。	false	布尔值
throwExceptionOnConnectFailed (advanced)	如果没有抛出连接失败（假设）By 默认异常，并且记录 WARN，则抛出异常。您可以使用它来启用抛出异常，并处理 org.apache.camel.spi.PollingConsumerPollStrategy 回滚方法中引发的异常。	false	布尔值
Timeout (advanced)	设置用于仅等待 FTPClient 使用的回复的数据超时。	30000	int
antExclude (filter)	Ant 样式过滤器排除。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。		字符串
antFilterCaseSensitive (filter)	在 ant 过滤器中设置问题单敏感标志。	true	布尔值
antInclude (filter)	Ant 样式过滤器包含.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 的限制是否为 eager。如果为 eager，则限制在文件扫描期间。其中为 false 将扫描所有文件，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，这需要较高的内存用量，因为所有文件详情都在内存中执行排序。	true	布尔值

Name	描述	默认值	类型
exclude (filter)	用于排除文件，如果文件名与正则表达式模式匹配（匹配区分大小写）。请注意，如果您使用符号（如加号），如果将其配置为 endpoint uri，则需要使用 RAW （）语法进行配置。有关配置 endpoint uris 的更多详细信息。		字符串
excludeExt (filter)	用于排除匹配文件扩展名名称的文件（不区分大小写）。例如，要排除 bak 文件，然后使用 excludeExt=bak。多个扩展可以用逗号分开，例如要排除 bak 和 dat 文件，请使用 excludeExt=bak,dat。请注意，文件扩展名包含所有部分，例如，具有名为 mydata.tar.gz 的文件将扩展为 tar.gz。要获得更大的灵活性，请使用 include/exclude 选项。		字符串
Filter (filter)	可插拔过滤器作为 org.apache.camel.component.file.GenericFileFilter 类。如果过滤器在其 accept （）方法中返回 false，则将跳过文件。		GenericFileFilter
filterDirectory (filter)	根据简单语言过滤目录。例如，要过滤当前日期，您可以使用一个简单的日期模式，如 $\\{date:now:yyyMMdd}。		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 $\\{file:size} 5000。		字符串
idempotent (filter)	使用 Idempotent Consumer EIP 模式的选项让 Camel 跳过已经处理的文件。默认情况下，将使用基于内存的 LRUCache 来保存 1000 条目。如果 noop=true，则同时启用幂等性，以避免再次消耗同一文件。	false	布尔值
idempotentKey (filter)	使用自定义幂等密钥。默认情况下，使用文件的绝对路径。您可以使用文件语言，例如要使用文件名和文件大小，您可以执行： idempotentKey=$\\{file:name}-$\\{file:size}-$\\{file:size}。		字符串
idempotentRepository (filter)	可插拔存储库 org.apache.camel.spi.IdempotentRepository，如果未指定，并且 idempotent 为 true，则默认使用 MemoryIdempotentRepository。		IdempotentRepository
Include (filter)	用于包括文件，如果文件名与正则表达式模式匹配（匹配区分大小写）。请注意，如果您使用符号（如加号），如果将其配置为 endpoint uri，则需要使用 RAW （）语法进行配置。有关配置 endpoint uris 的更多详细信息。		字符串

Name	描述	默认值	类型
includeExt (filter)	用于包括匹配文件扩展名名称的文件（不区分大小写）。例如，要包含 txt 文件，然后使用 includeExt=txt。多个扩展可以用逗号分开，例如要包含 txt 和 xml 文件，请使用 includeExt=txt,xml。请注意，文件扩展名包含所有部分，例如，具有名为 mydata.tar.gz 的文件将扩展为 tar.gz。要获得更大的灵活性，请使用 include/exclude 选项。		字符串
maxDepth (filter)	递归处理目录时要遍历的最大深度。	2147483647	int
maxMessagesPerPoll (filter)	定义每个轮询收集的最多消息。默认情况下，没有设置最大值。可用于设置限制，例如 1000 个，以避免启动有数千个文件的服务器。将值设为 0 或负数设置为禁用它。注意：如果此选项正在使用，则文件和 FTP 组件将在任何排序之前进行限制。例如，如果您有 100000 文件并使用 maxMessagesPerPoll=500，则只有前 500 个文件会被提取，然后排序。您可以使用 eagerMaxMessagesPerPoll 选项，并将其设置为 false 以允许首先扫描所有文件，然后在之后排序。		int
minDepth (filter)	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）用于在处理后移动文件名时动态设置文件名。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 实现。		GenericFileExclusiveReadLockStrategy

Name	描述	默认值	类型
readLock (lock)	供消费者使用，仅当文件上有独占的 read-lock （例如，该文件不是 in-progress 或被写入）时轮询文件。Camel 将等待文件锁定被赋予。此选项在策略中提供构建：- none - No read lock is in use - markerFile - Camel 创建一个标记文件(fileName.camelLock)，然后在其上保存锁定。这个选项不适用于 FTP 组件 - 更改 - Changed 使用文件长度/修改时间戳来检测文件当前是否正在复制。至少将使用 1 sec 来确定这一点，因此此选项不能像其他人一样快消耗文件，但可以更可靠，因为 JDK IO API 无法始终确定文件当前是否被其他进程使用。选项 readLockCheckInterval 可用于设置检查频率。- fileLock - 用于使用 java.nio.channels.FileLock。这个选项不适用于 Windows OS 和 FTP 组件。当通过 mount/share 访问远程文件系统时，应避免使用这种方法，除非该文件系统支持分布式文件锁定。- rename 使用尝试将文件重命名为测试（如果我们可以获得独家的 read-lock. - 幂等 - （仅针对文件组件）是使用幂等Repository 作为 read-lock。这允许在幂等存储库实施支持集群时使用读取锁定。- idempotent-changed - （仅适用于文件组件） idempotent-changed 用于使用 idempotentRepository 并更改为组合的 read-lock。这允许在幂等存储库实施支持集群时使用读取锁定。- idempotent-rename - （仅适用于文件组件） idempotent-rename 用于使用 idempotentRepository，并作为组合的 read-lock 重命名。如果幂等存储库实现支持集群，这允许使用支持集群的读取锁定。不同的读取锁定并不适合以集群模式工作，其中不同节点上的并发消费者对共享文件系统上的相同文件竞争。flagsFile 使用接近 atomic 操作来创建空标记文件，但无法保证在集群中工作。fileLock 可以更好地工作，但文件系统需要支持分布式文件锁定，以此类推。如果幂等存储库支持集群（如 Hazelcast 组件或 Infinispan），则使用幂等的读取锁定可以支持集群。 Enum 值： • none • markerFile • fileLock • rename • changed • idempotent • idempotent-changed • idempotent-rename	none	字符串

Name	描述	默认值	类型
readLockCheckInterval (lock)	如果读取锁定支持，则 read-lock 中的间隔为 millis。此间隔用于尝试获取读取锁之间的睡眠状态。例如，在使用更改的读取锁定时，您可以为文件设置更高的间隔周期，以便进行速度较慢的写入。默认值 1 sec。如果生成者的写入速度非常慢，则可能太快。注意：对于 FTP，默认的 readLockCheckInterval 是 5000。readLockTimeout 值必须大于 readLockCheckInterval，但 thumb 规则是有一个超时，它至少比 readLockCheckInterval 高 2 倍。这需要确保允许读取锁定进程在达到超时时间前尝试获取锁定。	1000	long
readLockDeleteOrphanLockFiles (lock)	如果 Camel 没有正确关闭（如 JVM 崩溃），在启动后是否应该使用标记文件读取锁定删除任何孤立的读取锁定文件（如 JVM 崩溃）。如果将此选项转换为 false，则任何孤立的锁文件将导致 Camel 不尝试选择该文件，也可能是因为另一个节点同时从同一共享目录读取文件。	true	布尔值
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 DEBUG。您可以更改此级别，例如，OFF 没有任何日志记录。这个选项只适用于 readLock 类型：change, fileLock, idempotent, idempotent-changed, idempotent-rename, rename。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	DEBUG	LoggingLevel
readLockMarkerFile (lock)	是否使用更改、重命名或专用读取锁定类型的标记文件。默认情况下，使用标志文件来保护获取同一文件的其他进程。通过将这个选项设置为 false 可关闭此行为。例如，如果您不希望 Camel 应用程序将标记文件写入文件系统。	true	布尔值
readLockMinAge (lock)	这个选项只适用于 readLock=changed。它允许在尝试获取读取锁定前指定文件的最短期限。例如，使用 readLockMinAge=300s 来要求文件在最后 5 分钟内。这可加快更改的读取锁，因为它将只尝试获取至少给定年龄的文件。	0	long

Name	描述	默认值	类型
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 期望文件包含数据，因此默认值为 1。您可以将这个选项设为零，以允许消耗零长度文件。	1	long
readLockRemoveOnCommit (lock)	这个选项只适用于 readLock=idempotent。它允许您指定在处理文件成功时是否从幂等存储库中删除文件名条目，以及提交发生。默认情况下，文件不会被删除，这样可确保不会发生任何竞争条件，因此另一个活动节点可能会试图获取该文件。相反，idempotent 存储库可能支持驱除策略，这些策略可在 X 分钟后驱除文件名条目 - 这确保了竞争条件没有问题。请参阅 readLockIdempotentReleaseDelay 选项的更多详情。	false	布尔值
readLockRemoveOnRollback (lock)	这个选项只适用于 readLock=idempotent。它指定在处理文件失败时是否从幂等存储库中删除文件名条目，以及进行回滚。如果此选项为 false，则确认文件名条目（就像文件执行提交一样）。	true	布尔值
readLockTimeout (lock)	如果 read-lock 支持，则 read-lock 中的可选超时（如果支持）。如果无法授予 read-lock，并且触发超时，则 Camel 将跳过该文件。在下次轮询 Camel 时，将再次尝试 文件，这一次可能被赋予读锁。使用 0 或更低的值来指示永久。目前 fileLock，更改并重命名支持超时。注意：对于 FTP，默认的 readLockTimeout 值为 20000，而不是 10000。readLockTimeout 值必须大于 readLockCheckInterval，但 thumb 规则是有一个超时，它至少比 readLockCheckInterval 高 2 倍。这需要确保允许读取锁定进程在达到超时时间前尝试获取锁定。	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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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
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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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
帐户 （安全性）	用于登录的帐户。		字符串
password (security)	用于登录的密码。		字符串
用户名 （安全性）	用于登录的用户名。		字符串
shuffle (sort)	要表示文件列表（按随机顺序排列）。	false	布尔值
sortby (sort)	使用文件语言进行内置排序。支持嵌套排序，因此您可以按文件名排序，并且按修改日期排序第二组。		字符串
sorter (sort)	可插拔排序器作为 java.util.Comparator 类。		比较器

37.6. 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，则必须使用 ftpClient.trustStore.xxx 选项或配置自定义 ftpClient 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：

```
from("ftp://foo@myserver?password=secret&ftpClient.dataTimeout=30000").to("bean:foo");
```

您可以混合和匹配，并使用这两个前缀，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");
```

尽可能多地使用这些选项。

有关可能的选项和更多详情，请参阅 Apache Commons FTP FTPClientConfig 的文档。以及 Apache Commons FTP FTPClient。

如果您不喜欢在 url 中有多个和长的配置，您可以通过在 Registry 中允许 Camel 查找来引用 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，让 Camel 查找此 bean。

```
from("ftp://foo@myserver?password=secret&ftpClientConfig=#myConfig").to("bean:foo");
```

37.7. 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
```

37.8. 并发

FTP Consumer 不支持并发

FTP 消费者（具有相同端点）不支持并发（支持的 FTP 客户端不是线程安全）。
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 这个问题，它支持并发。

37.9. 更多信息

此组件是 File 组件的扩展。因此，File 组件页面中还有更多示例和详情。

37.10. 默认消耗文件时

默认情况下，FTP 使用者将保留在远程 FTP 服务器上保持不变的文件。如果您希望删除文件或将其移动到另一个位置，则必须明确配置它。例如，您可以使用 delete=true 删除文件，或使用 move=.done 将文件移到隐藏的子目录中。

常规文件消费者不同，因为它默认将文件移动到 .camel 子目录中。对于 FTP 使用者，Camel 默认情况下 不这样做 的原因是，它默认可能会缺少权限，以便能够移动或删除文件。

37.10.1. 限制

选项 readLock 可用于强制 Camel 不 消耗当前写入过程中的文件。但是，此选项默认是关闭的，因为它要求用户具有写访问权限。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消耗当前通过 FTP 编写的文件；例如，您可以写入临时目标并在编写文件后移动该文件。

使用 move 或 preMove 选项移动文件时，文件将限制为 FTP_ROOT 文件夹。这可以防止您将文件移动到 FTP 区域外。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。

37.11. 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	指定在发送到端点时用于输出消息的输出文件名（相对于端点目录）。如果不存在任何表达式，则生成的消息 ID 将用作文件名。
CamelFileNameProduced	已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，其目的是向最终用户提供写入的文件的名称。
CamelFileNameConsumed	所消耗的文件的文件名
CamelFileHost	远程主机名。
CamelFileLocalWorkPath	如果使用本地工作目录，到本地工作文件的路径。

此外，FTP/FTPS 使用者和生产者将使用以下标头增强 Camel 消息

标头	描述
CamelFtpReplyCode	FTP 客户端回复代码（类型是一个整数）
CamelFtpReplyString	FTP 客户端回复字符串

37.11.1. Exchange Properties

Camel 设置以下交换属性

标头	描述
CamelBatchIndex	当前索引掉此批处理中消耗的文件总数。
CamelBatchSize	此批处理中消耗的文件总数。
CamelBatchComplete	如果此批处理中没有更多文件，则为 true。

37.12. 关于超时

两组库（请参见 top）具有不同的 API 来设置超时。您可以对这两者使用 connectTimeout 选项，在 millis 中设置超时来建立网络连接。也可以在 FTP/FTPS 上设置一个单独的 soTimeout，对应于使用 ftpClient.soTimeout。请注意，SFTP 将自动使用 connectTimeout 作为其 soTimeout。timeout 选项仅适用于 FTP/FTPS，作为数据超时，对应于 ftpClient.dataTimeout 值。所有超时值都在 millis 中。

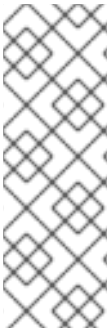
37.13. 使用本地工作目录

Camel 支持从远程 FTP 服务器使用，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入内存中，因为它使用 FileOutputStream 直接传输到本地文件。

Camel 将存储到名称与远程文件相同的本地文件，但在下载文件时，使用 .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 完成后，会删除本地文件。

因此，如果要从远程 FTP 服务器下载文件并将其存储为文件，则需要路由到文件端点，例如：

```
from("ftp://someone@someserver.com?
password=secret&localWorkDirectory=/tmp").to("file://inbox");
```



注意

以上路由非常高效，因为它避免将整个文件内容读到内存中。它将直接将远程文件下载到本地文件流。然后，java.io.File 句柄用作 Exchange 正文。文件生成者利用此事实，可以直接在工作文件 java.io.File 处理上工作，并对目标文件名执行 java.io.File.rename。当 Camel 知道它是本地的工作文件时，它可以优化和使用重命名，而不是文件副本，因为工作文件将随时删除。

37.14. 更改目录的步骤

在消耗文件时（例如下载）或生成文件（例如上传）时，Camel FTP 可以在两种模式下操作。

- 步骤
- Not stepwise

根据您的情况和安全问题，您可能需要选择一个。有些 Camel 最终用户只能在使用步骤时下载文件，而其他 Camel 最终用户只能下载这些文件。

您可以使用 stepwise 选项控制行为。

请注意，在大多数情况下，对目录进行分步更改只有在用户仅限于其主目录以及主目录报告为 "/" 时才起作用。

其中一个两者之间的差别最好在示例中进行说明。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
```

并且我们在每个子 A (a.txt)和 sub-b (b.txt)文件夹中都有一个文件。

37.15. 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.
```

正如您在启用步骤范围时所看到的那样，它将使用 CD xxx 遍历目录结构。

37.16. 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.
```

正如您在不使用步骤时所看到的那样，根本没有调用 CD 操作。

37.17. SAMPLES

在以下示例中，将 Camel 设置为每小时从 FTP 服务器下载一次(60 分钟)作为 BINARY 内容，并将它保存为本地文件系统中的文件。

使用 XML 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>
```

37.17.1. 使用远程 FTPS 服务器（意味着 SSL）和客户端验证

```
from("ftps://admin@localhost:2222/public/camel?
password=admin&securityProtocol=SSL&implicit=true
     &ftpClient.keyStore.file=./src/test/resources/server.jks
     &ftpClient.keyStore.password=password&ftpClient.keyStore.keyPassword=password")
  .to("bean:foo");
```

37.17.2. 使用远程 FTPS 服务器(explicit 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");
```

37.18. 自定义过滤

Camel 支持可插拔过滤策略。此策略使用 Java 中的 org.apache.camel.component.file.GenericFileFilter 中的构建。然后，您可以使用这样的过滤器配置端点，以便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，仅接受文件名中报告开头的文件。

然后，我们可以使用 filter 属性配置路由来引用我们在 spring XML 文件中定义的过滤器（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>
```

37.19. 使用 ANT 路径匹配程序进行过滤

ANT 路径匹配器是一个过滤器，它在 camel-spring jar 中提供。因此，如果您使用 Maven，则需要依赖于 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
- * 匹配零个或更多字符
- | 匹配路径中的零个或更多目录

以下示例演示了如何使用它：

37.20. 使用带有 SFTP 的代理

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，您还可以为代理分配用户名和密码。请参阅 com.jcraft.jsch.Proxy 文档来发现所有选项。

37.21. 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，如果没有公钥，您希望 Camel 尝试通过私有/公共 SSH 密钥进行身份验证，并回退到用户/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");
```

37.22. 使用固定名称消耗单个文件

当您要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，消费者仍会执行 FTP LIST 命令进行目录列表，然后根据 fileName 选项过滤这些文件。虽然在这个用例中，可能需要通过设置 useList=false 来关闭目录列表。例如，用于登录到 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 将其关闭，然后提供文件的固定名称以下载 fileName=myFileName.txt，然后 FTP 消费者仍然可以下载该文件。如果由于某些原因的文件不存在，则 Camel 默认会抛出异常，您可以通过设置 ignoreFileNotFoundOrPermissionError=true 来关闭并忽略它。

例如，要有一个 Camel 路由用于选择一个文件，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，我们使用了上面介绍的所有选项。

您还可以将其与 ConsumerTemplate 搭配使用。例如，要下载单个文件（如果存在），并将文件内容作为 String 类型获取：

```
String data = template.retrieveBodyNoWait("ftp://admin@localhost:21/nolist/?
password=admin&stepwise=false&useList=false&ignoreFileNotFoundOrPermissionError=true
&fileName=report.txt&delete=true", String.class);
```

37.23. 调试日志记录

此组件具有日志级别 TRACE，在遇到问题时很有用。

37.24. SPRING BOOT AUTO-CONFIGURATION

组件支持 13 个选项，如下所列。

Name	描述	默认值	类型

Name	描述	默认值	类型
camel.component.ftp.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.ftp.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.ftp.enabled	是否启用 ftp 组件的自动配置。这默认是启用的。		布尔值
camel.component.ftp.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.ftps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.ftps.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.ftps.enabled	是否启用 ftps 组件的自动配置。这默认是启用的。		布尔值
camel.component.ftps.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

Name	描述	默认值	类型
camel.component.ftps.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值
camel.component.sftp.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.sftp.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.sftp.enabled	是否启用 sftp 组件的自动配置。这默认是启用的。		布尔值
camel.component.sftp.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

第 38 章 GOOGLE BIGQUERY

从 Camel 2.20 开始

仅支持生成者。

Google Bigquery 组件通过链接：https://developers.google.com/api-client-library/java/apis/bigquery/v2 [Google Client Services API] 提供对 Cloud BigQuery 基础架构 的访问。

当前实现不使用 gRPC。

当前实施不支持查询 BigQuery，它只是一个制作者。

38.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 google-bigquery 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-google-bigquery-starter</artifactId>
</dependency>
```

38.2. 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 身份验证指南。

Google 安全凭证可以通过提供 GCP 凭证文件位置的路径来显式设置。

或者隐式设置，连接工厂会返回 应用程序默认凭证。

具有 服务帐户密钥 时，您可以为应用程序代码提供身份验证凭据。Google 安全凭证可以通过组件端点设置：

```
String endpoint = "google-bigquery://project-id:datasetId[:tableId]?
serviceAccountKey=/home/user/Downloads/my-key.json";
```

如果您不想设置文件系统路径，您还可以使用身份验证凭证文件的 base64 编码内容。

```
String endpoint = "google-bigquery://project-id:datasetId[:tableId]?
serviceAccountKey=base64:<base64 encoded>";
```

或者，通过设置环境变量 GOOGLE_APPLICATION_CREDENTIALS ：

```
export GOOGLE_APPLICATION_CREDENTIALS="/home/user/Downloads/my-key.json"
```

38.3. URI 格式

```
google-bigquery://project-id:datasetId[:tableId]?[options]
```

38.4. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

38.4.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

38.4.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

38.5. 组件选项

Google BigQuery 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
ConnectionFactory (producer)	**Autowired** ConnectionFactory 获取与 Bigquery 服务的连接。如果没有提供默认值，则将使用。		GoogleBigQueryConnectionFactory
datasetId (producer)	bigquery Dataset Id.		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
projectId (producer)	Google Cloud Project Id.		字符串
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

38.6. 端点选项

Google BigQuery 端点使用 URI 语法进行配置：

```
google-bigquery:projectId:datasetId:tableId
```

使用以下路径和查询参数：

38.6.1. 路径参数(3 参数)

Name	描述	默认值	类型
projectId (common)	**所需的** Google Cloud Project Id。		字符串
datasetId (common)	**必需** BigQuery Dataset Id。		字符串
tableid (common)	bigquery 表 ID。		字符串

38.6.2. 查询参数(4 参数)

Name	描述	默认值	类型
ConnectionFactory (producer)	**Autowired** ConnectionFactory 获取与 Bigquery 服务的连接。如果没有提供默认值，则将使用。		GoogleBigQueryConnectionFactory
useAsInsertId (producer)	用作插入 ID 的字段名称。		字符串
lazyStartProducer (producer (advanced))	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
serviceAccountKey (security)	json 格式的服务帐户密钥，将应用程序验证为 google 云平台的服务帐户。		字符串

38.7. 消息标头

Google BigQuery 组件支持 4 个消息标头，如下所列：

Name	描述	默认值	类型
CamelGoogleBigQueryTableSuffix (producer) 常数 ： TABLE_SUFFIX	插入数据时要使用的表后缀。		字符串
CamelGoogleBigQueryTableId (producer) 常数 ： TABLE_ID	提交数据的表 ID。如果指定，将覆盖端点配置。		字符串
CamelGoogleBigQueryInsertId (producer) 常数： INSERT_ID	InsertId 在插入数据时使用。		字符串
CamelGoogleBigQueryPartitionDecorator (producer) 常数： PARTITION_DECORATOR	分区 decorator 指定在插入数据时要使用的分区。		字符串

38.8. 生成者端点

生产者端点可以接受并传送到 BigQuery 个人和分组的交换。分组交换设置了 Exchange.GROUPED_EXCHANGE 属性。

Google BigQuery 生成者将在单个 api 调用中发送分组交换，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解码器写入。

Google BigQuery 端点需要有效负载是映射或映射列表。包含映射的有效负载将插入一行和包含映射列表的有效负载，将为列表中的每个条目插入一行。

38.9. 模板表

可以使用 GoogleBigQueryConstants.TABLE_SUFFIX 标头来指定模板表。例如，以下路由将创建表，并每天插入分片记录：

```
from("direct:start")
  .header(GoogleBigQueryConstants.TABLE_SUFFIX, "_${date:now:yyyyMMdd}")
  .to("google-bigquery:sampleDataset:sampleTable")
```

注意

建议您在此用例中使用分区。

有关 Template 表的更多信息，请参阅 模板表。

38.10. 分区

分区在创建表时指定，如果设置数据将自动分区到单独的表中。通过在交换中设置 GoogleBigQueryConstants.PARTITION_DECORATOR 标头来插入特定分区时，可以指定特定分区。

有关分区的更多信息，请参阅创建分区表。

38.11. 确保数据一致性

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。由于插入 ID 需要为每个行指定，当有效负载是列表时，无法使用插入的交换标头。如果有效负载是一个列表，则忽略 GoogleBigQueryConstants.INSERT_ID。在这种情况下，使用查询参数 useAsInsertId。

如需更多信息，请参阅 数据一致性

38.12. 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-bigquery-sql.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.google-bigquery-sql.connection-factory	ConnectionFactory 以获取与 Bigquery 服务的连接。如果没有提供默认值，则将使用。选项是 org.apache.camel.component.google.bigquery.GoogleBigQueryConnectionFactory 类型。		GoogleBigQueryConnectionFactory
camel.component.google-bigquery-sql.enabled	是否启用 google-bigquery-sql 组件的自动配置。这默认是启用的。		布尔值
camel.component.google-bigquery-sql.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.google-bigquery-sql.project-id	Google Cloud Project Id.		字符串
camel.component.google-bigquery.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.google-bigquery.connection-factory	ConnectionFactory 以获取与 Bigquery 服务的连接。如果没有提供默认值，则将使用。选项是 org.apache.camel.component.google.bigquery.GoogleBigQueryConnectionFactory 类型。		GoogleBigQueryConnectionFactory
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.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

Name	描述	默认值	类型
camel.component.google-bigquery.project-id	Google Cloud Project Id.		字符串

第 39 章 GOOGLE PUBSUB

从 Camel 2.19 开始

支持生成者和消费者。

Google Pubsub 组件通过 Google Cloud Java Client for Google Cloud Pub/Sub 提供对 Cloud Pub/Sub 基础架构 的访问。

39.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 google-pubsub 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-google-pubsub-starter</artifactId>
</dependency>
```

39.2. 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。

39.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

39.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

39.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL *作为在 Java 中配置端点和数据格式的安全* 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

39.4. 组件选项

Google Pubsub 组件支持 10 个选项，如下所列。

Name	描述	默认值	类型
authentication (common)	在与 PubSub 服务交互时使用凭证（在使用仿真程序时不需要身份验证）。	true	布尔值
endpoint (common)	与本地 Pub/Sub 模拟器一起使用的端点。		字符串

Name	描述	默认值	类型
serviceAccountKey (common)	用作 PubSub publisher/subscriber 的凭证的服务帐户密钥。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
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
synchronousPullRetryableCodes (consumer)	用于同步拉取的额外可重试错误代码的逗号分隔列表。默认情况下，PubSub 客户端库重试 ABORTED, UNAVAILABLE, UNKNOWN。		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
publisherCacheSize (producer)	要缓存的最大生成者数。如果您有大量不同主题的制作者，可能会增加它。		int
publisherCacheTimeout (producer)	每个制作者在缓存中保留多少毫秒。		int
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
publisherTerminationTimeout (advanced)	应允许生成者终止多少毫秒。		int

39.5. 端点选项

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：

39.5.1. 路径参数(2 参数)

Name	描述	默认值	类型
projectId (common)	**必需** Google Cloud PubSub Project Id。		字符串
destinationName (common)	**必需** 目标名称。对于消费者，这将是订阅名称，而对于制作者，这将是主题名称。		字符串

39.5.2. 查询参数(15 参数)

Name	描述	默认值	类型
authentication (common)	在与 PubSub 服务交互时使用凭证（在使用仿真程序时不需要身份验证）。	true	布尔值
loggerId (common)	与所需的父路由匹配时要使用的日志记录器 ID。		字符串
serviceAccountKey (common)	用作 PubSub publisher/subscriber 的凭证的服务帐户密钥。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
ackMode (consumer)	AUTO = Exchange 在完成后得到 ack'ed/nack。 NONE = 下游进程必须明确进行 ack/nack。 Enum 值： • AUTO • NONE	AUTO	AckMode
concurrentConsumers (consumer)	从订阅中使用的并行流数量。	1	整数
maxAckExtensionPeriod (consumer)	设置消息 ack 截止时间将延长的最大周期。以秒为单位的值。	3600	int
maxMessagesPerPoll (consumer)	在单个 API 调用中从服务器接收的最大消息数。	1	整数
synchronousPull (consumer)	同步拉取消息的批处理。	false	布尔值

Name	描述	默认值	类型
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
lazyStartProducer (producer (advanced))	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
messageOrderingEnabled (producer (advanced))	应启用消息排序。	false	布尔值
pubsubEndpoint (producer (advanced))	要使用的 pub/Sub 端点。使用消息排序时需要，并确保即使使用多个发布者，也会按顺序接收消息。		字符串
Serializer (producer (advanced))	**Autowired** A custom GooglePubsubSerializer 用于在制作者中序列化消息有效负载。		GooglePubsubSerializer

39.6. 消息标头

Google Pubsub 组件支持 5 个消息标头，如下所列：

Name	描述	默认值	类型
CamelGooglePubsubMessageId (common) 恒定：MESSAGE_ID	发布消息时由服务器分配的消息的 ID。		字符串
CamelGooglePubsubMsgAckId (consumer) 常数：ACK_ID	用于确认收到的消息的 ID。		字符串
CamelGooglePubsubPublishTime (consumer) 恒定：PUBLISH_TIME	发布消息的时间。		Timestamp
CamelGooglePubsubAttributes (common) 常数：ATTRIBUTES	消息的属性。		Map
CamelGooglePubsubOrderingKey (producer) 常数：ORDERING_KEY	如果非空，请标识应遵守发布顺序的相关消息。		字符串

39.7. 生成者端点

生产者端点可以接受并交付给 PubSub 个人和分组的交换。分组交换设置了 Exchange.GROUPED_EXCHANGE 属性。

Google PubSub 预期有效负载是 byte[] 数组，Producer 端点将发送：

- **字符串正文为 byte[]，编码为 UTF-8**

- byte[] 正文，如下所示
- 其他所有内容都会序列化为 byte[] 数组

Map 设置为消息标头 GooglePubsubConstants.ATTRIBUTES 将作为 PubSub 属性发送。

Google PubSub 支持订购消息交付。

要启用此设置，将选项 messageOrderingEnabled 设置为 true，将 pubsubEndpoint 设置为 GCP 区域。

在生成消息时，设置消息标头 GooglePubsubConstants.ORDERING_KEY。这将设置为消息的 PubSub orderingKey。

有关 订购消息 的更多信息。

交换发送至 PubSub 后，PubSub Message ID 将分配给标头 GooglePubsubConstants.MESSAGE_ID。

39.8. 消费者端点

如果 Google PubSub 在周期内还没有被确认，则 Google PubSub 将重新设计为订阅上的配置选项。

在交换处理完成后，组件将确认消息。

如果路由抛出异常，则交换标记为失败，组件将立即删除消息 - 它将立即重新设计。

要对消息进行 ack/nack，组件使用 Acknowledgement ID 存储为标头 GooglePubsubConstants.ACK_ID。如果删除或篡改了标头，则 ack 将失败，并在 ack 截止时间后再次更新消息。

39.9. 消息正文

消费者端点将消息的内容返回为 byte[] - 与底层系统发送的内容完全相同。它是对内容进行转换/unmarshall 的路由。

39.10. 身份验证配置

默认情况下，此组件使用 GoogleCredentials.getApplicationDefault （） 来查询凭证。可以通过将 authentication 选项设置为 false 来禁用此行为，在这种情况下，将对 Google API 的请求进行，而无需身份验证详情。*这只在针对仿真程序进行开发时才需要。可以通过提供服务帐户密钥文件的路径来更改此行为。*

39.11. 回滚和重新发送

Google PubSub 的回滚取决于 Acknowledgement Deadline (Google PubSub 期望接收确认的时间周期)。如果尚未收到确认，则消息为redelivered。

Google 提供了一个 API，用于扩展消息的截止时间。

Google PubSub 文档 的更多信息。

因此，回滚实际上是一个期限扩展 API 调用，其值为零 - 例如，现在已经达到截止时间，消息可以被重新分配给下一个消费者。

通过将消息标头 GooglePubsubConstants.ACK_DEADLINE 设置为以秒为单位，可以延迟消息重新发送，方法是为回滚明确设置确认期限。

39.12. 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-pubsub.authenticate	在与 PubSub 服务交互时使用凭证（在使用仿真程序时不需要身份验证）。	true	布尔值

Name	描述	默认值	类型
camel.component.google-pubsub.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.google-pubsub.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.google-pubsub.enabled	是否启用 google-pubsub 组件的自动配置。这默认是启用的。		布尔值
camel.component.google-pubsub.endpoint	与本地 Pub/Sub 模拟器一起使用的端点。		字符串
camel.component.google-pubsub.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.google-pubsub.publisher-cache-size	要缓存的最大生成者数。如果您有大量不同主题的制作者，可能会增加它。		整数
camel.component.google-pubsub.publisher-cache-timeout	每个制作者在缓存中保留多少毫秒。		整数
camel.component.google-pubsub.publisher-termination-timeout	应允许生成者终止多少毫秒。		整数
camel.component.google-pubsub.service-account-key	用作 PubSub publisher/subscriber 的凭证的服务帐户密钥。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

Name	描述	默认值	类型
camel.component.google-pubsub.synchronous-pull-retryable-codes	用于同步拉取的额外可重试错误代码的逗号分隔列表。默认情况下，PubSub 客户端库重试 ABORTED, UNAVAILABLE, UNKNOWN。		字符串

第 40 章 GRPC

从 Camel 2.19 开始

支持生成者和消费者

gRPC 组件允许您使用 协议缓冲(protobuf)交换格式通过 HTTP/2 传输调用或公开远程过程调用(RPC)服务。

40.1. 依赖项

当在 Camel Spring Boot 中使用 grpc 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-grpc-starter</artifactId>
</dependency>
```

40.2. URI 格式

```
grpc:host:port/service[?options]
```

40.3. 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

40.3.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

40.3.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

40.4. 组件选项

gRPC 组件支持 3 个选项，如下所列。

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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

40.5. 端点选项

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：

40.5.1. 路径参数(3 参数)

Name	描述	默认值	类型
host (common)	**必需** gRPC 服务器主机名。在使用生成者时作为消费者或远程服务器主机名，这是 localhost 或 0.0.0.0。		字符串
port (common)	**需要** gRPC 本地或远程服务器端口。		int
service (common)	从协议缓冲区描述符文件（软件包点服务定义名称）中所需的完全限定服务名称。		字符串

40.5.2. 查询参数(29 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
autoDiscoverServerInterceptors (consumer)	如果为 true 设置 autoDiscoverServerInterceptors 机制，则组件将自动在 registry 中查找 ServerInterceptor 实例，否则它将跳过该检查。	true	布尔值

Name	描述	默认值	类型
consumerStrategy (consumer)	这个选项指定在流传输模式下处理服务请求和响应的顶级策略。如果选择了聚合策略，则会在列表中累计所有请求，然后传送到流中，累积的响应将发送到发送者。如果选择了传播策略，则会将请求发送到流，响应将立即发送到发送者。 Enum 值： ● 聚合 ● PROPAGATION	PROPAGATION	GrpcConsumerStrategy
forwardOnCompleted (consumer)	确定是否应将 onCompleted 事件推送到 Camel 路由。	false	布尔值
forwardOnError (consumer)	确定是否应将 onError 事件推送到 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
routeControlledStreamObserver (consumer)	使路由能够控制流观察器。如果将此值设置为 true，则 gRPC 调用的响应观察器将使用 Exchange 对象中的名称 GrpcConstants.GRPC_RESPONSE_OBSERVER 设置。请注意，路由中应调用流 observer's onNext （）, onError （）, onCompleted （）方法。	false	布尔值
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： ● InOnly ● InOut		ExchangePattern

Name	描述	默认值	类型
autoDiscoverClientInterceptors (producer)	如果为 true 设置 autoDiscoverClientInterceptors 机制，则组件将自动在 registry 中查找 ClientInterceptor 实例，否则它将跳过该检查。	true	布尔值
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，单个交换被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' 类型）。 Enum 值： • SIMPLE • 流	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
lazyStartProducer (producer (advanced))	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
同步 (advanced)	设置是否应严格使用同步处理。	false	布尔值
authenticationType (security)	身份验证方法类型提前到 SSL/TLS 协商。 Enum 值： • NONE • GOOGLE • JWT	NONE	GrpcAuthType

Name	描述	默认值	类型
jwtAlgorithm (security)	JSON Web 令牌签名算法。 Enum 值： • HMAC256 • HMAC384 • HMAC512	HMAC 256	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 格式链接的 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
keyResource (security)	PKCS#8 私钥文件资源采用 PEM 格式链接。		字符串
negotiationType (security)	标识用于 HTTP/2 通信的安全协商类型。 Enum 值： • TLS • PLAINTEXT_UPGRADE • PLAINTEXT	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，用于验证远程端点的证书。		字符串

40.6. 消息标头

gRPC 组件支持 3 个消息标头，如下所列：

Name	描述	默认值	类型
CamelGrpcMethodName (consumer) 常量 ：GRPC_METHOD_NAME_HEADER	由消费者服务处理的方法名称。		字符串
CamelGrpcUserAgent (consumer) 常量 ：GRPC_USER_AGENT_HEADER	如果提供，给定代理将预先填充 gRPC 库的用户代理信息。		字符串
CamelGrpcEventType (consumer) 常量 ：GRPC_EVENT_TYPE_HEADER	从发送请求中接收的事件类型。可能的值：在Next onCompleted onError 上。		字符串

40.7. 传输安全性和身份验证支持

以下 身份验证机制 内置于 gRPC 并在此组件中可用：

- *SSL/TLS：gRPC 具有 SSL/TLS 集成，并推广使用 SSL/TLS 验证服务器，并加密在客户端和服务器之间交换的所有数据。可选的机制可用于客户端为 mutual 身份验证提供证书。*
- *使用 Google 来实现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。提供了额外的支持，以便在通过 gRPC 访问 Google API 时获取访问令牌。通常，必须在频道中使用这种机制以及 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	值	必填/选填
1	SSL/TLS	negotiationType	TLS	必填

num.	选项	参数	值	必填/选填
		keyCertChainResource		必填
		keyResource		必填
		keyPassword		选填
		trustCertCollectionResource		选填
2	**使用 Google API 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 令牌实现身份验证**	authenticationType	JWT	必填
		negotiationType	NONE 或 TLS	可选。TLS/SSL 不会检查此类型，但强烈建议使用。
		jwtAlgorithm	HMAC256 (default)或 (HMAC384, HMAC512)	选填
		jwtSecret		必填
		jwtIssuer		选填
		jwtSubject		选填

40.8. 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传入流参数的流程。

调用样式	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

调用样式	请求类型	响应类型	请求正文类型	结果正文类型
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
asynchronous	simple	simple	对象	List<Object>
asynchronous	simple	流	对象	List<Object>
asynchronous	流	simple	对象或 List<Object>	List<Object>
asynchronous	流	流	对象或 List<Object>	List<Object>

40.9. 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>
```

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 streaming producer 策略的 gRPC 服务制作者（需要使用"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}");
```

启用 gRPC 服务消费者 TLS/SSL 安全协商：

```
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 (JWT)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");
```

40.10. 配置

使用 protobuf-maven-plugin，它调用 Protocol Buffer Compiler (protoc)从 .proto （协议缓冲区定义）文件生成 Java 源文件。此插件将生成程序请求和响应类，其构建器和 gRPC 流程 stubs 类。

以下步骤是必需的：

在项目 pom.xml 的 < build> 标签内插入操作系统和 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>1.7.1</version>
  </extension>
</extensions>
```

–

将 gRPC 和 protobuf Java 代码生成器插件插入到项目的 pom.xml 的 < plugins > 标签中。

```
<plugin>
  <groupId>org.xolstice.maven.plugins</groupId>
  <artifactId>protobuf-maven-plugin</artifactId>
  <version>0.6.1</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>
```

40.11. 其他资源

请参阅这些资源：

- gRPC 项目站点
- Maven 协议缓冲器插件

40.12. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型

Name	描述	默认值	类型
camel.component.grpc.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.grpc.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.grpc.enabled	是否启用 grpc 组件的自动配置。这默认是启用的。		布尔值
camel.component.grpc.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

第 41 章 标头

标头表达式语言允许您提取命名标头的值。

41.1. 依赖项

Header 语言是 camel-core 的一部分。

当在 Red Hat build of Camel Spring Boot 中使用 标头 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-core-starter</artifactId>
</dependency>
```

41.2. 标头选项

标头语言支持 1 个选项，如下所列。

Name	默认值	Java 类型	描述
trim		布尔值	是否修剪值以移除前导和结尾的空格和换行符。

41.3. 用法示例

recipientList EIP 可以使用标头：

```
<route>
  <from uri="direct:a" />
  <recipientList>
    <header>myHeader</header>
  </recipientList>
</route>
```

在这种情况下，接收者列表包含在标头 'myHeader' 中。

以及 Java DSL 中的相同示例：

```
from("direct:a").recipientList(header("myHeader"));
```

41.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 147 选项，如下所列。

Name	描述	默认值	类型
camel.cloud.consul.service-discovery.acl-token	设置用于 Consul 的 ACL 令牌。		字符串
camel.cloud.consul.service-discovery.block-seconds	等待监视事件的秒数，默认为 10 秒。	10	整数
camel.cloud.consul.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.consul.service-discovery.connect-timeout-millis	OkHttpClient 的连接超时。		Long
camel.cloud.consul.service-discovery.datacenter	数据中心。		字符串
camel.cloud.consul.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.consul.service-discovery.password	设置用于基本身份验证的密码。		字符串

Name	描述	默认值	类型
camel.cloud.consul.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.consul.service-discovery.read-timeout-millis	OkHttpClient 的读取超时。		Long
camel.cloud.consul.service-discovery.url	Consul 代理 URL。		字符串
camel.cloud.consul.service-discovery.user-name	设置用于基本身份验证的用户名。		字符串
camel.cloud.consul.service-discovery.write-timeout-millis	OkHttpClient 的写入超时。		Long
camel.cloud.dns.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.dns.service-discovery.domain	域名；		字符串
camel.cloud.dns.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.dns.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.dns.service-discovery.proto	所需服务的传输协议。	_tcp	字符串

Name	描述	默认值	类型
camel.cloud.etcd.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.etcd.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.etcd.service-discovery.password	用于基本身份验证的密码。		字符串
camel.cloud.etcd.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.etcd.service-discovery.service-path	查找服务发现的路径。	/services/	字符串
camel.cloud.etcd.service-discovery.timeout	要设置操作可以采取的最长时间，请执行以下操作：		Long
camel.cloud.etcd.service-discovery.type	要设置发现类型，有效值为 on-demand 和 watch。	按需	字符串
camel.cloud.etcd.service-discovery.uris	客户端可以连接到的 URI。		字符串
camel.cloud.etcd.service-discovery.user-name	用于基本身份验证的用户名。		字符串
camel.cloud.kubernetes.service-discovery.api-version	使用客户端查找时设置 API 版本。		字符串

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.ca-cert-data	使用客户端查找时设置证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.ca-cert-file	在使用客户端查找时，设置从文件加载的证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-data	使用客户端查找时设置客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-file	在使用客户端查找时，设置从文件加载的客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-algo	设置客户端密钥存储算法，如使用客户端查找时 RSA。		字符串
camel.cloud.kubernetes.service-discovery.client-key-data	使用客户端查找时设置客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-file	在使用客户端查找时，设置从文件加载的客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-passphrase	使用客户端查找时设置客户端密钥存储密码短语。		字符串
camel.cloud.kubernetes.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.kubernetes.service-discovery.dns-domain	设置用于 DNS 查找的 DNS 域。		字符串

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.kubernetes.service-discovery.lookup	如何执行服务查找。可能的值有：client、dns、environment。在使用客户端时，客户端会查询 kubernetes master 来获取提供该服务的活跃 pod 列表，然后随机（或循环）选择一个 pod。当使用 dns 时，服务名称被解析为 name.namespace.svc.dnsDomain。当使用 dnssrv 时，服务名称使用 SRV 查询解析 ....svc... when using environment，环境变量用于查找服务。默认情况下使用环境。	环境	字符串
camel.cloud.kubernetes.service-discovery.master-url	在使用客户端查找时，将 URL 设置为 master。		字符串
camel.cloud.kubernetes.service-discovery.namespace	设置要使用的命名空间。默认情况下，将使用来自 ENV 变量 KUBERNETES_MASTER 的命名空间。		字符串
camel.cloud.kubernetes.service-discovery.oauth-token	在使用客户端查找时，为身份验证设置 OAUTH 令牌（而不是用户名/密码）。		字符串
camel.cloud.kubernetes.service-discovery.password	在使用客户端查找时设置用于身份验证的密码。		字符串
camel.cloud.kubernetes.service-discovery.port-name	设置用于 DNS/DNSSRV 查找的端口名称。		字符串
camel.cloud.kubernetes.service-discovery.port-protocol	设置用于 DNS/DNSSRV 查找的端口协议。		字符串
camel.cloud.kubernetes.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.trust-certs	设置在使用客户端查找时是否打开信任证书检查。	false	布尔值
camel.cloud.kubernetes.service-discovery.username	在使用客户端查找时设置用于身份验证的用户名。		字符串
camel.cloud.ribbon.load-balancer.client-name	设置 Ribbon 客户端名称。		字符串
camel.cloud.ribbon.load-balancer.configurations	定义其他配置定义。		Map
camel.cloud.ribbon.load-balancer.enabled	启用组件。	true	布尔值
camel.cloud.ribbon.load-balancer.namespace	命名空间。		字符串
camel.cloud.ribbon.load-balancer.password	密码。		字符串
camel.cloud.ribbon.load-balancer.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.ribbon.load-balancer.username	用户名。		字符串

Name	描述	默认值	类型
camel.hystrix.allow-maximum-size-to-diverge-from-core-size	允许配置使 maximumSize 生效。然后该值可以等于或大于 coreSize。	false	布尔值
camel.hystrix.circuit-breaker-enabled	是否使用 HystrixCircuitBreaker。如果为 false，则不会使用 断路器逻辑，并且所有允许的请求。这与 circuitBreakerForceClosed （）的影响类似，除非继续跟踪指标，知道它是否应该是 open/closed，此属性即使实例化一个断路器。	true	布尔值
camel.hystrix.circuit-breaker-error-threshold-percentage	错误百分比阈值（如 50 ）指向断路器将打开和拒绝请求。它将在 circuitBreakerSleepWindowInMilliseconds 中定义的持续时间保持出差；与 HystrixCommandMetrics.getHealthCounts （）进行比较的错误百分比。	50	整数
camel.hystrix.circuit-breaker-force-closed	如果为 true，HystrixCircuitBreaker#allowRequest （）将始终返回 true 以允许请求，无论 HystrixCommandMetrics.getHealthCounts （）的错误百分比如何。如果设为 true，则 circuitBreakerForceOpen （）属性具有优先权。	false	布尔值
camel.hystrix.circuit-breaker-force-open	如果为 true，HystrixCircuitBreaker.allowRequest （）将始终返回 false，从而导致电路变为开路（接受），并拒绝所有请求。此属性优先于 circuitBreakerForceClosed （）;。	false	布尔值
camel.hystrix.circuit-breaker-request-volume-threshold	metricsRollingStatisticalWindowInMilliseconds （）中的最少请求数必须存在于 HystrixCircuitBreaker 之前。如果此数字低于这个数字，无论错误百分比如何，电路都不会被出差。	20	整数
camel.hystrix.circuit-breaker-sleep-window-in-milliseconds	HystrixCircuitBreaker trips 之后的时间（以毫秒为单位），它应该在尝试请求前等待。	5000	整数
camel.hystrix.configurations	定义其他配置定义。		Map
camel.hystrix.core-pool-size	传递给 java.util.concurrent.ThreadPoolExecutor#setCorePoolSize (int)的核心 thread-pool 大小。	10	整数
camel.hystrix.enabled	启用组件。	true	布尔值

Name	描述	默认值	类型
camel.hystrix.execution-isolation-semaphore-max-concurrent-requests	允许 HystrixCommand.run （）的并发请求数。超过并发限制的请求将被拒绝。仅在执行IsolationStrategy == SEMAPHORE 时使用。	20	整数
camel.hystrix.execution-isolation-strategy	将通过什么隔离策略 HystrixCommand.run （）执行。如果 THREAD，它将在单独的线程上执行，并且受 thread-pool 中的线程数量限制的并发请求。如果 SEMAPHORE，它将在调用线程上执行，并且受 semaphore 数限制的并发请求。	线程	字符串
camel.hystrix.execution-isolation-thread-interrupt-on-timeout	当线程超时时，执行线程是否应该尝试中断（使用 future#cancel）。仅在执行IsolationStrategy （）== THREAD 时才适用。	true	布尔值
camel.hystrix.execution-timeout-enabled	此命令是否启用了超时机制。	true	布尔值
camel.hystrix.execution-timeout-in-milliseconds	以毫秒为单位，将命令超时和停止执行的时间（以毫秒为单位）。如果 executionIsolationThreadInterruptOnTimeout == true 且命令是线程隔离，则执行线程将中断。如果命令是 semaphore-isolated 和 HystrixObservableCommand，则该命令将被取消订阅。	1000	整数
camel.hystrix.fallback-enabled	出现故障时，是否应尝试 HystrixCommand.getFallback （）。	true	布尔值
camel.hystrix.fallback-isolation-semaphore-max-concurrent-requests	允许 HystrixCommand.getFallback （）的并发请求数。超过并发限制的请求将快速失败，且不会尝试检索回退。	10	整数
camel.hystrix.group-key	设置要使用的 group 键。默认值为 CamelHystrix。	Camel Hystrix	字符串
camel.hystrix.keep-alive-time	更长的时间（以分钟为单位）传递给 ThreadPoolExecutor#setKeepAliveTime (long,TimeUnit)。	1	整数

Name	描述	默认值	类型
camel.hystrix.max-queue-size	在 HystrixConcurrencyStrategy.getBlockingQueue (int)中传递给 BlockingQueue 的最大队列大小应该只影响 threadpool 的实例化 - 它不会立即更改队列大小。为此，请使用 queueSizeRejectionThreshold （）。	-1	整数
camel.hystrix.maximum-size	传递给 ThreadPoolExecutor#setMaximumPoolSize (int)的最大 thread-pool 大小。这是可在不开始拒绝 HystrixCommands 的情况下支持的最大并发数量。请注意，只有在您也设置了 allowMaximumSizeToDivergeFromCoreSize 时，此设置才会生效。	10	整数
camel.hystrix.metrics-health-snapshot-interval-in-milliseconds	在允许计算成功和错误百分比时等待的时间（以毫秒为单位），并影响 HystrixCircuitBreaker.isOpen （）状态。在高容量电路上，错误百分比的连续计算可能会成为 CPU 密集型，从而控制其计算的频率。	500	整数
camel.hystrix.metrics-rolling-percentile-bucket-size	滚动百分比的每个存储桶中存储的最大值数。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	10	整数
camel.hystrix.metrics-rolling-percentile-enabled	是否应该使用 HystrixRollingPercentile 内部 HystrixCommandMetrics 来捕获百分比的指标。	true	布尔值
camel.hystrix.metrics-rolling-percentile-window-buckets	滚动窗口的存储桶数量被分成。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	6	整数
camel.hystrix.metrics-rolling-percentile-window-in-milliseconds	以毫秒为单位的滚动窗口的持续时间。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	10000	整数
camel.hystrix.metrics-rolling-statistical-window-buckets	滚动统计窗口划分为的 bucket 数量。这在 HystrixCommandMetrics 中被传递给 HystrixRollingNumber。	10	整数

Name	描述	默认值	类型
camel.hystrix.metrics-rolling-statistical-window-in-milliseconds	此属性设置统计滚动窗口的持续时间，以毫秒为单位。这是为线程池保留指标的时间。窗口被分成 bucket，按这些增量回滚。	10000	整数
camel.hystrix.queue-size-rejection-threshold	队列大小拒绝阈值是 artificial max size，即使尚未达到 maxQueueSize，也会发生拒绝。这是因为 BlockingQueue 的 maxQueueSize 无法动态更改，我们希望动态更改影响拒绝的队列大小。在排队线程以进行执行时，HystrixCommand 会使用它。	5	整数
camel.hystrix.request-log-enabled	HystrixCommand 执行和事件是否应记录到 HystrixRequestLog。	true	布尔值
camel.hystrix.thread-pool-key	设置要使用的线程池密钥。默认情况下，将使用与 groupKey 配置相同的值。	Camel Hystrix	字符串
camel.hystrix.thread-pool-rolling-number-statistical-window-buckets	滚动统计窗口划分为的 bucket 数量。这会传递到每个 HystrixThreadPoolMetrics 实例内的 HystrixRollingNumber。	10	整数
camel.hystrix.thread-pool-rolling-number-statistical-window-in-milliseconds	统计滚动窗口的持续时间（以毫秒为单位）。这会传递到每个 HystrixThreadPoolMetrics 实例内的 HystrixRollingNumber。	10000	整数
camel.language.constant.enabled	是否启用恒定语言的自动配置。这默认是启用的。		布尔值
camel.language.constant.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.csimple.enabled	是否启用 csimple 语言的自动配置。这默认是启用的。		布尔值
camel.language.csimple.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.exchangeproperty.enabled	是否启用 exchangeProperty 语言的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.language.exchangeproperty.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.file.enabled	是否启用文件语言的自动配置。这默认是启用的。		布尔值
camel.language.file.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.header.enabled	是否启用标头语言的自动配置。这默认是启用的。		布尔值
camel.language.header.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.ref.enabled	是否启用 ref 语言的自动配置。这默认是启用的。		布尔值
camel.language.ref.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.simple.enabled	是否启用简单语言的自动配置。这默认是启用的。		布尔值
camel.language.simple.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.tokenize.enabled	是否启用令牌化语言的自动配置。这默认是启用的。		布尔值
camel.language.tokenize.group-delimiter	设置在分组时要使用的分隔符。如果没有设置，则令牌将用作分隔符。		字符串
camel.language.tokenize.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.resilience4j.automatic-transition-from-open-to-half-open-enabled	在通过 waitDurationInOpenState 后，启用从 OPEN 自动过渡到 HALF_OPEN 状态。	false	布尔值

Name	描述	默认值	类型
camel.resilience4j.circuit-breaker-ref	代表现有的 io.github.resilience4j.circuitbreaker.CircuitBreaker 实例从 registry 中查找和使用。使用此选项时，不使用任何其他断路器选项。		字符串
camel.resilience4j.config-ref	指的是现有的 io.github.resilience4j.circuitbreaker.CircuitBreakerConfig 实例，以便从 registry 中查找和使用。		字符串
camel.resilience4j.configurations	定义其他配置定义。		Map
camel.resilience4j.enabled	启用组件。	true	布尔值
camel.resilience4j.failure-rate-threshold	以百分比为单位配置故障率阈值。如果失败率相等或大于阈值，则 CircuitBreaker 过渡到 open，并启动短路调用。阈值必须大于 0，而不是大于 100。默认值为 50 百分比。		æµ®ç‚¹â€¼
camel.resilience4j.minimum-number-of-calls	在 CircuitBreaker 可以计算错误率之前，配置所需的最少调用数（每个滑动期限）。例如，如果 minimumNumberOfCalls 为 10，则必须至少记录 10 个调用，然后才能计算失败率。如果只记录了 9 个调用，则 CircuitBreaker 不会过渡到 open，即使所有 9 调用都失败。默认 minimumNumberOfCalls 为 100。	100	整数
camel.resilience4j.permitted-number-of-calls-in-half-open-state	配置 CircuitBreaker 为一半打开时允许的调用数量。大小必须大于 0。默认大小为 10。	10	整数
camel.resilience4j.sliding-window-size	配置滑动窗口的大小，该窗口用于在 CircuitBreaker 关闭时记录调用的结果。slidingWindowSize 配置滑动窗口的大小。滑动窗口可以是基于计数或基于时间的窗口。如果 slidingWindowType 是 COUNT_BASED，则最后的 slidingWindowSize 调用会被记录并聚合。如果 slidingWindowType 是 TIME_BASED，则最后 slidingWindowSize 秒的调用会被记录并聚合。slidingWindowSize 必须大于 0。minimumNumberOfCalls 必须大于 0。如果 slidingWindowType 是 COUNT_BASED，则 minimumNumberOfCalls 不能大于 slidingWindowSize。如果 slidingWindowType 是 TIME_BASED，您可以选择任何您需要的。默认 slidingWindowSize 为 100。	100	整数

Name	描述	默认值	类型
camel.resilience4j.sliding-window-type	配置滑动窗口的类型，用于记录 CircuitBreaker 关闭时调用的结果。滑动窗口可以是基于计数或基于时间的窗口。如果 slidingWindowType 是 COUNT_BASED，则最后的 slidingWindowSize 调用会被记录并聚合。如果 slidingWindowType 是 TIME_BASED，则最后 slidingWindowSize 秒的调用会被记录并聚合。默认 slidingWindowType 是 COUNT_BASED。	COUNT_BASED	字符串
camel.resilience4j.slow-call-duration-threshold	配置上面的持续时间阈值（秒），调用被视为缓慢，并增加较慢的调用百分比。默认值为 60 秒。	60	整数
camel.resilience4j.slow-call-rate-threshold	以百分比为单位配置阈值。当调用持续时间大于 slowCallDurationThreshold Duration 时，CircuitBreaker 会将调用视为较慢。当较慢的调用百分比相等或大于阈值时，CircuitBreaker 过渡到 open，并启动短路调用。阈值必须大于 0，而不是大于 100。默认值为 100 百分比，这意味着所有记录的调用都必须比 slowCallDurationThreshold 慢。		æµ®ç‚¹â€¼
camel.resilience4j.wait-duration-in-open-state	配置等待持续时间（以秒为单位），指定 CircuitBreaker 应该保持打开的时间，然后再切换到半次。默认值为 60 秒。	60	整数
camel.resilience4j.writable-stack-trace-enabled	启用可写入堆栈跟踪。当设置为 false 时，Exception.getStackTrace 返回一个零长度数组。当断路器处于开路状态时，这可用于减少日志垃圾邮件，因为存在例外的原因（断路器是短路调用）。	true	布尔值
camel.rest.api-component	用作 REST API 的 Camel 组件名称（如 swagger）如果没有明确配置 API 组件，则 Camel 会查找负责服务并生成 REST API 文档的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestApiProcessorFactory。如果找到其中任何一个，则使用它。		字符串
camel.rest.api-context-path	设置领导的 API 上下文路径将使用的 REST API 服务。这在使用使用 context-path 部署 Web 应用程序的组件（如 camel-servlet ）时使用它。		字符串
camel.rest.api-context-route-id	设置用于服务 REST API 的路由的路由 ID。默认情况下，路由将使用自动分配的路由 ID。		字符串
camel.rest.api-host	要将特定主机名用于 API 文档（如 swagger），这可用于用这个配置的主机名覆盖生成的主机。		字符串

Name	描述	默认值	类型
camel.rest.api-property	允许为 api 文档配置任意数量的附加属性(swagger)。例如，将属性 api.title 设置为我的冷却。		Map
camel.rest.api-vendor-extension	是否在 Rest API 中启用供应商扩展。如果启用，Camel 将包含额外信息作为厂商扩展名（例如，以 x- 开头的键），如路由 ID、类名称等。在导入 API 文档时，并非所有第三方 API 网关和工具都支持 vendor-extensions。	false	布尔值
camel.rest.binding-mode	设置要使用的绑定模式。默认值为 off。		RestBindingMode
camel.rest.client-request-validation	是否启用客户端请求验证，以检查客户端的 Content-Type 和 Accept 标头是否受到其 consume/produces 设置的 Rest-DSL 配置的支持。这可以打开，以启用此检查。如果验证错误，则返回 HTTP Status code 415 或 406。默认值为 false。	false	布尔值
camel.rest.component	用于 REST 传输(consumer)的 Camel Rest 组件，如 netty-http, jetty, servlet, undertow。如果没有明确配置组件，则 Camel 会查找是否有与 Rest DSL 集成的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestConsumerFactory。如果找到其中任何一个，则使用它。		字符串
camel.rest.component-property	允许为正在使用的其他组件配置任意数量的附加属性。		Map
camel.rest.consumer-property	允许为使用中的其他使用者配置任意数量的附加属性。		Map
camel.rest.context-path	设置 REST 服务将使用的前导上下文路径。这在使用使用 context-path 部署 Web 应用程序的组件（如 camel-servlet ）时使用它。或者对于包含 HTTP 服务器的 camel-jetty 或 camel-netty-http 等组件。		字符串
camel.rest.cors-headers	允许配置自定义 CORS 标头。		Map
camel.rest.data-format-property	允许为使用的数据格式配置多个额外属性。例如，将属性 prettyPrint 设置为 true，以便以用户友善模式输出 json。属性可以加上前缀来表示选项仅适用于 JSON 或 XML，以及 IN 或 OUT。前缀为： json.in. json.out. xml.in. xml.out。例如，值为 xml.out.mustBeJAXBElement 的键仅用于传出的 XML 数据格式。没有前缀的密钥是所有情况的通用密钥。		Map

Name	描述	默认值	类型
camel.rest.enable-cors	是否在 HTTP 响应中启用 CORS 标头。默认值为 false。	false	布尔值
camel.rest.endpoint-property	允许为使用中的其他端点配置多个额外的属性。		Map
camel.rest.host	用于公开 REST 服务的主机名。		字符串
camel.rest.host-name-resolver	如果没有明确配置的主机名，这个 resolver 会用于计算 REST 服务将要使用的主机名。		RestHostNameResolver
camel.rest.json-data-format	要使用的特定 json 数据格式的名称。默认将使用 json-jackson。重要：此选项仅用于设置数据格式的自定义名称，而不是引用现有数据格式实例。		字符串
camel.rest.port	用于公开 REST 服务的主机名。请注意，如果您使用 servlet 组件，则此处配置的端口号不适用，因为使用中的端口号是 servlet 组件使用的实际端口号。例如，如果使用 Apache Tomcat，它的 tomcat http 端口，如果使用 Apache Karaf，它的在 Karaf 中的 HTTP 服务，它默认使用端口 8181。虽然在这些情况下，这里设置端口号，但允许工具和 JMX 知道端口号，因此建议将端口号设置为 servlet 引擎使用的数字。		字符串
camel.rest.producer-api-doc	设置 api 文档的位置，REST 生成者将根据这个文档来验证 REST uri 和查询参数是否有效。这需要将 camel-swagger-java 添加到 classpath 中，任何缺失的配置都会导致 Camel 在启动时失败并报告错误。默认情况下从 classpath 加载的 api 文档的位置，但您可以使用 file: 或 http: 引用从文件或 http url 加载的资源。		字符串
camel.rest.producer-component	设置要用作 REST 生成者的 Camel 组件的名称。		字符串
camel.rest.scheme	用于公开 REST 服务的方案。通常支持 http 或 https。默认值为 http。		字符串
camel.rest.skip-binding-on-error-code	如果存在自定义 HTTP 错误代码标头，是否跳过输出绑定。这允许构建没有绑定到 json / xml 等自定义错误消息，否则成功信息会这样做。	false	布尔值
camel.rest.use-x-forward-headers	是否将 X-Forward 标头用于主机和相关设置。默认值为 true。	true	布尔值
camel.rest.xml-data-format	要使用的特定 XML 数据格式的名称。默认情况下将使用 jaxb。重要：此选项仅用于设置数据格式的自定义名称，而不是引用现有数据格式实例。		字符串

Name	描述	默认值	类型
camel.rest.api-context-id-pattern	**弃用** 设置 CamelContext id 特征，以只允许 CamelContext 中名称与特征匹配的其他服务的 Rest API。特征 name 指的是 CamelContext 名称，仅匹配当前的 CamelContext。对于任何其他值，特征使用来自 PatternHelper#matchPattern (String,String)的规则。		字符串
camel.rest.api-context-listing	**弃用** 设置是否启用了 JVM 中带有 REST 服务的所有可用 CamelContext 的列表。如果启用，它将允许发现这些上下文，如果为 false，则只使用当前的 CamelContext。	false	布尔值

第 42 章 HL7

HL7 组件用于使用 HL7 MLLP 协议和 HL7 v2 消息，使用 HAPI 库。

这个组件支持以下组件：

- 用于 Mina的 HL7 MLLP codec
- 用于 Netty的 HL7 MLLP codec
- 类型 Converter from/to HAPI 和 String
- HL7 DataFormat 使用 HAPI 库

42.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 hl7 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-hl7-starter</artifactId>
</dependency>
```

42.2. HL7 MLLP 协议

HL7 通常与 HL7 MLLP 协议一起使用，它是基于文本的 TCP 套接字协议。此组件提供了 Mina 和 Netty Codec，它符合 MLLP 协议，以便您可以轻松地公开可以接受通过 TCP 传输层的 HL7 请求的 HL7 侦听器。要公开 HL7 侦听器服务，camel-mina 或 link:camel-netty 组件与 HL7MLLPCodec (mina)或 HL7MLLPNettyDecoder/HL7MLLPNettyEncoder (Netty)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 默认值	用于 codec 的编码（费用名称）。如果没有提供，Camel 将使用 JVM 默认 Charset。
produceString	**true**	如果为 true，则 codec 使用定义的 charset 创建一个字符串。如果为 false，则 codec 会将普通字节数组发送到路由，以便 HL7 数据格式可以决定 HL7 消息内容中的实际 charset。
convertLFtoCR	**false**	将 **\n** 转换为 **\r** (**0x0d**, 13 十进制)作为 HL7 stipulates**\r** as segment terminators。HAPI 库需要使用 **\r**。

42.2.1. 使用 Mina 公开 HL7 侦听器

在 Spring XML 文件中，我们将 mina 端点配置为使用 TCP 在端口 8888 上侦听 HL7 请求：

```
<endpoint id="hl7MinaListener" uri="mina:tcp://localhost:8888?sync=true&amp;codec=#hl7codec"/>
```

sync=true 表示此监听器是同步的，因此会将 HL7 响应返回给调用者。HL7 codec 使用 codec=#hl7codec 设置。请注意，hl7codec 只是一个 Spring bean ID，因此它可以命名为 mygreatcodecforhl7 或 any。codec 也在 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 hl7MinaLlistener 可以在路由中使用，因为此 Java DSL 示例所示：

```
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 bean ID，在 Spring XML 中配置，如下所示：

```
<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
    }
```

42.2.2. 使用 Netty 公开 HL7 侦听器（从 Camel 2.15 开始可用）

在 Spring XML 文件中，我们将 netty 端点配置为使用 TCP 在端口 8888 上侦听 HL7 请求：

```
<endpoint id="hl7NettyListener" uri="netty:tcp://localhost:8888?
sync=true&amp;encoders=#hl7encoder&amp;decoders=#hl7decoder"/>
```

sync=true 表示此监听器是同步的，因此会将 HL7 响应返回给调用者。HL7 codec 使用 encoders=#hl7encoder*and*decoders=#hl7decoder 设置。请注意，hl7encoder 和 hl7decoder 只是 bean ID，因此它们可以被不同命名。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 可用作消费者的路由，如 Java DSL 示例所示：

```
from("hl7NettyListener")
  .bean("patientLookupService");
```

42.3. 使用 JAVA.LANG.STRING 或 BYTE[] 的 HL7 MODEL

HL7 MLLP codec 使用普通 String 作为其数据格式。Camel 使用其 Type Converter 将字符串转换为 HAPI HL7 模型对象，但如果您想要自行解析数据，您可以使用 plain String 对象。

您还可以通过将 produceString 属性设置为 false，使 Mina 和 Netty codecs 使用 plain byte[] 作为

其数据格式。Type Converter 也可以将 byte[] 转换为/来自 HAPI HL7 模型对象。

42.4. 使用 HAPI 的 HL7V2 MODEL

HL7v2 模型使用来自 HAPI 库的 Java 对象。使用这个库，您可以对大多数 HL7v2 使用的 EDI 格式 (ER7)进行编码和解码。

以下示例是通过病人 ID 0101701234 来查找病人请求。

```
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，您可以处理 ca.uhn.hl7v2.model.Message 对象，例如检索病人 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 侦听器结合使用时很强大，因为您不必使用 byte[]、String 或任何其他简单的对象格式。您只能使用 HAPI HL7v2 模型对象。如果您事先知道消息类型，则可以是更多 type-safe ：

```
QRY_A19 msg = exchange.getIn().getBody(QRY_A19.class);
String patientId = msg.getQRD().getWhoSubjectFilter(0).getIDNumber().getValue();
```

42.5. HL7 DATAFORMAT

camel-hl7 JAR 附带 HL7 数据格式，可用于 marshal 或 unmarshal HL7 模型对象。

HL7 dataformat 支持 1 个选项，如下所列。

Name	默认值	Java 类型	描述
validate		布尔值	默认情况下，是否验证 HL7 消息 Is true。

- marshal = 从消息到字节流（可以使用 HL7 MLLP codec 进行响应）

- unmarshal = 从字节流到消息（可以在从 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");
```

在上面的示例中，HL7 从 HAPI Message 对象放入一个字节流并放在 JMS 队列中。
下一个示例是相反的：

```
DataFormat hl7 = new HL7DataFormat();

from("jms:queue:hl7out")
  .unmarshal(hl7)
  .to("patientLookupService");
```

在这里，我们将字节流传输到 HAPI 消息对象，该对象传递到我们的病人查找服务。

42.5.1. 片段分隔符

unmarshalling 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#convertLFToCR 为这一目的提供了方便的表达式。

42.5.2. charset

marshal 和 unmarshal 评估在字段 MSH-18 中提供的 charset。如果此字段为空，则默认假定对应的 Camel charset 属性/header 中包含的 charset。从 HL7DataFormat 类继承时，您甚至可以通过覆盖 guessCharsetName 方法来更改此默认行为。

Camel 中有一个简写语法，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");
```

42.6. 消息标头

unmarshal 操作将 MSH 段中的这些字段添加为 Camel 消息上的标头：

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
CamelHL7TriggerEvent	MSH-9-2	A01
CamelHL7MessageControl	MSH-10	1234
CamelHL7ProcessingId	MSH-11	P
CamelHL7VersionId	MSH-12	2.4
CamelHL7Context	``	包含用于解析消息的

键	MSH 字段	示例
CamelHL7Charset	MSH-18	UNICODE UTF-8

CamelHL7Context 以外的所有标头都是 String 类型。如果缺少标头值，则其值为 null。

42.7. 依赖项

要在 Camel 路由中使用 HL7，您需要添加上面列出的 camel-hl7 的依赖，它实现了这个数据格式。

HAPI 库被分成一个库和几个结构 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 只引用 HAPI 基础库。应用程序负责包括结构库本身。例如，如果应用程序可

用于 HL7v2 消息版本 2.4 和 2.5，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、所有结构库和所需依赖项（在捆绑包类路径上）可以从 中央 Maven 存储库下载。

```
<dependency>
  <groupId>ca.uhn.hapi</groupId>
  <artifactId>hapi-osgi-base</artifactId>
  <version>2.2</version>
</dependency>
```

42.8. 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.hl7.enabled	是否启用 hl7 数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.hl7.validate	默认情况下，是否验证 HL7 消息 Is true。	true	布尔值
camel.language.hl7terser.enabled	是否启用 hl7terser 语言的自动配置。这默认是启用的。		布尔值
camel.language.hl7terser.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

第 43 章 HTTP

仅支持生成者

HTTP 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）。

43.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 http 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-http-starter</artifactId>
</dependency>
```

43.2. URI 格式

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

默认情况下，将端口 80 用于 HTTP，对于 HTTPS 使用 443。

43.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

43.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

43.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

43.4. 组件选项

HTTP 组件支持 37 选项，如下所列。

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，它是一个仅内存的 Cookie 存储。请注意，如果 bridgeEndpoint=true，则 Cookie 存储被强制为 noop cookie 存储，因为 Cookie 不应存储，因为我们只是桥接（如代理）。		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 交换标头中。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

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
responsePayloadStreamingThreshold (producer)	此阈值（以字节为单位）控制响应有效负载应存储在内存中作为字节数阵列还是基于流传输。把它设置为 -1 以始终使用流模式。	8192	int
skipRequestHeaders (producer (advanced))	是否跳过将所有 Camel 标头映射为 HTTP 请求标头。如果 HTTP 请求中没有来自 Camel 标头的数据，可以避免解析 JVM 垃圾收集器有很多对象分配的开销。	false	布尔值
skipResponseHeaders (producer (advanced))	是否跳过将所有 HTTP 响应标头映射到 Camel 标头。如果 HTTP 标头不需要数据，这样可避免解析 JVM 垃圾收集器的多个对象分配的开销。	false	布尔值
allowJavaSerializedObject (advanced)	当请求使用 context-type=application/x-java-serialized-object 时，是否允许 java 序列化。默认情况下是关闭的。如果启用此选项，则 Java 会将传入数据从请求反序列化到 Java，这可能会成为潜在的安全风险。	false	布尔值
authCachingDisabled (advanced)	禁用身份验证方案缓存。	false	布尔值
automaticRetriesDisabled (advanced)	禁用自动请求恢复并重新执行。	false	布尔值
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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。		HttpClientConnectionManager
connectionsPerRoute (advanced)	每个路由的最大连接数。	20	int

Name	描述	默认值	类型
connectionStateDisabled (advanced)	禁用连接状态跟踪。	false	布尔值
connectionTimeToLive (advanced)	与实时连接的时间，时间单位为 millisecond，默认值为始终保持活动状态。		long
ContentCompressionDisabled (advanced)	禁用自动内容解压缩。	false	布尔值
cookieManagementDisabled (advanced)	禁用状态(cookie)管理。	false	布尔值
defaultUserAgentDisabled (advanced)	如果用户未提供任何用户代理，则禁用此构建器设置的默认用户代理。	false	布尔值
httpBinding (advanced)	使用自定义 HttpBinding 来控制 Camel 消息和 HttpClient 之间的映射。		HttpBinding
httpClientConfigurer (advanced)	要使用自定义 HttpClientConfigurer 执行要使用的 HttpClientConfigurer 的配置。		HttpClientConfigurer
httpConfiguration (advanced)	将共享的 HttpConfiguration 用作基础配置。		HttpConfiguration
httpContext (advanced)	在执行请求时使用自定义 org.apache.http.protocol.HttpContext。		HttpContext
maxTotalConnections (advanced)	连接的最大数量。	200	int
redirectHandlingDisabled (advanced)	禁用自动重定向处理。	false	布尔值
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头到 Camel 消息。		HeaderFilterStrategy
proxyAuthDomain (proxy)	要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

Name	描述	默认值	类型
proxyAuthMethod (proxy)	要使用的代理验证方法。 Enum 值： • 基本的 • 摘要 • NTLM		字符串
proxyAuthNtHost (proxy)	与 NTML 搭配使用的代理身份验证域(workstation 名称)。		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。		整数
proxyAuthUsername (proxy)	代理身份验证用户名。		字符串
sslContextParameters (security)	使用 SSLContextParameters 配置安全性。重要： 每个 HttpComponent 仅支持 org.apache.camel.support.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
useGlobalSslContextParameters (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 NoopHostnameVerifier。		HostnameVerifier
connectionRequestTimeout (timeout)	从连接管理器请求连接时使用的超时时间（毫秒）。超时值为零被解释为无限超时。超时值为零被解释为无限超时。负值解释为未定义（系统默认值）。	-1	int
connectTimeout (timeout)	决定连接建立前的超时时间（毫秒）。超时值为零被解释为无限超时。超时值为零被解释为无限超时。负值解释为未定义（系统默认值）。	-1	int

Name	描述	默认值	类型
socketTimeout (timeout)	定义套接字超时（以毫秒为单位），这是等待数据的超时时间，或者以不同方式放置，在两个连续的数据数据包之间处于不活跃状态。超时值为零被解释为无限超时。负值解释为未定义（系统默认值）。	-1	int

43.5. 端点选项

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：

43.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
httpUri (common)	**必需** 要调用的 HTTP 端点的 url。		URI

43.5.2. 查询参数(51 参数)

Name	描述	默认值	类型
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
disableStreamCache (common)	确定来自 Servlet 的原始输入流是否被缓存(Camel 会将流读取到内存/流到文件、流缓存)缓存。默认情况下，Camel 将缓存 Servlet 输入流来支持多次读取它，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，您可以将这个选项设置为 true，比如将其直接流传输到文件或其他持久性存储。DefaultHttpBinding 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，以支持多次读取流。如果您使用 Servlet 桥接/代理，则考虑启用这个选项以提高性能，以防您不需要多次读取消息有效负载。http producer 默认将缓存响应正文流。如果将此选项设置为 true，则制作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 过滤标头到 Camel 消息。		HeaderFilterStrategy

Name	描述	默认值	类型
httpBinding (common (advanced))	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，HttpProducer 将忽略 Exchange.HTTP_URI 标头，并使用端点的 URI 请求。您还可以将选项 throwExceptionOnFailure 设置为 false，以让 HttpProducer 发送所有故障响应。	false	布尔值
clearExpiredCookies (producer)	在发送 HTTP 请求前，是否清除已过期的 Cookie。这样可确保 Cookie 存储不会通过添加新的 Cookie 来保持增长，这些 Cookie 在其过期时会被删除。如果组件禁用了 Cookie 管理，则此选项也会被禁用。	true	布尔值
connectionClose (producer)	指定是否需要将 Connection Close 标头添加到 HTTP 请求。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 交换标头中。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
customHostHeader (producer)	将自定义主机标头用于制作者。如果没有在查询中设置，则忽略。当设置时，将覆盖从 url 派生的主机标头。		字符串
httpMethod (producer)	配置要使用的 HTTP 方法。如果设置，HttpMethod 标头无法覆盖这个选项。 Enum 值： • GET • POST • PUT • DELETE • HEAD • 选项 • TRACE • PATCH		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

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
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，用于您希望下游服务器接收的主机标头来反映上游客户端调用的 URL。	false	布尔值
throwExceptionOnFailure (producer)	如果远程服务器失败响应，禁用禁用 HttpOperationFailedException 的选项。这样，无论 HTTP 状态代码是什么，您都可以获得所有响应。	true	布尔值
transferException (producer)	如果在消费者端启用并且 Exchange 失败，如果导致的 Exception 被发送序列化为 application/x-java-serialized-object 内容类型。在生成者一侧，异常将被反序列化和抛出，而不是 HttpOperationFailedException。导致异常需要被序列化。默认情况下是关闭的。如果启用此选项，则 Java 会将传入数据从请求反序列化到 Java，这可能会成为潜在的安全风险。	false	布尔值
cookieHandler (producer (advanced))	配置 Cookie 处理程序来维护 HTTP 会话。		CookieHandler
cookieStore (producer (advanced))	使用自定义 CookieStore。默认情况下，使用 BasicCookieStore，它是一个仅内存 Cookie 存储。请注意，如果 bridgeEndpoint=true，则 Cookie 存储被强制为 noop cookie 存储，因为 Cookie 不应存储，因为我们只是桥接（如代理）。如果设置了 CookieHandler，则 Cookie 存储也强制作为 Cookie 处理的 noop cookie 存储，然后由 CookieHandler 执行。		CookieStore
deleteWithBody (producer (advanced))	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 正文。但是，在一些罕见的情况下，用户可能需要包含消息正文。	false	布尔值
getWithBody (producer (advanced))	HTTP GET 是否应包含消息正文。默认情况下，HTTP GET 不包含任何 HTTP 正文。但是，在一些罕见的情况下，用户可能需要包含消息正文。	false	布尔值

Name	描述	默认值	类型
okStatusCodeRange (producer (advanced))	被视为成功响应的状态代码。这些值包含为。可以定义多个范围，用逗号分开，如 200-204,209,301-304。每个范围都必须是一个数字或 from-to，包括横线。	200-299	字符串
skipRequestHeaders (producer (advanced))	是否跳过将所有 Camel 标头映射为 HTTP 请求标头。如果 HTTP 请求中没有来自 Camel 标头的数据，可以避免解析 JVM 垃圾收集器有很多对象分配的开销。	false	布尔值
skipResponseHeaders (producer (advanced))	是否跳过将所有 HTTP 响应标头映射到 Camel 标头。如果 HTTP 标头不需要数据，这样可避免解析 JVM 垃圾收集器的多个对象分配的开销。	false	布尔值
userAgent (producer (advanced))	设置自定义 HTTP User-Agent 请求标头。		字符串
ClientBuilder (advanced)	提供对此端点的生产者或消费者使用的新 RequestConfig 实例中使用的 http 客户端请求参数的访问权限。		HttpClientBuilder
clientConnectionManager (advanced)	使用自定义 HttpClientConnectionManager 来管理连接。		HttpClientConnectionManager
connectionsPerRoute (advanced)	每个路由的最大连接数。	20	int
httpclient (advanced)	设置自定义 HttpClient，供制作者使用。		HttpClient
httpClientConfigurer (advanced)	为制作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)	若要利用 map 中的键/值来配置 HttpClient。		Map
httpContext (advanced)	使用自定义 HttpContext 实例。		HttpContext
maxTotalConnections (advanced)	连接的最大数量。	200	int
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

Name	描述	默认值	类型
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。 Enum 值： ● 基本的 ● 摘要 ● NTLM		字符串
proxyAuthNtHost (proxy)	与 NTML 搭配使用的代理身份验证域(workstation 名称)。		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)	要使用的代理验证方案。 Enum 值： ● http ● https		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
authDomain (security)	与 NTML 一起使用的身份验证域。		字符串
authenticationPreemptive (security)	如果此选项为 true，则 camel-http 会将抢占性基本身份验证发送到服务器。	false	布尔值
authHost (security)	与 NTML 一起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隔的值 Basic、Digest 或 NTLM 的列表。		字符串

Name	描述	默认值	类型
authMethodPriority (security)	哪个身份验证方法优先使用，可以是 Basic、Digest 或 NTLM。 Enum 值： • 基本的 • 摘要 • 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 仅支持一个 org.apache.camel.util.jsse.SSLContextParameters。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 NoopHostnameVerifier。		HostnameVerifier

43.6. 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。这个 uri 是要调用的 http 服务器的 uri。它和 Camel 端点 uri 不同，您可以在其中配置端点选项，如安全性等。此标头不支持它，它只是 http 服务器的 URI。
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。200 代表 OK。

Name	类型	描述
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。设置 IN 和 OUT 消息上以提供内容类型，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**。

43.7. 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。来自 IN 消息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。此外，Camel 将添加 HTTP 响应标头以及 OUT 消息标头。

43.8. 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

43.9. 响应代码

Camel 将根据 HTTP 响应代码处理：

- 响应代码范围为 100..299，Camel 会将它视为成功响应。
- 响应代码在范围 300..399中，Camel 会将它视为重定向响应，并将引发带有信息的 HttpOperationFailedException。
-

响应代码为 400+，Camel 会将它视为外部服务器失败，并将引发带有信息的 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。
下面是一个演示示例。

43.10. 例外

HttpOperationFailedException 异常包含以下信息：

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

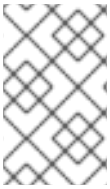
43.11. 将使用哪些 HTTP 方法

以下算法用于确定应使用的 HTTP 方法：
1。使用作为端点配置(httpMethod)的方法。
2.使用在标头中提供的方法(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 如果存在要发送的数据（用户不是 null）。
6.否则 GET。

43.12. 如何访问 HTTPSERVLETREQUEST 和 HTTPSERVLETRESPONSE

您可以使用以下方法使用 Camel 类型转换器系统访问这两个

```
HttpServletRequest request = exchange.getIn().getBody(HttpServletRequest.class);
HttpServletResponse response = exchange.getIn().getBody(HttpServletResponse.class);
```



注意

您只能在 camel-jetty 或 camel-cxf 端点后从处理器获取请求和响应。

43.13. 配置 URI 来调用

您可以直接设置 HTTP producer 的 URI 组成端点 URI。在以下路由中，Camel 将使用 HTTP 调用外部服务器 oldhost。

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

在上面的 Camel 示例中，尽管端点配置了 http://newhost/，但其端点 会调用 http://oldhost/。如果 http 端点在网桥模式下工作，它将忽略 Exchange.HTTP_URI 的消息标头。

43.14. 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，也可以是消息上密钥 Exchange.HTTP_QUERY 的标头。

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

或标头中提供的选项：

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

43.15. 如何将 HTTP 方法(GET/PATCH/POST/PUT/DELETE/HEAD/OPTIONS/TRACE)设置为 HTTP PRODUCER

HTTP 组件提供了一种通过设置消息标头来设置 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.http.HttpMethods.POST))
  .to("http://www.google.com")
  .to("mock:results");
```

可以使用字符串常量编写时间较短：

```
.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 name="CamelHttpMethod">
        <constant>POST</constant>
    </setHeader>
    <to uri="http://www.google.com"/>
    <to uri="mock:results"/>
  </route>
</camelContext>
```

43.16. 使用客户端超时 - SO_TIMEOUT

请参阅 **HttpSOTimeoutTest** 单元测试。

43.17. 配置代理

HTTP 组件提供了一种配置代理的方法。

```
from("direct:start")
  .to("http://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。

43.17.1. 使用 URI 之外的代理设置

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.getGlobalOptions().put("http.proxyHost", "172.168.18.9");
context.getGlobalOptions().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 首先设置来自 Java System 或 CamelContext Properties 的设置，如果提供，则端点代理选项。
因此，您可以使用端点选项覆盖系统属性。

还有一个 http.proxyScheme 属性，您可以设置为显式配置要使用的方案。

43.18. 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");
```

43.18.1. 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("http://www.google.com")
  .setHeader(FileComponent.HEADER_FILE_NAME, "message.html")
  .to("file:target/google");
```

43.18.2. 来自端点 URI 的 URI 参数

在这个示例中，我们拥有完整的 URI 端点，只是您在 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 这里没有问题。

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

43.18.3. 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("http://www.google.com/search", null, headers);
```

在上面的标头值中，它不应 以 ? 前缀，您可以像使用 & amp; char 来分隔参数。

43.18.4. 获取响应代码

您可以通过使用 Exchange.HTTP_RESPONSE_CODE 从 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);
```

43.19. 禁用 COOKIE

要禁用 Cookie，您可以通过添加以下 URI 选项将 HTTP 客户端设置为忽略 Cookie：

```
httpClient.cookieSpec=ignore
```

43.20. 带有流消息正文的基本身份验证

为了避免 NonRepeatableRequestException，您需要通过添加选项 authenticationPreemptive=true来进行 Preemptive Basic Authentication

43.21. 高级用法

如果您需要对 HTTP 生成者进行更多控制，您应该使用 HttpComponent，您可以在其中设置各种类来为您提供自定义行为。

43.21.1. 为 HTTP 客户端设置 SSL

使用 JSSE 配置实用程序

HTTP 组件通过 Camel JSSE 配置实用程序 支持 SSL/TLS 配置实用程序。这个工具大大减少了您需要写入的组件特定代码数量，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 HTTP 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("https", HttpComponent.class);
httpComponent.setSslContextParameters(scp);
```

基于 Spring DSL 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="https://127.0.0.1/mail/?sslContextParameters=#sslContextParameters"/>
```

直接配置 Apache HTTP 客户端

camel-http 组件基本上基于 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.http.HttpsServerTestSupport 单元测试基本类。
如果需要完全控制，您也可以实施自定义 org.apache.camel.component.http.HttpClientConfigurer 在 http 客户端上进行一些配置。

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 https 协议，提供上例中的密钥存储或信任存储。然后，在 camel route builder 类中可以 hook 类似如下：

```
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?httpClientConfigurer=myHttpClientConfigurer"/>
```

只要您实施 HttpClientConfigurer，并且配置密钥存储和信任存储，它就可以正常工作。

使用 HTTPS 验证 getchas

最终用户报告他在使用 HTTPS 进行身份验证时遇到了问题。这个问题最终通过提供自定义配置的 org.apache.http.protocol.HttpContext 来解决：

- 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 HttpContext：

```
<bean id="myHttpContext" factory-bean="httpContextFactory" factory-method="getObject"/>
```

- 3.引用 http URL 中的上下文：

```
<to uri="https://myhostname.com:443/myURL?httpContext=myHttpContext"/>
```

使用不同的 SSLContextParameters

HTTP 组件只支持每个组件的 org.apache.camel.support.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 HTTP 组件，如下所示。在我们有 2 个组件时，每个组件都使用自己的 sslContextParameters 属性实例。

```
<bean id="http-foo" class="org.apache.camel.component.http.HttpComponent">
  <property name="sslContextParameters" ref="sslContextParams1"/>
  <property name="x509HostnameVerifier" ref="hostnameVerifier"/>
</bean>
```

```
<bean id="http-bar" class="org.apache.camel.component.http.HttpComponent">
  <property name="sslContextParameters" ref="sslContextParams2"/>
  <property name="x509HostnameVerifier" ref="hostnameVerifier"/>
</bean>
```

43.22. 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.http.allow-java-serialized-object	当请求使用 context-type=application/x-java-serialized-object 时，是否允许 java 序列化。默认情况下是关闭的。如果启用此选项，则 Java 会将传入数据从请求反序列化到 Java，这可能会成为潜在的安全风险。	false	布尔值
camel.component.http.auth-caching-disabled	禁用身份验证方案缓存。	false	布尔值
camel.component.http.automatic-retries-disabled	禁用自动请求恢复并重新执行。	false	布尔值
camel.component.http.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.http.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。		HttpClientConnectionManager
camel.component.http.connect-timeout	决定连接建立前的超时时间（毫秒）。超时值为零被解释为无限超时。超时值为零被解释为无限超时。负值解释为未定义（系统默认值）。	-1	整数
camel.component.http.connection-request-timeout	从连接管理器请求连接时使用的超时时间（毫秒）。超时值为零被解释为无限超时。超时值为零被解释为无限超时。负值解释为未定义（系统默认值）。	-1	整数
camel.component.http.connection-state-disabled	禁用连接状态跟踪。	false	布尔值

Name	描述	默认值	类型
camel.component.http.connection-time-to-live	与实时连接的时间，时间单位为 millisecond，默认值为始终保持活动状态。		Long
camel.component.http.connections-per-route	每个路由的最大连接数。	20	整数
camel.component.http.content-compression-disabled	禁用自动内容解压缩。	false	布尔值
camel.component.http.cookie-management-disabled	禁用状态(cookie)管理。	false	布尔值
camel.component.http.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，它是一个仅内存的 Cookie 存储。请注意，如果 bridgeEndpoint=true，则 Cookie 存储被强制为 noop cookie 存储，因为 Cookie 不应存储，因为我们只是桥接（如代理）。选项是 org.apache.http.client.CookieStore 类型。		CookieStore
camel.component.http.copy-headers	如果此选项为 true，则 IN Exchange 标头将根据复制策略复制到 OUT 交换标头中。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
camel.component.http.default-user-agent-disabled	如果用户未提供任何用户代理，则禁用此构建器设置的默认用户代理。	false	布尔值
camel.component.http.enabled	是否启用 http 组件的自动配置。这默认是启用的。		布尔值
camel.component.http.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头到 Camel 消息。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		HeaderFilterStrategy
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。		HttpBinding

Name	描述	默认值	类型
camel.component.http.http-client-configurer	要使用自定义 HttpClientConfigurer 执行要使用的 HttpClientConfigurer 的配置。选项是 org.apache.camel.component.http.HttpClientConfigurer 类型。		HttpClientConfigurer
camel.component.http.http-configuration	将共享的 HttpConfiguration 用作基础配置。选项是 org.apache.camel.http.common.HttpConfiguration 类型。		HttpConfiguration
camel.component.http.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。		HttpContext
camel.component.http.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.http.max-total-connections	连接的最大数量。	200	整数
camel.component.http.proxy-auth-domain	要使用的代理身份验证域。		字符串
camel.component.http.proxy-auth-host	代理身份验证主机。		字符串
camel.component.http.proxy-auth-method	要使用的代理验证方法。		字符串
camel.component.http.proxy-auth-nt-host	与 NTML 搭配使用的代理身份验证域(workstation 名称)。		字符串
camel.component.http.proxy-auth-password	代理身份验证密码。		字符串
camel.component.http.proxy-auth-port	代理身份验证端口。		整数

Name	描述	默认值	类型
camel.component.http.proxy-auth-username	代理身份验证用户名。		字符串
camel.component.http.redirect-handling-disabled	禁用自动重定向处理。	false	布尔值
camel.component.http.response-payload-streaming-threshold	此阈值（以字节为单位）控制响应有效负载应存储在内存中作为字节数阵列还是基于流传输。把它设置为 -1 以始终使用流模式。	8192	整数
camel.component.http.skip-request-headers	是否跳过将所有 Camel 标头映射为 HTTP 请求标头。如果 HTTP 请求中没有来自 Camel 标头的数据，可以避免解析 JVM 垃圾收集器有很多对象分配的开销。	false	布尔值
camel.component.http.skip-response-headers	是否跳过将所有 HTTP 响应标头映射到 Camel 标头。如果 HTTP 标头不需要数据，这样可避免解析 JVM 垃圾收集器的多个对象分配的开销。	false	布尔值
camel.component.http.socket-timeout	定义套接字超时（以毫秒为单位），这是等待数据的超时时间，或者以不同方式放置，在两个连续的数据数据包之间处于不活跃状态。超时值为零被解释为无限超时。负值解释为未定义（系统默认值）。	-1	整数
camel.component.http.ssl-context-parameters	使用 SSLContextParameters 配置安全性。重要： 每个 HttpComponent 仅支持 org.apache.camel.support.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.support.jsse.SSLContextParameters 类型。		SSLContextParameters
camel.component.http.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值
camel.component.http.x509-hostname-verifier	使用自定义 X509HostnameVerifier，如 DefaultHostnameVerifier 或 NoopHostnameVerifier。选项是 javax.net.ssl.HostnameVerifier 类型。		HostnameVerifier

第 44 章 INFINISPAN

支持生成者和消费者

此组件允许您使用 Hot Rod procol 与 Infinispan 分布式数据网格/缓存交互。Infinispan 是一个非常可扩展、高度可用的键/值数据存储和 Java 编写的数据网格平台。

44.1. 依赖项

当在 Camel Spring Boot 中使用 infinispan 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-infinispan-starter</artifactId>
</dependency>
```

44.2. URI 格式

```
infinispan://cacheName?[options]
```

生产者允许使用 HotRod 协议将消息发送到远程缓存。消费者允许使用 HotRod 协议从远程缓存侦听事件。

44.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

44.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

44.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

44.4. 组件选项

Infinispan 组件支持 26 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	组件配置.		InfinispanRemoteConfiguration
hosts (common)	指定 Infinispan 实例上缓存的主机。		字符串
queryBuilder (common)	指定查询构建器。		InfinispanQueryBuilder
secure (common)	定义是否连接到安全 Infinispan 实例。	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
customListener (consumer)	如果提供，则返回正在使用的自定义监听程序。		InfinispanRemoteCustomListener
eventTypes (consumer)	指定要由消费者注册的事件类型集合。Multiple 事件可以用逗号分开。可能的事件类型是：CLIENT_CACHE_ENTRY_CREATED, CLIENT_CACHE_ENTRY_MODIFIED, CLIENT_CACHE_ENTRY_REMOVED, CLIENT_CACHE_ENTRY_EXPIRED, CLIENT_CACHE_FAILOVER。		字符串
DefaultValue (producer)	为一些制作者操作设置特定的默认值。		对象
key (producer)	为制作者操作设置特定的密钥。		对象
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
oldValue (producer)	为一些制作者操作设置特定的旧值。		对象

Name	描述	默认值	类型
operation (producer)	要执行的操作。 Enum 值： • PUT • PUTASYNC • PUTALL • PUTALLASYNC • PUTIFABSENT • PUTIFABSENTASYNC • GET • GETORDEFAULT • CONTAINSKEY • CONTAINSVALUE • 删除 • REMOVEASYNC • REPLACE • REPLACEASYNC • SIZE • 清除 • CLEARASYNC • 查询 • STATS • COMPUTE • COMPUTEASYNC	PUT	InfinispanOperation
value (producer)	为制作者操作设置特定值。		对象
密码 （安全）	定义用于访问 infinispan 实例的密码。		字符串
saslMechanism (security)	定义用于访问 infinispan 实例的 SASL 机制。		字符串
securityRealm (security)	定义用于访问 infinispan 实例的安全域。		字符串

Name	描述	默认值	类型
securityServerName (security)	定义用于访问 infinispan 实例的安全服务器名称。		字符串
用户名 （安全）	定义用于访问 infinispan 实例的用户名。		字符串
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
cacheContainer (advanced)	**Autowired** 指定要连接的缓存容器。		RemoteCacheManager
cacheContainerConfiguration (advanced)	**Autowired** The CacheContainer 配置。如果没有定义 cacheContainer，则使用。		配置
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)	在每个缓存调用中，默认应用以逗号分隔的 org.infinispan.client.hotrod.Flag 的列表。		字符串
remappingFunction (advanced)	设置要在计算操作中使用的特定 remappingFunction。		BiFunction
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，则该值将用作标头的名称，以存储查询结果，并保留原始消息正文。这个值可以被名为：CamelInfinispanOperationResultHeader 的消息标头覆盖。		字符串

44.5. 端点选项

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：

44.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
cacheName (common)	**必需**使用的缓存的名称。使用 current 使用当前配置的缓存管理器中的现有缓存名称。或者默认缓存管理器名称使用 default。		字符串

44.5.2. 查询参数(26 参数)

Name	描述	默认值	类型
hosts (common)	指定 Infinispan 实例上缓存的主机。		字符串
queryBuilder (common)	指定查询构建器。		InfinispanQueryBuilder
secure (common)	定义是否连接到安全 Infinispan 实例。	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
customListener (consumer)	如果提供，则返回正在使用的自定义监听程序。		InfinispanRemoteCustomListener
eventTypes (consumer)	指定要由消费者注册的事件类型集合。Multiple 事件可以用逗号分开。可能的事件类型是：CLIENT_CACHE_ENTRY_CREATED, CLIENT_CACHE_ENTRY_MODIFIED, CLIENT_CACHE_ENTRY_REMOVED, CLIENT_CACHE_ENTRY_EXPIRED, CLIENT_CACHE_FAILOVER。		字符串
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
DefaultValue (producer)	为一些制作者操作设置特定的默认值。		对象
key (producer)	为制作者操作设置特定的密钥。		对象
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
oldValue (producer)	为一些制作者操作设置特定的旧值。		对象

Name	描述	默认值	类型
operation (producer)	要执行的操作。 Enum 值： • PUT • PUTASYNC • PUTALL • PUTALLASYNC • PUTIFABSENT • PUTIFABSENTASYNC • GET • GETORDEFAULT • CONTAINSKEY • CONTAINSVALUE • 删除 • REMOVEASYNC • REPLACE • REPLACEASYNC • SIZE • 清除 • CLEARASYNC • 查询 • STATS • COMPUTE • COMPUTEASYNC	PUT	InfinispanOperation
value (producer)	为制作者操作设置特定值。		对象
密码 （安全）	定义用于访问 infinispan 实例的密码。		字符串
saslMechanism (security)	定义用于访问 infinispan 实例的 SASL 机制。		字符串
securityRealm (security)	定义用于访问 infinispan 实例的安全域。		字符串

Name	描述	默认值	类型
securityServerName (security)	定义用于访问 infinispan 实例的安全服务器名称。		字符串
用户名 （安全）	定义用于访问 infinispan 实例的用户名。		字符串
cacheContainer (advanced)	**Autowired** 指定要连接的缓存容器。		RemoteCacheManager
cacheContainerConfiguration (advanced)	**Autowired** The CacheContainer 配置。如果没有定义 cacheContainer，则使用。		配置
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)	在每个缓存调用中，默认应用以逗号分隔的 org.infinispan.client.hotrod.Flag 的列表。		字符串
remappingFunction (advanced)	设置要在计算操作中使用的特定 remappingFunction。		BiFunction
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，则该值将用作标头的名称，以存储查询结果，并保留原始消息正文。这个值可以被名为：CamelInfinispanOperationResultHeader 的消息标头覆盖。		字符串

44.6. CAMEL OPERATIONS

本节列出了所有可用的操作，及其标题信息。

表 44.1. 表 1.Put Operations

操作名称	描述
InfinispanOperation.PUT	在缓存中放置键/值对，可以选择过期
InfinispanOperation.PUTASYNC	异步将键/值对放在缓存中，可以选择过期

操作名称	描述
InfinispanOperation.PUTIFABSENT	如果不存在，在缓存中放置键/值对（如果不存在），可以选择过期
InfinispanOperation.PUTIFABSENTASYNC	如果不存在，异步将键/值对放在缓存中，可以选择过期

- ***所需的**标头 **：***
  - ***CamelInfinispanKey***
  - ***CamelInfinispanValue***
- ***可**选标头 **：***
  - ***CamelInfinispanLifespanTime***
  - ***CamelInfinispanLifespanTimeUnit***
  - ***CamelInfinispanMaxIdleTime***
  - ***CamelInfinispanMaxIdleTimeUnit***
- *结**果**标头 **：***
  - ***CamelInfinispanOperationResult***

表 44.2. 表 2。放置所有操作

操作名称	描述

操作名称	描述
InfinispanOperation.PUTALL	在缓存中添加多个条目，可以选择过期
CamelInfinispanOperation.PUTALLASYNC	异步向缓存添加多个条目，可以选择过期

- ***所需的**标头 **：***
  - ***CamelInfinispanMap***
- ***可**选标头 **：***
  - ***CamelInfinispanLifespanTime***
  - ***CamelInfinispanLifespanTimeUnit***
  - ***CamelInfinispanMaxIdleTime***
  - ***CamelInfinispanMaxIdleTimeUnit***

表 44.3. 表 3。 获取操作

操作名称	描述
InfinispanOperation.GET	从缓存中检索与特定键关联的值
InfinispanOperation.GETORDEFAULT	从缓存中检索与特定键关联的值或默认值

- ***所需的**标头 **：***
  - ***CamelInfinispanKey***

表 44.4. 表 4包含密钥操作

操作名称	描述
InfinispanOperation.CONTAINSKEY	决定缓存是否包含特定密钥

- ***所需的**标头*
  - ***CamelInfinispanKey***
- *结**果**标头*
  - ***CamelInfinispanOperationResult***

表 44.5. 表 5包含 Value Operation

操作名称	描述
InfinispanOperation.CONTAINSVALUE	决定缓存是否包含特定值

- ***所需的**标头 **：***
  - ***CamelInfinispanKey***

表 44.6. 表 6。删除操作

操作名称	描述
InfinispanOperation.REMOVE	从缓存中删除条目，只有在值与给定值匹配时才可以选择
InfinispanOperation.REMOVEASYNC	异步从缓存中删除条目，可选择仅在值与给定值匹配时才从缓存中删除条目

- ***所需的**标头 **：***
  - ***CamelInfinispanKey***

- *可选标头：*
  - ***CamelInfinispanValue***
- *结果标头：*
  - ***CamelInfinispanOperationResult***

表 44.7. 表 7。替换 Operations

操作名称	描述
InfinispanOperation.REPLACE	条件地替换缓存中的条目，可选使用过期
InfinispanOperation.REPLACEASYNC	异步替换缓存中的条目，可选使用过期

- *所需的标头：*
  - ***CamelInfinispanKey***
  - ***CamelInfinispanValue***
  - ***CamelInfinispanOldValue***
- *可选标头：*
  - ***CamelInfinispanLifespanTime***
  - ***CamelInfinispanLifespanTimeUnit***
  - ***CamelInfinispanMaxIdleTime***

- ***CamelInfinispanMaxIdleTimeUnit***

- *结**果**标头 **：***

  - ***CamelInfinispanOperationResult***

表 44.8. 表 8。清除操作

操作名称	描述
InfinispanOperation.CLEAR	清除缓存
InfinispanOperation.CLEARASYNC	异步清除缓存

表 44.9. 表 9。大小操作

操作名称	描述
InfinispanOperation.SIZE	返回缓存中的条目数

- *结**果**标头*

  - ***CamelInfinispanOperationResult***

表 44.10. 表 10。stats Operation

操作名称	描述
InfinispanOperation.STATS	返回有关缓存的统计信息

- *结**果**标头 **：***

  - ***CamelInfinispanOperationResult***

表 44.11. 表 11.查询操作

操作名称	描述
InfinispanOperation.QUERY	在缓存上执行查询

- ***所需的**标头 **：***
  - ***CamelInfinispanQueryBuilder***
- *结**果**标头 **：***
  - ***CamelInfinispanOperationResult***

注意

***默**认**情况下，放置（密**钥、*值*）**和 remove (key)等写方法不会返回前面的**值。*

*44.7. 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	对或生成事件的密钥执行操作的密钥。
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。
CamelInfinispanLifespanTime	**null**	long	制作者	缓存内值的 Lifespan 时间。负值解释为 infinity。

Name	默认值	类型	Context	描述
CamelInfinispanTimeUnit	**null**	字符串	制作者	条目 Lifespan 时间的时间单位.
CamelInfinispanMaxIdleTime	**null**	long	制作者	在被视为过期前允许闲置条目的最大时间。
CamelInfinispanMaxIdleTimeUnit	**null**	字符串	制作者	条目 Max Idle Time 的时间单位。
CamelInfinispanQueryBuilder	null	InfinispanQueryBuilder	制作者	如果命令不存在为 InifinispanConfiguration，则 QueryBuilde 用于 QUERY 命令
CamelInfinispanOperationResultHeader	null	字符串	制作者	将操作结果存储在标头中，而不是消息正文

44.8. 例子

- *将键/值放在命名缓存中：*

```
from("direct:start")
   .setHeader(InfinispanConstants.OPERATION).constant(InfinispanOperation.PUT) (1)
   .setHeader(InfinispanConstants.KEY).constant("123") (2)
   .to("infinispan:myCacheName&cacheContainer=#cacheContainer"); (3)
```

其中，

- *1 - 设置要执行的操作*
- *2 - 设置用于识别缓存中元素的密钥*
- *3 - 使用 registry 中配置的缓存管理器 cacheContainer 将元素放在名为 myCacheName的缓存中*

可以在条目过期前配置生命周期和/或闲置时间，并从缓存中驱除，例如：

```
from("direct:start")
   .setHeader(InfinispanConstants.OPERATION).constant(InfinispanOperation.GET)
```

```
    .setHeader(InfinispanConstants.KEY).constant("123")
    .setHeader(InfinispanConstants.LIFESPAN_TIME).constant(100L) (1)

.setHeader(InfinispanConstants.LIFESPAN_TIME_UNIT.constant(TimeUnit.MILLISECO
NDS.toString()) (2)
    .to("infinispan:myCacheName");
```

其中,

- 1 - 设置条目的寿命
- 2 - 设置生命周期的时间单位

queries

```
from("direct:start")
   .setHeader(InfinispanConstants.OPERATION, InfinispanConstants.QUERY)
   .setHeader(InfinispanConstants.QUERY_BUILDER, new InfinispanQueryBuilder() {
      @Override
      public Query build(QueryFactory<Query> qf) {
         return qf.from(User.class).having("name").like("%abc%").build();
      }
   })
   .to("infinispan:myCacheName?cacheContainer=#cacheManager") ;
```

注意

域对象的 .proto 描述符必须注册到远程数据网格服务器，请参阅官方 Infinispan 文档中的 Remote Query 示例。

自定义 Listeners

```
from("infinispan://?cacheContainer=#cacheManager&customListener=#myCustomListener")
  .to("mock:result");
```

myCustomListener 的实例必须存在，Camel 应能够从 Registry 查找它。我们鼓励用户扩展 org.apache.camel.component.infinispan.remote.InfinispanRemoteCustomListener 类，并给生成的类标上 @ClientListener，可在软件包 org.infinispan.client.hotrod.annotation 中找到。

44.9. 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。

Java 示例

```
InfinispanRemoteConfiguration conf = new InfinispanRemoteConfiguration(); (1)
conf.setHosts("localhost:1122")

InfinispanRemoteIdempotentRepository repo = new
InfinispanRemoteIdempotentRepository("idempotent");  (2)
repo.setConfiguration(conf);

context.addRoutes(new RouteBuilder() {
    @Override
    public void configure() {
        from("direct:start")
            .idempotentConsumer(header("MessageID"), repo) (3)
            .to("mock:result");
    }
});
```

其中，

- 1 - 配置缓存
- 2 - 配置存储库 bean
- 3 - 将存储库设置为路由

XML 示例

```
<bean id="infinispanRepo"
class="org.apache.camel.component.infinispan.remote.InfinispanRemoteIdempotentRepository"
destroy-method="stop">
  <constructor-arg value="idempotent"/> (1)
  <property name="configuration"> (2)
    <bean class="org.apache.camel.component.infinispan.remote.InfinispanRemoteConfiguration">
      <property name="hosts" value="localhost:11222"/>
    </bean>
  </property>
</bean>

<camelContext xmlns="http://camel.apache.org/schema/spring">
    <route>
        <from uri="direct:start" />
        <idempotentConsumer messageIdRepositoryRef="infinispanRepo"> (3)
            <header>MessageID</header>
            <to uri="mock:result" />
        </idempotentConsumer>
    </route>
</camelContext>
```

其中，

- 1 - 设置存储库将使用的缓存名称
- 2 - 配置存储库 bean
- 3 - 将存储库设置为路由

44.10. 使用基于 INFINISPAN 的聚合存储库

在本节中，我们将使用基于 Infinispan 的聚合存储库。

Java 示例

```
InfinispanRemoteConfiguration conf = new InfinispanRemoteConfiguration(); (1)
conf.setHosts("localhost:1122")

InfinispanRemoteAggregationRepository repo = new
```

```
InfinispanRemoteAggregationRepository();  (2)
repo.setCacheName("aggregation");
repo.setConfiguration(conf);

context.addRoutes(new RouteBuilder() {
   @Override
   public void configure() {
      from("direct:start")
            .aggregate(header("MessageID"))
            .completionSize(3)
            .aggregationRepository(repo) (3)
            .aggregationStrategyRef("myStrategy")
            .to("mock:result");
   }
});
```

其中,

- 1 - 配置缓存
- 2 - 创建存储库 bean
- 3 - 将存储库设置为路由

XML 示例

```
<bean id="infinispanRepo"
class="org.apache.camel.component.infinispan.remote.InfinispanRemoteAggregationRepository"
destroy-method="stop">
  <constructor-arg value="aggregation"/> (1)
  <property name="configuration"> (2)
    <bean class="org.apache.camel.component.infinispan.remote.InfinispanRemoteConfiguration">
      <property name="hosts" value="localhost:11222"/>
    </bean>
  </property>
</bean>

<camelContext xmlns="http://camel.apache.org/schema/spring">
   <route>
      <from uri="direct:start" />
      <aggregate strategyRef="myStrategy"
            completionSize="3"
            aggregationRepositoryRef="infinispanRepo"> (3)
         <correlationExpression>
            <header>MessageID</header>
```

```
            </correlationExpression>
            <to uri="mock:result"/>
        </aggregate>
    </route>
</camelContext>
```

其中,

- *1 - 设置存储库将使用的缓存名称*
- *2 - 配置存储库 bean*
- *3 - 将存储库设置为路由*

注意

随着 Infinispan 11 的发布，需要在创建的任何缓存上设置编码配置。这对消耗的事件也至关重要。如需更多信息，请参阅官方 Infinispan 文档中的 数据编码和 MediaTypes。

44.11. 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.infinispan.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.infinispan.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.infinispan.cache-container	指定要连接的缓存容器。选项是 org.infinispan.client.hotrod.RemoteCacheManager 类型。		RemoteCacheManager
camel.component.infinispan.cache-container-configuration	CacheContainer 配置。如果没有定义 cacheContainer，则使用。选项是 org.infinispan.client.hotrod.configuration.Configuration 类型。		配置
camel.component.infinispan.configuration	组件配置.选项是 org.apache.camel.component.infinispan.remote.InfinispanRemoteConfiguration 类型。		InfinispanRemoteConfiguration
camel.component.infinispan.configuration-properties	为 CacheManager 实现特定属性。		Map
camel.component.infinispan.configuration-uri	CacheManager 的实现特定 URI。		字符串
camel.component.infinispan.custom-listener	如果提供，则返回正在使用的自定义监听程序。选项是 org.apache.camel.component.infinispan.remote.InfinispanRemoteCustomListener 类型。		InfinispanRemoteCustomListener
camel.component.infinispan.enabled	是否启用 infinispan 组件的自动配置。这默认是启用的。		布尔值
camel.component.infinispan.event-types	指定要由消费者注册的事件类型集合。Multiple 事件可以用逗号分开。可能的事件类型是：CLIENT_CACHE_ENTRY_CREATED, CLIENT_CACHE_ENTRY_MODIFIED, CLIENT_CACHE_ENTRY_REMOVED, CLIENT_CACHE_ENTRY_EXPIRED, CLIENT_CACHE_FAILOVER。		字符串
camel.component.infinispan.flags	在每个缓存调用中，默认应用以逗号分隔的 org.infinispan.client.hotrod.Flag 的列表。		字符串

Name	描述	默认值	类型
camel.component.infinispan.hosts	指定 Infinispan 实例上缓存的主机。		字符串
camel.component.infinispan.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.infinispan.operation	要执行的操作。		InfinispanOperation
camel.component.infinispan.password	定义用于访问 infinispan 实例的密码。		字符串
camel.component.infinispan.query-builder	指定查询构建器。选项是 org.apache.camel.component.infinispan.InfinispanQueryBuilder 类型。		InfinispanQueryBuilder
camel.component.infinispan.remapping-function	设置要在计算操作中使用的特定 remappingFunction。选项是一个 java.util.function.BiFunction 类型。		BiFunction
camel.component.infinispan.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，则该值将用作标头的名称，以存储查询结果，并保留原始消息正文。这个值可以被名为：CamelInfinispanOperationResultHeader 的消息标头覆盖。		字符串
camel.component.infinispan.sasl-mechanism	定义用于访问 infinispan 实例的 SASL 机制。		字符串
camel.component.infinispan.secure	定义是否连接到安全 Infinispan 实例。	false	布尔值
camel.component.infinispan.security-realm	定义用于访问 infinispan 实例的安全域。		字符串
camel.component.infinispan.security-server-name	定义用于访问 infinispan 实例的安全服务器名称。		字符串

Name	描述	默认值	类型
camel.component.infinispan.username	定义用于访问 infinispan 实例的用户名。		字符串

第 45 章 INFINISPAN EMBEDDED

从 Camel 2.13 开始

支持生成者和消费者

此组件允许您与 Infinispan 分布式数据网格/缓存交互。Infinispan 是一个非常可扩展、高度可用的键/值数据存储和 Java 编写的数据网格平台。

camel-infinispan-embedded 组件包括以下功能：

- 本地 Camel Consumer - Receives 缓存更改通知，并将其发送到处理。这可以同步或异步完成，并且也支持复制或分布式缓存。
- 本地 Camel Producer - 生成者创建消息并将其发送到端点。camel-infinispan producer 使用 GET、PUT、REMOVE 和 CLEAR 操作。本地制作者也支持复制或分布式缓存。

事件异步处理。

45.1. 依赖项

当在 Camel Spring Boot 中使用 infinispan-embedded 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-infinispan-embedded-starter</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

45.2. URI 格式

```
infinispan-embedded://cacheName?[options]
```

生产者允许发送消息到本地 infinispan 缓存。使用者允许从本地 infinispan 缓存侦听事件。

如果没有提供缓存配置，则直接在组件中创建嵌入的 cacheContainer。

45.3. 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

45.3.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

45.3.2. 配置端点选项

端点有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点直接在端点 URI 中作为路径和查询参数完成。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式 的安全 方法。

使用 Property Placeholders 配置不允许硬编码 URL、端口号、敏感信息和其他设置的选项。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

45.4. 组件选项

Infinispan 嵌入式组件支持下面所列的 20 个选项。

Name	描述	默认值	类型
configuration (common)	组件配置.		InfinispanEmbeddedConfiguration
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
customListener (consumer)	如果提供，则返回正在使用的自定义监听程序。		InfinispanEmbeddedCustomListener

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合。Multiple 事件可以用逗号分开。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, cache_ENTRY_EXPIRED, data_REHASHED, topOLOGY_CHANGED, partition_STATUS_CHANGED, persistence_AVAILABILITY_CHANGED.		字符串
sync (consumer)	如果为 true，消费者将同步接收通知。	true	布尔值
DefaultValue (producer)	为一些制作者操作设置特定的默认值。		对象
key (producer)	为制作者操作设置特定的密钥。		对象

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。通过延迟启动，启动失败可以在路由消息期间通过 Camel 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
oldValue (producer)	为一些制作者操作设置特定的旧值。		对象
operation (producer)	要执行的操作。 Enum 值： * PUTASYNC * PUTALL * PUTALLASYNC * PUTIFABSENT * PUTIFABSENTASYNC * GET * GETORDEFAULT * CONTAINSKEY * REMOVE * REMOVEASYNC * REMOVE * REMOVEASYNC * REPLACE * REPLACEASYNC * SIZE * CLEAR * CLEARASYNC * QUERY * STATS * COMPUTE * COMPUTEASYNC	PUT	InfinispanOperation
value*(producer)	为制作者操作设置特定值。		对象

Name	描述	默认值	类型
autowiredEnabled (advanced)	是否启用自动关闭。这用于自动写入选项（选项必须标记为 auto-wired），方法是在 registry 中查找是否有单个匹配类型实例，然后在组件上配置。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
cacheContainer (advanced)	**Autowired** 指定要连接的缓存容器。		EmbeddedCacheManager
cacheContainerConfiguration (advanced)	**Autowired** The CacheContainer 配置。如果没有定义 cacheContainer，则使用。		配置
configurationUri (advanced)	CacheManager 的实现特定 URI。		字符串
标记 (advanced)	以逗号分隔的 org.infinispan.context.Flag 列表，默认在每个缓存调用中应用。		字符串
remappingFunction (advanced)	设置要在计算操作中使用的特定 remappingFunction。		BiFunction
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，则该值将用作标头的名称，以存储查询结果，并保留原始消息正文。这个值可以被名为：CamelInfinispanOperationResultHeader 的消息标头覆盖。		字符串

45.5. 端点选项

Infinispan Embedded 端点使用 URI 语法进行配置。

```
infinispan-embedded:cacheName
```

以下是 path 和 查询参数。

45.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
cacheName (common)	**必需**使用的缓存的名称。使用 current 使用当前配置的缓存管理器中的现有缓存名称。或者默认缓存管理器名称使用 default。		字符串

45.5.2. 查询参数 (20 参数)

Name	描述	默认值	类型
queryBuilder (common)	指定查询构建器。		InfinispanQueryBuilder
clusteredListener (consumer)	如果为 true，则会为整个集群安装监听程序。	false	布尔值
customListener (consumer)	如果提供，则返回正在使用的自定义监听程序。		InfinispanEmbeddedCustomListener

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合。Multiple 事件可以用逗号分开。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, cache_ENTRY_EXPIRED, data_REHASHED, topOLOGY_CHANGED, partition_STATUS_CHANGED, persistence_AVAILABILITY_CHANGED.		字符串
sync (consumer)	如果为 true，消费者将同步接收通知。	true	布尔值
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： * InOnly * InOut * InOptionalOut		ExchangePattern
DefaultValue (producer)	为一些制作者操作设置特定的默认值。		对象
key (producer)	为制作者操作设置特定的密钥。		对象
oldValue (producer)	为一些制作者操作设置特定的旧值。		对象
operation (producer)	要执行的操作。 Enum 值： * PUTASYNC * PUTALL * PUTALLASYNC * PUTIFABSENT * PUTIFABSENTASYNC * GET * GETORDEFAULT * CONTAINSKEY * REMOVE * REMOVEASYNC * REMOVE * REMOVEASYNC * REPLACE * REPLACEASYNC * SIZE * CLEAR * CLEARASYNC * QUERY * STATS * COMPUTE * COMPUTEASYNC	PUT	InfinispanOperation
value (producer)	为制作者操作设置特定值。		对象

Name	描述	默认值	类型
lazyStartProducer (producer (advanced))	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。通过延迟启动，启动失败可以在路由消息期间通过 Camel 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
cacheContainer (advanced)	**Autowired** 指定要连接的缓存容器。		EmbeddedCacheManager
cacheContainerConfiguration (advanced)	**Autowired** The CacheContainer 配置。如果没有定义 cacheContainer，则使用。		配置
configurationUri (advanced)	CacheManager 的实现特定 URI。		字符串
标记 (advanced)	以逗号分隔的 org.infinispan.context.Flag 列表，默认在每个缓存调用中应用。		字符串
remappingFunction (advanced)	设置要在计算操作中使用的特定 remappingFunction。		BiFunction

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，则该值将用作标头的名称，以存储查询结果，并保留原始消息正文。这个值可以被名为：CamelInfinispanOperationResultHeader 的消息标头覆盖。		字符串

45.6. 消息标头

Infinispan Embedded 组件支持下面列出的 22 个消息标头。

Name	描述	默认值	类型
CamelInfinispanEventType (consumer) 常量 ： **EVENT_TYPE**	接收的事件的类型。		字符串
CamelInfinispanIsPre (consumer) 常数 ： **IS_PRE**	如果通知在事件发生之前，则为 true，如果事件发生后则为 false。		布尔值
CamelInfinispanCacheName (common) 常量 ： **CACHE_NAME**	参与操作或事件的缓存。		字符串
CamelInfinispanKey (common) 常数： **KEY**	对或生成事件的密钥执行操作的密钥。		对象
CamelInfinispanValue (producer) constant: **VALUE**	用于操作的值。		对象

Name	描述	默认值	类型
CamelInfinispanDefaultValue (producer) constant: **DEFAULT_VALUE**	用于 getOrDefault 的默认值。		对象
CamelInfinispanOldValue (producer) constant: **OLD_VALUE**	用于替换的旧值。		对象
CamelInfinispanMap (producer) 常量 ： **MAP**	CamelInfinispanOperationPutAll 操作时要使用的映射。		Map
CamelInfinispanOperation (producer) 常数 ： **OPERATION**	要执行的操作。 Enum 值： * PUTASYNC * PUTALL * PUTALLASYNC * PUTIFABSENT * PUTIFABSENTASYNC * GET * GETORDEFAULT * CONTAINSKEY * REMOVE * REMOVEASYNC * REMOVE * REMOVEASYNC * REPLACE * REPLACEASYNC * SIZE * CLEAR * CLEARASYNC * QUERY * STATS * COMPUTE * COMPUTEASYNC		InfinispanOperation
CamelInfinispanOperationResult (producer) 恒定： **RESULT**	其值为结果的标头名称。		字符串
CamelInfinispanOperationResultHeader (producer) 恒定： **RESULT_HEADER**	将操作结果存储在标头中，而不是消息正文。		字符串

Name	描述	默认值	类型
CamelInfinispanLifespanTime (producer) 常量 ： **LIFESPAN_TIME**	缓存内值的 Lifespan 时间。负值解释为 infinity。		long
CamelInfinispanTimeUnit (producer) 常量 ： **LIFESPAN_TIME_UNIT**	条目 Lifespan 时间的时间单位. Enum 值： * NANOSECONDS * MICROSECONDS * MILLISECONDS * SECONDS * SECONDS * MINUTES * HOURS * DAYS		TimeUnit
CamelInfinispanMaxIdleTime (producer) 常数 ： **MAX_IDLE_TIME**	在被视为过期前允许闲置条目的最大时间。		long
CamelInfinispanMaxIdleTimeUnit (producer) 常数 ： **MAX_IDLE_TIME_UNIT**	条目 Max Idle Time 的时间单位。 Enum 值： * NANOSECONDS * MICROSECONDS * MILLISECONDS * SECONDS * SECONDS * MINUTES * HOURS * DAYS		TimeUnit
CamelInfinispanIgnoreReturnValues (consumer) 恒定： **IGNORE_RETURN_VALUES**	信号写入操作的返回值会被忽略，因此不需要从存储或远程节点读取现有值。	false	布尔值
CamelInfinispanEventData (consumer) 常量 ： **EVENT_DATA**	事件数据。		对象

Name	描述	默认值	类型
CamelInfinispanQueryBuilder (producer) 常数 ： **QUERY_BUILDER**	如果命令不存在为 InifinispanConfiguration ，则 QueryBuilder 用于 QUERY 命令。		InfinispanQueryBuilder
CamelInfinispanCommandRetried (consumer) 常数： **COMMAND_RETRIED**	如果因为拓扑更改，导致这个问题的 write 命令再次被重试，则会出现这种情况。		布尔值
CamelInfinispanEntryCreated (consumer) 常数： **ENTRY_CREATED**	指明缓存条目修改事件是否为正在创建缓存条目的结果。		布尔值
CamelInfinispanOriginLocal (consumer) 常数： **ORIGIN_LOCAL**	如果调用源自本地缓存实例，则为 true；如果源自于远程实例，则为 false。		布尔值
CamelInfinispanCurrentState (consumer) constant: **CURRENT_STATE**	如果此事件从现有条目生成为监听器具有 Listener，则为 true。		布尔值

45.7. CAMEL OPERATIONS

本节列出了所有可用的操作及其标题信息。

表 45.1. 表 1.Put Operations

操作名称	描述
InfinispanOperation.PUT	在缓存中放置键/值对，可以选择过期
InfinispanOperation.PUTASYNC	异步将键/值对放在缓存中，可以选择过期
InfinispanOperation.PUTIFABSENT	如果不存在，在缓存中放置键/值对（如果不存在），可以选择过期

操作名称	描述
InfinispanOperation.PUTIFABSENTASYNC	如果不存在，异步将键/值对放在缓存中，可以选择过期

- ***所需的**标头 ：*
  - ***CamelInfinispanKey***
  - ***CamelInfinispanValue***
- ***可**选标头 ：*
  - ***CamelInfinispanLifespanTime***
  - ***CamelInfinispanLifespanTimeUnit***
  - ***CamelInfinispanMaxIdleTime***
  - ***CamelInfinispanMaxIdleTimeUnit***
- *结**果**标头 ：*
  - ***CamelInfinispanOperationResult***

表 45.2. 表 2。放置所有操作

操作名称	描述
InfinispanOperation.PUTALL	在缓存中添加多个条目，可以选择过期
CamelInfinispanOperation.PUTALLASYNC	异步向缓存添加多个条目，可以选择过期

- ***所需的**标头 **：***
  - ***CamelInfinispanMap***
- ***可**选标头 **：***
  - ***CamelInfinispanLifespanTime***
  - ***CamelInfinispanLifespanTimeUnit***
  - ***CamelInfinispanMaxIdleTime***
  - ***CamelInfinispanMaxIdleTimeUnit***

表 45.3. 表 3。 获取操作

操作名称	描述
InfinispanOperation.GET	从缓存中检索与特定键关联的值
InfinispanOperation.GETORDEFAULT	从缓存中检索与特定键关联的值或默认值

- ***所需的**标头 **：***
  - ***CamelInfinispanKey***

表 45.4. 表 4包含密钥操作

操作名称	描述
InfinispanOperation.CONTAINSKEY	决定缓存是否包含特定密钥

- ***所需的**标头*

- ***CamelInfinispanKey***

- *结**果**标头*

- ***CamelInfinispanOperationResult***

表 45.5. 表 5包含 Value Operation

操作名称	描述
InfinispanOperation.CONTAINSVALUE	决定缓存是否包含特定值

- ***所需的**标头 **：***

- ***CamelInfinispanKey***

表 45.6. 表 6。 删**除操作

操作名称	描述
InfinispanOperation.REMOVE	从缓存中删除条目，只有在值与给定值匹配时才可以选择
InfinispanOperation.REMOVEASYNC	异步从缓存中删除条目，可选择仅在值与给定值匹配时才从缓存中删除条目

- ***所需的**标头 **：***

- ***CamelInfinispanKey***

- ***可**选标头 **：***

- ***CamelInfinispanValue***

- *结**果**标头*：
  - ***CamelInfinispanOperationResult***

表 45.7. 表 7。替换 Operations

操作名称	描述
InfinispanOperation.REPLACE	条件地替换缓存中的条目，可选使用过期
InfinispanOperation.REPLACEASYNC	异步替换缓存中的条目，可选使用过期

- ***所需的**标头*：
  - ***CamelInfinispanKey***
  - ***CamelInfinispanValue***
  - ***CamelInfinispanOldValue***
- ***可**选标头*：
  - ***CamelInfinispanLifespanTime***
  - ***CamelInfinispanLifespanTimeUnit***
  - ***CamelInfinispanMaxIdleTime***
  - ***CamelInfinispanMaxIdleTimeUnit***
- *结**果**标头*：

- ***CamelInfinispanOperationResult***

表 45.8. 表 8。清除操作

操作名称	描述
InfinispanOperation.CLEAR	清除缓存
InfinispanOperation.CLEARASYNC	异步清除缓存

表 45.9. 表 9。大小操作

操作名称	描述
InfinispanOperation.SIZE	返回缓存中的条目数

- *结**果**标头*
  - ***CamelInfinispanOperationResult***

表 45.10. 表 10。stats Operation

操作名称	描述
InfinispanOperation.STATS	返回有关缓存的统计信息

- *结**果**标头 **：***
  - ***CamelInfinispanOperationResult***

表 45.11. 表 11.查询操作

操作名称	描述
InfinispanOperation.QUERY	在缓存上执行查询

- ***所需的**标头 **：***

- CamelInfinispanQueryBuilder

- 结果标头：

  - CamelInfinispanOperationResult

注意

默认情况下，放置（密钥、值）和 remove (key)等写方法不会返回前面的值。

45.8. 例子

- 将键/值放在命名缓存中：

```
from("direct:start")
   .setHeader(InfinispanConstants.OPERATION).constant(InfinispanOperation.PUT) (1)
   .setHeader(InfinispanConstants.KEY).constant("123") (2)
   .to("infinispan:myCacheName&cacheContainer=#cacheContainer"); (3)
```

  - 设置要执行的操作

  - 设置用于识别缓存中元素的密钥

  - 使用 registry 中配置的缓存管理器 cacheContainer 将元素放在名为 myCacheName的缓存中

    可以在条目过期前配置生命周期和/或空闲时间，并从缓存中驱除，例如：

```
from("direct:start")
   .setHeader(InfinispanConstants.OPERATION).constant(InfinispanOperation.GET)
   .setHeader(InfinispanConstants.KEY).constant("123")
   .setHeader(InfinispanConstants.LIFESPAN_TIME).constant(100L) (1)

.setHeader(InfinispanConstants.LIFESPAN_TIME_UNIT.constant(TimeUnit.MILLISECO
NDS.toString()) (2)
   .to("infinispan:myCacheName");
```

- 设置条目的 Lifespan
- 设置 lifespan 的时间单位

- queries

```
from("direct:start")
   .setHeader(InfinispanConstants.OPERATION, InfinispanConstants.QUERY)
   .setHeader(InfinispanConstants.QUERY_BUILDER, new InfinispanQueryBuilder() {
      @Override
      public Query build(QueryFactory<Query> qf) {
         return qf.from(User.class).having("name").like("%abc%").build();
      }
   })
   .to("infinispan:myCacheName?cacheContainer=#cacheManager") ;
```

- 自定义 Listeners

```
from("infinispan://?
cacheContainer=#cacheManager&customListener=#myCustomListener")
 .to("mock:result");
```

- myCustomListener 的实例必须存在，Camel 应能够从 Registry 查找它。我们鼓励用户扩展 org.apache.camel.component.infinispan.embedded.InfinispanEmbeddedCustomListener 类，并给生成的类标上 @Listener，该类可在软件包 org.infinispan.notifications 中找到。

45.9. 使用基于 INFINISPAN 的幂等存储库

Java 示例

```
InfinispanEmbeddedConfiguration conf = new InfinispanEmbeddedConfiguration(); (1)
conf.setConfigurationUri("classpath:infinispan.xml")

InfinispanEmbeddedIdempotentRepository repo = new
InfinispanEmbeddedIdempotentRepository("idempotent");  (2)
repo.setConfiguration(conf);

context.addRoutes(new RouteBuilder() {
   @Override
   public void configure() {
      from("direct:start")
```

```
        .idempotentConsumer(header("MessageID"), repo) (3)
        .to("mock:result");
    }
});
```

- 配置缓存
- 配置存储库 bean
- 将存储库设置为路由

XML 示例

```
<bean id="infinispanRepo"
class="org.apache.camel.component.infinispan.embedded.InfinispanEmbeddedIdempotentRe
pository" destroy-method="stop">
  <constructor-arg value="idempotent"/> (1)
  <property name="configuration"> (2)
    <bean
class="org.apache.camel.component.infinispan.embedded.InfinispanEmbeddedConfiguration
">
      <property name="configurationUrl" value="classpath:infinispan.xml"/>
    </bean>
  </property>
</bean>

<camelContext xmlns="http://camel.apache.org/schema/spring">
    <route>
        <from uri="direct:start" />
        <idempotentConsumer idempotentRepository="infinispanRepo"> (3)
            <header>MessageID</header>
            <to uri="mock:result" />
        </idempotentConsumer>
    </route>
</camelContext>
```

- 设置存储库将使用的缓存名称
- 配置存储库 bean
- 将存储库设置为路由

45.10. 使用基于 INFINISPAN 的聚合存储库

Java 示例

```
InfinispanEmbeddedConfiguration conf = new InfinispanEmbeddedConfiguration(); (1)
conf.setConfigurationUri("classpath:infinispan.xml")

InfinispanEmbeddedAggregationRepository repo = new
InfinispanEmbeddedAggregationRepository("aggregation");  (2)
repo.setConfiguration(conf);

context.addRoutes(new RouteBuilder() {
   @Override
   public void configure() {
      from("direct:start")
            .aggregate(header("MessageID"))
            .completionSize(3)
            .aggregationRepository(repo) (3)
            .aggregationStrategy("myStrategy")
            .to("mock:result");
   }
});
```

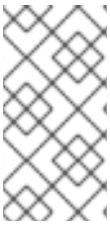
- 配置缓存
- 创建存储库 bean
- 将存储库设置为路由

XML 示例

```
<bean id="infinispanRepo"
class="org.apache.camel.component.infinispan.embedded.InfinispanEmbeddedAggregationR
epository" destroy-method="stop">
  <constructor-arg value="aggregation"/> (1)
  <property name="configuration"> (2)
    <bean
class="org.apache.camel.component.infinispan.embedded.InfinispanEmbeddedConfiguration
">
      <property name="configurationUrl" value="classpath:infinispan.xml"/>
    </bean>
  </property>
</bean>

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

```
    <route>
      <from uri="direct:start" />
      <aggregate aggregationStrategy="myStrategy"
              completionSize="3"
              aggregationRepository="infinispanRepo"> (3)
        <correlationExpression>
          <header>MessageID</header>
        </correlationExpression>
        <to uri="mock:result"/>
      </aggregate>
    </route>
  </camelContext>
```

- 设置存储库将使用的缓存名称
- 配置存储库 bean
- 将存储库设置为路由



注意

随着 Infinispan 11 的发布，需要在创建的任何缓存上设置编码配置。这对消耗的事件也至关重要。如需更多信息，请参阅官方 Infinispan 文档中的 数据编码和 MediaTypes。

45.11. 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.infinispan-embedded.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.infinispan-embedded.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.infinispan-embedded.cache-container	指定要连接的缓存容器。选项是 org.infinispan.manager.EmbeddedCacheManager 类型。		EmbeddedCacheManager
camel.component.infinispan-embedded.cache-container-configuration	CacheContainer 配置。如果没有定义 cacheContainer，则使用。选项是一个 org.infinispan.configuration.cache.Configuration 类型。		配置
camel.component.infinispan-embedded.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-embedded.configuration	组件配置.选项是 org.apache.camel.component.infinispan.embedded.InfinispanEmbeddedConfiguration 类型。		InfinispanEmbeddedConfiguration
camel.component.infinispan-embedded.configuration-uri	CacheManager 的实现特定 URI。		字符串

Name	描述	默认值	类型
camel.component.infinispan-embedded.custom-listener	如果提供，则返回正在使用的自定义监听程序。选项是 org.apache.camel.component.infinispan.embedded.InfinispanEmbeddedCustomListener 类型。		InfinispanEmbeddedCustomListener
camel.component.infinispan-embedded.enabled	是否启用 infinispan-embedded 组件的自动配置。这默认是启用的。		布尔值
camel.component.infinispan-embedded.event-types	指定要由消费者注册的事件类型集合。Multiple 事件可以用逗号分开。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, cache_ENTRY_EXPIRED, data_REHASHED, topOLOGY_CHANGED, partition_STATUS_CHANGED, persistence_AVAILABILITY_CHANGED.		字符串
camel.component.infinispan-embedded.flags	以逗号分隔的 org.infinispan.context.Flag 列表，默认在每个缓存调用中应用。		字符串

Name	描述	默认值	类型
camel.component.infinispan-embedded.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。通过延迟启动，启动失败可以在路由消息期间通过 Camel 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.infinispan-embedded.operation	要执行的操作。		InfinispanOperation
camel.component.infinispan-embedded.query-builder	指定查询构建器。选项是 org.apache.camel.component.infinispan.InfinispanQueryBuilder 类型。		InfinispanQueryBuilder
camel.component.infinispan-embedded.remapping-function	设置要在计算操作中使用的特定 remappingFunction。选项是一个 java.util.function.BiFunction 类型。		BiFunction
camel.component.infinispan-embedded.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，则该值将用作标头的名称，以存储查询结果，并保留原始消息正文。这个值可以被名为：CamelInfinispanOperationResultHeader 的消息标头覆盖。		字符串

Name	描述	默认值	类型
camel.component.infinispan-embedded.sync	如果为 true，消费者将同步接收通知。	true	布尔值

第 46 章 JACKSONXML

Jackson XML 是一个数据格式，它使用带有 XMLMapper 扩展的 Jackson 库，将一个 XML 有效负载 unmarshal 到一个 Java 对象，或将 Java 对象 marshal 到一个 XML 有效负载。注意：如果您熟悉 Jackson，此 XML 数据格式的行为与其 JSON 对应部分相同，因此可用于为 JSON 序列化/反序列化/序列化标注的类。

此扩展也模拟 JAXB 的"代码第一"方法。

此数据格式依赖于 Woodstox （特别是用于用户打印等功能），它是一个快速有效的 XML 处理器。

```
from("activemq:My.Queue").
  unmarshal().jacksonxml().
  to("mqseries:Another.Queue");
```

46.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 jacksonxml 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-jacksonxml-starter</artifactId>
</dependency>
```

46.2. JACKSONXML 选项

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	查找并使用给定 id 的现有 XmlMapper。
prettyPrint	false	布尔值	要启用用户化的打印输出，请执行以下操作：默认为 false。
unmarshalType		字符串	当 unmarshalling 时使用的 java 类型的类名称。

Name	默认值	Java 类型	描述
jsonView		字符串	当 marshalling a POJO to JSON 时，您可能想要从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现此目的。此选项是引用具有 JsonView 注释的类。
Include		字符串	如果您想 marshal a pojo to JSON，并且 pojo 具有一些带有 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		布尔值	用于 JMS 用户，以允许 JMS spec 中的 JMSType 标头指定一个 FQN 类名称来用于 unmarshal。
collectionType		字符串	引用要使用的自定义集合类型，以便在 registry 中查找。这个选项应该很少被使用，但允许使用与基于 java.util.Collection 不同的集合类型。
useList		布尔值	To unmarshal 到 Map 列表或 Pojo 的列表。
enableJaxbAnnotationModule		布尔值	在使用 jackson 时，是否启用 JAXB 注释模块。启用之后，Jackson 可以使用 JAXB 注释。
moduleClassNames		字符串	使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String with FQN 类名称。可以使用逗号分隔多个类。
moduleRefs		字符串	使用 Camel registry 中引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。
enableFeatures		字符串	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上启用的功能集合。这个功能应该是与 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature multiple features 分开的名称。
disableFeatures		字符串	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的功能集合。这个功能应该是与 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature multiple features 分开的名称。
allowUnmarshallType		布尔值	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。这只有在需要使用时才启用。
contentTypeHeader		布尔值	数据格式是否应该使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式到 XML 的 application/xml 或用于数据格式的 application/json 发送到 JSON。

46.2.1. 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。这在 DataFormats XML 标签中完成。

```
<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 unmarshalType is optional, if not provided Camel will use a Map as the type -->
    <jacksonxml id="jack" unmarshalType="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><custom ref="jack"/></unmarshal>
    <to uri="mock:reverse"/>
</route>
```

46.2.2. 从 marshalling 中排除 POJO 字段

当 marshalling a POJO to XML 时，您可能想要从 XML 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现此目的。首先创建一个或多个标记类。

使用带 @JsonView 注释的标记类来包含/排除某些字段。该注释也适用于 getters。

最后，使用 Camel JacksonXMLDataFormat 将以上 POJO 放入 XML。

请注意，生成的 XML 中缺少 weight 字段：

```
<pojo age="30" weight="70"/>
```

46.3. 使用带有 'JACKSONXML'DATAFORMAT 的 JSONVIEW 属性的 INCLUDE/EXCLUDE 字段

作为使用此属性的示例，您可以改为：

```
JacksonXMLDataFormat ageViewFormat = new JacksonXMLDataFormat(TestPojoView.class,
```

```
Views.Age.class);
from("direct:inPojoAgeView").
  marshal(ageViewFormat);
```

直接将 Java DSL 中的 JSON 视图 指定为：

```
from("direct:inPojoAgeView").
  marshal().jacksonxml(TestPojoView.class, Views.Age.class);
```

在 XML DSL 中相同：

```
<from uri="direct:inPojoAgeView"/>
  <marshal>
    <jacksonxml unmarshalType="org.apache.camel.component.jacksonxml.TestPojoView"
jsonView="org.apache.camel.component.jacksonxml.Views$Age"/>
  </marshal>
```

46.4. 设置序列化包括选项

如果您想 marshal a pojo to XML，并且 pojo 具有一些带有 null 值的字段。您想要跳过这些 null 值，那么您需要在 pojo 上设置注解，

```
@JsonInclude(Include.NON_NULL)
public class MyPojo {
   ...
}
```

但是，这要求您在 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");
```

或者来自 XML DSL，将其配置为

```
    <dataFormats>
      <jacksonxml id="jacksonxml" include="NON_NULL"/>
    </dataFormats>
```

46.5. 使用动态类名称从 XML 到 POJO 的 UNMARSHALING

如果您使用 jackson to unmarshal XML to POJO，那么您现在可以在消息中指定标头，以指示哪个类名称到 unmarshal。
如果消息中存在该标头，则标头的键为 CamelJacksonUnmarshalType，则 Jackson 会将该标头用作 POJO 类的 FQN，以免于 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);
```

或者来自 XML DSL，将其配置为

```
<dataFormats>
  <jacksonxml id="jacksonxml" allowJmsType="true"/>
</dataFormats>
```

46.6. 从 XML 到 LIST<MAP> 或 LIST<POJO>

如果您使用 Jackson unmarshal XML 到映射/pojo 列表，您现在可以通过设置 useList="true" 或使用 org.apache.camel.component.jacksonxml.ListJacksonXMLDataFormat 来指定这一点。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，您可以将 配置为使用 useList 属性，如下所示：

```
<dataFormats>
  <jacksonxml id="jack" useList="true"/>
</dataFormats>
```

您还可以指定 pojo 类型

```
<dataFormats>
  <jacksonxml id="jack" useList="true" unmarshalType="com.foo.MyPojo"/>
</dataFormats>
```

46.7. 使用自定义 JACKSON 模块

您可以使用 moduleClassNames 选项指定类名称来使用自定义 Jackson 模块，如下所示。

```
<dataFormats>
  <jacksonxml id="jack" useList="true" unmarshalType="com.foo.MyPojo"
moduleClassNames="com.foo.MyModule,com.foo.MyOtherModule"/>
</dataFormats>
```

在使用 moduleClassNames 时，不会配置自定义 jackson 模块，由使用默认构造器创建并按原样使用。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 modulesRefs 引用模块，如下所示：

```
<bean id="myJacksonModule" class="com.foo.MyModule">
  ... // configure the module as you want
</bean>

<dataFormats>
  <jacksonxml id="jacksonxml" useList="true" unmarshalType="com.foo.MyPojo"
moduleRefs="myJacksonModule"/>
</dataFormats>
```

```
Multiple modules can be specified separated by comma, such as
moduleRefs="myJacksonModule,myOtherModule"
```

46.8. 使用 JACKSON 启用或禁用功能

Jacks 具有多个功能，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 配置它：

```
<dataFormats>
  <jacksonxml id="jacksonxml" unmarshalType="com.foo.MyPojo"
disableFeatures="FAIL_ON_UNKNOWN_PROPERTIES"/>
</dataFormats>
```

您可以使用逗号分隔值来禁用多个功能。功能的值必须是以下 enum 类中 Jackson 的 enums 的名称

- com.fasterxml.jackson.databind.SerializationFeature
- com.fasterxml.jackson.databind.DeserializationFeature

- com.fasterxml.jackson.databind.MapperFeature

要启用某个功能，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);
```

46.9. 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 注册表中有一个 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。

46.10. 格式化的 XML MARSHALLING (PRETTY-PRINTING)

使用 prettyPrint 选项，可以在 marshalling 时输出一个良好格式化的 XML：

```
<dataFormats>
    <jacksonxml id="jack" prettyPrint="true"/>
</dataFormats>
```

在 Java DSL 中：

```
from("direct:inPretty").marshal().jacksonxml(true);
```

请注意，有 5 个不同的过载 jacksonxml （） DSL 方法支持 prettyPrint 选项以及

unmarshalType、jsonView 等其他设置。

46.11. 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，以允许 JMS spec 中的 JMSType 标头指定一个 FQN 类名称来用于 unmarshal。	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。这只有在需要使用时才启用。	false	布尔值
camel.dataformat.jacksonxml.collection-type	引用要使用的自定义集合类型，以便在 registry 中查找。这个选项应该很少被使用，但允许使用与基于 java.util.Collection 不同的集合类型。		字符串
camel.dataformat.jacksonxml.content-type-header	数据格式是否应该使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式到 XML 的 application/xml 或用于数据格式的 application/json 发送到 JSON。	true	布尔值
camel.dataformat.jacksonxml.disable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的功能集合。这个功能应该是与 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature multiple features 分开的名称。		字符串
camel.dataformat.jacksonxml.enable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上启用的功能集合。这个功能应该是与 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature multiple features 分开的名称。		字符串
camel.dataformat.jacksonxml.enable-jaxb-annotation-module	在使用 jackson 时，是否启用 JAXB 注释模块。启用之后，Jackson 可以使用 JAXB 注释。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.jacksonxml.enabled	是否启用 jacksonxml 数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.jacksonxml.include	如果您想 marshal a pojo to JSON，并且 pojo 具有一些带有 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.jacksonxml.json-view	当 marshalling a POJO to JSON 时，您可能想要从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现此目的。此选项是引用具有 JsonView 注释的类。		字符串
camel.dataformat.jacksonxml.module-class-names	使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String with 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 中引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。		字符串
camel.dataformat.jacksonxml.pretty-print	要启用用户化的打印输出，请执行以下操作：默认为 false。	false	布尔值
camel.dataformat.jacksonxml.unmarshal-type	当 unmarshalling 时使用的 java 类型的类名称。		字符串
camel.dataformat.jacksonxml.use-list	To unmarshal 到 Map 列表或 Pojo 的列表。	false	布尔值
camel.dataformat.jacksonxml.xml-mapper	查找并使用给定 id 的现有 XmlMapper。		字符串

第 47 章 JAXB

Since Camel 1.0

JAXB 是一个数据格式，它使用 JAXB XML marshalling 标准将 XML 有效负载合并到 Java 对象，或将 Java 对象放入 XML 有效负载中。

47.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 jaxb 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-jaxb-starter</artifactId>
</dependency>
```

47.2. 选项

JAXB dataformat 支持 20 个选项，如下所列。

Name	默认值	Java 类型	描述
contextPath		字符串	JAXB 类所在的 **必需** 软件包名称。
contextPathIsClassName	false	布尔值	这可设置为 true 来标记 contextPath 引用一个 classname 而不是软件包名称。
schema		字符串	针对现有架构进行验证。您可以使用前缀 classpath:、file: 或 http: 指定应如何解析资源。您可以使用 ',' 字符分隔多个架构文件。

Name	默认值	Java 类型	描述
schemaSeverityLevel	0	Enum	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 0 （警告）表示任何错误（警告、错误或严重错误）将触发 JAXB 来停止。有三个级别： 0=warning, 1=error, 2=fatal 错误。 Enum 值： • 0 • 1 • 2
prettyPrint	false	布尔值	要启用用户化的打印输出，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。这只适用于没有通过 JAXB 注解的 POJO 类，并提供 jaxb.index 描述符文件。
ignoreJAXBElement	false	布尔值	是否忽略 JAXBElement 元素 - 只在非常特殊用例中，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。这个选项可以被设置为 false 来放宽，比如当数据已采用 XML 格式时。
filterNonXmlChars	false	布尔值	忽略非 xml characheters，并将其替换为空空间。
编码		字符串	overrule 并使用特定的编码。
片段	false	布尔值	打开 marshalling XML 片段树。默认情况下，JAXB 在给定类上查找 XmlRootElement 注释，以便在整个 XML 树上运行。这很有用，但有时生成的代码没有 XmlRootElement 注解，有时您需要 unmarshall 只是树的一部分。在这种情况下，您可以使用 partial 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	用于碎片解析的类名称。请参阅 fragment 选项的更多详情。
partNamespace		字符串	用于碎片解析的 XML 命名空间。请参阅 fragment 选项的更多详情。
namespacePrefixRef		字符串	使用 JAXB 或 SOAP 总结时，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。

Name	默认值	Java 类型	描述
xmlStreamWriterWrapper		字符串	使用自定义 xml 流写器。
schemaLocation		字符串	定义架构的位置。
noNamespaceSchemaLocation		字符串	定义无命名空间模式的位置。
jaxbProviderProperties		字符串	指的是要在含有要与 JAXB marshaller 搭配使用的自定义 JAXB 提供程序属性的注册表中查找的自定义 java.util.Map。
contentTypeHeader	true	布尔值	数据格式是否应该使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式到 XML 的 application/xml 或用于数据格式的 application/json 发送到 JSON。
accessExternalSchemaProtocols	false	字符串	仅在启用了 schema 验证时使用。限制对由 schemaLocation 属性 Import 和 Include 元素设置的用于外部引用指定的协议的访问。协议的示例包括 file、http、jar:file. false 或 none 用于拒绝对外部引用的所有访问；特定的协议（如 file）来仅授予该协议的权限；关键字 all 为所有协议授予权限。

47.3. 使用 JAVA DSL

以下示例使用 jaxb 的命名 DataFormat，它配置有 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");
```

您可以使用命名引用的数据格式，然后在 Registry 中定义，如通过 Spring XML 文件。

```
from("activemq:My.Queue").
  unmarshal("myJaxbDataType").
  to("mqseries:Another.Queue");
```

47.4. 使用 SPRING XML

以下示例演示了如何配置 JaxbDataFormat 并在多个路由中使用它。

```
<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="myJaxb" class="org.apache.camel.converter.jaxb.JaxbDataFormat">
  <property name="contextPath" value="org.apache.camel.example"/>
 </bean>

 <camelContext xmlns="http://camel.apache.org/schema/spring">
  <route>
   <from uri="direct:start"/>
   <marshal><custom ref="myJaxb"/></marshal>
   <to uri="direct:marshalled"/>
  </route>
  <route>
   <from uri="direct:marshalled"/>
   <unmarshal><custom ref="myJaxb"/></unmarshal>
   <to uri="mock:result"/>
  </route>
 </camelContext>

</beans>
```

47.5. 多上下文路径

将此数据格式与多个上下文路径一起使用。您可以使用 : 指定多个上下文路径，如 com.mycompany:com.mycompany2。

47.6. 部分总结 / UNMARSHALLING

JAXB 2 支持 marshalling 和 unmarshalling XML 树片段。默认情况下，JAXB 在给定类上查找 @XmlRootElement 注释，以在整个 XML 树上运行。有时生成的代码没有 @XmlRootElement 注释，有时您需要 unmarshall 只是树的一部分。

在这种情况下，您可以使用 partial unmarshalling。若要启用此功能，您需要在 JaxbDataFormat 上设置属性 partClass。Camel 会将此类传递给 JAXB unmarshaller。如果将 JaxbConstants.JAXB_PART_CLASS 设置为交换标头之一，则其值用于覆盖 JaxbDataFormat 上的 partClass 属性。

对于 marshalling，您必须使用目标命名空间的 QName 添加 partNamespace 属性。

如果将 JaxbConstants.JAXB_PART_NAMESPACE 设置为一个交换标头，则其值用于覆盖 JaxbDataFormat 上的 partNamespace 属性。

在通过 JaxbConstants.JAXB_PART_NAMESPACE 设置 partNamespace 时，请注意您需要以 {namespaceUri}localPart 格式指定其值，如下例所示。

```
.setHeader(JaxbConstants.JAXB_PART_NAMESPACE, constant("
{http://www.camel.apache.org/jaxb/example/address/1}address"));
```

47.7. 片段

JaxbDataFormat 具有名为 fragment 的属性，它可以设置 JAXB Marshaller 上的 Marshaller.JAXB_FRAGMENT 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。

47.8. 忽略 NON-XML CHARACTERS

JaxbDataFormat 支持忽略 Non-XML Characters。将 filterNonXmlChars 属性设置为 true。JaxbDataFormat 将把任何非 XML 字符替换为空格字符('')，在消息 marshalling 或 unmarshalling 期间使用空格字符替换。您还可以设置 Exchange 属性 Exchange.FILTER_NON_XML_CHARS。

	JDK 1.5	JDK 1.6+
使用中的过滤	stax API 和实现	否
未使用过滤	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。

JaxbDataFormat 现在允许您自定义用于 marshal 流到 XML 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，同时启用 Camel 的非 XML 过滤：

```
<bean id="testXmlStreamWriterWrapper" class="org.apache.camel.jaxb.TestXmlStreamWriter"/>
<jaxb filterNonXmlChars="true" contextPath="org.apache.camel.foo.bar"
xmlStreamWriterWrapper="#testXmlStreamWriterWrapper" />
```

47.9. 使用 OBJECTFACTORY

如果您使用 XJC 从架构创建 java 类，您将收到 JAXB 上下文的 ObjectFactory。由于 ObjectFactory 使用 JAXBElement 来保存 schema 和 element 实例值的引用，JaxbDataformat 将默认忽略 JAXBElement，并且您将从 unmarshaled 消息正文获取 element 实例值，而不是 JAXBElement 对象。

如果要获取 JAXBElement 对象组成 unmarshaled 消息正文，您需要将 JaxbDataFormat ignoreJAXBElement 属性设置为 false。

47.10. 设置编码

您可以在 JaxbDataFormat 上设置 encoding 选项，以配置 JAXB Marshaller 上的 Marshaller.JAXB_ENCODING 编码属性。

您可以设置在声明 JaxbDataFormat 时要使用的编码。您还可以在 Exchange 属性 Exchange.CHARSET_NAME 中提供编码。此属性将覆盖 JaxbDataFormat 上设置的编码。

47.11. 控制命名空间前缀映射

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。

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 映射到将 soap 用作前缀。虽然我们的自定义命名空间 http://www.mycompany.com/foo/2 没有使用任何前缀。

```
<util:map id="myMap">
  <entry key="http://www.w3.org/2003/05/soap-envelope" value="soap"/>
  <!-- we don't want any prefix for our namespace -->
  <entry key="http://www.mycompany.com/foo/2" value=""/>
</util:map>
```

要在 JAXB 或 SOAP 数据格式中使用它，您可以使用 namespacePrefixRef 属性引用此映射，如下所示。然后，Camel 将在 Registry 中查找 ID 为 my Map 的 java.util.Map，这是我们所定义的内容。

```
<marshal>
  <soap version="1.2" contextPath="com.mycompany.foo" namespacePrefixRef="myMap"/>
</marshal>
```

47.12. 模式验证

JaxbDataFormat 支持通过 marshalling 和 unmarshalling 从 / 到 XML 进行验证。您可以使用前缀 classpath:、file: 或 http: 指定应如何解析资源。您可以使用 、 字符分隔多个架构文件。

注意

如果 XSD 模式文件导入/访问其他文件，则需要启用文件协议（或其他人允许访问）。

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");
jaxbDataFormat.setAccessExternalSchemaProtocols("file");
```

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"
      accessExternalSchemaProtocols="file"/>
</marshal>
```

47.13. 模式位置

JaxbDataFormat 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>
```

47.14. 已 XML 的 MARSHAL 数据

JAXB marshaller 要求消息正文兼容 JAXB，例如它是 JAXBElement，是具有 JAXB 注释的 java 实例，或者扩展 JAXBElement。在某些情况下，消息正文已经在 XML 中，例如来自 String 类型。

JaxbDataFormat 具有一个名为 mustBeJAXBElement 的选项，您可以将其设置为 false 以放宽此检查，并且只有 JAXB marshaller 尝试在 JAXBElement 上尝试 marshalling (javax.xml.bind.JAXBIntrospector#isElement 返回 true)。在这些情况下，marshaller 将回退到 marshal 消息正文（按原样）。

47.15. 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.dataformat.jaxb.access-external-schema-protocols	仅在启用了 schema 验证时使用。限制对由 schemaLocation 属性 Import 和 Include 元素设置的用于外部引用指定的协议的访问。协议的示例包括 file、http、jar:file. false 或 none 用于拒绝对外部引用的所有访问；特定的协议（如 file）来仅授予该协议的权限；关键字 all 为所有协议授予权限。	false	字符串
camel.dataformat.jaxb.content-type-header	数据格式是否应该使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式到 XML 的 application/xml 或用于数据格式的 application/json 发送到 JSON。	true	布尔值
camel.dataformat.jaxb.context-path	JAXB 类所在的软件包名称。		字符串

Name	描述	默认值	类型
camel.dataformat.jaxb.context-path-is-class-name	这可设置为 true 来标记 contextPath 引用一个 classname 而不是软件包名称。	false	布尔值
camel.dataformat.jaxb.enabled	是否启用 jaxb 数据格式的自动配置。这默认是启用的。		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	忽略非 xml characheters，并将其替换为空空间。	false	布尔值
camel.dataformat.jaxb.fragment	打开 marshalling XML 片段树。默认情况下，JAXB 在给定类上查找 XmlRootElement 注释，以便在整个 XML 树上运行。这很有用，但有时生成的代码没有 XmlRootElement 注解，有时您需要 unmarshall 只是树的一部分。在这种情况下，您可以使用 partial 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	是否忽略 JAXBElement 元素 - 只在非常特殊用例中，只需要设置为 false。	false	布尔值
camel.dataformat.jaxb.jaxb-provider-properties	指的是要在含有要与 JAXB marshaller 搭配使用的自定义 JAXB 提供程序属性的注册表中查找的自定义 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。这个选项可以被设置为 false 来放宽，比如当数据已采用 XML 格式时。	false	布尔值
camel.dataformat.jaxb.namespace-prefix-ref	使用 JAXB 或 SOAP 总结时，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。这只适用于没有通过 JAXB 注解的 POJO 类，并提供 jaxb.index 描述符文件。	false	布尔值
camel.dataformat.jaxb.part-class	用于碎片解析的类名称。请参阅 fragment 选项的更多详情。		字符串
camel.dataformat.jaxb.part-namespace	用于碎片解析的 XML 命名空间。请参阅 fragment 选项的更多详情。		字符串
camel.dataformat.jaxb.pretty-print	要启用用户化的打印输出，请执行以下操作：默认为 false。	false	布尔值
camel.dataformat.jaxb.schema	针对现有架构进行验证。您可以使用前缀 classpath:、file: 或 http: 指定应如何解析资源。您可以使用 ',' 字符分隔多个架构文件。		字符串
camel.dataformat.jaxb.schema-location	定义架构的位置。		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 0 （警告）表示任何错误（警告、错误或严重错误）将触发 JAXB 来停止。有三个级别： 0=warning, 1=error, 2=fatal 错误。	0	整数
camel.dataformat.jaxb.xml-stream-writer-wrapper	使用自定义 xml 流写器。		字符串

第 48 章 JASYPT

从 Camel 2.5 开始

Jasypt 是一个简化的加密库，可以轻松地加密和解密。Camel 与 Jasypt 集成，以允许 加密 属性文件中的敏感信息。在类路径上丢弃 camel-jasypt，这些加密值将自动被 Camel 解密。这样可确保人类眼于不容易发现敏感信息，如用户名和密码。

48.1. 依赖项

当在 Camel Spring Boot 中使用 camel-jasypt 时，请将以下 Maven 依赖项添加到 pom.xml 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-jasypt-starter</artifactId>
</dependency>
```

48.2. 工具

Jasypt 组件是一个可运行的 JAR，它提供了一个命令行工具来加密或解密值。使用文档可以输出到控制台，以描述它提供的语法和选项：

```
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
 -rsga or -algorithm <algorithm> = Optional random salt generator algorithm to use
 -riga or -algorithm <algorithm> = Optional random iv generator algorithm to use
```

运行该工具的简单方法是使用 JBang。例如，要加密值 tiger，您可以使用以下参数：确保指定要使用的 camel-jasypt 版本。

```
$ jbang org.apache.camel:camel-jasypt:<camel version here> -c encrypt -p secret -i tiger
```

哪些输出结果

```
Encrypted text: qaEEacuW7BUti8LcMgyjKw==
```

这意味着，如果您知道 secret 的 master 密码，加密表示 qaEEacuW7BU ti 8LcMgyjKw== 可以被解密。

如果您再次运行该工具，则加密的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。

您可以使用以下参数运行工具来测试解密值：

```
$ jbang org.apache.camel:camel-jasypt:<camel version here> -c decrypt -p secret -i
qaEEacuW7BUti8LcMgyjKw==
```

哪些输出结果如下：

```
Decrypted text: tiger
```

然后，在 Properties 文件中使用这些加密值。例如，

```
# Encrypted value for 'tiger'
my.secret = ENC(qaEEacuW7BUti8LcMgyjKw==)
```

48.3. 保护 MASTER 密码

必须提供 Jasypt 使用的 master 密码，以便可以解密值。但是，在打开时使用此 master 密码可能不是理想的解决方案。因此，您可以将其作为 JVM 系统属性或 OS 环境设置提供。如果您决定这样做，则 password 选项支持指定此前缀的前缀。

- sysenv：使用给定密钥查找操作系统环境。
- sys：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
```

在 Spring Boot 和 Quarkus 等运行时，您可以在 application.properties 文件中配置 password 属性，如下所示：

```
password=sysenv:CAMEL_ENCRYPTION_PASSWORD
```

或者，如果手动配置 JasyptPropertiesParser，您可以设置如下密码。

```
jasyptPropertiesParser.setPassword("sysenv:CAMEL_ENCRYPTION_PASSWORD");
```

48.4. 使用 JAVA DSL 的示例

在 Spring Boot 和 Quarkus 运行时中，Camel Jasypt 可通过配置属性进行配置。如需更多信息，请参阅对应的文档页面。

在 Java DSL 中，您需要将 Jasyptpt 配置为 JasyptPropertiesParser 实例，并在 Properties 组件中设置属性，如下所示：

```
// create the jasypt properties parser
JasyptPropertiesParser jasypt = new JasyptPropertiesParser();
// set the master password (see above for how to do this in a secure way)
jasypt.setPassword("secret");

// create the properties' component
PropertiesComponent pc = new PropertiesComponent();
pc.setLocation("classpath:org/apache/camel/component/jasypt/secret.properties");
// and use the jasypt properties parser, so we can decrypt values
pc.setPropertiesParser(jasypt);
// end enable nested placeholder support
pc.setNestedPlaceholder(true);

// add properties component to camel context
context.setPropertiesComponent(pc);
```

可以在 JasyptPropertiesParser 上配置自定义算法，如下所示：

```
JasyptPropertiesParser jasyptPropertiesParser = new JasyptPropertiesParser();

jasyptPropertiesParser.setAlgorithm("PBEWithHmacSHA256AndAES_256");
jasyptPropertiesParser.setRandomSaltGeneratorAlgorithm("PKCS11");
jasyptPropertiesParser.setRandomIvGeneratorAlgorithm("PKCS11");
```

属性文件 secret.properties 将包含加密的配置值，如下所示。请注意密码值如何加密，并周围为 ENC（这里的值）。

```
my.secret.password=ENC(bsW9uV37gQ0QHFu7KO03Ww==)
```

48.5. SPRING XML 示例

在 Spring XML 中，您需要配置 JasyptPropertiesParser，它如下所示。然后，Camel Properties 组件被告知使用 jasypt 作为属性解析程序，这意味着 Jasypt 具有解密值的机会。

```
<!-- 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/secret.properties"/>
    <!-- and let it leverage the jasypt parser -->
    <property name="propertiesParser" ref="jasypt"/>
    <!-- end enable nested placeholder -->
    <property name="nestedPlaceholder" value="true"/>
</bean>
```

Properties 组件也可以内联在 < camelContext> 标签内，如下所示。请注意，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"
                         nestedPlaceholder="true"
```

```
                propertiesParserRef="jasypt"/>
    <route>
        <from uri="direct:start"/>
        <to uri="{{cool.result}}"/>
    </route>
</camelContext>
```

48.6. 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.jasypt.algorithm	用于解密的算法。	PBEWithMD5AndDES	字符串
camel.component.jasypt.enabled	启用组件。	false	布尔值
camel.component.jasypt.iv-generator-class-name	在解密操作中应用的初始化向量(IV)生成器。默认：org.jasypt.iv。		字符串
camel.component.jasypt.password	Jasypt 用于解密值的主密码。此选项支持影响 master 密码查找的前缀：sysenv: 是指使用给定键查找 OS 系统环境。sys: 表示查找 JVM 系统属性。		字符串
camel.component.jasypt.provider-name	用于获取加密算法的安全提供程序的类名称。		字符串
camel.component.jasypt.random-iv-generator-algorithm	随机 iv 生成器的算法。	SHA1PRNG	字符串
camel.component.jasypt.random-salt-generator-algorithm	salt 生成器的算法。	SHA1PRNG	字符串

Name	描述	默认值	类型
camel.component.jasypt.salt-generator-class-name	在解密操作中应用的 salt 生成器。默认：org.jasypt.salt.RandomSaltGenerator。	org.jasypt.salt.RandomSaltGenerator	字符串

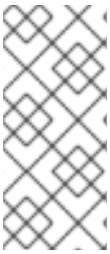
4.0// ParentAssemblies: assemblies/

第 49 章 JDBC

自 Camel 1.2 开始

仅支持生成者

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 等)在消息正文中发送。此组件使用标准 JDBC API。



注意

先决条件

这个组件不支持开箱即用的事务。对于事务，我们建议使用 Spring JDBC 组件。

49.1. 依赖项

当在 Camel Spring Boot 中使用 jdbc 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-jdbc-starter</artifactId>
</dependency>
```

49.2. URI 格式

```
jdbc:dataSourceName[?options]
```

49.3. 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

49.3.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 URL。

组件具有常用的默认值。因此，您只需要在组件上配置几个选项或根本不配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

49.3.2. 配置端点选项

端点有许多选项，允许您配置您需要的端点。这些选项也被归类为：端点是否用作消费者（来自）还是作为生成者(to)用于两者。

配置端点直接在端点 URI 中作为路径和查询参数完成。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式 的安全 方法。

当配置选项使用 Property Placeholders 时，它允许硬编码 URL、端口号、敏感信息和其他设置。占位符允许从您的代码外部配置，并提供更大的灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

49.4. 组件选项

JDBC 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
datasource (producer)	使用 DataSource 实例，而不是根据 registry 中的名称查找数据源。		DataSource

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
autowiredEnabled (adcanced)	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
connectionStrategy (advanced)	使用自定义策略来使用连接。在使用 spring-jdbc 组件时不要使用自定义策略，因为默认使用特殊的 Spring ConnectionStrategy 支持 Spring Transactions。		ConnectionStrategy

49.5. 端点选项

JDBC 端点使用 URI 语法进行配置：

```
jdbc:dataSourceName
```

使用以下 path 和 查询参数：

49.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
dataSourceName (producer)	在 Registry 中查找所需的数据源名称。**如果名称是 dataSource 或 default，则 Camel 将尝试从 registry 中查找默认 DataSource，这意味着如果只有一个找到的 DataSource 实例，则会使用此 DataSource。**		字符串

49.5.2. 查询参数 (14 参数)

Name	描述	默认值	类型
allowNamedParameters (producer)	是否在查询中使用命名参数。	True	布尔值
outputClass (producer)	指定在 outputType=SelectOne 或 SelectList 时用作转换的完整软件包和类名称。		字符串
outputType (producer)	确定制作者应使用的输出。Enum 值： • SelectOne • SelectList • StreamList	SelectList	JdbcOutputType
parameters (producer)	java.sql.Statement 的可选参数。例如，要设置 maxRows、fetchSize 等。		Map
readSize (producer)	轮询查询可以读取的默认最大行数。默认值为 0。		int
resetAutoCommit (producer)	Camel 将 JDBC 连接上的 autoCommit 设置为 false，在执行声明后提交更改，并在末尾重置连接的 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，您很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)	是否使用事务。	False	布尔值
useGetBytesForBlob (producer)	以字节而不是字符串数据形式读取 BLOB 列。对于某些数据库（如 Oracle ）可能需要这个数据库，其中必须以字节形式读取 BLOB 列。	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。这允许使用指定占位符定义查询，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 或 JDBC 3.0 的旧语义。JDBC 4.0 使用 columnLabel 获取列名称，其中 JDBC 3.0 使用 columnName 或 columnLabel。不幸的是，JDBC 驱动程序的行为不同，如果您使用此组件解决了这个问题，则可以使用此选项来排除 JDBC 驱动程序的问题。	True	布尔值

Name	描述	默认值	类型
lazyStartProducer (producer (advanced))	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
beanRowMapper (advanced)	使用 outputClass 时使用自定义 org.apache.camel.component.jdbc.BeanRowMapper 。默认实现会小写，行名称和跳过下划线和横线。例如 CUST_ID 被映射为 custId。		BeanRowMapper
connectionStrategy (advanced)	使用自定义策略来使用连接。在使用 spring-jdbc 组件时不要使用自定义策略，因为默认使用特殊的 Spring ConnectionStrategy 支持 Spring Transactions。		ConnectionStrategy
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。		JdbcPrepareStatementStrategy

49.6. 消息标头

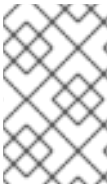
JBBC 组件支持 8 个消息标头，如下所列：

Name	描述	默认值	类型
CamelJdbcUpdateCount (producer) 常量： JDBC_UPDATE_COUNT	如果查询是 UPDATE，则查询此 OUT 标头中返回更新计数。		int
CamelJdbcRowCount (producer) 常数： JDBC_ROW_COUNT	如果查询是 SELECT，则查询此 OUT 标头中返回的行数。		int
CamelJdbcColumnNames (producer) 常数： JDBC_COLUMN_NAMES	ResultSet 中的列名称作为 java.util.Set 类型。		Set

Name	描述	默认值	类型
CamelJdbcParameters (producer) 常数： JDBC_PARAMETERS	如果启用了 useHeadersAsParameters，具有要使用的标头的 **java.util.Map**。		Map
CamelRetrieveGeneratedKeys (producer) 常量： JDBC_RETRIEVE_GENERATED_KEYS	将其值设为 true 以检索生成的密钥。	False	布尔值
CamelGeneratedColumns (producer) 常数： JDBC_GENERATED_COLUMNS	把它设置为指定预期的生成的列。		string[] 或 int[]
CamelGeneratedKeysRowCount (producer) 常数： JDBC_GENERATED_KEYS_ROW_COUNT	包含生成的键的标头中的行数。		int
CamelGeneratedKeysRows (producer) 常量： JDBC_GENERATED_KEYS_DATA	包含生成的密钥的行。		list

49.7. 结果

默认情况下，OUT 正文中作为 ArrayList<HashMap<String, Object& gt;> 返回的结果。List 对象包含行列表，而 Map 对象包含每行的 String 键作为列名称。您可以使用选项 outputType 来控制结果。



注意

此组件获取 ResultSetMetaData，以便能够将列名称返回为 Map 中的键。

49.8. 生成的密钥

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 producer 在标头中返回生成的密钥。为此，可设置标头 CamelRetrieveGeneratedKeys=true。然后，生成的密钥将以标头形式提供，其中包含上表中列出的键。

注意

使用生成的密钥不能与命名参数一起使用。

49.9. 使用命名参数

在以下给定路由中，我们希望从 projects 表中获取所有项目。请注意，SQL 查询具有 2 个命名的参数 :?lic 和 :?min. Camel，然后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(simple("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，并使用键 CamelJdbcParameters 在标头中存储映射。

49.10. SAMPLES

在以下示例中，我们将设置 camel-jdbc 所需的 DataSource。首先，我们将 Camel 注册表中的数据源注册为 testdb ：

```
EmbeddedDatabase db = new EmbeddedDatabaseBuilder()
    .setType(EmbeddedDatabaseType.DERBY).addScript("sql/init.sql").build();

CamelContext context = ...
context.getRegistry().bind("testdb", db);
```

然后，我们将配置路由到 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：

```
from("direct:hello")
  .to("jdbc:testdb");
```

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。查询的结果会在 OUT 正文中返回：

```
Endpoint endpoint = context.getEndpoint("direct:hello");
Exchange exchange = endpoint.createExchange();
// then we set the SQL on the in body
exchange.getMessage().setBody("select * from customer order by ID");
// now we send the exchange to the endpoint, and receives the response from Camel
Exchange out = template.send(endpoint, exchange);
```

如果您想一次在一行中工作，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");
```

49.11. 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，我们需要将它与轮询调度程序（如 Timer 或 Quartz 等）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");
```

49.12. 示例 - 在数据源之间移动数据

常见用例是查询数据，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");
```

49.13. 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.jdbc.autowired-enabled	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
camel.component.jdbc.connection-strategy	使用自定义策略来使用连接。在使用 spring-jdbc 组件时不要使用自定义策略，因为默认使用特殊的 Spring ConnectionStrategy 支持 Spring Transactions。选项是一个 org.apache.camel.component.jdbc.ConnectionStrategy 类型。		ConnectionStrategy
camel.component.jdbc.enabled	是否启用 jdbc 组件的自动配置。这默认是启用的。		布尔值
camel.component.jdbc.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

第 50 章 JIRA

支持生成者和消费者

JIRA 组件通过封装 Atlassian 的 REST Java 客户端 JIRA 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、添加注释、更改问题、添加/删除监视器、添加附件并转换问题状态。

此端点依赖于简单的轮询，而不是 Webhook。原因包括：

- 对可靠性/可靠性的关注
- 我们轮询的有效负载类型通常不大（加分页可在 API 中提供）
- 在 Webhook 失败时，需要支持在某个地方无法公开访问的应用程序

请注意 JIRA API 的扩展比较合理。因此，这个组件可以被轻松扩展，以提供额外的交互。

50.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 jira 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-jira-starter</artifactId>
</dependency>
```

50.2. URI 格式

```
jira://type[?options]
```

JIRA 类型接受以下操作：

对于消费者：

- newIssues ：在路由启动后只检索新问题
- 新注释：在路由启动后仅检索新的注释
- watchUpdates ：只检索根据提供的 jql 更新的字段问题

对于制作者：

- addIssue ：添加一个问题
- 添加注释：对给定问题添加注释
- attach：在给定问题中添加附件
- deleteIssue ：删除给定问题
- updateIssue ：更新给定问题的字段
- transitionIssue ：转换给定问题的状态
- watchers：添加/删除给定问题的监视者

当 JIRA 可以完全自定义时，您必须确保项目和工作流存在字段 ID，因为它们在不同的 JIRA 服务器之间可能会改变。

50.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

50.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

50.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

50.4. 组件选项

JIRA 组件支持 12 个选项，如下所列。

Name	描述	默认值	类型
delay (common)	下一次轮询经过的时间（毫秒）。	6000	整数
jiraUrl (common)	**必需** JIRA 服务器 url，例如：		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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
configuration (advanced)	使用共享基础 jira 配置。		JiraConfiguration
accessToken (security)	（仅限OAuth） JIRA 服务器生成的访问令牌。		字符串
consumerKey (security)	(OAuth only) JIRA 设置中的使用者密钥。		字符串
password (security)	（仅限基本身份验证）用于向 JIRA 服务器进行身份验证的密码。仅在使用用户名基本身份验证时使用。		字符串
PrivateKey (security)	（仅限 OAuth）客户端生成的私钥，以加密与服务器的对话。		字符串
用户名 （安全性）	（仅限基本身份验证）向 JIRA 服务器进行身份验证的用户名。仅在 JIRA 服务器上没有启用 OAuth 时才使用。如果同时设置了 username 和 OAuth 令牌参数，则不要设置它们，则用户名基本身份验证具有优先权。		字符串
verifyCode (security)	（仅限OAuth）授权 proccess 的第一个步骤中生成的 JIRA 的验证代码。		字符串

50.5. 端点选项

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：

50.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
type (common)	**执行** 所需的操作。消费者：NewIssues，新评论.生产者：添加Isue、AttachFile、DeleteIsue、TransitionIssue、UpdateIssue、Watchers。如需更多信息，请参阅此类 javadoc 描述。 Enum 值： • ADDCOMMENT • ADDISSUE • ATTACH • DELETEISSUE • NEWISSUES • NEWCOMMENTS • WATCHUPDATES • UPDATEISSUE • TRANSITIONISSUE • WATCHERS • ADDISSUELINK • ADDWORKLOG • FETCHISSUE • FETCHCOMMENTS		JiraType

50.5.2. 查询参数 (16 参数)

Name	描述	默认值	类型
delay (common)	下一次轮询经过的时间（毫秒）。	6000	整数
jiraUrl (common)	**必需** JIRA 服务器 url，例如：		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
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 product 键。务必要使用 RAW （）并在其中设置 JQL 以防止 camel 解析它，例如： RAW （project in (MYP, COM) AND resolution = Unresolved。		字符串
maxResults (consumer)	要搜索的最大问题数。	50	整数
sendOnlyUpdatedField (consumer)	仅发送交换正文或问题对象中已更改字段的指示器。默认情况下，消费者仅发送更改的字段。	true	布尔值
watchedFields (consumer)	要监视更改的以逗号分隔的字段列表。status,Priority 是默认值。	status, Priority	字符串
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern

Name	描述	默认值	类型
lazyStartProduce r (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
accessToken (security)	（仅限OAuth） JIRA 服务器生成的访问令牌。		字符串
consumerKey (security)	(OAuth only) JIRA 设置中的使用者密钥。		字符串
password (security)	（仅限基本身份验证）用于向 JIRA 服务器进行身份验证的密码。仅在使用用户名基本身份验证时使用。		字符串
PrivateKey (security)	（仅限 OAuth）客户端生成的私钥，以加密与服务器的对话。		字符串
用户名 （安全性）	（仅限基本身份验证）向 JIRA 服务器进行身份验证的用户名。仅在 JIRA 服务器上没有启用 OAuth 时才使用。如果同时设置了 username 和 OAuth 令牌参数，则不要设置它们，则用户名基本身份验证具有优先权。		字符串
verifyCode (security)	（仅限OAuth）授权 proccess 的第一个步骤中生成的 JIRA 的验证代码。		字符串

50.6. CLIENT FACTORY

您可以将 JIRARestClientFactory 与 registry 中名为 JIRARestClientFactory 的绑定，使其在 JIRA 端点中自动设置。

50.7. 身份验证

camel-jira 支持 基本身份验证 和 OAuth 3 委派的身份验证。

我们建议尽可能使用 OAuth，因为它为您的用户和系统提供最佳安全性。

50.7.1. 基本身份验证要求：

- 用户名和密码

50.7.2. OAuth 身份验证要求：

按照 JIRA OAuth 文档中的 教程生成客户端私钥、使用者密钥、验证代码和访问令牌。

- 在您的系统上本地生成的私钥。
- JIRA 服务器生成的验证代码。
- consumer 密钥，在 JIRA 服务器设置中设置。
- JIRA 服务器生成的访问令牌。

50.8. JQL

JQL URI 选项供两个消费者端点使用。在理论上，如 "project key" 等项目可以是 URI 选项本身。但是，通过需要使用 JQL，使用者变得更加灵活、强大。

至少，消费者需要以下内容：

```
jira://[type]?[required options]&jql=project=[project key]
```

需要注意的一件事是 newIssues 消费者将自动设置为 JQL：

- 将 ORDER BY key desc 附加到 JQL
- prepend id > latestIssueId，以检索在 camel 路由启动后添加的问题。

这是为了优化启动处理，而不必索引项目中每一个问题。

另一个注意的是，新注释使用者必须索引项目中每一个问题 和注释。因此，对于大型项目，尽可能优化 JQL 表达式非常重要。例如，JIRA Toolkit 插件在查询中包含 "Number of comments" custom field iwl-wagonuse '"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://[type]?[required options]&jql=RAW(project=[project key] AND status in (Open, \"Coding In Progress\") AND \"Number of comments\">0)"
```

50.9. 操作

在使用 JIRA 操作时，请参阅要设置的所需标头列表。producer 的 author 字段会在 JIRA 端自动设置为经过身份验证的用户。

如果没有设置任何必填字段，则抛出 IllegalArgumentException。

有一些操作需要 id 用于字段，如：问题类型、优先级、转换。检查 jira 项目的有效 id，因为它们可能在 jira 安装和项目工作流上有所不同。

50.10. ADDISSUE

必需：

- ProjectKey ：项目密钥，例如：CAMEL、HHH、MYP。
- IssueTypeId 或 IssueTypeName: 问题类型的 id 或问题类型的名称，您可以在 http://jira_server/rest/api/2/issue/createmeta?projectKeys=SAMPLE_KEY 中看到有效列表。
- IssueSummary: 问题概述。

可选：

- IssueAssignee: assignee 用户
- IssuePriorityId 或 IssuePriorityName ：问题的优先级，您可以在 http://jira_server/rest/api/2/priority 中看到有效列表。

- ***IssueComponents: 具有有效组件名称的字符串列表。***
- ***IssueWatchersAdd ：要添加到监视器列表中的用户名字符串列表。***
- ***IssueDescription:** 问题**的描述。***

50.11. ADDCOMMENT

必需：

- ***IssueKey ：**问题键标识**符。***
- ***交换的正文是描述。***

50.12. ATTACH

每个调用中只有一个文件。

必需：

- ***IssueKey ：**问题键标识**符。***
- ***交换的正文应该是 文件类型文件***

50.13. DELETEISSUE

必需：

- ***IssueKey ：**问题键标识**符。***

50.14. TRANSITIONISSUE

必需：

- ***IssueKey ：**问题键标识**符。***
- ***IssueTransitionId ：**问题转换 **id。***
- ***交**换**的正文是描述。***

50.15. UPDATEISSUE

- ***IssueKey ：**问题键标识**符。***
- ***IssueTypeId 或 IssueTypeName:** 问题类**型的 id 或**问题类**型的名称，您可以在 http://jira_server/rest/api/2/issue/createmeta?projectKeys=SAMPLE_KEY 中看到有效列表。***
- ***IssueSummary:** 问题**概述。***
- ***IssueAssignee: assignee 用**户***
- ***IssuePriorityId 或 IssuePriorityName ：**问题**的**优先**级，您可以在 http://jira_server/rest/api/2/priority 中看到有效列表。***
- ***IssueComponents: 具有有效**组**件名称的字符串列表。***
- ***IssueDescription:** 问题**的描述。***

50.16. WATCHER

- ***IssueKey ：**问题键标识**符。***
- ***IssueWatchersAdd ：要添加到**监视**器列表中的用**户**名字符串列表。***

- *IssueWatchersRemove: 一个字符串列表，它带有要从 watcher 列表中删除的用户名。*

50.17. WATCHUPDATES (CONSUMER)

- *watchedFields Comma separated list of fields 以监控更改，如 Status,Priority,Assignee,Components 等。*

- *sendOnlyUpdatedField By 默认仅更改的字段作为正文发送。*

所有消息也包含以下标头，它们添加了有关更改的额外信息：

- *issueKey ：更新的问题的密钥*

- *changed: 更新字段的名称（即 Status）*

- *watchedIssues: 更新时监视的所有问题键的列表*

50.18. 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.jira.access-token	（仅限OAuth） JIRA 服务器生成的访问令牌。		字符串
camel.component.jira.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.jira.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.jira.configuration	使用共享基础 jira 配置。选项是 org.apache.camel.component.jira.JiraConfiguration 类型。		JiraConfiguration
camel.component.jira.consumer-key	(OAuth only) JIRA 设置中的使用者密钥。		字符串
camel.component.jira.delay	下一次轮询经过的时间（毫秒）。	6000	整数
camel.component.jira.enabled	是否启用 jira 组件的自动配置。这默认是启用的。		布尔值
camel.component.jira.jira-url	JIRA 服务器 url，例如： http://my_jira.com:8081/。		字符串
camel.component.jira.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.jira.password	（仅限基本身份验证）用于向 JIRA 服务器进行身份验证的密码。仅在使用用户名基本身份验证时使用。		字符串
camel.component.jira.private-key	（仅限 OAuth）客户端生成的私钥，以加密与服务器的对话。		字符串
camel.component.jira.username	（仅限基本身份验证）向 JIRA 服务器进行身份验证的用户名。仅在 JIRA 服务器上没有启用 OAuth 时才使用。如果同时设置了 username 和 OAuth 令牌参数，则不要设置它们，则用户名基本身份验证具有优先权。		字符串

Name	描述	默认值	类型
camel.component.jira.verification-code	（仅限OAuth）授权 proccess 的第一个步骤中生成的 JIRA 的验证代码。		字符串

第 51 章 JMS

支持生成者和消费者

此组件允许将消息发送到（或从中消耗）一个 JMS Queue 或 Topic。它使用 Spring 的 JMS 支持来声明性事务，包括 Spring 的 JmsTemplate 发送和 MessageListenerContainer 以供使用。

51.1. 依赖项

在将 jms 与 Red Hat build of Camel Spring Boot 搭配使用时，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-jms-starter</artifactId>
</dependency>
```

注意

使用 ActiveMQ

时使用了 Apache ActiveMQ，则应优先选择 ActiveMQ 组件，因为它针对 ActiveMQ 进行了优化。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

请查看以下第 Transactions 和 Cache Levels 部分（如果使用带有 JMS 的事务，因为它可能会影响性能）。

注意

通过 JMS

Reply over JMS确保阅读本页中的 Request-reply 部分，以了解有关请求/回复的重要备注，因为 Camel 提供了多个用于性能配置的选项以及集群环境。

51.2. 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，targetName 被解释为队列名称。例如，要连接到队列，FOO.BAR 使用：

```
jms:FOO.BAR
```

如果需要，您可以包含可选的 queue: 前缀：

```
jms:queue:FOO.BAR
```

要连接到一个主题，您必须包含 topic: 前缀。例如，要连接到主题 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&option=value&...
```

51.2.1. 使用 ActiveMQ

JMS 组件重复使用 Spring 2 的 JmsTemplate 发送消息。这不是在非J2EE 容器中使用的理想选择，在 JMS 提供程序中通常需要一些缓存来避免 性能不佳。

如果要将 Apache ActiveMQ 用作您的消息代理，则建议进行以下操作之一：

- 使用 ActiveMQ 组件，该组件已优化，以便有效地使用 ActiveMQ
- 使用 ActiveMQ 中的 PoolingConnectionFactory。

51.2.2. 事务和缓存级别

如果您消耗消息并使用事务(transacted=true)，则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。

如果没有使用 XA，您应该考虑缓存速度，因为它会加快性能，如设置 cacheLevelName=CACHE_CONSUMER。

cacheLevelName 的默认设置是 CACHE_AUTO。这个默认自动检测模式，并将缓存级别相应地设置为：

- CACHE_CONSUMER if transacted=false
- CACHE_NONE if 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。

51.2.3. 带有 JMS 1.1 的持久化订阅

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。

注意

如果您使用 Apache ActiveMQ Classic，您可能更喜欢使用名为 Virtual Topic 的功能。这应该删除具有唯一 clientId 的要求。您可以查阅 Artemis 或 ActiveMQ Classic 的特定文档，以了解有关如何使用此功能的详细信息。您可以在此处找到有关 ActiveMQ 经典持久消息传递的更多详细信息。http://activemq.apache.org/how-do-durable-queues-and-topics-work.html

51.2.3.1. 带有 JMS 2.0 的持久化订阅

如果要使用持久主题订阅，则需要指定 durableSubscriptionName。

51.2.4. 消息标头映射

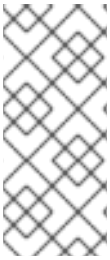
在使用消息标头时，JMS 规格指出标头名称必须是有效的 Java 标识符。因此，尝试将您的标头命名为有效的 Java 标识符。这样做的一个好处是，您可以在 JMS Selector 中使用标头（其 SQL92 语法强制用于标头的 Java 标识符语法）。

默认使用用于映射标头名称的简单策略。该策略是替换标头名称中的任何句点和连字符，如下所示，并在标头名称从通过有线发送的 JMS 消息恢复时反转替换。这意味着什么？不再丢失在 bean 组件上调用的方法名称，不再增加文件组件的文件名标头，以此类推。

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，在 Camel 使用消息时替换替换
- hyphen 被 'HYPHEN' 替代，当 Camel 使用消息时替换替换。

您可以在 JMS 端点上配置许多不同的属性，它们映射到 JMSConfiguration 对象上的属性。



注意

映射到 Spring JMS
很多属性映射到 Spring JMS 的属性，Camel 用来发送和接收信息。因此，您可以通过咨询相关的 Spring 文档来获取这些属性的更多信息。

51.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

51.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

51.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

51.4. 组件选项

JMS 组件支持 98 选项，如下所列。

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，且只能被单个 JMS 连接实例使用。**clientId** 选项与 JMS 1.1 durable 主题订阅相一致，因为客户端 ID 用于控制必须存储哪些客户端消息。使用 JMS 2.0 客户端时，clientId 可能会省略，这会创建一个 'global' 订阅。如果使用 Apache ActiveMQ，您可能更喜欢使用 Virtual Topics。		字符串
ConnectionFactory (common)	要使用的连接工厂。连接工厂必须在组件或端点上配置。		ConnectionFactory

Name	描述	默认值	类型
disableReplyTo (common)	指定 Camel 是否忽略消息中的 JMSReplyTo 标头。如果为 true，Camel 不会向 JMSReplyTo 标头中指定的目的地发送回复。如果您希望 Camel 消耗路由，且您不想 Camel 自动发送回复消息，您可以使用这个选项，因为代码中的另一个组件处理回复消息。如果要使用 Camel 作为不同消息代理之间的代理，而您想要将消息从一个系统路由到另一个系统，也可以使用此选项。	false	布尔值
durableSubscriptionName (common)	用于指定持久主题订阅的持久订阅者名称。必须为 JMS 1.1 持久订阅配置 **clientId** 选项，并且可以针对 JMS 2.0 配置，以创建私有持久订阅。		字符串
jmsMessageType (common)	允许您强制使用特定的 javax.jms.Message 实现来发送 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 body 类型中使用的 JMS 消息类型。这个选项允许您指定它。 Enum 值： • Bytes • Map • 对象 • Stream • 文本		JmsMessageType
replyTo (common)	提供显式 ReplyTo 目的地（会覆盖消费者中的 Message.getJMSReplyTo （）的任何传入值）。		字符串
testConnectionOnStartup (common)	指定是否在启动时测试连接。这样可确保 Camel 启动所有 JMS 用户具有与 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 没有使用失败的连接启动。JMS 制作者也经过测试。	false	布尔值

Name	描述	默认值	类型
confirmmentModeName (consumer)	JMS 确认名称，即： SESSION_TRANSACTED、CLIENT_ACKNOWLEDGE、AUTO_ACKNOWLEDGE、DUPS_OK_ACKNOWLEDGE。 Enum 值： ● SESSION_TRANSACTED ● CLIENT_ACKNOWLEDGE ● AUTO_ACKNOWLEDGE ● DUPS_OK_ACKNOWLEDGE	AUTO_ACKNOWLEDGE	字符串
artemisConsumerPriority (consumer)	通过消费者优先级，您可以确保高优先级消费者在激活时收到消息。通常，连接到队列的活动用户以轮循方式从它接收消息。使用消费者优先级时，如果有多个活跃的消费者具有相同的高优先级，则会发送循环消息。只有高优先级消费者没有使用消息的信用时，消息才会受到较低优先级的用户，或者那些高优先级消费者已拒绝接受该消息（例如，因为它不符合与消费者关联的任何选择器的条件）。		int
asyncConsumer (consumer)	JmsConsumer 是否异步处理 Exchange。如果启用，则 JmsConsumer 可能会从 JMS 队列中获取下一个消息，而前面的消息会被异步处理（通过异步路由引擎）。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 从 JMS 队列获取下一个消息前完全处理 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该自动启动。	true	布尔值
cacheLevel (consumer)	根据 ID 为底层 JMS 资源设置缓存级别。如需了解更多详细信息，请参阅 cacheLevelName 选项。		int

Name	描述	默认值	类型
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。 Enum 值： • CACHE_AUTO • CACHE_CONNECTION • CACHE_CONSUMER • CACHE_NONE • CACHE_SESSION	CACHE_AUTO	字符串
concurrentConsumers (consumer)	指定从 JMS 消耗时的默认并发用户数（不适用于 JMS 上的请求/回复）。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。当对 JMS 进行请求/回复时，选项 replyToConcurrentConsumers 用于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（不适用于 JMS 上的请求/回复）。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。当对 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消费者数量。		int
replyToDeliveryPersistent (consumer)	指定是否默认使用持久性发送进行回复。	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化。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
subscriptionName (consumer)	设置要创建的订阅的名称。在带有共享或持久订阅的主题(pub-sub domain)时应用。如果配置了客户端 ID，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：每个订阅只允许 1 个并发消费者（这是此消息监听程序容器的默认值），除了一个共享订阅（需要 JMS 2.0）。		字符串
subscriptionShared (consumer)	设置是否共享订阅。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 来注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以作为订阅名称使用）。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)时有意义，因此此方法也会切换 pubSubDomain 标志。需要 JMS 2.0 兼容消息代理。	false	布尔值
acceptMessagesWhileStopping (consumer (advanced))	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，您可以考虑启用此选项，同时仍然在队列上排队消息。如果此选项为 false，并且您停止了 JMS 路由，则消息可能会被拒绝，而 JMS 代理必须尝试重新设计，这一次可能被拒绝，最终该消息可能会在 JMS 代理上的死信队列中移动。要避免这种情况，建议启用这个选项。	false	布尔值
allowReplyManagerQuickStop (consumer (advanced))	是否在回复管理器中使用 DefaultMessageListenerContainer 用于 request-reply 消息，都允许 DefaultMessageListenerContainer.runningAllowed 标志在启用了 JmsConfiguration#isAcceptMessagesWhileStopping 时快速停止，并且 org.apache.camel.CamelContext 当前已停止。在常规 JMS 消费者中默认启用这种快速停止功能，但为了启用此标志，您必须启用此标志。	false	布尔值

Name	描述	默认值	类型
consumerType (consumer (advanced))	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。指定 Custom 时，messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 选项将决定要使用的 org.springframework.jms.listener.AbstractMessageListenerContainer。 Enum 值： • Simple（简单） • 默认值 • Custom	默认值	ConsumerType
defaultTaskExecutorType (consumer (advanced))	指定 DefaultMessageListenerContainer 中使用哪些默认 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 - 缓存的 threadpool-like）。如果没有设置，则默认为前面的行为，它对消费者使用缓存的线程池，并将 SimpleAsync 用于回复消费者。建议使用 ThreadPool 来减少弹性配置中的线程垃圾箱，并动态增加和减少并发消费者。 Enum 值： • ThreadPool • SimpleAsync		DefaultTaskExecutorType
eagerLoadingOfProperties (consumer (advanced))	加载消息后马上启用对 JMS 属性和有效负载的 eager 加载，这通常效率低下，因为 JMS 属性可能不需要，但有时可以尽早地捕获与底层 JMS 提供程序和使用 JMS 属性相关的问题。另请参阅 eagerPoisonBody 选项。	false	布尔值

Name	描述	默认值	类型
eagerPoisonBody (consumer (advanced))	如果启用了 eagerLoadingOfProperties，并且 JMS 消息有效负载(JMS 正文或 JMS 属性)为 poison （不能读取/映射），则将此文本设置为消息正文，以便处理消息正文（导致 poison 的原因在 Exchange 上已存储为例外）。这可以通过设置 eagerPoisonBody=false 来关闭。另请参阅选项 eagerLoadingOfProperties。	因 $\{exception.message} 导致的 Poison JMS 消息	字符串
exposeListenerSession (consumer (advanced))	指定在消耗消息时是否应公开侦听器会话。	false	布尔值
replyToSameDestinationAllowed (consumer (advanced))	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发送相同消息到其自身的无端循环。	false	布尔值
taskExecutor (consumer (advanced))	允许您指定自定义任务 executor 以供使用消息。		TaskExecutor
deliveryDelay (producer)	设置用于为 JMS 发送调用的交付延迟。这个选项需要 JMS 2.0 兼容代理。	-1	long
deliveryMode (producer)	指定要使用的交付模式。可能的值有 javax.jms.DeliveryMode 定义的值。NON_PERSISTENT = 1 和 PERSISTENT = 2. Enum 值： • 1 • 2		整数
deliveryPersistent (producer)	指定是否默认使用持久性交付。	true	布尔值
explicitQosEnabled (producer)	设定在发送消息时应使用 deliveryMode、priority 或 timeToLiveity 的服务质量。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用于当前端点。这与 preserveMessageQos 选项不同，该选项以消息粒度运行，读取仅来自 Camel In 消息标头的 QoS 属性。	false	布尔值

Name	描述	默认值	类型
formatDateHeadersToIso8601 (producer)	设置 JMS 日期属性是否应根据 ISO 8601 标准进行格式化。	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
preserveMessageQos (producer)	如果设置为 true，如果要使用消息中指定的 QoS 设置发送消息，而不是 JMS 端点上的 QoS 设置。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供 all 或 only some them。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 选项将使用端点上设置的选项，而不是来自消息标头的值。	false	布尔值
priority (producer)	大于 1 的值在发送时指定消息优先级（其中 1 是最低优先级，9 为最高）。还必须启用 explicitQosEnabled 选项，以便此选项有任何效果。 Enum 值： • 1 • 2 • 3 • 4 • 5 • 6 • 7 • 8 • 9	4	int
replyToConcurrentConsumers (producer)	指定在对 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，以控制线程的动态扩展/关闭。	1	int
replyToMaxConcurrentConsumers (producer)	指定在 JMS 上使用请求/回复时的最大并发用户数。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。		int

Name	描述	默认值	类型
replyToOnTimeoutMaxcurrentConsumers (producer)	指定在通过 JMS 使用请求/回复时，进行超时时继续路由的并发消费者的最大数量。	1	int
replyToOverride (producer)	在 JMS 消息中提供显式 ReplyTo 目的地，它会覆盖 replyTo 的设置。如果您要将消息转发到远程队列，并从 ReplyTo 目的地接收回复消息，这很有用。		字符串
replyToType (producer)	在通过 JMS 进行 request/reply 时，允许显式指定用于 replyTo 队列的策略类型。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。这个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的影响的信息，以及共享回复队列的性能比其 alternatives Temporary 和 Exclusive 的性能较低。 Enum 值： ● 临时 ● 共享 ● exclusive		ReplyToType
requestTimeout (producer)	使用 InOut Exchange Pattern （毫秒）时等待回复的超时时间。默认值为 20 秒。您可以包含标头 CamelJmsRequestTimeout 来覆盖这个端点配置的超时值，因此每个消息单个超时值。另请参阅 requestTimeoutCheckerInterval 选项。	20000	long
timeToLive (producer)	在发送消息时，指定消息的生存时间（以毫秒为单位）。	-1	long
allowAdditionalHeaders (producer (advanced))	此选项用于允许其他标头，它们可能具有根据 JMS 规范无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 来执行此操作，其中包含带有字节数组或其他无效类型的值。您可以指定多个标头名称，用逗号分开，并使用 作为通配符匹配的后缀。		字符串
allowNullBody (producer (advanced))	是否允许发送不包含正文的消息。如果此选项为 false，并且消息正文为 null，则会抛出一个 JMSException。	true	布尔值

Name	描述	默认值	类型
alwaysCopyMessage (producer (advanced))	如果为 true，Camel 会在传递给生成者发送时始终生成邮件的 JMS 消息副本。在某些情况下需要复制消息，比如当设置了 replyToDestinationSelectorName 时（通常，如果设置了 replyToDestinationSelectorName），则 Camel 会将 alwaysCopyMessage 选项设置为 true。	false	布尔值
correlationProperty (producer (advanced))	使用 InOut 交换模式时，请使用此 JMS 属性而不是 JMSCorrelationID JMS 属性来关联消息。如果设置消息将只与此属性的 JMSCorrelationID 属性的值关联，则忽略且不由 Camel 设置。		字符串
disableTimeToLive (producer (advanced))	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 值作为发送消息的时间。问题是发送方和接收器系统必须同步其时钟，因此它们同步。这并非始终容易存档。因此，您可以使用 disableTimeToLive=true 来在发送的消息上将时间设置为 live 值。然后，该消息不会在接收器系统上过期。如需了解更多详细信息，请参见以下小节中关于 live 的时间。	false	布尔值
forceSendOriginalMessage (producer (advanced))	使用 mapJmsMessage=false Camel 时，如果您在路由过程中涉及标头(get 或 set)，则创建新的 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 (advanced))	仅在使用 InOnly 发送到 JMS 目的地时（例如触发和忘记）。启用此选项将增强 Camel Exchange 与实际的 JMSMessageID，在消息发送到 JMS 目的地时供 JMS 客户端使用。	false	布尔值

Name	描述	默认值	类型
replyToCacheLevelName (producer (advanced))	在通过 JMS 进行请求/回复时，根据回复消费者设置缓存级别。这个选项只适用于使用固定回复队列（而非临时）。默认情况下，Camel 将使用：CACHE_CONSUMER 用于 exclusive 或 shared w/ replyToSelectorName。用于没有 replyToSelectorName 的共享的 CACHE_SESSION。IBM WebSphere 等一些 JMS 代理可能需要设置 replyToCacheLevelName=CACHE_NONE 才能工作。注意：如果使用临时队列，则不允许 CACHE_NONE，您必须使用更高的值，如 CACHE_CONSUMER 或 CACHE_SESSION。 Enum 值： • CACHE_AUTO • CACHE_CONNECTION • CACHE_CONSUMER • CACHE_NONE • CACHE_SESSION		字符串
replyToDestinationSelectorName (producer (advanced))	使用要使用的固定名称设置 JMS Selector，以便在使用共享队列时过滤您自己的回复（即，如果您不使用临时回复队列）。		字符串
streamMessageTypeEnabled (producer (advanced))	设置 StreamMessage 类型是否已启用。通过以 BytesMessage 或 StreamMessage 发送的消息有效负载，如 files、InputStream 等。这个选项控制将使用的 kind。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。通过启用这个选项，消息有效负载以块的形式读取到内存中，每个块都会被写入 StreamMessage，直到没有更多数据。	false	布尔值
allowAutoWiredConnectionFactory (advanced)	如果没有配置连接工厂，是否从 registry 中自动发现 ConnectionFactory。如果只找到了一个 ConnectionFactory 实例，则会使用它。这默认是启用的。	true	布尔值
allowAutoWiredDestinationResolver (advanced)	如果没有配置目标解析器，是否从 registry 中自动发现 DestinationResolver。如果只找到一个 DestinationResolver 实例，则会使用它。这默认是启用的。	true	布尔值
allowSerializedHeaders (advanced)	控制是否包含序列化标头。仅在 transferExchange 为 true 时应用。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值

Name	描述	默认值	类型
artemisStreamingEnabled (advanced)	是否针对 Apache Artemis 流模式进行优化。当将 Artemis 与 JMS StreamMessage 类型一起使用时，这可以减少内存开销。只有在使用 Apache Artemis 时，才必须启用此选项。	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 无法获得与远程 JMS 代理的连接，那么在重试和/或故障转移时可能会阻止它。这会导致 Camel 在启动路由时阻止。通过将这个选项设置为 true，您可以让路由启动，而 JmsConsumer 使用异步模式的专用线程连接到 JMS 代理。如果使用此选项，请注意，如果无法建立连接，则会在 WARN 级别记录异常，消费者将无法接收消息；然后，您可以重启要重试的路由。	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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
configuration (advanced)	使用共享的 JMS 配置。		JmsConfiguration
destinationResolver (advanced)	可插拔 org.springframework.jms.support.destination.DestinationResolver，允许您使用自己的解析器（例如，在 JNDI 注册表中查找实际目的地）。		DestinationResolver
errorHandler (advanced)	指定在处理消息时抛出异常时调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置 errorHandler，则会在 WARN 级别中记录这些例外。您可以配置日志记录级别，以及堆栈跟踪是否应该使用 errorHandlerLoggingLevel 和 errorHandlerLogStackTrace 选项记录。这样可以更容易配置，而不必对自定义 errorHandler 进行编码。		ErrorHandler
exceptionListener (advanced)	指定要收到任何底层 JMS 异常的 JMS Exception Listener。		ExceptionListener
idleConsumerLimit (advanced)	指定允许在任何给定时间闲置的用户数量的限制。	1	int

Name	描述	默认值	类型
idleTaskExecutionLimit (advanced)	指定接收任务的闲置执行的限制，而不是在其执行中收到任何消息。如果达到这个限制，则任务将关闭并离开接收其他执行任务（在动态调度的情况下；请参阅 maxConcurrentConsumers 设置）。Spring 中还有额外的文档。	1	int
includeAllJMSXProperties (advanced)	在从 JMS 到 Camel 消息映射时，是否包含所有 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，则这个选项不适用。	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。默认策略将安全地 marshal 句点和连字符(. 和 -)。passthrough 策略将密钥保留为原样。可用于 JMS 代理，这些代理不关心 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。 Enum 值： • default • passthrough		JmsKeyFormatStrategy
mapJmsMessage (advanced)	指定 Camel 是否将收到的 JMS 消息自动映射到适合的有效负载类型，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（例如 min max），则此选项可用于设置 eg 100 来控制在需要较少工作时消费者缩小的速度。	-1	int
messageConverter (advanced)	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)	使用在 Camel 发送 JMS 消息时调用的 given MessageCreatedStrategy，在 Camel 创建 javax.jms.Message 对象的新实例时调用。		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 供应商接受这个 hint，则这些消息必须将消息 ID 设置为 null；如果供应商忽略 hint，则消息 ID 必须设置为其正常唯一值。	true	布尔值

Name	描述	默认值	类型
messageListenerContainerFactory (advanced)	用于决定使用消息的 org.springframework.jms.listener.AbstractMessageListenerContainer 的 MessageListenerContainerFactory 的 registry ID。设置此选项会自动将 consumerType 设置为 Custom。		MessageListenerContainerFactory
messageTimestampEnabled (advanced)	指定默认情况下，是否应在发送消息时启用时间戳。这只是 JMS 代理的提示。如果 JMS 供应商接受这个 hint，则这些消息必须将时间戳设置为零；如果供应商忽略 hint，则必须将时间戳设置为其正常值。	true	布尔值
pubSubNoLocal (advanced)	指定是否禁止发送其自身连接发布的消息。	false	布尔值
queueBrowseStrategy (advanced)	在浏览队列时使用自定义 QueueBrowseStrategy。		QueueBrowseStrategy
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，即刷新连接时（以毫秒为单位）。默认值为 5000 ms，即 5 秒。	5000	long
requestTimeoutCheckerInterval (advanced)	配置 Camel 在通过 JMS 进行请求/回复时应检查超时交换的频率。默认情况下，Camel 会每秒检查一次。但是，如果您在超时时必须更快地响应，您可以降低这个间隔，以便更频繁地检查。超时由选项 requestTimeout 决定。	1000	long
同步 (advanced)	设置是否应严格使用同步处理。	false	布尔值
transferException (advanced)	如果启用了，并且您在使用 Request Reply messaging (InOut)，且 Exchange 在消费者端失败，则原因例外将作为 javax.jms.ObjectMessage 发回。如果客户端是 Camel，则返回的例外将被重新箭头。这样，您可以在路由中使用 Camel JMS 作为网桥 - 例如，使用持久性队列启用可靠的路由。请注意，如果您也启用了 transferExchange，这个选项将具有优先权。需要 Caught 异常才能按顺序排序。消费者端的原始 Exception 可以嵌套在外部异常中，如返回到制作者时 org.apache.camel.RuntimeCamelException。请小心谨慎，因为数据使用 Java 对象序列化，并且要求收到的能够在类级别上反序列化数据，这会强行生产者和消费者之间的强耦合。	false	布尔值

Name	描述	默认值	类型
transferExchange (advanced)	您可以通过线路传输交换，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 headers, exchange properties, exchange exception。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，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 知道有效负载是一个交换而不是常规有效负载。请小心谨慎，因为数据使用 Java 对象序列化，并且要求接收器能够在类级别上反序列化数据，这会强制生产者与需要使用兼容 Camel 版本的消费者之间强耦耦合！	false	布尔值
useMessageIDAsCorrelationID (advanced)	指定 JMSMessageID 是否始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值
waitForProvisionCorrelationToBeUpdatedCounter (advanced)	在通过 JMS 进行请求/回复时，等待 provisional correlation id 更新为实际关联 ID 的次数，以及启用选项 useMessageIDAsCorrelationID 的时间。	50	int
waitForProvisionCorrelationToBeUpdatedThreadSleepingTime (advanced)	等待更新调配关联 id 期间每次处于睡眠状态的间隔。	100	long
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头到 Camel 消息。		HeaderFilterStrategy
errorHandlerLoggingLevel (logging)	允许为日志记录无法捕获的异常配置默认 errorHandler 日志记录级别。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	WARN	LoggingLevel

Name	描述	默认值	类型
errorHandlerLogStackTrace (logging)	允许默认 errorHandler 控制是否应记录 stacktrace。	true	布尔值
password (security)	用于 ConnectionFactory 的密码。您也可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
用户名 （安全性）	用于 ConnectionFactory 的用户名。您也可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
转换 （事务）	指定是否使用 transacted 模式。	false	布尔值
transactedInOut (transaction)	指定 InOut 操作（请求回复）是否默认使用 transacted 模式（如果此标志设为 true），则 Spring JmsTemplate 会将 sessionTransacted 设置为 true，以及 acknowledgeMode 作为 InOut 操作的 JmsTemplate 的 transacted。注意从 Spring JMS：在 JTA 事务中，传递给 createQueue 的参数，不会考虑 createTopic 方法。根据 Java EE 事务上下文，容器对这些值自行做出决定。类似地，这些参数不会被在本地管理的事务中考虑，因为本例中的 Spring JMS 在现有的 JMS Session 上运行。在受管事务之外运行时，将此标志设置为 true 将使用简短的本地 JMS 事务，并在存在受管事务（并非 XA 事务）的情况下同步的本地 JMS 事务。这将与主事务一起管理本地 JMS 事务（可能是原生 JDBC 事务），而 JMS 事务会在主事务后提交右边。	false	布尔值
lazyCreateTransactionManager (transaction (advanced))	如果为 true，则 Camel 将创建一个 JmsTransactionManager，如果没有在选项 transacted=true 时注入任何 transactionManager。	true	布尔值
transactionManager (transaction (advanced))	要使用的 Spring 事务管理器。		PlatformTransactionManager
transactionName (transaction (advanced))	要使用的事务的名称。		字符串
transactionTimeout (transaction (advanced))	使用 transacted 模式，事务的超时值（以秒为单位）。	-1	int

51.5. 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：

51.5.1. 路径参数(2 参数)

Name	描述	默认值	类型
destinationType (common)	要使用的目标类型。 Enum 值： • queue • topic • temp-queue • temp-topic	queue	字符串
destinationName (common)	用作目标的队列或主题 **必需** 名称。		字符串

51.5.2. 查询参数(95 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，且只能被单个 JMS 连接实例使用。它通常只需要使用 JMS 1.1 的持久主题订阅。如果使用 Apache ActiveMQ，您可能更喜欢使用 Virtual Topics。		字符串
ConnectionFactory (common)	要使用的连接工厂。连接工厂必须在组件或端点上配置。		ConnectionFactory
disableReplyTo (common)	指定 Camel 是否忽略消息中的 JMSReplyTo 标头。如果为 true，Camel 不会向 JMSReplyTo 标头中指定的目的地发送回复。如果您希望 Camel 消耗路由，且您不想 Camel 自动发送回复消息，您可以使用这个选项，因为代码中的另一个组件处理回复消息。如果要使用 Camel 作为不同消息代理之间的代理，而您想要将消息从一个系统路由到另一个系统，也可以使用此选项。	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订阅者名称。也必须配置 clientId 选项。		字符串
jmsMessageType (common)	允许您强制使用特定的 javax.jms.Message 实现来发送 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 body 类型中使用的 JMS 消息类型。这个选项允许您指定它。 Enum 值： ● Bytes ● Map ● 对象 ● Stream ● 文本		JmsMessageType
replyTo (common)	提供显式 ReplyTo 目的地（会覆盖消费者中的 Message.getJMSReplyTo （）的任何传入值）。		字符串
testConnectionOnStartup (common)	指定是否在启动时测试连接。这样可确保 Camel 启动所有 JMS 用户具有与 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 没有使用失败的连接启动。JMS 制作者也经过测试。	false	布尔值
confirmmentModeName (consumer)	JMS 确认名称，即： SESSION_TRANSACTED、CLIENT_ACKNOWLEDGE、AUTO_ACKNOWLEDGE、DUPS_OK_ACKNOWLEDGE。 Enum 值： ● SESSION_TRANSACTED ● CLIENT_ACKNOWLEDGE ● AUTO_ACKNOWLEDGE ● DUPS_OK_ACKNOWLEDGE	AUTO_ACKNOWLEDGE	字符串

Name	描述	默认值	类型
artemisConsumerPriority (consumer)	通过消费者优先级，您可以确保高优先级消费者在激活时收到消息。通常，连接到队列的活动用户以轮循方式从它接收消息。使用消费者优先级时，如果有多个活跃的消费者具有相同的高优先级，则会发送循环消息。只有高优先级消费者没有使用消息的信用时，消息才会受到较低优先级的用户，或者那些高优先级消费者已拒绝接受该消息（例如，因为它不符合与消费者关联的任何选择器的条件）。		int
asyncConsumer (consumer)	JmsConsumer 是否异步处理 Exchange。如果启用，则 JmsConsumer 可能会从 JMS 队列中获取下一个消息，而前面的消息会被异步处理（通过异步路由引擎）。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 从 JMS 队列获取下一个消息前完全处理 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该自动启动。	true	布尔值
cacheLevel (consumer)	根据 ID 为底层 JMS 资源设置缓存级别。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。 Enum 值： • CACHE_AUTO • CACHE_CONNECTION • CACHE_CONSUMER • CACHE_NONE • CACHE_SESSION	CACHE_AUTO	字符串
concurrentConsumers (consumer)	指定从 JMS 消耗时的默认并发用户数（不适用于 JMS 上的请求/回复）。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。当对 JMS 进行请求/回复时，选项 replyToConcurrentConsumers 用于控制回复消息监听器上的并发用户数量。	1	int

Name	描述	默认值	类型
maxConcurrentConsumers (consumer)	指定从 JMS 消耗时的最大并发消费者数（不适用于 JMS 上的请求/回复）。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。当对 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消费者数量。		int
replyToDeliveryPersistent (consumer)	指定是否默认使用持久性发送进行回复。	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化。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。在带有共享或持久订阅的主题(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：每个订阅只允许 1 个并发消费者（这是此消息监听程序容器的默认值），除了一个共享订阅（需要 JMS 2.0）。		字符串
subscriptionShared (consumer)	设置是否共享订阅。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 来注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以作为订阅名称使用）。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)时有意义，因此此方法也会切换 pubSubDomain 标志。需要 JMS 2.0 兼容消息代理。	false	布尔值
acceptMessagesWhileStopping (consumer (advanced))	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，您可以考虑启用此选项，同时仍然在队列上排队消息。如果此选项为 false，并且您停止了 JMS 路由，则消息可能会被拒绝，而 JMS 代理必须尝试重新设计，这一次可能被拒绝，最终该消息可能会在 JMS 代理上的死信队列中移动。要避免这种情况，建议启用这个选项。	false	布尔值

Name	描述	默认值	类型
allowReplyManagerQuickStop (consumer (advanced))	是否在回复管理器中使用 DefaultMessageListenerContainer 用于 request-reply 消息，都允许 DefaultMessageListenerContainer.runningAllowed 标志在启用了 JmsConfiguration#isAcceptMessagesWhileStopping 时快速停止，并且 org.apache.camel.CamelContext 当前已停止。在常规 JMS 消费者中默认启用这种快速停止功能，但为了启用此标志，您必须启用此标志。	false	布尔值
consumerType (consumer (advanced))	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。指定 Custom 时，messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 选项将决定要使用的 org.springframework.jms.listener.AbstractMessageListenerContainer。 Enum 值： • Simple（简单） • 默认值 • Custom	默认值	ConsumerType
defaultTaskExecutorType (consumer (advanced))	指定 DefaultMessageListenerContainer 中使用哪些默认 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 - 缓存的 threadpool-like）。如果没有设置，则默认为前面的行为，它对消费者使用缓存的线程池，并将 SimpleAsync 用于回复消费者。建议使用 ThreadPool 来减少弹性配置中的线程垃圾箱，并动态增加和减少并发消费者。 Enum 值： • ThreadPool • SimpleAsync		DefaultTaskExecutorType

Name	描述	默认值	类型
eagerLoadingOfProperties (consumer (advanced))	加载消息后马上启用对 JMS 属性和有效负载的 eager 加载，这通常效率低下，因为 JMS 属性可能不需要，但有时可以尽早地捕获与底层 JMS 提供程序和使用 JMS 属性相关的问题。另请参阅 eagerPoisonBody 选项。	false	布尔值
eagerPoisonBody (consumer (advanced))	如果启用了 eagerLoadingOfProperties，并且 JMS 消息有效负载(JMS 正文或 JMS 属性)为 poison （不能读取/映射），则将此文本设置为消息正文，以便处理消息正文（导致 poison 的原因在 Exchange 上已存储为例外）。这可以通过设置 eagerPoisonBody=false 来关闭。另请参阅选项 eagerLoadingOfProperties。	因 $\{exception.message} 导致的 Poison JMS 消息	字符串
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
exposeListenerSession (consumer (advanced))	指定在消耗消息时是否应公开侦听器会话。	false	布尔值
replyToSameDestinationAllowed (consumer (advanced))	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发送相同消息到其自身的无端循环。	false	布尔值
taskExecutor (consumer (advanced))	允许您指定自定义任务 executor 以供使用消息。		TaskExecutor
deliveryDelay (producer)	设置用于为 JMS 发送调用的交付延迟。这个选项需要 JMS 2.0 兼容代理。	-1	long

Name	描述	默认值	类型
deliveryMode (producer)	指定要使用的交付模式。可能的值有 javax.jms.DeliveryMode 定义的值。NON_PERSISTENT = 1 和 PERSISTENT = 2. Enum 值： • 1 • 2		整数
deliveryPersistent (producer)	指定是否默认使用持久性交付。	true	布尔值
explicitQosEnabled (producer)	设定在发送消息时应使用 deliveryMode、priority 或 timeToLiveity 的服务质量。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用于当前端点。这与 preserveMessageQos 选项不同，该选项以消息粒度运行，读取仅来自 Camel In 消息标头的 QoS 属性。	false	布尔值
formatDateHeadersToIso8601 (producer)	设置 JMS 日期属性是否应根据 ISO 8601 标准进行格式化。	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
preserveMessageQos (producer)	如果设置为 true，如果要使用消息中指定的 QoS 设置发送消息，而不是 JMS 端点上的 QoS 设置。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供 all 或 only some them。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 选项将使用端点上设置的选项，而不是来自消息标头的值。	false	布尔值

Name	描述	默认值	类型
priority (producer)	大于 1 的值在发送时指定消息优先级（其中 1 是最低优先级，9 为最高）。还必须启用 explicitQosEnabled 选项，以便此选项有任何效果。 Enum 值： • 1 • 2 • 3 • 4 • 5 • 6 • 7 • 8 • 9	4	int
replyToConcurrentConsumers (producer)	指定在对 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，以控制线程的动态扩展/关闭。	1	int
replyToMaxConcurrentConsumers (producer)	指定在 JMS 上使用请求/回复时的最大并发用户数。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。		int
replyToOnTimeoutMaxcurrentConsumers (producer)	指定在通过 JMS 使用请求/回复时，进行超时时继续路由的并发消费者的最大数量。	1	int
replyToOverride (producer)	在 JMS 消息中提供显式 ReplyTo 目的地，它会覆盖 replyTo 的设置。如果您要将消息转发到远程队列，并从 ReplyTo 目的地接收回复消息，这很有用。		字符串

Name	描述	默认值	类型
replyToType (producer)	在通过 JMS 进行 request/reply 时，允许显式指定用于 replyTo 队列的策略类型。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。这个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的影响的信息，以及共享回复队列的性能比其 alternatives Temporary 和 Exclusive 的性能较低。 Enum 值： • 临时 • 共享 • exclusive		ReplyToType
requestTimeout (producer)	使用 InOut Exchange Pattern （毫秒）时等待回复的超时时间。默认值为 20 秒。您可以包含标头 CamelJmsRequestTimeout 来覆盖这个端点配置的超时值，因此每个消息单个超时值。另请参阅 requestTimeoutCheckerInterval 选项。	20000	long
timeToLive (producer)	在发送消息时，指定消息的生存时间（以毫秒为单位）。	-1	long
allowAdditionalHeaders (producer (advanced))	此选项用于允许其他标头，它们可能具有根据 JMS 规范无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 来执行此操作，其中包含带有字节数组或其他无效类型的值。您可以指定多个标头名称，用逗号分开，并使用 作为通配符匹配的后缀。		字符串
allowNullBody (producer (advanced))	是否允许发送不包含正文的消息。如果此选项为 false，并且消息正文为 null，则会抛出一个 JMSException。	true	布尔值
alwaysCopyMessage (producer (advanced))	如果为 true，Camel 会在传递给生成者发送时始终生成邮件的 JMS 消息副本。在某些情况下需要复制消息，比如当设置了 replyToDestinationSelectorName 时（通常，如果设置了 replyToDestinationSelectorName），则 Camel 会将 alwaysCopyMessage 选项设置为 true。	false	布尔值
correlationProperty (producer (advanced))	使用 InOut 交换模式时，请使用此 JMS 属性而不是 JMSCorrelationID JMS 属性来关联消息。如果设置消息将只与此属性的 JMSCorrelationID 属性的值关联，则忽略且不由 Camel 设置。		字符串

Name	描述	默认值	类型
disableTimeToLive (producer (advanced))	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 值作为发送消息的时间。问题是发送方和接收器系统必须同步其时钟，因此它们同步。这并非始终容易存档。因此，您可以使用 disableTimeToLive=true 来在发送的消息上将时间设置为 live 值。然后，该消息不会在接收器系统上过期。如需了解更多详细信息，请参见以下小节中关于 live 的时间。	false	布尔值
forceSendOriginalMessage (producer (advanced))	使用 mapJmsMessage=false Camel 时，如果您在路由过程中涉及标头(get 或 set)，则创建新的 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 (advanced))	仅在使用 InOnly 发送到 JMS 目的地时（例如触发和忘记）。启用此选项将增强 Camel Exchange 与实际的 JMSMessageID，在消息发送到 JMS 目的地时供 JMS 客户端使用。	false	布尔值
replyToCacheLevelName (producer (advanced))	在通过 JMS 进行请求/回复时，根据回复消费者设置缓存级别。这个选项只适用于使用固定回复队列（而非临时）。默认情况下，Camel 将使用：CACHE_CONSUMER 用于 exclusive 或 shared w/ replyToSelectorName。用于没有 replyToSelectorName 的共享的 CACHE_SESSION。IBM WebSphere 等一些 JMS 代理可能需要设置 replyToCacheLevelName=CACHE_NONE 才能工作。注意：如果使用临时队列，则不允许 CACHE_NONE，您必须使用更高的值，如 CACHE_CONSUMER 或 CACHE_SESSION。 Enum 值： • CACHE_AUTO • CACHE_CONNECTION • CACHE_CONSUMER • CACHE_NONE • CACHE_SESSION		字符串
replyToDestinationSelectorName (producer (advanced))	使用要使用的固定名称设置 JMS Selector，以便在使用共享队列时过滤您自己的回复（即，如果您不使用临时回复队列）。		字符串

Name	描述	默认值	类型
streamMessageTypeEnabled (producer (advanced))	设置 StreamMessage 类型是否已启用。通过以 BytesMessage 或 StreamMessage 发送的消息有效负载，如 files、InputStream 等。这个选项控制将使用的 kind。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。通过启用这个选项，消息有效负载以块的形式读取到内存中，每个块都会被写入 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 时应用。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值
artemisStreamingEnabled (advanced)	是否针对 Apache Artemis 流模式进行优化。当将 Artemis 与 JMS StreamMessage 类型一起使用时，这可以减少内存开销。只有在使用 Apache Artemis 时，才必须启用此选项。	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 无法获得与远程 JMS 代理的连接，那么在重试和/或故障转移时可能会阻止它。这会导致 Camel 在启动路由时阻止。通过将这个选项设置为 true，您可以让路由启动，而 JmsConsumer 使用异步模式的专用线程连接到 JMS 代理。如果使用此选项，请注意，如果无法建立连接，则会在 WARN 级别记录异常，消费者将无法接收消息；然后，您可以重启要重试的路由。	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)	指定在处理消息时抛出异常时调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置 errorHandler，则会在 WARN 级别中记录这些例外。您可以配置日志记录级别，以及堆栈跟踪是否应该使用 errorHandlerLoggingLevel 和 errorHandlerLogStackTrace 选项记录。这样可以更容易配置，而不必对自定义 errorHandler 进行编码。		ErrorHandler
exceptionListener (advanced)	指定要收到任何底层 JMS 异常的 JMS Exception Listener。		ExceptionListener
headerFilterStrategy (advanced)	使用自定义 HeaderFilterStrategy 过滤标头到 Camel 消息。		HeaderFilterStrategy

Name	描述	默认值	类型
idleConsumerLimit (advanced)	指定允许在任何给定时间闲置的用户数量的限制。	1	int
idleTaskExecutionLimit (advanced)	指定接收任务的闲置执行的限制，而不是在其执行中收到任何消息。如果达到这个限制，则任务将关闭并离开接收其他执行任务（在动态调度的情况下；请参阅 maxConcurrentConsumers 设置）。Spring 中还有额外的文档。	1	int
includeAllJMSXProperties (advanced)	在从 JMS 到 Camel 消息映射时，是否包含所有 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，则这个选项不适用。	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。默认策略将安全地 marshal 句点和连字符(. 和 -)。passthrough 策略将密钥保留为原样。可用于 JMS 代理，这些代理不关心 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。 Enum 值： • default • passthrough		JmsKeyFormatStrategy
mapJmsMessage (advanced)	指定 Camel 是否将收到的 JMS 消息自动映射到适合的有效负载类型，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（例如 min max），则此选项可用于设置 eg 100 来控制在需要较少工作时消费者缩小的速度。	-1	int
messageConverter (advanced)	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)	使用在 Camel 发送 JMS 消息时调用的 given MessageCreatedStrategy，在 Camel 创建 javax.jms.Message 对象的新实例时调用。		MessageCreatedStrategy

Name	描述	默认值	类型
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 供应商接受这个 hint，则这些消息必须将消息 ID 设置为 null；如果供应商忽略 hint，则消息 ID 必须设置为其正常唯一值。	true	布尔值
messageListenerContainerFactory (advanced)	用于决定使用消息的 org.springframework.jms.listener.AbstractMessageListenerContainer 的 MessageListenerContainerFactory 的 registry ID。设置此选项会自动将 consumerType 设置为 Custom。		MessageListenerContainerFactory
messageTimestampEnabled (advanced)	指定默认情况下，是否应在发送消息时启用时间戳。这只是 JMS 代理的提示。如果 JMS 供应商接受这个 hint，则这些消息必须将时间戳设置为零；如果供应商忽略 hint，则必须将时间戳设置为其正常值。	true	布尔值
pubSubNoLocal (advanced)	指定是否禁止发送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，即刷新连接时（以毫秒为单位）。默认值为 5000 ms，即 5 秒。	5000	long
requestTimeoutCheckerInterval (advanced)	配置 Camel 在通过 JMS 进行请求/回复时应检查超时交换的频率。默认情况下，Camel 会每秒检查一次。但是，如果您在超时时必须更快地响应，您可以降低这个间隔，以便更频繁地检查。超时由选项 requestTimeout 决定。	1000	long
同步 (advanced)	设置是否应严格使用同步处理。	false	布尔值
transferException (advanced)	如果启用了，并且您在使用 Request Reply messaging (InOut)，且 Exchange 在消费者端失败，则原因例外将作为 javax.jms.ObjectMessage 发回。如果客户端是 Camel，则返回的例外将被重新箭头。这样，您可以在路由中使用 Camel JMS 作为网桥 - 例如，使用持久性队列启用可靠的路由。请注意，如果您也启用了 transferExchange，这个选项将具有优先权。需要 Caught 异常才能按顺序排序。消费者端的原始 Exception 可以嵌套在外部异常中，如返回到制作者时 org.apache.camel.RuntimeCamelException。请小心谨慎，因为数据使用 Java 对象序列化，并且要求收到的能够在类级别上反序列化数据，这会强行生产者和消费者之间的强耦合。	false	布尔值

Name	描述	默认值	类型
transferExchange (advanced)	您可以通过线路传输交换，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 headers, exchange properties, exchange exception。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，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 知道有效负载是一个交换而不是常规有效负载。请小心谨慎，因为数据使用 Java 对象序列化，并且要求接收器能够在类级别上反序列化数据，这会强制生产者与需要使用兼容 Camel 版本的消费者之间强耦耦合！	false	布尔值
useMessageIDAsCorrelationID (advanced)	指定 JMSMessageID 是否始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值
waitForProvisionCorrelationToBeUpdatedCounter (advanced)	在通过 JMS 进行请求/回复时，等待 provisional correlation id 更新为实际关联 ID 的次数，以及启用选项 useMessageIDAsCorrelationID 的时间。	50	int
waitForProvisionCorrelationToBeUpdatedThreadSleepingTime (advanced)	等待更新调配关联 id 期间每次处于睡眠状态的间隔。	100	long
errorHandlerLoggingLevel (logging)	允许为日志记录无法捕获的异常配置默认 errorHandler 日志记录级别。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	WARN	LoggingLevel
errorHandlerLogStackTrace (logging)	允许默认 errorHandler 控制是否应记录 stacktrace。	true	布尔值
password (security)	用于 ConnectionFactory 的密码。您也可以直接在 ConnectionFactory 上配置用户名/密码。		字符串

Name	描述	默认值	类型
用户名 （安全性）	用于 ConnectionFactory 的用户名。您也可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
转换 （事务）	指定是否使用 transacted 模式。	false	布尔值
transactedInOut (transaction)	指定 InOut 操作（请求回复）是否默认使用 transacted 模式（如果此标志设为 true），则 Spring JmsTemplate 会将 sessionTransacted 设置为 true，以及 acknowledgeMode 作为 InOut 操作的 JmsTemplate 的 transacted。注意从 Spring JMS：在 JTA 事务中，传递给 createQueue 的参数，不会考虑 createTopic 方法。根据 Java EE 事务上下文，容器对这些值自行做出决定。类似地，这些参数不会被在本地管理的事务中考虑，因为本例中的 Spring JMS 在现有的 JMS Session 上运行。在受管事务之外运行时，将此标志设置为 true 将使用简短的本地 JMS 事务，并在存在受管事务（并非 XA 事务）的情况下同步的本地 JMS 事务。这将与主事务一起管理本地 JMS 事务（可能是原生 JDBC 事务），而 JMS 事务会在主事务后提交右边。	false	布尔值
lazyCreateTransactionManager (transaction (advanced))	如果为 true，则 Camel 将创建一个 JmsTransactionManager，如果没有在选项 transacted=true 时注入任何 transactionManager。	true	布尔值
transactionManager (transaction (advanced))	要使用的 Spring 事务管理器。		PlatformTransactionManager
transactionName (transaction (advanced))	要使用的事务的名称。		字符串
transactionTimeout (transaction (advanced))	使用 transacted 模式，事务的超时值（以秒为单位）。	-1	int

51.6. SAMPLES

JMS 也用于其他组件的许多示例。但我们提供了几个示例以开始。

51.6.1. 从 JMS 接收

在以下示例中，我们配置一个路由，接收 JMS 消息并将消息路由到 POJO：

```
from("jms:queue:foo").
  to("bean:myBusinessLogic");
```

您可以使用任何 EIP 模式，以便路由可以基于上下文。例如，如何为大型手过滤一个订购主题：

```
from("jms:topic:OrdersTopic").
  filter().method("myBean", "isGoldCustomer").
  to("jms:queue:BigSpendersQueue");
```

51.6.2. 发送到 JMS

在以下示例中，我们轮询一个文件文件夹，并将文件内容发送到 JMS 主题。当我们希望文件内容为 TextMessage 而不是 BytesMessage 时，我们需要将正文转换为 String ：

```
from("file://orders").
  convertBodyTo(String.class).
  to("jms:topic:OrdersTopic");
```

51.6.3. 使用注解

Camel 还具有注解，因此您可以使用 POJO Consuming 和 POJO Producing。

51.6.4. Spring DSL 示例

前面的示例使用 Java DSL。Camel 还支持 Spring XML DSL。以下是使用 Spring DSL 的大成本示例：

```
<route>
  <from uri="jms:topic:OrdersTopic"/>
  <filter>
    <method ref="myBean" method="isGoldCustomer"/>
    <to uri="jms:queue:BigSpendersQueue"/>
  </filter>
</route>
```

51.6.5. 其他示例

JMS 会出现在其他组件和 EIP 模式的许多示例中，以及此 Camel 文档。因此可以自由浏览文档。

51.6.6. 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，您可以使用 JMS 死信队列上的 transferExchange 选项指示 Camel 将整个交换存储在队列中，以存放 org.apache.camel.support.DefaultExchangeHolder 的 javax.jms.ObjectMessage。这可让您从死信队列使用，并通过密钥 Exchange.EXCEPTION_CAUGHT 从 Exchange 属性中检索导致异常。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，您可以使用 JMS 队列并分析问题：

```
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();
```

51.6.7. 使用 JMS 作为死信频道仅存储错误

您可以使用 JMS 存储原因错误消息或存储自定义正文，您可以自行初始化。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，我们仅在转换中存储原始原因信息。但是，您可以使用任何表达式发送您喜欢的任何表达式。例如，您可以在 Bean 上调用方法或使用自定义处理器。

51.7. JMS 和 CAMEL 之间的消息映射

Camel 会在 javax.jms.Message 和 org.apache.camel.Message 之间自动映射消息。

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**。
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[]**
javax.jms.MapMessage	**Map<String, Object>**
javax.jms.ObjectMessage	**对象**

51.7.1. 禁用 JMS 消息的自动映射

您可以使用 mapJmsMessage 选项禁用上述自动映射。如果禁用，Camel 不会尝试映射收到的 JMS 消息，而是直接使用它作为有效负载。这可让您避免映射开销，并让 Camel 只需通过 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。

51.7.2. 使用自定义 MessageConverter

您可以使用 messageConverter 选项在 Spring org.springframework.jms.support.converter.MessageConverter 类中执行自己的映射。

例如，在下面的路由中，我们在发送消息到 JMS 顺序队列时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。

51.7.3. 控制所选的映射策略

您可以使用端点 URL 上的 jmsMessageType 选项为所有消息强制使用特定的消息类型。

在以下路由中，我们从文件夹轮询文件并将其作为 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 设置标头来指定每个消息使用的消息类型。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。

51.8. 发送时的消息格式

通过 JMS 线发送的交换必须符合 JMS 消息规格。

对于 exchange.in.header，以下规则适用于标头 键 ：

- 以 JMS 或 JMSX 开头的键被保留。
- exchange.in.headers 键必须是 literals，且所有为有效的 Java 标识符（请勿在键名称中使用点）。
-

当使用 JMS 消息时，Camel 替换了点 &连字符和反向替换，当 Camel 使用消息时，Camel 会替换 'DOT' 和反向替换。
- 在 Camel 使用消息时替换为 'HYPHEN' 和反向替换。

- 另请参阅 jmsKeyFormatStrategy 选项，它允许使用您自己的自定义策略进行格式化密钥。

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 整数、长、Character）。Type, String,CharSequence,Date,BigDecimal 和 BigInteger 都转换为其 toString （） 表示。所有其他类型都会被丢弃。

如果 Camel 丢弃给定的标头值，Camel 将以类别 org.apache.camel.component.jms.JmsBinding 记录到 DEBUG 级别。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}
```

51.9. 接收时的消息格式

Camel 在 Exchange 收到消息时添加以下属性：

属性	类型	描述
org.apache.camel.jms.replyDestination	**javax.jms.Destination**	回复目的地。

Camel 在接收 JMS 消息时将以下 JMS 属性添加到 In 消息标头中：

标头	类型	描述
JMSCorrelationID	**字符串**	JMS 关联 ID。
JMSDeliveryMode	**int**	JMS 交付模式。

标头	类型	描述
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（具有 0 作为最低优先级，最高为 9）。
JMSRedelivered	**布尔值**	是 JMS 消息 redelivered。
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。

由于上述所有信息都是标准 JMS，您可以检查 JMS 文档以了解 更多详细信息。

51.10. 关于使用 CAMEL 来发送和接收消息和 JMSREPLYTO

JMS 组件比较复杂，您必须密切注意它在某种情况下的工作方式。因此，这是要查找的一些区域/区域的简短摘要。

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，***
- ***是否在端点或消息标头中设置 JMSReplyTo。***
- ***是否在 JMS 端点上设置了任何以下选项：disableReplyTo,preserveMessageQos,explicitQosEnabled。***

所有这些都很复杂，需要理解并配置 以支持您的用例。

51.10.1. JmsProducer

根据配置， Jms Producer 的行为如下：

Exchange Pattern	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	设置了 **JMSReplyTo**	Camel 将预期回复，在发送消息后，它将开始侦听指定的 **JMSReplyTo** 队列上的回复消息。
InOnly	-	Camel 将发送消息，**而不是** 预期回复。
InOnly	设置了 **JMSReplyTo**	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 在 **WARN** 级别的日志中记录此日志（以后从 **Camel 2.6** 更改到 **DEBUG** 级别）。您可以使用 **preserveMessageQuo=true** 来指示 Camel 保留 **JMSReplyTo**。在所有情况下，J **msProducer 不会** 期望任何回复，因此在发送消息后继续。

51.10.2. JmsConsumer

JmsConsumer 的行为如下，具体取决于配置：

Exchange Pattern	其他选项	描述
InOut	-	Camel 将回复发回到 **JMSReplyTo** 队列。
InOnly	-	Camel 不会发送回复，因为模式是 *InOnly*。
-	**disableReplyTo =true**	这个选项阻止回复。

*请**注意在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，这非常有用：

```
from("activemq:queue:in")
  .to("bean:validateOrder")
  .to(ExchangePattern.InOnly, "activemq:topic:order")
  .to("bean:handleOrder");
```

51.11. 重复使用端点并发送到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，可以重复使用 JMS 端点并在消息标头中指定实际目的地。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中创建的端点数量。

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**	目的地名称。

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 只是一个占位符。它必须作为 JMS 端点 URL 的一部分提供，但本例中将忽略它。

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，并将其用作端点中配置的目的地。因此，在这个示例中，Camel 将消息发送到 activemq:queue:order:2，假设 id 值为 2。

如果设置了 CamelJmsDestination 和 CamelJmsDestinationName 标头，则 CamelJmsDestination 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）。

51.12. 配置不同的 JMS 供应商

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 activemq 组件。您可以执行相同的操作来配置 MQ 系列、TibCo、BEA、SEA 等等。

命名 JMS 组件后，您可以使用 URI 引用该组件内的端点。例如，对于组件名称 activemq，您可以使用 URI 格式引用目的地，activemq:[queue:|topic:]destinationName。您可以将相同的方法用于所有其他 JMS 提供程序。

这适用于 SpringCamelContext lazily 从用于 Endpoint URI 的方案名称的 spring 上下文获取组件，并且组件解析端点 URI。

51.12.1. 使用 JNDI 查找 ConnectionFactory

如果使用 J2EE 容器，您可能需要查找 JNDI 来查找 JMS ConnectionFactory，而不是在 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。

51.13. 并发使用

JMS 的一个常见要求是同时使用多个线程中的消息，以便应用更迅速地响应。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,
- 在端点 URI 或.
- 在 JmsEndpoint 上直接调用 setConcurrentConsumers （）。

51.13.1. 使用 async consumer 的并发使用

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，以便消费者从 JMS 队列中获取下一个消息，同时将前面的消息异步处理（通过异步路由引擎）。请参阅页顶部的有关 asyncConsumer 选项的详情。

```
from("jms:SomeQueue?concurrentConsumers=20&asyncConsumer=true").
  bean(MyClass.class);
```

51.14. JMS 上的 REQUEST-REPLY

Camel 支持 Request Reply over JMS。本质上，当您向 JMS 队列发送消息时，Exchange 的 MEP 应该为 InOut。

Camel 提供了很多选项，来通过 JMS 配置请求/回复，这会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，由 Camel 自动创建。要使用此选项，请不要 指定 replyTo 队列名称。另外，您还可以配置 **replyToType=Temporary**，使它成为正在使用该临时队列。

选项	性能	集群	描述
共享	速度较慢	是	共享持久队列用作回复队列。必须先创建队列，但某些代理可以实时创建它们，如 Apache ActiveMQ。要使用此功能，您必须指定 replyTo 队列名称。另外，您还可以配置 **replyToType=Shared**，使其认为使用共享队列。共享队列可用于同时运行此 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；这会影响到性能。JMS 消息选择器较慢，因此无法快速作为 **Temporary** 或 **Exclusive** 队列。请参阅以下如何调整它以提高性能。
exclusive	速度快	否(*是)	专用持久队列用作回复队列。必须先创建队列，但某些代理可以实时创建它们，如 Apache ActiveMQ。要使用此功能，您必须指定 replyTo 队列名称。此外，**您必须配置 replyToType=Exclusive** 来指示 Camel 使用专用队列，因为如果配置了 **replyTo** 队列名称，则默认使用 **Shared**。在使用专用回复队列时，**不使用** JMS 消息选择器，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** 如果您为每个节点配置带有唯一的名称的 Exclusive reply 队列，您可以在集群环境中运行。与那时，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	允许使用并发消息监听程序同时处理回复消息。您可以使用 **concurrentConsumers** 和 **maxConcurrentConsumers** 选项指定范围。**注意：使用共享** 回复队列可能无法用于并发监听程序，因此请谨慎使用这个选项。
maxConcurrentConsumers	速度快	是	允许使用并发消息监听程序同时处理回复消息。您可以使用 **concurrentConsumers** 和 **maxConcurrentConsumers** 选项指定范围。**注意：使用共享** 回复队列可能无法用于并发监听程序，因此请谨慎使用这个选项。

JmsProducer 检测到 InOut，并提供 JMSReplyTo 标头，其中包含要使用的回复目的地。默认情况下，Camel 使用临时队列，但您可以使用端点上的 replyTo 选项指定固定回复队列（请参阅以下关于固定回复队列的信息）。

Camel 将自动设置侦听回复队列的消费者，因此 您不应 执行任何操作。
*这**个消费者是一个 Spring DefaultMessageListenerContainer，用于侦听回复。但是，它被修复为 1 个并发消费者。***

这意味着，回复将按顺序处理，因为只有 1 个线程来处理回复。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 中更轻松地进行配置，如下所示：

```
from(xxx)
.inOut().to("activemq:queue:foo?concurrentConsumers=5")
.to(yyy)
.to(zzz);
```

在这个路由中，我们指示 Camel 使用有 5 个线程的线程池异步路由回复。

51.14.1. 通过 JMS 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 时使用了固定的回复队列，如以下示例所示，请留意。

```
from(xxx)
.inOut().to("activemq:queue:foo?replyTo=bar")
.to(yyy)
```

在本例中，使用了名为 "bar" 的固定回复队列。默认情况下，Camel 假设队列在使用固定回复队列时共享，因此它使用 JMSSelector 仅提取预期的回复消息（例如，基于 JMSCorrelationID）。有关专用固定回复队列，请参见下一部分。这意味着它不像临时队列一样快。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，要更快地将其设置为 250 个 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。通常建议尽可能使用临时队列。

51.14.2. 通过 JMS 进行请求回复，并使用一个专用的固定回复队列

在上例中，Camel 会预计名为"bar"的固定回复队列是共享的，因此它使用 JMSSelector 仅消耗它期望的回复消息。但是，由于 JMS 选择器会较慢，所以这样做有一个缺陷。此外，回复队列上的消费者使用新的 JMS 选择器 ID 进行更新较慢。实际上，它只在 receiveTimeout 选项超时时更新，默认为 1 秒。因此，在理论上，回复消息可能需要大约 1 秒才会被检测到。另一方面，如果固定的回复队列专用于 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，像使用临时队列一样快。有 ReplyToType 选项，您可以将它配置为 Exclusive
来告知 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)
```

请记住，队列必须为每个端点和每个端点排斥。因此，如果您有两个路由，则每个路由都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，则同样适用。然后，集群中的每个节点都必须使用唯一的回复队列名称。如其他情况下，集群中的每个节点可能会获取旨在作为另一节点上的回复的消息。对于集群环境，建议改为使用共享回复队列。

51.15. 在发送方和接收方之间同步时钟

在系统间执行消息传递时，需要系统已同步时钟。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方都有同步时钟。如果使用 ActiveMQ，您可以使用 timestamp 插件 同步时钟。

51.16. 关于生存时间

阅读上面的有关时钟时钟的上方。

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/低值来控制这一点。但是，在要发送的消息上仍然设置了生存时间值。这需要在系统间同步时钟。如果没有，您可能需要禁用要设置的实时值的时间。现在，可以使用 Camel 2.8 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。也可以配置该选项。因此，这两个选项 requestTimeout 和 disableTimeToLive 可让您在进行 request/reply 时进行精细的控制。

您可以在消息中提供标头来覆盖，并用作请求超时值，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 endpoint configured requestTimeout 为 30 秒。因此，Camel 将等待 30 秒，以便回复消息回到栏队列。如果没有收到回复消息，则在 Exchange 上设置了 org.apache.camel.ExchangeTimedOutException，并且 Camel 继续路由消息，然后因为异常而失败，Camel 的错误处理器响应响应。

如果要使用每个消息超时值，您可以使用键 org.apache.camel.component.jms.JmsConstants#JMS_REQUEST_TIMEOUT 设置标头，其常量值为 "CamelJmsRequestTimeout"，其超时值为长类型。

例如，我们可以使用 bean 计算每个独立消息的超时值，如调用服务 bean 上的 "whatIs TheTimeout" 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，要指示 5 秒。设置 timeToLive=5000。选项 disableTimeToLive 可用于强制禁用生存时间，也可以用于 InOnly messaging。requestTimeout 选项不用于 InOnly messaging。

51.17. 启用转换的消耗

常见要求是从事务中的队列使用，然后使用 Camel 路由处理消息。要做到这一点，请确定您在 component/endpoint 中设置以下属性：

- transacted = true
- transactionManager = Transsaction Manager - 通常 JmsTransactionManager

详情请查看 Transactional Client EIP 模式。

通过 JMS 进行事务处理和 [Request Reply]

当使用 Request Reply over JMS 时，您无法使用单个事务；JMS 不会在执行提交前发送任何消息，因此在事务提交前，服务器端不会收到任何内容。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接收响应。

要解决这个问题，JMS 组件使用不同的属性来指定用于单向消息传递和请求回复消息传递的事务：

transacted 属性只适用于 InOnly message Exchange Pattern (MEP)。

您可以使用以下 component/endpoint 的属性利用 DMLC 转换会话 API ：

- transacted = true
- lazyCreateTransactionManager = false

这样做的好处是，在没有配置的 TransactionManager 的情况下使用本地事务时，将遵循 cacheLevel 设置。当配置了 TransactionManager 时，在 DMLC 级别不需要缓存，需要依赖池的连接工厂。有关这类设置的详情，请查看 这里 和 此处。

51.18. 使用 JMSREPLYTO 进行更新的回复

当使用 Camel 作为 JMS 侦听器时，它会设置一个 Exchange 属性，其值为 ReplyTo javax.jms.Destination 对象，其键为 ReplyTo。您可以按如下方式 获取 此目标：

```
Destination replyDestination =
exchange.getIn().getHeader(JmsConstants.JMS_REPLY_DESTINATION, Destination.class);
```

然后，稍后使用它来使用常规 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 将获取此属性并将其用于实际目的地。端点 URI 必须包含一个 dummy 目的地。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.");
  }
}
```

51.19. 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 方法 = InOut）发送到 Camel 中进一步处理较慢的队列，并等待返回回复：

51.20. 发送 INONLY 消息并保留 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。但是，在某些情况下，您可能要发送 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，但带有 bar 队列的 JMSReplyTo，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。

51.21. 在目的地上设置 JMS 供应商选项

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 目的地名称上设置它，如下所示：

```
// ...
.setHeader("CamelJmsDestinationName", constant("queue:///MY_QUEUE?targetClient=1"))
.to("wmq:queue:MY_QUEUE?useMessageIDAsCorrelationID=true");
```

有些版本的 WMQ 不会在目标名称上接受这个选项，您会看到如下例外：

```
com.ibm.msg.client.jms.DetailedJMSException: JMSCC0005: The specified
value 'MY_QUEUE?targetClient=1' is not allowed for
'XMSC_DESTINATION_NAME'
```

一个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");
    }
});
```

51.22. 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.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，您可以考虑启用此选项，同时仍然在队列上排队消息。如果此选项为 false，并且您停止了 JMS 路由，则消息可能会被拒绝，而 JMS 代理必须尝试重新设计，这一次可能被拒绝，最终该消息可能会在 JMS 代理上的死信队列中移动。要避免这种情况，建议启用这个选项。	false	布尔值
camel.component.jms.acknowledgement-mode-name	JMS 确认名称，即： SESSION_TRANSACTED、CLIENT_ACKNOWLEDGE、AUTO_ACKNOWLEDGE、DUPS_OK_ACKNOWLEDGE。	AUTO_ACKNOWLEDGE	字符串
camel.component.jms.allow-additional-headers	此选项用于允许其他标头，它们可能具有根据 JMS 规范无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 来执行此操作，其中包含带有字节数组或其他无效类型的值。您可以指定多个标头名称，用逗号分开，并使用 作为通配符匹配的后缀。		字符串

Name	描述	默认值	类型
camel.component.jms.allow-auto-wired-connection-factory	如果没有配置连接工厂，是否从 registry 中自动发现 ConnectionFactory。如果只找到了一个 ConnectionFactory 实例，则会使用它。这默认是启用的。	true	布尔值
camel.component.jms.allow-auto-wired-destination-resolver	如果没有配置目标解析器，是否从 registry 中自动发现 DestinationResolver。如果只找到一个 DestinationResolver 实例，则会使用它。这默认是启用的。	true	布尔值
camel.component.jms.allow-null-body	是否允许发送不包含正文的消息。如果此选项为 false，并且消息正文为 null，则会抛出一个 JMSException。	true	布尔值
camel.component.jms.allow-reply-manager-quick-stop	是否在回复管理器中使用 DefaultMessageListenerContainer 用于 request-reply 消息，都允许 DefaultMessageListenerContainer.runningAllowed 标志在启用了 JmsConfiguration#isAcceptMessagesWhileStopping 时快速停止，并且 org.apache.camel.CamelContext 当前已停止。在常规 JMS 消费者中默认启用这种快速停止功能，但为了启用此标志，您必须启用此标志。	false	布尔值
camel.component.jms.allow-serialized-headers	控制是否包含序列化标头。仅在 transferExchange 为 true 时应用。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值
camel.component.jms.always-copy-message	如果为 true，Camel 会在传递给生成者发送时始终生成邮件的 JMS 消息副本。在某些情况下需要复制消息，比如当设置了 replyToDestinationSelectorName 时（通常，如果设置了 replyToDestinationSelectorName），则 Camel 会将 alwaysCopyMessage 选项设置为 true。	false	布尔值
camel.component.jms.artemis-consumer-priority	通过消费者优先级，您可以确保高优先级消费者在激活时收到消息。通常，连接到队列的活动用户以轮循方式从它接收消息。使用消费者优先级时，如果有多个活跃的消费者具有相同的高优先级，则会发送循环消息。只有高优先级消费者没有使用消息的信用时，消息才会受到较低优先级的用户，或者那些高优先级消费者已拒绝接受该消息（例如，因为它不符合与消费者关联的任何选择器的条件）。		整数

Name	描述	默认值	类型
camel.component.jms.artemis-streaming-enabled	是否针对 Apache Artemis 流模式进行优化。当将 Artemis 与 JMS StreamMessage 类型一起使用时，这可以减少内存开销。只有在使用 Apache Artemis 时，才必须启用此选项。	false	布尔值
camel.component.jms.async-consumer	JmsConsumer 是否异步处理 Exchange。如果启用，则 JmsConsumer 可能会从 JMS 队列中获取下一个消息，而前面的消息会被异步处理（通过异步路由引擎）。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 从 JMS 队列获取下一个消息前完全处理 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 无法获得与远程 JMS 代理的连接，那么在重试和/或故障转移时可能会阻止它。这会导致 Camel 在启动路由时阻止。通过将这个选项设置为 true，您可以让路由启动，而 JmsConsumer 使用异步模式的专用线程连接到 JMS 代理。如果使用此选项，请注意，如果无法建立连接，则会在 WARN 级别记录异常，消费者将无法接收消息；然后，您可以重启要重试的路由。	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该自动启动。	true	布尔值
camel.component.jms.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.jms.cache-level	根据 ID 为底层 JMS 资源设置缓存级别。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

Name	描述	默认值	类型
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，且只能被单个 JMS 连接实例使用。它通常只需要使用 JMS 1.1 的持久主题订阅		字符串
camel.component.jms.concurrent-consumers	指定从 JMS 消耗时的默认并发用户数（不适用于 JMS 上的请求/回复）。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。当对 JMS 进行请求/回复时，选项 replyToConcurrentConsumers 用于控制回复消息监听器上的并发用户数量。	1	整数
camel.component.jms.configuration	使用共享的 JMS 配置。选项是 org.apache.camel.component.jms.JmsConfiguration 类型。		JmsConfiguration
camel.component.jms.connection-factory	要使用的连接工厂。连接工厂必须在组件或端点上配置。选项是 javax.jms.ConnectionFactory 类型。		ConnectionFactory
camel.component.jms.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。指定 Custom 时，messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 选项将决定要使用的 org.springframework.jms.listener.AbstractMessageListenerContainer。		ConsumerType
camel.component.jms.correlation-property	使用 InOut 交换模式时，请使用此 JMS 属性而不是 JMSCorrelationID JMS 属性来关联消息。如果设置消息将只与此属性的 JMSCorrelationID 属性的值关联，则忽略且不由 Camel 设置。		字符串
camel.component.jms.default-task-executor-type	指定 DefaultMessageListenerContainer 中使用哪些默认 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 - 缓存的 threadpool-like）。如果没有设置，则默认为前面的行为，它对消费者使用缓存的线程池，并将 SimpleAsync 用于回复消费者。建议使用 ThreadPool 来减少弹性配置中的线程垃圾箱，并动态增加和减少并发消费者。		DefaultTaskExecutorType

Name	描述	默认值	类型
camel.component.jms.delivery-delay	设置用于为 JMS 发送调用的交付延迟。这个选项需要 JMS 2.0 兼容代理。	-1	Long
camel.component.jms.delivery-mode	指定要使用的交付模式。可能的值有 javax.jms.DeliveryMode 定义的值。NON_PERSISTENT = 1 和 PERSISTENT = 2.		整数
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。		DestinationResolver
camel.component.jms.disable-reply-to	指定 Camel 是否忽略消息中的 JMSReplyTo 标头。如果为 true，Camel 不会向 JMSReplyTo 标头中指定的目的地发送回复。如果您希望 Camel 消耗路由，且您不想 Camel 自动发送回复消息，您可以使用这个选项，因为代码中的另一个组件处理回复消息。如果要使用 Camel 作为不同消息代理之间的代理，而您想要将消息从一个系统路由到另一个系统，也可以使用此选项。	false	布尔值
camel.component.jms.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 值作为发送消息的时间。问题是发送方和接收器系统必须同步其时钟，因此它们同步。这并非始终容易存档。因此，您可以使用 disableTimeToLive=true 来在发送的消息上将时间设置为 live 值。然后，该消息不会在接收器系统上过期。如需了解更多详细信息，请参见以下小节中关于 live 的时间。	false	布尔值
camel.component.jms.durable-subscription-name	用于指定持久主题订阅的持久订阅者名称。也必须配置 clientId 选项。		字符串
camel.component.jms.eager-loading-of-properties	加载消息后马上启用对 JMS 属性和有效负载的 eager 加载，这通常效率低下，因为 JMS 属性可能不需要，但有时可以尽早地捕获与底层 JMS 提供程序和使用 JMS 属性相关的问题。另请参阅 eagerPoisonBody 选项。	false	布尔值

Name	描述	默认值	类型
camel.component.jms.eager-poison-body	如果启用了 eagerLoadingOfProperties，并且 JMS 消息有效负载(JMS 正文或 JMS 属性)为 poison （不能读取/映射），则将此文本设置为消息正文，以便处理消息正文（导致 poison 的原因在 Exchange 上已存储为例外）。这可以通过设置 eagerPoisonBody=false 来关闭。另请参阅选项 eagerLoadingOfProperties。	因 $\{exception.message} 导致的 Poison JMS 消息	字符串
camel.component.jms.enabled	是否启用 jms 组件的自动配置。这默认是启用的。		布尔值
camel.component.jms.error-handler	指定在处理消息时抛出异常时调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置 errorHandler，则会在 WARN 级别中记录这些例外。您可以配置日志记录级别，以及堆栈跟踪是否应该使用 errorHandlerLoggingLevel 和 errorHandlerLogStackTrace 选项记录。这样可以更容易配置，而不必对自定义 errorHandler 进行编码。选项是一个 org.springframework.util.ErrorHandler 类型。		ErrorHandler
camel.component.jms.error-handler-log-stack-trace	允许默认 errorHandler 控制是否应记录 stacktrace。	true	布尔值
camel.component.jms.error-handler-logging-level	允许为日志记录无法捕获的异常配置默认 errorHandler 日志记录级别。		LoggingLevel
camel.component.jms.exception-listener	指定要收到任何底层 JMS 异常的 JMS Exception Listener。选项是 javax.jms.ExceptionListener 类型。		ExceptionListener
camel.component.jms.explicit-qos-enabled	设定在发送消息时应使用 deliveryMode、priority 或 timeToLiveity 的服务质量。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用于当前端点。这与 preserveMessageQos 选项不同，该选项以消息粒度运行，读取仅来自 Camel In 消息标头的 QoS 属性。	false	布尔值
camel.component.jms.expose-listener-session	指定在消耗消息时是否应公开侦听器会话。	false	布尔值

Name	描述	默认值	类型
camel.component.jms.force-send-original-message	使用 mapJmsMessage=false Camel 时，如果您在路由过程中涉及标头(get 或 set)，则创建新的 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 JMS 日期属性是否应根据 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 过滤标头到 Camel 消息。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		HeaderFilterStrategy
camel.component.jms.idle-consumer-limit	指定允许在任何给定时间闲置的用户数量的限制。	1	整数
camel.component.jms.idle-task-execution-limit	指定接收任务的闲置执行的限制，而不是在其执行中收到任何消息。如果达到这个限制，则任务将关闭并离开接收其他执行任务（在动态调度的情况下；请参阅 maxConcurrentConsumers 设置）。Spring 中还有额外的文档。	1	整数
camel.component.jms.include-all-j-m-s-x-properties	在从 JMS 到 Camel 消息映射时，是否包含所有 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，则这个选项不适用。	false	布尔值
camel.component.jms.include-sent-j-m-s-message-i-d	仅在使用 InOnly 发送到 JMS 目的地时（例如触发和忘记）。启用此选项将增强 Camel Exchange 与实际的 JMSMessageID，在消息发送到 JMS 目的地时供 JMS 客户端使用。	false	布尔值
camel.component.jms.jms-key-format-strategy	用于编码和解码 JMS 密钥的可插拔策略，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。默认策略将安全地 marshal 句点和连字符(. 和 -)。passthrough 策略将密钥保留为原样。可用于 JMS 代理，这些代理不关心 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.jms-message-type	允许您强制使用特定的 javax.jms.Message 实现来发送 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 body 类型中使用的 JMS 消息类型。这个选项允许您指定它。		JmsMessageType

Name	描述	默认值	类型
camel.component.jms.lazy-create-transaction-manager	如果为 true，则 Camel 将创建一个 JmsTransactionManager，如果没有在选项 transacted=true 时注入任何 transactionManager。	true	布尔值
camel.component.jms.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.jms.map-jms-message	指定 Camel 是否将收到的 JMS 消息自动映射到适合的有效负载类型，如 javax.jms.TextMessage 到 String 等。	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（不适用于 JMS 上的请求/回复）。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。当对 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消费者数量。		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（例如 min max），则此选项可用于设置 eg 100 来控制在需要较少工作时消费者缩小的速度。	-1	整数
camel.component.jms.message-converter	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。		MessageConverter
camel.component.jms.message-created-strategy	使用在 Camel 发送 JMS 消息时调用的 given MessageCreatedStrategy，在 Camel 创建 javax.jms.Message 对象的新实例时调用。选项是一个 org.apache.camel.component.jms.MessageCreatedStrategy 类型。		MessageCreatedStrategy
camel.component.jms.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 供应商接受这个 hint，则这些消息必须将消息 ID 设置为 null；如果供应商忽略 hint，则消息 ID 必须设置为其正常唯一值。	true	布尔值

Name	描述	默认值	类型
camel.component.jms.message-listener-container-factory	用于决定使用消息的 org.springframework.jms.listener.AbstractMessageListenerContainer 的 MessageListenerContainerFactory 的 registry ID。设置此选项会自动将 consumerType 设置为 Custom。选项是 org.apache.camel.component.jms.MessageListenerContainerFactory 类型。		MessageListenerContainerFactory
camel.component.jms.message-timestamp-enabled	指定默认情况下，是否应在发送消息时启用时间戳。这只是 JMS 代理的提示。如果 JMS 供应商接受这个 hint，则这些消息必须将时间戳设置为零；如果供应商忽略 hint，则必须将时间戳设置为其正常值。	true	布尔值
camel.component.jms.password	用于 ConnectionFactory 的密码。您也可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
camel.component.jms.preserve-message-qos	如果设置为 true，如果要使用消息中指定的 QoS 设置发送消息，而不是 JMS 端点上的 QoS 设置。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供 all 或 only some them。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 选项将使用端点上设置的选项，而不是来自消息标头的值。	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 1 是最低优先级，9 为最高）。还必须启用 explicitQosEnabled 选项，以便此选项有任何效果。	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。		QueueBrowseStrategy
camel.component.jms.receive-timeout	接收消息的超时时间（以毫秒为单位）。选项是一个长类型。	1000	Long
camel.component.jms.recovery-interval	指定恢复尝试之间的间隔，即刷新连接时（以毫秒为单位）。默认值为 5000 ms，即 5 秒。选项是一个长类型。	5000	Long

Name	描述	默认值	类型
camel.component.jms.reply-to	提供显式 ReplyTo 目的地（会覆盖消费者中的 Message.getJMSReplyTo （）的任何传入值）。		字符串
camel.component.jms.reply-to-cache-level-name	在通过 JMS 进行请求/回复时，根据回复消费者设置缓存级别。这个选项只适用于使用固定回复队列（而非临时）。默认情况下，Camel 将使用：CACHE_CONSUMER 用于 exclusive 或 shared w/ replyToSelectorName。用于没有 replyToSelectorName 的共享的 CACHE_SESSION。IBM WebSphere 等一些 JMS 代理可能需要设置 replyToCacheLevelName=CACHE_NONE 才能工作。注意：如果使用临时队列，则不允许 CACHE_NONE，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	指定在对 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，以控制线程的动态扩展/关闭。	1	整数
camel.component.jms.reply-to-delivery-persistent	指定是否默认使用持久性发送进行回复。	true	布尔值
camel.component.jms.reply-to-destination-selector-name	使用要使用的固定名称设置 JMS Selector，以便在使用共享队列时过滤您自己的回复（即，如果您不使用临时回复队列）。		字符串
camel.component.jms.reply-to-max-concurrent-consumers	指定在 JMS 上使用请求/回复时的最大并发用户数。另请参阅 maxMessagesPerTask 选项，以控制线程的动态扩展/关闭。		整数
camel.component.jms.reply-to-on-timeout-max-concurrent-consumers	指定在通过 JMS 使用请求/回复时，进行超时时继续路由的并发消费者的最大数量。	1	整数
camel.component.jms.reply-to-override	在 JMS 消息中提供显式 ReplyTo 目的地，它会覆盖 replyTo 的设置。如果您要将消息转发到远程队列，并从 ReplyTo 目的地接收回复消息，这很有用。		字符串
camel.component.jms.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发送相同消息到其自身的无端循环。	false	布尔值

Name	描述	默认值	类型
camel.component.jms.reply-to-type	在通过 JMS 进行 request/reply 时，允许显式指定用于 replyTo 队列的策略类型。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。这个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的影响的信息，以及共享回复队列的性能比其 alternatives Temporary 和 Exclusive 的性能较低。		ReplyToType
camel.component.jms.request-timeout	使用 InOut Exchange Pattern （毫秒）时等待回复的超时时间。默认值为 20 秒。您可以包含标头 CamelJmsRequestTimeout 来覆盖这个端点配置的超时值，因此每个消息单个超时值。另请参阅 requestTimeoutCheckerInterval 选项。选项是一个长类型。	20000	Long
camel.component.jms.request-timeout-checker-interval	配置 Camel 在通过 JMS 进行请求/回复时应检查超时交换的频率。默认情况下，Camel 会每秒检查一次。但是，如果您在超时时必须更快地响应，您可以降低这个间隔，以便更频繁地检查。超时由选项 requestTimeout 决定。选项是一个长类型。	1000	Long
camel.component.jms.selector	设置要使用的 JMS 选择器。		字符串
camel.component.jms.stream-message-type-enabled	设置 StreamMessage 类型是否已启用。通过以 BytesMessage 或 StreamMessage 发送的消息有效负载，如 files、InputStream 等。这个选项控制将使用的 kind。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。通过启用这个选项，消息有效负载以块的形式读取到内存中，每个块都会被写入 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化。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。在带有共享或持久订阅的主题(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：每个订阅只允许 1 个并发消费者（这是此消息监听程序容器的默认值），除了一个共享订阅（需要 JMS 2.0）。		字符串

Name	描述	默认值	类型
camel.component.jms.subscription-shared	设置是否共享订阅。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 来注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以作为订阅名称使用）。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)时有意义，因此此方法也会切换 pubSubDomain 标志。需要 JMS 2.0 兼容消息代理。	false	布尔值
camel.component.jms.synchronous	设置是否应严格使用同步处理。	false	布尔值
camel.component.jms.task-executor	允许您指定自定义任务 executor 以供使用消息。选项是一个 org.springframework.core.task.TaskExecutor 类型。		TaskExecutor
camel.component.jms.test-connection-on-startup	指定是否在启动时测试连接。这样可确保 Camel 启动所有 JMS 用户具有与 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 没有使用失败的连接启动。JMS 制作者也经过测试。	false	布尔值
camel.component.jms.time-to-live	在发送消息时，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 transacted 模式。	false	布尔值
camel.component.jms.transacted-in-out	指定 InOut 操作（请求回复）是否默认使用 transacted 模式（如果此标志设为 true），则 Spring JmsTemplate 会将 sessionTransacted 设置为 true，以及 acknowledgeMode 作为 InOut 操作的 JmsTemplate 的 transacted。注意从 Spring JMS：在 JTA 事务中，传递给 createQueue 的参数，不会考虑 createTopic 方法。根据 Java EE 事务上下文，容器对这些值自行做出决定。类似地，这些参数不会被在本地管理的事务中考虑，因为本例中的 Spring JMS 在现有的 JMS Session 上运行。在受管事务之外运行时，将此标志设置为 true 将使用简短的本地 JMS 事务，并在存在受管事务（并非 XA 事务）的情况下同步的本地 JMS 事务。这将与主事务一起管理本地 JMS 事务（可能是原生 JDBC 事务），而 JMS 事务会在主事务后提交右边。	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。		PlatformTransactionManager

Name	描述	默认值	类型
camel.component.jms.transaction-name	要使用的事务的名称。		字符串
camel.component.jms.transaction-timeout	使用 transacted 模式，事务的超时值（以秒为单位）。	-1	整数
camel.component.jms.transfer-exception	如果启用了，并且您在使用 Request Reply messaging (InOut)，且 Exchange 在消费者端失败，则原因例外将作为 javax.jms.ObjectMessage 发回。如果客户端是 Camel，则返回的例外将被重新箭头。这样，您可以在路由中使用 Camel JMS 作为网桥 - 例如，使用持久性队列启用可靠的路由。请注意，如果您也启用了 transferExchange，这个选项将具有优先权。需要 Caught 异常才能按顺序排序。消费者端的原始 Exception 可以嵌套在外部异常中，如返回到制作者时 org.apache.camel.RuntimeCamelException。请小心谨慎，因为数据使用 Java 对象序列化，并且要求收到的能够在类级别上反序列化数据，这会强行生产者和消费者之间的强耦合。	false	布尔值
camel.component.jms.transfer-exchange	您可以通过线路传输交换，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 headers, exchange properties, exchange exception。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，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 知道有效负载是一个交换而不是常规有效负载。请小心谨慎，因为数据使用 Java 对象序列化，并且要求接收器能够在类级别上反序列化数据，这会强制生产者与需要使用兼容 Camel 版本的消费者之间强耦耦合！	false	布尔值
camel.component.jms.use-message-i-d-as-correlation-i-d	指定 JMSMessageID 是否始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值
camel.component.jms.username	用于 ConnectionFactory 的用户名。您也可以直接在 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，等待 provisional correlation id 更新为实际关联 ID 的次数，以及启用选项 useMessageIDAsCorrelationID 的时间。	50	整数

Name	描述	默认值	类型
camel.component.jms.wait-for-provision-correlation-to-be-updated-thread-sleeping-time	等待更新调配关联 id 期间每次处于睡眠状态的间隔。选项是一个长类型。	100	Long

第 52 章 JPA

Since Camel 1.0

支持生成者和消费者。

JPA 组件使您能够使用 EJB 3 的 Java 持久性架构(jpa)从持久性存储中存储和检索 Java 对象。Java Persistence 架构(JPA)是一个标准接口层，用于包装对象/关系映射(ORM)产品，如 OpenuildDefaults、Hibernate、SupLink。

52.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 jpa 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-jpa-starter</artifactId>
</dependency>
```

52.2. 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，其上带有 @Entity 注释的 POJO）或实体 Bean 的集合或数组。

如果正文是实体列表，请使用 entityType=java.util.List 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，请在端点前放置一个 Message Translator，请首先执行必要的转换。

您还可以将名为Query 或 nativeQuery 的查询用于制作者。对于 参数的值，您可以使用 Simple 表达式，允许您从 Message body、标头等中检索参数值。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 语句执行批量更新/删除。请注意，如果使用 namedQuery 执行 UPDATE/DELETE，则需要指定 useExecuteUpdate to true，因为 camel 不会查找与查询和 nativeQuery 不同命名的查询。

52.3. 从端点消耗

从 JPA consumer 端点消耗消息会删除（或更新）数据库中实体 Bean。这样，您可以将数据库表用作逻辑队列：消费者从队列中获取消息，然后将其删除/更新来逻辑地从队列中删除它们。

如果您不想在处理时（以及路由完成后）删除实体 bean，您可以在 URI 中指定 consumeDelete=false。这将导致处理每个轮询的实体。

如果您要对实体执行一些更新，将其标记为已处理（例如，将它从未来的查询中排除），那么您可以在处理实体 Bean 时给一个 @Consumed 标注方法，该方法将在实体 Bean 上调用（以及路由完成时）。

您可以使用 @PreConsumed，它会在处理之前在实体 bean 上调用（在路由之前）。

如果您消耗了大量行(100K+)，并遇到 OutOfMemory 问题，您应该将 maximumResults 设置为 sensible 值。

52.4. URI 格式

```
jpa:entityClassName[?options]
```

要发送到端点，entityClassName 是可选的。如果指定，它可以帮助 Type Converter 来确保正文是正确的类型。

为消耗，实体ClassName 是强制的。

52.5. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

52.5.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

52.5.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

52.6. 组件选项

JPA 组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
别名 (common)	将别名映射到 JPA 实体类。然后，别名可以在端点 URI （而不是完全限定类名称）中使用。		Map
entityManagerFactory (common)	使用 EntityManagerFactory。强烈建议您进行配置。		EntityManagerFactory
joinTransaction (common)	camel-jpa 组件默认加入事务。您可以使用这个选项来关闭这个选项，例如，如果您使用 LOCAL_RESOURCE，并加入事务与 JPA 供应商无法工作。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

Name	描述	默认值	类型
sharedEntityManager (common)	是否将 Spring 的 SharedEntityManager 用于 consumer/producer。在大多数情况下，joinTransaction 应设置为 false，因为这不是 EXTENDED EntityManager。	false	布尔值
transactionManager (common)	使用 PlatformTransactionManager 管理事务。		PlatformTransactionManager
transactionStrategy (common)	使用 TransactionStrategy 在事务中运行操作。		TransactionStrategy
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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

52.6.1. 端点选项

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：

52.6.1.1. 路径参数(1 参数)

Name	描述	默认值	类型
entityType (common)	**必需的** 实体类名称。		类

52.6.1.2. 查询参数(44 参数)

Name	描述	默认值	类型
joinTransaction (common)	camel-jpa 组件默认加入事务。您可以使用这个选项来关闭这个选项，例如，如果您使用 LOCAL_RESOURCE，并加入事务与 JPA 供应商无法工作。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
namedQuery (common)	使用命名查询。		字符串
nativeQuery (common)	使用自定义原生查询。在使用原生查询时，您可能还想使用选项 resultClass。		字符串
PersistenceUnit (common)	**需要** 默认情况下使用的 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的类型（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。仅在使用数据时与原生查询结合使用时具有影响。		类
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)	指定是否在处理轮询结果时在每个实体 bean 上设置专用锁定。	true	布尔值
deleteHandler (consumer)	使用自定义 DeleteHandler 在处理交换后删除行。		DeleteHandler

Name	描述	默认值	类型
lockModeType (consumer)	在消费者上配置锁定模式。 Enum 值： • READ • 写 • OPTIMISTIC • OPTIMISTIC_FORCE_INCREMENT • PESSIMISTIC_READ • PESSIMISTIC_WRITE • PESSIMISTIC_FORCE_INCREMENT • NONE	PESSIMISTIC_WRITE	LockModeType
maxMessagesPerPoll (consumer)	整数值，用于定义每个轮询收集的最大消息数。默认情况下，不会设置最大值。可用于避免在启动服务器时轮询数千个消息。将值设为 0 或负数设置为 disable。		int
preDeleteHandler (consumer)	使用自定义 Pre-DeleteHandler 在消费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)	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)	是否在处理整个批处理时以转换模式运行消费者，所有消息都将提交或回滚。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
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
参数 (consumer (advanced))	此键/值映射用于构建查询参数。它预期是通用类型 java.util.Map，其中键是给定 JPA 查询的命名参数，值是您要为选择的对应有效值。当用于制作者时，简单表达式可用作参数值。它允许您从消息正文、标头等中检索参数值。		Map
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
findEntity (producer)	如果启用，则制作者将通过将消息正文用作 key 和 entityType 作为类类型来查找单个实体。这可用于查找单个实体，而不使用查询。	false	布尔值
flushOnSend (producer)	在实体 bean 持久保留后，清除 EntityManager。	true	布尔值
remove (producer)	指明使用 entityManager.remove (entity)。	false	布尔值
useExecuteUpdate (producer)	配置在生成者执行查询时是否使用 executeUpdate（）。当您将 INSERT、UPDATE 或 DELETE 语句用作命名查询时，您需要将此选项指定为 'true'。		布尔值
usePersist (producer)	指明使用 entityManager.persist (entity)而不是 entityManager.merge (entity)。注：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
lazyStartProducer (producer (advanced))	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
usePassedInEntityManager (producer (advanced))	如果设置为 true，则 Camel 将使用标题 JpaConstants.ENTITY_MANAGER 中的 EntityManager，而不是组件/端点上配置的实体管理器。这允许最终用户控制将使用的实体管理器。	false	布尔值
entityManagerProperties (advanced)	要使用的实体管理器的其他属性。		Map
sharedEntityManager (advanced)	是否将 Spring 的 SharedEntityManager 用于 consumer/producer。在大多数情况下，joinTransaction 应设置为 false，因为这不是 EXTENDED EntityManager。	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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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

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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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

52.7. 消息标头

JPA 组件支持 2 个消息标头，如下所列：

Name	描述	默认值	类型
CamelEntityManager (common) 常数：ENTITY_MANAGER	JPA EntityManager 对象。		EntityManager
CamelJpaParameters (producer) 恒定：link: JPA_PARAMETERS_HEADER	将查询参数作为 Exchange 标头传递的替代方法。		Map

52.8. 配置 ENTITYMANAGERFACTORY

建议将 JPA 组件配置为使用特定的 EntityManagerFactory 实例。如果这样做失败，每个 JpaEndpoint 将自动创建自己的 EntityManagerFactory 实例，这最常不是您需要的内容。

例如，您可以实例化引用 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>
```

JpaComponent 会自动从 Registry 中查找 EntityManagerFactory，这意味着您不需要在 JpaComponent 上进行配置。只有有不确定性时，您只需要这样做，在这种情况下，Camel 将记录 WARN。

52.9. 配置 TRANSACTIONMANAGER

JpaComponent 会自动从 Registry 中查找 TransactionManager。如果 Camel 找不到注册了任何 TransactionManager 实例，它也会查找 TransactionTemplate，并尝试从其中提取 TransactionManager。

如果 registry 中没有 TransactionTemplate，则 JpaEndpoint 将自动创建自己的 TransactionManager 实例，其中最多不是您想要的内容。

如果找到多个 TransactionManager 实例，Camel 将记录 WARN。在这种情况下，您可能想要实例化并明确配置引用 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>
```

52.10. 使用带有命名查询的消费者

对于只使用所选实体，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 {
  ...
}
```

之后，您可以定义一个消费者 uri，如下所示：

```
from("jpa://org.apache.camel.examples.MultiSteps?namedQuery=step1")
.to("bean:myBusinessLogic");
```

52.11. 使用带有查询的消费者

对于只使用所选实体，您可以使用 查询 URI 查询选项。您只需要定义查询选项：

```
from("jpa://org.apache.camel.examples.MultiSteps?query=select o from
org.apache.camel.examples.MultiSteps o where o.step = 1")
.to("bean:myBusinessLogic");
```

52.12. 使用带有原生查询的消费者

对于只使用所选实体，您可以使用 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?nativeQuery=select * from MultiSteps
where step = 1")
.to("bean:myBusinessLogic");
```

如果使用 native query 选项，您将在消息正文中收到一个对象数组。

52.13. 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 {
    ...
}
```

之后，您可以定义一个制作者 uri，如下所示：

```
from("direct:namedQuery")
.to("jpa://org.apache.camel.examples.MultiSteps?namedQuery=step1");
```

请注意，您需要指定 useExecuteUpdate 选项为 true，以执行 UPDATE/DELETE 语句作为命名查询。

52.14. 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");
```

52.15. 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，您将在消息正文中收到对象数组。

52.16. 使用基于 JPA 的 IDEMPOTENT 存储库

来自 EIP 模式的 Idempotent Consumer 用于过滤重复的信息。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，它供 org.apache.camel.processor.idempotent.jpa.JpaMessageIdRepository.
3. 配置错误格式化宏： snippet: java.lang.IndexOutOfBoundsException: Index: 20, Size: 20
4. 将 idempotent 存储库配置为 org.apache.camel.processor.idempotent.jpa.JpaMessageIdRepository。
5. 在 Spring XML 文件中创建 JPA idempotent 存储库，如下所示：

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <route id="JpaMessageIdRepositoryTest">
    <from uri="direct:start" />
    <idempotentConsumer idempotentRepository="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，那么您可能会看到例外：

```
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:427)
    at
org.springframework.transaction.support.AbstractPlatformTransactionManager.getTransaction(Abstract
PlatformTransactionManager.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.junit5.CamelTestSupport.doSetUp(CamelTestSupport.java:238)
    at org.apache.camel.test.junit5.CamelTestSupport.setUp(CamelTestSupport.java:208)
```

这里的问题是源已通过 IDE 编译或重新编译，而不是通过 Maven 进行编译，这会在构建时增强字节代码。要克服这个问题，您需要启用 OpenzFCP 的动态字节代码增强。例如，假设 Camel 中使用的当前 Opensolutions 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
```

52.17. 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.jpa.aliases	将别名映射到 JPA 实体类。然后，别名可以在端点 URI （而不是完全限定类名称）中使用。		Map
camel.component.jpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.jpa.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.jpa.enabled	是否启用 jpa 组件的自动配置。这默认是启用的。		布尔值
camel.component.jpa.entity-manager-factory	使用 EntityManagerFactory。强烈建议您进行配置。选项是 javax.persistence.EntityManagerFactory 类型。		EntityManagerFactory
camel.component.jpa.join-transaction	camel-jpa 组件默认加入事务。您可以使用这个选项来关闭这个选项，例如，如果您使用 LOCAL_RESOURCE，并加入事务与 JPA 供应商无法工作。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.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.jpa.shared-entity-manager	是否将 Spring 的 SharedEntityManager 用于 consumer/producer。在大多数情况下，joinTransaction 应设置为 false，因为这不是 EXTENDED EntityManager。	false	布尔值
camel.component.jpa.transaction-manager	使用 PlatformTransactionManager 管理事务。选项是一个 org.springframework.transaction.PlatformTransactionManager 类型。		PlatformTransactionManager
camel.component.jpa.transaction-strategy	使用 TransactionStrategy 在事务中运行操作。选项是一个 org.apache.camel.component.jpa.TransactionStrategy 类型。		TransactionStrategy

第 53 章 JQ

自 Camel 3.18 起

Camel 支持 JQ 允许在 JSON 消息上使用 Expression 或 Predicate。

53.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 jq 时，请使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-jq-starter</artifactId>
</dependency>
```

53.2. JQ 选项

JQ 语言支持 4 个选项，如下所列。

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，它的优先级高于 propertyName （如果两者都已设置）。
propertyName		字符串	用作输入的属性名称，而不是消息正文。如果两者都被设置，则它比 headerName 低。
resultType		字符串	设置结果类型的类（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。

53.3. 例子

例如，您可以将 Predicate 中的 JQ 与基于内容的路由 EIP 搭配使用。

```
from("queue:books.new")
  .choice()
    .when().jq(".store.book.price < 10)")
```

```
        .to("jms:queue:book.cheap")
    .when().jq(".store.book.price < 30)")
        .to("jms:queue:book.average")
    .otherwise()
        .to("jms:queue:book.expensive");
```

53.4. 消息正文类型

Camel JQ 利用 camel-jackson 进行类型转换。要启用 camel-jackson POJO 类型转换，请参阅 Camel Jackson 文档。

53.5. 使用标头作为输入

默认情况下，JQ 使用消息正文作为输入源。但是，您还可以通过指定 headerName 选项使用标头作为输入。

例如，要计算存储在名为 book 的标头中的 JSON 文档中的 图书 数量，您可以执行以下操作：

```
from("direct:start")
    .setHeader("numberOfBooks")
        .jq(".store.books | length", int.class, "books")
    .to("mock:result");
```

53.6. CAMEL 提供的 JQ 功能

camel-jq 添加以下功能：

- header - 允许访问 JQ 表达式中的 Message 标头。

例如，使用 Message header 'MyHeader' 的值设置属性 foo ：

```
from("direct:start")
    .transform()
        .jq(".foo = header(\"MyHeader\")")
    .to("mock:result");
```

53.7. 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.language.jq .enabled	是否启用 jq 语言的自动配置。这默认是启用的。		布尔值
camel.language.jq .header-name	用作输入的标头名称，而不是消息正文，它的优先级高于 propertyName （如果两者都已设置）。		字符串
camel.language.jq .property-name	用作输入的属性名称，而不是消息正文。如果两者都被设置，则它比 headerName 低。		字符串
camel.language.jq .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

第 54 章 JSLT

Since Camel 3.1

仅支持生成者

JSLT 组件允许您使用 JSLT 表达式来处理 JSON 消息。在进行 JSON 转换或查询数据时，这是理想的选择。

54.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 jslt 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-jslt-starter</artifactId>
</dependency>
```

54.2. URI 格式

```
jslt:specName[?options]
```

其中 specName 是要调用的规格的 classpath-local URI，或远程规格的完整 URL （例如 file://folder/myfile.vm）。

54.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

54.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

54.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

54.4. 组件选项

JSLT 组件支持 5 个选项，如下所列。

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（默认为 false）。启用此选项后，可以通过消息标头指定动态模板。但是，如果标头来自恶意用户，则可以将其视为潜在的安全漏洞，因此请谨慎使用它。	false	布尔值

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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
functions (advanced)	JSLT 可以通过插入使用 Java 编写的功能来扩展。		集合
objectFilter (advanced)	JSLT 可以通过插入自定义 jslt 对象过滤器来扩展。		JsonFilter

54.4.1. 端点选项

JSLT 端点使用 URI 语法进行配置：

```
jslt:resourceUri
```

使用以下路径和查询参数：

54.4.1.1. 路径参数(1 参数)

Name	描述	默认值	类型
resourceUri (producer)	**所需资源** 的路径。您可以为前缀：classpath, file, http, ref, 或 bean. classpath, file 和 http 使用这些协议加载资源(classpath 为 default)。ref 将查询 registry 中的资源。bean 将调用要用作资源的 bean 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

54.4.1.2. 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细信息。默认情况下，只能访问消息正文和标头。这个选项可以启用对当前 Exchange 和 CamelContext 的完整访问权限。这样做会造成潜在的安全风险，因为这将打开对 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（默认为 false）。启用此选项后，可以通过消息标头指定动态模板。但是，如果标头来自恶意用户，则可以将其视为潜在的安全漏洞，因此请谨慎使用它。	false	布尔值
contentCache (producer)	设置是否使用资源内容缓存。	false	布尔值
mapBigDecimalAsFloats (producer)	如果为 true，则映射程序将在序列化功能中使用 USE_BIG_DECIMAL_FOR_FLOATS。	false	布尔值
ObjectMapper (producer)	设置自定义 JSON 对象映射程序。		ObjectMapper
prettyPrint (common)	如果为 true，则输出消息中的 JSON 会输出为用户。	false	布尔值
lazyStartProducer (producer (advanced))	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

54.5. 消息标头

JSLT 组件支持 2 个消息标头，如下所列：

Name	描述	默认值	类型
CamelJsltString (producer) 常量 ： HEADER_JSLT_STRING	JSLT 模板作为 String。		字符串

Name	描述	默认值	类型
CamelJsltResourceUri (producer) 常量 ：HEADER_JSLT_RESOURCE_URI	资源 URI。		字符串

54.6. 将值传递给 JSLT

在对正文应用 JSLT 表达式时，Camel 可以提供交换信息作为变量。Exchange 中的可用变量有：

name	value
标头	**In** 消息的标头作为 json 对象
exchange.properties	**Exchange** 属性作为 json 对象。Exchange 是变量的名称，*属性是* 交换属性的路径。**如果 allowContextMapAll** 选项为 **true，**则可用。

所有无法使用 Jackson 转换为 json 的值都被拒绝，且不会在 jslt 表达式中提供。

例如，可以访问名为"type"和交换属性"instance"的标头，如下所示

```
{
  "type": $headers.type,
  "instance": $exchange.properties.instance
}
```

54.7. SAMPLES

示例示例如下。

```
from("activemq:My.Queue").
  to("jslt:com/acme/MyResponse.json");
```

以及基于文件的资源：

```
from("activemq:My.Queue").
  to("jslt:file://myfolder/MyResponse.json?contentCache=true").
  to("activemq:Another.Queue");
```

您还可以指定组件应通过标头使用哪个 JSLT 表达式，例如：

```
from("direct:in").
  setHeader("CamelJsltResourceUri").constant("path/to/my/spec.json").
  to("jslt:dummy?allowTemplateFromHeader=true");
```

或通过标头发送整个 jslt 表达式：（可用于查询）

```
from("direct:in").
  setHeader("CamelJsltString").constant(".published").
  to("jslt:dummy?allowTemplateFromHeader=true");
```

将交换属性传递给 jslt 表达式可以如下所示

```
from("direct:in").
  to("jslt:com/acme/MyResponse.json?allowContextMapAll=true");
```

54.8. 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.jslt.allow-template-from-header	是否允许使用来自标头的资源模板（默认为 false）。启用此选项后，可以通过消息标头指定动态模板。但是，如果标头来自恶意用户，则可以将其视为潜在的安全漏洞，因此请谨慎使用它。	false	布尔值
camel.component.jslt.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.jslt.enabled	是否启用 jslt 组件的自动配置。这默认是启用的。		布尔值
camel.component.jslt.functions	JSLT 可以通过插入使用 Java 编写的功能来扩展。		集合

Name	描述	默认值	类型
camel.component.jslt.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.jslt.object-filter	JSLT 可以通过插入自定义 jslt 对象过滤器来扩展。选项是一个 com.schibsted.spt.data.jslt.filters.JsonFilter 类型。		JsonFilter

第 55 章 JSON GSON

gson 是使用 Gson 库的数据格式。

```
from("activemq:My.Queue").
  marshal().json(JsonLibrary.Gson).
  to("mqseries:Another.Queue");
```

55.1. 依赖项

当在 Camel Spring Boot 中使用 json-gson 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-gson-starter</artifactId>
</dependency>
```

55.2. GSON 选项

JSON Gson dataformat 支持 3 个选项，如下所列。

Name	默认值	Java 类型	描述
prettyPrint		布尔值	要启用用户化的打印输出，请执行以下操作：默认为 false。
unmarshalType		字符串	当 unmarshalling 时使用的 java 类型的类名称。
contentTypeHeader		布尔值	数据格式是否应该使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式到 XML 的 application/xml 或用于数据格式的 application/json 发送到 JSON。

55.3. 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.dataformat.json-gson.content-type-header	数据格式是否应该使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式到 XML 的 application/xml 或用于数据格式的 application/json 发送到 JSON。	true	布尔值
camel.dataformat.json-gson.enabled	是否启用 json-gson 数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.json-gson.pretty-print	要启用用户化的打印输出，请执行以下操作：默认为 false。	false	布尔值
camel.dataformat.json-gson.unmarshal-type	当 unmarshalling 时使用的 java 类型的类名称。		字符串

第 56 章 JSON JACKSON

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");
```

56.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 json-jackson 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-jackson-starter</artifactId>
</dependency>
```

56.2. JACKSON 选项

JSON Jackson dataformat 支持 20 个选项，如下所列。

Name	默认值	Java 类型	描述
objectMapper		字符串	使用 Jackson 时，查找并使用带有给定 id 的现有 ObjectMapper。
useDefaultObjectMapper		布尔值	是否从 registry 中查找和使用默认 Jackson ObjectMapper。
prettyPrint		布尔值	要启用用户化的打印输出，请执行以下操作：默认为 false。
unmarshalType		字符串	当 unmarshalling 时使用的 java 类型的类名称。

Name	默认值	Java 类型	描述
jsonView		字符串	当 marshalling a POJO to JSON 时，您可能想要从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现此目的。此选项是引用具有 JsonView 注释的类。
Include		字符串	如果您想 marshal a pojo to JSON，并且 pojo 具有一些带有 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		布尔值	用于 JMS 用户，以允许 JMS spec 中的 JMSType 标头指定一个 FQN 类名称来用于 unmarshal。
collectionType		字符串	引用要使用的自定义集合类型，以便在 registry 中查找。这个选项应该很少被使用，但允许使用与基于 java.util.Collection 不同的集合类型。
useList		布尔值	To unmarshal 到 Map 列表或 Pojo 的列表。

Name	默认值	Java 类型	描述
moduleClassNames		字符串	使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String with FQN 类名称。可以使用逗号分隔多个类。
moduleRefs		字符串	使用 Camel registry 中引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。
enableFeatures		字符串	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上启用的功能集合。这个功能应该是与 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature multiple features 分开的名称。

Name	默认值	Java 类型	描述
disableFeatures		字符串	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的功能集合。这个功能应该是与 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature multiple features 分开的名称。
allowUnmarshallType		布尔值	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。这只有在需要使用时才启用。
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 没有影响，如 gson、fastjson 和 xstream。
autoDiscoverObjectMapper		布尔值	如果设置为 true，则 Jackson 将把 objectMapper 来查找到 registry 中。

Name	默认值	Java 类型	描述
contentTypeHeader		布尔值	数据格式是否应该使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式到 XML 的 application/xml 或用于数据格式的 application/json 发送到 JSON。
schemaResolver		字符串	可选的模式解析器用于查找传输中数据的模式。
autoDiscoverSchemaResolver		布尔值	如果没有禁用，SchemaResolver 将查找到 registry 中。
namingStrategy		字符串	如果设置随后，Jackson 将使用定义的 PropertyNamingStrategy.Possible 值：LOWER_CAMEL_CASE、LOWER_DOT_CASE、LOWER_CASE、KEBAB_CASE、SNAKE_CASE 和 UPPER_CAMEL_CASE。

56.3. 使用自定义 OBJECTMAPPER

如果需要更多地控制映射配置，您可以将 JacksonDataFormat 配置为使用自定义 ObjectMapper。

如果您在 registry 中设置单个 ObjectMapper，则 Camel 将自动查找并使用此 ObjectMapper。例如，如果您使用 Spring Boot，如果启用了 Spring MVC，Spring Boot 可以为您提供默认的 ObjectMapper。这将允许 Camel 检测 Spring Boot bean registry 中是否有 ObjectMapper 类类型，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。

56.4. 使用 JACKSON 进行自动类型转换

camel-jackson 模块允许将 Jackson 集成为 Type Converter。这与 JAXB 类似，其与 Camel 类型转换器集成。

要使用此 camel-jackson，必须在 CamelContext 全局选项上设置以下选项来实现，如下所示：

```
@Bean
CamelContextConfiguration contextConfiguration() {
   return new CamelContextConfiguration() {
      @Override
      public void beforeApplicationStart(CamelContext context) {
      // Enable Jackson JSON type converter.
         context.getGlobalOptions().put(JacksonConstants.ENABLE_TYPE_CONVERTER,
"true");
      // Allow Jackson JSON to convert to pojo types also
      // (by default Jackson only converts to String and other simple types)
         getContext().getGlobalOptions().put(JacksonConstants.TYPE_CONVERTER_TO_POJO,
"true");
      }

      @Override
      public void afterApplicationStart(CamelContext camelContext) {

      }
   };
}
```

camel-jackson 类型转换器与 JAXB 集成，这意味着您可以将 POJO 类标注为 Jackson 可以使用的 JAXB 注释。您还可以在 POJO 类上使用 Jackson 自己的注解。

56.5. 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.dataformat.json-jackson.allow-jms-type	用于 JMS 用户，以允许 JMS spec 中的 JMSType 标头指定一个 FQN 类名称来用于 unmarshal。	false	布尔值

Name	描述	默认值	类型
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。这只有在需要使用时才启用。	false	布尔值
camel.dataformat.json-jackson.auto-discover-object-mapper	如果设置为 true，则 Jackson 将把 objectMapper 来查找到 registry 中。	false	布尔值
camel.dataformat.json-jackson.auto-discover-schema-resolver	如果没有禁用，SchemaResolver 将查找到 registry 中。	true	布尔值
camel.dataformat.json-jackson.collection-type	引用要使用的自定义集合类型，以便在 registry 中查找。这个选项应该很少被使用，但允许使用与基于 java.util.Collection 不同的集合类型。		字符串
camel.dataformat.json-jackson.content-type-header	数据格式是否应该使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式到 XML 的 application/xml 或用于数据格式的 application/json 发送到 JSON。	true	布尔值
camel.dataformat.json-jackson.disable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的功能集合。这个功能应该是与 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature multiple features 分开的名称。		字符串
camel.dataformat.json-jackson.enable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上启用的功能集合。这个功能应该是与 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature multiple features 分开的名称。		字符串
camel.dataformat.json-jackson.enabled	是否启用 json-jackson 数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.json-jackson.include	如果您想 marshal a pojo to JSON，并且 pojo 具有一些带有 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.json-jackson.json-view	当 marshalling a POJO to JSON 时，您可能想要从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现此目的。此选项是引用具有 JsonView 注释的类。		字符串
camel.dataformat.json-jackson.module-class-names	使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String with 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 中引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。		字符串
camel.dataformat.json-jackson.naming-strategy	如果设置随后，Jackson 将使用定义的 Property Naming Strategy.Possible 值：LOWER_CAMEL_CASE、LOWER_DOT_CASE、LOWER_CASE、KEBAB_CASE、SNAKE_CASE 和 UPPER_CAMEL_CASE。		字符串
camel.dataformat.json-jackson.object-mapper	使用 Jackson 时，查找并使用带有给定 id 的现有 ObjectMapper。		字符串
camel.dataformat.json-jackson.pretty-print	要启用用户化的打印输出，请执行以下操作：默认为 false。	false	布尔值
camel.dataformat.json-jackson.schema-resolver	可选的模式解析器用于查找传输中数据的模式。		字符串
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 没有影响，如 gson、fastjson 和 xstream。		字符串
camel.dataformat.json-jackson.unmarshal-type	当 unmarshalling 时使用的 java 类型的类名称。		字符串

Name	描述	默认值	类型
camel.dataformat.json-jackson.use-default-object-mapper	是否从 registry 中查找和使用默认 Jackson ObjectMapper。	true	布尔值
camel.dataformat.json-jackson.use-list	To unmarshal 到 Map 列表或 Pojo 的列表。	false	布尔值

第 57 章 JSONPATH

Camel 支持 JSONPath 允许在 JSON 消息上使用 Expression 或 Predicate。

57.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 jsonpath 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-jsonpath-starter</artifactId>
</dependency>
```

57.2. JSONPATH 选项

JSONPath 语言支持 8 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		布尔值	是否阻止异常，如 PathNotFoundException。
allowSimple		布尔值	是否在 JSONPath 表达式中允许内联简单异常。
allowEasyPredicate		布尔值	是否允许使用 easy predicate 解析器来预解析 predicates。
writeAsString		布尔值	是否将每行/元素的输出写入为 JSON String 值，而不是映射/POJO 值。
headerName		字符串	作为输入的标头名称，而不是消息的正文。

Name	默认值	Java 类型	描述
选项		**Enum**	在 JSONPath 上配置附加选项。可以使用逗号分隔多个值。 Enum 值： • DEFAULT_PATH_LEAF_TO_NULL • ALWAYS_RETURN_LIST • AS_PATH_LIST • SUPPRESS_EXCEPTIONS • REQUIRE_PROPERTIES
trim		**布尔值**	是否修剪值以移除前导和结尾的空格和换行符。

57.3. 例子

例如，您可以使用带有基于内容的路由 EIP 的 Predicate 中的 JSONPath。

```
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");
```

在 XML DSL 中：

```
<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>
```

57.4. JSONPATH SYNTAX

使用 JSONPath 语法需要一些时间才能学习，即使基本 predicates 也是如此。例如，要查找您需要做的所有成本图书：

```
$.store.book[?(@.price < 20)]
```

57.4.1. 简单 JSONPath 语法

但是，如果您只将其写为：

```
store.book.price < 20
```

如果您只想使用价格键查看节点，可以省略该路径：

```
price < 20
```

要提供支持，有一个 EasyPredicateParser，如果您在使用基本风格定义了 predicate 时启动它。这意味着 predicate 不得以 $ 符号开头，且仅包含一个操作器。

简单语法是：

```
left OP right
```

您可以在右运算符中使用 Camel 简单语言，例如：

```
store.book.price < ${header.limit}
```

有关更多语法示例，请参阅 JSONPath 项目页面。

57.5. 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	从文件读取
字符串	普通字符串
Map	消息正文作为 **java.util.Map** 类型
list	消息正文作为 **java.util.List** 类型
POJO	**可选**，如果 Jackson 位于 classpath 上，则 camel-jsonpath 可以使用 Jackson 将消息正文读取为 POJO，并转换为 JSonPath 支持的 **java.util.Map**。例如，您可以添加 **camel-jackson** 作为依赖项，使其包含 Jackson。
InputStream	如果上述类型都不匹配，则 Camel 将尝试以 **java.io.InputStream** 形式读取消息正文。

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 配置为禁止异常（请参阅以下）

57.6. 抑制例外

默认情况下，如果 json 有效负载没有与配置的 jsonpath 表达式对应的有效路径，则 jsonpath 将抛出异常。在某些用例中，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>
```

此选项也可以在 @JsonPath 注释上提供。

57.7. 内联简单表达式

可以使用简单语法 ${xxx}，在 JSONPath 表达式中内联 Simple 语言。

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 &lt; ${header.cheap})]</jsonpath>
      <to uri="mock:cheap"/>
    </when>
    <when>
      <jsonpath>$.store.book[?(@.price &lt; ${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 &lt; 10)]</jsonpath>
```

57.8. JSONPATH 注入

您可以使用 Bean 集成 在 bean 上调用方法，并使用各种语言，如 JSONPath （通过 @JsonPath 注释）从消息中提取值并将其绑定到 method 参数，如下所示：

```
public class Foo {

    @Consume("activemq:queue:books.new")
    public void doSomething(@JsonPath("$.store.book[*].author") String author, @Body String
json) {
      // process the inbound message here
    }
}
```

57.9. 编码检测

如果文档采用 unicode (UTF-8, UTF-16LE, UTF-16BE, UTF-32LE, UTF-32LE, UTF-32BE)，则会自动检测到 JSON 文档的编码，如 RFC-4627 中指定的。如果编码是非 unicode 编码，您可以确保以 String 格式输入文档到 JSONPath，或者您可以在标头 CamelJsonPathJsonEncoding 中指定编码，该编码定义为：JsonpathConstants.HEADER_JSON_ENCODING。

57.10. 以 JSON 用户身份将 JSON 数据分成子行

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，会记录每个图书，但消息正文都是 Map 实例。有时，您可能希望将其输出为普通 String JSON 值，这可使用 writeAsString 选项完成，如下所示：

```
from("direct:start")
    .split().jsonpathWriteAsString("$.store.book[*]")
    .to("log:book");
```

然后，每个书都记录为 String JSON 值。

57.11. 使用标头作为输入

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 选项使用标头作为输入。

例如，要计算存储在名为 book 的标头中的 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，我们将标题名称指定为 books，我们还要告知我们希望通过 int.class 将结果转换为整数。

XML DSL 中的示例相同：

```
<route>
  <from uri="direct:start"/>
  <setHeader name="numberOfBooks">
    <jsonpath headerName="books" resultType="int">$..store.book.length()</jsonpath>
  </setHeader>
  <to uri="mock:result"/>
</route>
```

57.12. 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.language.jsonpath.allow-easy-predicate	是否允许使用 easy predicate 解析器来预解析 predicates。	true	布尔值

Name	描述	默认值	类型
camel.language.jsonpath.allow-simple	是否在 JSONPath 表达式中允许内联简单异常。	true	布尔值
camel.language.jsonpath.enabled	是否启用 jsonpath 语言的自动配置。这默认是启用的。		布尔值
camel.language.jsonpath.header-name	作为输入的标头名称，而不是消息的正文。		字符串
camel.language.jsonpath.option	在 JSONPath 上配置附加选项。可以使用逗号分隔多个值。		字符串
camel.language.jsonpath.suppress-exceptions	是否阻止异常，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	是否将每行/元素的输出写入为 JSON String 值，而不是映射/POJO 值。	false	布尔值

第 58 章 KAFKA

支持生成者和消费者

Kafka 组件用于与 Apache Kafka 消息代理通信。

58.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 kafka 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-kafka-starter</artifactId>
</dependency>
```

58.2. URI 格式

```
kafka:topic[?options]
```

58.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

58.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

58.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

58.4. 组件选项

Kafka 组件支持 104 选项，如下所列。

Name	描述	默认值	类型
additionalProperties (common)	如果无法直接在 camel 配置上设置 kafka consumer 或 kafka producer 的额外属性（例如：在 Camel 配置中还没有反映的新 Kafka 属性），属性必须加上 additionalProperties 前缀。例如：additionalProperties.transactional.id=12345&additionalProperties.schema.registry.url=http://localhost:8811/avro。		Map
brokers (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或指向代理子集的 VIP。这个选项在 Kafka 文档中称为 bootstrap.servers。		字符串
clientId (common)	客户端 ID 是每个请求中发送的用户指定字符串，以帮助追踪调用。它应该逻辑地标识发出请求的应用程序。		字符串
configuration (common)	允许使用端点将重复使用的通用选项预配置 Kafka 组件。		KafkaConfiguration

Name	描述	默认值	类型
headerFilterStrategy (common)	使用自定义 HeaderFilterStrategy 过滤标头到 Camel 消息。		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
shutdownTimeout (common)	以毫秒为单位，等待消费者或制作者正常关闭并终止其 worker 线程。	30000	int
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 消息标头中，它允许最终用户访问此 API 并通过 Kafka 消费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。当进程失败作为新消费者开始的位置时，将使用此提交的偏移。	true	布尔值
autoCommitIntervalMs (consumer)	消费者偏移提交至 zookeeper 的频率(ms)。	5000	整数
autoCommitOnStop (consumer)	是否在消费者停止时执行显式自动提交，以确保代理从最后使用的消息中有提交。这要求打开了选项 autoCommitEnable。可能的值有： sync、sync 或 none。sync 是默认值。 Enum 值： • 同步 • async • none	同步	字符串
autoOffsetReset (consumer)	当 ZooKeeper 中没有初始偏移时，或偏移没有范围：earliest : automatically the offset to the earliest offset : automatically the offset to the latest offset failed: throw exception to the consumer. Enum 值： • latest • earliest • none	latest	字符串

Name	描述	默认值	类型
breakOnFirstError (consumer)	此选项控制消费者处理交换时会发生什么，并且失败。如果选项为 false，则消费者将继续进入下一个消息并进行处理。如果选项为 true，则消费者将显示为导致失败的消息偏移，然后重新尝试处理此消息。但是，如果绑定到每次失败，这可能会导致正常处理同一消息的处理，例如 poison 消息。因此，建议您使用 Camel 的错误处理程序来应对这种情况。	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
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
commitTimeoutMs (consumer)	代码等待同步提交完成的最长时间（以毫秒为单位）。	5000	Long
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 服务器的使用者数量。每个消费者都在单独的线程上运行，用于检索和处理传入的数据。	1	int
fetchMaxBytes (consumer)	如果获取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为 fetch 请求。这不是绝对的，如果 fetch 的第一个非空分区中的信息大于这个值，则消息仍会返回，以确保消费者能够进行进度。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立即满足 fetch.min.bytes，则服务器在回答获取请求前将阻断的最大时间。	500	整数

Name	描述	默认值	类型
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
groupInstanceId (consumer)	最终用户提供的消费者实例的唯一标识符。只允许非空字符串。如果设置，消费者被视为静态成员，这意味着任何时候都只允许具有此 ID 的一个实例。这可与更大的会话超时结合使用，以避免由临时不可用造成的组重新平衡（如进程重启）。如果没有设置，消费者将加入组作为动态成员，这是传统行为。		字符串
headerDeserializer (consumer)	使用自定义 KafkaHeaderDeserializer 来反序列化 kafka 标头值。		KafkaHeaderDeserializer
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，但通常不应设置高于该值的 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
keyDeserializer (consumer)	deserializer 类用于实现 Deserializer 接口的密钥。	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一样大，否则制作者可能会发送大于消费者可以获取的消息。如果发生这种情况，消费者可能会卡住在某个分区中获取大量消息。	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)	单个调用返回到 poll （）中返回的最大记录数。	500	整数
offsetRepository (consumer)	用来本地存储主题每个分区的偏移量的偏移存储库。定义将禁用 autocommit。		StateRepository

Name	描述	默认值	类型
partitionAssignor (consumer)	使用组管理时，客户端将使用的类名称在消费者实例之间分发分区所有权。	org.apache.kafka.clients.consumer.RangeAssignor	字符串
pollOnError (consumer)	如果 kafka threw 异常轮询新消息，则该怎么办。默认情况下，将使用来自组件配置的值，除非在端点级别上配置了显式值。DISCARD 将丢弃消息并继续轮询下一个消息。ERROR_HANDLER 将使用 Camel 的错误处理程序来处理异常，之后继续轮询下一个消息。RECONNECT 将重新连接消费者，并尝试再次轮询消息 RETRY 将使消费者重试轮询同一消息，而 STOP 将停止消费者（如果消费者应该再次消耗消息，则必须手动启动/重新启动）。 Enum 值： • DISCARD • ERROR_HANDLER • RECONNECT • RETRY • STOP	ERROR_HANDLER	PollOnError
pollTimeoutMs (consumer)	轮询 KafkaConsumer 时使用的超时。	5000	Long
resumeStrategy (consumer)	此选项允许用户设置自定义恢复策略。当分配了分区时（例如：连接或重新连接时），执行恢复策略。它允许实施自定义如何恢复操作，并作为 seekTo 和 offsetRepository 机制的更灵活的替代方法。有关实现详情，请参阅 KafkaConsumerResumeStrategy。此选项不会影响自动提交设置。使用此设置的实现可能还希望通过手动提交选项来评估。		KafkaConsumerResumeStrategy
seekTo (consumer)	设置 KafkaConsumer 是否可以从启动时读取或结束：从开始 : read from end : read from end This is replace the earlier property seekToBeginning。 Enum 值： • 开始 • end		字符串

Name	描述	默认值	类型
sessionTimeoutMs (consumer)	使用 Kafka 的组管理功能时检测失败的超时。	10000	整数
specificAvroReader (consumer)	这可让使用特定的 Avro reader 与 Confluent Platform 模式 registry 和 io.confluent.kafka.serializers.KafkaAvroDeserializer 一起使用。这个选项仅适用于 Confluent Platform （非标准 Apache Kafka）。	false	布尔值
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动态主题数量。	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
kafkaManualCommitFactory (consumer (advanced))	**autowired** Factory 用于创建 KafkaManualCommit 实例。这允许在手动提交来自开箱即用的默认实现时，自定义工厂创建自定义 KafkaManualCommit 实例。		KafkaManualCommitFactory
pollExceptionStrategy (consumer (advanced))	**Autowired** 使用带有消费者的自定义策略来控制如何在池消息时处理 Kafka 代理中引发的异常。		PollExceptionStrategy
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。如果发送记录的速度比生成者可以更快地发送到服务器，则根据 block.on.buffer.full 指定的首选项阻止或抛出异常。此设置应该与生成者使用的总内存对应，但不是硬绑定，因为生成者都用于缓冲区。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

Name	描述	默认值	类型
compressionCodec (producer)	此参数允许您为此制作者生成的所有数据指定压缩代码c。有效值为 none、gzip 和 snappy。 Enum 值： • none • gzip • snappy • lz4	none	字符串
connectionMaxIdleMs (producer)	在这个配置指定的毫秒数后关闭闲置连接。	540000	整数
deliveryTimeoutMs (producer)	在调用 send （）返回后报告成功或失败的上限。这限制了发送前记录总时间、从代理等待确认时间（如果预期），以及可重新发送失败的时间。	120000	整数
enableIdempotence (producer)	如果设置为 'true'，则制作者将确保每个消息的确切副本写入流。如果 'false'，则制作者重试可能会在流中写入重试消息的副本。如果设置为 true，则此选项将需要 max.in.flight.requests.per.connection 设置为 1，重试不能为零，另外 acks 必须设置为 'all'。	false	布尔值
headerSerializer (producer)	使用自定义 KafkaHeaderSerializer 来序列化 kafka 标头值。		KafkaHeaderSerializer
key (producer)	记录密钥（如果没有指定密钥，则为 null）。如果配置了这个选项，则它优先于标头 KafkaConstants#KEY。		字符串
keySerializer (producer)	密钥的序列化器类（如果没有提供，默认为与消息相同）。	org.apache.kafka.common.serialization.StringSerializer	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

Name	描述	默认值	类型
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人工延迟来实现此目的，即不立即发送记录，而不是立即发送记录，让生产者最多等待给定延迟，以允许发送其他记录，以便可将发送批处理在一起。这可能被视为与 TCP 中的 Nagle 算法类似。此设置提供批处理延迟的上限：一旦我们获得批处理。无论此设置如何，无论此设置如何，它都会立即发送一次记录，但是如果我们为此分区累积了这个字节，我们将"linger"用于等待更多记录显示。此设置默认为 0（例如，无延迟）。例如，设置 linger.ms=5 效果会减少发送的请求数量，但会向负载中发送的记录添加最多 5ms 的延迟。	0	整数
maxBlockMs (producer)	配置控制发送到 kafka 将阻止的时长。出于多种原因，这些方法可以被阻止。例如：缓冲区满，元数据不可用。此配置对获取元数据、密钥和值、执行 send （）时缓冲内存所花费的总时间实施最大限制。如果是 partitionsFor （），此配置会在等待元数据时实施最长时间阈值。	60000	整数
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，如果将此设置设置为大于 1，且发送失败，则可能会因为重试而重新排序消息的风险（例如，如果启用了重试）。	5	整数
maxRequestSize (producer)	请求的最大大小。这也实际上是最大记录大小的上限。请注意，服务器在记录大小上具有自己的上限，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。通过实施 MetricReporter 接口，可以插入将收到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

Name	描述	默认值	类型
partitioner (producer)	用于分区子主题中分区消息的分区类。默认分区器基于密钥的哈希。	org.apache.kafka.clients.producer.internals.DefaultPartitioner	字符串
PartitionKey (producer)	将发送记录的分区（如果未指定分区，则为 null）。如果配置了这个选项，则它优先于标头 KafkaConstants#PARTITION_KEY。		整数
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）。不会尝试批处理记录大于这个大小。发送到代理的Requests 将包含多个批处理，每个分区都可用于发送数据。小批处理大小会减少批处理，并可能会减少吞吐量（零的批处理大小将完全禁用批处理）。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 模式时可以放入生产者的最大未发送消息数，然后才能阻止生产者或必须丢弃数据。	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等待的时间。这可避免在紧密循环中重复连接到主机。此 backoff 应用到消费者发送到代理的所有请求。	50	整数
recordMetadata (producer)	producer 是否应该存储来自发送到 Kafka 的 RecordMetadata 结果。结果存储在包含 RecordMetadata 元数据的列表中。该列表存储在一个带有键 KafkaConstants#KAFKA_RECORDMETA 的标头中。	true	布尔值

Name	描述	默认值	类型
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。以下设置很常见：acks=0 如果设为零，则生成者不会等待来自服务器的任何确认。记录将立即添加到套接字缓冲区中并被视为发送。无法保证服务器已收到记录，重试配置不会生效（因为客户端通常不知道任何失败）。对每个记录给出的偏移始终设置为 -1 acks=1，这意味着领导机将记录写入其本地日志，但不会等待所有后续者完全确认。在这种情况下，领导机会在确认记录后立即失败，但在后续者复制之前，记录将会丢失。acks=all 意味着领导机将等待整个同步副本集合来确认记录。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。 Enum 值： ● -1 ● 0 ● 1 ● all	1	字符串
requestTimeoutMs (producer)	在将一个错误发送到客户端前，代理会等待尝试满足 request.required.acks 要求的时间。	30000	整数
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，因为如果两个记录发送到单个分区，则第一次失败并且被重试，但第二个记录会先出现，然后是第二个记录。	0	整数
retryBackoffMs (producer)	每次重试前，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，因此此属性指定生成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
valueSerializer (producer)	serializer 类用于消息。	org.apache.kafka.common.serialization.StringSerializer	字符串

Name	描述	默认值	类型
workerpool (producer)	要在 kafka 服务器确认从 KafkaProducer 发送的消息使用异步非阻塞处理，使用自定义 worker 池继续路由交换。如果使用这个选项，您必须处理线程池的生命周期，以便在不再需要时关闭池。		ExecutorService
workerPoolCoreSize (producer)	kafka 服务器在 kafka 服务器确认从 KafkaProducer 发送的消息（使用异步非阻塞处理）来继续路由交换的 worker 池的核心线程数量。	10	整数
workerPoolMaxSize (producer)	kafka 服务器之后，worker 池的最大线程数量用于继续路由交换，使用异步非块处理确认从 KafkaProducer 发送的消息。	20	整数
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
kafkaClientFactory (advanced)	**autowired** Factory 用于创建 org.apache.kafka.clients.consumer.KafkaConsumer 和 org.apache.kafka.clients.producer.KafkaProducer 实例。这允许配置自定义工厂来创建带有扩展 vanilla Kafka 客户端的逻辑的实例。		KafkaClientFactory
同步 (advanced)	设置是否应严格使用同步处理。	false	布尔值
schemaRegistryURL (confluent)	要使用的 Confluent Platform 模式 registry 服务器的 URL。格式为 host1:port1,host2:port2。这在 Confluent Platform 文档中被称为 schema.registry.url。这个选项仅适用于 Confluent Platform （非标准 Apache Kafka）。		字符串
interceptorClasses (monitoring)	为制作者或消费者设置拦截器。制作者拦截器必须是实施 org.apache.kafka.clients.producer.ProducerInterceptor 拦截器的类，必须是实施 org.apache.kafka.clients.consumer.ConsumerInterceptor 的类。如果您对消费者使用 Producer 拦截器，它将在运行时抛出类 cast 异常。		字符串
kerberosBeforeReloginMinTime (security)	刷新尝试之间的登录线程睡眠时间。	60000	整数
kerberosInitCmd (security)	Kerberos kinit 命令路径。默认为 /usr/bin/kinit。	/usr/bin/kinit	字符串

Name	描述	默认值	类型
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，形式 {username}/{hostname}{REALM} 的主体名称映射到 {username}。有关格式的详情，请查看安全授权和 acls 文档。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。有关有效值，请参阅。	GSSAPI	字符串
securityProtocol (security)	用于与代理通信的协议。支持 SASL_PLAINTEXT, 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。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。如果配置它，它会在其他 SSL 端点参数之前应用。注意：Kafka 只支持从文件位置加载密钥存储，因此请在 KeyStoreParameters.resource 选项中使用 file: 前缀。		SSLContextParameters
sslEnabledProtocols (security)	为 SSL 连接启用的协议列表。TLSv1.2、TLSv1.1 和 TLSv1 默认启用。		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。	https	字符串
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

Name	描述	默认值	类型
sslKeyPassword (security)	密钥存储文件中的私钥密码。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。对于客户端，这是可选的，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，这适用于大多数情况。最近的 JVM 中允许的值为 TLS、TLSv1.1 和 TLSv1.2。较旧的 JVM 中可能支持 SSL、SSLv2 和 SSLv3，但由于已知的安全漏洞，不建议使用它们。		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
useGlobalSslContextParameters (security)	启用使用全局 SSL 上下文参数。	false	布尔值

58.5. 端点选项

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：

58.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
topic (common)	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。制作者只能发送消息到单个主题。		字符串

58.5.2. 查询参数(102 参数)

Name	描述	默认值	类型
additionalProperties (common)	如果无法直接在 camel 配置上设置 kafka consumer 或 kafka producer 的额外属性（例如：在 Camel 配置中还没有反映的新 Kafka 属性），属性必须加上 additionalProperties 前缀。例如：additionalProperties.transactional.id=12345&additionalProperties.schema.registry.url=http://localhost:8811/avro。		Map
brokers (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或指向代理子集的 VIP。这个选项在 Kafka 文档中称为 bootstrap.servers。		字符串
clientId (common)	客户端 ID 是每个请求中发送的用户指定字符串，以帮助追踪调用。它应该逻辑地标识发出请求的应用程序。		字符串
headerFilterStrategy (common)	使用自定义 HeaderFilterStrategy 过滤标头到 Camel 消息。		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
shutdownTimeout (common)	以毫秒为单位，等待消费者或制作者正常关闭并终止其 worker 线程。	30000	int
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 消息标头中，它允许最终用户访问此 API 并通过 Kafka 消费者执行手动偏移提交。	false	布尔值

Name	描述	默认值	类型
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。当进程失败作为新消费者开始的位置时，将使用此提交的偏移。	true	布尔值
autoCommitIntervalMs (consumer)	消费者偏移提交至 zookeeper 的频率(ms)。	5000	整数
autoCommitOnStop (consumer)	是否在消费者停止时执行显式自动提交，以确保代理从最后使用的消息中有提交。这要求打开了选项 autoCommitEnable。可能的值有： sync、sync 或 none。sync 是默认值。 Enum 值： • 同步 • async • none	同步	字符串
autoOffsetReset (consumer)	当 ZooKeeper 中没有初始偏移时，或偏移没有范围：earliest : automatically the offset to the earliest offset : automatically the offset to the latest offset failed: throw exception to the consumer. Enum 值： • latest • earliest • none	latest	字符串
breakOnFirstError (consumer)	此选项控制消费者处理交换时会发生什么，并且失败。如果选项为 false，则消费者将继续进入下一个消息并进行处理。如果选项为 true，则消费者将显示为导致失败的消息偏移，然后重新尝试处理此消息。但是，如果绑定到每次失败，这可能会导致正常处理同一消息的处理，例如 poison 消息。因此，建议您使用 Camel 的错误处理程序来应对这种情况。	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
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
commitTimeoutMs (consumer)	代码等待同步提交完成的最长时间（以毫秒为单位）。	5000	Long
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 服务器的使用者数量。每个消费者都在单独的线程上运行，用于检索和处理传入的数据。	1	int
fetchMaxBytes (consumer)	如果获取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为 fetch 请求。这不是绝对的，如果 fetch 的第一个非空分区中的信息大于这个值，则消息仍会返回，以确保消费者能够进行进度。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立即满足 fetch.min.bytes，则服务器在回答获取请求前将阻断的最大时间。	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
groupInstanceId (consumer)	最终用户提供的消费者实例的唯一标识符。只允许非空字符串。如果设置，消费者被视为静态成员，这意味着任何时候都只允许具有此 ID 的一个实例。这可与更大的会话超时结合使用，以避免由临时不可用造成的组重新平衡（如进程重启）。如果没有设置，消费者将加入组作为动态成员，这是传统行为。		字符串
headerDeserializer (consumer)	使用自定义 KafkaHeaderDeserializer 来反序列化 kafka 标头值。		KafkaHeaderDeserializer

Name	描述	默认值	类型
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，但通常不应设置高于该值的 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
keyDeserializer (consumer)	deserializer 类用于实现 Deserializer 接口的密钥。	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一样大，否则制作者可能会发送大于消费者可以获取的消息。如果发生这种情况，消费者可能会卡住在某个分区中获取大量消息。	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)	单个调用返回到 poll （）中返回的最大记录数。	500	整数
offsetRepository (consumer)	用来本地存储主题每个分区的偏移量的偏移存储库。定义将禁用 autocommit。		StateRepository
partitionAssignor (consumer)	使用组管理时，客户端将使用的类名称在消费者实例之间分发分区所有权。	org.apache.kafka.clients.consumer.RangeAssignor	字符串

Name	描述	默认值	类型
pollOnError (consumer)	如果 kafka threw 异常轮询新消息，则该怎么办。默认情况下，将使用来自组件配置的值，除非在端点级别上配置了显式值。DISCARD 将丢弃消息并继续轮询下一个消息。ERROR_HANDLER 将使用 Camel 的错误处理程序来处理异常，之后继续轮询下一个消息。RECONNECT 将重新连接消费者，并尝试再次轮询消息 RETRY 将使消费者重试轮询同一消息，而 STOP 将停止消费者（如果消费者应该再次消耗消息，则必须手动启动/重新启动）。 Enum 值： • DISCARD • ERROR_HANDLER • RECONNECT • RETRY • STOP	ERROR_HANDLER	PollOnError
pollTimeoutMs (consumer)	轮询 KafkaConsumer 时使用的超时。	5000	Long
resumeStrategy (consumer)	此选项允许用户设置自定义恢复策略。当分配了分区时（例如：连接或重新连接时），执行恢复策略。它允许实施自定义如何恢复操作，并作为 seekTo 和 offsetRepository 机制的更灵活的替代方法。有关实现详情，请参阅 KafkaConsumerResumeStrategy。此选项不会影响自动提交设置。使用此设置的实现可能还希望通过手动提交选项来评估。		KafkaConsumerResumeStrategy
seekTo (consumer)	设置 KafkaConsumer 是否可以从启动时读取或结束：从开始 : read from end : read from end This is replace the earlier property seekToBeginning。 Enum 值： • 开始 • end		字符串
sessionTimeoutMs (consumer)	使用 Kafka 的组管理功能时检测失败的超时。	10000	整数
specificAvroReader (consumer)	这可让使用特定的 Avro reader 与 Confluent Platform 模式 registry 和 io.confluent.kafka.serializers.KafkaAvroDeserializer 一起使用。这个选项仅适用于 Confluent Platform （非标准 Apache Kafka）。	false	布尔值

Name	描述	默认值	类型
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动态主题数量。	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
kafkaManualCommitFactory (consumer (advanced))	用于创建 KafkaManualCommit 实例的工厂。这允许在手动提交来自开箱即用的默认实现时，自定义工厂创建自定义 KafkaManualCommit 实例。		KafkaManualCommitFactory
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。如果发送记录的速度比生成者可以更快地发送到服务器，则根据 block.on.buffer.full 指定的首选项阻止或抛出异常。此设置应该与生成者使用的总内存对应，但不是硬绑定，因为生成者都用于缓冲区。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

Name	描述	默认值	类型
compressionCodec (producer)	此参数允许您为此制作者生成的所有数据指定压缩代码c。有效值为 none、gzip 和 snappy。 Enum 值： • none • gzip • snappy • lz4	none	字符串
connectionMaxIdleMs (producer)	在这个配置指定的毫秒数后关闭闲置连接。	540000	整数
deliveryTimeoutMs (producer)	在调用 send （）返回后报告成功或失败的上限。这限制了发送前记录总时间、从代理等待确认时间（如果预期），以及可重新发送失败的时间。	120000	整数
enableIdempotence (producer)	如果设置为 'true'，则制作者将确保每个消息的确切副本写入流。如果 'false'，则制作者重试可能会在流中写入重试消息的副本。如果设置为 true，则此选项将需要 max.in.flight.requests.per.connection 设置为 1，重试不能为零，另外 acks 必须设置为 'all'。	false	布尔值
headerSerializer (producer)	使用自定义 KafkaHeaderSerializer 来序列化 kafka 标头值。		KafkaHeaderSerializer
key (producer)	记录密钥（如果没有指定密钥，则为 null）。如果配置了这个选项，则它优先于标头 KafkaConstants#KEY。		字符串
keySerializer (producer)	密钥的序列化器类（如果没有提供，默认为与消息相同）。	org.apache.kafka.common.serialization.StringSerializer	字符串

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
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人工延迟来实现此目的，即不立即发送记录，而不是立即发送记录，让生产者最多等待给定延迟，以允许发送其他记录，以便可将发送批处理在一起。这可能被视为与 TCP 中的 Nagle 算法类似。此设置提供批处理延迟的上限：一旦我们获得批处理。无论此设置如何，无论此设置如何，它都会立即发送一次记录，但是如果我们为此分区累积了这个字节，我们将"linger"用于等待更多记录显示。此设置默认为 0（例如，无延迟）。例如，设置 linger.ms=5 效果会减少发送的请求数量，但会向负载中发送的记录添加最多 5ms 的延迟。	0	整数
maxBlockMs (producer)	配置控制发送到 kafka 将阻止的时长。出于多种原因，这些方法可以被阻止。例如：缓冲区满，元数据不可用。此配置对获取元数据、密钥和值、执行 send （） 时缓冲内存所花费的总时间实施最大限制。如果是 partitionsFor （），此配置会在等待元数据时实施最长时间阈值。	60000	整数
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，如果将此设置设置为大于 1，且发送失败，则可能会因为重试而重新排序消息的风险（例如，如果启用了重试）。	5	整数
maxRequestSize (producer)	请求的最大大小。这也实际上是最大记录大小的上限。请注意，服务器在记录大小上具有自己的上限，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。通过实施 MetricReporter 接口，可以插入将收到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
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)	用于分区子主题中分区消息的分区类。默认分区器基于密钥的哈希。	org.apache.kafka.clients.producer.internals.DefaultPartitioner	字符串
PartitionKey (producer)	将发送记录的分区（如果未指定分区，则为 null）。如果配置了这个选项，则它优先于标头 KafkaConstants#PARTITION_KEY。		整数
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）。不会尝试批处理记录大于这个大小。发送到代理的Requests 将包含多个批处理，每个分区都可用于发送数据。小批处理大小会减少批处理，并可能会减少吞吐量（零的批处理大小将完全禁用批处理）。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 模式时可以放入生产者的最大未发送消息数，然后才能阻止生产者或必须丢弃数据。	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等待的时间。这可避免在紧密循环中重复连接到主机。此 backoff 应用到消费者发送到代理的所有请求。	50	整数
recordMetadata (producer)	producer 是否应该存储来自发送到 Kafka 的 RecordMetadata 结果。结果存储在包含 RecordMetadata 元数据的列表中。该列表存储在一个带有键 KafkaConstants#KAFKA_RECORDMETA 的标头中。	true	布尔值

Name	描述	默认值	类型
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。以下设置很常见：acks=0 如果设为零，则生成者不会等待来自服务器的任何确认。记录将立即添加到套接字缓冲区中并被视为发送。无法保证服务器已收到记录，重试配置不会生效（因为客户端通常不知道任何失败）。对每个记录给出的偏移始终设置为 -1 acks=1，这意味着领导机将记录写入其本地日志，但不会等待所有后续者完全确认。在这种情况下，领导机会在确认记录后立即失败，但在后续者复制之前，记录将会丢失。acks=all 意味着领导机将等待整个同步副本集合来确认记录。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。 Enum 值： ● -1 ● 0 ● 1 ● all	1	字符串
requestTimeoutMs (producer)	在将一个错误发送到客户端前，代理会等待尝试满足 request.required.acks 要求的时间。	30000	整数
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，因为如果两个记录发送到单个分区，则第一次失败并且被重试，但第二个记录会先出现，然后是第二个记录。	0	整数
retryBackoffMs (producer)	每次重试前，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，因此此属性指定生成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
valueSerializer (producer)	serializer 类用于消息。	org.apache.kafka.common.serialization.StringSerializer	字符串

Name	描述	默认值	类型
workerpool (producer)	要在 kafka 服务器确认从 KafkaProducer 发送的消息使用异步非阻塞处理，使用自定义 worker 池继续路由交换。如果使用这个选项，您必须处理线程池的生命周期，以便在不再需要时关闭池。		ExecutorService
workerPoolCoreSize (producer)	kafka 服务器在 kafka 服务器确认从 KafkaProducer 发送的消息（使用异步非阻塞处理）来继续路由交换的 worker 池的核心线程数量。	10	整数
workerPoolMaxSize (producer)	kafka 服务器之后，worker 池的最大线程数量用于继续路由交换，使用异步非块处理确认从 KafkaProducer 发送的消息。	20	整数
kafkaClientFactory (advanced)	用于创建 org.apache.kafka.clients.consumer.KafkaConsumer 和 org.apache.kafka.clients.producer.KafkaProducer 实例的工厂。这允许配置自定义工厂来创建带有扩展 vanilla Kafka 客户端的逻辑的实例。		KafkaClientFactory
同步 (advanced)	设置是否应严格使用同步处理。	false	布尔值
schemaRegistryURL (confluent)	要使用的 Confluent Platform 模式 registry 服务器的 URL。格式为 host1:port1,host2:port2。这在 Confluent Platform 文档中被称为 schema.registry.url。这个选项仅适用于 Confluent Platform （非标准 Apache Kafka）。		字符串
interceptorClasses (monitoring)	为制作者或消费者设置拦截器。制作者拦截器必须是实施 org.apache.kafka.clients.producer.ProducerInterceptor 拦截器的类，必须是实施 org.apache.kafka.clients.consumer.ConsumerInterceptor 的类。如果您对消费者使用 Producer 拦截器，它将在运行时抛出类 cast 异常。		字符串
kerberosBeforeReloginMinTime (security)	刷新尝试之间的登录线程睡眠时间。	60000	整数
kerberosInitCmd (security)	Kerberos kinit 命令路径。默认为 /usr/bin/kinit。	/usr/bin/kinit	字符串

Name	描述	默认值	类型
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，形式 {username}/{hostname}{REALM} 的主体名称映射到 {username}。有关格式的详情，请查看安全授权和 acls 文档。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。有关有效值，请参阅。	GSSAPI	字符串
securityProtocol (security)	用于与代理通信的协议。支持 SASL_PLAINTEXT, 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。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。如果配置它，它会在其他 SSL 端点参数之前应用。注意：Kafka 只支持从文件位置加载密钥存储，因此请在 KeyStoreParameters.resource 选项中使用 file: 前缀。		SSLContextParameters
sslEnabledProtocols (security)	为 SSL 连接启用的协议列表。TLSv1.2、TLSv1.1 和 TLSv1 默认启用。		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。	https	字符串
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

Name	描述	默认值	类型
sslKeyPassword (security)	密钥存储文件中的私钥密码。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。对于客户端，这是可选的，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，这适用于大多数情况。最近的 JVM 中允许的值为 TLS、TLSv1.1 和 TLSv1.2。较旧的 JVM 中可能支持 SSL、SSLv2 和 SSLv3，但由于已知的安全漏洞，不建议使用它们。		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

***有关 Producer/Consumer 配置的更多信息，** 请参阅：*

- ***http://kafka.apache.org/documentation.html#newconsumerconfigs***
- ***http://kafka.apache.org/documentation.html#producerconfigs***

58.6. 消息标头

58.6.1. 消费者标头

当消耗来自 Kafka 的消息时，可以使用以下标头。

标头常数	标头值	类型	描述
KafkaConstants.TOPIC	"kafka.TOPIC"	字符串	消息源自的位置的主题
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	如果配置了，消息的密钥
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	在提交前是否是最后的记录（仅在 **autoCommitEnable** endpoint 参数为 **false**时才可用）
KafkaConstants.LAST_POLL_RECORD	"kafka.LAST_POLL_RECORD"	布尔值	指明当前轮询请求中的最后一条记录（仅在 **autoCommitEnable** endpoint 参数为 **false** 或 **allowManualCommit** 为 **true**时才可用）
KafkaConstants.MANUAL_COMMIT	"CamelKafkaManualCommit"	KafkaManualCommit	可用于在使用 Kafka 消费者时强制手动偏移提交。

58.6.2. 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** 消息的键，以确保所有相关消息都在同一分区中

标头常数	标头值	类型	描述
KafkaConstants.OVERRIDE_TOPIC	**"kafka.OVERRIDE_TOPIC"**	**字符串**	发送消息的主题（覆盖和优先权），并且标头不会被保留。
KafkaConstants.OVERRIDE_TIMESTAMP	**"kafka.OVERRIDE_TIMESTAMP"**	**Long**	ProducerRecord 也有一个关联的时间戳。如果用户提供时间戳，则生成者将使用提供的时间戳标记记录，且不会保留标头。
KafkaConstants.PARTITION_KEY	**"kafka.PARTITION_KEY"**	**整数**	明确指定分区

如果要将消息发送到动态主题，则使用 KafkaConstants.OVERRIDE_TOPIC 作为其用作不发送消息的一次性标头，作为其在生成者中删除的一次性标头。

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（只在 **recordMetadata** endpoint 参数为 **true**时才配置

58.7. 消费者错误处理

虽然 kafka consumer 正在轮询来自 kafka 代理的消息，但可能会出现错误。本节描述了发生的情况以及您可以配置的内容。

在调用 Kafka 轮询 API 时，消费者可能会抛出异常。例如，如果消息因为无效数据而无法序列化，以及许多其他错误。这些错误采用 KafkaException 的形式，可以是 retry 或 not。可以重试的例外 (RetriableException)将再次重试（间隔出现轮询超时）。所有其他类型的异常会根据 pollOnError 配置进行处理。此配置具有以下值：

- *DISCARD 将丢弃消息并继续轮询下一个消息。*
- *ERROR_HANDLER 将使用 Camel 的错误处理程序来处理异常，之后继续轮询下一个消息。*

- RECONNECT 将重新连接消费者，并再次尝试轮询消息。
- RETRY 将使消费者再次重试轮询相同的消息
- STOP 将停止消费者（如果消费者应能够再次使用消息，则必须手动启动/重新启动）。

默认值为 ERROR_HANDLER，它允许 Camel 的错误处理程序（若有配置）处理原因异常。然后，继续轮询下一个消息。这个行为与 Camel 组件拥有的 bridgeErrorHandler 选项类似。

对于高级控制，可以在组件级别上配置 org.apache.camel.component.kafka.PollExceptionStrategy 的自定义实现，它允许控制上述策略出现哪些例外情况。

58.8. SAMPLES

58.8.1. 使用来自 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]}")
```

也可以订阅多个主题，提供模式作为主题名称，并使用 topicIsPattern 选项。

```
from("kafka:test*?brokers=localhost:9092&topicIsPattern=true")
  .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，组件需要 StateRepository 实现，如 FileStateRepository。此 bean 应该位于 registry 中。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
Registry registry = createCamelRegistry();
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");
    }
});
```

58.8.2. 将信息生成到 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");
```

58.9. 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。

第一种方法是通过许多 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
Registry registry = createCamelRegistry();
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");
  }
});
```

58.10. 使用 KAFKA IDEMPOTENT 存储库

camel-kafka 库提供了一个基于 Kafka 主题的幂等存储库。

此存储库将所有更改广播到 Kafka 主题中的幂等状态(add/remove)，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。使用的主题必须为每个幂等存储库实例唯一。

机制对于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子没有任何要求。

每个使用主题的存储库实例（例如，通常在并行运行的不同机器上）控制自己的消费者组，因此，使用相同主题的 10 Camel 进程集群将控制其自身偏移。

启动时，实例会订阅主题，并将偏移重新放在开始，将缓存重新构建到最新的状态。缓存不会被视为温，直到长度为 pollDurationMs 轮询返回 0 记录。在缓存已温启动或 30 秒前，启动才会完成；如果后者发生幂等存储库可能处于不一致的状态，直到其消费者捕获到主题的末尾为止。

请注意用于唯一性检查的标头格式。默认情况下，它使用 Strings 作为数据类型。使用原语数字格式时，必须相应地反序列化标头。查看以下示例示例。

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 consumer 上的 **bootstrap.servers** 属性。如果没有设置 **consumerConfig** 和 **producerConfig**，则将其用作简短的。如果使用，此组件将为生成者和消费者应用可诊断的默认配置。
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 消费者用于从主题中填充缓存的属性。覆盖 **bootstrapServers**，因此必须定义 Kafka **bootstrap.servers** 属性本身
maxCacheSize	最近使用的密钥应存储在内存中（默认值 1000）。
pollDurationMs	Kafka 消费者的轮询持续时间。本地缓存会立即更新。这个值会影响从主题更新其缓存的其他同级服务器相对于发送缓存的幂等消费者实例。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消费者和 Kafka 代理之间的网络流量卷之间有一个利弊。缓存温过程还取决于有一个轮询，没有任何内容 - 这表示流已被消耗到当前点。如果轮询持续时间过长，对于在主题上发送消息的速率，则缓存可能会进行温处理，并且与其对等点相比将以不一致的状态运行，直到捕获为止。

可以通过定义 topic 和 bootstrapServers，或者 producerConfig 和 consumerConfig 属性集来实例化存储库，以启用 SSL/SASL 等功能。要使用，此存储库必须手动或注册为 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>
```

使用带有数字标识符的 idempotency 时可以选择 3 个替代方案。第一个方法是使用来自 org.apache.camel.component.kafka.serde.KafkaSerdeHelper 的静态方法 numericHeader 方法为您

执行转换：

```
from("direct:performInsert")

.idempotentConsumer(numericHeader("id")).messageIdRepositoryRef("insertDbIdemRepo")
      // once-only insert into database
   .end()
```

另外，也可以使用通过路由 URL 配置的自定义序列化器来执行转换：

```
public class CustomHeaderDeserializer extends DefaultKafkaHeaderDeserializer {
   private static final Logger LOG =
LoggerFactory.getLogger(CustomHeaderDeserializer.class);

   @Override
   public Object deserialize(String key, byte[] value) {
      if (key.equals("id")) {
         BigInteger bi = new BigInteger(value);

         return String.valueOf(bi.longValue());
      } else {
         return super.deserialize(key, value);
      }
   }
}
```

最后，也可以在处理器中这样做：

```
from(from).routeId("foo")
   .process(exchange -> {
      byte[] id = exchange.getIn().getHeader("id", byte[].class);

      BigInteger bi = new BigInteger(id);
      exchange.getIn().setHeader("id", String.valueOf(bi.longValue()));
   })
   .idempotentConsumer(header("id"))
   .messageIdRepositoryRef("kafkaIdempotentRepository")
   .to(to);
```

58.11. 使用带有 KAFKA 消费者的手动提交

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。这需要将在 KafkaComponent 或端点上的选项 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 代码中的 KafkaManualCommit，如 Camel Processor ：

```
public void process(Exchange exchange) {
    KafkaManualCommit manual =
        exchange.getIn().getHeader(KafkaConstants.MANUAL_COMMIT,
KafkaManualCommit.class);
    manual.commit();
}
```

这将强制进行同步提交，该提交将阻止在 Kafka 上确认提交，或者在抛出异常时失败。您还可以使用一个异步提交，使用 'DefaultKafkaManualAsyncCommitFactory' 实现配置 KafkaManualCommitFactory ' 实现。

然后，提交将在下一个消费者循环中使用 kafka 异步提交 api 进行。请注意，分区中的记录必须由唯一的线程处理和提交。如果没有，这可能会导致行为不一致。这主要用于聚合的完成超时策略。

如果要使用 KafkaManualCommit 的自定义实现，您可以在 KafkaComponent 上创建自定义实现实例上配置自定义 KafkaManualCommitFactory。

58.12. KAFKA HEADERS PROPAGATION

当消耗来自 Kafka 的消息时，标头会自动传播到 camel 的交换标头。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，因此如果 camel Exchange 标头被传播它的值应该被序列化为 bytes[]，否则会跳过标头。支持以下标头值类型： String,Integer,Long, Double, Boolean,byte[]。注：所有标头生成的 从 kafka 到 camel 的交换默认都会包括值 byte[]。若要覆盖默认功能，uri 参数可以设置： headerDeserializer 用于 from 路由，headerSerializer 用于 to 路由。Example:

```
from("kafka:my_topic?headerDeserializer=#myDeserializer")
...
.to("kafka:my_topic?headerSerializer=#mySerializer")
```

默认情况下，所有标头都由 KafkaHeaderFilterStrategy 过滤。策略过滤掉以 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，必须手动或注册为 Spring/Blueprint 中的 bean，因为它是 CamelContext 感知。

58.13. SPRING BOOT AUTO-CONFIGURATION

组件支持 105 选项，如下所列。

Name	描述	默认值	类型
camel.component.kafka.additional-properties	如果无法直接在 camel 配置上设置 kafka consumer 或 kafka producer 的额外属性（例如：在 Camel 配置中还没有反映的新 Kafka 属性），属性必须加上 additionalProperties 前缀。例如：additionalProperties.transactional.id=12345&additionalProperties.schema.registry.url=http://localhost:8811/avro。		Map
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 消息标头中，它允许最终用户访问此 API 并通过 Kafka 消费者执行手动偏移提交。	false	布尔值
camel.component.kafka.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。当进程失败作为新消费者开始的位置时，将使用此提交的偏移。	true	布尔值
camel.component.kafka.auto-commit-interval-ms	消费者偏移提交至 zookeeper 的频率(ms)。	5000	整数
camel.component.kafka.auto-commit-on-stop	是否在消费者停止时执行显式自动提交，以确保代理从最后使用的消息中有提交。这要求打开了选项 autoCommitEnable。可能的值有： sync、sync 或 none。sync 是默认值。	同步	字符串

Name	描述	默认值	类型
camel.component.kafka.auto-offset-reset	当 ZooKeeper 中没有初始偏移时，或偏移没有范围：earliest : automatically the offset to the earliest offset : automatically the offset to the latest offset failed: throw exception to the consumer.	latest	字符串
camel.component.kafka.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kafka.break-on-first-error	此选项控制消费者处理交换时会发生什么，并且失败。如果选项为 false，则消费者将继续进入下一个消息并进行处理。如果选项为 true，则消费者将显示为导致失败的消息偏移，然后重新尝试处理此消息。但是，如果绑定到每次失败，这可能会导致正常处理同一消息的处理，例如 poison 消息。因此，建议您使用 Camel 的错误处理程序来应对这种情况。	false	布尔值
camel.component.kafka.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.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或指向代理子集的 VIP。这个选项在 Kafka 文档中称为 bootstrap.servers。		字符串
camel.component.kafka.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。如果发送记录的速度比生成者可以更快地发送到服务器，则根据 block.on.buffer.full 指定的首选项阻止或抛出异常。此设置应该与生成者使用的总内存对应，但不是硬绑定，因为生成者都用于缓冲区。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.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.client-id	客户端 ID 是每个请求中发送的用户指定字符串，以帮助追踪调用。它应该逻辑地标识发出请求的应用程序。		字符串

Name	描述	默认值	类型
camel.component.kafka.commit-timeout-ms	代码等待同步提交完成的最长时间（以毫秒为单位）。选项是一个 java.lang.Long 类型。	5000	Long
camel.component.kafka.compression-codec	此参数允许您为此制作者生成的所有数据指定压缩代码c。有效值为 none、gzip 和 snappy。	none	字符串
camel.component.kafka.configuration	允许使用端点将重复使用的通用选项预配置 Kafka 组件。选项是一个 org.apache.camel.component.kafka.KafkaConfiguration 类型。		KafkaConfiguration
camel.component.kafka.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.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.consumers-count	连接到 kafka 服务器的使用者数量。每个消费者都在单独的线程上运行，用于检索和处理传入的数据。	1	整数
camel.component.kafka.delivery-timeout-ms	在调用 send （）返回后报告成功或失败的上限。这限制了发送前记录总时间、从代理等待确认时间（如果预期），以及可重新发送失败的时间。	120000	整数
camel.component.kafka.enable-idempotence	如果设置为 'true'，则制作者将确保每个消息的确切副本写入流。如果 'false'，则制作者重试可能会在流中写入重试消息的副本。如果设置为 true，则此选项将需要 max.in.flight.requests.per.connection 设置为 1，重试不能为零，另外 acks 必须设置为 'all'。	false	布尔值
camel.component.kafka.enabled	是否启用 kafka 组件的自动配置。这默认是启用的。		布尔值
camel.component.kafka.fetch-max-bytes	如果获取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为 fetch 请求。这不是绝对的，如果 fetch 的第一个非空分区中的信息大于这个值，则消息仍会返回，以确保消费者能够进行进度。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

Name	描述	默认值	类型
camel.component.kafka.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.fetch-wait-max-ms	如果没有足够的数据立即满足 fetch.min.bytes，则服务器在回答获取请求前将阻断的最大时间。	500	整数
camel.component.kafka.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
camel.component.kafka.group-instance-id	最终用户提供的消费者实例的唯一标识符。只允许非空字符串。如果设置，消费者被视为静态成员，这意味着任何时候都只允许具有此 ID 的一个实例。这可与更大的会话超时结合使用，以避免由临时不可用造成的组重新平衡（如进程重启）。如果没有设置，消费者将加入组作为动态成员，这是传统行为。		字符串
camel.component.kafka.header-deserializer	使用自定义 KafkaHeaderDeserializer 来反序列化 kafka 标头值。选项是一个 org.apache.camel.component.kafka.serde.KafkaHeaderDeserializer 类型。		KafkaHeaderDeserializer
camel.component.kafka.header-filter-strategy	使用自定义 HeaderFilterStrategy 过滤标头到 Camel 消息。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		HeaderFilterStrategy
camel.component.kafka.header-serializer	使用自定义 KafkaHeaderSerializer 来序列化 kafka 标头值。选项是一个 org.apache.camel.component.kafka.serde.KafkaHeaderSerializer 类型。		KafkaHeaderSerializer
camel.component.kafka.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，但通常不应设置高于该值的 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.interceptor-classes	为制作者或消费者设置拦截器。制作者拦截器必须是实施 org.apache.kafka.clients.producer.ProducerInterceptor 拦截器的类，必须是实施 org.apache.kafka.clients.consumer.ConsumerInterceptor 的类。如果您对消费者使用 Producer 拦截器，它将在运行时抛出类 cast 异常。		字符串

Name	描述	默认值	类型
camel.component.kafka.kafka-client-factory	用于创建 org.apache.kafka.clients.consumer.KafkaConsumer 和 org.apache.kafka.clients.producer.KafkaProducer 实例的工厂。这允许配置自定义工厂来创建带有扩展 vanilla Kafka 客户端的逻辑的实例。选项是一个 org.apache.camel.component.kafka.KafkaClientFactory 类型。		KafkaClientFactory
camel.component.kafka.kafka-manual-commit-factory	用于创建 KafkaManualCommit 实例的工厂。这允许在手动提交来自开箱即用的默认实现时，自定义工厂创建自定义 KafkaManualCommit 实例。选项是 org.apache.camel.component.kafka.KafkaManualCommitFactory 类型。		KafkaManualCommitFactory
camel.component.kafka.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.kerberos-init-cmd	Kerberos kinit 命令路径。默认为 /usr/bin/kinit。	/usr/bin/kinit	字符串
camel.component.kafka.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，形式 {username}/{hostname}{REALM} 的主体名称映射到 {username}。有关格式的详情，请查看安全授权和 acls 文档。可以使用逗号分隔多个值。	DEFAULT	字符串
camel.component.kafka.kerberos-renew-jitter	添加到续订时间的随机 jitter 的百分比。		å�Œ
camel.component.kafka.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.key	记录密钥（如果没有指定密钥，则为 null）。如果配置了这个选项，则它优先于标头 KafkaConstants#KEY。		字符串

Name	描述	默认值	类型
camel.component.kafka.key-deserializer	deserializer 类用于实现 Deserializer 接口的密钥。	org.apache.kafka.common.serialization.StringDeserializer	字符串
camel.component.kafka.key-serializer	密钥的序列化器类（如果没有提供，默认为与消息相同）。	org.apache.kafka.common.serialization.StringSerializer	字符串
camel.component.kafka.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.kafka.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人工延迟来实现此目的，即不立即发送记录，而不是立即发送记录，让生产者最多等待给定延迟，以允许发送其他记录，以便可将发送批处理在一起。这可能被视为与 TCP 中的 Nagle 算法类似。此设置提供批处理延迟的上限：一旦我们获得批处理。无论此设置如何，无论此设置如何，它都会立即发送一次记录，但是如果我们为此分区累积了这个字节，我们将"linger"用于等待更多记录显示。此设置默认为 0（例如，无延迟）。例如，设置 linger.ms=5 效果会减少发送的请求数量，但会向负载中发送的记录添加最多 5ms 的延迟。	0	整数
camel.component.kafka.max-block-ms	配置控制发送到 kafka 将阻止的时长。出于多种原因，这些方法可以被阻止。例如：缓冲区满，元数据不可用。此配置对获取元数据、密钥和值、执行 send （）时缓冲内存所花费的总时间实施最大限制。如果是 partitionsFor （），此配置会在等待元数据时实施最长时间阈值。	60000	整数

Name	描述	默认值	类型
camel.component.kafka.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，如果将此设置设置为大于 1，且发送失败，则可能会因为重试而重新排序消息的风险（例如，如果启用了重试）。	5	整数
camel.component.kafka.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一样大，否则制作者可能会发送大于消费者可以获取的消息。如果发生这种情况，消费者可能会卡住在某个分区中获取大量消息。	1048576	整数
camel.component.kafka.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。选项是一个 java.lang.Long 类型。		Long
camel.component.kafka.max-poll-records	单个调用返回到 poll （）中返回的最大记录数。	500	整数
camel.component.kafka.max-request-size	请求的最大大小。这也实际上是最大记录大小的上限。请注意，服务器在记录大小上具有自己的上限，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.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.metric-reporters	用作指标报告器的类列表。通过实施 MetricReporter 接口，可以插入将收到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
camel.component.kafka.metrics-sample-window-ms	为计算指标维护的示例数量。	30000	整数
camel.component.kafka.no-of-metrics-sample	为计算指标维护的示例数量。	2	整数

Name	描述	默认值	类型
camel.component.kafka.offset-repository	用来本地存储主题每个分区的偏移量的偏移存储库。定义将禁用 autocommit。选项是一个 org.apache.camel.spi.StateRepository<java.lang.String, java.lang.String> 类型。		StateRepository
camel.component.kafka.partition-assignor	使用组管理时，客户端将使用的类名称在消费者实例之间分发分区所有权。	org.apache.kafka.clients.consumer.RangeAssignor	字符串
camel.component.kafka.partition-key	将发送记录的分区（如果未指定分区，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.partitioner	用于分区子主题中分区消息的分区类。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.poll-exception-strategy	要将自定义策略与消费者一起使用，以控制如何在池消息时处理 Kafka 代理的异常。选项是一个 org.apache.camel.component.kafka.PollExceptionStrategy 类型。		PollExceptionStrategy
camel.component.kafka.poll-on-error	如果 kafka threw 异常轮询新消息，则该怎么办。默认情况下，将使用来自组件配置的值，除非在端点级别上配置了显式值。DISCARD 将丢弃消息并继续轮询下一个消息。ERROR_HANDLER 将使用 Camel 的错误处理程序来处理异常，之后继续轮询下一个消息。RECONNECT 将重新连接消费者，并尝试再次轮询消息 RETRY 将使消费者重试轮询同一消息，而 STOP 将停止消费者（如果消费者应该再次消耗消息，则必须手动启动/重新启动）。		PollOnError
camel.component.kafka.poll-timeout-ms	轮询 KafkaConsumer 时使用的超时。选项是一个 java.lang.Long 类型。	5000	Long

Name	描述	默认值	类型
camel.component.kafka.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）。不会尝试批处理记录大于这个大小。发送到代理的Requests 将包含多个批处理，每个分区都可用于发送数据。小批处理大小会减少批处理，并可能会减少吞吐量（零的批处理大小将完全禁用批处理）。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.queue-buffering-max-messages	在使用 async 模式时可以放入生产者的最大未发送消息数，然后才能阻止生产者或必须丢弃数据。	10000	整数
camel.component.kafka.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.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.reconnect-backoff-ms	尝试重新连接给定主机前等待的时间。这可避免在紧密循环中重复连接到主机。此 backoff 应用到消费者发送到代理的所有请求。	50	整数
camel.component.kafka.record-metadata	producer 是否应该存储来自发送到 Kafka 的 RecordMetadata 结果。结果存储在包含 RecordMetadata 元数据的列表中。该列表存储在一个带有键 KafkaConstants#KAFKA_RECORDMETA 的标头中。	true	布尔值
camel.component.kafka.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。以下设置很常见：acks=0 如果设为零，则生成者不会等待来自服务器的任何确认。记录将立即添加到套接字缓冲区中并被视为发送。无法保证服务器已收到记录，重试配置不会生效（因为客户端通常不知道任何失败）。对每个记录给出的偏移始终设置为 -1 acks=1，这意味着领导机将记录写入其本地日志，但不会等待所有后续者完全确认。在这种情况下，领导机会在确认记录后立即失败，但在后续者复制之前，记录将会丢失。acks=all 意味着领导机将等待整个同步副本集合来确认记录。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

Name	描述	默认值	类型
camel.component.kafka.request-timeout-ms	在将一个错误发送到客户端前，代理会等待尝试满足 request.required.acks 要求的时间。	30000	整数
camel.component.kafka.resume-strategy	此选项允许用户设置自定义恢复策略。当分配了分区时（例如：连接或重新连接时），执行恢复策略。它允许实施自定义如何恢复操作，并作为 seekTo 和 offsetRepository 机制的更灵活的替代方法。有关实现详情，请参阅 KafkaConsumerResumeStrategy。此选项不会影响自动提交设置。使用此设置的实现可能还希望通过手动提交选项来评估。选项是一个 org.apache.camel.component.kafka.consumer.support.KafkaConsumerResumeStrategy 类型。		KafkaConsumerResumeStrategy
camel.component.kafka.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，因为如果两个记录发送到单个分区，则第一次失败并且被重试，但第二个记录会先出现，然后是第二个记录。	0	整数
camel.component.kafka.retry-backoff-ms	每次重试前，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，因此此属性指定生成者在刷新元数据前等待的时间。	100	整数
camel.component.kafka.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.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.sasl-mechanism	使用简单身份验证和安全层(SASL)机制。有关有效值，请参阅。	GSSAPI	字符串
camel.component.kafka.schema-registry-u-r-l	要使用的 Confluent Platform 模式 registry 服务器的 URL。格式为 host1:port1,host2:port2。这在 Confluent Platform 文档中被称为 schema.registry.url。这个选项仅适用于 Confluent Platform （非标准 Apache Kafka）。		字符串
camel.component.kafka.security-protocol	用于与代理通信的协议。支持 SASL_PLAINTEXT, PLAINTEXT 和 SSL。	PLAINTEXT	字符串

Name	描述	默认值	类型
camel.component.kafka.seek-to	设置 KafkaConsumer 是否可以从启动时读取或结束：从开始 : read from end : read from end This is replace the earlier property seekToBeginning。		字符串
camel.component.kafka.send-buffer-bytes	套接字写入缓冲区大小。	131072	整数
camel.component.kafka.session-timeout-ms	使用 Kafka 的组管理功能时检测失败的超时。	10000	整数
camel.component.kafka.shutdown-timeout	以毫秒为单位，等待消费者或制作者正常关闭并终止其 worker 线程。	30000	整数
camel.component.kafka.specific-avro-reader	这可让使用特定的 Avro reader 与 Confluent Platform 模式 registry 和 io.confluent.kafka.serializers.KafkaAvroDeserializer 一起使用。这个选项仅适用于 Confluent Platform （非标准 Apache Kafka）。	false	布尔值
camel.component.kafka.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。支持所有可用的密码套件。		字符串
camel.component.kafka.ssl-context-parameters	使用 Camel SSLContextParameters 对象的 SSL 配置。如果配置它，它会在其他 SSL 端点参数之前应用。注意：Kafka 只支持从文件位置加载密钥存储，因此请在 KeyStoreParameters.resource 选项中使用 file: 前缀。选项是 org.apache.camel.support.jsse.SSLContextParameters 类型。		SSLContextParameters
camel.component.kafka.ssl-enabled-protocols	为 SSL 连接启用的协议列表。TLSv1.2、TLSv1.1 和 TLSv1 默认启用。		字符串
camel.component.kafka.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。	https	字符串
camel.component.kafka.ssl-key-password	密钥存储文件中的私钥密码。这对客户端是可选的。		字符串

Name	描述	默认值	类型
camel.component.kafka.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.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.ssl-keystore-password	密钥存储文件的存储密码。对于客户端，这是可选的，只有在配置了 ssl.keystore.location 时才需要。		字符串
camel.component.kafka.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.ssl-protocol	用于生成 SSLContext 的 SSL 协议。默认设置为 TLS，这适用于大多数情况。最近的 JVM 中允许的值为 TLS、TLSv1.1 和 TLSv1.2。较旧的 JVM 中可能支持 SSL、SSLv2 和 SSLv3，但由于已知的安全漏洞，不建议使用它们。		字符串
camel.component.kafka.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.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.ssl-truststore-location	信任存储文件的位置。		字符串
camel.component.kafka.ssl-truststore-password	信任存储文件的密码。		字符串
camel.component.kafka.ssl-truststore-type	信任存储文件的文件格式。默认值为 JKS。	JKS	字符串

Name	描述	默认值	类型
camel.component.kafka.synchronous	设置是否应严格使用同步处理。	false	布尔值
camel.component.kafka.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动态主题数量。	false	布尔值
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.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.value-serializer	serializer 类用于消息。	org.apache.kafka.common.serialization.StringSerializer	字符串
camel.component.kafka.worker-pool	要在 kafka 服务器确认从 KafkaProducer 发送的消息使用异步非阻塞处理，使用自定义 worker 池继续路由交换。如果使用这个选项，您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。		ExecutorService
camel.component.kafka.worker-pool-core-size	kafka 服务器在 kafka 服务器确认从 KafkaProducer 发送的消息（使用异步非阻塞处理）来继续路由交换的 worker 池的核心线程数量。	10	整数
camel.component.kafka.worker-pool-max-size	kafka 服务器之后，worker 池的最大线程数量用于继续路由交换，使用异步非块处理确认从 KafkaProducer 发送的消息。	20	整数

第 59 章 KAMELET

支持生成者和消费者

Kamelet 组件支持使用 Endpoint semantic 与 Camel Route Template 引擎交互。

59.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 kamelet 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-kamelet-starter</artifactId>
</dependency>
```

59.2. URI 格式

```
kamelet:templateId/routeId[?options]
```

59.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

59.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

59.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

59.4. 组件选项

Kamelet 组件支持 9 个选项，如下所列。

Name	描述	默认值	类型
location (common)	文件系统中 Kamelets 的位置。可以用逗号分隔多个位置。	classpath:/kamelets	字符串
routeProperties (common)	设置路由本地参数。		Map
templateProperties (common)	设置模板本地参数。		Map
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
block (producer)	如果发送消息到没有活跃消费者的 kamelet 端点，则我们可以告知生成者阻止，并等待消费者变为活动状态。	true	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
timeout (producer)	如果启用了块，要使用的超时值。	30000	long
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
routeTemplateLoaderListener (advanced)	当 Kamelet 组件从外部资源加载 Kamelets 时，autowired to 插件了一个自定义监听程序。		RouteTemplateLoaderListener

59.5. 端点选项

Kamelet 端点使用 URI 语法进行配置：

```
kamelet:templateId/routeId
```

使用以下路径和查询参数：

59.5.1. 路径参数(2 参数)

Name	描述	默认值	类型
templateId (common)	**必需** 路由模板 ID。		字符串

Name	描述	默认值	类型
routeId (common)	路由 ID。默认值通知：如果不提供，则 ID 将自动生成。		字符串

59.5.2. 查询参数(8 参数)

Name	描述	默认值	类型
location (common)	使用 Kamelet 的位置，它可以指定为文件系统、classpath 等资源。位置无法使用通配符，且必须引用包括扩展名的文件，例如 file:/etc/foo-kamelet.xml。		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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
block (producer)	如果向没有活跃消费者的直接端点发送消息，则我们可以告知生成者阻止，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 kamelet 端点时，生成者是否应该通过抛出异常失败。	true	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

Name	描述	默认值	类型
timeout (producer)	如果启用了块，要使用的超时值。	30000	long

注意

kamelet 端点 被盗，这意味着端点接受传递给引擎并在路由材料化时消耗的额外参数。

59.6. DISCOVERY（发现）

如果没有找到 Route 模板，kamelet 端点会尝试从文件系统中加载相关的 kamelet 定义（默认为 classpath:/kamelets）。默认解析机制预期 kamelet 文件具有 .kamelet.yaml 扩展。

59.7. SAMPLES

kamelets 可以像标准 Camel 组件一样使用。例如，假设我们创建了一个路由模板，如下所示：

```
routeTemplate("setMyBody")
   .templateParameter("bodyValue")
   .from("kamelet:source")
      .setBody().constant("{{bodyValue}}");
```

注意

要让 Kamelet 组件将材料化路由到调用器处理器，我们需要识别路由的输入和输出端点，这通过使用 kamele:source 标记输入端点和 kamelet:sink 作为输出端点。

然后，模板可以被实例化并调用，如下所示：

```
from("direct:setMyBody")
   .to("kamelet:setMyBody?bodyValue=myKamelet");
```

在 scenes 之后，Kamelet 组件执行以下操作：

1.

它实例化了一个路由，来自于给定 templateId path 参数标识的 Route 模板（本例中为 setBody）

2. 它将充当 直接 组件，并将当前路由连接到材料化。

如果您需要以编程方式进行，它类似如下：

```
routeTemplate("setMyBody")
   .templateParameter("bodyValue")
   .from("direct:{{foo}}")
      .setBody().constant("{{bodyValue}}");

TemplatedRouteBuilder.builder(context, "setMyBody")
   .parameter("foo", "bar")
   .parameter("bodyValue", "myKamelet")
   .add();

from("direct:template")
   .to("direct:bar");
```

59.8. 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.kamelet.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kamelet.block	如果发送消息到没有活跃消费者的 kamelet 端点，则我们可以告知生成者阻止，并等待消费者变为活动状态。	true	布尔值
camel.component.kamelet.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

Name	描述	默认值	类型
camel.component.kamelet.enabled	是否启用 kamelet 组件的自动配置。这默认是启用的。		布尔值
camel.component.kamelet.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.kamelet.location	文件系统中 Kamelets 的位置。可以用逗号分隔多个位置。	classpath:/kamelets	字符串
camel.component.kamelet.route-properties	设置路由本地参数。		Map
camel.component.kamelet.route-template-loader-listener	当 Kamelet 组件从外部资源加载 Kamelets 时，要插入自定义监听程序。选项是一个 org.apache.camel.spi.RouteTemplateLoaderListener 类型。		RouteTemplateLoaderListener
camel.component.kamelet.template-properties	设置模板本地参数。		Map
camel.component.kamelet.timeout	如果启用了块，要使用的超时值。	30000	Long

第 60 章 KAMELET MAIN

Since Camel 3.11

一个主要的类，用于 boostrap，并运行带有 Kamelets （或普通 YAML 路由）的 Camel 独立，用于开发和演示目的。

60.1. 初始配置

KameletMain 预先配置了以下属性：

```
camel.component.kamelet.location = classpath:/kamelets,github:apache:camel-kamelets/kamelets
camel.component.rest.consumerComponentName = platform-http
camel.component.rest.producerComponentName = vertx-http
```

您可以通过更新 application.properties 中的配置来覆盖这些设置。

60.2. 自动依赖项下载

Kamelet Main 可以通过 http/https 从远程位置自动下载 Kamelet YAML 文件，也可以从 github 中下载。

Apache Camel Kamelet Catalog 中的官方 Kamelets 存储在 github 上，并可按原样使用它们。

例如，Camel 路由可以在 YAML 中编码，该 YAML 使用目录中的 Earthquake Kamelet，如下所示：

```
- route:
    from: "kamelet:earthquake-source"
    steps:
      - unmarshal:
          json: {}
      - log: "Earthquake with magnitude ${body[properties][mag]} at ${body[properties][place]}"
```

在上例中，earthquake kamelet 将从 github 下载，及其所需的依赖项。

如需更多信息，** 请参阅 **Kamelet Main 示例

第 61 章 KUBERNETES

从 Camel 2.17 开始

Kubernetes 组件将您的应用程序与 Kubernetes 独立或 Openshift 之上集成。

61.1. KUBERNETES 组件

如需了解每个组件的使用，请参阅以下内容：

Kubernetes ConfigMap 对 Kubernetes ConfigMap 执行操作，并获得 ConfigMap 更改通知。

Kubernetes 自定义资源 对 Kubernetes 自定义资源执行操作，并获得 Deployment 更改通知。

Kubernetes Deployments 对 Kubernetes Deployment 执行操作，并获得 Deployment 更改通知。

Kubernetes 事件 对 Kubernetes 事件执行操作，并获得对 Events 更改的通知。

Kubernetes HPA 对 Kubernetes Horizontal Pod Autoscalers (HPA)执行操作，并获得 HPA 更改的通知。

Kubernetes 任务 对 Kubernetes 任务执行操作。

Kubernetes 命名空间 对 Kubernetes 命名空间执行操作，并获得命名空间更改通知。

Kubernetes 节点在 Kubernetes 节点上执行操作，并获得对节点更改的通知。

Kubernetes 持久性卷 对 Kubernetes 持久性卷执行操作，并获得对持久性卷更改的通知。

Kubernetes 持久性卷声明 对 Kubernetes 持久性卷声明执行操作，并获得对持久性卷声明更改的通知。

Kubernetes Pod 对 Kubernetes Pod 执行操作，并获得 Pod 更改通知。

Kubernetes Replication Controller 对 Kubernetes Replication Controller 执行操作，并在 Replication Controller 更改时获得通知。

Kubernetes 资源配额 对 Kubernetes 资源配额执行操作。

Kubernetes Secret 对 Kubernetes Secret 执行操作。

Kubernetes 服务帐户 对 Kubernetes 服务帐户执行操作。

Kubernetes Services 对 Kubernetes 服务执行操作，并获得对 Service 更改的通知。

OpenShift Build Config 对 OpenShift 构建配置执行操作.

OpenShift 构建 对 OpenShift 构建执行操作。

OpenShift Deployment Configs 对 Openshift Deployment Configs 执行操作，并获得 Deployment Config 更改通知。

61.2. 依赖项

将此组件的 pom.xml 添加以下依赖项：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

61.3. 使用方法

61.3.1. 生成者示例

在这里，我们显示一些使用 camel-kubernetes 的制作者示例。

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。

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。

61.4. 使用 KUBERNETES CONFIGMAP 和 SECRET

camel-kubernetes 组件还提供从 Kubernetes'ConfigMaps 或 Secrets 加载属性值的功能。

如需更多信息，请参阅。

61.5. SPRING BOOT AUTO-CONFIGURATION

组件支持 102 选项，如下所列。

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å‍Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long

Name	描述	默认值	类型
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-config-maps.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-config-maps.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.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-deployments.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-events.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.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-hpa.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.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
camel.component.kubernetes-hpa.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-job.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-namespaces.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.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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
camel.component.kubernetes-replication-controllers.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-replication-controllers.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

Name	描述	默认值	类型
camel.component.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-secrets.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

Name	描述	默认值	类型
camel.component.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-services.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.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
camel.component.kubernetes-services.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-deploymentconfigs.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

第 62 章 KUBERNETES CONFIGMAP

从 Camel 2.17 开始

支持生成者和消费者

Kubernetes ConfigMap 组件是 Kubernetes 组件 之一，它为执行 Kubernetes ConfigMap 操作和消费者使用与 ConfigMap 对象相关的事件提供了一个制作者。

62.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 kubernetes-config-maps 时，使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

62.2. 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

62.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

62.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

62.3. 组件选项

Kubernetes ConfigMap 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (common)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

62.4. 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：

62.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterUrl (common)	**所需的** Kubernetes 主 url。		字符串

62.4.2. 查询参数(33 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (common)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (common)	命名空间。		字符串
portName (common)	用于 ServiceCall EIP 的端口名称。		字符串

Name	描述	默认值	类型
portProtocol (common)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
crdGroup (consumer)	您要监视的 Consumer CRD 资源组。		字符串
crdName (consumer)	我们想监视的 Consumer CRD 资源名称。		字符串
crdPlural (consumer)	Consumer CRD Resource Plural we like to watch。		字符串
crdScope (consumer)	Consumer CRD 资源范围我们想监视。		字符串
crdVersion (consumer)	我们想监视的 Consumer CRD 资源版本。		字符串
labelKey (consumer)	当监视某些资源时，Consumer Label 键。		字符串
labelValue (consumer)	Consumer Label 值在监视某些资源时。		字符串
poolSize (consumer)	Consumer 池大小。	1	int
resourceName (consumer)	要监视的消费者资源名称。		字符串
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut		ExchangePattern
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串

Name	描述	默认值	类型
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

62.5. 消息标头

Kubernetes ConfigMap 组件支持 7 个消息标头，如下所列：

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNamespaceName (producer) 常量 ：KUBERNETES_NAMESPACE_NAME	命名空间名称。		字符串
CamelKubernetesConfigMapsLabels (producer) 恒定：KUBERNETES_CONFIGMAPS_LABELS	ConfigMap 标签。		Map
CamelKubernetesConfigMapName (producer) 常量 ：KUBERNETES_CONFIGMAP_NAME	ConfigMap 名称。		字符串

Name	描述	默认值	类型
CamelKubernetesConfigData (producer) 常量 ：KUBERNETES_CONFIGMAP_DATA	ConfigMap 数据。		Map
CamelKubernetesEventAction (consumer) 常量 ：KUBERNETES_EVENT_ACTION	消费者监视的操作。 Enum 值： • ADDED • MODIFIED • 已删除 • ERROR • 书签		操作
CamelKubernetesEventTimestamp (consumer) 恒定：KUBERNETES_EVENT_TIMESTAMP	消费者监视的操作的时间戳。		long

62.6. 支持的制作者操作

- ***listConfigMaps***
- ***listConfigMapsByLabels***
- ***getConfigMap***
- ***createConfigMap***

- updateConfigMap
- deleteConfigMap

62.7. KUBERNETES CONFIGMAPS PRODUCER 示例

- ListConfigMaps：此操作列出了 configmaps

```
from("direct:list").
    to("kubernetes-config-maps:///?
kubernetesClient=#kubernetesClient&operation=listConfigMaps").
    to("mock:result");
```

此操作会返回集群中的 ConfigMap 列表。

- listConfigMapsByLabels：此操作列出了标签选择的 configmaps。

```
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_CONFIGMAPS_LABELS,
labels);
            }
        });
    to("kubernetes-config-maps:///?
kubernetesClient=#kubernetesClient&operation=listConfigMapsByLabels").
    to("mock:result");
```

此操作使用标签选择器（带有 key1 和 key2，值为 value1 和 value2）来返回来自集群的 ConfigMap 列表。

62.8. KUBERNETES CONFIGMAPS 消费者示例

```
fromF("kubernetes-config-maps://%s?oauthToken=%s", host, authToken)
    .setHeader(KubernetesConstants.KUBERNETES_NAMESPACE_NAME, constant("default"))
    .setHeader(KubernetesConstants.KUBERNETES_CONFIGMAP_NAME, constant("test"))
    .process(new KubernertesProcessor()).to("mock:result");
```

```
    public class KubernertesProcessor implements Processor {
        @Override
        public void process(Exchange exchange) throws Exception {
            Message in = exchange.getIn();
            ConfigMap cm = exchange.getIn().getBody(ConfigMap.class);
            log.info("Got event with configmap name: " + cm.getMetadata().getName() + " and
    action " + in.getHeader(KubernetesConstants.KUBERNETES_EVENT_ACTION));
        }
    }
```

此消费者返回配置映射测试的命名空间 default 上的事件列表。

62.9. SPRING BOOT AUTO-CONFIGURATION

组件支持 102 选项，如下所列。

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å�Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串

Name	描述	默认值	类型
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-config-maps.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-config-maps.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.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-deployments.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.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
camel.component.kubernetes-deployments.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-events.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.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-hpa.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.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
camel.component.kubernetes-hpa.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-job.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-namespaces.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.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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

Name	描述	默认值	类型
camel.component.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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.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
camel.component.kubernetes-pods.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-replication-controllers.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.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-secrets.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.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-services.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.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
camel.component.kubernetes-services.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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

Name	描述	默认值	类型
camel.component.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.openshift-deploymentconfigs.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

第 63 章 KUBERNETES 自定义资源

自 Camel 3.7 起

支持生成者和消费者

Kubernetes 自定义资源组件是 Kubernetes 组件 之一，它为执行 Kubernetes 自定义资源操作和消费者使用与 Node 对象相关的事件提供了一个制作者。

63.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 kubernetes-custom-resources 时，请使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

63.2. 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

63.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

63.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

63.3. 组件选项

Kubernetes 自定义资源组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (common)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

63.4. 端点选项

Kubernetes 自定义资源端点使用 URI 语法进行配置：

```
kubernetes-custom-resources:masterUrl
```

使用以下路径和查询参数：

63.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterUrl (common)	**所需的** Kubernetes 主 url。		字符串

63.4.2. 查询参数(33 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (common)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (common)	命名空间。		字符串
portName (common)	用于 ServiceCall EIP 的端口名称。		字符串

Name	描述	默认值	类型
portProtocol (common)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
crdGroup (consumer)	您要监视的 Consumer CRD 资源组。		字符串
crdName (consumer)	我们想监视的 Consumer CRD 资源名称。		字符串
crdPlural (consumer)	Consumer CRD Resource Plural we like to watch。		字符串
crdScope (consumer)	Consumer CRD 资源范围我们想监视。		字符串
crdVersion (consumer)	我们想监视的 Consumer CRD 资源版本。		字符串
labelKey (consumer)	当监视某些资源时，Consumer Label 键。		字符串
labelValue (consumer)	Consumer Label 值在监视某些资源时。		字符串
poolSize (consumer)	Consumer 池大小。	1	int
resourceName (consumer)	要监视的消费者资源名称。		字符串
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut		ExchangePattern
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串

Name	描述	默认值	类型
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

63.5. 消息标头

Kubernetes 自定义资源组件支持 13 个消息标头，如下所列：

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNamespaceName (producer) 常量 ：KUBERNETES_NAMESPACE_NAME	命名空间名称。		字符串
CamelKubernetesCRDInstanceName (producer) 恒定：KUBERNETES_CRD_INSTANCE_NAME	部署名称。		字符串
CamelKubernetesCRDEventTimestamp (consumer) 恒定：KUBERNETES_CRD_EVENT_TIMESTAMP	消费者监视的操作的时间戳。		long

Name	描述	默认值	类型
CamelKubernetesCRDEventAction (consumer) 恒定：KUBERNETES_CRD_EVENT_ACTION	消费者监视的操作。 Enum 值： • ADDED • MODIFIED • 已删除 • ERROR • 书签		操作
CamelKubernetesCRDName (producer) 常量 ：KUBERNETES_CRD_NAME	我们想监视的 Consumer CRD 资源名称。		字符串
CamelKubernetesCRDGroup (producer) 常量 ：KUBERNETES_CRD_GROUP	您要监视的 Consumer CRD 资源组。		字符串
CamelKubernetesCRDScope (producer) 恒定：KUBERNETES_CRD_SCOPE	Consumer CRD 资源范围我们想监视。		字符串
CamelKubernetesCRDVersion (producer) 常量 ：KUBERNETES_CRD_VERSION	我们想监视的 Consumer CRD 资源版本。		字符串

Name	描述	默认值	类型
CamelKubernetesCRDPlural (producer) 恒定：KUBERNETES_CRD_PLURAL	Consumer CRD Resource Plural we like to watch。		字符串
CamelKubernetesCRDLabels (producer) 恒定：KUBERNETES_CRD_LABELS	CRD 资源标签。		Map
CamelKubernetesCRDInstance (producer) 恒定：KUBERNETES_CRD_INSTANCE	要作为 JSON 字符串创建的 CRD 资源清单。		字符串
CamelKubernetesDeleteResult (producer) 恒定：KUBERNETES_DELETE_RESULT	删除操作的结果。		布尔值

63.6. 支持的制作者操作

- ***listCustomResources***
- ***listCustomResourcesByLabels***
- ***getCustomResource***
- ***deleteCustomResource***

- ***createCustomResource***
- ***updateCustomResource***

63.7. SPRING BOOT AUTO-CONFIGURATION

*组**件支持 102** 选项，**如下所列。***

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å�Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long

Name	描述	默认值	类型
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-config-maps.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-config-maps.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.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-deployments.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.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
camel.component.kubernetes-deployments.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-events.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.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-hpa.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.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
camel.component.kubernetes-hpa.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-job.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-namespaces.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.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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

Name	描述	默认值	类型
camel.component.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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.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
camel.component.kubernetes-pods.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-replication-controllers.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.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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

Name	描述	默认值	类型
camel.component.kubernetes-secrets.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-secrets.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.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-services.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.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
camel.component.kubernetes-services.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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

Name	描述	默认值	类型
camel.component.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.openshift-deploymentconfigs.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

第 64 章 KUBERNETES DEPLOYMENTS

从 Camel 2.20 开始

支持生成者和消费者

Kubernetes Deployments 组件是 Kubernetes 组件 之一，它为执行 Kubernetes Deployments 操作和消费者使用与 Deployments 对象相关的事件提供了一个制作者。

64.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 kubernetes-deployments 时，使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

64.2. 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

64.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

64.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

64.3. 组件选项

Kubernetes Deployments 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (common)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

64.4. 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：

64.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterUrl (common)	**所需的** Kubernetes 主 url。		字符串

64.4.2. 查询参数(33 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (common)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (common)	命名空间。		字符串
portName (common)	用于 ServiceCall EIP 的端口名称。		字符串

Name	描述	默认值	类型
portProtocol (common)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
crdGroup (consumer)	您要监视的 Consumer CRD 资源组。		字符串
crdName (consumer)	我们想监视的 Consumer CRD 资源名称。		字符串
crdPlural (consumer)	Consumer CRD Resource Plural we like to watch。		字符串
crdScope (consumer)	Consumer CRD 资源范围我们想监视。		字符串
crdVersion (consumer)	我们想监视的 Consumer CRD 资源版本。		字符串
labelKey (consumer)	当监视某些资源时，Consumer Label 键。		字符串
labelValue (consumer)	Consumer Label 值在监视某些资源时。		字符串
poolSize (consumer)	Consumer 池大小。	1	int
resourceName (consumer)	要监视的消费者资源名称。		字符串
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut		ExchangePattern
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串

Name	描述	默认值	类型
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

64.5. 消息标头

Kubernetes Deployments 组件支持 8 个消息标头，如下所列：

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNamespaceName (producer) 常量 ：KUBERNETES_NAMESPACE_NAME	命名空间名称。		字符串
CamelKubernetesDeploymentsLabels (producer) 恒定：KUBERNETES_DEPLOYMENTS_LABELS	部署标签。		Map
CamelKubernetesDeploymentName (producer) 常数 ：KUBERNETES_DEPLOYMENT_NAME	部署名称。		字符串

Name	描述	默认值	类型
CamelKubernetesDeploymentSpec (producer) 常数 ：KUBERNETES_DEPLOYMENT_SPEC	部署的 spec。		DeploymentSpec
CamelKubernetesDeploymentReplicas (producer) 常数 ：KUBERNETES_DEPLOYMENT_REPLICAS	所需的实例数。		整数
CamelKubernetesEventAction (consumer) 常量 ：KUBERNETES_EVENT_ACTION	消费者监视的操作。 Enum 值： • ADDED • MODIFIED • 已删除 • ERROR • 书签		操作
CamelKubernetesEventTimestamp (consumer) 恒定：KUBERNETES_EVENT_TIMESTAMP	消费者监视的操作的时间戳。		long

64.6. 支持的制作者操作

- ***listDeployments***
- ***listDeploymentsByLabels***

- *getDeployment*
- *createDeployment*
- *updateDeployment*
- *deleteDeployment*
- *scaleDeployment*

64.7. KUBERNETES DEPLOYMENTS PRODUCER 示例

- *ListDeployments：此操作列出了 kubernetes 集群上的部署。*

```
from("direct:list").
   toF("kubernetes-deployments:///?
kubernetesClient=#kubernetesClient&operation=listDeployments").
   to("mock:result");
```

此操作会返回集群中的 Deployment 列表。

- *listDeploymentsByLabels：此操作使用 kubernetes 集群上的标签列出部署。*

```
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_DEPLOYMENTS_LABELS,
labels);
        }
     });
   toF("kubernetes-deployments:///?
kubernetesClient=#kubernetesClient&operation=listDeploymentsByLabels").
   to("mock:result");
```

此操作使用标签选择器（带有 key1 和 key2 的值2）返回来自集群的 Deployment 列表。

64.7.1. Kubernetes Deployments Consumer 示例

```
fromF("kubernetes-deployments://%s?
oauthToken=%s&namespace=default&resourceName=test", host, authToken).process(new
KubernertesProcessor()).to("mock:result");
    public class KubernertesProcessor implements Processor {
        @Override
        public void process(Exchange exchange) throws Exception {
            Message in = exchange.getIn();
            Deployment dp = exchange.getIn().getBody(Deployment.class);
            log.info("Got event with configmap name: " + dp.getMetadata().getName() + " and
action " + in.getHeader(KubernetesConstants.KUBERNETES_EVENT_ACTION));
        }
    }
```

此消费者返回部署测试的命名空间 default 上的事件列表。

64.8. SPRING BOOT AUTO-CONFIGURATION

组件支持 102 选项，如下所列。

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串

Name	描述	默认值	类型
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å�Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-config-maps.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.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

Name	描述	默认值	类型
camel.component.kubernetes-config-maps.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-config-maps.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.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-deployments.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.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
camel.component.kubernetes-deployments.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-events.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.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-hpa.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.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
camel.component.kubernetes-hpa.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-job.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-namespaces.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.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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

Name	描述	默认值	类型
camel.component.kubernetes-persistent-volumes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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.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
camel.component.kubernetes-pods.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-replication-controllers.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.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
camel.component.kubernetes-replication-controllers.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-replication-controllers.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.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-secrets.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.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-services.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.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
camel.component.kubernetes-services.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 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.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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

Name	描述	默认值	类型
camel.component.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-deploymentconfigs.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

第 65 章 KUBERNETES 事件

从 Camel 3.20 开始

支持生成者和消费者

Kubernetes 事件组件是 Kubernetes 组件 之一，它为执行 Kubernetes 事件操作和消费者使用与事件对象相关的事件提供了一个制作者。

65.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 kubernetes-events 时，使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

65.2. 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

65.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

65.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

65.3. 组件选项

Kubernetes 事件组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (common)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

65.4. 端点选项

Kubernetes 事件端点使用 URI 语法进行配置：

```
kubernetes-events:masterUrl
```

使用以下路径和查询参数：

65.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterUrl (common)	**所需的** Kubernetes 主 url。		字符串

65.4.2. 查询参数(33 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (common)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (common)	命名空间。		字符串
portName (common)	用于 ServiceCall EIP 的端口名称。		字符串

Name	描述	默认值	类型
portProtocol (common)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
crdGroup (consumer)	您要监视的 Consumer CRD 资源组。		字符串
crdName (consumer)	我们想监视的 Consumer CRD 资源名称。		字符串
crdPlural (consumer)	Consumer CRD Resource Plural we like to watch。		字符串
crdScope (consumer)	Consumer CRD 资源范围我们想监视。		字符串
crdVersion (consumer)	我们想监视的 Consumer CRD 资源版本。		字符串
labelKey (consumer)	当监视某些资源时，Consumer Label 键。		字符串
labelValue (consumer)	Consumer Label 值在监视某些资源时。		字符串
poolSize (consumer)	Consumer 池大小。	1	int
resourceName (consumer)	要监视的消费者资源名称。		字符串
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut		ExchangePattern
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串

Name	描述	默认值	类型
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

65.5. 消息标头

Kubernetes 事件组件支持 14 个消息标头，如下所列：

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNamespaceName (producer) 常量 ：KUBERNETES_NAMESPACE_NAME	命名空间名称。		字符串
CamelKubernetesEventsLabels (producer) 恒定：KUBERNETES_EVENTS_LABELS	事件标签。		Map
CamelKubernetesEventTime (producer) 恒定：KUBERNETES_EVENT_TIME	ISO-8601 中的事件时间扩展偏移日期日期，如 '2011-12-03T10:15:3001:00'。	服务器时间	字符串

Name	描述	默认值	类型
CamelKubernetesEventAction (consumer) 常量 ：KUBERNETES_EVENT_ACTION	消费者监视的操作。 Enum 值： • ADDED • MODIFIED • 已删除 • ERROR • 书签		操作
CamelKubernetesEventType (producer) 恒定：KUBERNETES_EVENT_TYPE	事件类型。		字符串
CamelKubernetesEventReason (producer) 恒定：KUBERNETES_EVENT_REASON	事件原因。		字符串
CamelKubernetesEventNote (producer) 恒定：KUBERNETES_EVENT_NOTE	事件备注。		字符串
CamelKubernetesEventRegarding (producer) 恒定：KUBERNETES_EVENT_REGARDING	相关事件。		ObjectReference

Name	描述	默认值	类型
CamelKubernetesEventRelated (producer) 恒定：KUBERNETES_EVENT_RELATED	相关事件。		ObjectReference
CamelKubernetesEventReportingController (producer) 恒定：KUBERNETES_EVENT_REPORTING_CONTROLLER	事件报告控制器。		字符串
CamelKubernetesEventReportingInstance (producer) 恒定：KUBERNETES_EVENT_REPORTING_INSTANCE	事件报告实例。		字符串
CamelKubernetesEventName (producer) 常量 ：KUBERNETES_EVENT_NAME	事件名称。		字符串
CamelKubernetesEventTimestamp (consumer) 恒定：KUBERNETES_EVENT_TIMESTAMP	消费者监视的操作的时间戳。		long

65.6. 支持的制作者操作

- ***listEvents***

- listEventsByLabels
- getEvent
- createEvent
- updateEvent
- deleteEvent

65.7. KUBERNETES 事件 PRODUCER 示例

- listEvents ：此操作列出了事件。

```
from("direct:list").
   to("kubernetes-events:///?kubernetesClient=#kubernetesClient&operation=listEvents").
   to("mock:result");
```

此操作会返回集群中的事件列表。事件的类型是 io.fabric8.kubernetes.api.model.events.v1.Event。

要指示事件来自哪个命名空间，可以设置消息标头 CamelKubernetesNamespaceName。默认情况下，返回所有命名空间的事件。

- listEventsByLabels：此操作列出了标签选择的事件。

```
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_EVENTS_LABELS,
labels);
        }
     });
```

```
    to("kubernetes-events:///?
kubernetesClient=#kubernetesClient&operation=listEventsByLabels").
    to("mock:result");
```

此操作会返回集群中发生的事件列表，使用标签选择器（上例中仅预期标签 "key1" 设置为 "value1" 的事件），标签 "key2" 设置为 "value2"）。事件的类型是 io.fabric8.kubernetes.api.model.events.v1.Event。

此操作需要消息标头 CamelKubernetesEventsLabels 设置为 Map<String, String >，其中键值对代表预期的标签名称和值。

- getEvent ：此操作提供特定事件。

```
from("direct:get").process(new Processor() {

        @Override
        public void process(Exchange exchange) throws Exception {

exchange.getIn().setHeader(KubernetesConstants.KUBERNETES_NAMESPACE_NAME,
"test");
            exchange.getIn().setHeader(KubernetesConstants.KUBERNETES_EVENT_NAME,
"event1");
        }
    });
    to("kubernetes-events:///?kubernetesClient=#kubernetesClient&operation=getEvent").
    to("mock:result");
```

此操作返回与集群中条件匹配的事件。事件的类型是 io.fabric8.kubernetes.api.model.events.v1.Event。

此操作需要两个消息标头（即 CamelKubernetesNamespaceName 和 CamelKubernetesEventName ），第一个标头需要设置为目标命名空间的名称，需要设置为事件的目标名称。

如果无法找到匹配的事件，则返回 null。

- createEvent ：此操作会创建一个新事件。

```
from("direct:get").process(new Processor() {

        @Override
        public void process(Exchange exchange) throws Exception {
```

```
exchange.getIn().setHeader(KubernetesConstants.KUBERNETES_NAMESPACE_NAME,
"default");
            exchange.getIn().setHeader(KubernetesConstants.KUBERNETES_EVENT_NAME,
"test1");
            Map<String, String> labels = new HashMap<>();
            labels.put("this", "rocks");
            exchange.getIn().setHeader(KubernetesConstants.KUBERNETES_EVENTS_LABELS,
labels);

exchange.getIn().setHeader(KubernetesConstants.KUBERNETES_EVENT_ACTION_PRODUCE
R, "Some Action");
            exchange.getIn().setHeader(KubernetesConstants.KUBERNETES_EVENT_TYPE,
"Normal");
            exchange.getIn().setHeader(KubernetesConstants.KUBERNETES_EVENT_REASON,
"Some Reason");

exchange.getIn().setHeader(KubernetesConstants.KUBERNETES_EVENT_REPORTING_CONT
ROLLER, "Some-Reporting-Controller");

exchange.getIn().setHeader(KubernetesConstants.KUBERNETES_EVENT_REPORTING_INSTA
NCE, "Some-Reporting-Instance");
        }
    });
  to("kubernetes-events:///?kubernetesClient=#kubernetesClient&operation=createEvent").
  to("mock:result");
```

此操作会在集群中发布新事件。可以通过两种方法从消息标头创建事件，或者从 io.fabric8.kubernetes.api.model.events.v1.EventBuilder 处创建。

创建事件的任何方法：

- 该操作需要两个消息标头，它们是 CamelKubernetesNamespaceName 和 CamelKubernetesEventName，以分别设置命名空间的名称以及生成的事件的名称。
- 该操作支持消息标头 CamelKubernetesEventsLabels，将标签设置为生成的事件。

用于创建事件的消息标头是 CamelKubernetesEventTime,CamelKubernetesEventAction,CamelKubernetesEventType, CamelKubernetesEventReason ,CamelKubernetesEventReason,CamelKubernetesEvent Regarding,CamelKubernetesEventRelated,CamelKubernetesEventReportingController 和 CamelKubernetesEventReportingInstance。

如果支持的消息标头不足以用于特定的用例，仍然可以使用类型为 io.fabric8.kubernetes.api.model.events.v1.EventBuilder 的对象来设置消息正文，以便在创建事件时

使用。请注意，标签、事件名称和命名空间名称始终从消息标头设置，即使提供了构建器。

- updateEvent ：此操作更新现有事件。

该行为与 createEvent 完全相同，只有操作的名称有所不同。

- DeleteEvent ：此操作会删除现有事件。

```
from("direct:get").process(new Processor() {

        @Override
        public void process(Exchange exchange) throws Exception {

exchange.getIn().setHeader(KubernetesConstants.KUBERNETES_NAMESPACE_NAME,
"default");
            exchange.getIn().setHeader(KubernetesConstants.KUBERNETES_EVENT_NAME,
"test1");
        }
    });
  to("kubernetes-events:///?kubernetesClient=#kubernetesClient&operation=deleteEvent").
  to("mock:result");
```

此操作会从集群中移除现有事件。它返回一个 布尔值，以指示操作是否成功。

此操作需要两个消息标头（即 CamelKubernetesNamespaceName 和 CamelKubernetesEventName ），第一个标头需要设置为目标命名空间的名称，需要设置为事件的目标名称。

65.8. KUBERNETES EVENTS CONSUMER 示例

```
fromF("kubernetes-events://%s?oauthToken=%s", host, authToken)
  .setHeader(KubernetesConstants.KUBERNETES_NAMESPACE_NAME, constant("default"))
  .setHeader(KubernetesConstants.KUBERNETES_EVENT_NAME, constant("test"))
  .process(new KubernertesProcessor()).to("mock:result");

  public class KubernertesProcessor implements Processor {
    @Override
    public void process(Exchange exchange) throws Exception {
      Message in = exchange.getIn();
      Event cm = exchange.getIn().getBody(Event.class);
      log.info("Got event with event name: " + cm.getMetadata().getName() + " and action " +
in.getHeader(KubernetesConstants.KUBERNETES_EVENT_ACTION));
    }
  }
```

此消费者返回事件 "default" 在命名空间 "default" 上收到的每个事件的消息。它还在消息标头 CamelKubernetesEventAction 和消息标头 CamelKubernetesEventTimestamp 中设置操作 (io.fabric8.kubernetes.client.Watcher.Action)。

65.9. SPRING BOOT AUTO-CONFIGURATION

组件支持 102 选项，如下所列。

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å�Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串

Name	描述	默认值	类型
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-config-maps.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-config-maps.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

Name	描述	默认值	类型
camel.component.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-deployments.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

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
camel.component.kubernetes-deployments.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-events.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

Name	描述	默认值	类型
camel.component.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-hpa.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.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
camel.component.kubernetes-hpa.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-job.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

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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-namespaces.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

Name	描述	默认值	类型
camel.component.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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
camel.component.kubernetes-replication-controllers.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-replication-controllers.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

Name	描述	默认值	类型
camel.component.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-secrets.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

Name	描述	默认值	类型
camel.component.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-services.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.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
camel.component.kubernetes-services.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-deploymentconfigs.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

第 66 章 KUBERNETES HPA

从 Camel 2.23 开始

支持生成者和消费者

Kubernetes HPA 组件是 Kubernetes 组件 之一，它提供了一个生成者来执行 kubernetes Horizontal Pod Autoscaler 操作，以及一个消费者，以使用与 Horizontal Pod Autoscaler 对象相关的事件。

66.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 kubernetes-hpa 时，使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

66.2. 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

66.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

66.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

66.3. 组件选项

Kubernetes HPA 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (common)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

66.4. 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：

66.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterUrl (common)	**所需的** Kubernetes 主 url。		字符串

66.4.2. 查询参数(33 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (common)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (common)	命名空间。		字符串
portName (common)	用于 ServiceCall EIP 的端口名称。		字符串

Name	描述	默认值	类型
portProtocol (common)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
crdGroup (consumer)	您要监视的 Consumer CRD 资源组。		字符串
crdName (consumer)	我们想监视的 Consumer CRD 资源名称。		字符串
crdPlural (consumer)	Consumer CRD Resource Plural we like to watch。		字符串
crdScope (consumer)	Consumer CRD 资源范围我们想监视。		字符串
crdVersion (consumer)	我们想监视的 Consumer CRD 资源版本。		字符串
labelKey (consumer)	当监视某些资源时，Consumer Label 键。		字符串
labelValue (consumer)	Consumer Label 值在监视某些资源时。		字符串
poolSize (consumer)	Consumer 池大小。	1	int
resourceName (consumer)	要监视的消费者资源名称。		字符串
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut		ExchangePattern
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串

Name	描述	默认值	类型
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

66.5. 消息标头

***Kubernetes HPA** 组件支持 7 个消息标头，如下所列：*

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNamespaceName (producer) 常量 ：KUBERNETES_NAMESPACE_NAME	命名空间名称。		字符串
CamelKubernetesHPAName (producer) 常量 ：KUBERNETES_HPA_NAME	HPA 名称。		字符串
CamelKubernetesHPASpec (producer) 恒定：KUBERNETES_HPA_SPEC	HPA 的 spec。		HorizontalPodAutoscalerSpec

Name	描述	默认值	类型
CamelKubernetesHPALabels (producer) 恒定：KUBERNETES_HPA_LABELS	HPA 标签。		Map
CamelKubernetesEventAction (consumer) 常量 ：KUBERNETES_EVENT_ACTION	消费者监视的操作。 Enum 值： • ADDED • MODIFIED • 已删除 • ERROR • 书签		操作
CamelKubernetesEventTimestamp (consumer) 恒定：KUBERNETES_EVENT_TIMESTAMP	消费者监视的操作的时间戳。		long

66.6. 支持的制作者操作

- ***listHPA***
- ***listHPAByLabels***
- ***getHPA***
- ***createHPA***

- updateHPA

- deleteHPA

66.7. KUBERNETES HPA PRODUCER 示例

- ListHPA：此操作列出了 kubernetes 集群上的 HPA。

```
from("direct:list").
   toF("kubernetes-hpa:///?kubernetesClient=#kubernetesClient&operation=listHPA").
   to("mock:result");
```

此操作会返回集群中的 HPA 列表。

- listDeploymentsByLabels：此操作根据 kubernetes 集群上的标签列出 HPA。

```
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_HPA_LABELS,
labels);
        }
     });
   toF("kubernetes-hpa:///?
kubernetesClient=#kubernetesClient&operation=listHPAByLabels").
   to("mock:result");
```

此操作使用标签选择器（值为 key1 和 key2）从集群中返回 HPAs 列表。

66.8. KUBERNETES HPA CONSUMER 示例

```
fromF("kubernetes-hpa://%s?oauthToken=%s&namespace=default&resourceName=test",
host, authToken).process(new KubernertesProcessor()).to("mock:result");
   public class KubernertesProcessor implements Processor {
      @Override
      public void process(Exchange exchange) throws Exception {
         Message in = exchange.getIn();
         HorizontalPodAutoscaler hpa =
exchange.getIn().getBody(HorizontalPodAutoscaler.class);
```

```
            log.info("Got event with hpa name: " + hpa.getMetadata().getName() + " and action " +
in.getHeader(KubernetesConstants.KUBERNETES_EVENT_ACTION));
        }
    }
```

此消费者返回 hpa 测试的命名空间 default 上的事件列表。

66.9. SPRING BOOT AUTO-CONFIGURATION

组件支持 102 选项，如下所列。

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å�Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long

Name	描述	默认值	类型
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-config-maps.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-config-maps.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.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-deployments.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.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
camel.component.kubernetes-deployments.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-events.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.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-hpa.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.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
camel.component.kubernetes-hpa.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-job.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-namespaces.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.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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

Name	描述	默认值	类型
camel.component.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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.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
camel.component.kubernetes-pods.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-replication-controllers.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.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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

Name	描述	默认值	类型
camel.component.kubernetes-secrets.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-secrets.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.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-services.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.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
camel.component.kubernetes-services.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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

Name	描述	默认值	类型
camel.component.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.openshift-deploymentconfigs.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

第 67 章 KUBERNETES 任务

从 Camel 2.23 开始

支持生成者和消费者

Kubernetes 作业组件是 Kubernetes 组件的一员，它提供生成者来执行 kubernetes 作业操作和消费者，以使用与作业对象相关的事件。

67.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 kubernetes-job 时，使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

67.2. 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

67.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

67.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

67.3. 组件选项

Kubernetes 作业组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (common)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

67.4. 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：

67.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterUrl (common)	**所需的** Kubernetes 主 url。		字符串

67.4.2. 查询参数(33 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (common)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (common)	命名空间。		字符串
portName (common)	用于 ServiceCall EIP 的端口名称。		字符串

Name	描述	默认值	类型
portProtocol (common)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
crdGroup (consumer)	您要监视的 Consumer CRD 资源组。		字符串
crdName (consumer)	我们想监视的 Consumer CRD 资源名称。		字符串
crdPlural (consumer)	Consumer CRD Resource Plural we like to watch。		字符串
crdScope (consumer)	Consumer CRD 资源范围我们想监视。		字符串
crdVersion (consumer)	我们想监视的 Consumer CRD 资源版本。		字符串
labelKey (consumer)	当监视某些资源时，Consumer Label 键。		字符串
labelValue (consumer)	Consumer Label 值在监视某些资源时。		字符串
poolSize (consumer)	Consumer 池大小。	1	int
resourceName (consumer)	要监视的消费者资源名称。		字符串
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut		ExchangePattern
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串

Name	描述	默认值	类型
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

67.5. 消息标头

Kubernetes 作业组件支持 5 个消息标头，如下所列：

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNamespaceName (producer) 常量 ：KUBERNETES_NAMESPACE_NAME	命名空间名称。		字符串
CamelKubernetesJobName (producer) 恒定：KUBERNETES_JOB_NAME	作业名称。		字符串
CamelKubernetesJobSpec (producer) 恒定：KUBERNETES_JOB_SPEC	作业的 spec。		JobSpec

Name	描述	默认值	类型
CamelKubernetesJobLabels (producer) 恒定：KUBERNETES_JOB_LABELS	Job 标签。		Map

67.6. 支持的制作者操作

- ***listJob***
- ***listJobByLabels***
- ***getJob***
- ***createJob***
- ***updateJob***
- ***deleteJob***

67.7. KUBERNETES 任务 PRODUCER 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");
```

此操作会返回集群中的作业列表。

- listJobByLabels：此操作通过 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）从集群中返回一个作业列表。

- CreateJob：此操作在 Kubernetes 集群上创建一个作业。

示例（请参阅 创建作业示例 以了解更多信息）

```
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_NAME,
"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;
    }
}
```

67.8. SPRING BOOT AUTO-CONFIGURATION

组件支持 102 选项，如下所列。

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值

Name	描述	默认值	类型
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å�Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-config-maps.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

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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-config-maps.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.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-deployments.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.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
camel.component.kubernetes-deployments.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-events.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.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-hpa.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.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
camel.component.kubernetes-hpa.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-job.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-namespaces.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.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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

Name	描述	默认值	类型
camel.component.kubernetes-persistent-volumes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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.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
camel.component.kubernetes-pods.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-replication-controllers.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.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
camel.component.kubernetes-replication-controllers.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-replication-controllers.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.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-secrets.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.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-services.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.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
camel.component.kubernetes-services.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 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.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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

Name	描述	默认值	类型
camel.component.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-deploymentconfigs.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

第 68 章 KUBERNETES 命名空间

从 Camel 2.17 开始

支持生成者和消费者

Kubernetes 命名空间组件是 Kubernetes 组件 之一，它提供了一个制作者来执行 Kubernetes 命名空间操作，以及消费者使用与 Namespace 事件相关的事件。

68.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 kubernetes-namespaces 时，使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

68.2. 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

68.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

68.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

68.3. 组件选项

Kubernetes 命名空间组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (common)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

68.4. 端点选项

Kubernetes 命名空间端点使用 URI 语法进行配置：

```
kubernetes-namespaces:masterUrl
```

使用以下路径和查询参数：

68.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterUrl (common)	**所需的** Kubernetes 主 url。		字符串

68.4.2. 查询参数(33 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (common)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (common)	命名空间。		字符串
portName (common)	用于 ServiceCall EIP 的端口名称。		字符串

Name	描述	默认值	类型
portProtocol (common)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
crdGroup (consumer)	您要监视的 Consumer CRD 资源组。		字符串
crdName (consumer)	我们想监视的 Consumer CRD 资源名称。		字符串
crdPlural (consumer)	Consumer CRD Resource Plural we like to watch。		字符串
crdScope (consumer)	Consumer CRD 资源范围我们想监视。		字符串
crdVersion (consumer)	我们想监视的 Consumer CRD 资源版本。		字符串
labelKey (consumer)	当监视某些资源时，Consumer Label 键。		字符串
labelValue (consumer)	Consumer Label 值在监视某些资源时。		字符串
poolSize (consumer)	Consumer 池大小。	1	int
resourceName (consumer)	要监视的消费者资源名称。		字符串
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut		ExchangePattern
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串

Name	描述	默认值	类型
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

68.5. 消息标头

Kubernetes Namespaces 组件支持 5 个消息标头，如下所列：

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNamespaceName (producer) 常量 ：KUBERNETES_NAMESPACE_NAME	命名空间名称。		字符串
CamelKubernetesNamespaceLabels (producer) 恒定：KUBERNETES_NAMESPACE_LABELS	命名空间标签。		Map

Name	描述	默认值	类型
CamelKubernetesEventAction (consumer) 常量 ：KUBERNETES_EVENT_ACTION	消费者监视的操作。 Enum 值： • ADDED • MODIFIED • 已删除 • ERROR • 书签		操作
CamelKubernetesEventTimestamp (consumer) 恒定：KUBERNETES_EVENT_TIMESTAMP	消费者监视的操作的时间戳。		long

68.6. 支持的制作者操作

- ***listNamespaces***
- ***listNamespacesByLabels***
- ***getNamespace***
- ***createNamespace***
- ***updateNamespace***
- ***deleteNamespace***

68.7. KUBERNETES 命名空间 PRODUCER 示例

- ListNamespaces ：此操作列出了 kubernetes 集群中的命名空间。

```
from("direct:list").
   toF("kubernetes-deployments:///?
kubernetesClient=#kubernetesClient&operation=listNamespaces").
   to("mock:result");
```

此操作会返回集群中的命名空间列表。

- listNamespacesByLabels：此操作通过 kubernetes 集群上的标签列出命名空间。

```
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_NAMESPACES_LABELS,
labels);
        }
     });
   toF("kubernetes-deployments:///?
kubernetesClient=#kubernetesClient&operation=listNamespacesByLabels").
   to("mock:result");
```

此操作使用标签选择器（值为 key1 和 key2）返回集群中的命名空间列表。

68.8. KUBERNETES NAMESPACES CONSUMER 示例

```
fromF("kubernetes-namespaces://%s?oauthToken=%s&namespace=default", host,
authToken).process(new KubernertesProcessor()).to("mock:result");
   public class KubernertesProcessor implements Processor {
      @Override
      public void process(Exchange exchange) throws Exception {
         Message in = exchange.getIn();
         Namespace ns = exchange.getIn().getBody(Namespace.class);
         log.info("Got event with configmap name: " + ns.getMetadata().getName() + " and action
" + in.getHeader(KubernetesConstants.KUBERNETES_EVENT_ACTION));
      }
   }
```

此消费者返回命名空间 default 上的事件列表。

68.9. SPRING BOOT AUTO-CONFIGURATION

组件支持 102 选项，如下所列。

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å�Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数

Name	描述	默认值	类型
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-config-maps.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-config-maps.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.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-deployments.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-events.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.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-hpa.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.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
camel.component.kubernetes-hpa.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-job.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-namespaces.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.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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
camel.component.kubernetes-replication-controllers.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-replication-controllers.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

Name	描述	默认值	类型
camel.component.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-secrets.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

Name	描述	默认值	类型
camel.component.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-services.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.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
camel.component.kubernetes-services.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-deploymentconfigs.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

第 69 章 KUBERNETES 节点

从 Camel 2.17 开始

支持生成者和消费者

Kubernetes 节点组件是 Kubernetes 组件 之一，它为执行 Kubernetes 节点操作和消费者使用与 Node 对象相关的事件提供了一个制作者。

69.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 kubernetes-nodes 时，使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

69.2. 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

69.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

69.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

69.3. 组件选项

Kubernetes 节点组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (common)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

69.4. 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：

69.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterUrl (common)	**所需的** Kubernetes 主 url。		字符串

69.4.2. 查询参数(33 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (common)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (common)	命名空间。		字符串
portName (common)	用于 ServiceCall EIP 的端口名称。		字符串

Name	描述	默认值	类型
portProtocol (common)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
crdGroup (consumer)	您要监视的 Consumer CRD 资源组。		字符串
crdName (consumer)	我们想监视的 Consumer CRD 资源名称。		字符串
crdPlural (consumer)	Consumer CRD Resource Plural we like to watch。		字符串
crdScope (consumer)	Consumer CRD 资源范围我们想监视。		字符串
crdVersion (consumer)	我们想监视的 Consumer CRD 资源版本。		字符串
labelKey (consumer)	当监视某些资源时，Consumer Label 键。		字符串
labelValue (consumer)	Consumer Label 值在监视某些资源时。		字符串
poolSize (consumer)	Consumer 池大小。	1	int
resourceName (consumer)	要监视的消费者资源名称。		字符串
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut		ExchangePattern
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串

Name	描述	默认值	类型
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

69.5. 消息标头

Kubernetes Nodes 组件支持 6 个消息标头，如下所列：

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNodesLabels (producer) 恒定：KUBERNETES_NODES_LABELS	节点标签。		Map
CamelKubernetesNodeName (producer) 常量 ：KUBERNETES_NODE_NAME	节点名称。		字符串
CamelKubernetesNodeSpec (producer) 常量 ：KUBERNETES_NODE_SPEC	节点的 spec。		NodeSpec

Name	描述	默认值	类型
CamelKubernetesEventAction (consumer) 常量 ：KUBERNETES_EVENT_ACTION	消费者监视的操作。 Enum 值： • ADDED • MODIFIED • 已删除 • ERROR • 书签		操作
CamelKubernetesEventTimestamp (consumer) 恒定：KUBERNETES_EVENT_TIMESTAMP	消费者监视的操作的时间戳。		long

69.6. 支持的制作者操作

- ***listNodes***
- ***listNodesByLabels***
- ***getNode***
- ***createNode***
- ***updateNode***
- ***deleteNode***

69.7. KUBERNETES 节点生成者示例

- listNodes：此操作列出了 kubernetes 集群中的节点。

```
from("direct:list").
    toF("kubernetes-nodes:///?kubernetesClient=#kubernetesClient&operation=listNodes").
    to("mock:result");
```

此操作会从集群中返回节点列表。

- listNodesByLabels：此操作通过 kubernetes 集群上的标签列出节点。

```
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_NODES_LABELS,
labels);
            }
        });
    toF("kubernetes-deployments:///?
kubernetesClient=#kubernetesClient&operation=listNodesByLabels").
    to("mock:result");
```

此操作使用标签选择器（值为 key1 和 key2）从集群中返回节点列表。

69.8. KUBERNETES 节点消费者示例

```
fromF("kubernetes-nodes://%s?oauthToken=%s&resourceName=test", host,
authToken).process(new KubernertesProcessor()).to("mock:result");
    public class KubernertesProcessor implements Processor {
        @Override
        public void process(Exchange exchange) throws Exception {
            Message in = exchange.getIn();
            Node node = exchange.getIn().getBody(Node.class);
            log.info("Got event with configmap name: " + node.getMetadata().getName() + " and
action " + in.getHeader(KubernetesConstants.KUBERNETES_EVENT_ACTION));
        }
    }
```

此消费者返回节点测试的事件列表。

69.9. SPRING BOOT AUTO-CONFIGURATION

*组**件支持 102** 选项，**如下所列。***

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å�Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串

Name	描述	默认值	类型
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-config-maps.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-config-maps.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.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-deployments.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-events.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

Name	描述	默认值	类型
camel.component.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-hpa.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.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
camel.component.kubernetes-hpa.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-job.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.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

Name	描述	默认值	类型
camel.component.kubernetes-job.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-namespaces.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

Name	描述	默认值	类型
camel.component.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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

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
camel.component.kubernetes-pods.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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
camel.component.kubernetes-replication-controllers.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-replication-controllers.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

Name	描述	默认值	类型
camel.component.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-secrets.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

Name	描述	默认值	类型
camel.component.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-services.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.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
camel.component.kubernetes-services.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-deploymentconfigs.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

第 70 章 KUBERNETES 持久性卷

从 Camel 2.17 开始

仅支持生成者

Kubernetes 持久性卷组件是 Kubernetes 组件之一，它为执行 Kubernetes 持久性卷操作提供一个制作者。

70.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 kubernetes-persistent-volumes 时，请使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

70.2. 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

70.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

70.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

70.3. 组件选项

Kubernetes 持久性卷组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (producer)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

70.4. 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：

70.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterURL (producer)	**所需的** Kubernetes 主 url。		字符串

70.4.2. 查询参数(21 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (producer)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (producer)	命名空间。		字符串
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
PORTNAME (producer)	用于 ServiceCall EIP 的端口名称。		字符串
portProtocol (producer)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

70.5. 消息标头

Kubernetes 持久性卷组件支持 3 个消息标头，如下所列：

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesPersistentVolumesLabels (producer) 常数 ：KUBERNETES_PERSISTENT_VOLUMES_LABELS	持久性卷标签。		Map
CamelKubernetesPersistentVolumeName (producer) 常数 ：KUBERNETES_PERSISTENT_VOLUME_NAME	持久性卷名称。		字符串

70.6. 支持的制作者操作

- ***listPersistentVolumes***
- ***listPersistentVolumesByLabels***
- ***getPersistentVolume***

70.7. KUBERNETES 持久性卷 PRODUCER 示例

- ***ListPersistentVolume：此操作列出了 kubernetes 集群上的 pv。***

```
from("direct:list").
    toF("kubernetes-persistent-volumes:///?
```

```
kubernetesClient=#kubernetesClient&operation=listPersistentVolumes").
    to("mock:result");
```

此操作会返回集群中的 pv 列表。

- listPersistentVolumesByLabels：此操作在 kubernetes 集群中按标签列出 pv

```
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_PERSISTENT_VOLUMES_L
ABELS, labels);
        }
    });
    toF("kubernetes-persistent-volumes:///?
kubernetesClient=#kubernetesClient&operation=listPersistentVolumesByLabels").
    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 pv 列表。

70.8. SPRING BOOT AUTO-CONFIGURATION

组件支持 102 选项，如下所列。

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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

Name	描述	默认值	类型
camel.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å€Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-config-maps.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-config-maps.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.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 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-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-deployments.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.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
camel.component.kubernetes-deployments.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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

Name	描述	默认值	类型
camel.component.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-events.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.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-hpa.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-job.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-job.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

Name	描述	默认值	类型
camel.component.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-namespaces.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.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-nodes.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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.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
camel.component.kubernetes-pods.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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

Name	描述	默认值	类型
camel.component.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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
camel.component.kubernetes-replication-controllers.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-replication-controllers.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.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-secrets.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.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-services.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.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
camel.component.kubernetes-services.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 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.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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

Name	描述	默认值	类型
camel.component.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-deploymentconfigs.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

第 71 章 KUBERNETES 持久性卷声明

从 Camel 2.17 开始

仅支持生成者

Kubernetes 持久性卷声明组件是 Kubernetes 组件之一，它为执行 Kubernetes 持久性卷声明操作提供一个制作者。

71.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 kubernetes-persistent-volumes-claims 时，使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

71.2. 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

71.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

71.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

71.3. 组件选项

Kubernetes 持久性卷声明组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (producer)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

71.4. 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：

71.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterURL (producer)	**所需的** Kubernetes 主 url。		字符串

71.4.2. 查询参数(21 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (producer)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (producer)	命名空间。		字符串
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
PORTNAME (producer)	用于 ServiceCall EIP 的端口名称。		字符串
portProtocol (producer)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

71.5. 消息标头

Kubernetes 持久性卷声明组件支持 5 个消息标头，如下所列：

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNamespaceName (producer) 常量 ：KUBERNETES_NAMESPACE_NAME	命名空间名称。		字符串
CamelKubernetesPersistentVolumesClaimsLabels (producer) 恒定：KUBERNETES_PERSISTENT_VOLUMES_CLAIMS_LABELS	持久性卷声明标签。		Map
CamelKubernetesPersistentVolumeClaimName (producer) 常量 ：KUBERNETES_PERSISTENT_VOLUME_CLAIM_NAME	持久性卷声明名称。		字符串
CamelKubernetesPersistentVolumeClaimSpec (producer) 常量 ：KUBERNETES_PERSISTENT_VOLUME_CLAIM_SPEC	持久性卷声明的 spec。		PersistentVolumeClaimSpec

71.6. 支持的制作者操作

- listPersistentVolumesClaims
- listPersistentVolumesClaimsByLabels
- getPersistentVolumeClaim
- createPersistentVolumeClaim
- updatePersistentVolumeClaim
- deletePersistentVolumeClaim

71.7. KUBERNETES 持久性卷声明 PRODUCER 示例

- listPersistentVolumesClaims：此操作列出了 kubernetes 集群上的 pvc。

```
from("direct:list").
   toF("kubernetes-persistent-volumes-claims:///?
kubernetesClient=#kubernetesClient&operation=listPersistentVolumesClaims").
   to("mock:result");
```

此操作会返回集群中的 pvc 列表。

- listPersistentVolumesClaimsByLabels：此操作通过 kubernetes 集群上的标签列出 pvc。

```
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_PERSISTENT_VOLUMES_C
LAIMS_LABELS, labels);
        }
     });
```

```
    toF("kubernetes-persistent-volumes-claims:///?
kubernetesClient=#kubernetesClient&operation=listPersistentVolumesClaimsByLabels").
    to("mock:result");
```

此操作使用标签选择器（值为 value1 和 key2）从集群中返回 pvc 列表。

71.8. SPRING BOOT AUTO-CONFIGURATION

组件支持 102 选项，如下所列。

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å�Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long

Name	描述	默认值	类型
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-config-maps.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-config-maps.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.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-deployments.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.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
camel.component.kubernetes-deployments.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-events.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.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-hpa.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.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
camel.component.kubernetes-hpa.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-job.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-namespaces.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.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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

Name	描述	默认值	类型
camel.component.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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.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
camel.component.kubernetes-pods.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-replication-controllers.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.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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

Name	描述	默认值	类型
camel.component.kubernetes-secrets.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-secrets.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.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-services.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.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
camel.component.kubernetes-services.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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

Name	描述	默认值	类型
camel.component.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.openshift-deploymentconfigs.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

第 72 章 KUBERNETES POD

从 Camel 2.17 开始

支持生成者和消费者

Kubernetes Pod 组件是 Kubernetes 组件 之一，它为执行 Kubernetes Pod 操作和消费者使用与 Pod 对象相关的事件提供了一个制作者。

72.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 kubernetes-pods 时，使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

72.2. 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

72.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

72.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

72.3. 组件选项

Kubernetes Pod 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (common)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

72.4. 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：

72.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterUrl (common)	**所需的** Kubernetes 主 url。		字符串

72.4.2. 查询参数(33 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (common)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (common)	命名空间。		字符串
portName (common)	用于 ServiceCall EIP 的端口名称。		字符串

Name	描述	默认值	类型
portProtocol (common)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
crdGroup (consumer)	您要监视的 Consumer CRD 资源组。		字符串
crdName (consumer)	我们想监视的 Consumer CRD 资源名称。		字符串
crdPlural (consumer)	Consumer CRD Resource Plural we like to watch。		字符串
crdScope (consumer)	Consumer CRD 资源范围我们想监视。		字符串
crdVersion (consumer)	我们想监视的 Consumer CRD 资源版本。		字符串
labelKey (consumer)	当监视某些资源时，Consumer Label 键。		字符串
labelValue (consumer)	Consumer Label 值在监视某些资源时。		字符串
poolSize (consumer)	Consumer 池大小。	1	int
resourceName (consumer)	要监视的消费者资源名称。		字符串
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut		ExchangePattern
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串

Name	描述	默认值	类型
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

72.5. 消息标头

Kubernetes Pod 组件支持 7 个消息标头，如下所列：

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNamespaceName (producer) 常量 ：KUBERNETES_NAMESPACE_NAME	命名空间名称。		字符串
CamelKubernetesPodsLabels (producer) 恒定：KUBERNETES_PODS_LABELS	pod 标签。		Map
CamelKubernetesPodName (producer) 常量 ：KUBERNETES_POD_NAME	pod 名称。		字符串

Name	描述	默认值	类型
CamelKubernetesPodSpec (producer) 常量 ：KUBERNETES_POD_SPEC	pod 的 spec。		PodSpec
CamelKubernetesEventAction (consumer) 常量 ：KUBERNETES_EVENT_ACTION	消费者监视的操作。 Enum 值： • ADDED • MODIFIED • 已删除 • ERROR • 书签		操作
CamelKubernetesEventTimestamp (consumer) 恒定：KUBERNETES_EVENT_TIMESTAMP	消费者监视的操作的时间戳。		long

72.6. 支持的制作者操作

- ***listPods***
- ***listPodsByLabels***
- ***getPod***
- ***createPod***

- updatePod

- deletePod

72.7. KUBERNETES POD PRODUCER 示例

- listPods：此操作列出了 kubernetes 集群中的 pod。

```
from("direct:list").
   toF("kubernetes-pods:///?kubernetesClient=#kubernetesClient&operation=listPods").
   to("mock:result");
```

此操作会返回集群中的 Pod 列表。

- listPodsByLabels：此操作使用 kubernetes 集群上的标签列出 pod。

```
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_PODS_LABELS,
labels);
        }
     });
   toF("kubernetes-pods:///?
kubernetesClient=#kubernetesClient&operation=listPodsByLabels").
   to("mock:result");
```

此操作使用标签选择器（带有 key1 和 key2 的值2）返回集群中的 Pod 列表。

72.8. KUBERNETES PODS 消费者示例

```
fromF("kubernetes-pods://%s?oauthToken=%s&namespace=default&resourceName=test",
host, authToken).process(new KubernertesProcessor()).to("mock:result");
   public class KubernertesProcessor implements Processor {
      @Override
      public void process(Exchange exchange) throws Exception {
         Message in = exchange.getIn();
         Pod pod = exchange.getIn().getBody(Pod.class);
         log.info("Got event with configmap name: " + pod.getMetadata().getName() + " and
```

```
action " + in.getHeader(KubernetesConstants.KUBERNETES_EVENT_ACTION));
        }
    }
```

此消费者返回 pod 测试的命名空间 default 上的事件列表。

72.9. SPRING BOOT AUTO-CONFIGURATION

组件支持 102 选项，如下所列。

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å�Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long

Name	描述	默认值	类型
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-config-maps.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-config-maps.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.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-deployments.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.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
camel.component.kubernetes-deployments.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-events.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.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-hpa.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.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
camel.component.kubernetes-hpa.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-job.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-namespaces.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.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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

Name	描述	默认值	类型
camel.component.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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.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
camel.component.kubernetes-pods.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-replication-controllers.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.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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

Name	描述	默认值	类型
camel.component.kubernetes-secrets.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-secrets.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.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-services.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.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
camel.component.kubernetes-services.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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

Name	描述	默认值	类型
camel.component.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.openshift-deploymentconfigs.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

第 73 章 KUBERNETES 复制控制器

从 Camel 2.17 开始

支持生成者和消费者

Kubernetes Replication Controller 组件是一个 Kubernetes 组件，它提供了一个生成者来执行 Kubernetes Replication 控制器操作，以及一个消费者，以使用与 Replication Controller 对象相关的事件。

73.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 kubernetes-replication-controllers 时，使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

73.2. 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

73.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

73.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

73.3. 组件选项

Kubernetes Replication Controller 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (common)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

73.4. 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：

73.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterUrl (common)	**所需的** Kubernetes 主 url。		字符串

73.4.2. 查询参数(33 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (common)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (common)	命名空间。		字符串
portName (common)	用于 ServiceCall EIP 的端口名称。		字符串

Name	描述	默认值	类型
portProtocol (common)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
crdGroup (consumer)	您要监视的 Consumer CRD 资源组。		字符串
crdName (consumer)	我们想监视的 Consumer CRD 资源名称。		字符串
crdPlural (consumer)	Consumer CRD Resource Plural we like to watch。		字符串
crdScope (consumer)	Consumer CRD 资源范围我们想监视。		字符串
crdVersion (consumer)	我们想监视的 Consumer CRD 资源版本。		字符串
labelKey (consumer)	当监视某些资源时，Consumer Label 键。		字符串
labelValue (consumer)	Consumer Label 值在监视某些资源时。		字符串
poolSize (consumer)	Consumer 池大小。	1	int
resourceName (consumer)	要监视的消费者资源名称。		字符串
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut		ExchangePattern
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串

Name	描述	默认值	类型
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

73.5. 消息标头

Kubernetes Replication Controller 组件支持 8 个消息标头，如下所列：

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNamespaceName (producer) 常量 ：KUBERNETES_NAMESPACE_NAME	命名空间名称。		字符串
CamelKubernetesReplicationControllersLabels (producer) 恒定：KUBERNETES_REPLICATION_CONTROLLERS_LABELS	复制控制器标签。		Map

Name	描述	默认值	类型
CamelKubernetesReplicationControllerName (producer) 常量 ：KUBERNETES_REPLICATION_CONTROLLER_NAME	复制控制器名称。		字符串
CamelKubernetesReplicationControllerSpec (producer) 恒定：KUBERNETES_REPLICATION_CONTROLLER_SPEC	复制控制器的 spec。		ReplicationControllerSpec
CamelKubernetesReplicationControllerReplicas (producer) 恒定：KUBERNETES_REPLICATION_CONTROLLER_REPLICAS	在扩展操作过程中复制控制器的副本数。		整数
CamelKubernetesEventAction (consumer) 常量 ：KUBERNETES_EVENT_ACTION	消费者监视的操作。 Enum 值： • ADDED • MODIFIED • 已删除 • ERROR • 书签		操作
CamelKubernetesEventTimestamp (consumer) 恒定：KUBERNETES_EVENT_TIMESTAMP	消费者监视的操作的时间戳。		long

73.6. 支持的制作者操作

- listReplicationControllers
- listReplicationControllersByLabels
- getReplicationController
- createReplicationController
- updateReplicationController
- deleteReplicationController
- scaleReplicationController

73.7. KUBERNETES REPLICATION CONTROLLER PRODUCER 示例

- ListReplicationControllers：此操作列出了 kubernetes 集群中的 RC。

```
from("direct:list").
    toF("kubernetes-replication-controllers:///?
kubernetesClient=#kubernetesClient&operation=listReplicationControllers").
    to("mock:result");
```

此操作会返回集群中的 RC 列表。

- listReplicationControllersByLabels：此操作会根据 kubernetes 集群上的标签列出 RC。

```
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_REPLICATION_CONTROLL
ERS_LABELS, labels);
            }
        });
    toF("kubernetes-replication-controllers:///?
kubernetesClient=#kubernetesClient&operation=listReplicationControllersByLabels").
    to("mock:result");
```

此操作使用标签选择器（带有 key1 和 key2 的值2）返回集群中的 RC 列表。

73.8. KUBERNETES REPLICATION CONTROLLERS CONSUMER 示例

```
fromF("kubernetes-replication-controllers://%s?
oauthToken=%s&namespace=default&resourceName=test", host, authToken).process(new
KubernertesProcessor()).to("mock:result");
    public class KubernertesProcessor implements Processor {
        @Override
        public void process(Exchange exchange) throws Exception {
            Message in = exchange.getIn();
            ReplicationController rc = exchange.getIn().getBody(ReplicationController.class);
            log.info("Got event with configmap name: " + rc.getMetadata().getName() + " and action
" + in.getHeader(KubernetesConstants.KUBERNETES_EVENT_ACTION));
        }
    }
```

此消费者返回 rc 测试的命名空间 default 上的事件列表。

73.9. SPRING BOOT AUTO-CONFIGURATION

组件支持 102 选项，如下所列。

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串

Name	描述	默认值	类型
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å�Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-config-maps.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-config-maps.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.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 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-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-deployments.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.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
camel.component.kubernetes-deployments.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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

Name	描述	默认值	类型
camel.component.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-events.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.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-hpa.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-job.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-namespaces.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.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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

Name	描述	默认值	类型
camel.component.kubernetes-persistent-volumes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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.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
camel.component.kubernetes-pods.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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

Name	描述	默认值	类型
camel.component.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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
camel.component.kubernetes-replication-controllers.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-replication-controllers.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.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-secrets.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.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-services.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.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
camel.component.kubernetes-services.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 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.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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

Name	描述	默认值	类型
camel.component.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-deploymentconfigs.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

第 74 章 KUBERNETES 资源配额

从 Camel 2.17 开始

仅支持生成者

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。

74.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 kubernetes-resources-quota 时，请使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

74.2. 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

74.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

74.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

74.3. 组件选项

Kubernetes Resources Quota 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (producer)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

74.4. 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：

74.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterURL (producer)	**所需的** Kubernetes 主 url。		字符串

74.4.2. 查询参数(21 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (producer)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (producer)	命名空间。		字符串
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
PORTNAME (producer)	用于 ServiceCall EIP 的端口名称。		字符串
portProtocol (producer)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

74.5. 消息标头

***Kubernetes Resources Quota** 组件支持 5 个消息标头，如下所列：*

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNamespaceName (producer) 常量 ：KUBERNETES_NAMESPACE_NAME	命名空间名称。		字符串
CamelKubernetesResourcesQuotaLabels (producer) 恒定：KUBERNETES_RESOURCES_QUOTA_LABELS	资源配额标签。		Map
CamelKubernetesResourcesQuotaName (producer) 常量 ：KUBERNETES_RESOURCES_QUOTA_NAME	资源配额名称。		字符串
CamelKubernetesResourceQuotaSpec (producer) 常量：KUBERNETES_RESOURCE_QUOTA_SPEC	资源配额的 spec。		ResourceQuotaSpec

74.6. 支持的制作者操作

- ***listResourcesQuota***

- listResourcesQuotaByLabels
- getResourcesQuota
- createResourcesQuota
- updateResourceQuota
- deleteResourcesQuota

74.7. KUBERNETES 资源配额生产示例

- listResourcesQuota：此操作列出了 kubernetes 集群上的 Resource Quotas。

```
from("direct:list").
    toF("kubernetes-resources-quota:///?
kubernetesClient=#kubernetesClient&operation=listResourcesQuota").
    to("mock:result");
```

此操作会返回集群中的资源配额列表。

- listResourcesQuotaByLabels：此操作通过 kubernetes 集群上的标签列出 Resource Quotas。

```
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_RESOURCES_QUOTA_LAB
ELS, labels);
            }
        });
    toF("kubernetes-resources-quota:///?
kubernetesClient=#kubernetesClient&operation=listResourcesQuotaByLabels").
    to("mock:result");
```

此操作使用标签选择器（带有 key1 和 key2 的值2）返回来自集群的 Resource Quotas 列表。

74.8. SPRING BOOT AUTO-CONFIGURATION

组件支持 102 选项，如下所列。

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å�Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串

Name	描述	默认值	类型
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-config-maps.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-config-maps.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

Name	描述	默认值	类型
camel.component.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-deployments.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

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
camel.component.kubernetes-deployments.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-events.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

Name	描述	默认值	类型
camel.component.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-hpa.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.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
camel.component.kubernetes-hpa.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-job.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.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

Name	描述	默认值	类型
camel.component.kubernetes-job.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-namespaces.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.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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

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
camel.component.kubernetes-pods.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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
camel.component.kubernetes-replication-controllers.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-replication-controllers.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.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-secrets.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.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-services.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 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.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-deploymentconfigs.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

第 75 章 KUBERNETES SECRET

从 Camel 2.17 开始

仅**支持生成者**

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。**

75.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 kubernetes-secrets 时，**使用以下 Maven 依**赖项**来支持自**动**配置：**

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

75.2. 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

75.2.1. 配置组件选项

组**件**级别**是最高**级别，**它包含端点**继**承的常**规**配置。例如，一个**组**件可能具有安全**设**置、用于身份**验证**的凭**证、**用于网**络连**接的 url 等等。**

某些组**件只有几个**选项，**其他**组**件可能会有**许多选项。**由于**组**件通常已配置了常用的默**认值，**因此通常只需要在**组**件上配置几个**选项，**或者根本不需要配置任何**选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

75.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

75.3. 组件选项

Kubernetes Secrets 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (producer)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

75.4. 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：

75.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterURL (producer)	**所需的** Kubernetes 主 url。		字符串

75.4.2. 查询参数(21 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (producer)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (producer)	命名空间。		字符串
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
PORTNAME (producer)	用于 ServiceCall EIP 的端口名称。		字符串
portProtocol (producer)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

75.5. 消息标头

***Kubernetes Secrets** 组件支持 5 个消息标头，如下所列：*

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNamespaceName (producer) 常量 ：KUBERNETES_NAMESPACE_NAME	命名空间名称。		字符串
CamelKubernetesSecretsLabels (producer) 恒定：KUBERNETES_SECRETS_LABELS	secret 标签。		Map
CamelKubernetesSecretName (producer) 恒定：KUBERNETES_SECRET_NAME	机密名称。		字符串
CamelKubernetesSecret (producer) 恒定：KUBERNETES_SECRET	一个 secret 对象。		Secret

75.6. 支持的制作者操作

- ***listSecrets***
- ***listSecretsByLabels***

- *getSecret*
- *createSecret*
- *updateSecret*
- *deleteSecret*

75.7. KUBERNETES SECRETS PRODUCER 示例

- *listSecrets：此操作列出了 kubernetes 集群中的 secret。*

```
from("direct:list").
   toF("kubernetes-secrets:///?kubernetesClient=#kubernetesClient&operation=listSecrets").
   to("mock:result");
```

此操作会返回集群中的 secret 列表。

- *listSecretsByLabels：此操作通过 kubernetes 集群上的标签列出 Secret。*

```
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_SECRETS_LABELS, labels);
        }
     });
   toF("kubernetes-secrets:///?
kubernetesClient=#kubernetesClient&operation=listSecretsByLabels").
   to("mock:result");
```

此操作使用标签选择器（带有 key1 和 key2 的值2）返回集群中的 Secret 列表。

75.8. SPRING BOOT AUTO-CONFIGURATION

*组**件支持 102** 选项，**如下所列。***

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å�Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串

Name	描述	默认值	类型
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-config-maps.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-config-maps.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.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-deployments.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-events.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

Name	描述	默认值	类型
camel.component.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-hpa.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.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
camel.component.kubernetes-hpa.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-job.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.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

Name	描述	默认值	类型
camel.component.kubernetes-job.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-namespaces.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

Name	描述	默认值	类型
camel.component.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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

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
camel.component.kubernetes-pods.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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
camel.component.kubernetes-replication-controllers.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-replication-controllers.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

Name	描述	默认值	类型
camel.component.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-secrets.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

Name	描述	默认值	类型
camel.component.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-services.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.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
camel.component.kubernetes-services.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-deploymentconfigs.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

第 76 章 KUBERNETES 服务帐户

从 Camel 2.17 开始

仅支持生成者

Kubernetes Service Account 组件是 Kubernetes 组件 之一，它为执行 Kubernetes 服务帐户操作提供一个制作者。

76.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 kubernetes-service-accounts 时，请使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

76.2. 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

76.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

76.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

76.3. 组件选项

Kubernetes Service Account 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (producer)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

76.4. 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：

76.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterURL (producer)	**所需的** Kubernetes 主 url。		字符串

76.4.2. 查询参数(21 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (producer)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (producer)	命名空间。		字符串
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
PORTNAME (producer)	用于 ServiceCall EIP 的端口名称。		字符串
portProtocol (producer)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

76.5. 消息标头

***Kubernetes Service Account** 组件支持 5 个消息标头，如下所列：*

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNamespaceName (producer) 常量 ：KUBERNETES_NAMESPACE_NAME	命名空间名称。		字符串
CamelKubernetesServiceAccountsLabels (producer) 恒定：KUBERNETES_SERVICE_ACCOUNTS_LABELS	服务帐户标签。		Map
CamelKubernetesServiceAccountName (producer) 常量 ：KUBERNETES_SERVICE_ACCOUNT_NAME	服务帐户名称。		字符串
CamelKubernetesServiceAccount (producer) 常量 ：KUBERNETES_SERVICE_ACCOUNT	服务帐户对象。		ServiceAccount

76.6. 支持的制作者操作

- ***listServiceAccounts***

- listServiceAccountsByLabels
- getServiceAccount
- createServiceAccount
- updateServiceAccount
- deleteServiceAccount

76.7. KUBERNETES SERVICEACCOUNTS PRODUCE 示例

- listServiceAccounts：此操作列出了 kubernetes 集群中的服务帐户。

```
from("direct:list").
    toF("kubernetes-service-accounts:///?
kubernetesClient=#kubernetesClient&operation=listServiceAccounts").
    to("mock:result");
```

此操作会返回集群中的服务列表。

- listServiceAccountsByLabels：此操作通过 kubernetes 集群上的标签列出服务帐户。

```
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_SERVICE_ACCOUNTS_LAB
ELS, labels);
            }
        });
    toF("kubernetes-service-accounts:///?
kubernetesClient=#kubernetesClient&operation=listServiceAccountsByLabels").
    to("mock:result");
```

此操作使用标签选择器（值为 key1 和 key2）从集群中返回服务列表。

76.8. SPRING BOOT AUTO-CONFIGURATION

组件支持 102 选项，如下所列。

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å‚Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串

Name	描述	默认值	类型
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-config-maps.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-config-maps.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

Name	描述	默认值	类型
camel.component.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-deployments.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

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
camel.component.kubernetes-deployments.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-events.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

Name	描述	默认值	类型
camel.component.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-hpa.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.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
camel.component.kubernetes-hpa.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-job.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.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

Name	描述	默认值	类型
camel.component.kubernetes-job.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-namespaces.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.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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

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
camel.component.kubernetes-pods.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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
camel.component.kubernetes-replication-controllers.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-replication-controllers.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.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-secrets.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.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-services.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 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.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-deploymentconfigs.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

第 77 章 KUBERNETES 服务

从 Camel 2.17 开始

支持生成者和消费者

Kubernetes Services 组件是 Kubernetes 组件 之一，它为执行 Kubernetes Service 操作和消费者使用与 Service 对象相关的事件提供了一个制作者。

77.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 kubernetes-services 时，使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

77.2. 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

77.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

77.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

77.3. 组件选项

Kubernetes Services 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (common)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

77.4. 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：

77.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterUrl (common)	**所需的** Kubernetes 主 url。		字符串

77.4.2. 查询参数(33 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (common)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (common)	命名空间。		字符串
portName (common)	用于 ServiceCall EIP 的端口名称。		字符串

Name	描述	默认值	类型
portProtocol (common)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
crdGroup (consumer)	您要监视的 Consumer CRD 资源组。		字符串
crdName (consumer)	我们想监视的 Consumer CRD 资源名称。		字符串
crdPlural (consumer)	Consumer CRD Resource Plural we like to watch。		字符串
crdScope (consumer)	Consumer CRD 资源范围我们想监视。		字符串
crdVersion (consumer)	我们想监视的 Consumer CRD 资源版本。		字符串
labelKey (consumer)	当监视某些资源时，Consumer Label 键。		字符串
labelValue (consumer)	Consumer Label 值在监视某些资源时。		字符串
poolSize (consumer)	Consumer 池大小。	1	int
resourceName (consumer)	要监视的消费者资源名称。		字符串
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut		ExchangePattern
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串

Name	描述	默认值	类型
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

77.5. 消息标头

Kubernetes Services 组件支持 7 个消息标头，如下所列：

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNamespaceName (producer) 常量 ：KUBERNETES_NAMESPACE_NAME	命名空间名称。		字符串
CamelKubernetesServiceLabels (producer) 恒定：KUBERNETES_SERVICE_LABELS	服务标签。		Map
CamelKubernetesServiceName (producer) 常量 ：KUBERNETES_SERVICE_NAME	服务名称。		字符串

Name	描述	默认值	类型
CamelKubernetesServiceSpec (producer) 恒定：KUBERNETES_SERVICE_SPEC	服务的 spec。		ServiceSpec
CamelKubernetesEventAction (consumer) 常量 ：KUBERNETES_EVENT_ACTION	消费者监视的操作。 Enum 值： • ADDED • MODIFIED • 已删除 • ERROR • 书签		操作
CamelKubernetesEventTimestamp (consumer) 恒定：KUBERNETES_EVENT_TIMESTAMP	消费者监视的操作的时间戳。		long

77.6. 支持的制作者操作

- ***listServices***
- ***listServicesByLabels***
- ***getService***
- ***createService***

- deleteService

77.7. KUBERNETES SERVICES PRODUCER 示例

- listServices：此操作列出了 kubernetes 集群中的服务。

```
from("direct:list").
   toF("kubernetes-services:///?kubernetesClient=#kubernetesClient&operation=listServices").
   to("mock:result");
```

此操作会返回集群中的服务列表。

- listServicesByLabels：此操作通过 kubernetes 集群上的标签列出部署。

```
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_SERVICE_LABELS, labels);
        }
     });
   toF("kubernetes-services:///?
kubernetesClient=#kubernetesClient&operation=listServicesByLabels").
   to("mock:result");
```

此操作使用标签选择器（值为 key1 和 key2）从集群中返回服务列表。

77.8. KUBERNETES SERVICES CONSUMER 示例

```
fromF("kubernetes-services://%s?oauthToken=%s&namespace=default&resourceName=test",
host, authToken).process(new KubernertesProcessor()).to("mock:result");

   public class KubernertesProcessor implements Processor {
      @Override
      public void process(Exchange exchange) throws Exception {
         Message in = exchange.getIn();
         Service sv = exchange.getIn().getBody(Service.class);
         log.info("Got event with configmap name: " + sv.getMetadata().getName() + " and action
" + in.getHeader(KubernetesConstants.KUBERNETES_EVENT_ACTION));
      }
   }
```

-

此消费者返回服务测试的命名空间 default 上的事件列表。

77.8.1. Spring Boot Auto-Configuration

组件支持 102 选项，如下所列。

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å‚Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串

Name	描述	默认值	类型
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-config-maps.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-config-maps.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

Name	描述	默认值	类型
camel.component.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-deployments.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

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
camel.component.kubernetes-deployments.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-events.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

Name	描述	默认值	类型
camel.component.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-hpa.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.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
camel.component.kubernetes-hpa.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-job.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.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

Name	描述	默认值	类型
camel.component.kubernetes-job.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-namespaces.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.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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

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
camel.component.kubernetes-pods.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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
camel.component.kubernetes-replication-controllers.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-replication-controllers.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.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-secrets.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.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-services.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 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.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-deploymentconfigs.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

第 78 章 OPENSHIFT 构建

从 Camel 2.17 开始

仅支持生成者

Openshift Builds 组件是 Kubernetes 组件 之一，它为执行 Openshift 构建操作提供一个制作者。

78.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 openshift-builds 时，使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

78.2. 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

78.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

78.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

78.3. 组件选项

Openshift 构建组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (producer)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

78.4. 端点选项

***Openshift** 构建端点使用 **URI** 语法进行配置：*

```
openshift-builds:masterUrl
```

使用以下路径和查询参数：

78.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterURL (producer)	**所需的** Kubernetes 主 url。		字符串

78.4.2. 查询参数(21 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (producer)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (producer)	命名空间。		字符串
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
PORTNAME (producer)	用于 ServiceCall EIP 的端口名称。		字符串
portProtocol (producer)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

78.5. 消息标头

Openshift Builds 组件支持 4 个消息标头，如下所列：

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNamespaceName (producer) 常量 ：KUBERNETES_NAMESPACE_NAME	命名空间名称。		字符串
CamelKubernetesBuildsLabels (producer) 恒定：KUBERNETES_BUILDS_LABELS	Openshift 构建标签。		Map
CamelKubernetesBuildName (producer) 常量 ：KUBERNETES_BUILD_NAME	Openshift 构建名称。		字符串

78.6. 支持的制作者操作

- ***listBuilds***
- ***listBuildsByLabels***
- ***getBuild***

78.7. OPENSHIFT BUILD PRODUCER 示例

- *ListBuilds：此操作列出了 Openshift 集群上的构建。*

```
from("direct:list").
    toF("openshift-builds:///?kubernetesClient=#kubernetesClient&operation=listBuilds").
    to("mock:result");
```

此操作返回 Openshift 集群中的构建列表。

- *listBuildsByLabels：此操作按 Openshift 集群上的标签列出构建。*

```
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_BUILDS_LABELS,
labels);
            }
        });
    toF("openshift-builds:///?
kubernetesClient=#kubernetesClient&operation=listBuildsByLabels").
    to("mock:result");
```

此操作使用标签选择器（带有 key1 和 key2 的值2）返回集群中的构建列表。

78.8. SPRING BOOT AUTO-CONFIGURATION

组件支持 102 选项，如下所列。

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串

Name	描述	默认值	类型
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å�Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-config-maps.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-config-maps.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.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 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-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-deployments.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.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
camel.component.kubernetes-deployments.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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

Name	描述	默认值	类型
camel.component.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-events.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.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-hpa.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-job.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-job.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

Name	描述	默认值	类型
camel.component.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-namespaces.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.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-nodes.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

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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes-claims.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

Name	描述	默认值	类型
camel.component.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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.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
camel.component.kubernetes-pods.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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
camel.component.kubernetes-replication-controllers.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-replication-controllers.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.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-secrets.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.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-services.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.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
camel.component.kubernetes-services.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-services.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

Name	描述	默认值	类型
camel.component.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-builds.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

Name	描述	默认值	类型
camel.component.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-deploymentconfigs.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

第 79 章 OPENSHIFT 构建配置

从 Camel 2.17 开始

仅支持生成者

OpenShift Build Config 组件是 Kubernetes 组件 之一，它为执行 Openshift Build Configs 操作提供一个制作者。

79.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 openshift-build-configs 时，请使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

79.2. 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

79.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

79.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

79.3. 组件选项

Openshift Build Config 组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (producer)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

79.4. 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：

79.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterURL (producer)	**所需的** Kubernetes 主 url。		字符串

79.4.2. 查询参数(21 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (producer)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (producer)	命名空间。		字符串
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
PORTNAME (producer)	用于 ServiceCall EIP 的端口名称。		字符串
portProtocol (producer)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

79.5. 消息标头

Openshift Build Config 组件支持 4 个消息标头，如下所列：

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNamespaceName (producer) 常量 ：KUBERNETES_NAMESPACE_NAME	命名空间名称。		字符串
CamelKubernetesBuildConfigsLabels (producer) 恒定：KUBERNETES_BUILD_CONFIGS_LABELS	Openshift Config Build 标签。		Map
CamelKubernetesBuildConfigName (producer) 常量 ：KUBERNETES_BUILD_CONFIG_NAME	Openshift Config Build 名称。		字符串

79.6. 支持的制作者操作

- ***listBuildConfigs***
- ***listBuildConfigsByLabels***
- ***getBuildConfig***

79.7. OPENSHIFT BUILD CONFIGS PRODUCER 示例

- ListBuilds：此操作列出了 Openshift 集群上的构建配置。

```
from("direct:list").
    toF("openshift-build-configs:///?
kubernetesClient=#kubernetesClient&operation=listBuildConfigs").
    to("mock:result");
```

此操作返回 Openshift 集群中的构建列表。

- listBuildsByLabels：此操作根据 Openshift 集群上的标签列出构建配置。

```
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_BUILD_CONFIGS_LABELS,
labels);
            }
        });
    toF("openshift-build-configs:///?
kubernetesClient=#kubernetesClient&operation=listBuildConfigsByLabels").
    to("mock:result");
```

此操作使用标签选择器（值为 value1 和 key2）从集群中返回 Build 配置列表。

79.8. SPRING BOOT AUTO-CONFIGURATION

组件支持 102 选项，如下所列。

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map

Name	描述	默认值	类型
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å�Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long

Name	描述	默认值	类型
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-config-maps.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-config-maps.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.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-custom-resources.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

Name	描述	默认值	类型
camel.component.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-deployments.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.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
camel.component.kubernetes-deployments.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-deployments.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.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-events.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.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-hpa.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.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
camel.component.kubernetes-hpa.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-job.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-namespaces.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.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
camel.component.kubernetes-namespaces.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-namespaces.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.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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

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
camel.component.kubernetes-pods.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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
camel.component.kubernetes-replication-controllers.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-replication-controllers.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.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.kubernetes-secrets.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.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-services.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

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
camel.component.kubernetes-services.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 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.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-deploymentconfigs.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

第 80 章 OPENSHIFT 部署配置

自 Camel 3.18 起

支持生成者和消费者

Openshift Deployment Configs 组件是 Kubernetes 组件 之一，它提供生成者来执行 Openshift Deployment Configs 操作，以及一个消费者，以使用与 Deployment Configs 对象相关的事件。

80.1. 依赖项

当在 Red Hat build of Apache Camel for Spring Boot 中使用 openshift-deploymentconfigs 时，使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-kubernetes-starter</artifactId>
</dependency>
```

80.2. 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

80.2.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

80.2.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

80.3. 组件选项

Openshift Deployment Configs 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
kubernetesClient (common)	**Autowired** 使用现有的 kubernetes 客户端。		KubernetesClient
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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

80.4. 端点选项

Openshift Deployment Configs 端点使用 URI 语法进行配置：

```
openshift-deploymentconfigs:masterUrl
```

使用以下路径和查询参数：

80.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
masterUrl (common)	**所需的** Kubernetes 主 url。		字符串

80.4.2. 查询参数(33 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)	dns 域，用于 ServiceCall EIP。		字符串
kubernetesClient (common)	如果提供，要使用的默认 KubernetesClient。		KubernetesClient
namespace (common)	命名空间。		字符串
portName (common)	用于 ServiceCall EIP 的端口名称。		字符串

Name	描述	默认值	类型
portProtocol (common)	用于 ServiceCall EIP 的端口协议。	tcp	字符串
crdGroup (consumer)	您要监视的 Consumer CRD 资源组。		字符串
crdName (consumer)	我们想监视的 Consumer CRD 资源名称。		字符串
crdPlural (consumer)	Consumer CRD Resource Plural we like to watch。		字符串
crdScope (consumer)	Consumer CRD 资源范围我们想监视。		字符串
crdVersion (consumer)	我们想监视的 Consumer CRD 资源版本。		字符串
labelKey (consumer)	当监视某些资源时，Consumer Label 键。		字符串
labelValue (consumer)	Consumer Label 值在监视某些资源时。		字符串
poolSize (consumer)	Consumer 池大小。	1	int
resourceName (consumer)	要监视的消费者资源名称。		字符串
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut		ExchangePattern
operation (producer)	在 Kubernetes 上执行制作者操作。		字符串
lazyStartProducer (producer (advanced))	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
connectionTimeout (advanced)	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
caCertData (security)	CA 认证数据.		字符串
caCertFile (security)	CA 认证文件.		字符串
clientCertData (security)	客户端认证数据.		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 令牌.		字符串

Name	描述	默认值	类型
password (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 （安全性）	连接到 Kubernetes 的用户名。		字符串

80.5. 消息标头

Openshift Deployment Configs 组件支持 8 个消息标头，如下所列：

Name	描述	默认值	类型
CamelKubernetesOperation (producer) 常数 ：KUBERNETES_OPERATION	Producer 操作。		字符串
CamelKubernetesNamespaceName (producer) 常量 ：KUBERNETES_NAMESPACE_NAME	命名空间名称。		字符串
CamelKubernetesDeploymentsLabels (producer) 恒定：KUBERNETES_DEPLOYMENTS_LABELS	部署标签。		Map
CamelKubernetesDeploymentName (producer) 常数 ：KUBERNETES_DEPLOYMENT_NAME	部署名称。		字符串

Name	描述	默认值	类型
CamelKubernetesDeploymentReplicas (producer) 常数 ：KUBERNETES_DEPLOYMENT_REPLICAS	所需的实例数。		整数
CamelKubernetesDeploymentConfigSpec (producer) 常数 ：KUBERNETES_DEPLOYMENT_CONFIG_SPEC	部署配置的 spec。		DeploymentConfigSpec
CamelKubernetesEventAction (consumer) 常量 ：KUBERNETES_EVENT_ACTION	消费者监视的操作。 Enum 值： • ADDED • MODIFIED • 已删除 • ERROR • 书签		操作
CamelKubernetesEventTimestamp (consumer) 恒定：KUBERNETES_EVENT_TIMESTAMP	消费者监视的操作的时间戳。		long

80.6. 支持的制作者操作

- ***listDeploymentConfigs***
- ***listDeploymentsConfigsByLabels***

- *getDeploymentConfig*
- *createDeploymentConfig*
- *updateDeploymentConfig*
- *deleteDeploymentConfig*
- *scaleDeploymentConfig*

80.7. OPENSHIFT DEPLOYMENT CONFIGS PRODUCER 示例

- *ListDeploymentConfig：此操作列出了 Openshift 集群上的部署。*

```
from("direct:list").
    toF("openshift-deploymentconfigs:///?
kubernetesClient=#kubernetesClient&operation=listDeploymentConfigs").
    to("mock:result");
```

此操作会返回集群中的部署配置列表。

- *listDeploymentConfigsByLabels：此操作根据 Openshift 集群上的标签列出部署配置。*

```
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_DEPLOYMENTS_LABELS,
labels);
            }
        });
    toF("openshift-deploymentconfigs:///?
kubernetesClient=#kubernetesClient&operation=listDeploymentConfigsByLabels").
    to("mock:result");
```

此操作使用标签选择器（带有 key1 和 key2 的值2）返回来自集群的 Deployment Configs 列表。

80.8. OPENSHIFT DEPLOYMENT CONFIGS CONSUMER 示例

```
fromF("openshift-deploymentconfigs://%s?
oauthToken=%s&namespace=default&resourceName=test", host, authToken).process(new
OpenshiftProcessor()).to("mock:result");
    public class OpenshiftProcessor implements Processor {
        @Override
        public void process(Exchange exchange) throws Exception {
            Message in = exchange.getIn();
            DeploymentConfig dp = exchange.getIn().getBody(DeploymentConfig.class);
            log.info("Got event with configmap name: " + dp.getMetadata().getName() + " and
action " + in.getHeader(KubernetesConstants.KUBERNETES_EVENT_ACTION));
        }
    }
```

此消费者返回部署配置测试的命名空间 default 上的事件列表。

80.9. SPRING BOOT AUTO-CONFIGURATION

组件支持 102 选项，如下所列。

Name	描述	默认值	类型
camel.cluster.kubernetes.attributes	自定义服务属性。		Map
camel.cluster.kubernetes.cluster-labels	设置用于识别组成集群的 pod 的标签。		Map
camel.cluster.kubernetes.config-map-name	设置用于进行 optimistic locking （默认为 'leaders）的 ConfigMap 名称。		字符串
camel.cluster.kubernetes.connection-timeout-millis	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.cluster.kubernetes.enabled	设定是否应启用 Kubernetes 集群服务，默认为 false。	false	布尔值
camel.cluster.kubernetes.id	集群服务 ID。		字符串

Name	描述	默认值	类型
camel.cluster.kubernetes.jitter-factor	要应用的 jitter 因素，以防止所有 pod 在同一即时调用 Kubernetes API。		å�Œ
camel.cluster.kubernetes.kubernetes-namespace	设置包含 pod 和 configmap 的 Kubernetes 命名空间的名称（默认自动探测到）。		字符串
camel.cluster.kubernetes.lease-duration-millis	当前领导的租期的默认持续时间。		Long
camel.cluster.kubernetes.master-url	设置 Kubernetes 主机的 URL （默认为从 Kubernetes 客户端属性读取）。		字符串
camel.cluster.kubernetes.order	服务查找顺序/优先级.		整数
camel.cluster.kubernetes.pod-name	设置当前 pod 的名称（默认为从容器主机名检测）。		字符串
camel.cluster.kubernetes.renew-deadline-millis	领导机必须停止其服务的截止时间，因为它可能已失去领导机。		Long
camel.cluster.kubernetes.retry-period-millis	连续两次尝试检查并获取领导权利之间的时间。它使用 jitter 因子来随机化。		Long
camel.component.kubernetes-config-maps.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-config-maps.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.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

Name	描述	默认值	类型
camel.component.kubernetes-config-maps.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-config-maps.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.kubernetes-custom-resources.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-custom-resources.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.kubernetes-custom-resources.enabled	是否启用 kubernetes-custom-resources 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-custom-resources.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-custom-resources.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

Name	描述	默认值	类型
camel.component.kubernetes-deployments.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-deployments.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.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
camel.component.kubernetes-deployments.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-deployments.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.kubernetes-events.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-events.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

Name	描述	默认值	类型
camel.component.kubernetes-events.enabled	是否启用 kubernetes-events 组件的自动配置。这默认是启用的。		布尔值
camel.component.kubernetes-events.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-events.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.kubernetes-hpa.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-hpa.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.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
camel.component.kubernetes-hpa.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-hpa.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

Name	描述	默认值	类型
camel.component.kubernetes-job.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-job.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-job.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.kubernetes-namespaces.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-namespaces.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-namespaces.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.kubernetes-nodes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-nodes.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.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
camel.component.kubernetes-nodes.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-nodes.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.kubernetes-persistent-volumes-claims.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes-claims.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.kubernetes-persistent-volumes.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-persistent-volumes.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

Name	描述	默认值	类型
camel.component.kubernetes-pods.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-pods.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.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
camel.component.kubernetes-pods.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-pods.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.kubernetes-replication-controllers.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kubernetes-replication-controllers.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.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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-replication-controllers.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.kubernetes-resources-quota.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-resources-quota.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.kubernetes-secrets.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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

Name	描述	默认值	类型
camel.component.kubernetes-secrets.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-secrets.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.kubernetes-service-accounts.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-service-accounts.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.kubernetes-services.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.kubernetes-services.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.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
camel.component.kubernetes-services.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.kubernetes-services.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.openshift-build-configs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-build-configs.enabled	是否启用 openshift-build-configs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-build-configs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-build-configs.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

Name	描述	默认值	类型
camel.component.openshift-builds.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-builds.enabled	是否启用 openshift-builds 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-builds.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient
camel.component.openshift-builds.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.openshift-deploymentconfigs.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.openshift-deploymentconfigs.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.openshift-deploymentconfigs.enabled	是否启用 openshift-deploymentconfigs 组件的自动配置。这默认是启用的。		布尔值
camel.component.openshift-deploymentconfigs.kubernetes-client	使用现有的 kubernetes 客户端。选项是一个 io.fabric8.kubernetes.client.KubernetesClient 类型。		KubernetesClient

Name	描述	默认值	类型
camel.component.openshift-deploymentconfigs.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

第 81 章 KUDU

Since Camel 3.0

仅支持生成者

Kudu 组件支持从/到 Apache Kudu 存储和检索数据，Apache Hadoop 生态系统的免费开源列导向型数据存储。

81.1. 依赖项

当在 Camel Spring Boot 中使用 kudu 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-kudu-starter</artifactId>
</dependency>
```

81.2. 先决条件

您必须有一个有效的 Kudu 实例正在运行。如需更多信息，请参阅 Apache Kudu。

81.3. 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

81.3.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

81.3.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

81.4. 组件选项

Kudu 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）。 这允许 CamelContext 和路由在生成者失败的情况下启动。 在 lazy 启动时，可通过 Camel 的路由错误处理程序在路由消息期间处理启动失败。 **注意** 处理第一个消息后，可能需要稍等片刻才能创建和启动生成者来增加总处理时间。	false	布尔值

Name	描述	默认值	类型
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

81.5. 端点选项

Kudu 端点使用 URI 语法进行配置：

```
kudu:host:port/tableName
```

使用以下路径和查询参数：

81.5.1. 路径参数(3 参数)

Name	描述	默认值	类型
host (common)	要连接的服务器的主机。		字符串
port (common)	要连接的服务器的端口。		字符串
tableName (common)	要连接的表。		字符串

81.5.2. 查询参数(2 参数)

Name	描述	默认值	类型

Name	描述	默认值	类型
operation (producer)	要执行的操作。 Enum 值： • INSERT • DELETE • UPDATE （更新） • UPSERT • CREATE_TABLE • SCAN		KuduOperations
lazyStartProducer (producer (advanced))	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

81.6. 消息标头

Kudu 组件支持 5 个消息标头，如下所列：

Name	描述	默认值	类型
CamelKuduSchema (producer) 恒定：CAMEL_KUDU_SCHEMA	模式。		模式
CamelKuduTableOptions (producer) 常数：CAMEL_KUDU_TABLE_OPTIONS	create 表选项。		CreateTableOptions

Name	描述	默认值	类型
CamelKuduScanColumnNames (producer) 恒定：CAMEL_KUDU_SCAN_COLUMN_NAMES	扫描操作投射列名称。		list
CamelKuduScanPredicate (producer) 恒定：CAMEL_KUDU_SCAN_PREDICATE	扫描操作的 predicate。		KuduPredicate
CamelKuduScanLimit (producer) 常数：CAMEL_KUDU_SCAN_LIMIT	扫描操作的行数的限制。		long

81.7. 输入正文格式

81.7.1. insert、delete、update 和 upsert

输入正文格式必须是 java.util.Map<String, Object>。此映射将代表表的行，其元素为列，其中键是列名称，值为列的值。

81.8. 输出正文格式

81.8.1. 扫描

输出正文格式将是 java.util.List<java.util.Map<String, Object>>。列表的每个元素将是表的不同行。每行都是 Map<String, Object>，其元素是该行的列 name 和 column 值的每对。

81.9. 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.kudu.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.kudu.enabled	是否启用 kudu 组件的自动配置。这默认是启用的。		布尔值
camel.component.kudu.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

第 82 章 语言

仅支持生成者

Language 组件允许您将交换发送到端点，该端点由 Camel 中任何受支持的语言执行脚本。通过让组件执行语言脚本，它允许更多动态路由功能。例如，通过使用 Routing Slip 或 Dynamic Router EIP，您还可以发送消息到脚本被动态定义的语言端点。

此组件在 camel-core 中提供了，因此不需要额外的 JAR。只有选择的语言（如使用 Groovy 或 JavaScript 语言）时，才需要包含额外的 Camel 组件。

82.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 语言 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-language-starter</artifactId>
</dependency>
```

82.2. URI 格式

```
language://languageName[:script][?options]
```

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]
```

82.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

82.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

82.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

82.4. 组件选项

Language 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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

82.5. 端点选项

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：

82.5.1. 路径参数(2 参数)

Name	描述	默认值	类型

Name	描述	默认值	类型
languageName (producer)	**必需** 设置要使用的语言的名称。 Enum 值： • Bean • constant • exchangeProperty • file • groovy • header • JavaScript • jsonpath • mvel • OGNL • ref • simple • spel • sql • terser • tokenize • XPath • xquery • xtokenize		字符串
resourceUri (producer)	资源的路径，或在 Registry 中查找 bean 的引用，以用作资源。		字符串

82.5.2. 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细信息。默认情况下，只能访问消息正文和标头。这个选项可以启用对当前 Exchange 和 CamelContext 的完整访问权限。这样做会造成潜在的安全风险，因为这将打开对 CamelContext API 的完整功能的访问。	false	布尔值

Name	描述	默认值	类型
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对下一个 org.apache.camel.Exchange 处理一个 Camel org.apache.camel.Exchange 造成副作用。	false	布尔值
contentCache (producer)	设置是否使用资源内容缓存。	true	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
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。这个选项默认为 true。	true	布尔值

82.6. 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	要在标头中提供的执行的脚本。优先于端点上配置的脚本。

82.7. 例子

例如，您可以使用 Simple 语言将消息用于 Message Translator。

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);
```

82.8. 从资源载入脚本

您可以为在端点 uri 或 Exchange.LANGUAGE_SCRIPT 标头中加载的脚本指定一个资源 uri。uri 必须以以下方案之一开始：file:、classpath: 或 http ：

默认情况下，该脚本会加载一次并缓存。但是，您可以禁用 contentCache 选项，并在每次评估上载入脚本。例如，如果在磁盘上更改了 myscript.txt 文件，则使用更新的脚本：

您可以通过使用 "resource:" 前缀来引用与 Camel 中 其他语言 类似的资源，如下所示。

82.9. 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.language.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.language.enabled	是否启用语言组件的自动配置。这默认是启用的。		布尔值
camel.component.language.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

第 83 章 LDAP

自 Camel 1.5 起

仅支持生成者

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。此组件使用标准的 JNDI (javax.naming package)来访问服务器。

83.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 ldap 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-ldap-starter</artifactId>
</dependency>
```

83.2. URI 格式

```
ldap:ldapServerBean[?options]
```

URI 中的 ldapServerBean 指的是 registry 中的 DirContext bean。LDAP 组件只支持生成者端点，这意味着在路由开始时无法从中出现 ldap URI。

83.3. 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

83.3.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

83.3.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

83.4. 组件选项

LDAP 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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

83.5. 端点选项

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：

83.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
dirContextName (producer)	在 registry 中查找的 javax.naming.directory.DirContext 或 java.util.Hashtable 或 Map bean 的必要名称。**如果 bean 是 Hashtable 或 Map，则会为每个用途创建一个新的 javax.naming.directory.DirContext 实例。如果 bean 是 javax.naming.directory.DirContext，则 bean 将用作 given。在所有不能共享 javax.naming.directory.DirContext 的情形中，后者可能无法使用 java.util.Hashtable 或 Map。**		字符串

83.5.2. 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)	指定 ldap 模块时，使用分页来检索所有结果（大多数 LDAP 服务器在尝试检索一个查询中超过 1000 个条目时抛出异常）。要能够使用此 LdapContext （子 DirContext 的子类）必须传递为 ldapServerBean （否则抛出异常）。		整数

Name	描述	默认值	类型
returnedAttributes (producer)	以逗号分隔的属性列表，这些属性应在结果的每个条目中设置。		字符串
scope (producer)	指定如何深度搜索条目树，从基本 DN 开始。 Enum 值： • object • onelevel • subtree	subtree	字符串
lazyStartProducer (producer (advanced))	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

83.6. 结果

*结果作为 **List<javax.naming.directory.SearchResult >** 对象返回到 **Out body**。*

83.7. DIRCONTEXT

URI ldap:ldapserver 引用了 ID 为 ldapserver 的 Spring bean。ldapserver bean 可以定义如下：

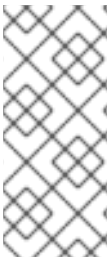
```
<bean id="ldapserver" class="javax.naming.directory.InitialDirContext" scope="prototype">
  <constructor-arg>
    <props>
      <prop key="java.naming.factory.initial">com.sun.jndi.ldap.LdapCtxFactory</prop>
      <prop key="java.naming.provider.url">ldap://localhost:10389</prop>
      <prop key="java.naming.security.authentication">none</prop>
    </props>
  </constructor-arg>
</bean>
```

前面的示例声明了一个基于 Sun 的 LDAP DirContext，它匿名连接到本地托管的 LDAP 服务器。

注意

DirContext 对象 不需要 根据合同支持并发。因此，务必要在 bean 定义或上下文支持 concurrency 中通过设置 scope="prototype" 声明目录上下文。在 Spring 框架中，每次查找时，对范围的对象进行实例化。

83.8. 与 LDAP 注入相关的安全顾虑



注意

camel-ldap 组件使用消息正文来过滤搜索结果。因此，消息正文应该受到 LDAP 注入的保护。为了协助，您可以使用 org.apache.camel.component.ldap.LdapHelper 工具类来转义带有方法的字符串值，以便安全 LDAP 注入。

如需更多信息，请参阅 LDAP 注入。

83.9. SAMPLES

在以上 Spring 配置中，以下代码示例发送 LDAP 请求来过滤成员的搜索组。然后，从响应中提取通用名称。

```
ProducerTemplate template = exchange.getContext().createProducerTemplate();

Collection<SearchResult> results = template.requestBody(
    "ldap:ldapserver?base=ou=mygroup,ou=groups,ou=system",
    "(member=uid=huntc,ou=users,ou=system)", Collection.class);

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 条目有一个通用名称，请使用如下过滤器表达式：

```
(cn=*)
```

83.9.1. 使用凭证绑定

Camel 最终用户贡献了这个示例代码，他用来使用凭证绑定到 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");

DefaultRegistry reg = new DefaultRegistry();
reg.bind("myldap", new InitialLdapContext(props, null));

CamelContext context = new DefaultCamelContext(reg);
context.addRoutes(
   new RouteBuilder() {
      @Override
      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.getMessage().getBody(Collection.class);
assert data != null;
assert !data.isEmpty();

System.out.println(out.getMessage().getBody());

context.stop();
```

83.10. 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，请参阅以下示例。

SSL 配置

```
<?xml version="1.0" encoding="UTF-8"?>
<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"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
    http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd">

    <sslContextParameters xmlns="http://camel.apache.org/schema/spring"
id="sslContextParameters" >
        <keyManagers keyPassword="{{keystore.pwd}}">
            <keyStore resource="{{keystore.url}}" password="{{keystore.pwd}}"/>
        </keyManagers>
    </sslContextParameters>

    <bean id="customSocketFactory" class="com.example.ldap.CustomSocketFactory">
        <constructor-arg index="0" ref="sslContextParameters"/>
    </bean>

    <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">ldaps://127.0.0.1:10636</prop>
                <prop key="java.naming.security.protocol">ssl</prop>
                <prop key="java.naming.security.authentication">none</prop>
                <prop
key="java.naming.ldap.factory.socket">com.example.ldap.CustomSocketFactory</prop>
            </props>
        </constructor-arg>
    </bean>
</beans>
```

自定义套接字工厂

```
package com.example.ldap;

import java.io.IOException;
import java.net.InetAddress;
import java.net.Socket;
import java.security.KeyStore;

import javax.net.SocketFactory;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.TrustManagerFactory;
```

```
import org.apache.camel.support.jsse.SSLContextParameters;

/**
 * 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 Spring Boot DI to initialize an instance of SocketFactory
     */
    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);
        }
    }

    /**
     * Getter for the SocketFactory
     */
    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);
    }
}
```

83.11. 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.ldap.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.ldap.enabled	是否启用 ldap 组件的自动配置。这默认是启用的。		布尔值
camel.component.ldap.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

第 84 章 LOG

仅支持生成者

Log 组件日志消息交换到底层日志记录机制。

Camel 使用 SLF4J，它允许您通过其他配置日志：

- Log4j
- Logback
- Java Util Logging

84.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用日志时，请确保使用以下 Maven 依赖项来支持自动配置：

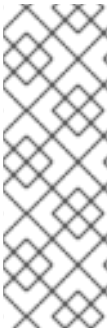
```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-log-starter</artifactId>
</dependency>
```

84.2. URI 格式

```
log:loggingCategory[?options]
```

其中 loggingCategory 是要使用的日志记录类别的名称。您可以以以下格式将查询选项附加到 URI 中，

```
?option=value&option=value&...
```



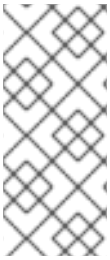
注意

使用来自 Registry 的 Logger 实例
如果在 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，如果没有注册和 URI 日志记录器 参数，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 Throughput 日志记录器，每当指定 groupSize 选项时使用。



注意

另外在 DSL 中有一个日志
在 DSL 中也直接有一个 log，但它具有不同的目的。它适用于轻量级和人为日志。请参阅 LogEIP 的更多详细信息。

84.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

84.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

84.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

84.4. 组件选项

日志组件支持 3 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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
exchangeFormatter (advanced)	**Autowired** 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，我们默认为 DefaultExchangeFormatter。		ExchangeFormatter

84.5. 端点选项

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：

84.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
loggerName (producer)	要使用的日志类别的 **必需** 名称。		字符串

84.5.2. 查询参数(27 参数)

Name	描述	默认值	类型
groupActiveOnly (producer)	如果为 true，则无论消息流量如何，如果没有收到新消息的时间间隔，则会隐藏统计信息，如果为 false，显示统计。	true	布尔值
groupDelay (producer)	设置 stats 的初始延迟（在 millis 中）。		Long
groupInterval (producer)	如果指定将按这个时间间隔分组消息统计数据（在 millis 中）。		Long
groupSize (producer)	为吞吐量日志记录指定组大小的整数。		整数
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
level (producer)	要使用的日志记录级别。默认值为 INFO。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	INFO	字符串
logMask (producer)	如果为 true，请在日志中屏蔽密码或密码短语等敏感信息。		布尔值
marker (producer)	要使用的可选标记名称。		字符串
exchangeFormatter (advanced)	使用自定义交换格式器。		ExchangeFormatter
maxChars (formatting)	限制每行记录的字符数。	10000	int
多行 （格式）	如果启用，则每个信息都会在新行中输出。	false	布尔值
showAll （格式）	用于打开所有选项的快速选项。（如果要使用，则必须手动设置 maxChars）。	false	布尔值
showAllProperties (formatting)	显示所有交换属性（包括内部和自定义）。	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)	如果交换有清晰的异常，则显示异常消息（没有堆栈追踪）。Caught 异常作为属性存储在交换上（使用键 org.apache.camel.Exchange#EXCEPTION_CAUGHT）以及 doCatch 可以捕获异常。	false	布尔值
showException (formatting)	如果交换有例外，显示异常消息（没有堆栈追踪）。	false	布尔值

Name	描述	默认值	类型
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 （简称为 MEP）。	true	布尔值
showFiles (formatting)	如果启用了 Camel 将输出文件。	false	布尔值
showFuture (formatting)	如果启用 Camel 将于 future 对象等待其完成，以获取要记录有效负载。	false	布尔值
showHeaders (formatting)	显示消息标头。	false	布尔值
showProperties (formatting)	显示交换属性（仅自定义）。使用 showAllProperties 显示内部属性和自定义属性。	false	布尔值
showStackTrace (formatting)	显示堆栈追踪（如果交换有例外）。仅在启用了 showAll、showException 或 showCaughtException 之一时才有效。	false	布尔值
showStreams (formatting)	Camel 是否应该显示流正文（例如 java.io.InputStream）。如果您启用这个选项，则您可能以后无法访问消息正文，因为此日志记录器已读取流。要更正此问题，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 将保留正文中的任何行分隔符，然后将正文记录为。	true	布尔值
风格 （格式）	设置要使用的输出样式。 Enum 值： • 默认值 • 标签页 • 已修复	默认值	OutputStyle

84.6. 常规日志记录器示例

在以下路由中，我们在处理顺序前在 DEBUG 级别记录传入的顺序：

```
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>
```

84.7. 带有格式器示例的常规日志记录器

在以下路由中，我们会在处理顺序前在 INFO 级别记录传入的顺序。

```
from("activemq:orders").
    to("log:com.mycompany.order?showAll=true&multiline=true").to("bean:processOrder");
```

84.8. 带有 GROUPSIZE 示例的吞吐量日志记录器

在以下路由中，我们记录在 DEBUG 级别按 10 个消息分组的传入顺序的吞吐量。

```
from("activemq:orders").
    to("log:com.mycompany.order?level=DEBUG&groupSize=10").to("bean:processOrder");
```

84.9. 带有 GROUPINTERVAL 示例的吞吐量日志记录器

此路由将导致消息统计记录每 10 个，即使没有任何消息流量，也应该会显示初始 60s delay 和 stats。

```
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"
```

84.10. 屏蔽敏感信息，如密码

您可以通过将 logMask 标志设置为 true 来启用用于日志的安全屏蔽。请注意，这个选项也会影响 Log 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|off。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.support.processor.DefaultMaskingFormatter 默认用于屏蔽。如果要使用自定义屏蔽格式器，请将其放在 registry 中，其名称为 CamelCustomLogMask。请注意，掩码格式器必须实施 org.apache.camel.spi.MaskingFormatter。

84.11. 完全自定义日志输出

使用 部分中概述的选项，您可以控制日志记录器的大部分输出。但是，日志行总是遵循以下结构：

```
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，这种格式不适合，因为您需要...

- 过滤打印的标头和属性，以平衡洞察和详细程度。
- 将日志消息调整为您更易读的任何消息。
- 通过日志减减系统（如 Splunk）定制摘要日志消息。
- 打印特定正文类型会有所不同。

每当您需要绝对自定义时，您可以创建一个实现接口的类。通过 格式(Exchange) 方法，您可以访问完整的 Exchange，以便您可以选择并提取您需要的确切信息，以自定义方式对其进行格式化并返回。返回值将成为最终日志消息。

您可以通过以下两种方式之一获取自定义 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>
```

84.11.1. 与配置相关的约定

只需通过注册带有名称 logFormatter 的 bean ；Log 组件足以自动选择它。

```
<bean name="logFormatter" class="com.xyz.MyCustomExchangeFormatter" />
```

注意

ExchangeFormatter 应用到 Camel 上下文内的所有日志端点。如果不同端点需要不同的 ExchangeFormatters，请根据需要多次实例化 LogComponent，并使用相关的 bean 名称作为端点前缀。

使用自定义日志格式器时，您可以在 log uri 中指定参数，该参数在自定义日志格式器上配置。虽然当您这样做时，您应该将"logFormatter"定义为表格范围，因此如果您具有不同的参数，则不会共享它，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 的 Camel 路由：

```
<to uri="log:foo?param1=foo&amp;param2=100"/>

<to uri="log:bar?param1=bar&amp;param2=200"/>
```

84.12. 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.log.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.log.enabled	是否启用日志组件的自动配置。这默认是启用的。		布尔值
camel.component.log.exchange-formatter	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，我们默认为 DefaultExchangeFormatter。选项是 org.apache.camel.spi.ExchangeFormatter 类型。		ExchangeFormatter

Name	描述	默认值	类型
camel.component.log.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

第 85 章 LRA

从 Camel 2.21 开始

LRA 模块将 Saga EIP 与任何 MicroProfile 兼容 LRA Coordinator 绑定。

85.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 lra 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-lra-starter</artifactId>
</dependency>
```

85.2. SPRING BOOT AUTO-CONFIGURATION

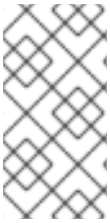
组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.lra.coordinator-context-path	LRA 协调器服务的上下文路径。		字符串
camel.lra.coordinator-url	LRA 协调器服务的基本 URL （例如 ）。		字符串
camel.lra.enabled	启用/禁用组件自动配置的全局选项，默认为 true。	true	布尔值
camel.lra.local-participant-context-path	本地参与回调服务的上下文路径。		字符串
camel.lra.local-participant-url	协调器应将回调发送到的本地 URL （例如 ）。		字符串

第 86 章 MAIL

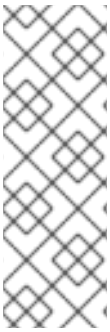
支持生成者和消费者

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。



注意

POP3 或 IMAP
POP3 有一些限制，并鼓励最终用户尽可能使用 IMAP。



注意

使用模拟电子邮件进行测试
，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的邮件服务器的情况下进行测试。但是，当您进入生产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模拟邮件。仅存在 classpath 上的 mock-javamail.jar 意味着它将在 中启动并避免发送邮件。

86.1. 依赖项

当在 Camel Spring Boot 中使用 camel-mail 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-mail-starter</artifactId>
</dependency>
```

86.2. URI 格式

邮件端点可以有下列 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]
```

86.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

86.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

86.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

86.4. 组件选项

Mail 组件支持 43 选项，如下所列。

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。将这个选项设置为 false，并且也具有 disconnect=false，然后消费者在轮询之间保持文件夹打开。	true	布尔值
copyTo (consumer)	在处理邮件后，可以使用指定名称将其复制到邮件文件夹中。您可以使用带有键 copyTo 的标头覆盖此配置值，允许您将消息复制到运行时配置的文件夹名称。		字符串
decodeFilename (consumer)	如果设置为 true，则使用 MimeUtility.decodeText 方法来解码文件名。这类似于设置 JVM 系统属性 mail.mime.encodefilename。	false	布尔值
delete (consumer)	处理消息后删除消息。这可以通过在邮件邮件中设置 DELETED 标志来完成。如果为 false，则设置 SEEN 标志。从 Camel 2.10 开始，您可以通过设置带有 key delete 的标头来覆盖此配置选项，以确定是否应删除邮件。	false	布尔值
disconnect (consumer)	消费者在轮询后是否应断开。如果启用，它会强制 Camel 在每个轮询上进行连接。	false	布尔值
handleFailedMessage (consumer)	如果邮件消费者无法检索给定的邮件邮件，则此选项允许处理消费者的错误处理程序造成的异常。通过在消费者上启用网桥错误处理程序，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，并且批处理中没有邮件可由 Camel 路由。	false	布尔值
mimeDecodeHeaders (consumer)	这个选项为邮件标头启用透明 MIME 解码和取消折叠。	false	布尔值
moveTo (consumer)	在处理邮件后，可以将其移动到具有指定名称的邮件文件夹中。您可以使用带有键 moveTo 的标头覆盖此配置值，允许您将消息移到运行时配置的文件夹名称。		字符串

Name	描述	默认值	类型
peek (consumer)	在处理邮件之前，将 javax.mail.Message 标记为 peeked。这只适用于 IMAPMessage 消息类型。通过使用 peek，邮件不会在邮件服务器上标记为 SEEN，如果 Camel 中存在错误处理，则我们就可以回滚邮件。	true	布尔值
skipFailedMessage (consumer)	如果邮件使用者无法检索给定的邮件邮件，则此选项允许跳过邮件并继续进行 来检索下一个邮件。默认行为是消费者抛出异常，并且批处理中没有邮件可由 Camel 路由。	false	布尔值
unseen (consumer)	是否仅受不可预见的邮件的限制。	true	布尔值
fetchSize (consumer (advanced))	设置在轮询期间要消耗的最大消息数。如果邮箱文件夹包含大量邮件，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，所有信息都会被使用。将值设为 0 是一个特殊角写，其中 Camel 将不会消耗任何消息。	-1	int
folderName (consumer (advanced))	要轮询的文件夹。	INBOX	字符串
mapMailMessage (consumer (advanced))	指定 Camel 是否应该将接收的邮件映射到 Camel body/headers/attachments。如果设置为 true，邮件消息的正文映射到 Camel IN 消息的正文，邮件标头映射到 IN 标头，并附加到 Camel IN attachment 消息。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)来检索此原始消息。	true	布尔值
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

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
replyTo (producer)	Reply-To 接收者（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)	发送的消息的主题。注：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 (advanced))	使用自定义 org.apache.camel.component.mail.JavaMailSender 来发送电子邮件。		JavaMailSender
additionalJavaMailProperties (advanced)	设置其他 java 邮件属性，这将根据所有其他选项附加/覆盖设置的任何默认属性。如果您需要添加一些特殊选项，但希望将其他选项保留原样，这将非常有用。		Properties
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 text/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
身份验证器 (advanced)	用于登录的验证器。如果设置，则忽略密码和用户名。可用于可过期的令牌，因此必须动态读取。		MailAuthenticator
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
configuration (advanced)	设置邮件配置。		MailConfiguration
connectionTimeout (advanced)	连接超时（以毫秒为单位）。	30000	int

Name	描述	默认值	类型
ContentType (advanced)	邮件消息内容类型。对 HTML 邮件使用 text/html。	text/plain	字符串
contentTypeResolver (advanced)	确定文件附加的 Content-Type 解析器。		ContentTypeResolver
debugMode (advanced)	在底层邮件框架上启用调试模式。默认情况下，SUN 邮件框架将调试消息记录到 System.out 中。	false	布尔值
ignoreUnsupportedCharset (advanced)	选项，可在发送邮件时让 Camel 忽略本地 JVM 中不支持的 charset。如果不支持 charset，则 charset=XXX （其中 XXX 代表不受支持的 charset）会从 content 类型中删除，它依赖于平台默认。	false	布尔值
ignoreUriScheme (advanced)	选项，可在发送邮件时让 Camel 忽略本地 JVM 中不支持的 charset。如果不支持 charset，则 charset=XXX （其中 XXX 代表不受支持的 charset）会从 content 类型中删除，它依赖于平台默认。	false	布尔值
javaMailProperties (advanced)	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
会话 (advanced)	指定 camel 应该用于所有邮件交互的邮件会话。在邮件会话由某些其他资源创建和管理的情况（如 hundreds 容器）时很有用。使用自定义邮件会话时，将使用来自邮件会话的主机名和端口（如果在会话中进行了配置）。		会话
useInlineAttachments (advanced)	是否使用分布内联或附加。	false	布尔值
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头到 Camel 消息。		HeaderFilterStrategy
password (security)	用于登录的密码。另请参阅 setAuthenticator (MailAuthenticator)。		字符串
sslContextParameters (security)	使用 SSLContextParameters 配置安全性。		SSLContextParameters
useGlobalSslContextParameters (security)	启用使用全局 SSL 上下文参数。	false	布尔值
用户名 （安全性）	用于登录的用户名。另请参阅 setAuthenticator (MailAuthenticator)。		字符串

86.5. 端点选项

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：

86.5.1. 路径参数(2 参数)

Name	描述	默认值	类型
host (common)	**必需** 邮件服务器主机名。		字符串
port (common)	邮件服务器的端口号。		int

86.5.2. 查询参数(66 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。将这个选项设置为 false，并且也具有 disconnect=false，然后消费者在轮询之间保持文件夹打开。	true	布尔值
copyTo (consumer)	在处理邮件后，可以使用指定名称将其复制到邮件文件夹中。您可以使用带有键 copyTo 的标头覆盖此配置值，允许您将消息复制到运行时配置的文件夹名称。		字符串
decodeFilename (consumer)	如果设置为 true，则使用 MimeUtility.decodeText 方法来解码文件名。这类似于设置 JVM 系统属性 mail.mime.encodefilename。	false	布尔值
delete (consumer)	处理消息后删除消息。这可以通过在邮件邮件中设置 DELETED 标志来完成。如果为 false，则设置 SEEN 标志。从 Camel 2.10 开始，您可以通过设置带有 key delete 的标头来覆盖此配置选项，以确定是否应删除邮件。	false	布尔值

Name	描述	默认值	类型
disconnect (consumer)	消费者在轮询后是否应断开。如果启用，它会强制 Camel 在每个轮询上进行连接。	false	布尔值
handleFailedMessage (consumer)	如果邮件消费者无法检索给定的邮件邮件，则此选项允许处理消费者的错误处理程序造成的异常。通过在消费者上启用网桥错误处理程序，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，并且批处理中没有邮件可由 Camel 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，不会设置最大值。可用于设置限制，例如 1000 个，以避免在服务器启动时下载数千个文件。设置 0 或 negative 值来禁用这个选项。		int
mimeDecodeHeaders (consumer)	这个选项为邮件标头启用透明 MIME 解码和取消折叠。	false	布尔值
moveTo (consumer)	在处理邮件后，可以将其移动到具有指定名称的邮件文件夹中。您可以使用带有键 moveTo 的标头覆盖此配置值，允许您将消息移到运行时配置的文件夹名称。		字符串
peek (consumer)	在处理邮件之前，将 javax.mail.Message 标记为 peeked。这只适用于 IMAPMessage 消息类型。通过使用 peek，邮件不会在邮件服务器上标记为 SEEN，如果 Camel 中存在错误处理，则我们就可以回滚邮件。	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的邮件邮件，则此选项允许跳过邮件并继续进行 来检索下一个邮件。默认行为是消费者抛出异常，并且批处理中没有邮件可由 Camel 路由。	false	布尔值
unseen (consumer)	是否仅受不可预见的邮件的限制。	true	布尔值
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
fetchSize (consumer (advanced))	设置在轮询期间要消耗的最大消息数。如果邮箱文件夹包含大量邮件，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，所有信息都会被使用。将值设为 0 是一个特殊角写，其中 Camel 将不会消耗任何消息。	-1	int
folderName (consumer (advanced))	要轮询的文件夹。	INBOX	字符串
mailUidGenerator (consumer (advanced))	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 (advanced))	指定 Camel 是否应该将接收的邮件映射到 Camel body/headers/attachments。如果设置为 true，邮件消息的正文映射到 Camel IN 消息的正文，邮件标头映射到 IN 标头，并附加到 Camel IN attachment 消息。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)来检索此原始消息。	true	布尔值
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
postProcessAction (consumer (advanced))	指向 MailBoxPostProcessAction，用于在正常处理正常处理后对邮箱进行发布后处理任务。		MailBoxPostProcessAction
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

Name	描述	默认值	类型
from (producer)	来自电子邮件地址。	camel@localhost	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
replyTo (producer)	Reply-To 接收者（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)	发送的消息的主题。注：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 (advanced))	使用自定义 org.apache.camel.component.mail.JavaMailSender 来发送电子邮件。		JavaMailSender
additionalJavaMailProperties (advanced)	设置其他 java 邮件属性，这将根据所有其他选项附加/覆盖设置的任何默认属性。如果您需要添加一些特殊选项，但希望将其他选项保留原样，这将非常有用。		Properties
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 text/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
身份验证器 (advanced)	用于登录的验证器。如果设置，则忽略密码和用户名。可用于可过期的令牌，因此必须动态读取。		MailAuthenticator
绑定 (advanced)	设置用于从 Camel 消息转换为和从邮件邮件转换的绑定。		MailBinding
connectionTimeout (advanced)	连接超时（以毫秒为单位）。	30000	int

Name	描述	默认值	类型
ContentType (advanced)	邮件消息内容类型。对 HTML 邮件使用 text/html。	text/plain	字符串
contentTypeResolver (advanced)	确定文件附加的 Content-Type 解析器。		ContentTypeResolver
debugMode (advanced)	在底层邮件框架上启用调试模式。默认情况下，SUN 邮件框架将调试消息记录到 System.out 中。	false	布尔值
headerFilterStrategy (advanced)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头。		HeaderFilterStrategy
ignoreUnsupportedCharset (advanced)	选项，可在发送邮件时让 Camel 忽略本地 JVM 中不支持的 charset。如果不支持 charset，则 charset=XXX （其中 XXX 代表不受支持的 charset）会从 content 类型中删除，它依赖于平台默认。	false	布尔值
ignoreUriScheme (advanced)	选项，可在发送邮件时让 Camel 忽略本地 JVM 中不支持的 charset。如果不支持 charset，则 charset=XXX （其中 XXX 代表不受支持的 charset）会从 content 类型中删除，它依赖于平台默认。	false	布尔值
javaMailProperties (advanced)	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
会话 (advanced)	指定 camel 应该用于所有邮件交互的邮件会话。在邮件会话由某些其他资源创建和管理的情况（如 hundreds 容器）时很有用。使用自定义邮件会话时，将使用来自邮件会话的主机名和端口（如果在会话中进行了配置）。		会话
useInlineAttachments (advanced)	是否使用分布内联或附加。	false	布尔值
idempotentRepository (filter)	可插拔存储库 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，不使用任何存储库。		IdempotentRepository
idempotentRepositoryRemoveOnCommit (filter)	使用幂等存储库时，当邮件邮件被成功处理并提交后，消息 id 应该从幂等存储库（默认）中删除或保存在存储库中。默认情况下，它假定消息 id 是唯一的，并且没有要保存在存储库中的值，因为邮件消息将标记为 see/moved 或删除，以防止它再次被消耗。因此，将消息 id 存储在幂等存储库中，因此没有值。但是，此选项允许存储消息 id，因为您可能具有的任何原因。	true	布尔值

Name	描述	默认值	类型
searchTerm (filter)	指的是 javax.mail.search.SearchTerm，它允许根据搜索条件（如主题、正文、来自、在特定日期后发送）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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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
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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService

Name	描述	默认值	类型
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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
password (security)	用于登录的密码。另请参阅 setAuthenticator (MailAuthenticator)。		字符串
sslContextParameters (security)	使用 SSLContextParameters 配置安全性。		SSLContextParameters
用户名 （安全性）	用于登录的用户名。另请参阅 setAuthenticator (MailAuthenticator)。		字符串
sortTerm (sort)	消息排序顺序。只有 IMAP 的原生支持。使用 POP3 或 IMAP 服务器没有 SORT 功能时，模拟到某种程度。		SortTerm[]

86.5.3. 端点示例

通常，您可以指定带有登录凭证的 URI，如下所示（例如，使用 SMTP 作为示例）：

```
smtp://[username@]host[:port][?password=somepwd]
```

另外，也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
```

86.5.4. 组件别名名称

- IMAP
- IMAPs
- POP3s
- SMTP
- SMTPs

86.5.5. 默认端口

支持默认端口号。如果省略了端口号，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

协议	默认端口号
POP3S	995
IMAP	143
IMAPS	993

86.6. 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必要的 Java 邮件 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。

86.6.1. 使用 JSSE 配置实用程序

邮件组件通过 Camel JSSE 配置实用程序 支持 SSL/TLS 配置实用程序。这个工具大大减少了您需要写入的组件特定代码数量，并在端点和组件级别进行配置。以下示例演示了如何将 实用程序与邮件组件一起使用。

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("smtps://smtp.google.com?
username=user@gmail.com&password=password&sslContextParameters=#sslContextParam
eters");
```

基于 Spring DSL 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"/
>...
```

86.6.2. 直接配置 JavaMail

Camel 使用 Jakarta JavaMail，它只信任由已知已知证书颁发机构（默认的 JVM 信任配置）发布的证书。如果您发布自己的证书，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（详情请参阅 JavaMail 中的 SSLNOTES.txt ）。

86.7. 邮件内容

Camel 使用消息交换的 IN body 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 subject 可使用 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 header 属性 To：

使用 MailProducer 将邮件发送到服务器时，您应能够从 Camel 消息标头中获取 MimeMessage 的消息 id，以及 Camel 消息标头中的密钥 CamelMailMessageId。

86.8. 标头优先于预先配置的接收者

在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，即您所获得的内容。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，电子邮件信息被发送到 davsclaus@apache.org，因为它优先于预先配置的接收者 info@mycompany.com。端点 URI 中的任何 CC 和 BCC 设置也会被忽略，这些接收者将不会接收任何邮件。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。无法混合和匹配标头和预先配置的设置。

```
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);
```

86.9. 多个接收者以简化配置

可以使用逗号分隔或分号分隔的列表来设置多个接收者。这适用于标头设置，以及端点 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。

86.10. 设置发件人名称和电子邮件

您可以以名为 < email> 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，您可以在消息中定义以下标头：

```
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");
```

86.11. JAVAMAIL API (EX 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 的功能集比 IMAP 更有限。

- javamail POP3 API
- javamail IMAP API
- 通常有关 MAIL 标记

86.12. SAMPLES

我们从一个简单的路由开始，该路由将从 JMS 队列接收的消息作为电子邮件发送。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。

```
from("imap://admin@mymailserver.com?password=secret&unseen=true&delay=60000")
    .to("seda://mails");
```

86.13. 使用附加示例发送邮件

注意

所有 Camel 组件都不支持附件。附件 API 基于 Java 激活框架，通常仅由 Mail API 使用。因为很多其他 Camel 组件不支持附件，连接可能会在路由中传播时丢失。因此，thumb 的规则是在向邮件端点发送消息前添加附件。

邮件组件支持附件。在以下示例中，我们发送一个包含纯文本消息并包含徽标文件附加的邮件。

86.14. SSL 示例

在本例中，我们希望为邮件轮询 Google 邮件。要将邮件下载到本地邮件客户端，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。Google 具有大量的有关如何执行此操作的文档。

```
from("imaps://imap.gmail.com?
username=YOUR_USERNAME@gmail.com&password=YOUR_PASSWORD"
    + "&delete=false&unseen=true&delay=60000").to("log:newmail");
```

前面的路由每分钟轮询一次新邮件的 Google 邮件，并将收到的消息记录到 新邮件 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=[...
```

86.15. 使用附加示例消耗邮件

在这个示例中，我们轮询一个邮箱，并将邮件中的所有附件存储为文件。首先，我们定义一个路由来轮询邮箱。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&delay=60000").process(new MyMailProcessor());
```

我们使用可从 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
    AttachmentMessage attachmentMessage =
exchange.getMessage(AttachmentMessage.class);
    Map<String, DataHandler> attachments = attachmentMessage.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 有点冲突，但可以获得 javax.activation.DataHandler，以便您可以使用标准 API 处理附件。

86.16. 如何使用附加分割邮件

在这个示例中，我们消耗可能有多个附件的邮件。我们想要做的是每个附件使用 Splitter EIP 来单独处理附件。例如，如果邮件消息有 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。为此，我们需要为 Splitter 提供自定义表达式，其中我们提供了一个 List<Message>，其中包含带有单个附件的五个消息。

该代码在 Camel 2.10 中提供了 camel-mail 组件中的框。代码位于类：org.apache.camel.component.mail.SplitAttachmentsExpression，您可以 在此处找到 源代码。

在 Camel 路由中，您需要在路由中使用此表达式，如下所示：

如果使用 XML DSL，则需要在 Splitter 中声明方法调用表达式，如下所示

```
<split>
  <method beanType="org.apache.camel.component.mail.SplitAttachmentsExpression"/>
  <to uri="mock:split"/>
</split>
```

您还可以将附件分成 byte[]，以存储为消息正文。这可以通过使用布尔值 true 创建表达式来完成

```
SplitAttachmentsExpression split = SplitAttachmentsExpression(true);
```

然后，使用带有分割 EIP 的表达式。

86.17. 使用自定义 SEARCHTERM

您可以在 MailEndpoint 上配置 searchTerm，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，表示我们希望搜索邮件主题或正文，使其包含单词 "Camel"。
类 org.apache.camel.component.mail.SimpleSearchTerm 有多个您可以配置的选项：

或者让新的不可预见电子邮件恢复 24 小时，您可以做。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，要使最后一个不可预见的电子邮件返回 24 小时，在邮件主题中有 Camel，您可以：

```
<route>
  <from uri="imaps://mymailseerver?
username=foo&password=secret&searchTerm.subject=Camel&searchTerm.fromSentDate=no
w-24h"/>
  <to uri="bean:myBean"/>
</route>
```

SimpleSearchTerm 旨在从 POJO 中轻松配置，因此您也可以使用 XML 中的 <bean> 风格进行配置

```
<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，您可以在 Camel 路由中使用 #beanId 引用此 bean，如下所示：

```
<route>
```

```
  <from uri="imaps://mymailseerver?
username=foo&password=secret&searchTerm=#mySearchTerm"/>
  <to uri="bean:myBean"/>
</route>
```

在 Java 中，有一个构建器类可用于使用 org.apache.camel.component.mail.SearchTermBuilder 类构建复合搜索 Terms。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();
```

86.18. 轮询优化

参数 maxMessagePerPoll 和 fetchSize 允许您限制应为每个轮询处理的数字消息。在使用包含大量消息的文件夹时，这些参数应有助于防止性能不良。在以前的版本中，这些参数被评估太晚，因此大型邮箱仍然可能会导致性能问题。在 Camel 3.1 时，这些参数会在轮询期间提前评估，以避免这些问题。

86.19. 使用带有其他 JAVA MAIL SENDER 属性的标头

在发送邮件时，您可以通过以 java.smtp. 开头的键为来自 Exchange 的 JavaMailSender 提供动态 java 邮件属性。

您可以设置任何 java.smtp 属性，您可以在 Java 邮件文档中找到。

例如，要在 java.smtp.from (SMTP MAIL 命令)中提供动态 uuid：

```
  .setHeader("from", constant("reply2me@foo.com"));
  .setHeader("java.smtp.from", method(UUID.class, "randomUUID"));
  .to("smtp://mymailserver:1234");
```

注意

这只在使用自定义 JavaMailSender 时被支持。

86.20. SPRING BOOT AUTO-CONFIGURATION

组件支持 50 个选项，如下所列。

Name	描述	默认值	类型
camel.component.mail.additional-java-mail-properties	设置其他 java 邮件属性，这将根据所有其他选项附加/覆盖设置的任何默认属性。如果您需要添加一些特殊选项，但希望将其他选项保留原样，这将非常有用。选项是一个 java.util.Properties 类型。		Properties
camel.component.mail.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 text/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
camel.component.mail.attachments-content-transfer-encoding-resolver	使用自定义 AttachmentsContentTransferEncodingResolver 来解决要用于附件的 content-type-encoding。选项是 org.apache.camel.component.mail.AttachmentsContentTransferEncodingResolver 类型。		AttachmentsContentTransferEncodingResolver
camel.component.mail.authenticator	用于登录的验证器。如果设置，则忽略密码和用户名。可用于可过期的令牌，因此必须动态读取。选项是 org.apache.camel.component.mail.MailAuthenticator 类型。		MailAuthenticator
camel.component.mail.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.mail.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.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.mail.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

Name	描述	默认值	类型
camel.component.mail.close-folder	消费者是否应该在轮询后关闭文件夹。将这个选项设置为 false，并且也具有 disconnect=false，然后消费者在轮询之间保持文件夹打开。	true	布尔值
camel.component.mail.configuration	设置邮件配置。选项是 org.apache.camel.component.mail.MailConfiguration 类型。		MailConfiguration
camel.component.mail.connection-timeout	连接超时（以毫秒为单位）。	30000	整数
camel.component.mail.content-type	邮件消息内容类型。对 HTML 邮件使用 text/html。	text/plain	字符串
camel.component.mail.content-type-resolver	确定文件附加的 Content-Type 解析器。选项是 org.apache.camel.component.mail.ContentTypeResolver 类型。		ContentTypeResolver
camel.component.mail.copy-to	在处理邮件后，可以使用指定名称将其复制到邮件文件夹中。您可以使用带有键 copyTo 的标头覆盖此配置值，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.debug-mode	在底层邮件框架上启用调试模式。默认情况下，SUN 邮件框架将调试消息记录到 System.out 中。	false	布尔值
camel.component.mail.decode-filename	如果设置为 true，则使用 MimeUtility.decodeText 方法来解码文件名。这类似于设置 JVM 系统属性 mail.mime.encodefilename。	false	布尔值
camel.component.mail.delete	处理消息后删除消息。这可以通过在邮件邮件中设置 DELETED 标志来完成。如果为 false，则设置 SEEN 标志。从 Camel 2.10 开始，您可以通过设置带有 key delete 的标头来覆盖此配置选项，以确定是否应删除邮件。	false	布尔值
camel.component.mail.disconnect	消费者在轮询后是否应断开。如果启用，它会强制 Camel 在每个轮询上进行连接。	false	布尔值
camel.component.mail.enabled	是否启用邮件组件的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.component.mail.fetch-size	设置在轮询期间要消耗的最大消息数。如果邮箱文件夹包含大量邮件，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，所有信息都会被使用。将值设为 0 是一个特殊角写，其中 Camel 将不会消耗任何消息。	-1	整数
camel.component.mail.folder-name	要轮询的文件夹。	INBOX	字符串
camel.component.mail.from	来自电子邮件地址。	camel@localhost	字符串
camel.component.mail.handle-failed-message	如果邮件消费者无法检索给定的邮件邮件，则此选项允许处理消费者的错误处理程序造成的异常。通过在消费者上启用网桥错误处理程序，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，并且批处理中没有邮件可由 Camel 路由。	false	布尔值
camel.component.mail.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头到 Camel 消息。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		HeaderFilterStrategy
camel.component.mail.ignore-unsupported-charset	选项，可在发送邮件时让 Camel 忽略本地 JVM 中不支持的 charset。如果不支持 charset，则 charset=XXX （其中 XXX 代表不受支持的 charset）会从 content 类型中删除，它依赖于平台默认。	false	布尔值
camel.component.mail.ignore-uri-scheme	选项，可在发送邮件时让 Camel 忽略本地 JVM 中不支持的 charset。如果不支持 charset，则 charset=XXX （其中 XXX 代表不受支持的 charset）会从 content 类型中删除，它依赖于平台默认。	false	布尔值
camel.component.mail.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。选项是一个 java.util.Properties 类型。		Properties
camel.component.mail.java-mail-sender	使用自定义 org.apache.camel.component.mail.JavaMailSender 来发送电子邮件。选项是 org.apache.camel.component.mail.JavaMailSender 类型。		JavaMailSender

Name	描述	默认值	类型
camel.component.mail.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.mail.map-mail-message	指定 Camel 是否应该将接收的邮件映射到 Camel body/headers/attachments。如果设置为 true，邮件消息的正文映射到 Camel IN 消息的正文，邮件标头映射到 IN 标头，并附加到 Camel IN attachment 消息。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)来检索此原始消息。	true	布尔值
camel.component.mail.mime-decode-headers	这个选项为邮件标头启用透明 MIME 解码和取消折叠。	false	布尔值
camel.component.mail.move-to	在处理邮件后，可以将其移动到具有指定名称的邮件文件夹中。您可以使用带有键 moveTo 的标头覆盖此配置值，允许您将消息移到运行时配置的文件夹名称。		字符串
camel.component.mail.password	用于登录的密码。另请参阅 setAuthenticator (MailAuthenticator)。		字符串
camel.component.mail.peek	在处理邮件之前，将 javax.mail.Message 标记为 peeked。这只适用于 IMAPMessage 消息类型。通过使用 peek，邮件不会在邮件服务器上标记为 SEEN，如果 Camel 中存在错误处理，则我们就可以回滚邮件。	true	布尔值
camel.component.mail.reply-to	Reply-To 接收者（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.session	指定 camel 应该用于所有邮件交互的邮件会话。在邮件会话由某些其他资源创建和管理的情况（如 hundreds 容器）时很有用。使用自定义邮件会话时，将使用来自邮件会话的主机名和端口（如果在会话中进行了配置）。选项是 javax.mail.Session 类型。		会话
camel.component.mail.skip-failed-message	如果邮件使用者无法检索给定的邮件邮件，则此选项允许跳过邮件并继续进行 来检索下一个邮件。默认行为是消费者抛出异常，并且批处理中没有邮件可由 Camel 路由。	false	布尔值

Name	描述	默认值	类型
camel.component.mail.ssl-context-parameters	使用 SSLContextParameters 配置安全性。选项是 org.apache.camel.support.jsse.SSLContextParameters 类型。		SSLContextParameters
camel.component.mail.subject	发送的消息的主题。注：在标头中设置主题优先于这个选项。		字符串
camel.component.mail.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.unseen	是否仅受不可预见的邮件的限制。	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.component.mail.use-inline-attachments	是否使用分布内联或附加。	false	布尔值
camel.component.mail.username	用于登录的用户名。另请参阅 setAuthenticator (MailAuthenticator)。		字符串
camel.dataformat.mime-multipart.binary-content	定义 MIME 多部分的内容是否为二进制(true)还是 Base-64 编码(false)默认值。	false	布尔值
camel.dataformat.mime-multipart.enabled	是否启用 mime-multipart 数据格式的自动配置。这默认是启用的。		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，或者设置为 Camel 标头(false)。默认值为 false。	false	布尔值
camel.dataformat.mime-multipart.include-headers	定义哪些 Camel 标头也包含在 MIME 多部分的正则表达式。这只有在 headersInline 设置为 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。默认为混合的。	mixed	字符串

Name	描述	默认值	类型
camel.dataformat.mime-multipart.multipart-without-attachment	定义消息是否也被分成一个 MIME 多部分（只有一个正文部分）。默认值为 false。	false	布尔值

第 87 章 邮件 MICROSOFT OAUTH

自 Camel 3.18.4 起。

Mail Microsoft OAuth2 提供了 org.apache.camel.component.mail.MailAuthenticator 的实现，以验证 IMAP/POP/SMTP 连接，并通过 Spring 邮件支持和底层 JavaMail 系统访问电子邮件。

87.1. 依赖项

将此组件的 pom.xml 添加以下依赖项：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-mail-microsoft-oauth</artifactId>
</dependency>
```

导入 camel-mail-microsoft-oauth 将自动导入 camel-mail 组件。

87.2. MICROSOFT EXCHANGE ONLINE OAUTH2 MAIL AUTHENTICATOR IMAP 示例

要使用 OAuth，应用程序必须使用 Azure Active Directory 注册。按照说明注册新应用程序。

流程

1. 使应用程序能够通过客户端凭证流访问 Exchange 邮箱。如需更多信息，请参阅使用 OAuth 验证 IMAP、POP 或 SMTP 连接

2. 设置所有内容后，在 registry 中声明和注册，即 org.apache.camel.component.mail.MicrosoftExchangeOnlineOAuth2MailAuthenticator 实例。

3. 例如，在 Spring Boot 应用程序中：

```
@BindToRegistry("auth")
public MicrosoftExchangeOnlineOAuth2MailAuthenticator exchangeAuthenticator(){
  return new MicrosoftExchangeOnlineOAuth2MailAuthenticator(tenantId, clientId,
```

```
clientSecret, "jon@doe.com");
}
```

1. 然后，在 Camel URI 中引用它，如下所示：

```
from("imaps://outlook.office365.com:993"
        + "?authenticator=#auth"
        + "&mail.imaps.auth.mechanisms=XOAUTH2"
        + "&debugMode=true"
        + "&delete=false")
```

第 88 章 MAPSTRUCT

Since Camel 3.19

仅支持生成者。

camel-mapstruct 组件用于使用 MapStruct 转换 POJO。

88.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 mapstruct 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-mapstruct-starter</artifactId>
</dependency>
```

88.2. URI 格式

```
mapstruct:className[?options]
```

其中 className 是要转换为的 POJO 的完全限定类名称。

88.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

88.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设

置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

88.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

88.4. 组件选项

MapStruct 组件支持 4 个选项，如下所列。

Name	描述	default	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
mapperPackageName (producer)	Camel 应该发现映射类所需的软件包名称。**可以使用逗号分隔多个软件包名称。**		字符串

Name	描述	default	类型
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
mapStructConverter (advanced)	**Autowired** 使用自定义 MapStructConverter，例如适应特殊的运行时。		MapStructMapperFinder

88.5. 端点选项

MapStruct 端点使用 URI 语法进行配置：

```
mapstruct:className
```

使用以下路径和查询参数：

88.5.1. 路径参数(1 参数)

Name	描述	default	类型
classname (producer)	**必需** 映射结构的 POJO 的完全限定类名称（目标）。		字符串

88.5.2. 查询参数(2 参数)

Name	描述	default	类型
mandatory (producer)	是否必须存在映射转换器才能转换为 POJO。	true	布尔值
lazyStartProducer (producer (advanced))	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

88.6. 设置 MAPSTRUCT

camel-mapstruct 组件必须配置有一个或多个软件包名称，用于 classpath 扫描 MapStruct Mapper 类。这是必要的，因为映射类将用于使用 MapStruct 转换 POJO。

例如，要设置两个软件包，您可以执行以下操作：

```
MapstructComponent mc = context.getComponent("mapstruct", MapstructComponent.class);
mc.setMapperPackageName("com.foo.mapper,com.bar.mapper");
```

这也可以在 application.properties 中配置：

```
camel.component.mapstruct.mapper-package-name = com.foo.mapper,com.bar.mapper
```

Camel 将在启动时扫描这些软件包以 映射程序 结尾的类。这些类随后被内省，以发现映射方法。这些映射方法随后被注册到 Camel registry 中。这意味着，您也可以使用类型转换器将 POJOs 与 MapStruct 一起使用，例如：

```
from("direct:foo")
  .convertBodyTo(MyFooDto.class);
```

其中 MyFooDto 是一个 POJO，mapStruct 能够转换到/从其中。

88.7. SPRING BOOT AUTO-CONFIGURATION

组件支持 5 个选项，如下所列。

Name	描述	default	类型
camel.component.mapstruct.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.mapstruct.enabled	是否启用 mapstruct 组件的自动配置。这默认是启用的。		布尔值

Name	描述	default	类型
camel.component.mapstruct.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.mapstruct.map-struct-converter	使用自定义 MapStructConverter，如适应特殊运行时。选项是 org.apache.camel.component.mapstruct.MapStructMapperFinder 类型。		MapStructMapperFinder
camel.component.mapstruct.mapper-package-name	Camel 应该发现 Mapstruct 映射类的软件包名称。可以使用逗号分隔多个软件包名称。		字符串

第 89 章 MASTER

仅支持消费者

Camel-Master 端点提供了确保集群中只有一个消费者从给定端点使用的方法；如果该 JVM 中断，自动故障转移。

如果您需要从某些传统后端使用，这些后端不支持并发消耗，或者由于商业或稳定性原因，您可以在任何时间点上都只有一个连接。

89.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 master 时，请确保使用以下 Maven 依赖项来支持自动配置：

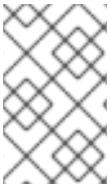
```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-master-starter</artifactId>
</dependency>
```

89.2. 使用 MASTER 端点

只需为任何带有 master:someName: 的 camel 端点添加前缀，其中 someName 是逻辑名称，用于获取 master 锁定。例如。

```
from("master:cheese:jms:foo").to("activemq:wine");
```

在本例中，master 组件可确保路由在集群中的任何给定时间只在一个节点中活跃。因此，如果集群中有 8 个节点，则 master 组件会将一个路由选为领导，只有此路由处于活动状态，因此只有此路由会消耗来自 jms:foo 的消息。如果此路由被停止或意外终止，则 master 组件将检测到这一点，并重新检查另一个节点处于活动状态，然后变为活动状态并开始使用 jms:foo 的消息。



注意

Apache ActiveMQ 5.x 具有此类功能，称为 Exclusive Consumers。

89.3. URI 格式

```
master:namespace:endpoint[?options]
```

其中 endpoint 是您要以主/从模式运行的任何 Camel 端点。

89.4. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

89.4.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

89.4.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

89.5. 组件选项

Master 组件支持 4 个选项，如下所列。

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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
service (advanced)	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

89.6. 端点选项

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：

89.6.1. 路径参数(2 参数)

Name	描述	默认值	类型
namespace (consumer)	**必需** 要使用的命名空间的名称。		字符串

Name	描述	默认值	类型
delegateUri (consumer)	**必需** 在 master/slave 模式中使用的端点 uri。		字符串

89.6.2. 查询参数(3 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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern

89.7. 示例

您可以保护集群的 Camel 应用程序，使其只消耗一个活动节点的文件。

```
// 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
```

89.8. 实现

Camel 提供以下 ClusterService 实现：

- *camel-consul*
- *camel-file*
- *camel-infinispan*
- *camel-jgroups-raft*
- *camel-jgroups*
- *camel-kubernetes*
- *camel-zookeeper*

89.9. 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.master.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.master.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

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.service	注入要使用的服务。选项是 org.apache.camel.cluster.CamelClusterService 类型。		CamelClusterService
camel.component.master.service-selector	注入用于查找要使用的 CamelClusterService 的服务选择器。选项是一个 org.apache.camel.cluster.CamelClusterService.Selector 类型。		CamelClusterService$Selector

第 90 章 MICROMETER

从 Camel 2.22 开始

仅支持生成者

Micrometer 组件允许从 Camel 路由直接收集各种指标。支持的指标类型是 计数器、摘要和 计时器。Micrometer 提供测量应用行为的简单方法。可配置报告后端（通过 Micrometer registry）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 MicrometerRoutePolicyFactory，它允许使用 Micrometer 和 EventNotifier 实施来公开路由统计信息，以计算从创建到其完成的时间交换。

90.1. 依赖项

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.springboot</groupId>
  <artifactId>camel-micrometer</artifactId>
</dependency>
```

同时使用以下命令更新 dependencyManagement 部分：

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>com.redhat.camel.springboot.platform</groupId>
            <artifactId>camel-spring-boot-bom</artifactId>
            <version>${camel-spring-boot-version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

90.2. URI 格式

```
micrometer:[ counter | summary | timer ]:metricname[?options]
```

90.3. 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

90.3.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

90.3.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

90.4. 组件选项

***Micrometer** 组件支持 3 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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	True	布尔值
metricsRegistry (advanced)	使用自定义配置的 MetricRegistry。		MeterRegistry

90.5. 端点选项

***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：

90.5.1. 路径参数(3 参数)

Name	描述	默认值	类型
metricsType (producer)	**所需** 指标类型。 Enum 值： • 计数 • summary • timer		类型

Name	描述	默认值	类型
metricsName (producer)	**所需** 指标名称。		字符串
tags (producer)	指标的标签。		可迭代

90.5.2. 查询参数(6 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。 Enum 值： • 开始 • stop		字符串
decrement (producer)	使用计数器类型时减少值表达式。		字符串
increment (producer)	使用计数器类型时递增值表达式。		布尔值
metricsDescription (producer)	指标描述。		字符串
value (producer)	使用直方图类型时的值表达式。		字符串
lazyStartProducer (producer (advanced))	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

90.6. 消息标头

Micrometer 组件支持 7 个消息标头，如下所列：

Name	描述	默认值	类型
CamelMetricsTimerAction (producer) 常量 ： HEADER_TIMER_ACTION	覆盖 URI 中的计时器操作。 Enum 值： • 开始 • stop		MicrometerTimerAction
CamelMetricsHistogramValue (producer) constant: HEADER_HISTOGRAM_VALUE	覆盖 URI 中的直方图值。		Long
CamelMetricsCounterDecrement (producer) 常量 ： HEADER_COUNTER_DECREMENT	覆盖 URI 中的减少值。		å�Œ
CamelMetricsCounterIncrement (producer) 常量 ： HEADER_COUNTER_INCREMENT	覆盖 URI 中的递增值。		å�Œ
CamelMetricsName (producer) 常量 ： HEADER_METRIC_NAME	覆盖 URI 中的 name 值。		字符串
CamelMetricsDescription (producer) 常量 ： HEADER_METRIC_DESCRIPTION	覆盖 URI 中的 description 值。		字符串
CamelMetricsTags (producer) 常量 ： HEADER_METRIC_TAGS	增强定义为 URI 参数的量表标签。		可迭代

90.7. 计量 REGISTRY

默认情况下，Camel Micrometer 组件会创建一个 SimpleMeterRegistry 实例，主要用于测试。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 可用于解决多个监控目标。

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 = MicrometerConstants.METRICS_REGISTRY_NAME)
    public MeterRegistry getMeterRegistry() {
        CompositeMeterRegistry registry = ...;
        registry.add(...);
        // ...
        return registry;
    }
}
```

或使用 CDI:

```
 @Override
    public void configure() {
      from("...")
          // Register the 'my-meter' meter in the MetricRegistry below
          .to("micrometer:meter:my-meter");
    }

    @Produces
    // If multiple MetricRegistry beans
    // @Named(MicrometerConstants.METRICS_REGISTRY_NAME)
    MetricRegistry registry() {
        CompositeMeterRegistry registry = ...;
        registry.add(...);
        // ...
        return registry;
    }
}
```

90.8. 默认 CAMEL 指标

一些特定于 Camel 的指标会开箱即用。

Name	类型	描述
camel.message.history	计时器	启用消息历史记录时，路由中每个节点的性能示例
camel.routes.added	量表	添加的路由数
camel.routes.running	量表	运行的路由数
camel.exchanges.inflight	量表	路由 inflight 信息
camel.exchanges.total	计数	处理的交换总数
camel.exchanges.succeeded	计数	成功完成交换的数量
camel.exchanges.failed	计数	失败的交换数
camel.exchanges.failures.handled	计数	处理的失败数
camel.exchanges.external.redeliveries	计数	外部启动的重新eliveries （如来自 JMS 代理）
camel.exchange.event.notifier	量表 + Summary	创建、发送、完成和失败的事件的消息的指标
camel.route.policy	量表 + Summary	路由性能指标
camel.route.policy.long.task	量表 + Summary	路由长任务指标

90.9. 使用制作者

每个计量都有类型和名称。支持的类型有 计数器、分发概述 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（请参见下面的），这还允许根据交换数据选择量表。

可选的 tags URI 参数是一个以逗号分隔的字符串，由 key=value 表达式组成。 键和值 都是字符串，它也被评估为 Simple 表达式。例如， URI 参数 tags=X=${header.Y} 会将标头 Y 的当前值分配给密钥 X。

90.9.1. Headers

URI 中定义的计量名称可以通过填充名称 CamelMetricsName 的标头来覆盖。定义为 URI 参数的 meter 标签可以通过填充名称 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("micrometer:counter:name.not.used?tags=key=value")
    .to("direct:out");
```

将更新名称为 new.name 而不是 name.not. 的计数器，除了标签键值为 外，使用标签 dynamic- key 和值 dynamic- value。

当 Micrometer 端点完成交换处理后，所有特定于指标的标头都会从消息中删除。在处理交换 Micrometer 端点时，将使用级别 warn 来捕获所有异常和写入日志条目。

90.10. 计数

```
micrometer:counter:name[?options]
```

90.10.1. 选项

Name	default	描述
increment		要添加到计数器的双值
decrement		从计数器中减去的双引号

如果未定义 递增 或减少，则计数器值将递增。如果 递增 和减少，则仅调用递增操作。

```
// 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 值被评估为 Simple 表达式，并带有 Double 结果，例如，如果标头 X 包含一个评估为 3.0 的值，则 simple.counter 计数器会减少 3.0：

```
// decrement counter simple.counter by 3
from("direct:in")
    .to("micrometer:counter:simple.counter?decrement=${header.X}")
    .to("direct:out");
```

90.10.2. Headers

与 camel-metrics 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
CamelMetricsCounterIncrement		要添加到计数器的双值
CamelMetricsCounterDecrement		从计数器中减去的双引号

```
from("direct:in")
    .setHeader(MicrometerConstants.HEADER_COUNTER_INCREMENT, constant(417.0D))
    .to("micrometer:counter:simple.counter?increment=7")
    .to("direct:out");
```

```
from("direct:in")
    .setHeader(MicrometerConstants.HEADER_COUNTER_INCREMENT,
simple("${body.length}"))
    .to("micrometer:counter:body.length")
    .to("direct:out");
```

90.11. 发行版概述

```
micrometer:summary:metricname[?options]
```

90.11.1. 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 的简单 表达式，例如，如果标头 X 包含一个评估为 3.0 的值，这个值使用 simple.histogram 注册 ：

```
from("direct:in")
    .to("micrometer:summary:simple.histogram?value=${header.X}")
    .to("direct:out");
```

90.11.2. Headers

与 camel-metrics 一样，可以使用特定的 Message 标头来覆盖 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")
```

90.12. 计时器

```
micrometer:timer:metricname[?options]
```

90.12.1. 选项

Name	default	描述
action		启动或停止

如果没有提供 操作 或无效值，则会在没有任何计时器更新的情况下记录警告。如果已经运行的计时器或 stop 在未知计时器上调用 action start，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 对象作为不同指标组件调用之间的 Exchange 属性保存。

操作 被评估为一个简单的表达式，返回类型为 MicrometerTimerAction 的结果。

90.12.2. Headers

与 camel-metrics 一样，可以使用特定的 Message 标头来覆盖 Micrometer 端点 URI 中指定的操作值。

Name	描述	预期类型
CamelMetricsTimerAction	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");
```

90.13. 使用 MICROMETER 路由策略工厂

MicrometerRoutePolicyFactory 允许为每个路由添加一个 RoutePolicy，以便使用 Micrometer 来公开路由利用率统计。此工厂可以在 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，您可以使用 MicrometerRoutePolicy Factory 定义您要检测的每个路由的专用 MicrometerRoutePolicy。

从 Java，您刚刚将工厂添加到 CamelContext 中，如下所示：

```
context.addRoutePolicyFactory(new MicrometerRoutePolicyFactory());
```

在 XML DSL 中，您可以定义一个 <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打印
meterRegistry		允许使用共享的 MeterRegistry。如果没有提供任何服务，则 Camel 将创建一个由此 CamelContext 使用的共享实例。
durationUnit	TimeUnit.MILLISECONDS	将统计信息转储为 json 时用于持续时间的单位。
配置	请参见以下	MicrometerRoutePolicyConfiguration.class

MicrometerRoutePolicyConfiguration 支持以下选项：

Name	default	描述
additionalCounters	true	激活所有附加计数器
exchangesSucceeded	true	激活成功交换的计数器
exchangesFailed	true	激活失败的交换计数器

Name	default	描述
exchangesTotal	true	激活交换总数的计数器
externalRedeliveries	true	激活交换的重新设计计数器
failuresHandled	true	激活处理失败的计数器
longTask	false	激活较长的任务计时器（当前处理微主题时间）
timerInitiator	null	用于自定义初始化计时器的 consumer<Timer.Builder>
longTaskInitiator	null	consumer<LongTaskTimer.Builder> 用于自定义初始化 LongTaskTimer

如果在 CamelContext 中启用了 JMX，则使用 name=MicrometerRoutePolicy 在 type=services 树中注册 MBean。

90.14. 使用 MICROMETER 消息历史记录工厂

MicrometerMessageHistoryFactory 允许在路由消息时使用指标捕获消息历史性能统计信息。它通过使用 Micrometer Timer 用于所有路由中的每个节点。此工厂可以在 Java 和 XML 中使用，如下例所示。

从 Java，您刚刚将工厂设置为 CamelContext，如下所示：

```
context.setMessageHistoryFactory(new MicrometerMessageHistoryFactory());
```

在 XML DSL 中，您可以定义一个 <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打印
meterRegistry		允许使用共享的 **MeterRegistry**。如果没有提供任何服务，则 Camel 将创建一个由此 CamelContext 使用的共享实例。
durationUnit	TimeUnit.MILLISECONDS	将统计信息转储为 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，允许以 json 输出的形式收集数据。

从 Java 代码中，您可以从 CamelContext 获取该服务，如下所示：

```
MicrometerMessageHistoryService service =
context.hasService(MicrometerMessageHistoryService.class);
String json = service.dumpStatisticsAsJson();
```

如果在 CamelContext 中启用了 JMX，MBean 会在 type=services 树中注册，其名称为 =MicrometerMessageHistory。

90.15. MICROMETER 事件通知

有一个 MicrometerRouteEventNotifier （添加的和运行路由）和一个 MicrometerExchangeEventNotifier （从创建到完成的交换器）。

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，允许以 json 输出的形式收集数据。

从 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，MBean 会在 type=services 树中注册，名称为 name=MicrometerEventNotifier。

90.16. 检测 CAMEL 线程池

InstrumentedThreadPoolFactory 允许您通过注入 InstrumentedThreadPoolFactory （从 Camel 内部收集信息）来收集 Camel 线程池的性能信息。有关使用 Spring 的 CamelContext 高级配置 的更多详细信息。

90.17. 在 JMX 中公开 MICROMETER 统计信息

Micrometer 使用 MeterRegistry 实施来发布统计信息。在生产环境中，建议选择专用后端，如 Prometheus 或 Graphite，但测试或本地部署可能足以将统计信息发布到 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 = MicrometerConstants.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 策略决定如何将计量名称和标签汇编成 MBean 名称。

90.18. 在 SPRING BOOT 中使用 CAMEL MICROMETER

当您将 camel-micrometer-starter 用于 Spring Boot 时，如果有 io.micrometer.core.instrument.MeterRegistry，则 Spring Boot 自动配置将自动启用指标捕获。

例如，要使用 Prometheus 捕获数据，您可以添加以下依赖项：

```
<dependency>
  <groupId>io.micrometer</groupId>
  <artifactId>micrometer-registry-prometheus</artifactId>
</dependency>
```

下表列出了要捕获的选项，或者指定要关闭的指标。

90.19. SPRING BOOT 自动配置

与普通的 camel 微主题器相比，Spring Boot 上的微主题组件提供了 10 个更多选项，如下所列：

Name	描述	默认值	类型
camel.component.micrometer.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	True	布尔值
camel.component.micrometer.enabled	是否启用 micrometer 组件的自动配置。这默认是启用的。		布尔值
camel.component.micrometer.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

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。		MeterRegistry
camel.metrics.enable-exchange-event-notifier	设置是否启用 MicrometerExchangeEventNotifier 来捕获交换处理时间的指标。	True	布尔值
camel.metrics.enable-message-history	设置是否启用 MicrometerMessageHistoryFactory，以便在单个路由节点处理时间上捕获指标。根据配置的路由节点数量，可能会创建大量指标。因此，默认禁用这个选项。	False	布尔值
camel.metrics.enable-route-event-notifier	设置是否启用 MicrometerRouteEventNotifier，以捕获路由总数和正在运行的路由总数。	True	布尔值
camel.metrics.enable-route-policy	设置是否启用 MicrometerRoutePolicyFactory 来捕获路由处理时间的指标。	True	布尔值
camel.metrics.uri-tag-dynamic	是否在捕获的指标中将静态或动态值用于 URI 标签。使用动态标签时，带有基本 URL 的 REST 服务：/users/{id} 将使用 uri 标签捕获带有实际动态值（如 /users/123）的指标。但是，这可能会导致许多标签作为 URI 为动态的，因此请谨慎使用。	False	布尔值
camel.metrics.uri-tag-enabled	是否应该在捕获的指标中启用 HTTP uri 标签。如果禁用，则 uri 标签，可能无法解析，并被标记为 UNKNOWN。	True	布尔值

第 91 章 MINIO

从 Camel 3.5 开始

支持生成者和消费者

Minio 组件支持从/到 Minio 服务存储和检索对象。

91.1. 先决条件

您必须具有有效的凭证，才能授权存储桶/文件夹的访问权限。如需更多信息，请访问 Minio。

91.2. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 minio 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-minio-starter</artifactId>
</dependency>
```

91.3. URI 格式

```
minio://bucketName[?options]
```

如果存储桶不存在，则会创建存储桶。您可以以以下格式将查询选项附加到 URI 中，

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

91.4. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

91.4.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

91.4.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

91.5. 组件选项

Minio 组件支持 47 选项，如下所列。

Name	描述	默认值	类型
autoCreateBucket (common)	如果存储桶名称不存在，则设置存储桶的自动创建。	true	布尔值
configuration (common)	组件配置。		MinioConfiguration
customHttpClient (common)	为经过身份验证的访问设置自定义 HTTP 客户端。		OkHttpClient
endpoint (common)	端点可以是 URL、域名、IPv4 地址或 IPv6 地址。		字符串
minioClient (common)	**Autowired** 对 registry 中的 Minio Client 对象的引用。		MinioClient
objectLock (common)	创建新存储桶时设置。	false	布尔值
policy (common)	此队列的策略在方法中设置。		字符串
proxyPort (common)	TCP/IP 端口号。80 和 443 用作 HTTP 和 HTTPS 的默认值。		整数
region (common)	Minio 客户端需要工作的区域。使用此参数时，配置将预期区域的小写名称（如 ap-east-1）。您需要使用名称 Region.EU_WEST_1.id （）。		字符串
secure (common)	指定使用到 minio 服务的安全连接的标志。	false	布尔值
ServerSideEncryption (common)	服务器端加密.		ServerSideEncryption
serverSideEncryptionCustomerKey (common)	复制/移动对象期间源对象的服务器端加密。		ServerSideEncryptionCustomerKey
autocloseBody (consumer)	如果此选项为 true，并且 includeBody 为 true，则在交换完成时调用 MinioObject.close （）方法。此选项与 includeBody 选项密切相关。如果将 includeBody 设为 true，autocloseBody 设为 false，它将是关闭 MinioObject 流的调用者。将 autocloseBody 设置为 true，将自动关闭 MinioObject 流。	true	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
bypassGovernanceMode (consumer)	如果要在删除特定对象时绕过GovernanceMode，请设置此标志。	false	布尔值
deleteAfterRead (consumer)	在检索后，从 Minio 中删除对象。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 MinioConstants#BUCKET_NAME 和 MinioConstants#OBJECT_NAME 标头进行过滤，或者只过滤 MinioConstants#OBJECT_NAME 标头。	true	布尔值
delimiter (consumer)	ListObjectsRequest 中使用的分隔符仅消耗我们感兴趣的对象。		字符串
destinationBucketName (consumer)	源存储桶名称。		字符串
destinationObjectName (consumer)	源对象名称。		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，则标头将使用 Minio 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，autocloseBody 设为 false，它将是关闭 MinioObject 流的调用者。将 autocloseBody 设置为 true，将自动关闭 MinioObject 流。	true	布尔值
includeFolders (consumer)	ListObjectsRequest 中使用的标记，用于设置 include 文件夹。	false	布尔值
includeUserMetadata (consumer)	ListObjectsRequest 中使用的标记，用于获取用户元数据的对象。	false	布尔值
includeVersions (consumer)	ListObjectsRequest 中使用的标记，以获取带有版本控制的对象。	false	布尔值

Name	描述	默认值	类型
length (consumer)	对象数据的字节数。		long
matchETag (consumer)	为 get 对象设置 match ETag 参数。		字符串
maxConnections (consumer)	在 minio 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。获取最大消息数，作为每次轮询轮询的限制。默认值为 10。使用 0 或负数设置为无限。	10	int
modifiedSince (consumer)	为 get 对象设置修改后的参数。		ZonedDateTime
moveAfterRead (consumer)	在检索后，将对象从 bucket 移到不同的存储桶。要完成操作，必须设置 destinationBucket 选项。仅当 Exchange 提交时，才会执行复制存储桶操作。如果进行回滚，则对象不会被移动。	false	布尔值
notMatchETag (consumer)	为 get 对象设置与 ETag 参数不匹配。		字符串
objectName (consumer)	要从具有给定对象名称的存储桶获取对象。		字符串
offset (consumer)	开始对象数据的字节位置。		long
prefix (consumer)	对象名称以前缀开头。		字符串
递归 (consumer)	比目录结构模拟递归列出。	false	布尔值
startAfter (consumer)	在此对象名称后，列出 bucket 中的对象。		字符串
unModifiedSince (consumer)	为 get 对象设置未修改后的参数。		ZonedDateTime
useVersion1 (consumer)	为 true 时，使用 REST API 版本 1。	false	布尔值
versionId (consumer)	在删除对象时设置对象的特定版本_ID。		字符串

Name	描述	默认值	类型
deleteAfterWrite (producer)	在 Minio 文件上传后删除文件对象。	false	布尔值
KeyName (producer)	通过端点参数在存储桶中设置元素的密钥名称。		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
operation (producer)	当用户不希望只进行上传时，要执行的操作。 Enum 值： • copyObject • listObjects • deleteObject • deleteObjects • deleteBucket • listBuckets • getObject • getObjectRange		MinioOperations
pojoRequest (producer)	如果您想要将 POJO 请求用作正文。	false	布尔值
storageClass (producer)	请求中设置的存储类。		字符串
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
accessKey (security)	Amazon AWS Secret Access Key 或 Minio 访问密钥。如果没有设置 camel，则 camel 将连接到服务以进行匿名访问。		字符串

Name	描述	默认值	类型
secretKey (security)	Amazon AWS 访问密钥 Id 或 Minio Secret Key。如果没有设置 camel，则 camel 将连接到服务以进行匿名访问。		字符串

91.6. 端点选项

Minio 端点使用 URI 语法进行配置：

```
minio:bucketName
```

使用以下路径和查询参数：

91.6.1. 路径参数(1 参数)

Name	描述	默认值	类型
bucketName (common)	**所需的** Bucket 名称。		字符串

91.6.2. 查询参数(63 参数)

Name	描述	默认值	类型
autoCreateBucket (common)	如果存储桶名称不存在，则设置存储桶的自动创建。	true	布尔值
customHttpClient (common)	为经过身份验证的访问设置自定义 HTTP 客户端。		OkHttpClient
endpoint (common)	端点可以是 URL、域名、IPv4 地址或 IPv6 地址。		字符串
minioClient (common)	**Autowired** 对 registry 中的 Minio Client 对象的引用。		MinioClient
objectLock (common)	创建新存储桶时设置。	false	布尔值
policy (common)	此队列的策略在方法中设置。		字符串

Name	描述	默认值	类型
proxyPort (common)	TCP/IP 端口号。80 和 443 用作 HTTP 和 HTTPS 的默认值。		整数
region (common)	Minio 客户端需要工作的区域。使用此参数时，配置将预期区域的小写名称（如 ap-east-1）。您需要使用名称 Region.EU_WEST_1.id （）。		字符串
secure (common)	指定使用到 minio 服务的安全连接的标志。	false	布尔值
ServerSideEncryption (common)	服务器端加密.		ServerSideEncryption
serverSideEncryptionCustomerKey (common)	复制/移动对象期间源对象的服务器端加密。		ServerSideEncryptionCustomerKey
autocloseBody (consumer)	如果此选项为 true，并且 includeBody 为 true，则在交换完成时调用 MinioObject.close （）方法。此选项与 includeBody 选项密切相关。如果将 includeBody 设为 true，autocloseBody 设为 false，它将是关闭 MinioObject 流的调用者。将 autocloseBody 设置为 true，将自动关闭 MinioObject 流。	true	布尔值
bypassGovernanceMode (consumer)	如果要在删除特定对象时绕过GovernanceMode，请设置此标志。	false	布尔值
deleteAfterRead (consumer)	在检索后，从 Minio 中删除对象。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 MinioConstants#BUCKET_NAME 和 MinioConstants#OBJECT_NAME 标头进行过滤，或者只过滤 MinioConstants#OBJECT_NAME 标头。	true	布尔值
delimiter (consumer)	ListObjectsRequest 中使用的分隔符仅消耗我们感兴趣的对象。		字符串
destinationBucketName (consumer)	源存储桶名称。		字符串
destinationObjectName (consumer)	源对象名称。		字符串

Name	描述	默认值	类型
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，则标头将使用 Minio 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，autocloseBody 设为 false，它将是关闭 MinioObject 流的调用者。将 autocloseBody 设置为 true，将自动关闭 MinioObject 流。	true	布尔值
includeFolders (consumer)	ListObjectsRequest 中使用的标记，用于设置 include 文件夹。	false	布尔值
includeUserMetadata (consumer)	ListObjectsRequest 中使用的标记，用于获取用户元数据的对象。	false	布尔值
includeVersions (consumer)	ListObjectsRequest 中使用的标记，以获取带有版本控制的对象。	false	布尔值
length (consumer)	对象数据的字节数。		long
matchETag (consumer)	为 get 对象设置 match ETag 参数。		字符串
maxConnections (consumer)	在 minio 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。获取最大消息数，作为每次轮询轮询的限制。默认值为 10。使用 0 或负数设置为无限。	10	int
modifiedSince (consumer)	为 get 对象设置修改后的参数。		ZonedDateTime
moveAfterRead (consumer)	在检索后，将对象从 bucket 移到不同的存储桶。要完成操作，必须设置 destinationBucket 选项。仅当 Exchange 提交时，才会执行复制存储桶操作。如果进行回滚，则对象不会被移动。	false	布尔值
notMatchETag (consumer)	为 get 对象设置与 ETag 参数不匹配。		字符串
objectName (consumer)	要从具有给定对象名称的存储桶获取对象。		字符串
offset (consumer)	开始对象数据的字节位置。		long
prefix (consumer)	对象名称以前缀开头。		字符串

Name	描述	默认值	类型
递归 (consumer)	比目录结构模拟递归列出。	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
startAfter (consumer)	在此对象名称后，列出 bucket 中的对象。		字符串
unModifiedSince (consumer)	为 get 对象设置未修改后的参数。		ZonedDateTime
useVersion1 (consumer)	为 true 时，使用 REST API 版本 1。	false	布尔值
versionId (consumer)	在删除对象时设置对象的特定版本_ID。		字符串
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
deleteAfterWrite (producer)	在 Minio 文件上传后删除文件对象。	false	布尔值

Name	描述	默认值	类型
KeyName (producer)	通过端点参数在存储桶中设置元素的密钥名称。		字符串
operation (producer)	当用户不希望只进行上传时，要执行的操作。 Enum 值： • copyObject • listObjects • deleteObject • deleteObjects • deleteBucket • listBuckets • getObject • getObjectRange		MinioOperations
pojoRequest (producer)	如果您想要将 POJO 请求用作正文。	false	布尔值
storageClass (producer)	请求中设置的存储类。		字符串
lazyStartProducer (producer (advanced))	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
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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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
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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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 Secret Access Key 或 Minio 访问密钥。如果没有设置 camel，则 camel 将连接到服务以进行匿名访问。		字符串
secretKey (security)	Amazon AWS 访问密钥 Id 或 Minio Secret Key。如果没有设置 camel，则 camel 将连接到服务以进行匿名访问。		字符串

您必须在 Registry 或 accessKey 和 secretKey 中提供 minioClient，才能访问 Minio。

91.7. 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。

91.8. 消息标头

Minio 组件支持 21 个消息标头，如下所列：

Name	描述	默认值	类型
CamelMinioBucketName (common) 常量 ：BUCKET_NAME	producer：存储此对象的 bucket 名称，或用于当前操作。consumer ：包含此对象的存储桶的名称。		字符串
CamelMinioDestinationBucketName (producer) 常量 ：DESTINATION_BUCKET_NAME	用于当前操作的存储桶目标名称。		字符串
CamelMinioContentControl (common) 常数 ：CACHE_CONTROL	producer ：此对象的内容控制。consumer ：可选的 Cache-Control HTTP 标头，允许用户在 HTTP 请求/reply 链中指定缓存行为。		字符串
CamelMinioContentDisposition (common) 常数：CONTENT_DISPOSITION	producer：此对象的内容分布。consumer：可选的 Content-Disposition HTTP 标头，它指定要保存的对象的建议文件名。		字符串
CamelMinioContentEncoding (common) 常量 ：CONTENT_ENCODING	producer ：此对象的内容编码。consumer: 可选的 Content-Encoding HTTP 标头，指定将什么内容编码应用到对象，以及必须应用哪些解码机制来获取 Content-Type 字段引用的 media-type。		字符串
CamelMinioContentLength (common) 常量 ：CONTENT_LENGTH	producer ：此对象的内容长度。consumer: Content-Length HTTP 标头表示关联的对象的大小（以字节为单位）。		Long
CamelMinioContentMD5 (common) 常数：CONTENT_MD5	producer：此对象的 md5 checksum。consumer：根据 RFC 1864，对相关对象(content - 不包括标头)的 base64 编码 128 位 MD5 摘要。此数据用作消息完整性检查，以验证 Minio 收到的数据是否与调用者发送的数据相同。		字符串

Name	描述	默认值	类型
CamelMinioContentType (common) 常数：CONTENT_TYPE	producer ：此对象的内容类型。consumer ：Content-Type HTTP 标头，它指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。		字符串
CamelMinioETag (common) 常数： E_TAG	producer：新上传对象的 ETag 值。consumer：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 Minio 发送的数据相同。		字符串
CamelMinioObjectName (common) 常数 ：OBJECT_NAME	producer：存储此对象或用于当前操作的密钥。consumer ：存储此对象的密钥。		字符串
CamelMinioDestinationObjectName (producer) 常量 ：DESTINATION_OBJECT_NAME	用于当前操作的 Destination 键。		字符串
CamelMinioLastModified (common) 常数：LAST_MODIFIED	producer：此对象的最后修改的时间戳。consumer ：Last-Modified 标头的值，指示 Minio 最后记录对相关对象的修改的日期和时间。		Date
CamelMinioStorageClass (producer) 恒定：STORAGE_CLASS	此对象的存储类。		字符串
CamelMinioVersionId (common) 常量：VERSION_ID	producer：要存储或从当前操作返回的对象的版本 Id。consumer ：关联的 Minio 对象的版本 ID （如果可用）。只有在对象上传到启用了对象版本控制的 Minio 存储桶时，才会将版本 ID 分配给对象。		字符串
CamelMinioCannedAcl (producer) 常量：CANNED_ACL	应用到对象的 canned acl。有关允许的值，请参阅 com.amazonaws.services.s3.model.CannedAccessControlList。		字符串

Name	描述	默认值	类型
CamelMinioOperation (producer) 常数 ：MINIO_OPERATION	要执行的操作。 Enum 值： • copyObject • listObjects • deleteObject • deleteObjects • deleteBucket • listBuckets • getObject • getPartialObject		MinioOperations
CamelMinioServerSideEncryption (common) 常数 ：SERVER_SIDE_ENCRYPTION	producer ：在使用 Minio-managed 密钥加密对象时设置服务器端加密算法。例如，使用 AES256。 consumer ：使用 Minio 管理的密钥加密对象时的服务器端加密算法。		字符串
CamelMinioExpirationTime (common) 恒定：EXPIRATION_TIME	过期时间。		字符串
CamelMinioReplicationStatus (common) 常数 ：REPLICATION_STATUS	复制状态。		字符串
CamelMinioOffset (producer) 常数 ： OFFSET	偏移。		字符串

Name	描述	默认值	类型
CamelMinioLength (producer) 常数 ： LENGTH	长度。		字符串

91.8.1. minio Producer 操作

camel-Minio 组件在制作者端提供以下操作：

- copyObject
- deleteObject
- deleteObjects
- listBuckets
- deleteBucket
- listObjects
- GetObject （这将返回 MinioObject 实例）
- getObjectRange （这将返回 MinioObject 实例）

91.8.2. 高级 Minio 配置

如果您的 Camel 应用程序在防火墙后面运行，或者需要对 MinioClient 实例配置有更多控制，您可以创建自己的实例并在 Camel minio 组件配置中引用它：

```
from("minio://MyBucket?minioClient=#client&delay=5000&maxMessagesPerPoll=5")
.to("mock:result");
```

91.8.3. minio Producer 操作示例

- CopyObject：此操作将对象从一个存储桶复制到不同的存储桶

```
from("direct:start").process(new Processor() {

    @Override
    public void process(Exchange exchange) throws Exception {
        exchange.getIn().setHeader(MinioConstants.DESTINATION_BUCKET_NAME,
"camelDestinationBucket");
        exchange.getIn().setHeader(MinioConstants.OBJECT_NAME, "camelKey");
        exchange.getIn().setHeader(MinioConstants.DESTINATION_OBJECT_NAME,
"camelDestinationKey");
    }
})
.to("minio://mycamelbucket?minioClient=#minioClient&operation=copyObject")
.to("mock:result");
```

此操作会将带有标头 camelDestinationKey 中的名称的对象复制到 Bucket mycamelbucket 中的 camelDestinationBucket 存储桶。

- DeleteObject：此操作从存储桶中删除对象

```
from("direct:start").process(new Processor() {

    @Override
    public void process(Exchange exchange) throws Exception {
        exchange.getIn().setHeader(MinioConstants.OBJECT_NAME, "camelKey");
    }
})
.to("minio://mycamelbucket?minioClient=#minioClient&operation=deleteObject")
.to("mock:result");
```

此操作将从 bucket mycamelbucket 中删除对象 camelKey。

- ListBuckets：此操作列出了此区域中此帐户的存储桶

```
from("direct:start")
.to("minio://mycamelbucket?minioClient=#minioClient&operation=listBuckets")
.to("mock:result");
```

此操作将列出此帐户的存储桶

- DeleteBucket：此操作删除指定为 URI 参数或标头的存储桶

```
from("direct:start")
.to("minio://mycamelbucket?minioClient=#minioClient&operation=deleteBucket")
.to("mock:result");
```

此操作将删除存储桶 mycamelbucket

- ListObjects ：此操作列表在特定存储桶中的对象

```
from("direct:start")
.to("minio://mycamelbucket?minioClient=#minioClient&operation=listObjects")
.to("mock:result");
```

此操作将列出 mycamelbucket bucket 中的对象

- GetObject：此操作获取特定存储桶中的单个对象

```
from("direct:start").process(new Processor() {

    @Override
    public void process(Exchange exchange) throws Exception {
        exchange.getIn().setHeader(MinioConstants.OBJECT_NAME, "camelKey");
    }
})
.to("minio://mycamelbucket?minioClient=#minioClient&operation=getObject")
.to("mock:result");
```

此操作将返回与 mycamelbucket bucket 中 camelKey 对象相关的 MinioObject 实例。

- GetObjectRange：此操作获得特定存储桶中的单个对象范围

```
from("direct:start").process(new Processor() {

    @Override
    public void process(Exchange exchange) throws Exception {
        exchange.getIn().setHeader(MinioConstants.OBJECT_NAME, "camelKey");
        exchange.getIn().setHeader(MinioConstants.OFFSET, "0");
        exchange.getIn().setHeader(MinioConstants.LENGTH, "9");
    }
```

```
    })
    .to("minio://mycamelbucket?minioClient=#minioClient&operation=getObjectRange")
    .to("mock:result");
```

此操作将返回与 mycamelbucket bucket 中 camelKey 对象相关的 MinioObject 实例，其中包含从 0 到 9 的字节数。

91.9. BUCKET 自动创建

使用选项 autoCreateBucket 用户可以在 Minio Bucket 不存在时避免自动创建。此选项的默认值是 true。如果设置为 false 任何在 Minio 中不存在的存储桶的操作，则会返回错误。

91.10. 在 REGISTRY 中自动检测 MINIO 客户端

组件可以检测在 registry 中存在 Minio bean。如果这是该类型的唯一实例，它将用作客户端，您不必将其定义为 uri 参数，如上例所示。这对端点的智能配置非常有用。

91.11. 在存储桶和其他存储桶间移动操作

有些用户（如从存储桶中消耗大量），并在不同的中移动内容，而无需使用这个组件的 copyObject 功能。如果是这样，请不要忘记从消费者的传入交换中删除 bucketName 标头，否则该文件将始终覆盖同一原始存储桶。

91.12. MOVEAFTERREAD CONSUMER 选项

除了 deleteAfterRead 外，还添加了另一个选项 moveAfterRead。启用此选项后，消耗的对象将移到目标 destinationBucket 中，而不是只被删除。这将需要指定 destinationBucket 选项。例如：

```
  from("minio://mycamelbucket?
minioClient=#minioClient&moveAfterRead=true&destinationBucketName=myothercamelbucket")
  .to("mock:result");
```

在这种情况下，消耗的对象将移到 myothercamelbucket bucket，并从原始存储桶中删除（因为 deleteAfterRead 设置为 true ）。

91.13. 使用 POJO 作为正文

由于多个选项，有时构建 Minio Request 可能会很复杂。我们介绍可能将 POJO 用作正文。在 Minio

中，您可以提交多个操作，作为 List 代理请求示例，您可以执行以下操作：

```
from("direct:minio")
    .setBody(ListObjectsArgs.builder()
                .bucket(bucketName)
                .recursive(getConfiguration().isRecursive())))
    .to("minio://test?minioClient=#minioClient&operation=listObjects&pojoRequest=true")
```

这样，您将直接传递请求，而无需专门传递与此操作相关的标头和选项。

91.14. 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.minio.access-key	Amazon AWS Secret Access Key 或 Minio 访问密钥。如果没有设置 camel，则 camel 将连接到服务以进行匿名访问。		字符串
camel.component.minio.auto-close-body	如果此选项为 true，并且 includeBody 为 true，则在交换完成时调用 MinioObject.close （）方法。此选项与 includeBody 选项密切相关。如果将 includeBody 设为 true，autocloseBody 设为 false，它将是关闭 MinioObject 流的调用者。将 autocloseBody 设置为 true，将自动关闭 MinioObject 流。	true	布尔值
camel.component.minio.auto-create-bucket	如果存储桶名称不存在，则设置存储桶的自动创建。	true	布尔值
camel.component.minio.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.minio.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

Name	描述	默认值	类型
camel.component.minio.bypass-governance-mode	如果要在删除特定对象时绕过GovernanceMode，请设置此标志。	false	布尔值
camel.component.minio.configuration	组件配置。选项是 org.apache.camel.component.minio.MinioConfiguration 类型。		MinioConfiguration
camel.component.minio.custom-http-client	为经过身份验证的访问设置自定义 HTTP 客户端。选项是 okhttp3.OkHttpClient 类型。		OkHttpClient
camel.component.minio.delete-after-read	在检索后，从 Minio 中删除对象。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 MinioConstants#BUCKET_NAME 和 MinioConstants#OBJECT_NAME 标头进行过滤，或者只过滤 MinioConstants#OBJECT_NAME 标头。	true	布尔值
camel.component.minio.delete-after-write	在 Minio 文件上传后删除文件对象。	false	布尔值
camel.component.minio.delimiter	ListObjectsRequest 中使用的分隔符仅消耗我们感兴趣的对象。		字符串
camel.component.minio.destination-bucket-name	源存储桶名称。		字符串
camel.component.minio.destination-object-name	源对象名称。		字符串
camel.component.minio.enabled	是否启用 minio 组件的自动配置。这默认是启用的。		布尔值
camel.component.minio.endpoint	端点可以是 URL、域名、IPv4 地址或 IPv6 地址。		字符串

Name	描述	默认值	类型
camel.component.minio.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，则标头将使用 Minio 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，autocloseBody 设为 false，它将是关闭 MinioObject 流的调用者。将 autocloseBody 设置为 true，将自动关闭 MinioObject 流。	true	布尔值
camel.component.minio.include-folders	ListObjectsRequest 中使用的标记，用于设置 include 文件夹。	false	布尔值
camel.component.minio.include-user-metadata	ListObjectsRequest 中使用的标记，用于获取用户元数据的对象。	false	布尔值
camel.component.minio.include-versions	ListObjectsRequest 中使用的标记，以获取带有版本控制的对象。	false	布尔值
camel.component.minio.key-name	通过端点参数在存储桶中设置元素的密钥名称。		字符串
camel.component.minio.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.minio.length	对象数据的字节数。		Long
camel.component.minio.match-e-tag	为 get 对象设置 match ETag 参数。		字符串
camel.component.minio.max-connections	在 minio 客户端配置中设置 maxConnections 参数。	60	整数
camel.component.minio.max-messages-per-poll	获取最大消息数，作为每次轮询轮询的限制。获取最大消息数，作为每次轮询轮询的限制。默认值为 10。使用 0 或负数设置为无限。	10	整数

Name	描述	默认值	类型
camel.component.minio.minio-client	对 registry 中的 Minio Client 对象的引用。选项是一个 io.minio.MinioClient 类型。		MinioClient
camel.component.minio.modified-since	为 get 对象设置修改后的参数。选项是一个 java.time.ZonedDateTime 类型。		ZonedDateTime
camel.component.minio.move-after-read	在检索后，将对象从 bucket 移到不同的存储桶。要完成操作，必须设置 destinationBucket 选项。仅当 Exchange 提交时，才会执行复制存储桶操作。如果进行回滚，则对象不会被移动。	false	布尔值
camel.component.minio.not-match-e-tag	为 get 对象设置与 ETag 参数不匹配。		字符串
camel.component.minio.object-lock	创建新存储桶时设置。	false	布尔值
camel.component.minio.object-name	要从具有给定对象名称的存储桶获取对象。		字符串
camel.component.minio.offset	开始对象数据的字节位置。		Long
camel.component.minio.operation	当用户不希望只进行上传时，要执行的操作。		MinioOperations
camel.component.minio.pojo-request	如果您想要将 POJO 请求用作正文。	false	布尔值
camel.component.minio.policy	此队列的策略在方法中设置。		字符串
camel.component.minio.prefix	对象名称以前缀开头。		字符串
camel.component.minio.proxy-port	TCP/IP 端口号。80 和 443 用作 HTTP 和 HTTPS 的默认值。		整数
camel.component.minio.recursive	比目录结构模拟递归列出。	false	布尔值

Name	描述	默认值	类型
camel.component.minio.region	Minio 客户端需要工作的区域。使用此参数时，配置将预期区域的小写名称（如 ap-east-1）。您需要使用名称 Region.EU_WEST_1.id （）。		字符串
camel.component.minio.secret-key	Amazon AWS 访问密钥 Id 或 Minio Secret Key。如果没有设置 camel，则 camel 将连接到服务以进行匿名访问。		字符串
camel.component.minio.secure	指定使用到 minio 服务的安全连接的标志。	false	布尔值
camel.component.minio.server-side-encryption	服务器端加密.选项是一个 io.minio.ServerSideEncryption 类型。		ServerSideEncryption
camel.component.minio.server-side-encryption-customer-key	复制/移动对象期间源对象的服务器端加密。选项是一个 io.minio.ServerSideEncryptionCustomerKey 类型。		ServerSideEncryptionCustomerKey
camel.component.minio.start-after	在此对象名称后，列出 bucket 中的对象。		字符串
camel.component.minio.storage-class	请求中设置的存储类。		字符串
camel.component.minio.un-modified-since	为 get 对象设置未修改后的参数。选项是一个 java.time.ZonedDateTime 类型。		ZonedDateTime
camel.component.minio.use-version1	为 true 时，使用 REST API 版本 1。	false	布尔值
camel.component.minio.version-id	在删除对象时设置对象的特定版本_ID。		字符串

第 92 章 MLLP

支持生成者和消费者

MLLP 组件专门设计来处理 MLLP 协议的细微性，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 组件提供简单的配置 URI，自动化的 HL7 确认生成和自动确认干预。

MLLP 协议通常不使用大量并发 TCP 连接 - 一个活跃的 TCP 连接是正常的。因此，MLLP 组件使用基于标准 Java 套接字的简单线程-per-connection 模型。这可使实施简单，并仅消除对 Camel 本身的依赖项。

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 生成者

MLLP 组件使用 byte[] 有效负载，并依赖于 Camel 类型转换将 byte[] 转换为其他类型。

92.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 mllp 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-mllp-starter</artifactId>
</dependency>
```

92.2. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

92.2.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

92.2.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

92.3. 组件选项

MLLP 组件支持 30 个选项，如下所列。

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
charsetName (common)	设定要使用的默认 charset。		字符串
configuration (common)	设置创建 MLLP 端点时要使用的默认配置。		MllpConfiguration
hl7Headers (common)	仅启用/禁用 HL7 Message MLLP Consumers 的自动生成消息标头。	true	布尔值
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。此设置控制是否需要最终 END_OF_DATA 字节或可选。	true	布尔值
stringPayload (common)	启用/禁用将有效负载转换为 String。如果启用，则从外部系统接收的 HL7 Payloads 将验证转换为字符串。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，则使用 MSH-18 的值来确定 th 合适的字符集。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 if enabled, HL7 Payloads if enabled, HL7 Payloads are received from external system isUtil.generateInvalidPayloadExceptionMessage 以了解验证的详情。如果检测到无效有效负载，则会抛出 MllpInvalidMessageException （用于消费者）或 MllpInvalidAcknowledgementException。	false	布尔值
acceptTimeout (consumer)	只等待 TCP 连接 TCP 服务器超时（以毫秒为单位）。	60000	int
backlog (consumer)	传入连接的最大队列长度（连接的请求）设置为 backlog 参数。如果连接指示在队列满时到达，连接将被拒绝。	5	整数
bindRetryInterval (consumer)	仅限 TCP 服务器 - 绑定尝试之间等待的毫秒数。	5000	int
bindTimeout (consumer)	仅限 TCP 服务器 - 重试绑定到服务器端口的毫秒数。	30000	int

Name	描述	默认值	类型
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的消息或类似信息时发生异常，现在将被作为消息进行处理，并由路由 Error Handler 处理。如果禁用，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理异常，方法是将它们记录到 WARN 或 ERROR 级别，并忽略。	true	布尔值
lenientBind (consumer)	TCP Server Only - 允许端点在 TCP ServerSocket 绑定前启动。在某些环境中，可能需要允许端点在 TCP ServerSocket 绑定之前启动。	false	布尔值
maxConcurrentConsumers (consumer)	允许的最大并发 MLLP Consumer 连接数。如果收到新连接并且已建立最大连接，则新连接将立即重置。	5	int
reuseAddress (consumer)	启用/禁用 SO_REUSEADDR 套接字选项。	false	布尔值
exchangePattern (consumer (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut	InOut	ExchangePattern
connectTimeout (producer)	仅为 TCP 连接 TCP 客户端建立超时（毫秒）。	30000	int
idleTimeoutStrategy (producer)	决定在发生闲置超时时要执行的操作。可能的值有：RESET: 将 SO_LINGER 设置为 0，并将套接字 CLOSE: 关闭套接字安全的默认值是 RESET。 Enum 值： • RESET • 关闭	RESET	MllpIdleTimeoutStrategy
keepalive (producer)	启用/禁用 SO_KEEPALIVE 套接字选项。	true	布尔值

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
tcpNoDelay (producer)	启用/禁用 TCP_NODELAY 套接字选项。	true	布尔值
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
defaultCharset (advanced)	设置用于字节到字符串转换的默认字符集。	ISO-8859-1	字符串
logPhi (advanced)	是否要记录 PHI。	true	布尔值
logPhiMaxBytes (advanced)	设置在日志条目中将登录的 PHI 的最大字节数。	5120	整数
readTimeout (advanced)	收到 MLLP 框架启动后使用的 SO_TIMEOUT 值（以毫秒为单位）。	5000	int
receiveBufferSize (advanced)	将 SO_RCVBUF 选项设置为指定的值（以字节为单位）。	8192	整数
receiveTimeout (advanced)	等待 MLLP 帧开始时使用的 SO_TIMEOUT 值（以毫秒为单位）。	15000	int
sendBufferSize (advanced)	将 SO_SNDBUF 选项设置为指定的值（以字节为单位）。	8192	整数
idletimeout (tcp)	在客户端 TCP 连接将重置前允许的大约空闲时间。null 值或小于或等于零的值将禁用闲置超时。		整数

92.4. 端点选项

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：

92.4.1. 路径参数(2 参数)

Name	描述	默认值	类型
hostname (common)	**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 (common)	TCP 连接所需的端口号。		int

92.4.2. 查询参数(26 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
charsetName (common)	设定要使用的默认 charset。		字符串
hl7Headers (common)	仅启用/禁用 HL7 Message MLLP Consumers 的自动生成消息标头。	true	布尔值
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。此设置控制是否需要最终 END_OF_DATA 字节或可选。	true	布尔值
stringPayload (common)	启用/禁用将有效负载转换为 String。如果启用，则从外部系统接收的 HL7 Payloads 将验证转换为字符串。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，则使用 MSH-18 的值来确定 th 合适的字符集。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 if enabled, HL7 Payloads if enabled, HL7 Payloads are received from external system isUtil.generateInvalidPayloadExceptionMessage 以了解验证的详情。如果检测到无效有效负载，则会抛出 MllpInvalidMessageException （用于消费者）或 MllpInvalidAcknowledgementException。	false	布尔值

Name	描述	默认值	类型
acceptTimeout (consumer)	只等待 TCP 连接 TCP 服务器超时（以毫秒为单位）。	60000	int
backlog (consumer)	传入连接的最大队列长度（连接的请求）设置为 backlog 参数。如果连接指示在队列满时到达，连接将被拒绝。	5	整数
bindRetryInterval (consumer)	仅限 TCP 服务器 - 绑定尝试之间等待的毫秒数。	5000	int
bindTimeout (consumer)	仅限 TCP 服务器 - 重试绑定到服务器端口的毫秒数。	30000	int
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的消息或类似信息时发生异常，现在将被作为消息进行处理，并由路由 Error Handler 处理。如果禁用，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理异常，方法是将它们记录到 WARN 或 ERROR 级别，并忽略。	true	布尔值
lenientBind (consumer)	TCP Server Only - 允许端点在 TCP ServerSocket 绑定前启动。在某些环境中，可能需要允许端点在 TCP ServerSocket 绑定之前启动。	false	布尔值
maxConcurrentConsumers (consumer)	允许的最大并发 MLLP Consumer 连接数。如果收到新连接并且已建立最大连接，则新连接将立即重置。	5	int
reuseAddress (consumer)	启用/禁用 SO_REUSEADDR 套接字选项。	false	布尔值
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut	InOut	ExchangePattern
connectTimeout (producer)	仅为 TCP 连接 TCP 客户端建立超时（毫秒）。	30000	int

Name	描述	默认值	类型
idleTimeoutStrategy (producer)	决定在发生闲置超时时要执行的操作。可能的值有：RESET: 将 SO_LINGER 设置为 0，并将套接字 CLOSE: 关闭套接字安全的默认值是 RESET。 Enum 值： • RESET • 关闭	RESET	MllpIdleTimeoutStrategy
keepalive (producer)	启用/禁用 SO_KEEPALIVE 套接字选项。	true	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
tcpNoDelay (producer)	启用/禁用 TCP_NODELAY 套接字选项。	true	布尔值
readTimeout (advanced)	收到 MLLP 框架启动后使用的 SO_TIMEOUT 值（以毫秒为单位）。	5000	int
receiveBufferSize (advanced)	将 SO_RCVBUF 选项设置为指定的值（以字节为单位）。	8192	整数
receiveTimeout (advanced)	等待 MLLP 帧开始时使用的 SO_TIMEOUT 值（以毫秒为单位）。	15000	int
sendBufferSize (advanced)	将 SO_SNDBUF 选项设置为指定的值（以字节为单位）。	8192	整数
idletimeout (tcp)	在客户端 TCP 连接将重置前允许的大约空闲时间。null 值或小于或等于零的值将禁用闲置超时。		整数

92.5. MLLP CONSUMER

MLLP Consumer 支持接收 MLLP-framed 消息并发送 HL7 Acknowledgements。MLLP Consumer 可以自动生成 HL7 Acknowledgement (HL7 Application Acknowledgements - AA, AE 和 AR)，或使用 CamelMllpAcknowledgement 交换属性来指定确认。另外，将通过设置 CamelMllpAcknowledgementType Exchange 属性来控制生成的确认类型。如果禁用了自动确认并且交换模式是 InOnly，则 MLLP Consumer 可以在不发送任何 HL7 Acknowledgement 的情况下读取消息。

92.5.1. 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。

92.5.2. Exchange Properties

MLLP Consumer 生成和 TCP 套接字状态的确认类型可以由 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 present 和 CamelMllpAcknowledgement 不存在，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，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，且 autoAck 为 true，则此属性可以用来指定 HL7 确认类型（如 AA、AE、AR）
CamelMllpAutoAcknowledge	布尔值	覆盖 autoAck 查询参数
CamelMllpCloseConnectionBeforeSend	布尔值	如果为 true，则套接字将在发送数据前关闭
CamelMllpResetConnectionBeforeSend	布尔值	如果为 true，则会在发送数据前重置套接字
CamelMllpCloseConnectionAfterSend	布尔值	如果为 true，则发送数据后套接字将立即关闭
CamelMllpResetConnectionAfterSend	布尔值	如果为 true，则发送任何数据后将立即重置套接字

92.6. MLLP PRODUCER

MLLP Producer 支持发送 MLLP-framed 消息并接收 HL7 Acknowledgements。MLLP Producer 交错了 HL7 Acknowledgments，并在收到负确认时引发异常。接收的确认被干扰，并在负确认时引发异常。当配置了 InOnly Exchange 模式时，MLLP Producer 可以忽略确认。

92.6.1. 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	接收 HL7 Acknowledgment byte[]
CamelMllpAcknowledgementString	收到 HL7 Acknowledgment，转换为字符串

92.6.2. Exchange Properties

TCP 套接字的状态可由 Camel 交换中的这些属性控制：

键	类型	描述
CamelMllpCloseConnectionBeforeSend	布尔值	如果为 true，则套接字将在发送数据前关闭
CamelMllpResetConnectionBeforeSend	布尔值	如果为 true，则会在发送数据前重置套接字
CamelMllpCloseConnectionAfterSend	布尔值	如果为 true，则发送数据后套接字将立即关闭
CamelMllpResetConnectionAfterSend	布尔值	如果为 true，则发送任何数据后将立即重置套接字

92.7. 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.accept-timeout	只等待 TCP 连接 TCP 服务器超时（以毫秒为单位）。	60000	整数
camel.component.mllp.auto-ack	仅启用/禁用 MLLP Acknowledgement MLLP Consumers 的自动生成。	true	布尔值

Name	描述	默认值	类型
camel.component.mllp.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.mllp.backlog	传入连接的最大队列长度（连接的请求）设置为 backlog 参数。如果连接指示在队列满时到达，连接将被拒绝。	5	整数
camel.component.mllp.bind-retry-interval	仅限 TCP 服务器 - 绑定尝试之间等待的毫秒数。	5000	整数
camel.component.mllp.bind-timeout	仅限 TCP 服务器 - 重试绑定到服务器端口的毫秒数。	30000	整数
camel.component.mllp.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的消息或类似信息时发生异常，现在将被作为消息进行处理，并由路由 Error Handler 处理。如果禁用，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理异常，方法是将它们记录到 WARN 或 ERROR 级别，并忽略。	true	布尔值
camel.component.mllp.charset-name	设定要使用的默认 charset。		字符串
camel.component.mllp.configuration	设置创建 MLLP 端点时要使用的默认配置。选项是 org.apache.camel.component.mllp.MllpConfiguration 类型。		MllpConfiguration
camel.component.mllp.connect-timeout	仅为 TCP 连接 TCP 客户端建立超时（毫秒）。	30000	整数
camel.component.mllp.default-charset	设置用于字节到字符串转换的默认字符集。	ISO-8859-1	字符串
camel.component.mllp.enabled	是否启用 mllp 组件的自动配置。这默认是启用的。		布尔值
camel.component.mllp.exchange-pattern	在消费者创建交换时设置交换模式。		ExchangePattern

Name	描述	默认值	类型
camel.component.mllp.hl7-headers	仅启用/禁用 HL7 Message MLLP Consumers 的自动生成消息标头。	true	布尔值
camel.component.mllp.idle-timeout	在客户端 TCP 连接将重置前允许的大约空闲时间。null 值或小于或等于零的值将禁用闲置超时。		整数
camel.component.mllp.idle-timeout-strategy	决定在发生闲置超时时要执行的操作。可能的值有：RESET: 将 SO_LINGER 设置为 0，并将套接字 CLOSE: 关闭套接字安全的默认值是 RESET。		MllpIdleTimeoutStrategy
camel.component.mllp.keep-alive	启用/禁用 SO_KEEPALIVE 套接字选项。	true	布尔值
camel.component.mllp.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.mllp.lenient-bind	TCP Server Only - 允许端点在 TCP ServerSocket 绑定前启动。在某些环境中，可能需要允许端点在 TCP ServerSocket 绑定之前启动。	false	布尔值
camel.component.mllp.log-phi	是否要记录 PHI。	true	布尔值
camel.component.mllp.log-phi-max-bytes	设置在日志条目中将登录的 PHI 的最大字节数。	5120	整数
camel.component.mllp.max-concurrent-consumers	允许的最大并发 MLLP Consumer 连接数。如果收到新连接并且已建立最大连接，则新连接将立即重置。	5	整数
camel.component.mllp.read-timeout	收到 MLLP 框架启动后使用的 SO_TIMEOUT 值（以毫秒为单位）。	5000	整数
camel.component.mllp.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.receive-timeout	等待 MLLP 帧开始时使用的 SO_TIMEOUT 值（以毫秒为单位）。	15000	整数

Name	描述	默认值	类型
camel.component.mllp.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。此设置控制是否需要最终 END_OF_DATA 字节或可选。	true	布尔值
camel.component.mllp.reuse-address	启用/禁用 SO_REUSEADDR 套接字选项。	false	布尔值
camel.component.mllp.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.string-payload	启用/禁用将有效负载转换为 String。如果启用，则从外部系统接收的 HL7 Payloads 将验证转换为字符串。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，则使用 MSH-18 的值来确定 th 合适的字符集。如果没有设置 MSH-18，则将使用默认 ISO-8859-1 字符集。	true	布尔值
camel.component.mllp.tcp-no-delay	启用/禁用 TCP_NODELAY 套接字选项。	true	布尔值
camel.component.mllp.validate-payload	启用/禁用 HL7 Payloads if enabled, HL7 Payloads if enabled, HL7 Payloads are received from external system isUtil.generateInvalidPayloadExceptionMessage 以了解验证的详情。如果检测到无效有效负载，则会抛出 MllpInvalidMessageException （用于消费者）或 MllpInvalidAcknowledgementException。	false	布尔值

第 93 章 MOCK

仅支持生成者

测试分布式和异步处理并不困难。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。

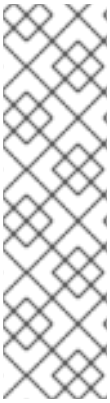
Mock 组件提供了一个强大的声明测试机制，它与 jMock 类似，它允许在测试开始前在任何 Mock 端点上创建声明性预期。然后，运行测试，这通常会将消息触发给一个或多个端点，最后，在测试情况下的预期预期以确保系统按预期工作。

这可让您测试各种内容，如下所示：

- 每个端点上都会收到正确的消息数，
- 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 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，或者消息与某些 predicate 匹配，如评估 XPath 或 XQuery Expression。

注意

另外，Test 端点 是 Mock 端点，但它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，它是一个 Mock 端点，它会自动从文件或 数据库中的一些示例消息设置其断言，例如：



注意

模拟端点会无限期地保留收到的交换。请记住，Mock 专为测试而设计。将 Mock 端点添加到路由时，发送到端点的每个 Exchange 都会在内存中存储（以允许后续验证），直到显式重置或 JVM 重启为止。如果您要发送大量卷和/或大消息，这可能会导致过量内存使用。如果您的目标是测试内联路由，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。有两个新选项 retainFirst 和 retainLast，可用于限制 Mock 端点在内存中保留的消息数量。

93.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 mock 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-mock-starter</artifactId>
</dependency>
```

93.2. URI 格式

```
mock:someName[?options]
```

其中 someName 可以是唯一标识端点的任何字符串。

93.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

93.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

93.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

93.4. 组件选项

Mock 组件支持 4 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
log (producer)	在模拟收到传入消息时打开日志记录。这只会记录传入消息的 INFO 级别一次。如需更详细的日志记录，然后将 org.apache.camel.component.mock.MockEndpoint 类的 logger 设置为 DEBUG 级别。	false	布尔值

Name	描述	默认值	类型
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
exchangeFormatter (advanced)	**Autowired** 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，我们默认为 DefaultExchangeFormatter。		ExchangeFormatter

93.5. 端点选项

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：

93.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
name (producer)	**必需的** 模拟端点的名称。		字符串

93.5.2. 查询参数(12 参数)

Name	描述	默认值	类型
assertPeriod (producer)	设置一个宽限期，在此期内，模拟端点将重新开始处理，以确保初始断言仍然有效。这用于 assert，用于准确有多个消息到达的信息。例如，如果 expectedMessageCount (int)被设置为 5，则在 5 或更多消息到达时满足断言。为确保正好 5 个消息到达，您需要稍等片刻，以确保没有进一步的消息到达。这是您可以使用此方法的方法。默认情况下禁用此周期。		long

Name	描述	默认值	类型
expectedCount (producer)	指定此端点应接收的预期消息交换数量。注意：如果您希望 0 个信息，然后进行额外操作，如测试启动时为 0 匹配，因此您需要设置一个 assert 周期，以便让测试运行一段时间，以确保还没有到达消息；对于使用 setAssertPeriod (long)。另一种方法是使用 NotifyBuilder，并在对模拟调用 assertIsSatisfied （）方法前，使用 notifier 知道 Camel 何时完成一些消息。这可让您不要使用固定的 assert 周期来加快测试时间。如果您要断言正好 n 个消息到达这个 mock 端点，那么还会看到 setAssertPeriod (long)方法了解更多详情。	-1	int
failfast (producer)	设置 assertIsSatisfied （）应该在第一次检测到的失败预期失败时快速失败，否则可能会等待所有预期消息在执行预期验证前到达。默认为 true。设置为 false 以使用 Camel 2.x 中的行为。	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
log (producer)	在模拟收到传入消息时打开日志记录。这只会记录传入消息的 INFO 级别一次。如需更详细的日志记录，然后将 org.apache.camel.component.mock.MockEndpoint 类的 logger 设置为 DEBUG 级别。	false	布尔值
reportGroup (producer)	用于根据大小组打开吞吐量日志的数字。		int
resultMinimumWaitTime (producer)	设置预期时间（在 millis 中），assertIsSatisfied （）将等待一个白板，直到其满足为止。		long
resultWaitTime (producer)	设置 assertIsSatisfied （）将等待的最长时间（在 millis 中），直到其满足为止。		long

Name	描述	默认值	类型
retainFirst (producer)	指定只保留第 n 个接收的交换数。这在使用大数据测试时使用，通过不存储每个交换端点接收的副本来减少内存消耗。重要：使用这个限制时，getReceivedCounter （）仍会返回接收的交换的实际数量。例如，如果我们收到 5000 Exchange，并且已配置为只保留前 10 个交换，则 getReceivedCounter （）仍会返回 5000，但 getExchanges （）和 getReceivedExchanges （）方法中只有前 10 个交换。使用此方法时，不支持一些其他预期方法，例如 expectedBodiesReceived (Object...)在收到第一个数量的正文上设置预期。您可以配置 setRetainFirst (int)和 setRetainLast (int)方法，以限制第一个和最后一个接收的限制。	-1	int
retainLast (producer)	指定只保留最后 n 个接收的交换数。这在使用大数据测试时使用，通过不存储每个交换端点接收的副本来减少内存消耗。重要：使用这个限制时，getReceivedCounter （）仍会返回接收的交换的实际数量。例如，如果我们收到 5000 Exchange，并且已配置为只保留最后 20 个交换，则 getReceivedCounter （）仍会返回 5000，但 getExchanges （）和 getReceivedExchanges （）方法中只有最后 20 个交换。使用此方法时，不支持一些其他预期方法，例如 expectedBodiesReceived (Object...)在收到第一个数量的正文上设置预期。您可以配置 setRetainFirst (int)和 setRetainLast (int)方法，以限制第一个和最后一个接收的限制。	-1	int
sleepForEmptyTest (producer)	当 expectedMessageCount (int)被调用为零时，允许指定 sleep 来检查此端点是否确实为空。		long
copyOnExchange (producer (advanced))	设置在这个模拟端点收到时是否制作传入 Exchange 的深度副本。默认为 true。	true	布尔值

93.6. 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，我们设置一个预期，然后在测试运行后，我们假设我们的预期已满足：

```
MockEndpoint resultEndpoint = context.getEndpoint("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();
```

您通常始终调用方法来测试在运行测试后是否满足预期。

当调用 assertIsSatisfied （） 时，Camel 默认等待 10 秒。这可以通过设置 setResultWaitTime (millis) 方法来配置。

93.7. 使用 ASSERTPERIOD

满足断言时，Camel 将停止等待并继续进行 assertIsSatisfied 方法。这意味着，如果新消息到达模拟端点，以后只需点，则 arrival 不会影响断言的结果。假设您想要在一段时间后测试新消息是否到达，那么您可以通过设置 setAssertPeriod 方法来完成此操作，例如：

```
MockEndpoint resultEndpoint = context.getEndpoint("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();
```

93.8. 设置预期

您可以从 MockEndpoint 的 Javadoc 中看到，可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数。
expectedMinimumMessageCount(int)	定义端点上预期消息的最小数量。
expectedBodiesReceived(...)	定义应接收的预期正文（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的消息，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的消息，请使用给定的 Expression 来比较消息。

æ–¹æ³•	描述
expectsNoDuplicates(Expression)	要添加没有收到重复消息的预期；使用 Expression 来计算每个消息的唯一标识符。这可能与 JMS 类似，或者消息中的一些唯一引用号。

下面是另一个示例：

```
resultEndpoint.expectedBodiesReceived("firstMessageBody", "secondMessageBody",
"thirdMessageBody");
```

93.9. 为特定消息添加预期

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（使用基于零的索引，如 java.util.List），您可以使用以下代码：

```
resultEndpoint.message(0).header("foo").isEqualTo("bar");
```

在 camel-core 处理器测试 中使用了 Mock 端点的一些示例。

93.10. 模拟现有端点

Camel 现在允许您在 Camel 路由中自动模拟现有端点。

注意

它如何工作
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和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").routeId("start")
                    .to("direct:foo").to("log:foo").to("mock:result");

            from("direct:foo").routeId("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：

建议 模拟所有端点

```
@Test
public void testAdvisedMockEndpoints() throws Exception {
    // advice the start route using the inlined AdviceWith lambda style route builder
    // which has extended capabilities than the regular route builder
    AdviceWith.adviceWith(context, "start", a ->
    // mock all endpoints
    a.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。在 INFO 级别 Camel 日志被模拟：

```
INFO  Adviced endpoint [direct://foo] with mock endpoint [mock:direct:foo]
```

注意

模拟端点没有参数
端点，这些端点将被剥离其参数。例如，端点 log:foo?showAll=true 将模拟到以下端点 mock:log:foo。请注意，参数已被删除。

它还可以使用模式来模拟特定的端点。例如，要模拟您执行 的所有日志 端点，如下所示：

建议 仅使用模式模拟日志端点

```
@Test
public void testAdvisedMockEndpointsWithPattern() throws Exception {
    // advice the start route using the inlined AdviceWith lambda style route builder
    // which has extended capabilities than the regular route builder
    AdviceWith.adviceWith(context, "start", a ->
    // mock only log endpoints
    a.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 Intercept 与 Camel 使用的相同匹配函数的更多详细信息。

注意

请记住，模拟端点会导致在消息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。

93.11. 使用 CAMEL-TEST 组件模拟现有端点

在使用 camel-test Test Kit 时，您可以轻松地启用此行为，而不是使用 recommendations With instruct Camel to mock 端点。

同一路由可以测试如下：请注意，我们从 isMockEndpoints 方法返回 "*"，它告知 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"));
            }
        };
    }
}
```

93.12. 使用 XML DSL 模拟现有端点

如果您没有将 camel-test 组件用于单元测试（如上所示），您可以在使用 XML 文件进行路由时使用不同的方法。
解决方法是创建一个由单元测试使用的新 XML 文件，然后包括具有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

```
<!-- 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，我们按如下方式创建新的 XML 文件，其中包括 camel-route.xml 文件，并定义一个带有类 org.apache.camel.impl.InterceptSendToMockEndpointStrategy 的 spring bean，该文件告知 Camel 模拟所有端点：

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>
```

93.13. 模拟端点并跳过发送到原始端点

有时，您想要轻松模拟并跳过发送到特定端点。因此，消息会被停用并仅发送到模拟端点。现在，您可以使用 AdviceWith 使用 mockEndpointsAndSkip 方法。以下示例将跳过发送到两个端点 "direct:foo" 和 "direct:bar" 的发送。

recommendationsWith mock，并跳过发送到端点的发送

```
@Test
public void testAdvisedMockEndpointsWithSkip() throws Exception {
    // advice the first route using the inlined AdviceWith route builder
    // which has extended capabilities than the regular route builder
```

```
        AdviceWith.adviceWith(context.getRouteDefinitions().get(0), 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 相同的示例

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");
            }
        };
    }
}
```

93.14. 限制要保留的消息数量

Mock 端点默认保留它收到的每个交换的副本。因此，如果您测试大量消息，它将消耗内存。我们引入了两个选项 retainFirst 和 retainLast，它们可以用来指定只保留第 N 个和/或最后一个交换。

例如，在下面的代码中，我们只想保留第一个 5 个，最后 5 个交换收到的 5 个交换。

```
MockEndpoint mock = getMockEndpoint("mock:data");
mock.setRetainFirst(5);
mock.setRetainLast(5);
mock.expectedMessageCount(2000);

mock.assertIsSatisfied();
```

使用这有一些限制。MockEndpoint 上的 getExchanges （） 和 getReceivedExchanges （） 方法将仅返回 Exchange 的保留副本。因此，在上面的示例中，列表将包含 10 个交换、前五个和最后 5 个。retainFirst 和 retainLast 选项还对预期可以使用的方法有限制。例如，在消息正文、标头等上工作 的预期XXX 方法将仅在保留的消息上运行。在上例中，它们只能测试 10 个保留的消息的预期。

93.15. 使用 ARRIVAL 时间进行测试

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);
```

您可以使用此信息知道消息何时到达模拟。但它还提供了基础，以便了解上一个和下一个消息到达模拟之间的时间间隔。您可以使用此选项在 Mock 端点上设置使用 arrives DSL 的预期。

例如，假设第一条消息应该在下一个下一个消息前达到 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（基于索引），在上一个消息后不应达到 0-2 秒：

```
mock.message(1).arrives().noLaterThan(2).seconds().afterPrevious();
```

您还可以在 间使用 来设置较低的绑定。例如，假设它应该在 1-4 秒之间：

```
mock.message(1).arrives().between(1, 4).seconds().afterPrevious();
```

您还可以设置所有消息的预期，例如：它们之间的差距应该最多为 1 秒：

```
mock.allMessages().arrives().noLaterThan(1).seconds().beforeNext();
```

注意

上面示例中的 时间单元
使用 秒 作为时间单位，但 Camel 还提供 毫秒 和 分钟。

93.16. 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.mock.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.mock.enabled	是否启用模拟组件的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.component.mock.exchange-formatter	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，我们默认为 DefaultExchangeFormatter。选项是 org.apache.camel.spi.ExchangeFormatter 类型。		ExchangeFormatter
camel.component.mock.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.mock.log	在模拟收到传入消息时打开日志记录。这只会记录传入消息的 INFO 级别一次。如需更详细的日志记录，然后将 org.apache.camel.component.mock.MockEndpoint 类的 logger 设置为 DEBUG 级别。	false	布尔值

第 94 章 MONGODB

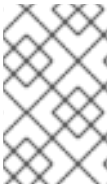
支持生成者和消费者

根据 Wikipedia: "NoSQL 是一个移动，它提升了由松散定义的非关系数据存储类，其与关系数据库的长历史记录和 ACID 保证中断"。过去数年来，NoSQL 解决方案一直流行，极端使用的站点和服务（如 Facebook、LinkedIn、Twitter 等）被广泛使用，以达到可扩展性和灵活性。

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 ACID 等事务处理或关系数据。相反，它们围绕灵活的数据结构和架构概念设计（例如，带有固定模式的数据库表的传统概念已被丢弃），在商业硬件上具有极高的可扩展性以及极快的处理。

MongoDB 是一个非常流行的 NoSQL 解决方案，camel-mongodb 组件将 Camel 与 MongoDB 集成，允许您将 MongoDB 集合作为生成者（集合的性能操作）和消费者（消耗来自 MongoDB 集合的文档）进行交互。

MongoDB 会围绕文档的概念（不是办公室文档，而是在 JSON/BSON 中定义的分层数据）和集合进行开发。此组件页面将假设您熟悉它们。否则，请访问 http://www.mongodb.org/。



注意

MongoDB Camel 组件使用 Mongo Java Driver 4.x。

94.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 mongodb 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-mongodb-starter</artifactId>
</dependency>
```

94.2. URI 格式

```
mongodb:connectionBean?
database=databaseName&collection=collectionName&operation=operationName[&moreOptions...]
```

94.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

94.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

94.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

94.4. 组件选项

***MongoDB** 组件支持 4 个选项，如下所列。*

Name	描述	默认值	类型
mongoConnection (common)	用于连接的 **Autowired** 共享客户端。组件生成的所有端点都将共享此连接客户端。		MongoClient
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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

94.5. 端点选项

***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：

94.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
connectionBean (common)	**必需** 设置 connection bean 引用，用于查找用于连接数据库的客户端。		字符串

94.5.2. 查询参数(27 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)	在初始时创建集合（如果不存在）。默认为 true。	true	布尔值
数据库 (common)	将 MongoDB 数据库的名称设置为 target。		字符串
hosts (common)	mongodb 服务器的主机地址，格式为 host:port。也可以使用多个地址，作为逗号分隔的主机列表：host1:port1,host2:port2。如果指定了 hosts 参数，则提供的 connectionBean 将被忽略。		字符串
mongoConnection (common)	设置连接 bean，用作连接数据库的客户端。		MongoClient
operation (common)	设置此端点将针对 MongoDB 执行的操作。 Enum 值： • findById • findOneByQuery • findAll • findDistinct • insert • save • update • remove • bulkWrite • 聚合 • getDbStats • getColStats • æ•°é‡ • 命令		MongoDbOperation

Name	描述	默认值	类型
outputType (common)	将制作者的输出转换为所选类型：DocumentList Document 或 MongoIterable。DocumentList 或 MongoIterable 适用于 findAll 和 aggregate。文档适用于所有其他操作。 Enum 值： • DocumentList • 文档 • MongoIterable		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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		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
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，则在一段时间后，MongoDB 服务器将自动释放和关闭光标。如果需要，客户端应该重新生成光标。这个值指定尝试获取新光标前等待的时间，如果尝试失败，则尝试失败，下一次尝试进行前的时长。默认值为 1000ms。	1000	long
dynamicity (advanced)	设置此端点是否将尝试从传入的 Exchange 属性动态解析目标数据库和集合。可用于在运行时覆盖其他静态端点 URI 中指定的数据库和集合。它默认是禁用的，以提高性能。启用它将占用最小的性能命中。	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。 Enum 值： • PRIMARY • PRIMARY_PREFERRED • SECONDARY • SECONDARY_PREFERRED • 接近	PRIMARY	字符串
writeConcern (advanced)	使用从 MongoDB 向独立 mongod、replicaset 或 cluster 请求的确认级别配置连接 bean。可能的值有 ACKNOWLEDGED, W1, W2, W3, UNACKNOWLEDGED, JOURNALED 或 MAJORITY。 Enum 值： • 已确认 • W1 • W2 • W3 • 未确认 • JOURNALED • 多数	已确认	字符串

Name	描述	默认值	类型
writeResultAsHeader (advanced)	在写入操作中，它决定是否返回 WriteResult 作为 OUT 消息的正文，我们将 IN 消息传送到 OUT，并将 WriteResult 附加为标头。	false	布尔值
streamFilter (changeStream)	用于更改流消费者的过滤条件。		字符串
password (security)	mongodb 连接的用户密码。		字符串
用户名 （安全性）	mongodb 连接的用户名。		字符串
persistentId (tail)	一个尾部跟踪集合可以为多个可尾部消费者托管许多跟踪器。为了保持它们独立，每个跟踪器都应该拥有自己的唯一的 persistentId。		字符串
persistentTailTracking (tail)	启用持久的尾部跟踪，这是一种在系统重启后跟踪最后一次消耗的消息的机制。下一次系统启动时，端点将从最后停止的滑动记录点中恢复。	false	布尔值
tailTrackCollection (tail)	会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)	指明尾部跟踪机制将保留的数据库。如果未指定，则默认选择当前数据库。即使启用，也不会考虑动态性，即尾部跟踪数据库不会因过去的端点初始而不同。		字符串
tailTrackField (tail)	放置最后一个跟踪值的字段。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会提高性质，并将在每次生成尾部时都用于定位尾部光标。使用 type: tailTrackIncreasingField 的查询创建光标将被(re)创建，它大于 lastValue （可能从持久尾部跟踪中恢复）。可以是 Integer, Date, String 等类型。注意：目前不支持点表示法，因此该字段应位于文档的顶部。		字符串

94.6. 在 SPRING XML 中配置数据库

以下 Spring XML 创建一个 bean，定义与 MongoDB 实例的连接。

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>
```

94.7. 路由示例

Spring XML 中定义的以下路由在集合上执行操作 getDbStats。

获取指定集合的 DB 统计

```
<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>
```

94.8. MONGODB OPERATIONS - PRODUCER 端点

94.8.1. 查询操作

94.8.1.1. findById

此操作仅从与 IN 消息正文的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 类型的任何对象。请参阅 http://bsonspec.org/spec.html 和 http://www.mongodb.org/display/DOCS/Java+Types。

```
from("direct:findById")
    .to("mongodb:myDb?database=flights&collection=tickets&operation=findById")
    .to("mock:resultFindById");
```

请注意，默认的 _id 被 Mongo 视为 和 ObjectId 类型，因此您可能需要正确转换它。

```
from("direct:findById")
    .convertBodyTo(ObjectId.class)
    .to("mongodb:myDb?database=flights&collection=tickets&operation=findById")
    .to("mock:resultFindById");
```

注意

支持可选参数
此操作支持投射操作器。请参阅 指定字段过滤器（项目 ）。

94.8.1.2. findOneByQuery

从与 MongoDB 查询选择器匹配的集合中检索第一个元素。如果设置了 CamelMongoDbCriteria 标头，则其值将用作查询选择器。如果 CamelMongoDbCriteria 标头为 null，则 IN 消息正文将用作查询选择器。在这两种情况下，查询选择器都应是 Bson 类型，或者转换为 Bson （例如，JSON 字符串或 HashMap）。如需更多信息，请参阅类型转换。

使用 MongoDB Driver 提供的 过滤器 创建查询选择器。

94.8.1.3. 没有查询选择器的示例（返回集合中的第一个文档）

```
from("direct:findOneByQuery")
    .to("mongodb:myDb?database=flights&collection=tickets&operation=findOneByQuery")
    .to("mock:resultFindOneByQuery");
```

94.8.1.4. 使用查询选择器的示例（返回集合中的第一个匹配文档）：

```
from("direct:findOneByQuery")
    .setHeader(MongoDbConstants.CRITERIA, constant(Filters.eq("name", "Raul Kripalani")))
    .to("mongodb:myDb?database=flights&collection=tickets&operation=findOneByQuery")
    .to("mock:resultFindOneByQuery");
```

注意

支持可选参数
此操作支持投射运算符和排序条款。请参阅 指定字段过滤器（项目 ），指定 sort 子句。

94.8.1.5. findAll

findAll 操作会返回与查询匹配的所有文档，或者根本没有返回集合中包含的所有文档。查询对象提取 CamelMongoDbCriteria 标头。如果 CamelMongoDbCriteria 标头为 null，则查询对象被提取消息正文，即 Bson 类型或可转换为 Bson。它可以是 JSON 字符串或 Hashmap。如需更多信息，请参阅类型转换。

94.8.1.5.1. 没有查询选择器的示例（返回集合中的所有文档）

```
from("direct:findAll")
   .to("mongodb:myDb?database=flights&collection=tickets&operation=findAll")
   .to("mock:resultFindAll");
```

94.8.1.5.2. 带有查询选择器的示例（返回集合中的所有匹配文档）

```
from("direct:findAll")
   .setHeader(MongoDbConstants.CRITERIA, Filters.eq("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高效的检索：

标头键	快速常量	描述（从 MongoDB API 文档中提取）	预期类型
CamelMongoDbNumToSkip	**MongoDbConstants.NUM_TO_SKIP**	在光标的开头丢弃给定数量的元素。	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。光标通常获取结果对象的批处理，并将它们存储在本地。如果 batchSize 为正数，它代表检索的每个批处理对象的大小。可以调整它以优化性能并限制数据传输。如果 batchSize 为负数，它将限制返回的数量对象，该对象位于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，那么服务器将返回最多 10 个文档，以及 4MB 中可以容纳的内容，然后关闭光标。请注意，这个功能与文档中的 limit （）不同，该文档中的大小必须符合最大大小，而且无需发送请求以关闭光标服务器端。即使光标迭代后也可以更改批处理大小，在这种情况下，设置将应用于下一个批处理检索。	int/Integer

标头键	快速常量	描述（从 MongoDB API 文档中提取）	预期类型
CamelMongoDbAllowDiskUse	**MongoDbConstants.ALLOW_DISK_USE**	设置 allowDiskUse MongoDB 标志。从 MongoDB Server 4.3.1 开始，这被支持。将这个标头与旧的 MongoDB Server 版本搭配使用可能会导致查询失败。	布尔值/布尔值

94.8.1.5.3. 选项 outputType=MongoIterable 和 batch size 的示例

```
from("direct:findAll")
   .setHeader(MongoDbConstants.BATCH_SIZE).constant(10)
   .setHeader(MongoDbConstants.CRITERIA, constant(Filters.eq("name", "Raul Kripalani")))
   .to("mongodb:myDb?
database=flights&collection=tickets&operation=findAll&outputType=MongoIterable")
   .to("mock:resultFindAll");
```

findAll 操作还会返回以下 OUT 标头，以便在使用分页时迭代结果页面：

标头键	快速常量	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 limit/skip。	int/Integer
CamelMongoDbResultPageSize	**MongoDbConstants.RESULT_PAGE_SIZE**	与查询匹配的对象数量。这不会考虑 limit/skip。	int/Integer

注意

支持可选参数
此操作支持投射运算符和排序条款。请参阅 指定字段过滤器（项目），指定 sort 子句。

94.8.1.6. 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" 集合中记录的数量。注意如何启用动态性，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);
```

您可以提供 查询对象提取 CamelMongoDbCriteria 标头。如果 CamelMongoDbCriteria 标头为 null，则查询对象被提取的消息正文，即 Bson 或 convertable to Bson。

```
Document query = ...
Long count = template.requestBodyAndHeader("direct:count", query,
MongoDbConstants.COLLECTION, "dynamicCollectionName");
```

94.8.1.7. 指定字段过滤器（项目）

默认情况下，查询操作将在其整个中返回匹配的对象（及其所有字段）。如果您的文档非常大，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，只需通过设置相关的 Bson （或 typeible to Bson，如 JSON String、Map 等）在 CamelMongoDbFieldsProjection 标头中指定字段过滤器。

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("mongodb: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("mongodb: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);
```

94.8.1.8. 指定 sort 子句

通常，根据特定字段排序从集合中获取 min/max 记录的要求，该字段使用 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("mongodb: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 标头可用于 findOneByQuery 操作，以实现相同的结果。如果还指定了 FIELDS_PROJECTION 标头，则操作将返回单个字段/值对，可直接传递给另一个组件（例如，参数 ized MyBatis SELECT 查询）。本例演示了如何根据 documentTimestamp 字段从集合中获取临时最新的文档，并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("mongodb:myDb?
database=local&collection=myDemoCollection&operation=findOneByQuery")
.to("direct:aMyBatisParameterizedSelect");
```

94.8.2. 创建/更新操作

94.8.2.1. insert

将新对象插入到 MongoDB 集合中，从 IN 消息正文获取。试图将类型转换转换为 文档或 列表。支持两种模式：单一插入和多个插入。对于多个插入，端点将预期任意类型的对象列表、数组或集合，只要它们是 - 或可以转换为 - 文档。Example:

```
from("direct:insert")
   .to("mongodb:myDb?database=flights&collection=tickets&operation=insert");
```

该操作将返回 WriteResult，具体取决于 WriteConcern 或 invokeGetLastError 选项的值，getLastError （） 将被调用。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 （） and/or 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值。但是，如果您确实指定了它，则插入操作将失败（对于 Camel 需要注意，您需要启用 invokeGetLastError 或设置等待写结果的 WriteConcern）。

这并不是组件的限制，而是在 MongoDB 中的工作方式以获得更高的吞吐量。如果您使用自定义 _id，您预期在应用程序级别确保是唯一的（这也是很好的做法）。

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>（如果插入多个记录）。

在 MongoDB Java Driver 3.x 中，insertOne 和 insertMany 操作返回 void。Camel 插入操作会返回插入的 Documents 或 Documents 列表。请注意，如果需要，每个文档都由一个新的 OID 更新。

94.8.2.2. save

save 操作等同于 upsert (UPdate，inSERT)操作，其中将更新记录，如果不存在，它将被插入一个原子操作。MongoDB 将根据 _id 字段执行匹配。

请注意，如果更新，对象会被完全被替换，而不允许使用 MongoDB 的 $modifier。因此，如果要操作对象已存在，则有两个选项：

1. 执行查询以首先检索整个对象及其所有字段（或者效率不高），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，并且创建新的 _id 将放在 CamelMongoOid 标头中。

例如：

```
from("direct:insert")
   .to("mongodb:myDb?database=flights&collection=tickets&operation=save");
```

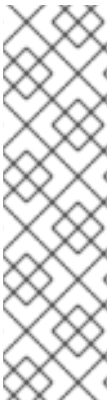
```
// route: from("direct:insert").to("mongodb:myDb?
database=flights&collection=tickets&operation=save");
org.bson.Document docForSave = new org.bson.Document();
```

```
docForSave.put("key", "value");
Object result = template.requestBody("direct:insert", docForSave);
```

94.8.2.3. update

更新集合上的一个或多个记录。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，并在 Body 中将更新规则定义为 Bson。



注意

在功能改进
后，使用 MongoDBConstants.CRITERIA 标头定义为 Bson 在进行更新前查询 mongodb，但您应该注意，在聚合中使用增强模式并应用 mongodb 更新，则需要在聚合过程中将其从生成的 camel Exchange 中删除，然后应用 mongodb 更新。如果您在聚合和/或重新定义 MongoDBConstants.CRITERIA 标头前没有删除这个标头，在将 camel Exchange 发送到 mongodb producer 端点前，您可能会在更新 mongodb 时最终使用无效的 camel Exchange payload。

第二种方法 Require a List<Bson> 作为 IN message body contains exactly 2 项：

- 元素 1 (index 0) jpeg 过滤器查询 TOKEN 确定受影响的对象，与典型的查询对象相同
- 元素 2 (index 1) TOKEN 更新规则 jpeg 如何更新匹配的对象。支持 MongoDB 中的所有 修饰符操作。

注意

Multiupdates
默认情况下，MongoDB 只会更新 1 个对象，即使多个对象与过滤器查询匹配。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 (MongoDbConstants.RECORDS_AFFECTED constant)，其记录数量已更新（从 WriteResult.getN（））。

支持以下 IN 消息标头：

标头键	快速常量	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所有匹配的对象。请参阅 http://www.mongodb.org/display/DOCS/Atomic+Operations	布尔值/布尔值
CamelMongoDbUpsert	MongoDbConstants.UPSERT	如果数据库应该创建元素（如果不存在）	布尔值/布尔值

例如，以下命令将 "scientist" 字段的值设置为 "Darwin" 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");
List<Bson> body = new ArrayList<>();
Bson filterField = Filters.eq("filterField", true);
body.add(filterField);
BsonDocument updateObj = new BsonDocument().append("$set", new
BsonDocument("scientist", new BsonString("Darwin")));
body.add(updateObj);
Object result = template.requestBodyAndHeader("direct:update", body,
MongoDbConstants.MULTIUPDATE, true);
```

```
// route: from("direct:update").to("mongodb: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("mongodb:myDb?
database=science&collection=notableScientists&operation=update");
String updateObj = "[{\"filterField\": true}, {\"$set\", {\"scientist\", \"Darwin\"}}]";
Object result = template.requestBodyAndHeader("direct:update", updateObj,
MongoDbConstants.MULTIUPDATE, true);
```

94.8.3. 删除操作

94.8.3.1. 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，它预期为 DBObject 类型或可转换的类型。

以下示例将删除在 Science 数据库, notableScientists 集合中，其字段 'conditionField' 等于 true 的所有对象：

```
// route: from("direct:remove").to("mongodb: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)。

94.8.4. 批量写操作

94.8.4.1. bulkWrite

批量执行写入操作，并控制执行顺序。需要 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上中断，而不处理列表中任何剩余的写入操作。要指示 MongoDB 继续处理列表中剩余的写入操作，请将 CamelMongoDbBulkOrdered IN 消息标头设置为 false。未排序的操作是并行执行的，无法保证此行为。

标头键	快速常量	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。	布尔值/布尔值

94.8.5. 其他操作

94.8.5.1. 聚合

使用正文中包含的给定管道执行聚合。聚合可能非常长且繁重的操作。请谨慎使用。

```
// route: from("direct:aggregate").to("mongodb: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("mongodb:myDb?
database=science&collection=notableScientists&operation=aggregate")
   .to("mock:resultAggregate");
```

支持以下 IN 消息标头：

标头键	快速常量	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。	布尔值/布尔值

默认情况下，返回所有结果的列表。根据结果的大小，这在内存上可能很重。一个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可迭代的结果，以便它逐一完成结果。因此，设置批处理大小并返回一个可迭代功能，以便有效地检索和处理结果。

一个示例类似如下：

```
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("mongodb:myDb?
database=science&collection=notableScientists&operation=aggregate&outputType=MongoIterable")
```

```
    .split(body())
    .streaming()
    .to("mock:resultAggregate");
```

请注意，调用 .split (body ()) 足以发送路由一对条目，但仍然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。

94.8.5.2. 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 Document", result instanceof Document);
```

该操作将返回与 shell 中显示的数据结构，格式为 OUT 消息正文中的 Document 形式。

94.8.5.3. 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 Document", result instanceof Document);
```

该操作将返回与 shell 中显示的数据结构，格式为 OUT 消息正文中的 Document 形式。

94.8.5.4. 命令

作为命令在数据库上运行正文。在获取主机信息、复制或分片状态时，对 admin 操作很有用。

这个操作不使用 collection 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);
```

94.8.6. 动态操作

Exchange 可以通过设置由 MongoDbConstants.OPERATION_HEADER 常数定义的 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);
```

94.9. 消费者

有几种类型的消费者：

1. Tailable Cursor Consumer
2. Change Streams Consumer

94.9.1. Tailable Cursor Consumer

MongoDB 提供了一种机制来即时消耗集合中持续数据的机制，使光标保持打开，就像是 *nix 系统的 tail -f 命令一样。由于服务器在客户端可用时将新数据推送至客户端，而不是让客户端在计划的时间间隔返回以获取新数据，因此这种机制比调度的轮询更高效。它还可减少其他冗余网络流量。

使用 tailable 光标仅有一个必要条件：集合必须是 "capped collection"，这意味着它只会保存 N 对象，当达到限制时，MongoDB 会按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，使此功能可用于 Camel 路由。当插入新对象时，MongoDB 会将它们作为 Document 推送到您的可尾部的光标消费者，后者会将它们转换为交换，并将触发您的路由逻辑。

94.10. TAILABLE 光标消费者的工作方式

要将光标转换为可尾部的光标，在首次生成光标时，将向 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，如果新数据没有在确定周期后显示，MongoDB 服务器保留自己终止您的光标的权利。如果您有兴趣继续使用新数据，则必须重新生成光标。为此，您必须记住您离开的位置，否则您将再次从上移使用的位置。

Camel MongoDB tailable 光标消费者为您处理所有这些任务。您只需要在增加性质的数据中为某些字段提供键，这将作为标记来每次重新生成光标时定位，例如时间戳、顺序 ID 等。它可以是 MongoDB 支持的任何 datatype。日期、字符串和整数可以正常工作。我们在此组件上下文中称为"定制"机制。

消费者将记住此字段的最后一个值，无论光标要重新生成时，它将使用类似： increasingField > lastValue 的过滤器运行查询，以便只消耗未读取的数据。

设置 increasing 字段： 在端点 URI tailTrackingIncreasingField 选项上设置 increasing 字段的键。在 Camel 2.10 中，它必须是数据的顶级字段，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 中创建一个票据。

光标重新生成延迟： 需要注意的是，如果新数据在初始时尚未可用，MongoDB 将立即终止光标。由于我们不想在此案例中对服务器进行重负，因此引入了一个 cursorRegenerationDelay 选项（默认值为 1000ms）。您可以修改以适合您的需求。

例如：

```
from("mongodb:myDb?
database=flights&collection=cancellations&tailTrackIncreasingField=departureTime")
    .id("tailableCursorConsumer1")
    .autoStartup(false)
    .to("mock:test");
```

以上路由将使用 "flights.cancellations" capped 集合，使用 "departureTime" 作为增加字段，默认重新生成光标延迟 1000ms。

94.11. 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，您需要立即重启 Camel 容器，然后，您的最后一个值将会丢失，并且您的 tailable 光标消费者再次从上面消耗，很可能将重复记录发送到您的路由中。

要解决这种情况，您可以启用 持久性尾部跟踪 功能，以跟踪 MongoDB 数据库中特殊集合中最后一次消耗的值。当消费者再次初始化时，它将恢复最后跟踪的值，并像没有发生任何情况一样继续。

最后的读取值会在两个字节上保留：每次光标都被重新生成以及消费者关闭时。如果需求需求需求，我们可能还会考虑在常规间隔中保留（每 5 秒刷新一次）。要请求此功能，请在 Camel JIRA 中创建一个

ticket。

94.12. 启用持久性尾部跟踪

要启用此功能，请在端点 URI 中设置以下选项：

- persistentTailTracking 选项为 true
- persistentId 选项为这个消费者的唯一标识符，以便可以在多个消费者间重复使用相同的集合

另外，您可以将 tailTrackDb、tailTrackCollection 和 tailTrackField 选项设置为自定义将存储运行时信息的选项。有关每个选项的描述，请参阅此页面顶部的端点选项表。

例如，以下路由会消耗来自 "flights.cancellations" capped 集合，使用 "departureTime" 作为增加字段，默认重新生成光标延迟 1000ms，并打开持久性尾部跟踪，并在"flights.camelTailTracking"上的"cancellationsTracker" ID 下保留。 在 "lastTrackingValue" 字段下存储最后处理的值 (camelTailTracking 和 lastTrackingValue 是默认值)。

```
from("mongodb:myDb?
database=flights&collection=cancellations&tailTrackIncreasingField=departureTime&persistentTailTracking=true" +
    "&persistentId=cancellationsTracker")
    .id("tailableCursorConsumer2")
    .autoStartup(false)
    .to("mock:test");
```

以下是与上面相同的另一个示例，但持久性尾部跟踪运行时信息将存储在 "trackers.camelTrackers" 集合中，在 "lastProcessedDepartureTime" 字段中：

```
from("mongodb:myDb?
database=flights&collection=cancellations&tailTrackIncreasingField=departureTime&persistentTailTracking=true" +

"&persistentId=cancellationsTracker&tailTrackDb=trackers&tailTrackCollection=camelTrackers" +
    "&tailTrackField=lastProcessedDepartureTime")
    .id("tailableCursorConsumer3")
    .autoStartup(false)
    .to("mock:test");
```

94.12.1. Change Streams Consumer

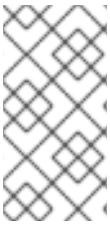
Change Streams 允许应用程序访问实时数据更改，而无需跟踪 MongoDB oplog 的复杂性和风险。应用程序可以使用更改流来订阅集合上的所有数据更改，并立即响应它们。由于更改流使用聚合框架，应用程序也可以过滤特定更改或转换 上的通知。交换正文将包含任何更改的完整文档。

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 MongoDB $match 聚合。它可使用 XML DSL 配置轻松指定：

```
<route id="filterConsumer">
    <from uri="mongodb:myDb?
consumerType=changeStreams&amp;database=flights&amp;collection=tickets&amp;streamFilter={
'$match':{'$or':[{'fullDocument.stringValue': 'specificValue'}]} }"/>
    <to uri="mock:test"/>
</route>
```

Java 配置：

```
from("mongodb:myDb?
consumerType=changeStreams&database=flights&collection=tickets&streamFilter={ '$match':
{'$or':[{'fullDocument.stringValue': 'specificValue'}]} }")
    .to("mock:test");
```



注意

您可以将 streamFilter 值外部化为属性占位符，允许清理端点 URI 参数并更易于阅读。

changeStreams consumer 类型也会返回以下 OUT 标头：

标头键	快速常量	描述（从 MongoDB API 文档中提取）	数据类型
CamelMongoDbStreamOperationType	**MongoDbConstants.STREAM_OPERATION_TYPE**	发生的操作类型。可以是以下值： insert, delete, replace, update, drop, rename, dropDatabase, invalidate。	字符串
_id	**MongoDbConstants.MONGO_ID**	包含由插入、替换、删除、更新操作（如 CRUD 操作）创建或修改的文档的文档。对于分片集合，也显示文档的完整分片密钥。如果 _id 字段已经是分片密钥的一部分，则不会重复它。	ObjectId

94.13. 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** 已经实施 **映射**。
fromStringToDocument	**字符串**	**文档**	使用 **com.mongodb.Document.parse (String s)**。
fromStringToObjectId	**字符串**	**ObjectId**	通过新的 **ObjectId (s)构造一个新的 ObjectId**
fromFileToDocument	**File**	**文档**	在 hood 下 **使用InputStreamToDocument**
fromInputStreamToDocument	**InputStream**	**文档**	将 inputstream 字节转换为 **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>。

这个类型转换器会自动发现，因此您不需要手动配置任何内容。

94.14. 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.mongodb.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.mongodb.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.mongodb.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.mongodb.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.mongodb.mongo-connection	用于连接的共享客户端。组件生成的所有端点都将共享此连接客户端。选项是一个 com.mongodb.client.MongoClient 类型。		MongoClient

第 95 章 MYBATIS

从 Camel 2.7 开始

支持生成者和消费者

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。

95.1. 依赖项

当在 Camel Spring Boot 中使用 mybatis 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-mybatis-starter</artifactId>
</dependency>
```

95.2. URI 格式

```
mybatis:statementName[?options]
```

其中 statementName 是 MyBatis XML 映射文件中的声明名称，它映射到您选择评估的查询、插入、更新或删除操作。

您可以使用以下格式在 URI 中附加查询选项 ?option=value&option=value&...

默认情况下，此组件将从类路径的根目录中加载 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。

95.3. 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

95.3.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

95.3.2. 配置端点选项

端点有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点直接在端点 URI 中作为路径和查询参数完成。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式 的安全 方法。

使用 Property Placeholders 配置不允许硬编码 URL、端口号、敏感信息和其他设置的选项。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

95.4. 组件选项

MyBatis 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型

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
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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
sqlSessionFactory (advanced)	使用 SqlSessionFactory。		SqlSessionFactory

95.5. 端点选项

MyBatis 端点使用 URI 语法进行配置：

```
mybatis:statement
```

以下是 path 和 查询参数。

95.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
statement (common)	**必需** 要评估的 MyBatis XML 映射文件中的声明名称，该文件映射到查询、插入、更新或删除操作。		字符串

95.5.2. 查询参数(30 参数)

Name	描述	默认值	类型
maxMessagesPerPoll (consumer)	这个选项旨在将数据库池返回的结果分成批处理，并将它们传送到多个交换中。此整数定义要在单个交换中传递的最大消息。默认情况下，不会设置最大值。可用于设置限制，例如 1000 个，以避免启动有数千个文件的服务器。设置 0 或 negative 值来禁用它。	0	int
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个跃点。	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

Name	描述	默认值	类型
transacted (consumer)	启用或禁用事务。如果启用，如果处理交换失败，则消费者中断处理任何进一步的交换，从而导致回滚延迟。	false	布尔值
useIterator (consumer)	单独或以列表形式设置进程结果。	true	布尔值
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： * InOnly * InOut * InOptionalOut		ExchangePattern
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy

Name	描述	默认值	类型
processingStrategy (consumer (advanced))	使用自定义 MyBatisProcessingStrategy。		MyBatisProcessingStrategy
executorType (producer)	执行语句时要使用的 executor 类型。simple - executor 不做任何特殊操作。重复使用 - executor 会重复使用准备的语句。executor 会重复使用语句和批处理更新。 Enum 值： * 简单 * 重复使用 X 批处理	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 参数从消息正文中获取。如果设置了 outputHeader，则使用值，并将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，则该值将用作标头的名称，以存储查询结果，并保留原始消息正文。设置 outputHeader 也会省略默认的 CamelMyBatisResult 标头，因为它与 outputHeader all time 相同。		字符串

Name	描述	默认值	类型
statementType (producer)	必须为制作者指定控制要调用的操作类型。 Enum 值： * SelectOne * SelectList * Insert * InsertList * Update * UpdateList * Delete * DeleteList		StatementType
lazyStartProducer (producer (advanced))	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
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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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
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。 Enum 值： * TRACE * DEBUG * INFO * WARN * ERROR * OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： * NANOSECONDS * MICROSECONDS * MILLISECONDS * SECONDS * SECONDS * MINUTES * HOURS * DAYS	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

95.6. 消息标头

***MyBatis** 组件支持下面列出的 2 个消息标头。*

Name	描述	默认值	类型
CamelMyBatisResult (producer) 恒定： **MYBATIS_RESULT**	在任何操作中从 MtBatis 返回的响应。例如，INSERT 可能会返回自动生成的密钥或行数等。		对象
CamelMyBatisStatementName (common) 常量： **MYBATIS_STATEMENT_NAME**	使用的 statementName（例如：insertAccount）。		字符串

95.7. 消息正文

***MyBatis** 的响应将仅设置为正文（如果它是 SELECT 语句）。例如，对于 INSERT 语句 Camel 不会替换正文。这可让您继续路由，并保留原始正文。MyBatis 的响应始终存储在带有键 CamelMyBatisResult 的标头中。*

95.8. SAMPLES

例如，如果您想要使用来自 JMS 队列的 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");
```

您必须指定 statementType，因为您需要指示 Camel 调用哪种操作。

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>
```

95.9. 使用声明TYPE 来更好地控制 MYBATIS

当路由到 MyBatis 端点时，您需要更精细的控制，以便您可以控制要执行的 SQL 语句是 SELECT、UPDATE、DELETE 或 INSERT 等。因此，如果我们想路由到一个 MyBatis 端点，IN 正文包含参数到 SELECT 语句中，我们可以进行：

在上面的代码中，我们可以调用 MyBatis 语句 selectAccountById，IN 正文应包含要检索的帐户 ID，如 Integer 类型。

对于某些其他操作，您可以执行相同的操作，如 SelectList ：

和 UPDATE 相同，您可以在其中将 Account 对象作为 IN 正文发送到 MyBatis ：

95.9.1. 使用 InsertList 语句Type

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 mapper XML 文件中使用 <foreach>。例如，如下所示：

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：

95.9.2. 使用 UpdateList 语句Type

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 mapper XML 文件中使用 <foreach>。例如，如下所示：

```
<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 消息来更新多个行，如下所示：

```
from("direct:start")
    .to("mybatis:batchUpdateAccount?statementType=UpdateList")
    .to("mock:result");
```

95.9.3. 使用 DeleteList 语句Type

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 mapper XML 文件中使用 <foreach>。例如，如下所示：

```
<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");
```

95.9.4. 注意 InsertList、UpdateList 和 DeleteList StatementTypes

任何类型(List、映射等)的参数可以传递给 mybatis，最终用户负责根据需要处理它，并帮助 mybatis 动态查询 功能。

95.9.5. cheduled 轮询示例

这个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 上的 "ScheduledPollConsumer Options"。

或者，您可以使用另一个机制来触发调度的轮询，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");
```

以及使用的 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>
```

95.9.6. 使用 onConsume

此组件支持在 Camel 消耗和处理数据 后执行 语句。这可让您在数据库中进行发布更新。请注意，所有语句都必须是 UPDATE 语句。Camel 支持执行多个名称应用逗号分开的语句。

以下路由说明了执行 consumeAccount 语句数据已被处理。这样，我们可以将数据库中的行的状态更改为处理，因此我们避免消耗两次或更多。

和 sqlmap 文件中的语句：

95.9.7. 参与事务

在 camel-mybatis 下设置事务管理器可能稍微小，因为它涉及在标准 MyBatis SqlMapConfig.xml 文件外部化数据库配置。

第一部分需要设置 DataSource。这通常是一个池(DBCP 或 c3p0)，它需要嵌套在 Spring 代理中。此代理启用了 DataSource 的非 Spring 使用来参与 Spring 事务(MyBatis SqlSessionFactory)。

```
<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>
```

这具有额外的好处，使数据库配置能够使用属性占位符进行外部化。

然后，将事务管理器配置为管理 外部数据源 ：

```
<bean id="txManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
   <property name="dataSource" ref="dataSource"/>
</bean>
```

mybatis-spring SqlSessionFactoryBean，然后换行同一 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，camel-mybatis 组件会使用该工厂进行配置：

```
<bean id="mybatis" class="org.apache.camel.component.mybatis.MyBatisComponent">
    <property name="sqlSessionFactory" ref="sqlSessionFactory"/>
</bean>
```

最后，事务策略在事务管理器的顶部定义，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>
```

95.10. MYBATIS SPRING BOOT STARTER 集成

Spring Boot 用户可以使用 mybatis-spring-boot-starter 工件由 mybatis 团队提供

```
<dependency>
  <groupId>org.mybatis.spring.boot</groupId>
  <artifactId>mybatis-spring-boot-starter</artifactId>
  <version>2.3.0</version>
</dependency>
```

特别是 AutoConfigured Bean from mybatis-spring-boot-starter，如下所示：

```
#application.properties
camel.component.mybatis.sql-session-factory=#sqlSessionFactory
```

95.11. 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.mybatis-bean.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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.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。通过延迟启动，启动失败可以在路由消息期间通过 Camel 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.mybatis-bean.sql-session-factory	使用 SqlSessionFactory。选项是 org.apache.ibatis.session.SqlSessionFactory 类型。		SqlSessionFactory

Name	描述	默认值	类型
camel.component.mybatis.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.mybatis.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.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	是否启用 mybatis 组件的自动配置。这默认是启用的。		布尔值
camel.component.mybatis.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。通过延迟启动，启动失败可以在路由消息期间通过 Camel 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
camel.component.mybatis.sql-session-factory	使用 SqlSessionFactory。选项是 org.apache.ibatis.session.SqlSessionFactory 类型。		SqlSessionFactory

第 96 章 NETTY

支持生成者和消费者

Camel 中的 Netty 组件是一个套接字通信组件，它基于 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 大大简化并简化 TCP 和 UDP 套接字服务器等网络编程。

此 camel 组件支持制作者和消费者端点。

Netty 组件具有多个选项，允许精细控制多个 TCP/UDP 通信参数（缓冲大小、keepAlives、tcpNoDelay 等），并促进 Camel 路由上的 In-Only 和 In-Out 通信。

96.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 netty 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-netty-starter</artifactId>
</dependency>
```

96.2. 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。

96.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

96.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

96.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

96.4. 组件选项

Netty 组件支持 73 选项，如下所列。

Name	描述	默认值	类型
configuration (common)	在创建端点时，使用 NettyConfiguration 作为配置。		NettyConfiguration
disconnect (common)	使用后是否从 Netty Channel 断开（关闭）可用于消费者和制作者。	false	布尔值
keepalive (common)	设置 以确保套接字不会因为不活跃而关闭。	true	布尔值
reuseAddress (common)	将 设置为方便套接字多路。	true	布尔值
reuseChannel (common)	此选项允许生产者和消费者（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 频道。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，通道在 Exchange 完成后不会返回到连接池；如果 disconnect 选项设为 true，则不会断开连接。重复使用的频道作为交换属性存储在 Exchange 上，其键为 NettyConstants#NETTY_CHANNEL，允许您在路由过程中获取频道，并使用它。	false	布尔值
sync (common)	设置，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)	设置 以选择通过 UDP 进行多播。	false	布尔值
clientMode (consumer)	如果 clientMode 为 true，则 netty consumer 将地址连接为 TCP 客户端。	false	布尔值
reconnect (consumer)	仅在消费者中的 clientMode 中使用，如果启用，消费者将尝试重新连接。	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（毫秒）。	10000	int

Name	描述	默认值	类型
backlog (consumer (advanced))	允许为 netty consumer (server)配置积压。请注意，后端只是根据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 backlog 依赖于 OS 设置。		int
bossCount (consumer (advanced))	当 netty 适用于 nio 模式时，它会使用来自 Netty 的默认 bossCount 参数，即 1。用户可以使用此选项覆盖 Netty 的默认 bossCount。	1	int
bossGroup (consumer (advanced))	设置 BossGroup，可用于处理 NettyEndpoint 中服务器端的新连接。		EventLoopGroup
disconnectOnNoReply (consumer (advanced))	如果启用了同步，这个选项会指定 NettyConsumer（如果应该断开连接，但没有回复来回发）。	true	布尔值
executorService (consumer (advanced))	使用给定的 EventExecutorGroup。		EventExecutorGroup
maximumPoolSize (consumer (advanced))	为 netty 消费者排序线程池设置最大线程池大小。默认大小为 2 x cpu_core 加上 1。将此值设置为 eg 10 时将使用 10 个线程，除非 2 x cpu_core 加上 1 是一个更高的值，然后将覆盖并使用。例如，如果存在 8 个内核，则消费者线程池为 17。此线程池用于路由 Camel 从 Netty 接收的消息。我们使用单独的线程池来确保消息排序，如果某些消息将阻断，则 nettys worker 线程（事件循环）不受影响。		int
nettyServerBootstrapFactory (consumer (advanced))	使用自定义 NettyServerBootstrapFactory。		NettyServerBootstrapFactory
networkInterface (consumer (advanced))	在使用 UDP 时，可以使用此选项按名称指定网络接口，如 eth0 来加入多播组。		字符串

Name	描述	默认值	类型
noReplyLogLevel (consumer (advanced))	如果启用了同步，这个选项指定 NettyConsumer，在日志没有回复时使用该级别。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	WARN	LoggingLevel
serverClosedChannelExceptionCaughtLogLevel (consumer (advanced))	如果服务器(NettyConsumer)捕获 java.nio.channels.ClosedChannelException，则其使用此日志记录级别记录。这用于避免记录关闭的频道异常，因为客户端可能会突然断开连接，然后在 Netty 服务器中造成大量关闭异常。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	DEBUG	LoggingLevel
serverExceptionCaughtLogLevel (consumer (advanced))	如果服务器(NettyConsumer)捕获异常，则使用此日志记录级别记录它。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	WARN	LoggingLevel

Name	描述	默认值	类型
serverInitializerFactory (consumer (advanced))	使用自定义 ServerInitializerFactory。		ServerInitializerFactory
使用ExecutorService (consumer (advanced))	是否使用排序的线程池，以确保在同一频道中按顺序处理事件。	true	布尔值
connectTimeout (producer)	等待套接字连接的时间。值以毫秒为单位。	10000	int
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
requestTimeout (producer)	在调用远程服务器时，允许为 Netty producer 使用超时。默认情况下，不使用超时。该值以秒为单位，因此 eg 30000 为 30 秒。requestTimeout 使用 Netty 的 ReadTimeoutHandler 触发超时。		long
clientInitializerFactory (producer (advanced))	使用自定义 ClientInitializerFactory。		ClientInitializerFactory
correlationManager (producer (advanced))	使用自定义关联管理器来管理在使用带有 netty producer 的 request/reply 时如何映射请求和回复消息。只有当您有将请求与回复进行映射时（例如，请求和回复信息中存在关联 ID）时，才应使用此请求和回复。如果要在 netty 中在同一频道(aka connection)上有多个并发消息，则可以使用它。在这样做时，您必须有关联请求和回复消息的方法，以便在继续路由前，将正确的回复存储在 inflight Camel Exchange 上。我们建议在构建自定义关联管理器时扩展 TimeoutCorrelationManagerSupport。这提供了对超时和其他复杂度的支持，否则您需要实施其他复杂性。如需了解更多详细信息，请参阅 producerPoolEnabled 选项。		NettyCamelStateCorrelationManager
lazyChannelCreation (producer (advanced))	如果远程服务器在 Camel 生成者启动时没有启动并运行，可以完全创建频道以避免异常。	true	布尔值

Name	描述	默认值	类型
producerPoolEnabled (producer (advanced))	生成者池是否启用。重要：如果您关闭此，则单个共享连接将用于生成者，您也可以执行 request/reply。这意味着，如果回复没有顺序，则交错响应可能存在潜在的问题。因此，您需要在请求和回复消息中有一个关联 id，以便您可以正确地将回复与负责继续处理 Camel 的消息的 Camel 回调相关联。为此，您需要实施 NettyCamelStateCorrelationManager 作为关联管理器，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
producerPoolMaxIdle (producer (advanced))	对池中空闲实例数量设置上限。	100	int
producerPoolMaxTotal (producer (advanced))	对池可以分配的对象数量（签出给客户端，或闲置一个给定时间等待签出）的对象数量设置上限。对没有限制使用负值。	-1	int
producerPoolMinEvictableIdle (producer (advanced))	在空闲对象驱除者有资格驱除前，对象设置在池中闲置的最小时间（值为 millis）。	300000	long
producerPoolMinIdle (producer (advanced))	在驱除器线程（如果活跃）生成新对象之前，设置制作者池中允许的最小实例数量。		int
udpConnectionlessSending (producer (advanced))	这个选项支持连接 less udp 发送，而这是真正触发并忘记的。如果没有侦听接收端口，则连接的 udp 会发送 PortUnreachableException。	false	布尔值
useByteBuf (producer (advanced))	如果 useByteBuf 为 true，则 netty producer 会将消息正文转换为 ByteBuf，然后再发送它。	false	布尔值
hostnameVerification (security)	要在 SSLEngine 上启用/禁用主机名验证。	false	布尔值
allowSerializedHeaders (advanced)	仅在 transferExchange 为 true 时使用 TCP。当设置为 true 时，标头中的串行对象和属性将添加到交换中。否则 Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值

Name	描述	默认值	类型
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
channelGroup (advanced)	使用明确的 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，且仅在某些平台上被支持。您需要为您要使用的主机操作系统添加 netty JAR。请参阅更多详情：	false	布尔值
选项 (advanced)	允许使用 option. 作为前缀配置额外的 netty 选项。例如，option.child.keepAlive=false 设置 netty 选项 child.keepAlive=false。有关使用的选项，请参阅 Netty 文档。		Map
receiveBufferSize (advanced)	在入站通信期间使用的 TCP/UDP 缓冲区大小。大小为字节。	65536	int
receiveBufferSize Predictor (advanced)	配置缓冲区大小预测器。请参阅 Jetty 文档以及此邮件线程的详细信息。		int
sendBufferSize (advanced)	在出站通信中使用的 TCP/UDP 缓冲区大小。大小为字节。	65536	int
transferExchange (advanced)	仅用于 TCP。您可以通过线路而不是只传输正文来传输交换。以下字段会被传输：在 body, Out body, fault body, In headers, Out headers, fault headers, exchange properties, exchange exception。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值
udpByteArrayCodec (advanced)	仅针对 UDP。如果使用字节数组 codec 而不是 Java 序列化协议启用。	false	布尔值
workerCount (advanced)	当 netty 适用于 nio 模式时，它将使用来自 Netty 的默认 workerCount 参数（即 cpu_core_threads x 2）。用户可以使用此选项覆盖 Netty 的默认 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，每个消费者或制作者都有自己的 worker 池，具有 2 个 x cpu 数核心线程。		EventLoopGroup

Name	描述	默认值	类型
allowDefaultCodec (codec)	如果两者都是 null，则 netty 组件会安装默认的 codec，textline 为 false。将 allowDefaultCodec 设置为 false 可防止 netty 组件作为过滤器链中的第一个元素安装默认的 codec。	true	布尔值
autoAppendDelimiter (codec)	在使用 textline codec 发送时，是否自动附加缺少的结束分隔符。	true	布尔值
decoderMaxLineLength (codec)	用于文本 codec 的最大行长度。	1024	int
decoders (codec)	要使用的解码器列表。您可以使用字符串，其值用逗号分开，并在 Registry 中查找值。只需记住，使用 # so Camel 知道它应该查找的值作为前缀。		list
delimiter (codec)	用于文本 codec 的分隔符。可能的值有 LINE 和 NULL。 Enum 值： • 行 • NULL	行	TextLineDelimiter
encoders (codec)	要使用的编码程序列表。您可以使用字符串，其值用逗号分开，并在 Registry 中查找值。只需记住，使用 # so Camel 知道它应该查找的值作为前缀。		list
编码 (codec)	用于文本 codec 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行的 codec；如果没有指定或值为 false，则通过 TCP 假设对象串行化 - 但是默认情况下，只允许字符串序列化。	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。		字符串

Name	描述	默认值	类型
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 （安全）	要使用的密码设置来加密/解密使用 SSH 发送的有效负载。		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 （安全性）	设置以指定 SSL 加密是否应用到此端点。	false	布尔值
sslClientCertHeaders (security)	启用和采用 SSL 模式时，Netty 使用者将增强 Camel 消息，其中包含有关客户端证书的信息，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)	对可用于返回 SSL 处理程序的类的引用。		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
useGlobalSslContextParameters (security)	启用使用全局 SSL 上下文参数。	false	布尔值

96.5. 端点选项

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：

96.5.1. 路径参数(3 参数)

Name	描述	默认值	类型
protocol (common)	**必需** 要使用的协议，可以是 tcp 或 udp。 Enum 值： • tcp • udp		字符串
host (common)	**必需** 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 (common)	**必需** 主机端口号。		int

96.5.2. 查询参数(71 参数)

Name	描述	默认值	类型
disconnect (common)	使用后是否从 Netty Channel 断开（关闭）可用于消费者和制作者。	false	布尔值
keepalive (common)	设置 以确保套接字不会因为不活跃而关闭。	true	布尔值
reuseAddress (common)	将 设置为方便套接字多路。	true	布尔值
reuseChannel (common)	此选项允许生产者和消费者（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 频道。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，通道在 Exchange 完成后不会返回到连接池；如果 disconnect 选项设为 true，则不会断开连接。重复使用的频道作为交换属性存储在 Exchange 上，其键为 NettyConstants#NETTY_CHANNEL，允许您在路由过程中获取频道，并使用它。	false	布尔值
sync (common)	设置，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

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
broadcast (consumer)	设置 以选择通过 UDP 进行多播。	false	布尔值
clientMode (consumer)	如果 clientMode 为 true，则 netty consumer 将地址连接为 TCP 客户端。	false	布尔值
reconnect (consumer)	仅在消费者中的 clientMode 中使用，如果启用，消费者将尝试重新连接。	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（毫秒）。	10000	int
backlog (consumer (advanced))	允许为 netty consumer (server)配置积压。请注意，后端只是根据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 backlog 依赖于 OS 设置。		int
bossCount (consumer (advanced))	当 netty 适用于 nio 模式时，它会使用来自 Netty 的默认 bossCount 参数，即 1。用户可以使用此选项覆盖 Netty 的默认 bossCount。	1	int
bossGroup (consumer (advanced))	设置 BossGroup，可用于处理 NettyEndpoint 中服务器端的新连接。		EventLoopGroup
disconnectOnNoReply (consumer (advanced))	如果启用了同步，这个选项会指定 NettyConsumer（如果应该断开连接，但没有回复来回发）。	true	布尔值
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
nettyServerBootstrapFactory (consumer (advanced))	使用自定义 NettyServerBootstrapFactory。		NettyServerBootstrapFactory
networkInterface (consumer (advanced))	在使用 UDP 时，可以使用此选项按名称指定网络接口，如 eth0 来加入多播组。		字符串
noReplyLogLevel (consumer (advanced))	如果启用了同步，这个选项指定 NettyConsumer，在日志没有回复时使用该级别。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	WARN	LoggingLevel

Name	描述	默认值	类型
serverClosedChannelExceptionCaughtLogLevel (consumer (advanced))	如果服务器(NettyConsumer)捕获 java.nio.channels.ClosedChannelException，则其使用此日志记录级别记录。这用于避免记录关闭的频道异常，因为客户端可能会突然断开连接，然后在 Netty 服务器中造成大量关闭异常。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	DEBUG	LoggingLevel
serverExceptionCaughtLogLevel (consumer (advanced))	如果服务器(NettyConsumer)捕获异常，则使用此日志记录级别记录它。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	WARN	LoggingLevel
serverInitializerFactory (consumer (advanced))	使用自定义 ServerInitializerFactory。		ServerInitializerFactory
使用 ExecutorService (consumer (advanced))	是否使用排序的线程池，以确保在同一频道中按顺序处理事件。	true	布尔值
connectTimeout (producer)	等待套接字连接的时间。值以毫秒为单位。	10000	int

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
requestTimeout (producer)	在调用远程服务器时，允许为 Netty producer 使用超时。默认情况下，不使用超时。该值以秒为单位，因此 eg 30000 为 30 秒。requestTimeout 使用 Netty 的 ReadTimeoutHandler 触发超时。		long
clientInitializerFactory (producer (advanced))	使用自定义 ClientInitializerFactory。		ClientInitializerFactory
correlationManager (producer (advanced))	使用自定义关联管理器来管理在使用带有 netty producer 的 request/reply 时如何映射请求和回复消息。只有当您有将请求与回复进行映射时（例如，请求和回复信息中存在关联 ID）时，才应使用此请求和回复。如果要在 netty 中在同一频道(aka connection)上有多个并发消息，则可以使用它。在这样做时，您必须有关联请求和回复消息的方法，以便在继续路由前，将正确的回复存储在 inflight Camel Exchange 上。我们建议在构建自定义关联管理器时扩展 TimeoutCorrelationManagerSupport。这提供了对超时和其他复杂度的支持，否则您需要实施其他复杂性。如需了解更多详细信息，请参阅 producerPoolEnabled 选项。		NettyCamelStateCorrelationManager
lazyChannelCreation (producer (advanced))	如果远程服务器在 Camel 生成者启动时没有启动并运行，可以完全创建频道以避免异常。	true	布尔值
producerPoolEnabled (producer (advanced))	生成者池是否启用。重要：如果您关闭此，则单个共享连接将用于生成者，您也可以执行 request/reply。这意味着，如果回复没有顺序，则交错响应可能存在潜在的问题。因此，您需要在请求和回复消息中有一个关联 id，以便您可以正确地将回复与负责继续处理 Camel 的消息的 Camel 回调相关联。为此，您需要实施 NettyCamelStateCorrelationManager 作为关联管理器，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
producerPoolMaxIdle (producer (advanced))	对池中空闲实例数量设置上限。	100	int

Name	描述	默认值	类型
producerPoolMaxTotal (producer (advanced))	对池可以分配的对象数量（签出给客户端，或闲置一个给定时间等待签出）的对象数量设置上限。对没有限制使用负值。	-1	int
producerPoolMinEvictableIdle (producer (advanced))	在空闲对象驱除者有资格驱除前，对象设置在池中闲置的最小时间（值为 millis）。	300000	long
producerPoolMinIdle (producer (advanced))	在驱除器线程（如果活跃）生成新对象之前，设置制作者池中允许的最小实例数量。		int
udpConnectionlessSending (producer (advanced))	这个选项支持连接 less udp 发送，而这是真正触发并忘记的。如果没有侦听接收端口，则连接的 udp 会发送 PortUnreachableException。	false	布尔值
useByteBuf (producer (advanced))	如果 useByteBuf 为 true，则 netty producer 会将消息正文转换为 ByteBuf，然后再发送它。	false	布尔值
hostnameVerification (security)	要在 SSLEngine 上启用/禁用主机名验证。	false	布尔值
allowSerializedHeaders (advanced)	仅在 transferExchange 为 true 时使用 TCP。当设置为 true 时，标头中的串行对象和属性将添加到交换中。否则 Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值
channelGroup (advanced)	使用明确的 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，且仅在某些平台上被支持。您需要为您要使用的主机操作系统添加 netty JAR。请参阅更多详情：	false	布尔值
选项 (advanced)	允许使用 option. 作为前缀配置额外的 netty 选项。例如，option.child.keepAlive=false 设置 netty 选项 child.keepAlive=false。有关使用的选项，请参阅 Netty 文档。		Map
receiveBufferSize (advanced)	在入站通信期间使用的 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器。请参阅 Jetty 文档以及此邮件线程的详细信息。		int

Name	描述	默认值	类型
sendBufferSize (advanced)	在出站通信中使用的 TCP/UDP 缓冲区大小。大小为字节。	65536	int
同步 (advanced)	设置是否应严格使用同步处理。	false	布尔值
transferExchange (advanced)	仅用于 TCP。您可以通过线路而不是只传输正文来传输交换。以下字段会被传输：在 body, Out body, fault body, In headers, Out headers, fault headers, exchange properties, exchange exception。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值
udpByteArrayCodec (advanced)	仅针对 UDP。如果使用字节数组 codec 而不是 Java 序列化协议启用。	false	布尔值
workerCount (advanced)	当 netty 适用于 nio 模式时，它将使用来自 Netty 的默认 workerCount 参数（即 cpu_core_threads x 2）。用户可以使用此选项覆盖 Netty 的默认 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，每个消费者或制作者都有自己的 worker 池，具有 2 个 x cpu 数核心线程。		EventLoopGroup
allowDefaultCodec (codec)	如果两者都是 null，则 netty 组件会安装默认的 codec，textline 为 false。将 allowDefaultCodec 设置为 false 可防止 netty 组件作为过滤器链中的第一个元素安装默认的 codec。	true	布尔值
autoAppendDelimiter (codec)	在使用 textline codec 发送时，是否自动附加缺少的结束分隔符。	true	布尔值
decoderMaxLineLength (codec)	用于文本 codec 的最大行长度。	1024	int
decoders (codec)	要使用的解码器列表。您可以使用字符串，其值用逗号分开，并在 Registry 中查找值。只需记住，使用 # so Camel 知道它应该查找的值作为前缀。		list
delimiter (codec)	用于文本 codec 的分隔符。可能的值有 LINE 和 NULL。 Enum 值： • 行 • NULL	行	TextLineDelimiter

Name	描述	默认值	类型
encoders (codec)	要使用的编码程序列表。您可以使用字符串，其值用逗号分开，并在 Registry 中查找值。只需记住，使用 # so Camel 知道它应该查找的值作为前缀。		list
编码 (codec)	用于文本 codec 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行的 codec；如果没有指定或值为 false，则通过 TCP 假设对象串行化 - 但是默认情况下，只允许字符串序列化。	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。		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 （安全）	要使用的密码设置来加密/解密使用 SSH 发送的有效负载。		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 （安全性）	设置以指定 SSL 加密是否应用到此端点。	false	布尔值
sslClientCertHeaders (security)	启用和采用 SSL 模式时，Netty 使用者将增强 Camel 消息，其中包含有关客户端证书的信息，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)	对可用于返回 SSL 处理程序的类的引用。		SslHandler

Name	描述	默认值	类型
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

96.6. 基于 REGISTRY 的选项

codec Handlers 和 SSL Keystores 可以列在 Registry 中，如 Spring XML 文件中。可以传递的值如下：

Name	描述
passphrase	使用密码设置来加密/解密使用 SSH 发送的有效负载
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：** 用于加密的客户端证书密钥存储
trustStoreFile	**弃用：** 用于加密的服务器端证书密钥存储
keyStoreResource	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	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。
sslHandler	对可用于返回 SSL 处理程序的类的引用
encoder	一个自定义 **ChannelHandler** 类，可用于执行特殊的出站有效负载。必须覆盖 io.netty.channel.ChannelInboundHandlerAdapter。
encoders	要使用的编码程序列表。您可以使用字符串，其值用逗号分开，并在 Registry 中查找值。只需记住，使用 # so Camel 知道它应该查找的值作为前缀。
decoder	一个自定义 **ChannelHandler** 类，可用于执行特殊的入站有效负载。必须覆盖 io.netty.channel.ChannelOutboundHandlerAdapter。

Name	描述
decoders	要使用的解码器列表。您可以使用字符串，其值用逗号分开，并在 Registry 中查找值。只需记住，使用 # so Camel 知道它应该查找的值作为前缀。

注意

请阅读以下关于使用不可共享编码器/解码器的信息。

96.6.1. 使用不可共享编码或解码器

如果您的编码者或解码器不可共享（例如，它们没有 @Shareable 类注解），那么您的 encoder/decoder 必须实施 org.apache.camel.component.netty.ChannelHandlerFactory 接口，并在 newChannelHandler 方法中返回新实例。这是为了确保安全使用编码器/解码器。否则，Netty 组件将在创建端点时记录 WARN。

Netty 组件提供了一个 org.apache.camel.component.netty.ChannelHandlerFactories factory 类，它有许多常用的方法。

96.7. 将消息发送到 NETTY 端点

96.7.1. 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。

生成者模式支持基于单向和请求响应的操作。

96.7.2. 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 text/xml、二进制和基于串行对象的有效负载在套接字上接收请求，*

- 将它们作为消息交换在路由上发送。

消费者模式支持基于单向和请求响应的操作。

96.8. 例子

96.8.1. 使用 Request-Reply 和 serialized 对象有效负载的 UDP Netty 端点

请注意，默认情况下不允许对象序列化，因此必须配置解码器。

```
@BindToRegistry("decoder")
public ChannelHandler getDecoder() throws Exception {
    return new DefaultChannelHandlerFactory() {
        @Override
        public ChannelHandler newChannelHandler() {
            return new
DatagramPacketObjectDecoder(ClassResolvers.weakCachingResolver(null));
        }
    };
}

RouteBuilder builder = new RouteBuilder() {
  public void configure() {
    from("netty:udp://0.0.0.0:5155?sync=true&decoders=#decoder")
      .process(new Processor() {
        public void process(Exchange exchange) throws Exception {
          Poetry poetry = (Poetry) exchange.getIn().getBody();
          // Process poetry in some way
          exchange.getOut().setBody("Message received);
        }
      }
    }
};
```

96.8.2. 使用单向通信的基于 TCP 的 Netty 消费者端点

```
RouteBuilder builder = new RouteBuilder() {
  public void configure() {
      from("netty:tcp://0.0.0.0:5150")
        .to("mock:result");
  }
};
```

96.8.3. 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

Netty 组件通过 Camel JSSE 配置实用程序 支持 SSL/TLS 配置实用程序。这个工具大大减少了您需要写入的组件特定代码数量，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 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 组件中使用基本 SSL/TLS 配置

```
Registry registry = context.getRegistry();
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.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);
        }
      }
  }
});
```

获取 SSLSession 和客户端证书的访问权限

如果需要获取客户端证书的详细信息，您可以获得 javax.net.ssl.SSLSession 的访问权限。当 ssl=true 之后，Netty 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();
```

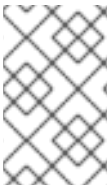
记住设置 needClientAuth=true 以验证客户端，否则 SSLSession 无法访问客户端证书的相关信息，您可能会得到异常 javax.net.ssl.SSLPeerUnverifiedException: peer 未验证。如果客户端证书已过期或者无效，您可能还会获得此例外。

注意

选项 sslClientCertHeaders 可以设置为 true，然后使用包含客户端证书详情的标头增强 Camel 消息。例如，主题名称在标头 CamelNettySSLClientCertSubjectName 中可用。

96.8.4. 使用多个 Codecs

在某些情况下，可能需要将编码器和解码器链添加到 netty 管道中。要将 multpile codecs 添加到 camel netty 端点中，应使用 'encoders' 和 'decoders' uri 参数。与用于提供引用(ChannelUpstreamHandlers 和 ChannelDownstreamHandlers)的 'encoder' 和 'decoder' 参数一样，这些参数应添加到管道中。请注意，如果指定了 encoders，则将忽略 encoder 参数，类似于解码器和解码器参数。



注意

阅读以上关于使用不可共享编码器/解码器的信息。

需要将 codecs 列表添加到 Camel 的 registry 中，以便在端点创建时解析它们。

```
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.netty.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.netty.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，bean 名称可以在 netty 端点定义中使用，可以是用逗号分开的列表，或者包含在 List 中。

```
 from("direct:multiple-codec").to("netty:tcp://0.0.0.0:{{port}}?
encoders=#encoders&sync=false");

 from("netty: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="netty:tcp://0.0.0.0:5150?encoders=#encoders&amp;sync=false"/>
    </route>
    <route>
        <from uri="netty:tcp://0.0.0.0:5150?decoders=#length-decoder,#string-
```

```
decoder&amp;sync=false"/>
        <to uri="mock:multiple-codec"/>
    </route>
</camelContext>
```

96.9. 完成后关闭频道

当用作服务器时，有时您想要关闭频道，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，您也可以根据每个消息指示 Camel，如下所示：
要指示 Camel 关闭频道，您应该添加一个标头，其键为 CamelNettyCloseChannelWhenComplete 设置为布尔值 true。
例如，以下示例会在将 bye 消息写回客户端后关闭频道：

```
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);
        }
    }
});
```

添加自定义频道管道工厂，以获取对创建的管道的完整控制。

96.10. 自定义管道

自定义频道管道通过插入自定义处理器、编码器和解码器，而无需以非常简单的方式在 Netty Endpoint URL 中指定对处理器/中断链的用户提供完全控制。

要添加自定义管道，必须创建一个自定义频道管道工厂，并通过上下文 registry (Registry 或 camel-spring ApplicationContextRegistry 等)进行注册。

自定义管道工厂必须构建如下

- Producer 链接频道管道工厂必须扩展 abstract 类 ClientPipelineFactory。
- Consumer linked 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 （） 方法会创建一个没有处理程序、编码或解码器的管道到管道。

以下示例演示了如何创建 ServerInitializerFactory 工厂

96.10.1. 使用自定义管道工厂

```
public class SampleServerInitializerFactory extends ServerInitializerFactory {
   private int maxLineSize = 1024;

   protected void initChannel(Channel ch) throws Exception {
      ChannelPipeline channelPipeline = ch.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();
registry.bind("spf", factory);
context.addRoutes(new RouteBuilder() {
 public void configure() {
    String netty_ssl_endpoint =
      "netty: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);
       }
```

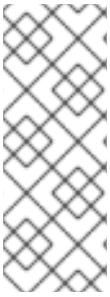
```
        }
    }
});
```

96.11. 重新使用 NETTY BOSS 和 WORKER 线程池

Netty 具有两种类型的线程池：boss 和 worker。默认情况下，每个 Netty 消费者和制作者都有自己的专用线程池。如果要在多个消费者或生成者间重复使用这些线程池，则必须在 Registry 中创建并加入线程池。

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>
```



注意

对于 boss 线程池，具有用于 Netty 使用者的 org.apache.camel.component.netty.NettyServerBossPoolBuilder 构建器，以及用于 Netty producer 的 org.apache.camel.component.netty.NettyClientBossPoolBuilder。

然后，在 Camel 路由中，我们可以通过配置 URI 中的 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;usingExecutorService=false"/>
  <to uri="log:result"/>
  ...
</route>
```

如果我们有另一个路由，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;usingExecutorService=false"/>
  <to uri="log:result"/>
  ...
</route>
```

以此类推。

96.12. 使用 REQUEST/REPLY 在单个连接中多路并发消息

当使用 Netty 进行请求/回复时，netty producer 会默认通过非共享连接（池）发送每个消息。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。换句话说，请求/回复消息之间的关联是开箱即用的，因为回复会返回到用于发送请求的相同连接；此连接不会与他人共享。当响应返回时，连接返回回连接池，以供他人重复使用。

但是，如果您要在单个共享连接上有多个x 并发请求/响应，则需要通过设置 producerPoolEnabled=false 来关闭连接池。现在，这意味着如果回复没有顺序，则交错响应可能存在潜在的问题。因此，您需要在请求和回复消息中有一个关联 id，以便您可以正确地将回复与负责继续处理 Camel 的消息的 Camel 回调相关联。要做到这一点，您需要将 NettyCamelStateCorrelationManager 作为关联管理器实现，并通过 correlationManager=#myManager 选项进行配置。

注意

我们建议在构建自定义关联管理器时扩展 TimeoutCorrelationManagerSupport。这提供了对超时和其他复杂度的支持，否则您需要实施其他复杂性。

您可以在 camel-example-netty-custom-correlation 目录下的示例目录中找到带有 Apache Camel 源代码的示例。

96.13. SPRING BOOT AUTO-CONFIGURATION

组件支持 74 选项，如下所列。

Name	描述	默认值	类型
camel.component.netty.allow-default-codec	如果两者都是 null，则 netty 组件会安装默认的 codec，textline 为 false。将 allowDefaultCodec 设置为 false 可防止 netty 组件作为过滤器链中的第一个元素安装默认的 codec。	true	布尔值

Name	描述	默认值	类型
camel.component.netty.allow-serialized-headers	仅在 transferExchange 为 true 时使用 TCP。当设置为 true 时，标头中的串行对象和属性将添加到交换中。否则 Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值
camel.component.netty.auto-append-delimiter	在使用 textline 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.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.netty.backlog	允许为 netty consumer (server)配置积压。请注意，后端只是根据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 backlog 依赖于 OS 设置。		整数
camel.component.netty.boss-count	当 netty 适用于 nio 模式时，它会使用来自 Netty 的默认 bossCount 参数，即 1。用户可以使用此选项覆盖 Netty 的默认 bossCount。	1	整数
camel.component.netty.boss-group	设置 BossGroup，可用于处理 NettyEndpoint 中服务器端的新连接。选项是一个 io.netty.channel.EventLoopGroup 类型。		EventLoopGroup
camel.component.netty.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.netty.broadcast	设置 以选择通过 UDP 进行多播。	false	布尔值
camel.component.netty.channel-group	使用明确的 ChannelGroup。选项是一个 io.netty.channel.group.ChannelGroup 类型。		ChannelGroup
camel.component.netty.client-initializer-factory	使用自定义 ClientInitializerFactory。选项是 org.apache.camel.component.netty.ClientInitializerFactory 类型。		ClientInitializerFactory

Name	描述	默认值	类型
camel.component.netty.client-mode	如果 clientMode 为 true，则 netty consumer 将地址连接为 TCP 客户端。	false	布尔值
camel.component.netty.configuration	在创建端点时，使用 NettyConfiguration 作为配置。选项是 org.apache.camel.component.netty.NettyConfiguration 类型。		NettyConfiguration
camel.component.netty.connect-timeout	等待套接字连接的时间。值以毫秒为单位。	10000	整数
camel.component.netty.correlation-manager	使用自定义关联管理器来管理在使用带有 netty producer 的 request/reply 时如何映射请求和回复消息。只有当您有将请求与回复进行映射时（例如，请求和回复信息中存在关联 ID）时，才应使用此请求和回复。如果要在 netty 中在同一频道(aka connection)上有多个并发消息，则可以使用它。在这样做时，您必须有关联请求和回复消息的方法，以便在继续路由前，将正确的回复存储在 inflight Camel Exchange 上。我们建议在构建自定义关联管理器时扩展 TimeoutCorrelationManagerSupport。这提供了对超时和其他复杂度的支持，否则您需要实施其他复杂性。如需了解更多详细信息，请参阅 producerPoolEnabled 选项。选项是 org.apache.camel.component.netty.NettyCamelStateCorrelationManager 类型。		NettyCamelStateCorrelationManager
camel.component.netty.decoder-max-line-length	用于文本 codec 的最大行长度。	1024	整数
camel.component.netty.decoders	要使用的解码器列表。您可以使用字符串，其值用逗号分开，并在 Registry 中查找值。只需记住，使用 # so Camel 知道它应该查找的值作为前缀。		字符串
camel.component.netty.delimiter	用于文本 codec 的分隔符。可能的值有 LINE 和 NULL。		TextLineDelimiter
camel.component.netty.disconnect	使用后是否从 Netty Channel 断开（关闭）可用于消费者和制作者。	false	布尔值
camel.component.netty.disconnect-on-no-reply	如果启用了同步，这个选项会指定 NettyConsumer（如果应该断开连接，但没有回复来回发）。	true	布尔值
camel.component.netty.enabled	是否启用 netty 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.netty.enabled-protocols	使用 SSL 时要启用的协议。	TLSv1, TLSv1.1, TLSv1.2	字符串
camel.component.netty.encoders	要使用的编码程序列表。您可以使用字符串，其值用逗号分开，并在 Registry 中查找值。只需记住，使用 # so Camel 知道它应该查找的值作为前缀。		字符串
camel.component.netty.encoding	用于文本 codec 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.executor-service	使用给定的 EventExecutorGroup。选项是一个 io.netty.util.concurrent.EventExecutorGroup 类型。		EventExecutorGroup
camel.component.netty.hostname-verification	要在 SSLEngine 上启用/禁用主机名验证。	false	布尔值
camel.component.netty.keep-alive	设置 以确保套接字不会因为不活跃而关闭。	true	布尔值
camel.component.netty.key-store-file	用于加密的客户端侧证书密钥存储。		File
camel.component.netty.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.netty.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.lazy-channel-creation	如果远程服务器在 Camel 生成者启动时没有启动并运行，可以完全创建频道以避免异常。	true	布尔值
camel.component.netty.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

Name	描述	默认值	类型
camel.component.netty.maximum-pool-size	为 netty 消费者排序线程池设置最大线程池大小。默认大小为 2 x cpu_core 加上 1。将此值设置为 eg 10 时将使用 10 个线程，除非 2 x cpu_core 加上 1 是一个更高的值，然后将覆盖并使用。例如，如果存在 8 个内核，则消费者线程池为 17。此线程池用于路由 Camel 从 Netty 接收的消息。我们使用单独的线程池来确保消息排序，如果某些消息将阻断，则 nettys worker 线程（事件循环）不受影响。		整数
camel.component.netty.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，且仅在某些平台上被支持。您需要为您要使用的主机操作系统添加 netty JAR。请参阅更多详情：	false	布尔值
camel.component.netty.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.netty-server-bootstrap-factory	使用自定义 NettyServerBootstrapFactory。选项是 org.apache.camel.component.netty.NettyServerBootstrapFactory 类型。		NettyServerBootstrapFactory
camel.component.netty.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 来加入多播组。		字符串
camel.component.netty.no-reply-log-level	如果启用了同步，这个选项指定 NettyConsumer，在日志没有回复时使用该级别。		LoggingLevel
camel.component.netty.options	允许使用 option. 作为前缀配置额外的 netty 选项。例如，option.child.keepAlive=false 设置 netty 选项 child.keepAlive=false。有关使用的选项，请参阅 Netty 文档。		Map
camel.component.netty.passphrase	要使用的密码设置来加密/解密使用 SSH 发送的有效负载。		字符串
camel.component.netty.producer-pool-enabled	生成者池是否启用。重要：如果您关闭此，则单个共享连接将用于生成者，您也可以执行 request/reply。这意味着，如果回复没有顺序，则交错响应可能存在潜在的问题。因此，您需要在请求和回复消息中有一个关联 id，以便您可以正确地将回复与负责继续处理 Camel 的消息的 Camel 回调相关联。为此，您需要实施 NettyCamelStateCorrelationManager 作为关联管理器，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

Name	描述	默认值	类型
camel.component.netty.producer-pool-max-idle	对池中空闲实例数量设置上限。	100	整数
camel.component.netty.producer-pool-max-total	对池可以分配的对象数量（签出给客户端，或闲置一个给定时间等待签出）的对象数量设置上限。对没有限制使用负值。	-1	整数
camel.component.netty.producer-pool-min-evictable-idle	在空闲对象驱除者有资格驱除前，对象设置在池中闲置的最小时间（值为 millis）。	300000	Long
camel.component.netty.producer-pool-min-idle	在驱除器线程（如果活跃）生成新对象之前，设置制作者池中允许的最小实例数量。		整数
camel.component.netty.receive-buffer-size	在入站通信期间使用的 TCP/UDP 缓冲区大小。大小为字节。	65536	整数
camel.component.netty.receive-buffer-size-predictor	配置缓冲区大小预测器。请参阅 Jetty 文档以及此邮件线程的详细信息。		整数
camel.component.netty.reconnect	仅在消费者中的 clientMode 中使用，如果启用，消费者将尝试重新连接。	true	布尔值
camel.component.netty.reconnect-interval	如果启用了 reconnect 和 clientMode，则使用。尝试重新连接的时间间隔（毫秒）。	10000	整数
camel.component.netty.request-timeout	在调用远程服务器时，允许为 Netty producer 使用超时。默认情况下，不使用超时。该值以秒为单位，因此 eg 30000 为 30 秒。requestTimeout 使用 Netty 的 ReadTimeoutHandler 触发超时。		Long
camel.component.netty.reuse-address	将 设置为方便套接字多路。	true	布尔值

Name	描述	默认值	类型
camel.component.netty.reuse-channel	此选项允许生产者和消费者（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 频道。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，通道在 Exchange 完成后不会返回到连接池；如果 disconnect 选项设为 true，则不会断开连接。重复使用的频道作为交换属性存储在 Exchange 上，其键为 NettyConstants#NETTY_CHANNEL，允许您在路由过程中获取频道，并使用它。	false	布尔值
camel.component.netty.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.netty.send-buffer-size	在出站通信中使用的 TCP/UDP 缓冲区大小。大小为字节。	65536	整数
camel.component.netty.server-closed-channel-exception-caught-log-level	如果服务器(NettyConsumer)捕获 java.nio.channels.ClosedChannelException，则其使用此日志记录级别记录。这用于避免记录关闭的频道异常，因为客户端可能会突然断开连接，然后在 Netty 服务器中造成大量关闭异常。		LoggingLevel
camel.component.netty.server-exception-caught-log-level	如果服务器(NettyConsumer)捕获异常，则使用此日志记录级别记录它。		LoggingLevel
camel.component.netty.server-initializer-factory	使用自定义 ServerInitializerFactory。选项是 org.apache.camel.component.netty.ServerInitializerFactory 类型。		ServerInitializerFactory
camel.component.netty.ssl	设置以指定 SSL 加密是否应用到此端点。	false	布尔值
camel.component.netty.ssl-client-cert-headers	启用和采用 SSL 模式时，Netty 使用者将增强 Camel 消息，其中包含有关客户端证书的信息，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.ssl-context-parameters	使用 SSLContextParameters 配置安全性。选项是 org.apache.camel.support.jsse.SSLContextParameters 类型。		SSLContextParameters
camel.component.netty.ssl-handler	对可用于返回 SSL 处理程序的类的引用。选项是 io.netty.handler.ssl.SslHandler 类型。		SslHandler

Name	描述	默认值	类型
camel.component.netty.sync	设置，将端点设置为单向或请求响应。	true	布尔值
camel.component.netty.tcp-no-delay	设置 以提高 TCP 协议性能。	true	布尔值
camel.component.netty.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行的 codec；如果没有指定或值为 false，则通过 TCP 假设对象串行化 - 但是默认情况下，只允许字符串序列化。	false	布尔值
camel.component.netty.transfer-exchange	仅用于 TCP。您可以通过线路而不是只传输正文来传输交换。以下字段会被传输：在 body, Out body, fault body, In headers, Out headers, fault headers, exchange properties, exchange exception。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值
camel.component.netty.trust-store-file	用于加密的服务器端证书密钥存储。		File
camel.component.netty.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.udp-byte-array-codec	仅针对 UDP。如果使用字节数组 codec 而不是 Java 序列化协议启用。	false	布尔值
camel.component.netty.udp-connectionless-sending	这个选项支持连接 less udp 发送，而这是真正触发并忘记的。如果没有侦听接收端口，则连接的 udp 会发送 PortUnreachableException。	false	布尔值
camel.component.netty.use-byte-buf	如果 useByteBuf 为 true，则 netty producer 会将消息正文转换为 ByteBuf，然后再发送它。	false	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.using-executor-service	是否使用排序的线程池，以确保在同一频道中按顺序处理事件。	true	布尔值

Name	描述	默认值	类型
camel.component.netty.worker-count	当 netty 适用于 nio 模式时，它将使用来自 Netty 的默认 workerCount 参数（即 cpu_core_threads x 2）。用户可以使用此选项覆盖 Netty 的默认 workerCount。		整数
camel.component.netty.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，每个消费者或制作者都有自己的 worker 池，具有 2 个 x cpu 数核心线程。选项是一个 io.netty.channel.EventLoopGroup 类型。		EventLoopGroup

第 97 章 NETTY HTTP

自 Camel 2.14 起

支持生成者和消费者

Netty HTTP 组件是 Netty 组件的扩展，可通过 Netty 联邦 HTTP 传输。

注意

Stream

Netty 基于流，这意味着它收到的输入将提交给 Camel 作为流。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，消息正文似乎为空，或者您需要多次访问数据（例如：执行多播或重新传送错误处理），您应该使用流缓存，或将消息正文转换为一个字符串，可以安全地重新读取多次。注意 Netty HTTP 使用 io.netty.handler.codec.http.HttpObjectAggregator 将整个流读取到内存中，以构建整个完整的 http 消息。但是生成的消息仍然是基于流的消息，一次可读取。

97.1. 依赖项

当在 Camel Spring Boot 中使用 netty-http 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-netty-http-starter</artifactId>
</dependency>
```

97.2. URI 格式

netty 组件的 URI 方案如下

```
netty-http:http://0.0.0.0:8080[?options]
```

注意

查询参数和端点选项

您可能会知道 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 compression 是 Camel 端点选项。在这种情形中，Camel 使用的策略是解析可用的端点选项并从 URI 中删除它们。这意味着，对于讨论的示例，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 解析和删除。另请注意，您不能使用动态标头（如 CamelHttpQuery）指定端点选项。端点选项只能在端点 URI 定义级别指定（例如，或从 DSL 元素指定）。

重要

此组件从 Netty 继承所有选项。请注意，在使用此 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。

97.3. 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

97.3.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

97.3.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

97.4. 组件选项

Netty HTTP 组件支持 80 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	在创建端点时，使用 NettyConfiguration 作为配置。		NettyConfiguration
disconnect (common)	使用后是否从 Netty Channel 断开（关闭）可用于消费者和制作者。	false	布尔值
keepalive (common)	设置 以确保套接字不会因为不活跃而关闭。	true	布尔值
reuseAddress (common)	将 设置为方便套接字多路。	true	布尔值
reuseChannel (common)	此选项允许生产者和消费者（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 频道。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，通道在 Exchange 完成后不会返回到连接池；如果 disconnect 选项设为 true，则不会断开连接。重复使用的频道作为交换属性存储在 Exchange 上，其键为 NettyConstants#NETTY_CHANNEL，允许您在路由过程中获取频道，并使用它。	false	布尔值
sync (common)	设置，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

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
broadcast (consumer)	设置 以选择通过 UDP 进行多播。	false	布尔值
clientMode (consumer)	如果 clientMode 为 true，则 netty consumer 将地址连接为 TCP 客户端。	false	布尔值
muteException (consumer)	如果对消费者启用并且交换失败，响应的正文不会包含异常的堆栈跟踪。	false	布尔值
reconnect (consumer)	仅在消费者中的 clientMode 中使用，如果启用，消费者将尝试重新连接。	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（毫秒）。	10000	int
backlog (consumer (advanced))	允许为 netty consumer (server)配置积压。请注意，后端只是根据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 backlog 依赖于 OS 设置。		int
bossCount (consumer (advanced))	当 netty 适用于 nio 模式时，它会使用来自 Netty 的默认 bossCount 参数，即 1。用户可以使用此选项覆盖 Netty 的默认 bossCount。	1	int
bossGroup (consumer (advanced))	设置 BossGroup，可用于处理 NettyEndpoint 中服务器端的新连接。		EventLoopGroup
disconnectOnNoReply (consumer (advanced))	如果启用了同步，这个选项会指定 NettyConsumer（如果应该断开连接，但没有回复来回发）。	true	布尔值
executorService (consumer (advanced))	使用给定的 EventExecutorGroup。		EventExecutorGroup

Name	描述	默认值	类型
maximumPoolSize (consumer (advanced))	为 netty 消费者排序线程池设置最大线程池大小。默认大小为 2 x cpu_core 加上 1。将此值设置为 eg 10 时将使用 10 个线程，除非 2 x cpu_core 加上 1 是一个更高的值，然后将覆盖并使用。例如，如果存在 8 个内核，则消费者线程池为 17。此线程池用于路由 Camel 从 Netty 接收的消息。我们使用单独的线程池来确保消息排序，如果某些消息将阻断，则 nettys worker 线程（事件循环）不受影响。		int
nettyServerBootstrapFactory (consumer (advanced))	使用自定义 NettyServerBootstrapFactory。		NettyServerBootstrapFactory
networkInterface (consumer (advanced))	在使用 UDP 时，可以使用此选项按名称指定网络接口，如 eth0 来加入多播组。		字符串
noReplyLogLevel (consumer (advanced))	如果启用了同步，这个选项指定 NettyConsumer，在日志没有回复时使用该级别。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	WARN	LoggingLevel

Name	描述	默认值	类型
serverClosedChannelExceptionCaughtLogLevel (consumer (advanced))	如果服务器(NettyConsumer)捕获 java.nio.channels.ClosedChannelException，则其使用此日志记录级别记录。这用于避免记录关闭的频道异常，因为客户端可能会突然断开连接，然后在 Netty 服务器中造成大量关闭异常。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	DEBUG	LoggingLevel
serverExceptionCaughtLogLevel (consumer (advanced))	如果服务器(NettyConsumer)捕获异常，则使用此日志记录级别记录它。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	WARN	LoggingLevel
serverInitializerFactory (consumer (advanced))	使用自定义 ServerInitializerFactory。		ServerInitializerFactory
使用 ExecutorService (consumer (advanced))	是否使用排序的线程池，以确保在同一频道中按顺序处理事件。	true	布尔值
connectTimeout (producer)	等待套接字连接的时间。值以毫秒为单位。	10000	int

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
requestTimeout (producer)	在调用远程服务器时，允许为 Netty producer 使用超时。默认情况下，不使用超时。该值以秒为单位，因此 eg 30000 为 30 秒。requestTimeout 使用 Netty 的 ReadTimeoutHandler 触发超时。		long
clientInitializerFactory (producer (advanced))	使用自定义 ClientInitializerFactory。		ClientInitializerFactory
correlationManager (producer (advanced))	使用自定义关联管理器来管理在使用带有 netty producer 的 request/reply 时如何映射请求和回复消息。只有当您有将请求与回复进行映射时（例如，请求和回复信息中存在关联 ID）时，才应使用此请求和回复。如果要在 netty 中在同一频道(aka connection)上有多个并发消息，则可以使用它。在这样做时，您必须有关联请求和回复消息的方法，以便在继续路由前，将正确的回复存储在 inflight Camel Exchange 上。我们建议在构建自定义关联管理器时扩展 TimeoutCorrelationManagerSupport。这提供了对超时和其他复杂度的支持，否则您需要实施其他复杂性。如需了解更多详细信息，请参阅 producerPoolEnabled 选项。		NettyCamelStateCorrelationManager
lazyChannelCreation (producer (advanced))	如果远程服务器在 Camel 生成者启动时没有启动并运行，可以完全创建频道以避免异常。	true	布尔值
producerPoolBlockWhenExhausted (producer (advanced))	设置 blockWhenExhausted 配置属性的值。它决定是否在池耗尽时是否调用 borrowObject （）方法（达到了最大活跃对象数）。	true	布尔值
producerPoolEnabled (producer (advanced))	生成者池是否启用。重要：如果您关闭此，则单个共享连接将用于生成者，您也可以执行 request/reply。这意味着，如果回复没有顺序，则交错响应可能存在潜在的问题。因此，您需要在请求和回复消息中有一个关联 id，以便您可以正确地将回复与负责继续处理 Camel 的消息的 Camel 回调相关联。为此，您需要实施 NettyCamelStateCorrelationManager 作为关联管理器，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

Name	描述	默认值	类型
producerPoolMaxIdle (producer (advanced))	对池中空闲实例数量设置上限。	100	int
producerPoolMaxTotal (producer (advanced))	对池可以分配的对象数量（签出给客户端，或闲置一个给定时间等待签出）的对象数量设置上限。对没有限制使用负值。	-1	int
producerPoolMaxWait (producer (advanced))	当池耗尽，producerPoolBlockWhenExhausted 为 true 时，设置 borrowObject （）方法的最长持续时间（值为 millis）在抛出异常前应阻止。当小于 0 时，borrowObject （）方法可能会无限期地阻止。	-1	long
producerPoolMinEvictableIdle (producer (advanced))	在空闲对象驱除者有资格驱除前，对象设置在池中闲置的最小时间（值为 millis）。	300000	long
producerPoolMinIdle (producer (advanced))	在驱除器线程（如果活跃）生成新对象之前，设置制作者池中允许的最小实例数量。		int
udpConnectionlessSending (producer (advanced))	这个选项支持连接 less udp 发送，而这是真正触发并忘记的。如果没有侦听接收端口，则连接的 udp 会发送 PortUnreachableException。	false	布尔值
useByteBuf (producer (advanced))	如果 useByteBuf 为 true，则 netty producer 会将消息正文转换为 ByteBuf，然后再发送它。	false	布尔值
hostnameVerification (security)	要在 SSLEngine 上启用/禁用主机名验证。	false	布尔值
allowSerializedHeaders (advanced)	仅在 transferExchange 为 true 时使用 TCP。当设置为 true 时，标头中的串行对象和属性将添加到交换中。否则 Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
channelGroup (advanced)	使用明确的 ChannelGroup。		ChannelGroup

Name	描述	默认值	类型
headerFilterStrategy (advanced)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 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，且仅在某些平台上被支持。您需要为您要使用的主机操作系统添加 netty JAR。请参阅更多详情：	false	布尔值
NettyHttpBinding (advanced)	使用自定义 org.apache.camel.component.netty.http.NettyHttpBinding 绑定到/从 Netty 和 Camel Message API 绑定。		NettyHttpBinding
选项 (advanced)	允许使用 option. 作为前缀配置额外的 netty 选项。例如，option.child.keepAlive=false 设置 netty 选项 child.keepAlive=false。有关使用的选项，请参阅 Netty 文档。		Map
receiveBufferSize (advanced)	在入站通信期间使用的 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器。请参阅 Jetty 文档以及此邮件线程的详细信息。		int
sendBufferSize (advanced)	在出站通信中使用的 TCP/UDP 缓冲区大小。大小为字节。	65536	int
transferExchange (advanced)	仅用于 TCP。您可以通过线路而不是只传输正文来传输交换。以下字段会被传输：在 body, Out body, fault body, In headers, Out headers, fault headers, exchange properties, exchange exception。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值
udpByteArrayCodec (advanced)	仅针对 UDP。如果使用字节数组 codec 而不是 Java 序列化协议启用。	false	布尔值
unixDomainSocketPath (advanced)	要使用的 unix 域套接字的路径，而不是 inet 套接字。但是，将不会使用主机和端口参数。为它们设置 dummy 值是 ok。必须与 nativeTransport=true 和 clientMode=false 一起使用。		字符串
workerCount (advanced)	当 netty 适用于 nio 模式时，它将使用来自 Netty 的默认 workerCount 参数（即 cpu_core_threads x 2）。用户可以使用此选项覆盖 Netty 的默认 workerCount。		int

Name	描述	默认值	类型
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，每个消费者或制作者都有自己的 worker 池，具有 2 个 x cpu 数核心线程。		EventLoopGroup
allowDefaultCodec (codec)	如果两者都是 null，则 netty 组件会安装默认的 codec，textline 为 false。将 allowDefaultCodec 设置为 false 可防止 netty 组件作为过滤器链中的第一个元素安装默认的 codec。	true	布尔值
autoAppendDelimiter (codec)	在使用 textline codec 发送时，是否自动附加缺少的结束分隔符。	true	布尔值
decoderMaxLineLength (codec)	用于文本 codec 的最大行长度。	1024	int
decoders (codec)	要使用的解码器列表。您可以使用字符串，其值用逗号分开，并在 Registry 中查找值。只需记住，使用 # so Camel 知道它应该查找的值作为前缀。		字符串
delimiter (codec)	用于文本 codec 的分隔符。可能的值有 LINE 和 NULL。 Enum 值： • 行 • NULL	行	TextLineDelimiter
encoders (codec)	要使用的编码程序列表。您可以使用字符串，其值用逗号分开，并在 Registry 中查找值。只需记住，使用 # so Camel 知道它应该查找的值作为前缀。		字符串
编码 (codec)	用于文本 codec 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行的 codec；如果没有指定或值为 false，则通过 TCP 假设对象串行化 - 但是默认情况下，只允许字符串序列化。	false	布尔值
enabledProtocols (security)	使用 SSL 时要启用的协议。	TLSv1.2,TLSv1.3	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 （安全）	要使用的密码设置来加密/解密使用 SSH 发送的有效负载。		字符串
securityConfiguration (security)	指的是 org.apache.camel.component.netty.http.NettyHttpSecurityConfiguration 来配置安全 Web 资源。		NettyHttpSecurityConfiguration
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 （安全性）	设置以指定 SSL 加密是否应用到此端点。	false	布尔值
sslClientCertHeaders (security)	启用和采用 SSL 模式时，Netty 使用者将增强 Camel 消息，其中包含有关客户端证书的信息，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)	对可用于返回 SSL 处理程序的类的引用。		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
useGlobalSslContextParameters (security)	启用使用全局 SSL 上下文参数。	false	布尔值

97.5. 端点选项

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：

97.5.1. 路径参数(4 参数)

Name	描述	默认值	类型
protocol (common)	**必需** ：要使用的协议，可以是 http、https 或 proxy - 仅限消费者选项。 Enum 值： • http • https		字符串
host (common)	当成为消费者时，**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 (common)	主机端口号。		int
path (common)	资源路径。		字符串

97.5.2. 查询参数(85 参数)

Name	描述	默认值	类型
bridgeEndpoint (common)	如果选项为 true，则制作者将忽略 NettyHttpConstants.HTTP_URI 标头，并使用端点的 URI 请求。您还可以将 throwExceptionOnFailure 设置为 false，以便生成者将所有故障响应发回。网桥模式中的消费者将跳过 gzip 压缩和 WWW URL 表单编码（通过将 Exchange.SKIP_GZIP_ENCODING 和 Exchange.SKIP_WW_FORM_URLENCODED 标头添加到被消耗的交换中）。	false	布尔值
disconnect (common)	使用后是否从 Netty Channel 断开（关闭）可用于消费者和制作者。	false	布尔值
keepalive (common)	设置 以确保套接字不会因为不活跃而关闭。	true	布尔值

Name	描述	默认值	类型
reuseAddress (common)	将 设置为方便套接字多路。	true	布尔值
reuseChannel (common)	此选项允许生产者和消费者（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 频道。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，通道在 Exchange 完成后不会返回到连接池；如果 disconnect 选项设为 true，则不会断开连接。重复使用的频道作为交换属性存储在 Exchange 上，其键为 NettyConstants#NETTY_CHANNEL，允许您在路由过程中获取频道，并使用它。	false	布尔值
sync (common)	设置，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
matchOnUriPrefix (consumer)	如果找不到完全匹配，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
muteException (consumer)	如果对消费者启用并且交换失败，响应的正文不会包含异常的堆栈跟踪。	false	布尔值
send503whenSuspended (consumer)	消费者暂停时，是否发送 HTTP 状态代码 503。如果选项为 false，则 Netty Acceptor 在消费者暂停时为 unbound，因此客户端无法再连接。	true	布尔值
backlog (consumer (advanced))	允许为 netty consumer (server)配置积压。请注意，后端只是根据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 backlog 依赖于 OS 设置。		int
bossCount (consumer (advanced))	当 netty 适用于 nio 模式时，它会使用来自 Netty 的默认 bossCount 参数，即 1。用户可以使用此选项覆盖 Netty 的默认 bossCount。	1	int
bossGroup (consumer (advanced))	设置 BossGroup，可用于处理 NettyEndpoint 中服务器端的新连接。		EventLoopGroup
bridgeErrorHandler (consumer (advanced))	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
chunkedMaxContentLength (consumer (advanced))	值（以字节为单位），每个块帧在 Netty HTTP 服务器上接收的最大内容长度。	1048576	int
compression (consumer (advanced))	如果客户端支持来自 HTTP 标头的 gzip/deflate，则允许使用 gzip/deflate 在 Netty HTTP 服务器上压缩。	false	布尔值
disconnectOnNoReply (consumer (advanced))	如果启用了同步，这个选项会指定 NettyConsumer（如果应该断开连接，但没有回复来回发）。	true	布尔值
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut		ExchangePattern
httpMethodRestrict (consumer (advanced))	要在 Netty HTTP 使用者中禁用 HTTP 方法。您可以指定用逗号分开的多个。		字符串
logWarnOnBadRequest (consumer (advanced))	如果解码 HTTP 请求失败并且返回 HTTP Status 400（请求），则返回 Netty HTTP 服务器。	true	布尔值
mapHeaders (consumer (advanced))	如果启用了这个选项，则在从 Netty 到 Camel Message 绑定时，也会映射标头（如将标头添加到 Camel 消息）。您可以关闭这个选项来禁用这个选项。仍可从 org.apache.camel.component.netty.http.NettyHttpMessage 消息访问标头，该消息返回 Netty HTTP 请求 io.netty.handler.codec.http.HttpRequest 实例。	true	布尔值
maxChunkSize (consumer (advanced))	内容或每个块的最大长度。如果内容长度（或每个块的长度）超过这个值，则内容或块将分成多个 io.netty.handler.codec.http.HttpContents，其长度为 maxChunkSize。请参阅 io.netty.handler.codec.http.HttpObjectDecoder。	8192	int

Name	描述	默认值	类型
maxHeaderSize (consumer (advanced))	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
maxInitialLineLength (consumer (advanced))	如果初始行的最大长度（如 \\{code GET / HTTP/1.0} 或 \\{code HTTP/1.0 200 OK}）如果初始行的长度超过这个值，则会引发 TooLongFrameException。请参阅 io.netty.handler.codec.http.HttpObjectDecoder。	4096	int
nettyServerBootstrapFactory (consumer (advanced))	使用自定义 NettyServerBootstrapFactory。		NettyServerBootstrapFactory
nettySharedHttpServer (consumer (advanced))	使用共享 Netty HTTP 服务器。如需了解更多详细信息，请参阅 Netty HTTP Server Example。		NettySharedHttpServer
noReplyLogLevel (consumer (advanced))	如果启用了同步，这个选项指定 NettyConsumer，在日志没有回复时使用该级别。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	WARN	LoggingLevel

Name	描述	默认值	类型
serverClosedChannelExceptionCaughtLogLevel (consumer (advanced))	如果服务器(NettyConsumer)捕获 java.nio.channels.ClosedChannelException，则其使用此日志记录级别记录。这用于避免记录关闭的频道异常，因为客户端可能会突然断开连接，然后在 Netty 服务器中造成大量关闭异常。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	DEBUG	LoggingLevel
serverExceptionCaughtLogLevel (consumer (advanced))	如果服务器(NettyConsumer)捕获异常，则使用此日志记录级别记录它。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	WARN	LoggingLevel
serverInitializerFactory (consumer (advanced))	使用自定义 ServerInitializerFactory。		ServerInitializerFactory
traceEnabled (consumer (advanced))	指定是否为此 Netty HTTP 使用者启用 HTTP TRACE。默认情况下关闭 TRACE。	false	布尔值
urlDecodeHeaders (consumer (advanced))	如果启用了这个选项，则在从 Netty 绑定到 Camel Message 时，标头值将被 URL 解码（例如 %20 将是一个空格字符。注意此选项供默认的 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，则需要将标头相应地解码到这个选项。	false	布尔值

Name	描述	默认值	类型
使用 ExecutorService (consumer (advanced))	是否使用排序的线程池，以确保在同一频道中按顺序处理事件。	true	布尔值
connectTimeout (producer)	等待套接字连接的时间。值以毫秒为单位。	10000	int
cookieHandler (producer)	配置 Cookie 处理程序来维护 HTTP 会话。		CookieHandler
requestTimeout (producer)	在调用远程服务器时，允许为 Netty producer 使用超时。默认情况下，不使用超时。该值以秒为单位，因此 eg 30000 为 30 秒。requestTimeout 使用 Netty 的 ReadTimeoutHandler 触发超时。		long
throwExceptionOnFailure (producer)	如果远程服务器失败响应，禁用禁用 HttpOperationFailedException 的选项。这样，无论 HTTP 状态代码是什么，您都可以获得所有响应。	true	布尔值
clientInitializerFactory (producer (advanced))	使用自定义 ClientInitializerFactory。		ClientInitializerFactory
lazyChannelCreation (producer (advanced))	如果远程服务器在 Camel 生成者启动时没有启动并运行，可以完全创建频道以避免异常。	true	布尔值
lazyStartProducer (producer (advanced))	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
okStatusCodeRange (producer (advanced))	被视为成功响应的状态代码。这些值包含为。可以定义多个范围，用逗号分开，如 200-204,209,301-304。每个范围都必须是一个数字或 from-to，包括横线。默认范围为 200-299。	200-299	字符串
producerPoolBlockWhenExhausted (producer (advanced))	设置 blockWhenExhausted 配置属性的值。它决定是否在池耗尽时是否调用 borrowObject （） 方法（达到了最大活跃对象数）。	true	布尔值

Name	描述	默认值	类型
producerPoolEnabled (producer (advanced))	生成者池是否启用。重要：如果您关闭此，则单个共享连接将用于生成者，您也可以执行 request/reply。这意味着，如果回复没有顺序，则交错响应可能存在潜在的问题。因此，您需要在请求和回复消息中有一个关联 id，以便您可以正确地将回复与负责继续处理 Camel 的消息的 Camel 回调相关联。为此，您需要实施 NettyCamelStateCorrelationManager 作为关联管理器，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
producerPoolMaxIdle (producer (advanced))	对池中空闲实例数量设置上限。	100	int
producerPoolMaxTotal (producer (advanced))	对池可以分配的对象数量（签出给客户端，或闲置一个给定时间等待签出）的对象数量设置上限。对没有限制使用负值。	-1	int
producerPoolMaxWait (producer (advanced))	当池耗尽，producerPoolBlockWhenExhausted 为 true 时，设置 borrowObject （）方法的最长持续时间（值为 millis）在抛出异常前应阻止。当小于 0 时，borrowObject （）方法可能会无限期地阻止。	-1	long
producerPoolMinEvictableIdle (producer (advanced))	在空闲对象驱除者有资格驱除前，对象设置在池中闲置的最小时间（值为 millis）。	300000	long
producerPoolMinIdle (producer (advanced))	在驱除器线程（如果活跃）生成新对象之前，设置制作者池中允许的最小实例数量。		int
useRelativePath (producer (advanced))	设置是否在 HTTP 请求中使用相对路径。	true	布尔值
hostnameVerification (security)	要在 SSLEngine 上启用/禁用主机名验证。	false	布尔值
allowSerializedHeaders (advanced)	仅在 transferExchange 为 true 时使用 TCP。当设置为 true 时，标头中的串行对象和属性将添加到交换中。否则 Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值
channelGroup (advanced)	使用明确的 ChannelGroup。		ChannelGroup

Name	描述	默认值	类型
configuration (advanced)	使用配置了 NettyHttpConfiguration 的自定义 NettyHttpConfiguration 来配置此端点。		NettyHttpConfiguration
disableStreamCache (advanced)	确定来自 Netty HttpRequest#getContent （）或 HttpResponset#getContent （）的原始输入流是否被缓存，或者没有缓存(Camel 将流读到基于轻量级内存流缓存中的流中。)缓存。默认情况下，Camel 将缓存 Netty 输入流来支持多次读取它，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，您可以将这个选项设置为 true，比如将其直接流传输到文件或其他持久性存储。请记住，如果您启用这个选项，则无法多次读取 Netty 流，您需要在 Netty 原始流上手动重置 reader 索引。另外，Netty 会在 Netty HTTP 服务器/HTTP 客户端完成处理时自动关闭 Netty 流，这意味着如果异步路由引擎正在使用异步路由，那么任何可能继续路由 org.apache.camel.Exchange 的异步线程可能无法读取 Netty 流，因为 Netty 已关闭。	false	布尔值
headerFilterStrategy (advanced)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 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，且仅在某些平台上被支持。您需要为您要使用的主机操作系统添加 netty JAR。请参阅更多详情：	false	布尔值
NettyHttpBinding (advanced)	使用自定义 org.apache.camel.component.netty.http.NettyHttpBinding 绑定到/从 Netty 和 Camel Message API 绑定。		NettyHttpBinding
选项 (advanced)	允许使用 option. 作为前缀配置额外的 netty 选项。例如，option.child.keepAlive=false 设置 netty 选项 child.keepAlive=false。有关使用的选项，请参阅 Netty 文档。		Map
receiveBufferSize (advanced)	在入站通信期间使用的 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器。请参阅 Jetty 文档以及此邮件线程的详细信息。		int
sendBufferSize (advanced)	在出站通信中使用的 TCP/UDP 缓冲区大小。大小为字节。	65536	int
同步 (advanced)	设置是否应严格使用同步处理。	false	布尔值

Name	描述	默认值	类型
transferException (advanced)	如果在消费者端启用并且 Exchange 失败，如果导致的 Exception 被发送序列化为 application/x-java-serialized-object 内容类型。在生成者一侧，异常将被反序列化和抛出，而不是 HttpOperationFailedException。导致异常需要被序列化。默认情况下是关闭的。如果启用此选项，则 Java 会将传入数据从请求反序列化到 Java，这可能会成为潜在的安全风险。	false	布尔值
transferExchange (advanced)	仅用于 TCP。您可以通过线路而不是只传输正文来传输交换。以下字段会被传输：在 body, Out body, fault body, In headers, Out headers, fault headers, exchange properties, exchange exception。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值
unixDomainSocketPath (advanced)	要使用的 unix 域套接字的路径，而不是 inet 套接字。但是，将不会使用主机和端口参数。为它们设置 dummy 值是 ok。必须与 nativeTransport=true 和 clientMode=false 一起使用。		字符串
workerCount (advanced)	当 netty 适用于 nio 模式时，它将使用来自 Netty 的默认 workerCount 参数（即 cpu_core_threads x 2）。用户可以使用此选项覆盖 Netty 的默认 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，每个消费者或制作者都有自己的 worker 池，具有 2 个 x cpu 数核心线程。		EventLoopGroup
decoders (codec)	要使用的解码器列表。您可以使用字符串，其值用逗号分开，并在 Registry 中查找值。只需记住，使用 # so Camel 知道它应该查找的值作为前缀。		字符串
encoders (codec)	要使用的编码程序列表。您可以使用字符串，其值用逗号分开，并在 Registry 中查找值。只需记住，使用 # so Camel 知道它应该查找的值作为前缀。		字符串
enabledProtocols (security)	使用 SSL 时要启用的协议。	TLSv1.2,TLSv1.3	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

Name	描述	默认值	类型
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 （安全）	要使用的密码设置来加密/解密使用 SSH 发送的有效负载。		字符串
securityConfiguration (security)	指的是 org.apache.camel.component.netty.http.NettyHttpSecurityConfiguration 来配置安全 Web 资源。		NettyHttpSecurityConfiguration
securityOptions (security)	使用来自映射的键/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 （安全性）	设置以指定 SSL 加密是否应用到此端点。	false	布尔值
sslClientCertHeaders (security)	启用和采用 SSL 模式时，Netty 使用者将增强 Camel 消息，其中包含有关客户端证书的信息，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)	对可用于返回 SSL 处理程序的类的引用。		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

97.6. 消息标头

Netty HTTP 组件支持 23 消息标头，如下所列：

Name	描述	默认	类型
CamelHttpAuthentication (common) 常数：HTTP_AUTHENTICATION	如果用户使用 HTTP Basic 进行身份验证，则使用值 Basic 添加此标头。		字符串
content-Type (common) 常数：CONTENT_TYPE	设置 HTTP 正文的内容类型：例如： text/plain; charset=UTF-8。		字符串
connection (common) 常数：CONNECTION	要使用的 HTTP 标头连接的值。		字符串
CamelNettyCloseChannelWhenComplete (common) 恒定：NETTY_CLOSE_CHANNEL_WHEN_COMPLETE	指明在完成后是否应关闭频道。		布尔值
CamelHttpResponseCode (common) 常量 ：HTTP_RESPONSE_CODE	允许设置要使用的 HTTP Status 代码。默认情况下，200 用于成功，500 代表失败。		整数
CamelHttpProtocolVersion (common) 常量 ：HTTP_PROTOCOL_VERSION	HTTP 协议的版本。	HTTP/1.1	字符串
CamelHttpMethod (common) 常数 ：HTTP_METHOD	使用的 HTTP 方法，如 GET、POST、TRACE 等。	GET	字符串

Name	描述	默认	类型
CamelHttpQuery (common) 常数 ：HTTP_QUERY	任何查询参数，如 foo=bar&beer=yes。		字符串
CamelHttpPath (common) 常数 ：HTTP_PATH	允许将 URI 上下文路径和查询参数作为覆盖端点配置的 String 值提供。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。		字符串
CamelHttpRawQuery (common) 常量：HTTP_RAW_QUERY	任何查询参数，如 foo=bar&beer=yes。存储在原始形式中，因为它们到达消费者（例如，在 URL 解码前）。		字符串
CamelHttpUrl (common) 常数 ：HTTP_URL	包括协议、主机和端口的 URL，等等：http://0.0.0.0:8080/myapp		字符串
CamelHttpCharacterEncoding (common) 常数 ：HTTP_CHARACTER_ENCODING	来自 content-type 标头的 charset。		字符串
CamelHttpUri (common) 常数：HTTP_URI	没有协议、主机和端口的 URI，等等：/myapp。		字符串
CamelNettyChannelHandlerContext (common) 恒定：NETTY_CHANNEL_HANDLER_CONTEXT	频道处理程序上下文。		ChannelHandlerContext

Name	描述	默认	类型
CamelNettyRemoteAddress (common) 常量 ：NETTY_REMOTE_ADDRESS	远程地址。		SocketAddress
CamelNettyLocalAddress (common) 常量 ：NETTY_LOCAL_ADDRESS	本地地址。		SocketAddress
CamelNettySSLSession (common) 常量 ：NETTY_SSL_SESSION	SSL 会话。		SSLSession
CamelNettySSLClientCertSubjectName (common) constant: NETTY_SSL_CLIENT_CERT_SUBJECT_NAME	SSL 客户端证书主题名称。		字符串
CamelNettySSLClientCertIssuerName (common) 常量 ：NETTY_SSL_CLIENT_CERT_ISSUER_NAME	SSL 客户端证书签发者名称。		字符串
CamelNettySSLClientCertSerialNumber (common) 常量 ：NETTY_SSL_CLIENT_CERT_SERIAL_NO	SSL 客户端证书序列号。		字符串

Name	描述	默认	类型
CamelNettySSLClientCertNotBefore (common) 常量 ：NETTY_SSL_CLIENT_CERT_NOT_BEFORE	之前的 SSL 客户端证书。		Date
CamelNettySSLClientCertNotAfter (common) 常量 ：NETTY_SSL_CLIENT_CERT_NOT_AFTER	不之后 SSL 客户端证书。		Date
CamelNettyRequestTimeout (common) 常量 ：NETTY_REQUEST_TIMEOUT	读取超时。		Long

97.7. 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。这允许最终用户访问原始 Netty 请求/响应实例，如下所示。请记住，原始响应可能无法随时访问。

```
io.netty.handler.codec.http.HttpRequest request =
exchange.getIn(NettyHttpMessage.class).getHttpRequest();
```

97.8. 例子

在以下路由中，我们使用 Netty HTTP 作为 HTTP 服务器，它会返回硬编码的"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，并使用 ProducerTemplate 调用，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);
```

我们返回"Bye World"作为输出。

97.8.1. 如何让 Netty 匹配通配符

默认情况下，Netty HTTP 仅匹配精确的 uri。但是，您可以指示 Netty 与前缀匹配。例如：

```
from("netty-http:http://0.0.0.0:8123/foo").to("mock:foo");
```

在以上 Netty HTTP 的路由中，只有 uri 是完全匹配，因此如果您输入 http://0.0.0.0:8123/foo 但没有匹配，它将匹配。http://0.0.0.0:8123/foo/bar

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");
```

现在，Netty 与任何以 foo 开头的端点匹配。

要匹配 任何 端点，您可以：

```
from("netty-http:http://0.0.0.0:8123?matchOnUriPrefix=true").to("mock:foo");
```

97.8.2. 使用具有相同端口的多个路由

在同一个 CamelContext 中，您可以有多个来自 Netty HTTP 的路由，它们共享相同的端口（如 io.netty.bootstrap.ServerBootstrap 实例）。这样做需要路由中的多个 bootstrap 选项相同，因为路由将共享相同的 io.netty.bootstrap.ServerBootstrap 实例。该实例将配置有第一个路由的选项。

路由的选项必须相同，这是 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，则 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。

下面是一个共享同一端口的两个路由的示例。

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 2 个路由示例，它没有与 1st 路由相同的 org.apache.camel.component.netty.NettyServerBootstrapConfiguration 选项。这将导致 Camel 在启动时失败。

共享同一端口的两个路由，但第 2 个路由配置错误

在启动时，和 将失败

```
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");
```

97.8.3. 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 类型的单一实例中配置 common server bootstrap 选项，我们可以在 Netty HTTP 用户中使用 bootstrapConfiguration 选项来引用并重复使用所有消费者中的相同选项。

```
<bean id="nettyHttpBootstrapOptions"
class="org.apache.camel.component.netty.NettyServerBootstrapConfiguration">
  <property name="backlog" value="200"/>
  <property name="connectionTimeout" value="20000"/>
  <property name="workerCount" value="16"/>
</bean>
```

在路由中，您可以引用这个选项，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>
```

97.8.4. 在 OSGi 容器中使用多个捆绑包间使用多个路由重复使用相同的服务器 bootstrap 配置

如需了解更多详细信息和示例，请参阅上述 Netty HTTP Server Example。

97.8.5. 实现反向代理

Netty HTTP 组件可以充当反向代理，在这种情况下，Exchange.HTTP_SCHEME、Exchange.HTTP_HOST 和 Exchange.HTTP_PORT 标头是从 HTTP 请求请求行上收到的绝对 URL 填充的。

以下是 HTTP 代理的示例，只是将响应从原始服务器转换为大写。

```
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));
}
```

97.9. 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>
```

域名称是启用基本身份验证所必需的。默认情况下，使用基于 JAAS 的验证器，它使用指定的 realm 名称（上例中的karaf），并使用 JAAS 域和这个域的 JAAS \\{{LoginModule}}s 进行身份验证。

Apache Karaf / ServiceMix 最终用户开箱即用有一个 karaf realm，因此上例将从这些容器开箱即用的原因。

97.9.1. 在 web 资源中指定 ACL

org.apache.camel.component.netty.http.SecurityConstraint 允许定义 web 资源的约束。org.apache.camel.component.netty.http.SecurityConstraintMapping 开箱即用，可轻松定义带有角色的包含和排除项。

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>
```

以上约束被定义，以便

- 对 **Attr 的访问会被限制，并接受任何角色（如果用户没有角色）**
- 访问 **/adminAttr 需要 admin 角色**
- 访问 **/guestAttr 需要 admin 或 guest 角色**
- 访问 **/publicAttr 是一个排除的，这意味着不需要身份验证，因此任何人都没有登录**

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>
```

97.10. SPRING BOOT AUTO-CONFIGURATION

组**件支持 67 个选项，如下所列。**

Name	描述	默认值	类型
camel.component .netty-http.allow-serialized-headers	仅在 transferExchange 为 true 时使用 TCP。当设置为 true 时，标头中的串行对象和属性将添加到交换中。否则 Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值
camel.component .netty-http.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component .netty-http.backlog	允许为 netty consumer (server)配置积压。请注意，后端只是根据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 backlog 依赖于 OS 设置。		整数

Name	描述	默认值	类型
camel.component.netty-http.boss-count	当 netty 适用于 nio 模式时，它会使用来自 Netty 的默认 bossCount 参数，即 1。用户可以使用此选项覆盖 Netty 的默认 bossCount。	1	整数
camel.component.netty-http.boss-group	设置 BossGroup，可用于处理 NettyEndpoint 中服务器端的新连接。选项是一个 io.netty.channel.EventLoopGroup 类型。		EventLoopGroup
camel.component.netty-http.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.netty-http.channel-group	使用明确的 ChannelGroup。选项是一个 io.netty.channel.group.ChannelGroup 类型。		ChannelGroup
camel.component.netty-http.client-initializer-factory	使用自定义 ClientInitializerFactory。选项是 org.apache.camel.component.netty.ClientInitializerFactory 类型。		ClientInitializerFactory
camel.component.netty-http.configuration	在创建端点时，使用 NettyConfiguration 作为配置。选项是 org.apache.camel.component.netty.NettyConfiguration 类型。		NettyConfiguration
camel.component.netty-http.connect-timeout	等待套接字连接的时间。值以毫秒为单位。	10000	整数
camel.component.netty-http.correlation-manager	使用自定义关联管理器来管理在使用带有 netty producer 的 request/reply 时如何映射请求和回复消息。只有当您有将请求与回复进行映射时（例如，请求和回复信息中存在关联 ID）时，才应使用此请求和回复。如果要在 netty 中在同一频道(aka connection)上有多个并发消息，则可以使用它。在这样做时，您必须有关联请求和回复消息的方法，以便在继续路由前，将正确的回复存储在 inflight Camel Exchange 上。我们建议在构建自定义关联管理器时扩展 TimeoutCorrelationManagerSupport。这提供了对超时和其他复杂度的支持，否则您需要实施其他复杂性。如需了解更多详细信息，请参阅 producerPoolEnabled 选项。选项是 org.apache.camel.component.netty.NettyCamelStateCorrelationManager 类型。		NettyCamelStateCorrelationManager

Name	描述	默认值	类型
camel.component.netty-http.decoders	要使用的解码器列表。您可以使用字符串，其值用逗号分开，并在 Registry 中查找值。只需记住，使用 # so Camel 知道它应该查找的值作为前缀。		字符串
camel.component.netty-http.disconnect	使用后是否从 Netty Channel 断开（关闭）可用于消费者和制作者。	false	布尔值
camel.component.netty-http.disconnect-on-no-reply	如果启用了同步，这个选项会指定 NettyConsumer（如果应该断开连接，但没有回复来回发）。	true	布尔值
camel.component.netty-http.enabled	是否启用 netty-http 组件的自动配置。这默认是启用的。		布尔值
camel.component.netty-http.enabled-protocols	使用 SSL 时要启用的协议。	TLSv1.2,TLSv1.3	字符串
camel.component.netty-http.encoders	要使用的编码程序列表。您可以使用字符串，其值用逗号分开，并在 Registry 中查找值。只需记住，使用 # so Camel 知道它应该查找的值作为前缀。		字符串
camel.component.netty-http.executor-service	使用给定的 EventExecutorGroup。选项是一个 io.netty.util.concurrent.EventExecutorGroup 类型。		EventExecutorGroup
camel.component.netty-http.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		HeaderFilterStrategy
camel.component.netty-http.hostname-verification	要在 SSLEngine 上启用/禁用主机名验证。	false	布尔值
camel.component.netty-http.keep-alive	设置 以确保套接字不会因为不活跃而关闭。	true	布尔值

Name	描述	默认值	类型
camel.component.netty-http.key-store-file	用于加密的客户端侧证书密钥存储。		File
camel.component.netty-http.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.netty-http.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-http.lazy-channel-creation	如果远程服务器在 Camel 生成者启动时没有启动并运行，可以完全创建频道以避免异常。	true	布尔值
camel.component.netty-http.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.netty-http.maximum-pool-size	为 netty 消费者排序线程池设置最大线程池大小。默认大小为 2 x cpu_core 加上 1。将此值设置为 eg 10 时将使用 10 个线程，除非 2 x cpu_core 加上 1 是一个更高的值，然后将覆盖并使用。例如，如果存在 8 个内核，则消费者线程池为 17。此线程池用于路由 Camel 从 Netty 接收的消息。我们使用单独的线程池来确保消息排序，如果某些消息将阻断，则 nettys worker 线程（事件循环）不受影响。		整数
camel.component.netty-http.mute-exception	如果对消费者启用并且交换失败，响应的正文不会包含异常的堆栈跟踪。	false	布尔值
camel.component.netty-http.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，且仅在某些平台上被支持。您需要为您要使用的主机操作系统添加 netty JAR。请参阅更多详情：	false	布尔值
camel.component.netty-http.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

Name	描述	默认值	类型
camel.component.netty-http.netty-http-binding	使用自定义 org.apache.camel.component.netty.http.NettyHttpBinding 绑定到/从 Netty 和 Camel Message API 绑定。选项是 org.apache.camel.component.netty.http.NettyHttpBinding 类型。		NettyHttpBinding
camel.component.netty-http.netty-server-bootstrap-factory	使用自定义 NettyServerBootstrapFactory。选项是 org.apache.camel.component.netty.NettyServerBootstrapFactory 类型。		NettyServerBootstrapFactory
camel.component.netty-http.no-reply-log-level	如果启用了同步，这个选项指定 NettyConsumer，在日志没有回复时使用该级别。		LoggingLevel
camel.component.netty-http.options	允许使用 option. 作为前缀配置额外的 netty 选项。例如，option.child.keepAlive=false 设置 netty 选项 child.keepAlive=false。有关使用的选项，请参阅 Netty 文档。		Map
camel.component.netty-http.passphrase	要使用的密码设置来加密/解密使用 SSH 发送的有效负载。		字符串
camel.component.netty-http.producer-pool-block-when-exhausted	设置 blockWhenExhausted 配置属性的值。它决定是否在池耗尽时是否调用 borrowObject （）方法（达到了最大活跃对象数）。	true	布尔值
camel.component.netty-http.producer-pool-enabled	生成者池是否启用。重要：如果您关闭此，则单个共享连接将用于生成者，您也可以执行 request/reply。这意味着，如果回复没有顺序，则交错响应可能存在潜在的问题。因此，您需要在请求和回复消息中有一个关联 id，以便您可以正确地将回复与负责继续处理 Camel 的消息的 Camel 回调相关联。为此，您需要实施 NettyCamelStateCorrelationManager 作为关联管理器，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.netty-http.producer-pool-max-idle	对池中空闲实例数量设置上限。	100	整数

Name	描述	默认值	类型
camel.component.netty-http.producer-pool-max-total	对池可以分配的对象数量（签出给客户端，或闲置一个给定时间等待签出）的对象数量设置上限。对没有限制使用负值。	-1	整数
camel.component.netty-http.producer-pool-max-wait	当池耗尽，producerPoolBlockWhenExhausted 为 true 时，设置 borrowObject （）方法的最长持续时间（值为 millis）在抛出异常前应阻止。当小于 0 时，borrowObject （）方法可能会无限期地阻止。	-1	Long
camel.component.netty-http.producer-pool-min-evictable-idle	在空闲对象驱除者有资格驱除前，对象设置在池中闲置的最小时间（值为 millis）。	300000	Long
camel.component.netty-http.producer-pool-min-idle	在驱除器线程（如果活跃）生成新对象之前，设置制作者池中允许的最小实例数量。		整数
camel.component.netty-http.receive-buffer-size	在入站通信期间使用的 TCP/UDP 缓冲区大小。大小为字节。	65536	整数
camel.component.netty-http.receive-buffer-size-predictor	配置缓冲区大小预测器。请参阅 Jetty 文档以及此邮件线程的详细信息。		整数
camel.component.netty-http.request-timeout	在调用远程服务器时，允许为 Netty producer 使用超时。默认情况下，不使用超时。该值以秒为单位，因此 eg 30000 为 30 秒。requestTimeout 使用 Netty 的 ReadTimeoutHandler 触发超时。		Long
camel.component.netty-http.reuse-address	将 设置为方便套接字多路。	true	布尔值

Name	描述	默认值	类型
camel.component.netty-http.reuse-channel	此选项允许生产者和消费者（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 频道。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，通道在 Exchange 完成后不会返回到连接池；如果 disconnect 选项设为 true，则不会断开连接。重复使用的频道作为交换属性存储在 Exchange 上，其键为 NettyConstants#NETTY_CHANNEL，允许您在路由过程中获取频道，并使用它。	false	布尔值
camel.component.netty-http.security-configuration	指的是 org.apache.camel.component.netty.http.NettyHttpSecurityConfiguration 来配置安全 Web 资源。选项是 org.apache.camel.component.netty.http.NettyHttpSecurityConfiguration 类型。		NettyHttpSecurityConfiguration
camel.component.netty-http.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.netty-http.send-buffer-size	在出站通信中使用的 TCP/UDP 缓冲区大小。大小为字节。	65536	整数
camel.component.netty-http.server-closed-channel-exception-caught-log-level	如果服务器(NettyConsumer)捕获 java.nio.channels.ClosedChannelException，则其使用此日志记录级别记录。这用于避免记录关闭的频道异常，因为客户端可能会突然断开连接，然后在 Netty 服务器中造成大量关闭异常。		LoggingLevel
camel.component.netty-http.server-exception-caught-log-level	如果服务器(NettyConsumer)捕获异常，则使用此日志记录级别记录它。		LoggingLevel
camel.component.netty-http.server-initializer-factory	使用自定义 ServerInitializerFactory。选项是 org.apache.camel.component.netty.ServerInitializerFactory 类型。		ServerInitializerFactory
camel.component.netty-http.ssl	设置以指定 SSL 加密是否应用到此端点。	false	布尔值

Name	描述	默认值	类型
camel.component.netty-http.ssl-client-cert-headers	启用和采用 SSL 模式时，Netty 使用者将增强 Camel 消息，其中包含有关客户端证书的信息，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-http.ssl-context-parameters	使用 SSLContextParameters 配置安全性。选项是 org.apache.camel.support.jsse.SSLContextParameters 类型。		SSLContextParameters
camel.component.netty-http.ssl-handler	对可用于返回 SSL 处理程序的类的引用。选项是 io.netty.handler.ssl.SslHandler 类型。		SslHandler
camel.component.netty-http.sync	设置，将端点设置为单向或请求响应。	true	布尔值
camel.component.netty-http.tcp-no-delay	设置 以提高 TCP 协议性能。	true	布尔值
camel.component.netty-http.transfer-exchange	仅用于 TCP。您可以通过线路而不是只传输正文来传输交换。以下字段会被传输：在 body, Out body, fault body, In headers, Out headers, fault headers, exchange properties, exchange exception。这要求对象是可序列化的。Camel 将排除任何非可序列化对象，并将其记录在 WARN 级别。	false	布尔值
camel.component.netty-http.trust-store-file	用于加密的服务器端证书密钥存储。		File
camel.component.netty-http.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-http.unix-domain-socket-path	要使用的 unix 域套接字的路径，而不是 inet 套接字。但是，将不会使用主机和端口参数。为它们设置 dummy 值是 ok。必须与 nativeTransport=true 和 clientMode=false 一起使用。		字符串
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

Name	描述	默认值	类型
camel.component.netty-http.using-executor-service	是否使用排序的线程池，以确保在同一频道中按顺序处理事件。	true	布尔值
camel.component.netty-http.worker-count	当 netty 适用于 nio 模式时，它将使用来自 Netty 的默认 workerCount 参数（即 cpu_core_threads x 2）。用户可以使用此选项覆盖 Netty 的默认 workerCount。		整数
camel.component.netty-http.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，每个消费者或制作者都有自己的 worker 池，具有 2 个 x cpu 数核心线程。选项是一个 io.netty.channel.EventLoopGroup 类型。		EventLoopGroup

第 98 章 OLINGO4

从 Camel 2.19 开始

支持生成者和消费者

Olingo4 组件使用 Apache Olingo 版本 4.0 API 与 OData 4.0 兼容服务交互。自版本 4.0 起，OData 是 OASIS 标准，许多流行的开源和商业供应商和产品支持此协议。可以在 OData 网站 中找到支持产品的示例列表。

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。

98.1. 依赖项

当在 Camel Spring Boot 中使用 camel-olingo4 时，请将以下 Maven 依赖项添加到 pom.xml 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-olingo4-starter</artifactId>
</dependency>
```

98.2. URI 格式

```
olingo4://endpoint/<resource-path>?[options]
```

98.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别

- 端点级别

98.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

98.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

98.4. 组件选项

Olingo4 组件支持 16 个选项，如下所列。

Name	描述	默认值	类型
configuration (common)	使用共享配置。		Olingo4Configuration

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 标头值可以用来指定 JSON 或 XML 消息格式，默认为 application/json;charset=utf-8。	application/json;charset=utf-8	字符串
filterAlreadySeen (common)	把它设置为 true，以过滤已由此组件通信的结果。	false	布尔值
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
proxy (common)	HTTP 代理服务器配置。		HttpHost
serviceUri (common)	目标 OData 服务基本 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
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 Camel 消费者试图获取传入的消息或类似信息时，会出现异常（如果可能），现在将被作为消息进行处理，并由路由 Error Handler 处理。重要：只有在第三方组件允许 Camel 抛出异常时，才能警报这一点。有些组件仅在内部处理，因此无法 bridgeErrorHandler。在其他情况下，我们可能会将 Camel 组件提高到第三方组件中的 hook，并使其可能用于将来的版本。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
splitResult (consumer)	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 splitResult 被设置为 false。	true	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

Name	描述	默认值	类型
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
httpAsyncClientBuilder (advanced)	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 socketTimeout MUST，否则 OData 请求可能会无限期阻止。		HttpAsyncClientBuilder
httpClientBuilder (advanced)	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 socketTimeout MUST，否则 OData 请求可能会无限期阻止。		HttpClientBuilder
sslContextParameters (security)	使用 SSLContextParameters 配置安全性。		SSLContextParameters
useGlobalSslContextParameters (security)	启用使用全局 SSL 上下文参数。	false	布尔值

98.5. 端点选项

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 ：

98.5.1. 路径参数(2 参数)

Name	描述	默认	类型
apiName (common)	**需要** 执行什么操作。 Enum 值： • DEFAULT		Olingo4ApiName

Name	描述	默认	类型
methodName (common)	**必需的**所选操作需要哪些子操作。		字符串

98.5.2. 查询参数(32 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 标头值可以用来指定 JSON 或 XML 消息格式，默认为 application/json;charset=utf-8。	application/json;charset=utf-8	字符串
filterAlreadySeen (common)	把它设置为 true，以过滤已由此组件通信的结果。	false	布尔值
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 服务基本 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
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
splitResult (consumer)	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 splitResult 被设置为 false。	true	布尔值

Name	描述	默认值	类型
bridgeErrorHandler (consumer (advanced))	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当 Camel 消费者试图获取传入的消息或类似信息时，会出现异常（如果可能），现在将被作为消息进行处理，并由路由 Error Handler 处理。重要：只有在第三方组件允许 Camel 抛出异常时，才能警报这一点。有些组件仅在内部处理，因此无法 bridgeErrorHandler。在其他情况下，我们可能会将 Camel 组件提高到第三方组件中的 hook，并使其可能用于将来的版本。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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： ● InOnly ● InOut		ExchangePattern
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
lazyStartProducer (producer (advanced))	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
httpAsyncClientBuilder (advanced)	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 socketTimeout MUST，否则 OData 请求可能会无限期阻止。		HttpAsyncClientBuilder
httpClientBuilder (advanced)	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 socketTimeout MUST，否则 OData 请求可能会无限期阻止。		HttpClientBuilder
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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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
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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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
sslContextParameters (security)	使用 SSLContextParameters 配置安全性。		SSLContextParameters

98.6. API 参数(1 API)

Olingo4 端点是一个基于 API 的组件，它根据使用 API 名称和 API 方法具有额外的参数。API 名称和 API 方法位于端点 URI 中，作为 apiName/methodName 路径参数：

```
olingo4:apiName/methodName
```

下表列出 1 API 名称：

API 名称	类型	描述
API:DEFAULT	两者都	Olingo4 Client Api Interface

每个 API 记录在以下部分中。

98.6.1. API: DEFAULT

支持生成者和消费者

DEFAULT API 在语法中定义，如下所示：

> olingo4:DEFAULT/methodName?[parameters]

下表中列出了 9 种方法，后跟每种方法的详细语法。(API 方法可以有一个简写 别名名称，可用于语法而不是名称)

æ–¹æ³•	Alias	描述
action		调用 OData 操作
batch		执行批处理请求
create		创建新的 OData 资源
delete		删除 OData 资源，并使用机构调用回调
merge		使用 HTTP MERGE 补丁/发布 OData 资源
patch		使用 HTTP PATCH 的补丁/发布 OData 资源
读取		读取 OData 资源，并使用适当的结果调用回调
update		更新 OData 资源
uread		读取 OData 资源，并使用未解析的输入流调用回调

98.6.1.1. 方法操作

signatures:

- void action (org.apache.olingo.commons.api.edm.Edm edm, String resourcePath, java.util.Map<String, String> endpointHttpHeaders, Object data, org.apache.camel.component.olingo4.api.Olingo4ResponseHandler responseHandler);

olingo4/action API 方法在下表中列出的参数：

参数	描述	类型
data	操作数据	对象
eDM	Service Edm	eDM
endpointHttpHeaders	用于添加/覆盖组件版本的 HTTP 标头	Map
resourcePath	操作的资源路径	字符串
responseHandler	org.apache.olingo.client.api.domain.ClientEntity 回调处理程序	Olingo4ResponseHandler

98.6.1.2. 方法批处理

signatures:

- *void batch （org.apache.olingo.commons.api.edm.Edm edm, java.util.Map<String, String> endpointHttpHeaders, Object data, org.apache.camel.component.olingo4.api.Olingo4ResponseHandler<java.util.List<org.apache.camel.component.olingo4.api.batch.Olingo4BatchResponse>Handler;*

olingo4/batch API 方法在下表中列出的参数：

参数	描述	类型
data	排序 org.apache.camel.component.olingo4.api.batch.Olingo4BatchRequest 列表	对象
eDM	Service Edm	eDM
endpointHttpHeaders	用于添加/覆盖组件版本的 HTTP 标头	Map
responseHandler	回调处理器	Olingo4ResponseHandler

98.6.1.3. 方法创建

signatures:

- ***void create (org.apache.olingo.commons.api.edm.Edm edm, String resourcePath, java.util.Map<String, String> endpointHttpHeaders, Object data, org.apache.camel.component.olingo4.api.Olingo4ResponseHandler responseHandler);***

olingo4/create API 方法在下表中列出的参数：

参数	描述	类型
data	请求数据	对象
eDM	Service Edm	eDM
endpointHttpHeaders	用于添加/覆盖组件版本的 HTTP 标头	Map
resourcePath	要创建的资源路径	字符串
responseHandler	回调处理器	Olingo4ResponseHandler

98.6.1.4. 方法删除

signatures:

- ***void delete (String resourcePath, java.util.Map<String, String> endpointHttpHeaders, org.apache.camel.component.olingo4.api.Olingo4ResponseHandler<org.apache.olingo.commons.api.http.HttpStatusCode> responseHandler);***

olingo4/delete API 方法在下表中列出的参数：

参数	描述	类型
endpointHttpHeaders	用于添加/覆盖组件版本的 HTTP 标头	Map
resourcePath	条目的资源路径	字符串
responseHandler	org.apache.olingo.commons.api.http.HttpStatusCode 回调处理程序	Olingo4ResponseHandler

98.6.1.5. 方法合并

signatures:

- ***void merge (org.apache.olingo.commons.api.edm.Edm edm, String resourcePath, java.util.Map<String, String> endpointHttpHeaders, Object data, org.apache.camel.component.olingo4.api.Olingo4ResponseHandler responseHandler);***

olingo4/merge API 方法在下表中列出的参数：

参数	描述	类型
data	patch/merge 数据	对象
eDM	Service Edm	eDM
endpointHttpHeaders	用于添加/覆盖组件版本的 HTTP 标头	Map
resourcePath	更新的资源路径	字符串
responseHandler	org.apache.olingo.client.api.domain.ClientEntity 回调处理程序	Olingo4ResponseHandler

98.6.1.6. 方法补丁

signatures:

- ***void patch (org.apache.olingo.commons.api.edm.Edm edm, String resourcePath, java.util.Map<String, String> endpointHttpHeaders, Object data, org.apache.camel.component.olingo4.api.Olingo4ResponseHandler responseHandler);***

olingo4/patch API 方法在下表中列出的参数：

参数	描述	类型
data	patch/merge 数据	对象
eDM	Service Edm	eDM

参数	描述	类型
endpointHttpHeaders	用于添加/覆盖组件版本的 HTTP 标头	Map
resourcePath	更新的资源路径	字符串
responseHandler	org.apache.olingo.client.api.domain.ClientEntity 回调处理程序	Olingo4ResponseHandler

98.6.1.7. 方法 读取

signatures:

- ***void read (org.apache.olingo.commons.api.edm.Edm edm, String resourcePath, java.util.Map<String, String> queryParams, java.util.Map<String, String> endpointHttpHeaders, org.apache.camel.component.olingo4.api.Olingo4.api.Olingo4ResponseHandler responseHandler);***

olingo4/read API 方法在下表中列出的参数：

参数	描述	类型
eDM	服务 Edm，从调用 read (null, $metdata, null, responseHandler)读取	eDM
endpointHttpHeaders	用于添加/覆盖组件版本的 HTTP 标头	Map
queryParams	OData 查询参数 http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-protocol.html#_Toc453752288	Map
resourcePath	OData 资源路径	字符串
responseHandler	回调处理器	Olingo4ResponseHandler

98.6.1.8. 方法更新

signatures:

-

void update (org.apache.olingo.commons.api.edm.Edm edm, String resourcePath, java.util.Map<String, String> endpointHttpHeaders, Object data, org.apache.camel.component.olingo4.api.Olingo4ResponseHandler responseHandler);

olingo4/update API 方法在下表中列出的参数：

参数	描述	类型
data	更新的数据	对象
eDM	Service Edm	eDM
endpointHttpHeaders	用于添加/覆盖组件版本的 HTTP 标头	Map
resourcePath	更新的资源路径	字符串
responseHandler	org.apache.olingo.client.api.domain.ClientEntity 回调处理程序	Olingo4ResponseHandler

98.6.1.9. 方法 uread

signatures:

- ***void uread (org.apache.olingo.commons.api.edm.Edm edm, String resourcePath, java.util.Map<String, String> queryParams, java.util.Map<String, String> endpointHttpHeaders, org.apache.camel.component.olingo4.api.Olingo4ResponseHandler<java.io.InputStream> responseHandler);***

olingo4/uread API 方法在下表中列出的参数：

参数	描述	类型
eDM	服务 Edm，从调用 read (null, $metdata, null, responseHandler)读取	eDM
endpointHttpHeaders	用于添加/覆盖组件版本的 HTTP 标头	Map
queryParams	OData 查询参数 http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-protocol.html#_Toc453752288	Map

参数	描述	类型
resourcePath	OData 资源路径	字符串
responseHandler	回调处理器	Olingo4ResponseHandler

除了上述参数外，olingo4 API 还可以使用上面列出的任何 Query 参数。

任何参数可以在端点 URI 中提供，也可以在消息标头中动态提供。消息标头名称必须是 CamelOlingo4.parameter 的格式。inBody 参数覆盖消息标头，即 endpoint 参数 inBody=myParameterNameHere 覆盖 CamelOlingo4.myParameterNameHere 标头。

98.7. 消息标头

Olingo4 组件支持 1 个消息标头，如下所列：

Name	描述	默认值	类型
CamelOlingo4.responseHttpHeaders (producer) 常量 ：FULL_RESPONSE_HTTP_HEADERS	响应 Http 标头。		Map

98.8. 端点 HTTP 标头

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>。

98.9. 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	$metadata	**org.apache.olingo.commons.api.edm.Edm**
服务文档	/	**org.apache.olingo.client.api.domain.ClientServiceDocument**
OData 实体集	<entity-set>	**org.apache.olingo.client.api.domain.ClientEntitySet**
OData 实体	<entity-set>(<key-predicate>)	**org.apache.olingo.client.api.domain.ClientEntity** for Out body (response)**java.util.Map<String, Object&gt**; 用于 In body (request)
simple 属性	<entity-set>(<key-predicate>)/<simple-property>	**org.apache.olingo.client.api.domain.ClientPrimitiveValue**
简单属性值	<entity-set>(<key-predicate>)/<simple-property>/$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**

98.10. SAMPLES

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");
```

以下路由使用正文消息中的 ClientEntity 调用 odata 操作。对于没有期望输入的操作，正文消息可能为 null。

```
from("direct:...")
    .to("olingo4://action/People");
```

98.11. 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.olingo4.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.olingo4.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 Camel 消费者试图获取传入的消息或类似信息时，会出现异常（如果可能），现在将被作为消息进行处理，并由路由 Error Handler 处理。重要：只有在第三方组件允许 Camel 抛出异常时，才能警报这一点。有些组件仅在内部处理，因此无法 bridgeErrorHandler。在其他情况下，我们可能会将 Camel 组件提高到第三方组件中的 hook，并使其可能用于将来的版本。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.olingo4.configuration	使用共享配置。选项是 org.apache.camel.component.olingo4.Olingo4Configuration 类型。		Olingo4Configuration
camel.component.olingo4.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

Name	描述	默认值	类型
camel.component.olingo4.content-type	content-Type 标头值可以用来指定 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.enabled	是否启用 olingo4 组件的自动配置。这默认是启用的。		布尔值
camel.component.olingo4.filter-already-seen	把它设置为 true，以过滤已由此组件通信的结果。	false	布尔值
camel.component.olingo4.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 socketTimeout MUST，否则 OData 请求可能会无限期阻止。选项是 org.apache.http.impl.nio.client.HttpAsyncClientBuilder 类型。		HttpAsyncClientBuilder
camel.component.olingo4.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 socketTimeout MUST，否则 OData 请求可能会无限期阻止。选项是 org.apache.http.impl.client.HttpClientBuilder 类型。		HttpClientBuilder
camel.component.olingo4.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.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.olingo4.proxy	HTTP 代理服务器配置。选项是 org.apache.http.HttpHost 类型。		HttpHost
camel.component.olingo4.service-uri	目标 OData 服务基础 URI，例如。		字符串

Name	描述	默认值	类型
camel.component.olingo4.socket-timeout	HTTP 请求超时（以毫秒为单位），默认为 30,000 (30 秒)。	30000	整数
camel.component.olingo4.split-result	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 splitResult 被设置为 false。	true	布尔值
camel.component.olingo4.ssl-context-parameters	使用 SSLContextParameters 配置安全性。选项是 org.apache.camel.support.jsse.SSLContextParameters 类型。		SSLContextParameters
camel.component.olingo4.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值

第 99 章 OPENAPI JAVA

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。

camel-openapi-java 模块可以从 REST 组件使用（不需要 servlet）。

99.1. 依赖项

当在 Camel Spring Boot 中使用 openapi-java 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-openapi-java-starter</artifactId>
</dependency>
```

99.2. 在 REST-DSL 中使用 OPENAPI

您可以通过配置 apiContextPath dsl 从 rest-dsl 启用 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");
    }
}
```

99.3. 选项

OpenApi 模块可以使用以下选项进行配置。要使用 servlet 配置，您可以使用前面所示的 init-param。直接在 rest-dsl 中配置时，您可以使用适当的方法，如 enableCORS、host、contextPath、dsl。api.xxx 的选项使用 apiProperty dsl 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，而不是对 REST 服务的实际访问。默认为 false。
openapi.version	字符串	OpenAPI spec 版本。是默认的 3.0。
主机	字符串	设置主机名的步骤：如果没有配置 camel-openapi-java，则将根据 localhost 来计算名称。
scheme	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** ：要设置 REST 服务可用的基本路径。该路径是 relative （不以 http/https 开始）和 camel-openapi-java 将在运行时计算绝对路径，这将是 **protocol://host:port/context-path/base.path**
api.path	字符串	要设置 API 可用的路径（如 /api-docs）。该路径是相对的（不以 http/https 开始）和 camel-openapi-java 将在运行时计算绝对路径，这将是 **protocol://host:port/context-path/api.path** So 使用相对路径更为简单。请参阅上面的示例。
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 服务条款的 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 API 相关关系的电子邮件。

选项	类型	描述
api.contact.url	字符串	网站 URL 以获取更多信息。
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。

99.4. 在 API 文档中添加安全定义

Rest DSL 现在支持在生成的 API 文档中声明 OpenApi securityDefinitions。例如，如下所示：

```
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")
```

在这里，我们设置了两个安全定义

- OAuth2 - 带有通过提供的 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)来指定使用其他操作。

```
.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 Api Key 安全性，put 操作使用带有允许的读取和写入片断范围的 OAuth 安全性。

99.5. JSON 或 YAML

camel-openapi-java 模块支持开箱即用 JSon 和 Yaml。您可以在 request url 中指定您想要的 /openapi.json 或 /openapi.yaml。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。如果接受或没有设置为接受，则 json 将返回默认格式。

99.6. USEXFORWARDHEADERS 和 API URL 解析

OpenApi 规格允许您指定提供 API 的主机、端口和路径。在 OpenApi V2 中，这通过 host 字段完成，并在 OpenAPI V3 中是 servers 字段的一部分。

默认情况下，这些字段的值由 X-Forwarded 标头 X-Forwarded-Host 和 X-Forwarded-Proto 决定。

这可以通过禁用 X-Forwarded 标头的查找并在 REST 配置中指定您自己的主机、端口和方案来覆盖。

```
restConfiguration().component("netty-http")
    .useXForwardHeaders(false)
    .apiProperty("schemes", "https");
    .host("localhost")
    .port(8080);
```

99.7. 例子

在 Apache Camel 发行版本中，我们提供 camel-example-openapi-cdi 和 camel-example-spring-boot-rest-openapi-simple，它演示了如何使用这个 OpenApi 组件。

99.8. SPRING BOOT AUTO-CONFIGURATION

组件支持 1 个选项，如下所列。

Name	描述	默认值	类型
camel.openapi.enabled	在 Spring Boot 中启用 Camel Rest DSL 自动注册其 OpenAPI （如 swagger doc），它允许 SpringDoc 等工具与 Camel 集成。	true	布尔值

第 100 章 OPENTELEMETRY

从 Camel 3.5 开始

OpenTelemetry 组件用于跟踪和使用 OpenTelemetry 的传入和传出 Camel 消息。

为发送到/来自 Camel 的传入和传出消息捕获事件(spans)。

100.1. 依赖项

将此组件的 pom.xml 添加以下依赖项：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-opentelemetry-starter</artifactId>
</dependency>
```

100.2. 配置

OpenTelemetry tracer 的配置属性是：

选项	默认	描述
excludePatterns		设置将禁用与模式匹配 Camel 消息的追踪的 exclude 模式。内容是一个 Set<String>，其中键是一个模式。模式使用来自 Intercept 的规则。
编码	false	设置标头键是否需要编码（特定于connector）。该值是一个布尔值。短划线需要为 JMS 属性键编码实例。

100.2.1. Configuration

除了与所选 OpenTelemetry 兼容 Tracer 关联的任何特定依赖项外，在 POM 中添加 camel-opentelemetry 组件。

要显式配置 OpenTelemetry 支持，请实例化 OpenTelemetryTracer 并初始化 camel 上下文。您可以选择指定一个 Tracer，也可以使用 Registry来隐式发现

```
OpenTelemetryTracer otelTracer = new OpenTelemetryTracer();
// By default it uses the DefaultTracer, but you can override it with a specific OpenTelemetry
Tracer implementation.
otelTracer.setTracer(...);
// And then initialize the context
otelTracer.init(camelContext);
```

100.3. SPRING BOOT

添加 camel-opentelemetry-starter 依赖项，然后通过使用 @CamelOpenTelemetry 注解主类来打开 OpenTracing。

OpenTelemetryTracer 从 camel 上下文的 Registry 中隐式获取，除非应用程序定义了 OpenTelemetryTracer bean。

100.4. JAVA 代理

下载 Java 代理。

这个软件包包括检测代理，以及所有支持的库和所有可用数据导出器的工具。包提供了完全自动的、开箱即用的体验。

在 JVM 中使用 -javaagent 标志启用检测代理。

```
java -javaagent:path/to/opentelemetry-javaagent.jar \
     -jar myapp.jar
```

默认情况下，OpenTelemetry Java 代理使用 OTLP exporter 将数据发送到 OpenTelemetry 收集器，地址为 http://localhost:4317。

配置参数作为 Java 系统属性(-D 标志)或环境变量传递。有关配置项目的完整列表，请参阅配置代理和 OpenTelemetry 自动配置。例如：

```
java -javaagent:path/to/opentelemetry-javaagent.jar \
     -Dotel.service.name=your-service-name \
     -Dotel.traces.exporter=jaeger \
     -jar myapp.jar
```

100.5. 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.opentelemetry.encoding	激活或取消激活标头中的横线编码(JMS 需要)用于消息传递。		布尔值
camel.opentelemetry.exclude-patterns	设置将禁用与模式匹配 Camel 消息的追踪的 exclude 模式。		Set

100.6. MDC LOGGING

当为活跃 Camel 上下文启用 MDC Logging 时，会为每个路由添加和删除 Trace ID 和 Span ID，其中键分别是 trace_id 和 span_id。

第 101 章 PAHO

支持生成者和消费者

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，因此如果您想将其与 Java 项目集成 - Camel Paho 连接器是一个要前往的方法。

101.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 paho 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-paho-starter</artifactId>
</dependency>
```

101.2. URI 格式

```
paho:topic[?options]
```

其中 topic 是主题的名称。

101.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

101.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

101.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

101.4. 组件选项

Paho 组件支持 31 个选项，如下所列。

Name	描述	默认值	类型
automaticReconnect (common)	设置在连接丢失时客户端是否自动尝试重新连接到服务器。如果设置为 false，则在连接丢失时客户端不会尝试自动重新连接到服务器。如果设置为 true，如果连接丢失，客户端将尝试重新连接到服务器。它初始会在尝试重新连接前等待 1 秒，对于每个失败的尝试，其延长会加倍，指定 2 分钟为止，此时，延迟会一直为 2 分钟。	true	布尔值
brokerUrl (common)	MQTT 代理的 URL。	tcp://localhost:1883	字符串

Name	描述	默认值	类型
cleanSession (common)	设置客户端和服务器在重新启动和重新连接后是否应记住状态。如果设置为 false，则客户端和服务器在重启客户端、服务器和连接后将维持状态。由于状态被维护：消息交付将可靠满足指定的 QOS，即使客户端、服务器或连接被重启也是如此。服务器会将订阅视为持久。如果设置为 true，则客户端和服务器在重启客户端、服务器或连接时不会维护状态。这意味着，如果客户端、服务器或连接被重启，则消息发送到指定的 QOS 无法被维护。服务器会将订阅视为非持久性。	true	布尔值
clientId (common)	MQTT 客户端标识符。标识符必须是唯一的。		字符串
configuration (common)	使用共享 Paho 配置。		PahoConfiguration
connectionTimeout (common)	设置连接超时值。这个值（以秒为单位）定义客户端等待 MQTT 服务器的网络连接的最大时间间隔（以秒为单位）。默认超时为 30 秒。值 0 禁用超时处理意味着客户端将等待到网络连接成功或失败。	30	int
filePersistenceDirectory (common)	文件持久性使用的基础目录。默认情况下，将使用 user 目录。		字符串
keepAliveInterval (common)	设置 keep alive 间隔。这个值（以秒为单位）定义发送或接收的消息之间的最大时间间隔。它使客户端能够检测服务器是否不再可用，而无需等待 TCP/IP 超时。客户端将确保每个保持活跃期间内至少有一个消息在网络间传输。如果在一段时间内没有与数据相关的消息，客户端会发送一个非常小的 ping 消息，后者将确认服务器。值 0 可禁用客户端中的 keepalive 处理。默认值为 60 秒。	60	int
maxInflight (common)	设置 max inflight。请在高流量环境中增加这个值。默认值为 10。	10	int
maxReconnectDelay (common)	获取重新连接之间等待的最长时间（以 millis）。	128000	int
mqttVersion (common)	设置 MQTT 版本。默认操作是与版本 3.1.1 连接，如果失败，则回退到 3.1。可以使用 MQTT_VERSION_3_1_1 或 MQTT_VERSION_3_1 选项，专门选择版本 3.1.1 或 3.1，且无回退。		int

Name	描述	默认值	类型
Persistence (common)	要使用的客户端持久性 - 内存或文件。 Enum 值： • FILE • MEMORY	MEMORY	PahoPersistence
QoS (common)	客户端服务质量级别(0-2)。	2	int
reserved (common)	Retain 选项。	false	布尔值
serverURIs (common)	设置客户端可以连接到的一个或多个 serverURI 的列表。可以使用逗号分隔多个服务器。每个 serverURI 指定客户端可以连接的服务器的地址。对于 TCP 连接，支持两种类型的连接，ssl:// 用于由 SSL/TLS 安全的 TCP 连接。例如： tcp://localhost:1883 ssl://localhost:8883 如果未指定端口，则为 tcp:// URIs 默认为 1883，对于 ssl:// URIs，8883 为 8883。如果设置了 serverURIs，则它将覆盖 MQTT 客户端构造器上传递的 serverURI 参数。当尝试连接时，客户端将从列表中的第一个 serverURI 开始，并操作列表，直到与服务器建立连接为止。如果无法对任何服务器建立连接，则连接尝试会失败。指定客户端可以连接到多个用途的服务器列表：高可用性和可靠的消息交付一些 MQTT 服务器支持高可用性功能，其中两个或更多 MQTT 服务器共享状态。MQTT 客户端可以连接到任何等同的服务器，并保证无论客户端连接到哪个服务器，都可靠交付和持久化订阅。如果需要 durable 订阅和/或可靠的消息交付，则必须将 cleansession 标志设置为 false。可以指定一组不相等的服务器集合（如高可用性选项中）。因为服务器之间没有共享状态，因此消息交付和持久化订阅无效。如果使用了 hunt 列表模式，则必须将 cleansession 标志设置为 true。		字符串
willPayload (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。要发布到消息的字节有效负载的主题。发布消息的服务质量(0、1 或 2)。消息是否应保留。		字符串
willQos (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。要发布到消息的字节有效负载的主题。发布消息的服务质量(0、1 或 2)。消息是否应保留。		int

Name	描述	默认值	类型
willRetained (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。要发布到消息的字节有效负载的主题。发布消息的服务质量(0、1 或 2)。消息是否应保留。	false	布尔值
willTopic (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。要发布到消息的字节有效负载的主题。发布消息的服务质量(0、1 或 2)。消息是否应保留。		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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
client (advanced)	使用共享 Paho 客户端。		MqttClient
customWebSocketHeaders (advanced)	为 WebSocket 连接设置自定义 WebSocket 标头。		Properties
executorServiceTimeout (advanced)	设置 executor 服务在强制终止前应等待的时间（以秒为单位）。除非绝对确定您需要修改这个值，否则不建议更改这个值。	1	int
httpsHostnameVerificationEnabled (security)	是否启用 SSL HostnameVerifier。默认值为 true。	true	布尔值

Name	描述	默认值	类型
password (security)	用于针对 MQTT 代理进行身份验证的密码。		字符串
socketFactory (security)	设置要使用的 SocketFactory。这允许应用程序在创建网络套接字时应用自己的策略。如果使用 SSL 连接，可以使用 SSLSocketFactory 提供应用特定的安全设置。		SocketFactory
sslClientProps (security)	设置连接的 SSL 属性。请注意，只有 Java 安全套接字扩展(JSSE)可用时，这些属性才有效。如果设置了自定义 SocketFactory，则不会使用这些属性。可以使用以下属性：com.ibm.ssl.protocol One of: SSL, SSLv3, TLS, TLSv1, SSL_TLS. com.ibm.ssl.contextProvider Underlying JSSE provider.例如，IBMJSSE2 或 SunJSSE com.ibm.ssl.keyStore 是包含您希望 KeyManager 使用的 KeyStore 对象的文件的名称。例如，/mydir/etc/key.p12 com.ibm.ssl.keyStorePassword 是您希望 KeyManager 使用的 KeyStore 对象的密码。密码可以是纯文本，也可以使用静态方法混淆：com.ibm.micro.security.Password.obfuscate （支付密码）。这使用简单不安全的 XOR 和 Base64 编码机制来混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。com.ibm.ssl.keyStoreType Type 为密钥存储，如 PKCS12、JKS 或 JCEKS. com.ibm.ssl.keyStoreProvider Key 存储供应商，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.trustStore 包含您希望 TrustManager 使用的 KeyStore 对象的文件的名称。com.ibm.ssl.trustStorePassword。您想要 TrustStore 对象的 TrustStore 对象的密码。密码可以是纯文本，也可以使用静态方法混淆：com.ibm.micro.security.Password.obfuscate （支付密码）。这使用简单不安全的 XOR 和 Base64 编码机制来混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。com.ibm.ssl.trustStoreType 是您希望默认 TrustManager 使用的 KeyStore 对象类型。与 keyStoreType. com.ibm.ssl.trustStoreProvider Trust store provider 相同的可能值，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.enabledCipherSuites A 列表启用了哪些密码。值依赖于提供程序，例如：SSL_RSA_WITH_AES_128_CBC_SHA;SSL_RSA_WITH_3DES_EDE_CBC_SHA. com.ibm.ssl.keyManager 设置用于实例化 KeyManagerFactory 对象而不是使用平台中提供的默认算法的算法。示例值：IbmX509 或 IBMJ9X509. com.ibm.ssl.trustManager 设置用于实例化 TrustManagerFactory 对象的算法，而不是使用平台中提供的默认算法。示例值：PKIX 或 IBMJ9X509.		Properties

Name	描述	默认值	类型
sslHostnameVerifier (security)	为 SSL 连接设置 HostnameVerifier。请注意，它将在连接上的握手后使用，您应该在验证主机名错误时自行执行操作。没有默认的 HostnameVerifier。		HostnameVerifier
用户名 （安全性）	用于针对 MQTT 代理进行身份验证的用户名。		字符串

101.5. 端点选项

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：

101.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
topic (common)	主题的 **必填** 名称。		字符串

101.5.2. 查询参数(31 参数)

Name	描述	默认值	类型
automaticReconnect (common)	设置在连接丢失时客户端是否自动尝试重新连接到服务器。如果设置为 false，则在连接丢失时客户端不会尝试自动重新连接到服务器。如果设置为 true，如果连接丢失，客户端将尝试重新连接到服务器。它初始会在尝试重新连接前等待 1 秒，对于每个失败的尝试，其延长会加倍，指定 2 分钟为止，此时，延迟会一直为 2 分钟。	true	布尔值
brokerUrl (common)	MQTT 代理的 URL。	tcp://localhost:1883	字符串

Name	描述	默认值	类型
cleanSession (common)	设置客户端和服务器在重新启动和重新连接后是否应记住状态。如果设置为 false，则客户端和服务器在重启客户端、服务器和连接后将维持状态。由于状态被维护：消息交付将可靠满足指定的 QOS，即使客户端、服务器或连接被重启也是如此。服务器会将订阅视为持久。如果设置为 true，则客户端和服务器在重启客户端、服务器或连接时不会维护状态。这意味着，如果客户端、服务器或连接被重启，则消息发送到指定的 QOS 无法被维护。服务器会将订阅视为非持久性。	true	布尔值
clientId (common)	MQTT 客户端标识符。标识符必须是唯一的。		字符串
connectionTimeout (common)	设置连接超时值。这个值（以秒为单位）定义客户端等待 MQTT 服务器的网络连接的最大时间间隔（以秒为单位）。默认超时为 30 秒。值 0 禁用超时处理意味着客户端将等待到网络连接成功或失败。	30	int
filePersistenceDirectory (common)	文件持久性使用的基础目录。默认情况下，将使用 user 目录。		字符串
keepAliveInterval (common)	设置 keep alive 间隔。这个值（以秒为单位）定义发送或接收的消息之间的最大时间间隔。它使客户端能够检测服务器是否不再可用，而无需等待 TCP/IP 超时。客户端将确保每个保持活跃期间内至少有一个消息在网络间传输。如果在一段时间内没有与数据相关的消息，客户端会发送一个非常小的 ping 消息，后者将确认服务器。值 0 可禁用客户端中的 keepalive 处理。默认值为 60 秒。	60	int
maxInflight (common)	设置 max inflight。请在高流量环境中增加这个值。默认值为 10。	10	int
maxReconnectDelay (common)	获取重新连接之间等待的最长时间（以 millis）。	128000	int
mqttVersion (common)	设置 MQTT 版本。默认操作是与版本 3.1.1 连接，如果失败，则回退到 3.1。可以使用 MQTT_VERSION_3_1_1 或 MQTT_VERSION_3_1 选项，专门选择版本 3.1.1 或 3.1，且无回退。		int
Persistence (common)	要使用的客户端持久性 - 内存或文件。 Enum 值： • FILE • MEMORY	MEMORY	PahoPersistence

Name	描述	默认值	类型
QoS (common)	客户端服务质量级别(0-2)。	2	int
reserved (common)	Retain 选项。	false	布尔值
serverURIs (common)	设置客户端可以连接到的一个或多个 serverURI 的列表。可以使用逗号分隔多个服务器。每个 serverURI 指定客户端可以连接的服务器的地址。对于 TCP 连接，支持两种类型的连接，ssl:// 用于由 SSL/TLS 安全的 TCP 连接。例如： tcp://localhost:1883 ssl://localhost:8883 如果未指定端口，则为 tcp:// URIs 默认为 1883，对于 ssl:// URIs，8883 为 8883。如果设置了 serverURIs，则它将覆盖 MQTT 客户端构造器上传递的 serverURI 参数。当尝试连接时，客户端将从列表中的第一个 serverURI 开始，并操作列表，直到与服务器建立连接为止。如果无法对任何服务器建立连接，则连接尝试会失败。指定客户端可以连接到多个用途的服务器列表：高可用性和可靠的消息交付一些 MQTT 服务器支持高可用性功能，其中两个或更多 MQTT 服务器共享状态。MQTT 客户端可以连接到任何等同的服务器，并保证无论客户端连接到哪个服务器，都可靠交付和持久化订阅。如果需要 durable 订阅和/或可靠的消息交付，则必须将 cleansession 标志设置为 false。可以指定一组不相等的服务器集合（如高可用性选项中）。因为服务器之间没有共享状态，因此消息交付和持久化订阅无效。如果使用了 hunt 列表模式，则必须将 cleansession 标志设置为 true。		字符串
willPayload (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。要发布到消息的字节有效负载的主题。发布消息的服务质量(0、1 或 2)。消息是否应保留。		字符串
willQos (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。要发布到消息的字节有效负载的主题。发布消息的服务质量(0、1 或 2)。消息是否应保留。		int
willRetained (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。要发布到消息的字节有效负载的主题。发布消息的服务质量(0、1 或 2)。消息是否应保留。	false	布尔值

Name	描述	默认值	类型
willTopic (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。要发布到消息的字节有效负载的主题。发布消息的服务质量(0、1 或 2)。消息是否应保留。		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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		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
client (advanced)	使用现有的 mqtt 客户端。		MqttClient
customWebSocketHeaders (advanced)	为 WebSocket 连接设置自定义 WebSocket 标头。		Properties
executorServiceTimeout (advanced)	设置 executor 服务在强制终止前应等待的时间（以秒为单位）。除非绝对确定您需要修改这个值，否则不建议更改这个值。	1	int
httpsHostnameVerificationEnabled (security)	是否启用 SSL HostnameVerifier。默认值为 true。	true	布尔值

Name	描述	默认值	类型
password (security)	用于针对 MQTT 代理进行身份验证的密码。		字符串
socketFactory (security)	设置要使用的 SocketFactory。这允许应用程序在创建网络套接字时应用自己的策略。如果使用 SSL 连接，可以使用 SSLSocketFactory 提供应用特定的安全设置。		SocketFactory
sslClientProps (security)	设置连接的 SSL 属性。请注意，只有 Java 安全套接字扩展(JSSE)可用时，这些属性才有效。如果设置了自定义 SocketFactory，则不会使用这些属性。可以使用以下属性：com.ibm.ssl.protocol One of: SSL, SSLv3, TLS, TLSv1, SSL_TLS. com.ibm.ssl.contextProvider Underlying JSSE provider.例如，IBMJSSE2 或 SunJSSE com.ibm.ssl.keyStore 是包含您希望 KeyManager 使用的 KeyStore 对象的文件的名称。例如，/mydir/etc/key.p12 com.ibm.ssl.keyStorePassword 是您希望 KeyManager 使用的 KeyStore 对象的密码。密码可以是纯文本，也可以使用静态方法混淆：com.ibm.micro.security.Password.obfuscate （支付密码）。这使用简单不安全的 XOR 和 Base64 编码机制来混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。com.ibm.ssl.keyStoreType Type 为密钥存储，如 PKCS12、JKS 或 JCEKS. com.ibm.ssl.keyStoreProvider Key 存储供应商，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.trustStore 包含您希望 TrustManager 使用的 KeyStore 对象的文件的名称。com.ibm.ssl.trustStorePassword。您想要 TrustStore 对象的 TrustStore 对象的密码。密码可以是纯文本，也可以使用静态方法混淆：com.ibm.micro.security.Password.obfuscate （支付密码）。这使用简单不安全的 XOR 和 Base64 编码机制来混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。com.ibm.ssl.trustStoreType 是您希望默认 TrustManager 使用的 KeyStore 对象类型。与 keyStoreType. com.ibm.ssl.trustStoreProvider Trust store provider 相同的可能值，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.enabledCipherSuites A 列表启用了哪些密码。值依赖于提供程序，例如：SSL_RSA_WITH_AES_128_CBC_SHA;SSL_RSA_WITH_3DES_EDE_CBC_SHA. com.ibm.ssl.keyManager 设置用于实例化 KeyManagerFactory 对象而不是使用平台中提供的默认算法的算法。示例值：IbmX509 或 IBMJ9X509. com.ibm.ssl.trustManager 设置用于实例化 TrustManagerFactory 对象的算法，而不是使用平台中提供的默认算法。示例值：PKIX 或 IBMJ9X509.		Properties

Name	描述	默认值	类型
sslHostnameVerifier (security)	为 SSL 连接设置 HostnameVerifier。请注意，它将在连接上的握手后使用，您应该在验证主机名错误时自行执行操作。没有默认的 HostnameVerifier。		HostnameVerifier
用户名 （安全性）	用于针对 MQTT 代理进行身份验证的用户名。		字符串

101.6. 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
CamelMqttQoS	PahoConstants.MQTT_QOS	消费者	整数	传入消息的 QualityOfService
CamelPahoOverrideTopic	PahoConstants.CAMEL_PAHO_OVERRIDE_TOPIC	制作者	字符串	用于覆盖和发送到端点上的主题名称，而不是在端点中指定主题

101.7. 默认有效负载类型

默认情况下，Camel Paho 组件对从 MQTT 消息中提取的二进制有效负载上运行：

```
// 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 （并反反）：

```
// Receive payload
String payload = consumerTemplate.receiveBody("paho:topic", String.class);
```

```
// Send payload
String payload = "message";
producerTemplate.sendBody("paho:topic", payload);
```

101.8. SAMPLES

例如，以下代码片段从与 Camel 路由器在同一主机上安装的 MQTT 代理读取消息：

```
from("paho:some/queue")
    .to("mock:test");
```

虽然下面的片断发送消息到 MQTT 代理：

```
from("direct:test")
    .to("paho:some/target/queue");
```

例如，这是如何从远程 MQTT 代理中读取消息：

```
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");
```

101.9. 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.paho.automatic-reconnect	设置在连接丢失时客户端是否自动尝试重新连接到服务器。如果设置为 false，则在连接丢失时客户端不会尝试自动重新连接到服务器。如果设置为 true，如果连接丢失，客户端将尝试重新连接到服务器。它初始会在尝试重新连接前等待 1 秒，对于每个失败的尝试，其延长会加倍，指定 2 分钟为止，此时，延迟会一直为 2 分钟。	true	布尔值

Name	描述	默认值	类型
camel.component.paho.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.paho.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.paho.broker-url	MQTT 代理的 URL。	tcp://localhost:1883	字符串
camel.component.paho.clean-session	设置客户端和服务器在重新启动和重新连接后是否应记住状态。如果设置为 false，则客户端和服务器在重启客户端、服务器和连接后将维持状态。由于状态被维护：消息交付将可靠满足指定的 QOS，即使客户端、服务器或连接被重启也是如此。服务器会将订阅视为持久。如果设置为 true，则客户端和服务器在重启客户端、服务器或连接时不会维护状态。这意味着，如果客户端、服务器或连接被重启，则消息发送到指定的 QOS 无法被维护。服务器会将订阅视为非持久性。	true	布尔值
camel.component.paho.client	使用共享 Paho 客户端。选项是一个 org.eclipse.paho.client.mqttv3.MqttClient 类型。		MqttClient
camel.component.paho.client-id	MQTT 客户端标识符。标识符必须是唯一的。		字符串
camel.component.paho.configuration	使用共享 Paho 配置。选项是 org.apache.camel.component.paho.PahoConfiguration 类型。		PahoConfiguration
camel.component.paho.connection-timeout	设置连接超时值。这个值（以秒为单位）定义客户端等待 MQTT 服务器的网络连接的最大时间间隔（以秒为单位）。默认超时为 30 秒。值 0 禁用超时处理意味着客户端将等待到网络连接成功或失败。	30	整数
camel.component.paho.custom-web-socket-headers	为 WebSocket 连接设置自定义 WebSocket 标头。选项是一个 java.util.Properties 类型。		Properties

Name	描述	默认值	类型
camel.component.paho.enabled	是否启用 paho 组件的自动配置。这默认是启用的。		布尔值
camel.component.paho.executor-service-timeout	设置 executor 服务在强制终止前应等待的时间（以秒为单位）。除非绝对确定您需要修改这个值，否则不建议更改这个值。	1	整数
camel.component.paho.file-persistence-directory	文件持久性使用的基础目录。默认情况下，将使用 user 目录。		字符串
camel.component.paho.https-hostname-verification-enabled	是否启用 SSL HostnameVerifier。默认值为 true。	true	布尔值
camel.component.paho.keep-alive-interval	设置 keep alive 间隔。这个值（以秒为单位）定义发送或接收的消息之间的最大时间间隔。它使客户端能够检测服务器是否不再可用，而无需等待 TCP/IP 超时。客户端将确保每个保持活跃期间内至少有一个消息在网络间传输。如果在一段时间内没有与数据相关的消息，客户端会发送一个非常小的 ping 消息，后者将确认服务器。值 0 可禁用客户端中的 keepalive 处理。默认值为 60 秒。	60	整数
camel.component.paho.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.paho.max-inflight	设置 max inflight。请在高流量环境中增加这个值。默认值为 10。	10	整数
camel.component.paho.max-reconnect-delay	获取重新连接之间等待的最长时间（以 millis）。	128000	整数
camel.component.paho.mqtt-version	设置 MQTT 版本。默认操作是与版本 3.1.1 连接，如果失败，则回退到 3.1。可以使用 MQTT_VERSION_3_1_1 或 MQTT_VERSION_3_1 选项，专门选择版本 3.1.1 或 3.1，且无回退。		整数

Name	描述	默认值	类型
camel.component.paho.password	用于针对 MQTT 代理进行身份验证的密码。		字符串
camel.component.paho.persistence	要使用的客户端持久性 - 内存或文件。		PahoPersistence
camel.component.paho.qos	客户端服务质量级别(0-2)。	2	整数
camel.component.paho.retained	Retain 选项。	false	布尔值
camel.component.paho.server-u-r-is	设置客户端可以连接到的一个或多个 serverURI 的列表。可以使用逗号分隔多个服务器。每个 serverURI 指定客户端可以连接的服务器的地址。对于 TCP 连接，支持两种类型的连接，ssl:// 用于由 SSL/TLS 安全的 TCP 连接。例如： tcp://localhost:1883 ssl://localhost:8883 如果未指定端口，则为 tcp:// URIs 默认为 1883，对于 ssl:// URIs，8883 为 8883。如果设置了 serverURIs，则它将覆盖 MQTT 客户端构造器上传递的 serverURI 参数。当尝试连接时，客户端将从列表中的第一个 serverURI 开始，并操作列表，直到与服务器建立连接为止。如果无法对任何服务器建立连接，则连接尝试会失败。指定客户端可以连接到多个用途的服务器列表：高可用性和可靠的消息交付一些 MQTT 服务器支持高可用性功能，其中两个或更多 MQTT 服务器共享状态。MQTT 客户端可以连接到任何等同的服务器，并保证无论客户端连接到哪个服务器，都可靠交付和持久化订阅。如果需要 durable 订阅和/或可靠的消息交付，则必须将 cleansession 标志设置为 false。可以指定一组不相等的服务器集合（如高可用性选项中）。因为服务器之间没有共享状态，因此消息交付和持久化订阅无效。如果使用了 hunt 列表模式，则必须将 cleansession 标志设置为 true。		字符串
camel.component.paho.socket-factory	设置要使用的 SocketFactory。这允许应用程序在创建网络套接字时应用自己的策略。如果使用 SSL 连接，可以使用 SSLSocketFactory 提供应用特定的安全设置。选项是 javax.net.SocketFactory 类型。		SocketFactory

Name	描述	默认值	类型
camel.component.paho.ssl-client-props	设置连接的 SSL 属性。请注意，只有 Java 安全套接字扩展(JSSE)可用时，这些属性才有效。如果设置了自定义 SocketFactory，则不会使用这些属性。可以使用以下属性：com.ibm.ssl.protocol One of: SSL, SSLv3, TLS, TLSv1, SSL_TLS. com.ibm.ssl.contextProvider Underlying JSSE provider.例如，IBMJSSE2 或 SunJSSE com.ibm.ssl.keyStore 是包含您希望 KeyManager 使用的 KeyStore 对象的文件的名称。例如，/mydir/etc/key.p12 com.ibm.ssl.keyStorePassword 是您希望 KeyManager 使用的 KeyStore 对象的密码。密码可以是纯文本，也可以使用静态方法混淆：com.ibm.micro.security.Password.obfuscate （支付密码）。这使用简单不安全的 XOR 和 Base64 编码机制来混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。com.ibm.ssl.keyStoreType Type 为密钥存储，如 PKCS12、JKS 或 JCEKS. com.ibm.ssl.keyStoreProvider Key 存储供应商，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.trustStore 包含您希望 TrustManager 使用的 KeyStore 对象的文件的名称。com.ibm.ssl.trustStorePassword。您想要 TrustStore 对象的 TrustStore 对象的密码。密码可以是纯文本，也可以使用静态方法混淆：com.ibm.micro.security.Password.obfuscate （支付密码）。这使用简单不安全的 XOR 和 Base64 编码机制来混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。com.ibm.ssl.trustStoreType 是您希望默认 TrustManager 使用的 KeyStore 对象类型。与 keyStoreType. com.ibm.ssl.trustStoreProvider Trust store provider 相同的可能值，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.enabledCipherSuites A 列表启用了哪些密码。值依赖于提供程序，例如：SSL_RSA_WITH_AES_128_CBC_SHA;SSL_RSA_WITH_3DES_EDE_CBC_SHA. com.ibm.ssl.keyManager 设置用于实例化 KeyManagerFactory 对象而不是使用平台中提供的默认算法的算法。示例值：IbmX509 或 IBMJ9X509. com.ibm.ssl.trustManager 设置用于实例化 TrustManagerFactory 对象的算法，而不是使用平台中提供的默认算法。示例值：PKIX 或 IBMJ9X509. 选项是一个 java.util.Properties 类型。		Properties
camel.component.paho.ssl-hostname-verifier	为 SSL 连接设置 HostnameVerifier。请注意，它将在连接上的握手后使用，您应该在验证主机名错误时自行执行操作。没有默认的 HostnameVerifier。选项是 javax.net.ssl.HostnameVerifier 类型。		HostnameVerifier
camel.component.paho.user-name	用于针对 MQTT 代理进行身份验证的用户名。		字符串

Name	描述	默认值	类型
camel.component.paho.will-payload	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。要发布到消息的字节有效负载的主题。发布消息的服务质量(0、1 或 2)。消息是否应保留。		字符串
camel.component.paho.will-qos	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。要发布到消息的字节有效负载的主题。发布消息的服务质量(0、1 或 2)。消息是否应保留。		整数
camel.component.paho.will-retained	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。要发布到消息的字节有效负载的主题。发布消息的服务质量(0、1 或 2)。消息是否应保留。	false	布尔值
camel.component.paho.will-topic	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。要发布到消息的字节有效负载的主题。发布消息的服务质量(0、1 或 2)。消息是否应保留。		字符串

第 102 章 PAHO MQTT 5

支持生成者和消费者

paho MQTT5 组件使用具有 MQTT v5 的 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，因此如果您想将其与 Java 项目集成 - Camel Paho 连接器是一个要前往的方法。

102.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 paho-mqtt5 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-paho-mqtt5-starter</artifactId>
</dependency>
```

102.2. URI 格式

```
paho-mqtt5:topic[?options]
```

其中 topic 是主题的名称。

102.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

102.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

102.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

102.4. 组件选项

Paho MQTT 5 组件支持 32 个选项，如下所列。

Name	描述	默认值	类型
automaticReconnect (common)	设置在连接丢失时客户端是否自动尝试重新连接到服务器。如果设置为 false，则在连接丢失时客户端不会尝试自动重新连接到服务器。如果设置为 true，如果连接丢失，客户端将尝试重新连接到服务器。它初始会在尝试重新连接前等待 1 秒，对于每个失败的尝试，其延长会加倍，指定 2 分钟为止，此时，延迟会一直为 2 分钟。	true	布尔值
brokerUrl (common)	MQTT 代理的 URL。	tcp://localhost:1883	字符串

Name	描述	默认值	类型
cleanStart (common)	设置客户端和服务器在重新启动和重新连接后是否应记住状态。如果设置为 false，则客户端和服务器在重启客户端、服务器和连接后将维持状态。由于状态被维护：消息交付将可靠满足指定的 QOS，即使客户端、服务器或连接被重启也是如此。服务器会将订阅视为持久。如果设置为 true，则客户端和服务器在重启客户端、服务器或连接时不会维护状态。这意味着，如果客户端、服务器或连接被重启，则消息发送到指定的 QOS 无法被维护。服务器会将订阅视为非持久性。	true	布尔值
clientId (common)	MQTT 客户端标识符。标识符必须是唯一的。		字符串
configuration (common)	使用共享 Paho 配置。		PahoMqtt5Configuration
connectionTimeout (common)	设置连接超时值。这个值（以秒为单位）定义客户端等待 MQTT 服务器的网络连接的最大时间间隔（以秒为单位）。默认超时为 30 秒。值 0 禁用超时处理意味着客户端将等待到网络连接成功或失败。	30	int
filePersistenceDirectory (common)	文件持久性使用的基础目录。默认情况下，将使用 user 目录。		字符串
keepAliveInterval (common)	设置 keep alive 间隔。这个值（以秒为单位）定义发送或接收的消息之间的最大时间间隔。它使客户端能够检测服务器是否不再可用，而无需等待 TCP/IP 超时。客户端将确保每个保持活跃期间内至少有一个消息在网络间传输。如果在一段时间内没有与数据相关的消息，客户端会发送一个非常小的 ping 消息，后者将确认服务器。值 0 可禁用客户端中的 keepalive 处理。默认值为 60 秒。	60	int
maxReconnectDelay (common)	获取重新连接之间等待的最长时间（以 millis）。	128000	int
Persistence (common)	要使用的客户端持久性 - 内存或文件。 Enum 值： • FILE • MEMORY	MEMORY	PahoMqtt5Persistence
QoS (common)	客户端服务质量级别(0-2)。	2	int

Name	描述	默认值	类型
receiveMaximum (common)	设置接收的最大值。这个值代表 QoS 1 和 QoS 2 发布的限制，客户端将同时处理。没有机制来限制服务器可能尝试发送的 QoS 0 出版物数。默认值为 65535。	65535	int
reserved (common)	Retain 选项。	false	布尔值
serverURIs (common)	设置客户端可以连接到的一个或多个 serverURI 的列表。可以使用逗号分隔多个服务器。每个 serverURI 指定客户端可以连接的服务器的地址。对于 TCP 连接，支持两种类型的连接，ssl:// 用于由 SSL/TLS 安全的 TCP 连接。例如： tcp://localhost:1883 ssl://localhost:8883 如果未指定端口，则为 tcp:// URIs 默认为 1883，对于 ssl:// URIs，8883 为 8883。如果设置了 serverURIs，则它将覆盖 MQTT 客户端构造器上传递的 serverURI 参数。当尝试连接时，客户端将从列表中的第一个 serverURI 开始，并操作列表，直到与服务器建立连接为止。如果无法对任何服务器建立连接，则连接尝试会失败。指定客户端可以连接到多个用途的服务器列表：高可用性和可靠的消息交付一些 MQTT 服务器支持高可用性功能，其中两个或更多 MQTT 服务器共享状态。MQTT 客户端可以连接到任何等同的服务器，并保证无论客户端连接到哪个服务器，都可靠交付和持久化订阅。如果需要 durable 订阅和/或可靠的消息交付，则必须将 cleansession 标志设置为 false。可以指定一组不相等的服务器集合（如高可用性选项中）。因为服务器之间没有共享状态，因此消息交付和持久化订阅无效。如果使用了 hunt 列表模式，则必须将 cleansession 标志设置为 true。		字符串
sessionExpiryInterval (common)	设置 Session Expiry Interval。这个值（以秒为单位）定义代理在客户端断开连接后维护会话的最长时间。如果客户端计划在以后的时间点连接到服务器，则客户端应仅与较长的会话过期间隔连接。默认情况下，这个值为 -1，因此不会发送，在这种情况下，会话不会过期。如果发送了 0，则会话将在网络连接关闭后立即结束。当客户端确定不再用于会话时，它应该断开连接，并将 Session Expiry Interval 设置为 0。	-1	long
willMqttProperties (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。为消息设置的 MQTT 属性。		MqttProperties
willPayload (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。消息的字节有效负载。		字符串

Name	描述	默认值	类型
willQos (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。发布消息的服务质量 (0、1 或 2)。	1	int
willRetained (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。消息是否应保留。	false	布尔值
willTopic (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。要发布到的主题。		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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
client (advanced)	使用共享 Paho 客户端。		MqttClient
customWebSocketHeaders (advanced)	为 WebSocket 连接设置自定义 WebSocket 标头。		Map
executorServiceTimeout (advanced)	设置 executor 服务在强制终止前应等待的时间（以秒为单位）。除非绝对确定您需要修改这个值，否则不建议更改这个值。	1	int

Name	描述	默认值	类型
httpsHostnameVerificationEnabled (security)	是否启用 SSL HostnameVerifier。默认值为 true。	true	布尔值
password (security)	用于针对 MQTT 代理进行身份验证的密码。		字符串
socketFactory (security)	设置要使用的 SocketFactory。这允许应用程序在创建网络套接字时应用自己的策略。如果使用 SSL 连接，可以使用 SSLSocketFactory 提供应用特定的安全设置。		SocketFactory

Name	描述	默认值	类型
sslClientProps (security)	设置连接的 SSL 属性。请注意，只有 Java 安全套接字扩展(JSSE)可用时，这些属性才有效。如果设置了自定义 SocketFactory，则不会使用这些属性。可以使用以下属性：com.ibm.ssl.protocol One of: SSL, SSLv3, TLS, TLSv1, SSL_TLS. com.ibm.ssl.contextProvider Underlying JSSE provider.例如，IBMJSSE2 或 SunJSSE com.ibm.ssl.keyStore 是包含您希望 KeyManager 使用的 KeyStore 对象的文件的名称。例如，/mydir/etc/key.p12 com.ibm.ssl.keyStorePassword 是您希望 KeyManager 使用的 KeyStore 对象的密码。密码可以是纯文本，也可以使用静态方法混淆：com.ibm.micro.security.Password.obfuscate （支付密码）。这使用简单不安全的 XOR 和 Base64 编码机制来混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。com.ibm.ssl.keyStoreType Type 为密钥存储，如 PKCS12、JKS 或 JCEKS. com.ibm.ssl.keyStoreProvider Key 存储供应商，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.trustStore 包含您希望 TrustManager 使用的 KeyStore 对象的文件的名称。com.ibm.ssl.trustStorePassword。您想要 TrustStore 对象的 TrustStore 对象的密码。密码可以是纯文本，也可以使用静态方法混淆：com.ibm.micro.security.Password.obfuscate （支付密码）。这使用简单不安全的 XOR 和 Base64 编码机制来混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。com.ibm.ssl.trustStoreType 是您希望默认 TrustManager 使用的 KeyStore 对象类型。与 keyStoreType. com.ibm.ssl.trustStoreProvider Trust store provider 相同的可能值，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.enabledCipherSuites A 列表启用了哪些密码。值依赖于提供程序，例如：SSL_RSA_WITH_AES_128_CBC_SHA;SSL_RSA_WITH_3DES_EDE_CBC_SHA. com.ibm.ssl.keyManager 设置用于实例化 KeyManagerFactory 对象而不是使用平台中提供的默认算法的算法。示例值：IbmX509 或 IBMJ9X509. com.ibm.ssl.trustManager 设置用于实例化 TrustManagerFactory 对象的算法，而不是使用平台中提供的默认算法。示例值：PKIX 或 IBMJ9X509.		Properties
sslHostnameVerifier (security)	为 SSL 连接设置 HostnameVerifier。请注意，它将在连接上的握手后使用，您应该在验证主机名错误时自行执行操作。没有默认的 HostnameVerifier。		HostnameVerifier
用户名 （安全性）	用于针对 MQTT 代理进行身份验证的用户名。		字符串

102.5. 端点选项

Paho MQTT 5 端点使用 URI 语法进行配置：

```
paho-mqtt5:topic
```

使用以下路径和查询参数：

102.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
topic (common)	主题的 **必填** 名称。		字符串

102.5.2. 查询参数(32 参数)

Name	描述	默认值	类型
automaticReconnect (common)	设置在连接丢失时客户端是否自动尝试重新连接到服务器。如果设置为 false，则在连接丢失时客户端不会尝试自动重新连接到服务器。如果设置为 true，如果连接丢失，客户端将尝试重新连接到服务器。它初始会在尝试重新连接前等待 1 秒，对于每个失败的尝试，其延长会加倍，指定 2 分钟为止，此时，延迟会一直为 2 分钟。	true	布尔值
brokerUrl (common)	MQTT 代理的 URL。	tcp://localhost:1883	字符串
cleanStart (common)	设置客户端和服务器在重新启动和重新连接后是否应记住状态。如果设置为 false，则客户端和服务器在重启客户端、服务器和连接后将维持状态。由于状态被维护：消息交付将可靠满足指定的 QOS，即使客户端、服务器或连接被重启也是如此。服务器会将订阅视为持久。如果设置为 true，则客户端和服务器在重启客户端、服务器或连接时不会维护状态。这意味着，如果客户端、服务器或连接被重启，则消息发送到指定的 QOS 无法被维护。服务器会将订阅视为非持久性。	true	布尔值
clientId (common)	MQTT 客户端标识符。标识符必须是唯一的。		字符串
connectionTimeout (common)	设置连接超时值。这个值（以秒为单位）定义客户端等待 MQTT 服务器的网络连接的最大时间间隔（以秒为单位）。默认超时为 30 秒。值 0 禁用超时处理意味着客户端将等待到网络连接成功或失败。	30	int

Name	描述	默认值	类型
file PersistenceDirectory (common)	文件持久性使用的基础目录。默认情况下，将使用 user 目录。		字符串
keepAliveInterval (common)	设置 keep alive 间隔。这个值（以秒为单位）定义发送或接收的消息之间的最大时间间隔。它使客户端能够检测服务器是否不再可用，而无需等待 TCP/IP 超时。客户端将确保每个保持活跃期间内至少有一个消息在网络间传输。如果在一段时间内没有与数据相关的消息，客户端会发送一个非常小的 ping 消息，后者将确认服务器。值 0 可禁用客户端中的 keepalive 处理。默认值为 60 秒。	60	int
maxReconnectDelay (common)	获取重新连接之间等待的最长时间（以 millis）。	128000	int
Persistence (common)	要使用的客户端持久性 - 内存或文件。 Enum 值： • FILE • MEMORY	MEMORY	PahoMqtt5Persistence
QoS (common)	客户端服务质量级别(0-2)。	2	int
receiveMaximum (common)	设置接收的最大值。这个值代表 QoS 1 和 QoS 2 发布的限制，客户端将同时处理。没有机制来限制服务器可能尝试发送的 QoS 0 出版物数。默认值为 65535。	65535	int
reserved (common)	Retain 选项。	false	布尔值

Name	描述	默认值	类型
serverURIs (common)	设置客户端可以连接到的一个或多个 serverURI 的列表。可以使用逗号分隔多个服务器。每个 serverURI 指定客户端可以连接的服务器的地址。对于 TCP 连接，支持两种类型的连接，ssl:// 用于由 SSL/TLS 安全的 TCP 连接。例如： tcp://localhost:1883 ssl://localhost:8883 如果未指定端口，则为 tcp:// URIs 默认为 1883，对于 ssl:// URIs，8883 为 8883。如果设置了 serverURIs，则它将覆盖 MQTT 客户端构造器上传递的 serverURI 参数。当尝试连接时，客户端将从列表中的第一个 serverURI 开始，并操作列表，直到与服务器建立连接为止。如果无法对任何服务器建立连接，则连接尝试会失败。指定客户端可以连接到多个用途的服务器列表：高可用性和可靠的消息交付一些 MQTT 服务器支持高可用性功能，其中两个或更多 MQTT 服务器共享状态。MQTT 客户端可以连接到任何等同的服务器，并保证无论客户端连接到哪个服务器，都可靠交付和持久化订阅。如果需要 durable 订阅和/或可靠的消息交付，则必须将 cleansession 标志设置为 false。可以指定一组不相等的服务器集合（如高可用性选项中）。因为服务器之间没有共享状态，因此消息交付和持久化订阅无效。如果使用了 hunt 列表模式，则必须将 cleansession 标志设置为 true。		字符串
sessionExpiryInterval (common)	设置 Session Expiry Interval。这个值（以秒为单位）定义代理在客户端断开连接后维护会话的最长时间。如果客户端计划在以后的时间点连接到服务器，则客户端应仅与较长的会话过期间隔连接。默认情况下，这个值为 -1，因此不会发送，在这种情况下，会话不会过期。如果发送了 0，则会话将在网络连接关闭后立即结束。当客户端确定不再用于会话时，它应该断开连接，并将 Session Expiry Interval 设置为 0。	-1	long
willMqttProperties (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。为消息设置的 MQTT 属性。		MqttProperties
willPayload (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。消息的字节有效负载。		字符串
willQos (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。发布消息的服务质量 (0、1 或 2)。	1	int

Name	描述	默认值	类型
willRetained (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。消息是否应保留。	false	布尔值
willTopic (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。要发布到的主题。		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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		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
client (advanced)	使用现有的 mqtt 客户端。		MqttClient
customWebSocketHeaders (advanced)	为 WebSocket 连接设置自定义 WebSocket 标头。		Map
executorServiceTimeout (advanced)	设置 executor 服务在强制终止前应等待的时间（以秒为单位）。除非绝对确定您需要修改这个值，否则不建议更改这个值。	1	int

Name	描述	默认值	类型
httpsHostnameVerificationEnabled (security)	是否启用 SSL HostnameVerifier。默认值为 true。	true	布尔值
password (security)	用于针对 MQTT 代理进行身份验证的密码。		字符串
socketFactory (security)	设置要使用的 SocketFactory。这允许应用程序在创建网络套接字时应用自己的策略。如果使用 SSL 连接，可以使用 SSLSocketFactory 提供应用特定的安全设置。		SocketFactory

Name	描述	默认值	类型
sslClientProps (security)	设置连接的 SSL 属性。请注意，只有 Java 安全套接字扩展(JSSE)可用时，这些属性才有效。如果设置了自定义 SocketFactory，则不会使用这些属性。可以使用以下属性：com.ibm.ssl.protocol One of: SSL, SSLv3, TLS, TLSv1, SSL_TLS. com.ibm.ssl.contextProvider Underlying JSSE provider.例如，IBMJSSE2 或 SunJSSE com.ibm.ssl.keyStore 是包含您希望 KeyManager 使用的 KeyStore 对象的文件的名称。例如，/mydir/etc/key.p12 com.ibm.ssl.keyStorePassword 是您希望 KeyManager 使用的 KeyStore 对象的密码。密码可以是纯文本，也可以使用静态方法混淆：com.ibm.micro.security.Password.obfuscate （支付密码）。这使用简单不安全的 XOR 和 Base64 编码机制来混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。com.ibm.ssl.keyStoreType Type 为密钥存储，如 PKCS12、JKS 或 JCEKS. com.ibm.ssl.keyStoreProvider Key 存储供应商，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.trustStore 包含您希望 TrustManager 使用的 KeyStore 对象的文件的名称。com.ibm.ssl.trustStorePassword。您想要 TrustStore 对象的 TrustStore 对象的密码。密码可以是纯文本，也可以使用静态方法混淆：com.ibm.micro.security.Password.obfuscate （支付密码）。这使用简单不安全的 XOR 和 Base64 编码机制来混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。com.ibm.ssl.trustStoreType 是您希望默认 TrustManager 使用的 KeyStore 对象类型。与 keyStoreType. com.ibm.ssl.trustStoreProvider Trust store provider 相同的可能值，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.enabledCipherSuites A 列表启用了哪些密码。值依赖于提供程序，例如：SSL_RSA_WITH_AES_128_CBC_SHA;SSL_RSA_WITH_3DES_EDE_CBC_SHA. com.ibm.ssl.keyManager 设置用于实例化 KeyManagerFactory 对象而不是使用平台中提供的默认算法的算法。示例值：IbmX509 或 IBMJ9X509. com.ibm.ssl.trustManager 设置用于实例化 TrustManagerFactory 对象的算法，而不是使用平台中提供的默认算法。示例值：PKIX 或 IBMJ9X509.		Properties
sslHostnameVerifier (security)	为 SSL 连接设置 HostnameVerifier。请注意，它将在连接上的握手后使用，您应该在验证主机名错误时自行执行操作。没有默认的 HostnameVerifier。		HostnameVerifier
用户名 （安全性）	用于针对 MQTT 代理进行身份验证的用户名。		字符串

102.6. 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
CamelMqttQoS	PahoConstants.MQTT_QOS	消费者	整数	传入消息的 QualityOfService
CamelPahoOverrideTopic	PahoConstants.CAMEL_PAHO_OVERRIDE_TOPIC	制作者	字符串	用于覆盖和发送到端点上的主题名称，而不是在端点中指定主题

102.7. 默认有效负载类型

默认情况下，Camel Paho 组件对从 MQTT 消息中提取的二进制有效负载上运行：

```
// 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 （并反反）：

```
// Receive payload
String payload = consumerTemplate.receiveBody("paho:topic", String.class);

// Send payload
String payload = "message";
producerTemplate.sendBody("paho:topic", payload);
```

102.8. SAMPLES

例如，以下代码片段从与 Camel 路由器在同一主机上安装的 MQTT 代理读取消息：

```
from("paho:some/queue")
    .to("mock:test");
```

虽然下面的片断发送消息到 MQTT 代理：

```
from("direct:test")
    .to("paho:some/target/queue");
```

例如，这是如何从远程 MQTT 代理中读取消息：

```
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");
```

102.9. 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.paho-mqtt5.automatic-reconnect	设置在连接丢失时客户端是否自动尝试重新连接到服务器。如果设置为 false，则在连接丢失时客户端不会尝试自动重新连接到服务器。如果设置为 true，如果连接丢失，客户端将尝试重新连接到服务器。它初始会在尝试重新连接前等待 1 秒，对于每个失败的尝试，其延长会加倍，指定 2 分钟为止，此时，延迟会一直为 2 分钟。	true	布尔值
camel.component.paho-mqtt5.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.paho-mqtt5.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

Name	描述	默认值	类型
camel.component.paho-mqtt5.broker-url	MQTT 代理的 URL。	tcp://localhost:1883	字符串
camel.component.paho-mqtt5.clean-start	设置客户端和服务器在重新启动和重新连接后是否应记住状态。如果设置为 false，则客户端和服务器在重启客户端、服务器和连接后将维持状态。由于状态被维护：消息交付将可靠满足指定的 QOS，即使客户端、服务器或连接被重启也是如此。服务器会将订阅视为持久。如果设置为 true，则客户端和服务器在重启客户端、服务器或连接时不会维护状态。这意味着，如果客户端、服务器或连接被重启，则消息发送到指定的 QOS 无法被维护。服务器会将订阅视为非持久性。	true	布尔值
camel.component.paho-mqtt5.client	使用共享 Paho 客户端。选项是一个 org.eclipse.paho.mqttv5.client.MqttClient 类型。		MqttClient
camel.component.paho-mqtt5.client-id	MQTT 客户端标识符。标识符必须是唯一的。		字符串
camel.component.paho-mqtt5.configuration	使用共享 Paho 配置。选项是 org.apache.camel.component.paho.mqtt5.PahoMqtt5Configuration 类型。		PahoMqtt5Configuration
camel.component.paho-mqtt5.connection-timeout	设置连接超时值。这个值（以秒为单位）定义客户端等待 MQTT 服务器的网络连接的最大时间间隔（以秒为单位）。默认超时为 30 秒。值 0 禁用超时处理意味着客户端将等待到网络连接成功或失败。	30	整数
camel.component.paho-mqtt5.custom-web-socket-headers	为 WebSocket 连接设置自定义 WebSocket 标头。		Map
camel.component.paho-mqtt5.enabled	是否启用 paho-mqtt5 组件的自动配置。这默认是启用的。		布尔值
camel.component.paho-mqtt5.executor-service-timeout	设置 executor 服务在强制终止前应等待的时间（以秒为单位）。除非绝对确定您需要修改这个值，否则不建议更改这个值。	1	整数

Name	描述	默认值	类型
camel.component.paho-mqtt5.file-persistence-directory	文件持久性使用的基础目录。默认情况下，将使用 user 目录。		字符串
camel.component.paho-mqtt5.https-hostname-verification-enabled	是否启用 SSL HostnameVerifier。默认值为 true。	true	布尔值
camel.component.paho-mqtt5.keep-alive-interval	设置 keep alive 间隔。这个值（以秒为单位）定义发送或接收的消息之间的最大时间间隔。它使客户端能够检测服务器是否不再可用，而无需等待 TCP/IP 超时。客户端将确保每个保持活跃期间内至少有一个消息在网络间传输。如果在一段时间内没有与数据相关的消息，客户端会发送一个非常小的 ping 消息，后者将确认服务器。值 0 可禁用客户端中的 keepalive 处理。默认值为 60 秒。	60	整数
camel.component.paho-mqtt5.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.paho-mqtt5.max-reconnect-delay	获取重新连接之间等待的最长时间（以 millis）。	128000	整数
camel.component.paho-mqtt5.password	用于针对 MQTT 代理进行身份验证的密码。		字符串
camel.component.paho-mqtt5.persistence	要使用的客户端持久性 - 内存或文件。		PahoMqtt5Persistence
camel.component.paho-mqtt5.qos	客户端服务质量级别(0-2)。	2	整数

Name	描述	默认值	类型
camel.component.paho-mqtt5.receive-maximum	设置接收的最大值。这个值代表 QoS 1 和 QoS 2 发布的限制，客户端将同时处理。没有机制来限制服务器可能尝试发送的 QoS 0 出版物数。默认值为 65535。	65535	整数
camel.component.paho-mqtt5.retained	Retain 选项。	false	布尔值
camel.component.paho-mqtt5.server-u-r-is	设置客户端可以连接到的一个或多个 serverURI 的列表。可以使用逗号分隔多个服务器。每个 serverURI 指定客户端可以连接的服务器的地址。对于 TCP 连接，支持两种类型的连接，ssl:// 用于由 SSL/TLS 安全的 TCP 连接。例如： tcp://localhost:1883 ssl://localhost:8883 如果未指定端口，则为 tcp:// URIs 默认为 1883，对于 ssl:// URIs，8883 为 8883。如果设置了 serverURIs，则它将覆盖 MQTT 客户端构造器上传递的 serverURI 参数。当尝试连接时，客户端将从列表中的第一个 serverURI 开始，并操作列表，直到与服务器建立连接为止。如果无法对任何服务器建立连接，则连接尝试会失败。指定客户端可以连接到多个用途的服务器列表：高可用性和可靠的消息交付一些 MQTT 服务器支持高可用性功能，其中两个或更多 MQTT 服务器共享状态。MQTT 客户端可以连接到任何等同的服务器，并保证无论客户端连接到哪个服务器，都可靠交付和持久化订阅。如果需要 durable 订阅和/或可靠的消息交付，则必须将 cleansession 标志设置为 false。可以指定一组不相等的服务器集合（如高可用性选项中）。因为服务器之间没有共享状态，因此消息交付和持久化订阅无效。如果使用了 hunt 列表模式，则必须将 cleansession 标志设置为 true。		字符串
camel.component.paho-mqtt5.session-expiry-interval	设置 Session Expiry Interval。这个值（以秒为单位）定义代理在客户端断开连接后维护会话的最长时间。如果客户端计划在以后的时间点连接到服务器，则客户端应仅与较长的会话过期间隔连接。默认情况下，这个值为 -1，因此不会发送，在这种情况下，会话不会过期。如果发送了 0，则会话将在网络连接关闭后立即结束。当客户端确定不再用于会话时，它应该断开连接，并将 Session Expiry Interval 设置为 0。	-1	Long
camel.component.paho-mqtt5.socket-factory	设置要使用的 SocketFactory。这允许应用程序在创建网络套接字时应用自己的策略。如果使用 SSL 连接，可以使用 SSLSocketFactory 提供应用特定的安全设置。选项是 javax.net.SocketFactory 类型。		SocketFactory

Name	描述	默认值	类型
camel.component.paho-mqtt5.ssl-client-props	设置连接的 SSL 属性。请注意，只有 Java 安全套接字扩展(JSSE)可用时，这些属性才有效。如果设置了自定义 SocketFactory，则不会使用这些属性。可以使用以下属性：com.ibm.ssl.protocol One of: SSL, SSLv3, TLS, TLSv1, SSL_TLS. com.ibm.ssl.contextProvider Underlying JSSE provider.例如，IBMJSSE2 或 SunJSSE com.ibm.ssl.keyStore 是包含您希望 KeyManager 使用的 KeyStore 对象的文件的名称。例如，/mydir/etc/key.p12 com.ibm.ssl.keyStorePassword 是您希望 KeyManager 使用的 KeyStore 对象的密码。密码可以是纯文本，也可以使用静态方法混淆：com.ibm.micro.security.Password.obfuscate （支付密码）。这使用简单不安全的 XOR 和 Base64 编码机制来混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。com.ibm.ssl.keyStoreType Type 为密钥存储，如 PKCS12、JKS 或 JCEKS. com.ibm.ssl.keyStoreProvider Key 存储供应商，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.trustStore 包含您希望 TrustManager 使用的 KeyStore 对象的文件的名称。com.ibm.ssl.trustStorePassword。您想要 TrustStore 对象的 TrustStore 对象的密码。密码可以是纯文本，也可以使用静态方法混淆：com.ibm.micro.security.Password.obfuscate （支付密码）。这使用简单不安全的 XOR 和 Base64 编码机制来混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。com.ibm.ssl.trustStoreType 是您希望默认 TrustManager 使用的 KeyStore 对象类型。与 keyStoreType. com.ibm.ssl.trustStoreProvider Trust store provider 相同的可能值，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.enabledCipherSuites A 列表启用了哪些密码。值依赖于提供程序，例如：SSL_RSA_WITH_AES_128_CBC_SHA;SSL_RSA_WITH_3DES_EDE_CBC_SHA. com.ibm.ssl.keyManager 设置用于实例化 KeyManagerFactory 对象而不是使用平台中提供的默认算法的算法。示例值：IbmX509 或 IBMJ9X509. com.ibm.ssl.trustManager 设置用于实例化 TrustManagerFactory 对象的算法，而不是使用平台中提供的默认算法。示例值：PKIX 或 IBMJ9X509. 选项是一个 java.util.Properties 类型。		Properties
camel.component.paho-mqtt5.ssl-hostname-verifier	为 SSL 连接设置 HostnameVerifier。请注意，它将在连接上的握手后使用，您应该在验证主机名错误时自行执行操作。没有默认的 HostnameVerifier。选项是 javax.net.ssl.HostnameVerifier 类型。		HostnameVerifier
camel.component.paho-mqtt5.user-name	用于针对 MQTT 代理进行身份验证的用户名。		字符串

Name	描述	默认值	类型
camel.component.paho-mqtt5.will-mqtt-properties	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。为消息设置的 MQTT 属性。选项是一个 org.eclipse.paho.mqttv5.common.packet.MqttProperties 类型。		MqttProperties
camel.component.paho-mqtt5.will-payload	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。消息的字节有效负载。		字符串
camel.component.paho-mqtt5.will-qos	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。发布消息的服务质量 (0、1 或 2)。	1	整数
camel.component.paho-mqtt5.will-retained	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。消息是否应保留。	false	布尔值
camel.component.paho-mqtt5.will-topic	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失与服务器的连接，服务器将使用提供的详细信息向其自身发布消息。要发布到的主题。		字符串

第 103 章 平台 HTTP

Since Camel 3.0

仅支持消费者

平台 HTTP 用于允许 Camel 使用运行时中的现有 HTTP 服务器，例如在 Spring Boot、Quarkus 或其他运行时运行 Camel 时。

103.1. 依赖项

当在 Camel Spring Boot 中使用 platform-http 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-platform-http-starter</artifactId>
</dependency>
```

103.2. 平台 HTTP 供应商

要使用 Platform HTTP，需要在 classpath 上提供供应商（引擎）。为不同的运行时（如 Quarkus、VertX 或 Spring Boot）具有驱动程序。

目前，只有 Quarkus 和 VertX 被 camel-platform-http-vertx 支持。这个 JAR 必须位于类路径上，否则无法使用 Platform HTTP 组件，并且在启动时抛出异常。

```
<dependency>
   <groupId>org.apache.camel</groupId>
   <artifactId>camel-platform-http-vertx</artifactId>
   <version>4.4.0.redhat-00019</version>
   <!-- use the same version as your Camel core version -->
</dependency>
```

103.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

103.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

103.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

103.4. 组件选项

平台 HTTP 组件支持 3 个选项，如下所列。

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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
引擎 (advanced)	用于服务请求的 HTTP 服务器引擎实施。		PlatformHttpEngine

103.4.1. 端点选项

平台 HTTP 端点使用 URI 语法进行配置：

```
platform-http:path
```

使用以下路径和查询参数：

103.4.1.1. 路径参数(1 参数)

Name	描述	默认值	类型
path (consumer)	**必需** 此端点提供 HTTP 请求的路径，用于代理使用 'proxy'。		字符串

103.4.1.2. 查询参数(11 参数)

Name	描述	默认值	类型
consume (consumer)	此端点的内容类型接受为输入，如 application/xml 或 application/json. null 或 / 意味着没有限制。		字符串
httpMethodRestrict (consumer)	以逗号分隔的 HTTP 方法列表，如 GET、POST。如果没有指定方法，则会提供所有方法。		字符串

Name	描述	默认值	类型
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
muteException (consumer)	如果对消费者启用并且交换失败，响应的正文不会包含异常的堆栈跟踪。	true	布尔值
generate (consumer)	此端点生成的内容类型，如 application/xml 或 application/json。		字符串
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
fileNameExtWhitelist (consumer (advanced))	以逗号分隔的文件扩展列表。具有这些扩展的上传将存储在本地。null 值或星号（）将允许所有文件。		字符串
headerFilterStrategy (advanced)	使用自定义 HeaderFilterStrategy 过滤标头到 Camel 消息。		HeaderFilterStrategy
platformHttpEngine (advanced)	用于服务此端点请求的 HTTP 服务器引擎实现。		PlatformHttpEngine

103.5. 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.platform-http.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.platform-http.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.platform-http.enabled	是否启用 platform-http 组件的自动配置。这默认是启用的。		布尔值
camel.component.platform-http.engine	用于服务请求的 HTTP 服务器引擎实施。选项是 org.apache.camel.component.platform.http.spi.PlatformHttpEngine 类型。		PlatformHttpEngine

103.5.1. 实现反向代理

平台 HTTP 组件可以充当反向代理，在这种情况下，一些标头会从 HTTP 请求的请求行中收到的绝对 URL 填充。这些标头特定于简化平台。

目前，这个功能只支持 camel-platform-http-vertx 组件中的 Vert.x。

第 104 章 PROTOBUF JACKSON

Jackson Protobuf 是一个数据格式，它使用带有 Protobuf 扩展的 Jackson 库将 Protobuf 有效负载 unmarshal a Protobuf payload to marshal Java object into a Protobuf payload。

注意

如果您熟悉 Jackson，这个 Protobuf 数据格式的行为与其 JSON 对应部分相同，因此可用于为 JSON 序列化/反序列化/序列化注解的类。

```
from("kafka:topic").
  unmarshal().protobuf(ProtobufLibrary.Jackson, JsonNode.class).
  to("log:info");
```

104.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 protobuf-jackson 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-jackson-protobuf-starter</artifactId>
</dependency>
```

104.2. 配置 SCHEMARESOLVER

由于 Protobuf 序列化是基于模式，因此此数据格式需要您提供一个 SchemaResolver 对象，该对象可以为每个将要进行 marshalled/unmarshalled 的交换查找 schema。

您可以将单个 SchemaResolver 添加到 registry 中，它将被自动查找。或者，您可以明确指定对自定义 SchemaResolver 的引用。

104.3. PROTOBUF JACKSON 选项

Protobuf Jackson dataformat 支持 18 个选项，如下所列。

Name	默认值	Java 类型	描述
contentTypeHeader		**布尔值**	数据格式是否应该使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式到 XML 的 application/xml 或用于数据格式的 application/json 发送到 JSON。
objectMapper		**字符串**	使用 Jackson 时，查找并使用带有给定 id 的现有 ObjectMapper。
useDefaultObjectMapper		**布尔值**	是否从 registry 中查找和使用默认 Jackson ObjectMapper。
unmarshalType		**字符串**	当 unmarshalling 时使用的 java 类型的类名称。
jsonView		**字符串**	当 marshalling a POJO to JSON 时，您可能想要从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现此目的。此选项是引用具有 JsonView 注释的类。
Include		**字符串**	如果您想 marshal a pojo to JSON，并且 pojo 具有一些带有 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		**布尔值**	用于 JMS 用户，以允许 JMS spec 中的 JMSType 标头指定一个 FQN 类名称来用于 unmarshal。
collectionType		**字符串**	引用要使用的自定义集合类型，以便在 registry 中查找。这个选项应该很少被使用，但允许使用与基于 java.util.Collection 不同的集合类型。
useList		**布尔值**	To unmarshal 到 Map 列表或 Pojo 的列表。
moduleClassNames		**字符串**	使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String with FQN 类名称。可以使用逗号分隔多个类。
moduleRefs		**字符串**	使用 Camel registry 中引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。
enableFeatures		**字符串**	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上启用的功能集合。这个功能应该是与 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature multiple features 分开的名称。

Name	默认值	Java 类型	描述
disableFeatures		字符串	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的功能集合。这个功能应该是与 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature multiple features 分开的名称。
allowUnmarshallType		布尔值	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。这只有在需要使用时才启用。
timezone		字符串	如果设置，则 Jackson 会在 marshalling/unmarshalling 时使用 Timezone。
autoDiscoverObjectMapper		布尔值	如果设置为 true，则 Jackson 将把 objectMapper 来查找到 registry 中。
schemaResolver		字符串	可选的模式解析器用于查找传输中数据的模式。
autoDiscoverSchemaResolver		布尔值	如果没有禁用，SchemaResolver 将查找到 registry 中。

104.4. 使用自定义 PROTOBUFMAPPER

如果需要更多控制映射配置，您可以将 JacksonProtobufDataFormat 配置为使用自定义 ProtobufMapper。

如果您在 registry 中设置单个 ProtobufMapper，则 Camel 将自动查找并使用此 ProtobufMapper。

104.5. 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.protobuf-jackson.allow-jms-type	用于 JMS 用户，以允许 JMS spec 中的 JMSType 标头指定一个 FQN 类名称来用于 unmarshal。	false	布尔值
camel.dataformat.protobuf-jackson.allow-unmarshall-type	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。这只有在需要使用时才启用。	false	布尔值
camel.dataformat.protobuf-jackson.auto-discover-object-mapper	如果设置为 true，则 Jackson 将把 objectMapper 来查找到 registry 中。	false	布尔值
camel.dataformat.protobuf-jackson.auto-discover-schema-resolver	如果没有禁用，SchemaResolver 将查找到 registry 中。	true	布尔值
camel.dataformat.protobuf-jackson.collection-type	引用要使用的自定义集合类型，以便在 registry 中查找。这个选项应该很少被使用，但允许使用与基于 java.util.Collection 不同的集合类型。		字符串
camel.dataformat.protobuf-jackson.content-type-header	数据格式是否应该使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式到 XML 的 application/xml 或用于数据格式的 application/json 发送到 JSON。	true	布尔值
camel.dataformat.protobuf-jackson.disable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的功能集合。这个功能应该是与 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature multiple features 分开的名称。		字符串
camel.dataformat.protobuf-jackson.enable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上启用的功能集合。这个功能应该是与 com.fasterxml.jackson.databind.SerializationFeature, com.fasterxml.jackson.databind.DeserializationFeature, 或 com.fasterxml.jackson.databind.MapperFeature multiple features 分开的名称。		字符串

Name	描述	默认值	类型
camel.dataformat.protobuf-jackson.enabled	是否启用 protobuf-jackson 数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.protobuf-jackson.include	如果您想 marshal a pojo to JSON，并且 pojo 具有一些带有 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.protobuf-jackson.json-view	当 marshalling a POJO to JSON 时，您可能想要从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现此目的。此选项是引用具有 JsonView 注释的类。		字符串
camel.dataformat.protobuf-jackson.module-class-names	使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为 String with FQN 类名称。可以使用逗号分隔多个类。		字符串
camel.dataformat.protobuf-jackson.module-refs	使用 Camel registry 中引用的自定义 Jackson 模块。可以使用逗号分隔多个模块。		字符串
camel.dataformat.protobuf-jackson.object-mapper	使用 Jackson 时，查找并使用带有给定 id 的现有 ObjectMapper。		字符串
camel.dataformat.protobuf-jackson.schema-resolver	可选的模式解析器用于查找传输中数据的模式。		字符串
camel.dataformat.protobuf-jackson.timezone	如果设置，则 Jackson 会在 marshalling/unmarshalling 时使用 Timezone。		字符串
camel.dataformat.protobuf-jackson.unmarshal-type	当 unmarshalling 时使用的 java 类型的类名称。		字符串
camel.dataformat.protobuf-jackson.use-default-object-mapper	是否从 registry 中查找和使用默认 Jackson ObjectMapper。	true	布尔值

Name	描述	默认值	类型
camel.dataformat.protobuf-jackson.use-list	To unmarshal 到 Map 列表或 Pojo 的列表。	false	布尔值

第 105 章 QUARTZ

仅支持消费者

Quartz 组件使用 Quartz Scheduler 2.x 提供计划的消息传输。每个端点代表一个不同的计时器（在 Quartz 术语中，一个 Trigger 和 JobDetail）。

105.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 quartz 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-quartz-starter</artifactId>
</dependency>
```

105.2. 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。

105.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

105.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

105.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

105.4. 组件选项

Quartz 组件支持 13 个选项，如下所列。

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
enableJmx (consumer)	是否启用 Quartz JMX，允许从 JMX 管理 Quartz 调度程序。这个选项默认为 true。	true	布尔值

Name	描述	默认值	类型
prefixInstanceName (consumer)	是否使用 CamelContext 名称作为 Quartz Scheduler 实例名称作为前缀。这默认是启用的，以便每个 CamelContext 默认使用自己的 Quartz 调度程序实例。您可以将此选项设置为 false，在多个 CamelContext 之间重复使用 Quartz 调度程序实例。	true	布尔值
prefixJobNameWithEndpointId (consumer)	是否使用端点 ID 为 quartz 任务添加前缀。这个选项默认为 false。	false	布尔值
properties (consumer)	配置 Quartz 调度程序的属性。		Map
propertiesFile (consumer)	从 classpath 加载的属性的文件名。		字符串
propertiesRef (consumer)	引用 registry 中要用来配置 quartz 的现有属性或映射查找。		字符串
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
scheduler (advanced)	要使用配置的 Quartz 调度程序，而不是创建新的调度程序。		scheduler
schedulerFactory (advanced)	使用自定义 SchedulerFactory，用于创建调度程序。		SchedulerFactory
autoStartScheduler (scheduler)	调度程序是否应自动启动。这个选项默认为 true。	true	布尔值
interruptJobsOnShutdown (scheduler)	是否在关闭时中断作业，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可能会因为中断而失败。当作业中断时，Camel 会将交换标记为停止路由，并将 java.util.concurrent.RejectedExecutionException 设置为 caused exception。因此，请谨慎使用它，因为它通常最好允许 Camel 作业正常完成和关闭。	false	布尔值
startDelayedSeconds (scheduler)	启动 quartz 调度程序前等待的秒数。		int

105.5. 端点选项

Quartz 端点使用 URI 语法进行配置：

```
quartz:groupName/triggerName
```

使用以下路径和查询参数：

105.5.1. 路径参数(2 参数)

Name	描述	默认值	类型
groupname (consumer)	要使用的 quartz 组名称。组名称和触发器名称的组合应该是唯一的。	Camel	字符串
triggerName (consumer)	***必需*** 要使用的 quartz 触发器名称。组名称和触发器名称的组合应该是唯一的。		字符串

105.5.2. 查询参数(17 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)	指定定义何时触发的 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。否则，如果设为 false，它将保留在调度程序中。当设置为 false 时，这也意味着用户也可以使用 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)	作业是否应在孤立后仍然保持存储（没有触发器指向该作业）。	false	布尔值

Name	描述	默认值	类型
pauseJob (consumer)	如果设置为 true，则触发器会在路由停止时自动暂停。否则，如果设为 false，它将保留在调度程序中。当设置为 false 时，这也意味着用户也可以使用 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，指示调度程序是否应该重新执行该作业。	false	布尔值
stateful (consumer)	使用 Quartz PersistJobDataAfterExecution 和 DisallowConcurrentExecution，而不是默认的作业。	false	布尔值
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
customCalendar (advanced)	指定自定义日历以避免特定日期范围。		日历
jobParameters (advanced)	要在作业上配置附加选项。		Map
prefixJobNameWithEndpointId (advanced)	作业名称是否应该带有端点 id 前缀。	false	布尔值
triggerParameters (advanced)	要在触发器上配置附加选项。		Map
usingFixedCamelContextName (advanced)	如果为 true，JobDataMap 使用 CamelContext 名称直接引用 CamelContext，如果为 false，则 JobDataMap 使用 CamelContext 管理名称，该名称可以在部署时更改。	false	布尔值
autoStartScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值

Name	描述	默认值	类型
startDelayedSeconds (scheduler)	启动 quartz 调度程序前等待的秒数。		int
triggerStartDelay (scheduler)	如果调度程序已启动，我们希望触发器在当前时间后稍有启动，以确保在作业启动前完全启动端点。负值转换过去触发器开始时间。	500	long

105.5.3. 配置 quartz.properties 文件

默认情况下，Qartz 将在 classpath 的 org/quartz 目录中查找 quartz.properties 文件。如果您使用 WAR 部署，这意味着仅丢弃 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**	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="quartz" class="org.apache.camel.component.quartz.QuartzComponent">
    <property name="propertiesFile" value="com/mycompany/myquartz.properties"/>
</bean>
```

105.6. 在 JMX 中启用 QUARTZ 调度程序

您需要配置 quartz 调度程序属性以启用 JMX。
这通常将选项 "org.quartz.scheduler.jmx.export" 设置为配置文件中的 true 值。

除非明确禁用，否则此选项默认设置为 true。

105.7. 启动 QUARTZ 调度程序

Quartz 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。

例如：

```
<bean id="quartz" class="org.apache.camel.component.quartz.QuartzComponent">
    <property name="startDelayedSeconds" value="5"/>
</bean>
```

105.8. 集群

如果您在集群模式中使用 Quartz，如 JobStore 是集群的。然后，当节点停止/关闭时，Qartz 组件不会暂停/删除触发器。https://access.redhat.com/documentation/zh-cn/red_hat_build_of_apache_camel/4.4/html-single/red_hat_build_of_apache_camel_for_spring_boot_reference/index#csb-camel-quartz-component-starter这允许触发器在集群中的其他节点上运行。

注意

在集群节点中运行时，不会进行检查以确保端点的唯一作业名称/组。

105.9. 消息标头

Camel 将 Quartz Execution Context 中的 getters 添加为标头值。以下标头被添加：calendar,fireTime,jobDetail,jobInstance,jobRuntTime,mergedJobDataMap,nextFireTime,previousFireTime,refireCount,result,scheduledFireTime,scheduler, triggerName ,triggerName,triggerGroup.

fireTime 标头包含交换触发时的 java.util.Date。

105.10. 使用 CRON TRIGGERS

quartz 支持与 Cron 类似表达式，用于以方便格式指定计时器。您可以在 cron URI 参数中使用这些表达式；但是，要保留有效的 URI 编码，我们允许使用 + 而不是空格。

例如，以下将每五分钟触发一次消息，从 12pm (noon)开始到每周天的 6pm：

```
from("quartz://myGroup/myTimerName?cron=0+0/5+12-18+?+*+MON-FRI")
    .to("activemq:Totally.Rocks");
```

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
+	空格

105.11. 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，您可以执行以下操作：

```
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。

105.12. 配置错误指令

可以使用错误指令配置 quartz 调度程序，以处理触发器的错误情况。您使用的 concrete 触发器类型将定义一组额外的 MISFIRE_INSTRUCTION_XXX 常数，这些常量可以设置为此属性的值。

例如，将简单触发器配置为使用错误指令 4：

```
quartz://myGroup/myTimerName?trigger.repeatInterval=2000&trigger.misfireInstruction=4
```

同样，您也可以使用其错误指令之一配置 cron 触发器：

```
quartz://myGroup/myTimerName?cron=0/2+*+*+*+*+?&trigger.misfireInstruction=2
```

simple 和 cron 触发器有以下错误指令代表：

105.12.1. SimpleTrigger.MISFIRE_INSTRUCTION_FIRE_NOW = 1 (default)

指示调度程序在错误错误的情况下，现在希望由调度程序触发 SimpleTrigger。

这个指令通常只用于 'one-shot'（非重复）触发器。如果在带有重复计数 > 0 的触发器上使用它，则等同于说明 MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_REMAINING_REPEAT_COUNT。

105.12.2. SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_EXISTING_REPEAT_COUNT = 2

指示调度程序在误报时，simpleTrigger 希望重新调度到"现在"（即使相关的 Calendar 排除了"now"），且按原样重复计数。这会模糊处理 Trigger 端到端，因此如果 'now' 在终止时间后，则 Trigger 不会再次触发。

使用这个指令会导致触发器"forget"开始时间和重复次数，即最初设置它时的重复计数（这只在某些原因希望能够告知以后的原始值时出现问题）。

105.12.3. SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_REMAINING_REPEAT_COUNT = 3

指示调度程序在误报时，simpleTrigger 希望重新调度到"现在"（即使相关的 Calendar 排除了"现在"），且重复计数设置为该情况（如果尚未错过任何触发）。这会模糊处理 Trigger 端到端，因此如果 'now' 在终止时间后，则 Trigger 不会再次触发。

使用这个指令，可让触发器"forget"开始时间和重复计数（最初通过.相反，触发器重复计数将更改为剩余的重复计数（这只在某些原因希望能够告知以后的原始值时出现问题）。

这个指令可能会导致 Trigger 在触发 'now' 后进入 'COMPLETE' 状态，如果所有重复的时间都丢失了。

105.12.4. SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_REMAINING_COUNT = 4

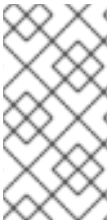
指示调度程序在误报时，simpleTrigger 希望在"现在"后重新调度到下一次调度时间 - 考虑任何关联的 Calendar，并且重复数设置为该情况（如果未错过任何触发）。

注意

如果所有丢失的触发时间，则此指令可能会导致 Trigger 直接进入 'COMPLETE' 状态。

105.12.5. SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_EXISTING_COUNT = 5

指示调度程序在误报时，simpleTrigger 希望在"现在"后重新调度到下一次调度时间 - 考虑任何关联的 Calendar，重复计数保持不变。



注意

如果触发器的结束时间已到达，则这个指令可能会导致 Trigger 直接进入 'COMPLETE' 状态。

105.12.6. CronTrigger.MISFIRE_INSTRUCTION_FIRE_ONCE_NOW = 1 (default)

指示调度程序在错误错误的情况下，现在希望由调度程序触发 CronTrigger。

105.12.7. CronTrigger.MISFIRE_INSTRUCTION_DO_NOTHING = 2

指示调度程序在误报时更新，C CronTrigger 希望在当前时间（考虑任何关联的 Calendar ）之后，C CronTrigger 希望将其更新为下一次调度中。

105.13. 使用 QUARTZSCHEDULEDPOLLCONSUMERSCHEDULER

Quartz 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。

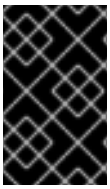
例如，要使用基于 cron 的表达式每第 2 秒轮询一次文件，可将 Camel 路由定义为：

```
from("file:inbox?scheduler=quartz&scheduler.cron=0/2+*+*+*+*+?")
  .to("bean:process");
```

请注意，我们定义 scheduler=quartz，以指示 Camel 使用基于 Quartz 的调度程序。然后，我们使用 scheduler.xxx 选项来配置调度程序。Quartz 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，则使用组件中的共享调度程序。
cron	null	字符串	必需 ：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，则生成和使用 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

记住从端点 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=quartz&scheduler.cron=0/2+*+*+*+*+?
&scheduler.triggerId=myId&scheduler.triggerGroup=myGroup")
        .to("bean:process");
```

Spring 中有一个 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");
```

105.14. CRON 组件支持

Quartz 组件可用作 Camel Cron 组件的实施。

Maven 用户需要将以下额外依赖项添加到其 pom.xml 中：

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

然后，用户可以使用 cron 组件而不是 quartz 组件，如以下路由中所示：

```
from("cron://name?schedule=0+0/5+12-18+?+*+MON-FRI")
.to("activemq:Totally.Rocks");
```

105.15. 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.quartz.auto-start-scheduler	调度程序是否应自动启动。这个选项默认为 true。	true	布尔值
camel.component.quartz.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.quartz.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.quartz.enable-jmx	是否启用 Quartz JMX，允许从 JMX 管理 Quartz 调度程序。这个选项默认为 true。	true	布尔值

Name	描述	默认值	类型
camel.component.quartz.enabled	是否启用 quartz 组件的自动配置。这默认是启用的。		布尔值
camel.component.quartz.interrupt-jobs-on-shutdown	是否在关闭时中断作业，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可能会因为中断而失败。当作业中断时，Camel 会将交换标记为停止路由，并将 java.util.concurrent.RejectedExecutionException 设置为 caused exception。因此，请谨慎使用它，因为它通常最好允许 Camel 作业正常完成和关闭。	false	布尔值
camel.component.quartz.prefix-instance-name	是否使用 CamelContext 名称作为 Quartz Scheduler 实例名称作为前缀。这默认是启用的，以便每个 CamelContext 默认使用自己的 Quartz 调度程序实例。您可以将此选项设置为 false，在多个 CamelContext 之间重复使用 Quartz 调度程序实例。	true	布尔值
camel.component.quartz.prefix-job-name-with-endpoint-id	是否使用端点 ID 为 quartz 任务添加前缀。这个选项默认为 false。	false	布尔值
camel.component.quartz.properties	配置 Quartz 调度程序的属性。		Map
camel.component.quartz.properties-file	从 classpath 加载的属性的文件名。		字符串
camel.component.quartz.properties-ref	引用 registry 中要用来配置 quartz 的现有属性或映射查找。		字符串
camel.component.quartz.scheduler	要使用配置的 Quartz 调度程序，而不是创建新的调度程序。选项是 org.quartz.Scheduler 类型。		scheduler
camel.component.quartz.scheduler-factory	使用自定义 SchedulerFactory，用于创建调度程序。选项是一个 org.quartz.SchedulerFactory 类型。		SchedulerFactory
camel.component.quartz.start-delayed-seconds	启动 quartz 调度程序前等待的秒数。		整数

第 106 章 REF

支持生成者和消费者

Ref 组件用于查找 Registry 中绑定的现有端点。

106.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 ref 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-ref-starter</artifactId>
</dependency>
```

106.2. URI 格式

```
ref:someName[?options]
```

其中 someName 是 Registry 中的端点的名称（通常为，但并不总是是 Spring registry）。如果使用 Spring registry，someName 将是 Spring registry 中端点的 bean ID。

106.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

106.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

106.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

106.4. 组件选项

Ref 组件支持 3 个选项，如下所列。

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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

106.5. 端点选项

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：

106.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
name (common)	在 registry 中查找的端点 **必需** 名称。		字符串

106.5.2. 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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		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

106.6. 运行时查找

当您需要在 Registry 中动态发现端点时，可以使用此组件来在运行时计算 URI。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);
```

您可以在 Registry 中定义端点列表，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>
```

106.7. 示例

在以下示例中，我们使用 URI 中的 ref: 使用 spring ID endpoint2 引用端点：

当然，您可以使用 ref 属性：

```
<to uri="ref:endpoint2"/>
```

这是编写它更常见的方法。

106.8. 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.ref.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.ref.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.ref.enabled	是否启用 ref 组件的自动配置。这默认是启用的。		布尔值
camel.component.ref.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

第 107 章 REF

Ref 表达式语言实际上只是一种从 Registry 中查询自定义 Expression 或 Predicate 的方式。

这在 XML DSL 中特别可用。

107.1. 依赖项

Ref 语言是 camel-core 的一部分。当在 Red Hat build of Camel Spring Boot 中使用 ref 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-core-starter</artifactId>
</dependency>
```

107.2. REF LANGUAGE 选项

Ref 语言支持 1 个选项，如下所列。

Name	默认值	Java 类型	描述
trim		布尔值	是否修剪值以移除前导和结尾的空格和换行符。

107.3. 用法示例

XML DSL 中的 Splitter EIP 可使用 < ref> 来使用 自定义表达式，如下所示：

```
<bean id="myExpression" class="com.mycompany.MyCustomExpression"/>

<route>
  <from uri="seda:a"/>
  <split>
    <ref>myExpression</ref>
    <to uri="mock:b"/>
  </split>
</route>
```

在这种情况下，来自 seda:a 端点的消息将使用在 Registry 中的 id 为myExpression 的自定义

Expression 分割。

使用 Java DSL 的同一示例：

```
from("seda:a").split().ref("myExpression").to("seda:b");
```

107.4. SPRING BOOT AUTO-CONFIGURATION

组件支持 147 选项，如下所列。

Name	描述	默认值	类型
camel.cloud.consul.service-discovery.acl-token	设置用于 Consul 的 ACL 令牌。		字符串
camel.cloud.consul.service-discovery.block-seconds	等待监视事件的秒数，默认为 10 秒。	10	整数
camel.cloud.consul.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.consul.service-discovery.connect-timeout-millis	OkHttpClient 的连接超时。		Long
camel.cloud.consul.service-discovery.datacenter	数据中心。		字符串
camel.cloud.consul.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.consul.service-discovery.password	设置用于基本身份验证的密码。		字符串

Name	描述	默认值	类型
camel.cloud.consul.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.consul.service-discovery.read-timeout-millis	OkHttpClient 的读取超时。		Long
camel.cloud.consul.service-discovery.url	Consul 代理 URL。		字符串
camel.cloud.consul.service-discovery.user-name	设置用于基本身份验证的用户名。		字符串
camel.cloud.consul.service-discovery.write-timeout-millis	OkHttpClient 的写入超时。		Long
camel.cloud.dns.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.dns.service-discovery.domain	域名；		字符串
camel.cloud.dns.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.dns.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.dns.service-discovery.proto	所需服务的传输协议。	_tcp	字符串

Name	描述	默认值	类型
camel.cloud.etcd.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.etcd.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.etcd.service-discovery.password	用于基本身份验证的密码。		字符串
camel.cloud.etcd.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.etcd.service-discovery.service-path	查找服务发现的路径。	/services/	字符串
camel.cloud.etcd.service-discovery.timeout	要设置操作可以采取的最长时间，请执行以下操作：		Long
camel.cloud.etcd.service-discovery.type	要设置发现类型，有效值为 on-demand 和 watch。	按需	字符串
camel.cloud.etcd.service-discovery.uris	客户端可以连接到的 URI。		字符串
camel.cloud.etcd.service-discovery.username	用于基本身份验证的用户名。		字符串
camel.cloud.kubernetes.service-discovery.api-version	使用客户端查找时设置 API 版本。		字符串

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.ca-cert-data	使用客户端查找时设置证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.ca-cert-file	在使用客户端查找时，设置从文件加载的证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-data	使用客户端查找时设置客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-file	在使用客户端查找时，设置从文件加载的客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-algo	设置客户端密钥存储算法，如使用客户端查找时 RSA。		字符串
camel.cloud.kubernetes.service-discovery.client-key-data	使用客户端查找时设置客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-file	在使用客户端查找时，设置从文件加载的客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-passphrase	使用客户端查找时设置客户端密钥存储密码短语。		字符串
camel.cloud.kubernetes.service-discovery.configurations	定义其他配置定义。		Map

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.dns-domain	设置用于 DNS 查找的 DNS 域。		字符串
camel.cloud.kubernetes.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.kubernetes.service-discovery.lookup	如何执行服务查找。可能的值有：client、dns、environment。在使用客户端时，客户端会查询 kubernetes master 来获取提供该服务的活跃 pod 列表，然后随机（或循环）选择一个 pod。当使用 dns 时，服务名称被解析为 name.namespace.svc.dnsDomain。当使用 dnssrv 时，服务名称使用 SRV 查询解析 ....svc... when using environment，环境变量用于查找服务。默认情况下使用环境。	环境	字符串
camel.cloud.kubernetes.service-discovery.master-url	在使用客户端查找时，将 URL 设置为 master。		字符串
camel.cloud.kubernetes.service-discovery.namespace	设置要使用的命名空间。默认情况下，将使用来自 ENV 变量 KUBERNETES_MASTER 的命名空间。		字符串
camel.cloud.kubernetes.service-discovery.oauth-token	在使用客户端查找时，为身份验证设置 OAUTH 令牌（而不是用户名/密码）。		字符串
camel.cloud.kubernetes.service-discovery.password	在使用客户端查找时设置用于身份验证的密码。		字符串
camel.cloud.kubernetes.service-discovery.port-name	设置用于 DNS/DNSSRV 查找的端口名称。		字符串
camel.cloud.kubernetes.service-discovery.port-protocol	设置用于 DNS/DNSSRV 查找的端口协议。		字符串

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.kubernetes.service-discovery.trust-certs	设置在使用客户端查找时是否打开信任证书检查。	false	布尔值
camel.cloud.kubernetes.service-discovery.username	在使用客户端查找时设置用于身份验证的用户名。		字符串
camel.cloud.ribbon.load-balancer.client-name	设置 Ribbon 客户端名称。		字符串
camel.cloud.ribbon.load-balancer.configurations	定义其他配置定义。		Map
camel.cloud.ribbon.load-balancer.enabled	启用组件。	true	布尔值
camel.cloud.ribbon.load-balancer.namespace	命名空间。		字符串
camel.cloud.ribbon.load-balancer.password	密码。		字符串
camel.cloud.ribbon.load-balancer.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map

Name	描述	默认值	类型
camel.cloud.ribbon.load-balancer.username	用户名。		字符串
camel.hystrix.allow-maximum-size-to-diverge-from-core-size	允许配置使 maximumSize 生效。然后该值可以等于或大于 coreSize。	false	布尔值
camel.hystrix.circuit-breaker-enabled	是否使用 HystrixCircuitBreaker。如果为 false，则不会使用 断路器逻辑，并且所有允许的请求。这与 circuitBreakerForceClosed （）的影响类似，除非继续跟踪指标，知道它是否应该是 open/closed，此属性即使实例化一个断路器。	true	布尔值
camel.hystrix.circuit-breaker-error-threshold-percentage	错误百分比阈值（如 50 ）指向断路器将打开和拒绝请求。它将在 circuitBreakerSleepWindowInMilliseconds 中定义的持续时间保持出差；与 HystrixCommandMetrics.getHealthCounts （）进行比较的错误百分比。	50	整数
camel.hystrix.circuit-breaker-force-closed	如果为 true，HystrixCircuitBreaker#allowRequest （）将始终返回 true 以允许请求，无论 HystrixCommandMetrics.getHealthCounts （）的错误百分比如何。如果设为 true，则 circuitBreakerForceOpen （）属性具有优先权。	false	布尔值
camel.hystrix.circuit-breaker-force-open	如果为 true，HystrixCircuitBreaker.allowRequest （）将始终返回 false，从而导致电路变为开路（接受），并拒绝所有请求。此属性优先于 circuitBreakerForceClosed （）;。	false	布尔值
camel.hystrix.circuit-breaker-request-volume-threshold	metricsRollingStatisticalWindowInMilliseconds （）中的最少请求数必须存在于 HystrixCircuitBreaker 之前。如果此数字低于这个数字，无论错误百分比如何，电路都不会被出差。	20	整数
camel.hystrix.circuit-breaker-sleep-window-in-milliseconds	HystrixCircuitBreaker trips 之后的时间（以毫秒为单位），它应该在尝试请求前等待。	5000	整数
camel.hystrix.configurations	定义其他配置定义。		Map
camel.hystrix.core-pool-size	传递给 java.util.concurrent.ThreadPoolExecutor#setCorePoolSize (int)的核心 thread-pool 大小。	10	整数

Name	描述	默认值	类型
camel.hystrix.enabled	启用组件。	true	布尔值
camel.hystrix.execution-isolation-semaphore-max-concurrent-requests	允许 HystrixCommand.run （）的并发请求数。超过并发限制的请求将被拒绝。仅在执行IsolationStrategy == SEMAPHORE 时使用。	20	整数
camel.hystrix.execution-isolation-strategy	将通过什么隔离策略 HystrixCommand.run （）执行。如果 THREAD，它将在单独的线程上执行，并且受 thread-pool 中的线程数量限制的并发请求。如果 SEMAPHORE，它将在调用线程上执行，并且受 semaphore 数限制的并发请求。	线程	字符串
camel.hystrix.execution-isolation-thread-interrupt-on-timeout	当线程超时时，执行线程是否应该尝试中断（使用 future#cancel）。仅在执行IsolationStrategy （）== THREAD 时才适用。	true	布尔值
camel.hystrix.execution-timeout-enabled	此命令是否启用了超时机制。	true	布尔值
camel.hystrix.execution-timeout-in-milliseconds	以毫秒为单位，将命令超时和停止执行的时间（以毫秒为单位）。如果 executionIsolationThreadInterruptOnTimeout == true 且命令是线程隔离，则执行线程将中断。如果命令是 semaphore-isolated 和 HystrixObservableCommand，则该命令将被取消订阅。	1000	整数
camel.hystrix.fallback-enabled	出现故障时，是否应尝试 HystrixCommand.getFallback （）。	true	布尔值
camel.hystrix.fallback-isolation-semaphore-max-concurrent-requests	允许 HystrixCommand.getFallback （）的并发请求数。超过并发限制的请求将快速失败，且不会尝试检索回退。	10	整数
camel.hystrix.group-key	设置要使用的 group 键。默认值为 CamelHystrix。	Camel Hystrix	字符串
camel.hystrix.keep-alive-time	更长的时间（以分钟为单位）传递给 ThreadPoolExecutor#setKeepAliveTime (long,TimeUnit)。	1	整数

Name	描述	默认值	类型
camel.hystrix.max-queue-size	在 HystrixConcurrencyStrategy.getBlockingQueue (int)中传递给 BlockingQueue 的最大队列大小应该只影响 threadpool 的实例化 - 它不会立即更改队列大小。为此，请使用 queueSizeRejectionThreshold （）。	-1	整数
camel.hystrix.maximum-size	传递给 ThreadPoolExecutor#setMaximumPoolSize (int)的最大 thread-pool 大小。这是可在不开始拒绝 HystrixCommands 的情况下支持的最大并发数量。请注意，只有在您也设置了 allowMaximumSizeToDivergeFromCoreSize 时，此设置才会生效。	10	整数
camel.hystrix.metrics-health-snapshot-interval-in-milliseconds	在允许计算成功和错误百分比时等待的时间（以毫秒为单位），并影响 HystrixCircuitBreaker.isOpen （）状态。在高容量电路上，错误百分比的连续计算可能会成为 CPU 密集型，从而控制其计算的频率。	500	整数
camel.hystrix.metrics-rolling-percentile-bucket-size	滚动百分比的每个存储桶中存储的最大值数。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	10	整数
camel.hystrix.metrics-rolling-percentile-enabled	是否应该使用 HystrixRollingPercentile 内部 HystrixCommandMetrics 来捕获百分比的指标。	true	布尔值
camel.hystrix.metrics-rolling-percentile-window-buckets	滚动窗口的存储桶数量被分成。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	6	整数
camel.hystrix.metrics-rolling-percentile-window-in-milliseconds	以毫秒为单位的滚动窗口的持续时间。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	10000	整数
camel.hystrix.metrics-rolling-statistical-window-buckets	滚动统计窗口划分为的 bucket 数量。这在 HystrixCommandMetrics 中被传递给 HystrixRollingNumber。	10	整数
camel.hystrix.metrics-rolling-statistical-window-in-milliseconds	此属性设置统计滚动窗口的持续时间，以毫秒为单位。这是为线程池保留指标的时间。窗口被分成 bucket，按这些增量回滚。	10000	整数

Name	描述	默认值	类型
camel.hystrix.queue-size-rejection-threshold	队列大小拒绝阈值是 artificial max size，即使尚未达到 maxQueueSize，也会发生拒绝。这是因为 BlockingQueue 的 maxQueueSize 无法动态更改，我们希望动态更改影响拒绝的队列大小。在排队线程以进行执行时，HystrixCommand 会使用它。	5	整数
camel.hystrix.request-log-enabled	HystrixCommand 执行和事件是否应记录到 HystrixRequestLog。	true	布尔值
camel.hystrix.thread-pool-key	设置要使用的线程池密钥。默认情况下，将使用与 groupKey 配置相同的值。	Camel Hystrix	字符串
camel.hystrix.thread-pool-rolling-number-statistical-window-buckets	滚动统计窗口划分为的 bucket 数量。这会传递到每个 HystrixThreadPoolMetrics 实例内的 HystrixRollingNumber。	10	整数
camel.hystrix.thread-pool-rolling-number-statistical-window-in-milliseconds	统计滚动窗口的持续时间（以毫秒为单位）。这会传递到每个 HystrixThreadPoolMetrics 实例内的 HystrixRollingNumber。	10000	整数
camel.language.constant.enabled	是否启用恒定语言的自动配置。这默认是启用的。		布尔值
camel.language.constant.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.csimple.enabled	是否启用 csimple 语言的自动配置。这默认是启用的。		布尔值
camel.language.csimple.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.exchangeproperty.enabled	是否启用 exchangeProperty 语言的自动配置。这默认是启用的。		布尔值
camel.language.exchangeproperty.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.file.enabled	是否启用文件语言的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.language.file.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.header.enabled	是否启用标头语言的自动配置。这默认是启用的。		布尔值
camel.language.header.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.ref.enabled	是否启用 ref 语言的自动配置。这默认是启用的。		布尔值
camel.language.ref.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.simple.enabled	是否启用简单语言的自动配置。这默认是启用的。		布尔值
camel.language.simple.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.tokenize.enabled	是否启用令牌化语言的自动配置。这默认是启用的。		布尔值
camel.language.tokenize.group-delimiter	设置在分组时要使用的分隔符。如果没有设置，则令牌将用作分隔符。		字符串
camel.language.tokenize.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.resilience4j.automatic-transition-from-open-to-half-open-enabled	在通过 waitDurationInOpenState 后，启用从 OPEN 自动过渡到 HALF_OPEN 状态。	false	布尔值
camel.resilience4j.circuit-breaker-ref	代表现有的 io.github.resilience4j.circuitbreaker.CircuitBreaker 实例从 registry 中查找和使用。使用此选项时，不使用任何其他断路器选项。		字符串
camel.resilience4j.config-ref	指的是现有的 io.github.resilience4j.circuitbreaker.CircuitBreakerConfig 实例，以便从 registry 中查找和使用。		字符串

Name	描述	默认值	类型
camel.resilience4j.configurations	定义其他配置定义。		Map
camel.resilience4j.enabled	启用组件。	true	布尔值
camel.resilience4j.failure-rate-threshold	以百分比为单位配置故障率阈值。如果失败率相等或大于阈值，则 CircuitBreaker 过渡到 open，并启动短路调用。阈值必须大于 0，而不是大于 100。默认值为 50 百分比。		æµ®ç‚¹å€¼
camel.resilience4j.minimum-number-of-calls	在 CircuitBreaker 可以计算错误率之前，配置所需的最少调用数（每个滑动期限）。例如，如果 minimumNumberOfCalls 为 10，则必须至少记录 10 个调用，然后才能计算失败率。如果只记录了 9 个调用，则 CircuitBreaker 不会过渡到 open，即使所有 9 调用都失败。默认 minimumNumberOfCalls 为 100。	100	整数
camel.resilience4j.permitted-number-of-calls-in-half-open-state	配置 CircuitBreaker 为一半打开时允许的调用数量。大小必须大于 0。默认大小为 10。	10	整数
camel.resilience4j.sliding-window-size	配置滑动窗口的大小，该窗口用于在 CircuitBreaker 关闭时记录调用的结果。slidingWindowSize 配置滑动窗口的大小。滑动窗口可以是基于计数或基于时间的窗口。如果 slidingWindowType 是 COUNT_BASED，则最后的 slidingWindowSize 调用会被记录并聚合。如果 slidingWindowType 是 TIME_BASED，则最后 slidingWindowSize 秒的调用会被记录并聚合。slidingWindowSize 必须大于 0。minimumNumberOfCalls 必须大于 0。如果 slidingWindowType 是 COUNT_BASED，则 minimumNumberOfCalls 不能大于 slidingWindowSize。如果 slidingWindowType 是 TIME_BASED，您可以选择任何您需要的。默认 slidingWindowSize 为 100。	100	整数
camel.resilience4j.sliding-window-type	配置滑动窗口的类型，用于记录 CircuitBreaker 关闭时调用的结果。滑动窗口可以是基于计数或基于时间的窗口。如果 slidingWindowType 是 COUNT_BASED，则最后的 slidingWindowSize 调用会被记录并聚合。如果 slidingWindowType 是 TIME_BASED，则最后 slidingWindowSize 秒的调用会被记录并聚合。默认 slidingWindowType 是 COUNT_BASED。	COUNT_BASED	字符串

Name	描述	默认值	类型
camel.resilience4j.slow-call-duration-threshold	配置上面的持续时间阈值（秒），调用被视为缓慢，并增加较慢的调用百分比。默认值为 60 秒。	60	整数
camel.resilience4j.slow-call-rate-threshold	以百分比为单位配置阈值。当调用持续时间大于 slowCallDurationThreshold Duration 时，CircuitBreaker 会将调用视为较慢。当较慢的调用百分比相等或大于阈值时，CircuitBreaker 过渡到 open，并启动短路调用。阈值必须大于 0，而不是大于 100。默认值为 100 百分比，这意味着所有记录的调用都必须比 slowCallDurationThreshold 慢。		æµ®ç‚¹â€¼
camel.resilience4j.wait-duration-in-open-state	配置等待持续时间（以秒为单位），指定 CircuitBreaker 应该保持打开的时间，然后再切换到半次。默认值为 60 秒。	60	整数
camel.resilience4j.writable-stack-trace-enabled	启用可写入堆栈跟踪。当设置为 false 时，Exception.getStackTrace 返回一个零长度数组。当断路器处于开路状态时，这可用于减少日志垃圾邮件，因为存在例外的原因（断路器是短路调用）。	true	布尔值
camel.rest.api-component	用作 REST API 的 Camel 组件名称（如 swagger）如果没有明确配置 API 组件，则 Camel 会查找负责服务并生成 REST API 文档的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestApiProcessorFactory。如果找到其中任何一个，则使用它。		字符串
camel.rest.api-context-path	设置领导的 API 上下文路径将使用的 REST API 服务。这在使用使用 context-path 部署 Web 应用程序的组件（如 camel-servlet ）时使用它。		字符串
camel.rest.api-context-route-id	设置用于服务 REST API 的路由的路由 ID。默认情况下，路由将使用自动分配的路由 ID。		字符串
camel.rest.api-host	要将特定主机名用于 API 文档（如 swagger），这可用于用这个配置的主机名覆盖生成的主机。		字符串
camel.rest.api-property	允许为 api 文档配置任意数量的附加属性(swagger)。例如，将属性 api.title 设置为我的冷却。		Map
camel.rest.api-vendor-extension	是否在 Rest API 中启用供应商扩展。如果启用，Camel 将包含额外信息作为厂商扩展名（例如，以 x- 开头的键），如路由 ID、类名称等。在导入 API 文档时，并非所有第三方 API 网关和工具都支持 vendor-extensions。	false	布尔值

Name	描述	默认值	类型
camel.rest.binding-mode	设置要使用的绑定模式。默认值为 off。		RestBindingMode
camel.rest.client-request-validation	是否启用客户端请求验证，以检查客户端的 Content-Type 和 Accept 标头是否受到其 consume/produces 设置的 Rest-DSL 配置的支持。这可以打开，以启用此检查。如果验证错误，则返回 HTTP Status code 415 或 406。默认值为 false。	false	布尔值
camel.rest.component	用于 REST 传输(consumer)的 Camel Rest 组件，如 netty-http, jetty, servlet, undertow。如果没有明确配置组件，则 Camel 会查找是否有与 Rest DSL 集成的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestConsumerFactory。如果找到其中任何一个，则使用它。		字符串
camel.rest.component-property	允许为正在使用的其他组件配置任意数量的附加属性。		Map
camel.rest.consumer-property	允许为使用中的其他使用者配置任意数量的附加属性。		Map
camel.rest.context-path	设置 REST 服务将使用的前导上下文路径。这在使用使用 context-path 部署 Web 应用程序的组件（如 camel-servlet ）时使用它。或者对于包含 HTTP 服务器的 camel-jetty 或 camel-netty-http 等组件。		字符串
camel.rest.cors-headers	允许配置自定义 CORS 标头。		Map
camel.rest.data-format-property	允许为使用的数据格式配置多个额外属性。例如，将属性 prettyPrint 设置为 true，以便以用户友善模式输出 json。属性可以加上前缀来表示选项仅适用于 JSON 或 XML，以及 IN 或 OUT。前缀为： json.in. json.out. xml.in. xml.out。例如，值为 xml.out.mustBeJAXBElement 的键仅用于传出的 XML 数据格式。没有前缀的密钥是所有情况的通用密钥。		Map
camel.rest.enable-cors	是否在 HTTP 响应中启用 CORS 标头。默认值为 false。	false	布尔值
camel.rest.endpoint-property	允许为使用中的其他端点配置多个额外的属性。		Map
camel.rest.host	用于公开 REST 服务的主机名。		字符串
camel.rest.host-name-resolver	如果没有明确配置的主机名，这个 resolver 会用于计算 REST 服务将要使用的主机名。		RestHostNameResolver

Name	描述	默认值	类型
camel.rest.json-data-format	要使用的特定 json 数据格式的名称。默认将使用 json-jackson。重要：此选项仅用于设置数据格式的自定义名称，而不是引用现有数据格式实例。		字符串
camel.rest.port	用于公开 REST 服务的主机名。请注意，如果您使用 servlet 组件，则此处配置的端口号不适用，因为使用中的端口号是 servlet 组件使用的实际端口号。例如，如果使用 Apache Tomcat，它的 tomcat http 端口，如果使用 Apache Karaf，它的在 Karaf 中的 HTTP 服务，它默认使用端口 8181。虽然在这些情况下，这里设置端口号，但允许工具和 JMX 知道端口号，因此建议将端口号设置为 servlet 引擎使用的数字。		字符串
camel.rest.producer-api-doc	设置 api 文档的位置，REST 生成者将根据这个文档来验证 REST uri 和查询参数是否有效。这需要将 camel-swagger-java 添加到 classpath 中，任何缺失的配置都会导致 Camel 在启动时失败并报告错误。默认情况下从 classpath 加载的 api 文档的位置，但您可以使用 file: 或 http: 引用从文件或 http url 加载的资源。		字符串
camel.rest.producer-component	设置要用作 REST 生成者的 Camel 组件的名称。		字符串
camel.rest.scheme	用于公开 REST 服务的方案。通常支持 http 或 https。默认值为 http。		字符串
camel.rest.skip-binding-on-error-code	如果存在自定义 HTTP 错误代码标头，是否跳过输出绑定。这允许构建没有绑定到 json / xml 等自定义错误消息，否则成功信息会这样做。	false	布尔值
camel.rest.use-x-forward-headers	是否将 X-Forward 标头用于主机和相关设置。默认值为 true。	true	布尔值
camel.rest.xml-data-format	要使用的特定 XML 数据格式的名称。默认情况下将使用 jaxb。重要：此选项仅用于设置数据格式的自定义名称，而不是引用现有数据格式实例。		字符串
camel.rest.api-context-id-pattern	**弃用** 设置 CamelContext id 特征，以只允许 CamelContext 中名称与特征匹配的其他服务的 Rest API。特征 name 指的是 CamelContext 名称，仅匹配当前的 CamelContext。对于任何其他值，特征使用来自 PatternHelper#matchPattern (String,String)的规则。		字符串

Name	描述	默认值	类型
camel.rest.api-context-listing	**弃用** 设置是否启用了 JVM 中带有 REST 服务的所有可用 CamelContext 的列表。如果启用，它将允许发现这些上下文，如果为 false，则只使用当前的 CamelContext。	false	布尔值

第 108 章 REST

支持生成者和消费者

REST 组件允许使用 Rest DSL 和插件来定义 REST 端点(consumer)，作为 REST 传输。

其余组件也可以用作客户端(producer)来调用 REST 服务。

108.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 其余 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-starter</artifactId>
</dependency>
```

108.2. URI 格式

```
rest://method:path[:uriTemplate]?[options]
```

108.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

108.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

108.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

108.4. 组件选项

REST 组件支持 8 个选项，如下所列。

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
consumerComponentName (consumer)	用于(consumer) REST 传输的 Camel Rest 组件，如 jetty, servlet, undertow。如果没有明确配置组件，则 Camel 会查找是否有与 Rest DSL 集成的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestConsumerFactory。如果找到其中任何一个，则使用它。		字符串

Name	描述	默认值	类型
apidoc (producer)	swagger api doc 资源要使用的。默认情况下，资源从 classpath 加载，且必须采用 JSON 格式。		字符串
componentName (producer)	**弃用了** 用于(producer) REST 传输的 Camel Rest 组件，如 http, undertow.如果没有明确配置组件，则 Camel 会查找是否有与 Rest DSL 集成的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestProducerFactory。如果找到其中任何一个，则使用它。		字符串
host (producer)	要使用的 HTTP 服务的主机和端口(swagger 模式中的覆盖主机)。		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
producerComponentName (producer)	用于(producer) REST 传输的 Camel Rest 组件，如 http, undertow。如果没有明确配置组件，则 Camel 会查找是否有与 Rest DSL 集成的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestProducerFactory。如果找到其中任何一个，则使用它。		字符串
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

108.5. 端点选项

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：

108.5.1. 路径参数(3 参数)

Name	描述	默认值	类型
method (common)	**需要使用所需的** HTTP 方法。 Enum 值： • get • post • put • delete • patch • head • trace • 连接 • 选项		字符串
path (common)	**必需** 的基本路径。		字符串
uritemplate (common)	uri 模板。		字符串

108.5.2. 查询参数 (16 参数)

Name	描述	默认值	类型
consume (common)	这个 REST 服务接受的介质类型，如 'text/xml' 或 'application/json'。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 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'，这个 REST 服务返回。		字符串
routeId (common)	此 REST 服务创建的路由名称。		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
consumerComponentName (consumer)	用于(consumer) REST 传输的 Camel Rest 组件，如 jetty, servlet, undertow。如果没有明确配置组件，则 Camel 会查找是否有与 Rest DSL 集成的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestConsumerFactory。如果找到其中任何一个，则使用它。		字符串
description (consumer)	记录此 REST 服务的人工描述。		字符串
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
apidoc (producer)	要使用的 openapi api doc 资源。默认情况下，资源从 classpath 加载，且必须采用 JSON 格式。		字符串

Name	描述	默认值	类型
bindingMode (producer)	配置制作者的绑定模式。如果设置为除 'off' 以外的任何内容，则生成者将尝试将传入消息的正文从 inType 转换为 json 或 xml，并从 json 或 xml 转换为 outType。 Enum 值： • auto • off • json • xml • json_xml		RestBindingMode
host (producer)	要使用的 HTTP 服务的主机和端口（在 openapi 模式中的覆盖主机）。		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
producerComponentName (producer)	用于(producer) REST 传输的 Camel Rest 组件，如 http, undertow。如果没有明确配置组件，则 Camel 会查找是否有与 Rest DSL 集成的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestProducerFactory。如果找到其中任何一个，则使用它。		字符串
queryParameters (producer)	要调用的 HTTP 服务的查询参数。查询参数可以包含多个参数，用 ampersand 分隔，如 foo=123&bar=456。		字符串

108.6. 支持的其余组件

以下组件支持其余消费者(Rest DSL)：

- ***camel-servlet***

- camel-platform-http

以下组件支持 rest producer：

- camel-http

108.7. 路径和 URITEMPLATE 语法

path 和 uriTemplate 选项使用 REST 语法来定义，您可以使用 support 参数定义 REST 上下文路径。

注意

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 REST 更常见的做法。

以下是仅使用路径的 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}");
```

以下示例将基本路径配置为"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}");
```

注意

Rest 端点路径不接受转义的字符，如 加号。这是 Apache Camel 3 的默认行为。

108.8. REST PRODUCER 示例

您可以使用其他组件调用 REST 服务，如任何其他 Camel 组件一样。

例如，要使用 hello/{me} 调用 REST 服务，您可以执行

```
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，您可以发送一个空消息正文和标头，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，而不是使用 host 选项，如下所示：

```
restConfiguration().host("myserver:8080/foo");

from("direct:start")
  .to("rest:get:hello/{me}");
```

您可以使用 producerComponent 选择将哪些 Camel 组件用作 HTTP 客户端，例如，要使用 http，您可以：

```
restConfiguration().host("myserver:8080/foo").producerComponent("http");

from("direct:start")
  .to("rest:get:hello/{me}");
```

108.9. REST PRODUCER 绑定

REST 生成者支持使用 JSon 或 XML （如 rest-dsl）进行绑定。

例如，要使用打开 json 绑定模式的 jetty，您可以在 rest 配置中配置它：

```
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，它将在调用 REST 服务前自动将任何 POJO 绑定到 json ：

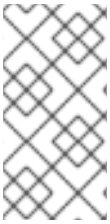
```
UserPojo user = new UserPojo();
user.setId(123);
user.setName("Donald Duck");

template.sendBody("direct:start", user);
```

在上例中，我们发送了一个 POJO 实例 UserPojo 作为消息正文。由于我们在其它配置中开启了 JSon 绑定，因此 POJO 将在调用 REST 服务前从 POJO 总结到 JSon。

但是，如果您想要为响应消息执行绑定（例如 REST 服务发回为响应），则需要配置 outType 选项，以指定 POJO 的类名称（从 JSon 变为 POJO）到 POJO。

例如，如果 REST 服务返回绑定到 com.foo.MyResponsePojo 的 JSon 有效负载，您可以将其配置为：

```
restConfiguration().component("jetty").host("localhost").port(8080).bindingMode(RestBinding
Mode.json);

  from("direct:start")
    .to("rest:post:user?outType=com.foo.MyResponsePojo");
```



注意

如果您希望为从调用 REST 服务收到的响应消息进行 POJO 绑定，您必须配置 outType 选项。

108.10. 更多示例

请参阅 **Rest DSL，它提供了更多示例，以及如何使用 Rest DSL 以 nicer RESTful 方式定义它们。**

Apache Camel 发行版中有一个 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，可以在 Apache Tomcat 或类似的 web 容器上部署。

108.11. 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.rest-api.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.rest-api.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-api.enabled	是否启用 rest-api 组件的自动配置。这默认是启用的。		布尔值
camel.component.rest.api-doc	swagger api doc 资源要使用的。默认情况下，资源从 classpath 加载，且必须采用 JSON 格式。		字符串
camel.component.rest.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.rest.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.consumer-component-name	用于(consumer) REST 传输的 Camel Rest 组件，如 jetty, servlet, undertow。如果没有明确配置组件，则 Camel 会查找是否有与 Rest DSL 集成的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestConsumerFactory。如果找到其中任何一个，则使用它。		字符串
camel.component.rest.enabled	是否启用其余组件的自动配置。这默认是启用的。		布尔值
camel.component.rest.host	要使用的 HTTP 服务的主机和端口(swagger 模式中的覆盖主机)。		字符串
camel.component.rest.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.producer-component-name	用于(producer) REST 传输的 Camel Rest 组件，如 http, undertow。如果没有明确配置组件，则 Camel 会查找是否有与 Rest DSL 集成的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestProducerFactory。如果找到其中任何一个，则使用它。		字符串
camel.component.rest.component-name	**弃用了** 用于(producer) REST 传输的 Camel Rest 组件，如 http, undertow.如果没有明确配置组件，则 Camel 会查找是否有与 Rest DSL 集成的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestProducerFactory。如果找到其中任何一个，则使用它。		字符串

第 109 章 SAGA

仅支持生成者

Saga 组件提供了一个网桥，用于使用 Saga EIP 在路由内执行自定义操作。

组件应用于高级任务，如决定完成或补偿将 completionMode 设置为 MANUAL 的 Saga。

有关在常见场景中使用 sagas 的帮助信息，请参阅 Saga EIP 文档。

109.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 saga 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-saga-starter</artifactId>
</dependency>
```

109.2. URI 格式

```
saga:action
```

109.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

109.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设

置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

109.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

109.4. 组件选项

Saga 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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

109.5. 端点选项

Saga 端点使用 URI 语法进行配置：

```
saga:action
```

使用以下路径和查询参数：

109.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
action (producer)	**执行** 所需的操作（完成或补偿）。 Enum 值： • COMPLETE • COMPENSATE		SagaEndpointAction

109.5.2. 查询参数(1 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

109.6. 使用带有 SPRING BOOT 和 LRA COORDINATOR 的 CAMEL-SAGA

本例演示了如何使用 Spring Boot 和 Narayana LRA Coordinator 与 Apache Camel Saga 一起工

作，以管理长时间运行的操作。如需更多信息，请参阅 Saga 示例。

109.7. 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.saga.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.saga.enabled	是否启用 saga 组件的自动配置。这默认是启用的。		布尔值
camel.component.saga.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

第 110 章 SALESFORCE

支持生成者和消费者

此组件支持生产者和消费者端点与 Salesforce 使用 Java DTO 进行通信。
有一个 companion maven plugin Camel Salesforce Plugin 来生成这些 DTO （请参阅以下内容）。

110.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 salesforce 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-salesforce-starter</artifactId>
</dependency>
```

默认情况下，camel-salesforce-maven-plugin 使用 TLSv1.3 与 salesforce 交互。TLS 版本可在插件上配置。FIPS 用户可以配置属性 sslContextParameters.secureSocketProtocol。要使用 maven-plugin，您必须将以下依赖项添加到 pom.xml 文件中。

```
<plugin>
  <groupId>org.apache.camel.maven</groupId>
  <artifactId>camel-salesforce-maven-plugin</artifactId>
  <version>${camel-community.version}</version>
  <executions>
    <execution>
      <goals>
        <goal>generate</goal>
      </goals>
      <configuration>
        <clientId>${camelSalesforce.clientId}</clientId>
        <clientSecret>${camelSalesforce.clientSecret}</clientSecret>
        <userName>${camelSalesforce.userName}</userName>
        <password>${camelSalesforce.password}</password>
        <sslContextParameters>
          <secureSocketProtocol>TLSv1.2</secureSocketProtocol>
        </sslContextParameters>
 <includes>
   <include>Contact</include>
 </includes>
      </configuration>
    </execution>
  </executions>
</plugin>
```

其中 camel-community.version 指的是您使用 camel-salesforce-maven-plugin 时使用的相应 Camel 社区版本。例如，对于红帽构建的 Camel Spring Boot 版本 4.4.0，您可以使用 Apache Camel 的 '4.4.0' 版本。

110.2. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

110.2.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

110.2.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

110.3. 组件选项

Salesforce 组件支持 90 个选项，如下所列。

Name	描述	默认值	类型
apexMethod (common)	APEX 方法名称。		字符串
apexQueryParams (common)	APEX 方法的查询参数。		Map
apiVersion (common)	Salesforce API 版本。	53.0	字符串
backoffIncrement (common)	流连接重启尝试的 backoff 间隔递增超过 CometD auto-reconnect。	1000	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. Enum 值： • XML • CSV • JSON • ZIP_XML • ZIP_CSV • ZIP_JSON		ContentType
defaultReplayId (common)	如果 initialReplayIdMap 中没有值，则默认 replayId 设置。	-1	Long
fallBackReplayId (common)	ReplayId 在 Invalid Replay Id 响应后回退到。	-1	Long

Name	描述	默认值	类型
format (common)	用于 Salesforce API 调用的有效负载格式(JSON 或 XML)默认为 JSON。自 Camel 3.12 起，此选项仅适用于 Raw 操作。 Enum 值： • JSON • XML		PayloadFormat
httpclient (common)	自定义 Jetty Http Client 用于连接到 Salesforce.		SalesforceHttpClient
httpClientConnectionTimeout (common)	连接到 Salesforce 服务器时 HttpClient 使用的连接超时。	60000	long
httpClientIdleTimeout (common)	当等待来自 Salesforce 服务器的响应时，HttpClient 使用的超时。	10000	long
httpMaxContentLength (common)	HTTP 响应的最大内容长度。		整数
httpRequestBufferSize (common)	HTTP 请求缓冲区大小。对于大型 SOQL 查询，可能需要增加。	8192	整数
includeDetails (common)	在 Salesforce1 Analytics 报告中包含详情，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
Limit (common)	对返回的记录数量的限制。适用于一些 API，请查看 Salesforce 文档。		整数
Locator (common)	salesforce Bulk 2.0 API 提供的定位器用于获取 Query 作业的结果。		字符串
maxBackoff (common)	Streaming 连接重启尝试的最大 backoff 间隔，超过 CometD auto-reconnect。	30000	long

Name	描述	默认值	类型
maxRecords (common)	对于 Bulk 2.0 Query，用于检索每个结果的最大记录数。请求仍会受到大小限制。如果您正在处理大量查询结果，则在从 Salesforce 接收所有数据前可能会遇到超时。要防止超时，请在 maxRecords 参数中指定您的客户端期望接收的最大记录数。这会将结果分成较小的集合，这个值作为最大大小。		整数
notFoundBehaviour (common)	设置从 Salesforce API 接收的 404 not found 状态。如果正文被设置为 NULL NotFoundBehaviour#NULL，或者应该在交换上发送异常 NotFoundBehaviour#EXCEPTION - 默认值。 Enum 值： • 例外 • NULL	例外	NotFoundBehaviour
notifyForFields (common)	通知字段，选项为 ALL, REFERENCED, SELECT, WHERE。 Enum 值： • ALL • REFERENCED • 选择 • 其中		NotifyForFieldsEnum
notifyForOperationCreate (common)	通知创建操作，默认为 false (API version = 29.0)。		布尔值
notifyForOperationDelete (common)	notify for delete 操作，默认为 false (API version = 29.0)。		布尔值
notifyForOperations (common)	通知操作，选项为 ALL、CREATE、EXTENDED、UPDATE (API 版本 29.0)。 Enum 值： • ALL • CREATE • EXTENDED • UPDATE （更新）		NotifyForOperationsEnum

Name	描述	默认值	类型
notifyForOperationUndelete (common)	通知取消删除操作，默认为 false (API version = 29.0)。		布尔值
notifyForOperationUpdate (common)	通知更新操作，默认为 false (API version = 29.0)。		布尔值
ObjectMapper (common)	自定义 Jackson ObjectMapper，以便在序列化/取消调试 Salesforce 对象时使用。		ObjectMapper
packages (common)	在哪些软件包中生成了 DTO 类。通常，使用 camel-salesforce-maven-plugin 生成类。如果使用生成的 DTOs 进行设置，以获取在 parameters/header 值中使用短 SObject 名称的好处。可以使用逗号分隔多个软件包。		字符串
pkChunking (common)	使用 PK Chunking。仅用于原始 Bulk API。如果需要，批量 2.0 API 会自动执行 PK 块。		布尔值
pkChunkingChunkSize (common)	用于 PK Chunking 的块大小。如果未指定，salesforce 默认为 100,000。最大大小为 250,000。		整数
pkChunkingParent (common)	当您启用 PK 块对共享对象查询时，指定父对象。块基于父对象的记录，而不是共享对象的记录。例如，在 AccountShare 上查询时，将 Account 指定为父对象。只要支持父对象，支持 PK 块进行共享对象。		字符串
pkChunkingStartRow (common)	指定用作第一个块的低边界的 15 个字符或 18 个字符记录 ID。在重启批处理间失败的作业时，使用此参数指定起始 ID。		字符串
queryLocator (common)	当查询结果超过单个调用中检索的记录数时，salesforce 提供的查询查找器可用。在后续调用中使用这个值来检索额外记录。		字符串
rawPayload (common)	使用原始有效负载字符串进行请求和响应(JSON 或 XML，具体取决于格式)，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 报告元数据以进行过滤。		ReportMetadata

Name	描述	默认值	类型
resultId (common)	批量 API 结果 ID。		字符串
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)	要检索的 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 external ID 字段值。		字符串
sObjectName (common)	API 需要或支持 SObject 名称。		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)	在使用 Streaming API 时是否要更新现有的 Push 主题，默认为 false。	false	布尔值
config (common (advanced))	全局端点配置 - 用于设置适用于所有端点的值。		SalesforceEndpointConfig
httpClientProperties (common (advanced))	用于设置可在底层 HTTP 客户端上配置的任何属性。查看 SalesforceHttpClient 的属性，以及所有可用选项的 Jetty HttpClient。		Map
longPollingTransportProperties (common (advanced))	用于设置由 streaming api 使用的 LongPollingTransport (CometD)使用的任何属性。		Map
workerPoolMaxSize (common (advanced))	用于处理 HTTP 响应的线程池的最大大小。	20	int

Name	描述	默认值	类型
workerPoolSize (common (advanced))	用于处理 HTTP 响应的线程池的大小。	10	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
allOrNone (producer)	复合 API 选项指示在没有任何成功时回滚所有记录。	false	布尔值
apexUrl (producer)	APEX 方法 URL。		字符串
compositeMethod (producer)	复合（原始）方法。		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
rawHttpHeaders (producer)	以逗号分隔的消息标头列表，以作为 Raw 操作的 HTTP 参数包含在内。		字符串
rawMethod (producer)	用于 Raw 操作的 HTTP 方法。		字符串
rawPath (producer)	域名后端点 URL 的部分。E.g., '/services/data/v52.0/sobjects/Account/'.		字符串
rawQueryParameters (producer)	以逗号分隔的消息标头列表，以作为 Raw 操作的查询参数包含。不要 url-encode 值，因为这将是自动进行的。		字符串
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
httpProxyExcludedAddresses (proxy)	不应使用 HTTP 代理服务器的地址列表。		Set
httpProxyHost (proxy)	要使用的 HTTP 代理服务器的主机名。		字符串
httpProxyIncludedAddresses (proxy)	应该使用 HTTP 代理服务器的地址列表。		Set
httpProxyPort (proxy)	要使用的 HTTP 代理服务器的端口号。		整数
httpProxySocks4 (proxy)	如果设置为 true，则将 HTTP 代理配置为用作 SOCKS4 代理。	false	布尔值
authenticationType (security)	要使用的显式验证方法，USERNAME_PASSWORD、REFRESH_TOKEN 或 JWT 之一。Salesforce 组件可以自动决定使用属性集的身份验证方法，设置此属性以消除任何不确定性。 Enum 值： • USERNAME_PASSWORD • REFRESH_TOKEN • JWT		AuthenticationType
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，需要配置一个连接的应用程序，但可以通过安装软件包来提供。		字符串
clientSecret (security)	在 Salesforce 实例设置中配置的连接应用程序的 OAuth 消费者 Secret。		字符串
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 httpProxyUsername 和 httpProxyPassword 用于身份验证。		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
httpProxyRealm (security)	代理服务器的域，用于针对 HTTP 代理服务器抢占的 Basic/Digest 身份验证方法。		字符串

Name	描述	默认值	类型
httpProxySecure (security)	如果设置为 false，则禁用访问 HTTP 代理时使用 TLS。	true	布尔值
httpProxyUseDigestAuth (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将使用，否则将使用基本授权方法。	false	布尔值
httpProxyUsername (security)	用于对 HTTP 代理服务器进行身份验证的用户名。		字符串
instanceUrl (security)	身份验证后使用的 Salesforce 实例的 URL，默认从 Salesforce 接收成功身份验证。		字符串
JWTAudience (security)	使用 OAuth JWT 流时用于 Audience 声明(aud)的值。如果没有设置，将使用登录 URL，这在大多数情形中都是合适的。		字符串
keystore (security)	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 不验证证书链，因此这可轻松是自签名证书。确保您将证书上传到对应的连接应用程序。		KeyStoreParameters
lazyLogin (security)	如果设置为 true，则组件可以在组件开始时向 Salesforce 进行身份验证。您通常会将其设置为 （默认）false，并立即了解任何身份验证问题。	false	布尔值
loginConfig (security)	一个嵌套 Bean 中的所有身份验证配置，也会直接在组件上设置所有属性。		SalesforceLoginConfig
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
password (security)	OAuth 流中使用的密码，以获取访问令牌的访问权限。易于开始使用密码 OAuth 流，但通常应该避免它，因为它比其他流的安全要低。确保使用安全令牌时将安全令牌附加到密码末尾。		字符串
refreshToken (security)	刷新令牌已在刷新令牌 OAuth 流中获取。需要设置 Web 应用并配置回调 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，然后在流结束时从 URL 检索 refresh_token。请注意，在开发过程中，Salesforce 允许在 localhost 托管回调 Web 应用程序。		字符串
sslContextParameters (security)	要使用的 SSL 参数，请参阅所有可用选项的 SSLContextParameters 类。		SSLContextParameters

Name	描述	默认值	类型
useGlobalSslContextParameters (security)	启用使用全局 SSL 上下文参数。	false	布尔值
用户名 （安全性）	OAuth 流中使用的用户名，以获取访问令牌的访问权限。易于开始使用密码 OAuth 流，但通常应该避免它，因为它比其他流的安全要低。		字符串

110.4. 端点选项

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：

110.4.1. 路径参数(2 参数)

Name	描述	默认值	类型
operationName (producer)	要使用的操作。 Enum 值： • getVersions • getResources • getGlobalObjects • getBasicInfo • getDescription • getSObject • createSObject • updateSObject • deleteSObject • getSObjectWithId • upsertSObject • deleteSObjectWithId • getBlobField		OperationName

Name	描述	默认值	类型
	• 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 • 批准 • 批准 • composite-tree • composite-batch • 复合 • compositeRetrieveSObjectCollections • compositeCreateSObjectCollections • compositeUpdateSObjectCollections		

Name	描述	默认值	类型
	• compositeUpsertSObjectCollections • compositeDeleteSObjectCollections • bulk2GetAllJobs • bulk2CreateJob • bulk2GetJob • bulk2CreateBatch • bulk2CloseJob • bulk2AbortJob • bulk2DeleteJob • bulk2GetSuccessfulResults • bulk2GetFailedResults • bulk2GetUnprocessedRecords • bulk2CreateQueryJob • bulk2GetQueryJob • bulk2GetAllQueryJobs • bulk2GetQueryJobResults • bulk2AbortQueryJob • bulk2DeleteQueryJob • raw		
topicName (consumer)	要使用的主题/频道的名称。		字符串

110.4.2. 查询参数(57 参数)

Name	描述	default	类型
apexMethod (common)	APEX 方法名称。		字符串
apexQueryParams (common)	APEX 方法的查询参数。		Map
apiVersion (common)	Salesforce API 版本。	53.0	字符串
backoffIncrement (common)	流连接重启尝试的 backoff 间隔递增超过 CometD auto-reconnect。	1000	long

Name	描述	default	类型
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. Enum 值： • XML • CSV • JSON • ZIP_XML • ZIP_CSV • ZIP_JSON		ContentType
defaultReplayId (common)	如果 initialReplayIdMap 中没有值，则默认 replayId 设置。	-1	Long
fallBackReplayId (common)	ReplayId 在 Invalid Replay Id 响应后回退到。	-1	Long
format (common)	用于 Salesforce API 调用的有效负载格式(JSON 或 XML)默认为 JSON。自 Camel 3.12 起，此选项仅适用于 Raw 操作。 Enum 值： • JSON • XML		PayloadFormat
httpclient (common)	自定义 Jetty Http Client 用于连接到 Salesforce.		SalesforceHttpClient
includeDetails (common)	在 Salesforce1 Analytics 报告中包含详情，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

Name	描述	default	类型
Limit (common)	对返回的记录数量的限制。适用于一些 API，请查看 Salesforce 文档。		整数
Locator (common)	salesforce Bulk 2.0 API 提供的定位器用于获取 Query 作业的结果。		字符串
maxBackoff (common)	Streaming 连接重启尝试的最大 backoff 间隔，超过 CometD auto-reconnect。	30000	long
maxRecords (common)	对于 Bulk 2.0 Query，用于检索每个结果的最大记录数。请求仍会受到大小限制。如果您正在处理大量查询结果，则在从 Salesforce 接收所有数据前可能会遇到超时。要防止超时，请在 maxRecords 参数中指定您的客户端期望接收的最大记录数。这会将结果分成较小的集合，这个值作为最大大小。		整数
notFoundBehaviour (common)	设置从 Salesforce API 接收的 404 not found 状态。如果正文被设置为 NULL NotFoundBehaviour#NULL，或者应该在交换上发送异常 NotFoundBehaviour#EXCEPTION - 默认值。 Enum 值： • 例外 • NULL	例外	NotFoundBehaviour
notifyForFields (common)	通知字段，选项为 ALL, REFERENCED, SELECT, WHERE。 Enum 值： • ALL • REFERENCED • 选择 • 其中		NotifyForFieldsEnum
notifyForOperationCreate (common)	通知创建操作，默认为 false (API version = 29.0)。		布尔值
notifyForOperationDelete (common)	notify for delete 操作，默认为 false (API version = 29.0)。		布尔值

Name	描述	default	类型
notifyForOperations (common)	通知操作，选项为 ALL、CREATE、EXTENDED、UPDATE (API 版本 29.0)。 Enum 值： • ALL • CREATE • EXTENDED • UPDATE （更新）		NotifyForOperationsEnum
notifyForOperationUndelete (common)	通知取消删除操作，默认为 false (API version = 29.0)。		布尔值
notifyForOperationUpdate (common)	通知更新操作，默认为 false (API version = 29.0)。		布尔值
ObjectMapper (common)	自定义 Jackson ObjectMapper，以便在序列化/取消调试 Salesforce 对象时使用。		ObjectMapper
pkChunking (common)	使用 PK Chunking。仅用于原始 Bulk API。如果需要，批量 2.0 API 会自动执行 PK 块。		布尔值
pkChunkingChunkSize (common)	用于 PK Chunking 的块大小。如果未指定，salesforce 默认为 100,000。最大大小为 250,000。		整数
pkChunkingParent (common)	当您启用 PK 块对共享对象查询时，指定父对象。块基于父对象的记录，而不是共享对象的记录。例如，在 AccountShare 上查询时，将 Account 指定为父对象。只要支持父对象，支持 PK 块进行共享对象。		字符串
pkChunkingStartRow (common)	指定用作第一个块的低边界的 15 个字符或 18 个字符记录 ID。在重启批处理间失败的作业时，使用此参数指定起始 ID。		字符串
queryLocator (common)	当查询结果超过单个调用中检索的记录数时，salesforce 提供的查询查找器可用。在后续调用中使用这个值来检索额外记录。		字符串
rawPayload (common)	使用原始有效负载字符串进行请求和响应(JSON 或 XML，具体取决于格式)，而不是 DTOs，默认为 false。	false	布尔值
reportId (common)	Salesforce1 Analytics 报告 Id.		字符串

Name	描述	default	类型
reportMetadata (common)	Salesforce1 Analytics 报告元数据以进行过滤。		ReportMetadata
resultId (common)	批量 API 结果 ID。		字符串
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)	要检索的 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 external ID 字段值。		字符串
sObjectName (common)	API 需要或支持 SObject 名称。		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)	在使用 Streaming API 时是否要更新现有的 Push 主题，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

Name	描述	default	类型
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
allOrNone (producer)	复合 API 选项指示在没有任何成功时回滚所有记录。	false	布尔值
apexUrl (producer)	APEX 方法 URL。		字符串
compositeMethod (producer)	复合（原始）方法。		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
rawHttpHeaders (producer)	以逗号分隔的消息标头列表，以作为 Raw 操作的 HTTP 参数包含在内。		字符串
rawMethod (producer)	用于 Raw 操作的 HTTP 方法。		字符串
rawPath (producer)	域名后端点 URL 的部分。E.g., '/services/data/v52.0/sobjects/Account/'.		字符串
rawQueryParameters (producer)	以逗号分隔的消息标头列表，以作为 Raw 操作的查询参数包含。不要 url-encode 值，因为这将是自动进行的。		字符串

110.5. 向 SALESFORCE 进行身份验证

组件支持三个 OAuth 身份验证流：

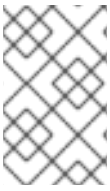
- OAuth 2.0 Username-Password 流
- OAuth 2.0 刷新令牌流
- OAuth 2.0 JWT Bearer 令牌流

对于每个需要设置不同属性集的流：

表 110.1. 表 1.为每个身份验证流设置的属性

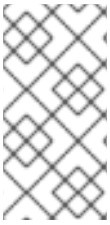
属性	在 Salesforce 上找到它	流
clientId	connected App, Consumer Key	所有流
clientSecret	connected App, Consumer Secret	username-Password, Refresh Token
userName	salesforce 用户用户名	username-Password, JWT Bearer Token
password	salesforce 用户密码	username-Password
refreshToken	从 OAuth 流回调	刷新令牌
keystore	连接的应用程序，数字证书	JWT Bearer 令牌

组件自动决定您要配置的流，以移除 authenticationType 属性。



注意

不建议在生产环境中使用 Username-Password 流。



注意

JWT Bearer 令牌流中使用的证书可以是自签名证书。包含证书的 KeyStore 和私钥必须只包含单个证书私钥条目。

110.6. URI 格式

当用作消费者时，接收流事件时，URI 方案是：

```
salesforce:topic?options
```

当用作生成者时，调用 Salesforce REST API 时，URI 方案是：

```
salesforce:operationName?options
```

110.7. 传递 SALESFORCE 标头并获取 SALESFORCE 响应标头

支持通过入站 消息标头传递 Salesforce 标头，从 Camel 消息上的 Sforce 或 x-sfdc 开始的标头将在请求中传递，以及以 Sforce 开头的响应标头将出现在出站消息标头中。

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);
    }
}
```

另外，HTTP 响应状态代码和文本作为标头 交换。HTTP_RESPONSE_CODE 和 Exchange.HTTP_RESPONSE_TEXT 提供。

110.8. 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 一次处理一个记录，Query API 可以检索多个记录。

110.8.1. REST API

对于 operationName，您可以使用以下内容：

- getVersions - 获取支持的 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 - 更新或使用外部 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（如果有大量结果）*
- *搜索 - 运行 Salesforce SOSL 查询*
- *限制 - 获取机构 API 使用限制*
- *recent - 获取最新的项目*
- *Approval - 为批准过程提交记录或记录（批量）*
- *Approvals - 获取所有批准过程列表*
- *复合 - 提交最多 25 个相关的 REST 请求并接收单个响应。也可以在不限制的情况下使用"原始"复合。*
- *composite-tree - 在一个 go 中创建最多 200 个记录，具有父子关系（最多 5 个级别）*
- *composite-batch - 提交批处理中的请求组成*
- *compositeRetrieveSObjectCollections - Retrieve 同一对象类型的一个或多个记录。*
- *compositeCreateSObjectCollections - 添加最多 200 个记录，返回 SaveSObjectResult 对象列表。*

- *compositeUpdateSObjectCollections - 更新最多 200 记录，返回 SaveSObjectResult 对象列表。*
- *compositeUpsertSObjectCollections - 根据外部 ID 字段创建或更新(upsert)，最多 200 个记录。返回 UpsertSObjectResult 对象列表。*
- *compositeDeleteSObjectCollections - 删除最多 200 记录，返回 SaveSObjectResult 对象列表。*
- *queryAll - 运行 SOQL 查询。它返回由于合并（最多三个记录出现）删除的结果，删除其他记录，以及重新父任何相关的记录）或删除。另外，返回有关归档任务和事件记录的信息。*
- *getBlobField - 从单个记录中检索指定的 blob 字段。*
- *apexCall - 执行用户定义的 APEX REST API 调用。*
- *raw - 将请求发送到 salesforce，并完全控制端点、参数、正文等。*

例如，以下制作者端点使用 upsertSObject API，sObjectIdName 参数将 '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。如果现有记录已更新，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，响应消息可以是 null。

```
...to("salesforce:upsertSObject?sObjectIdName=Name")...
```

110.8.2. 批量 2.0 API

Bulk 2.0 API 对原始 Bulk API 提供了简化的模型。使用它来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 的最小 API 版本为 v47.0。以下提到的 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 - 关闭作业。您必须关闭作业，以便它被处理或中止/删除。jobId 参数是必需的。***

- ***bulk2AbortJob - Abort a job。jobId 参数是必需的。***

- ***bulk2DeleteJob - 删除作业。jobId 参数是必需的。***

- ***bulk2GetSuccessfulResults - 获取作业成功的结果。返回的消息正文将包含 CSV 数据的 InputStream。jobId 参数是必需的。***

- ***bulk2GetFailedResults - 获取作业的失败结果。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 的实例。如果 done 属性为 false，则还有额外的页面可供获取，而 nextRecordsUrl 属性包含在后续调用的 queryLocator 参数中设置的值。***

- ***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。接受 maxRecords 和 locator 参数。响应消息标头包括 Sforce-NumberOfRecords 和 Sforce-Locator 标头。Sforce-Locator 的值可以通过 locator 参数传递给后续调用。***

- ***bulk2AbortQueryJob - Abort a bulk query job。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 QueryJobs 的实例。如果 done 属性为 false，则还有额外的页面可供获取，而 nextRecordsUrl 属性包含在后续调用的 queryLocator 参数中设置的值。*

110.8.3. REST Bulk (original) API

生产者端点可以使用以下 API：支持所有作业数据格式，如 xml、csv、zip/xml 和 zip/csv。请求和响应必须由路由处理/问题单。通常，请求将是 CSV 文件的一些流源，，响应也可以保存到要与请求关联的文件中。

对于 operationName，您可以使用以下内容：

- *CreateJob - 创建 Salesforce Bulk 作业。必须在正文中提供 JobInfo 实例。通过 pkChunking* 选项支持 PK Chunking。请参阅 这里 的解释。*

- *Get Job - 使用其 Salesforce Id 获取 作业*

- *Close Job - 关闭作业*

- *abortJob - Aborts a Job*

- *CreateBatch - 在 Bulk 作业中提交批处理*

- *getBatch - 使用 Id 获取批处理*

- *getAllBatches - 获取 Bulk 作业 Id 的所有批处理*

- *getRequest - 获取批处理的 Request 数据(XML/CSV)*

- *getResults - 完成后获取批处理的结果*
- *createBatchQuery - 从 SOQL 查询创建批处理*
- *getQueryResultIds - 为 Batch Query 获取 Result Ids 列表*
- *getQueryResult - 获取 Result Id 的结果*
- *getRecentReports - 通过向 Report List 资源发送 GET 请求，获取您最近查看的报告最多 200。*
- *getReportDescription - 获取报告的报告、报告类型和相关元数据，可以是 tabular 或 summary 或 matrix 格式。*
- *executeSyncReport - 异步运行带有或不更改过滤器的报告，并返回最新的摘要数据。*
- *executeAsyncReport - 异步运行带有或没有过滤器的报告实例，并使用或没有详情返回摘要数据。*
- *getReportInstances - 为请求异步运行的报告返回实例列表。列表中的每一项都被视为报告的独立实例。*
- *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。消息中的正文必须包含可转换为 InputStream （通常是来自文件的 UTF-8 CSV 或 XML 内容）的正文，以及作业内容类型的作业和 'contentType' 的标头字段 'jobId'，可以是 XML、CSV、ZIP_XML 或 ZIP_CSV。put 消息正文将包含 BatchInfo on success，或抛出 SalesforceException on 错误。

```
...to("salesforce:createBatch").
```

110.8.4. REST Streaming API

消费者端点可使用以下语法来流传输端点，以便在 create/update 上接收 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")...
```

110.8.5. 平台事件

要发出平台事件，请使用 createSObject 操作。然后，设置消息正文可以是 JSON 字符串或 InputStream，带有 key-value datacategories-wagonin，该情况需要设置为 sObjectName 的 API 名称，或使用事件的适当类名称扩展类。

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" + exchange.getProperty(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" + exchange.getProperty(Exchange.TIMER_COUNTER);
```

```
        in.setBody("{\"OrderNumber\":\"" + orderNumber + "\"}");
    })
    .to("salesforce:createSObject?sObjectName=Order_Event__e");
```

要接收平台事件，请使用带有带有 event/（或 / event/ ）的平台事件的 API 名称的消费者端点，例如： salesforce:events/Order_Event__e。来自该端点的处理器分别在正文中接收 org.apache.camel.component.salesforce.api.dto.PlatformEvent 对象或 org.cometd.bayeux.Message，具体取决于 rawPayload 为 false 或 true。

例如，使用最简单的形式来消耗一个事件：

```
PlatformEvent event = consumer.receiveBody("salesforce:event/Order_Event__e",
PlatformEvent.class);
```

110.8.6. 更改数据捕获事件

另一方面，Salesforce 可以被配置为发出用于选择对象的记录更改的通知。另一方面，Camel Salesforce 组件可以响应此类通知，允许实例将这些更改同步到外部系统。

感兴趣的通知可以在 Camel 路由的 from ("salesforce:XXX") 子句中指定，例如：

```
from("salesforce:data/ChangeEvents?replayId=-1").log("being notified of all change events")
from("salesforce:data/AccountChangeEvent?replayId=-1").log("being notified of change
events for Account records")
from("salesforce:data/Employee__ChangeEvent?replayId=-1").log("being notified of change
events for Employee__c custom object")
```

接收的消息在正文中包含 java.util.Map<String,Object > 或 org.cometd.bayeux.Message，具体取决于 rawPayload 为 false 或 true。CamelSalesforceChangeType 标头可以被视为 CREATE、UPDATE、DELETE 或 UNDELETE 之一。

有关如何使用 Camel Salesforce 组件更改数据捕获功能的更多详细信息，请参阅 ChangeEventsConsumerIntegrationTest。

Salesforce 开发人员指南 非常适合更好地了解实施更改数据捕获集成应用程序的子公司。更改事件正文字段的动态性质、高级别复制步骤以及安全注意事项。

110.9. 例子

110.9.1. 将文档上传到内容工作区

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");
```

110.10. 使用 SALESFORCE LIMITS API

通过 salesforce:limits 操作，您可以从 Salesforce 获取 API 限制，然后对收到的数据执行操作。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，以便为其他路由保留每日 API 请求的 10%。输出消息的正文包含 org.apache.camel.component.salesforce.api.dto.Limits 对象实例，可与基于内容的路由路由器和基于内容的路由路由器和基于 Spring Expression Language (SpEL) 结合使用，以便在执行查询时选择。

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()
```

110.11. 使用批准

所有属性的名称与 Salesforce REST API 中带有 批准前缀完全相同。您可以通过设置 Endpoint 的 approval.PropertyName 设置批准属性来设置批准属性，这些属性将用作 template iwl-PROFILEmeaning，因为正文中不存在的任何属性，或者从 Endpoint 配置中获取标头。或者，您可以通过将 approval 属性分配给对 Registry 中的 bean 的引用来设置 Endpoint 上的批准模板。

您还可以使用传入消息标头中的同一 approval.PropertyName 提供标头值。

最后，最后一个正文可以包含一个 AprovalRequest 或 ApprovalRequest 对象来作为批处理处理。

请记住，需要记住的是这三个机制中指定的值的优先级：

1. body 中的值在任何其他之前具有优先权
2. 消息标头中的值优先于模板值
3. 如果未给出标头或正文中的其他值，则将设置模板中的值

例如，要使用标头中的值为批准发送一条记录，请使用：

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");
```

您可以使用以下方法发送记录进行批准：

```
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);
```

110.12. 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}");
```

110.13. 使用 SALESFORCE COMPOSITE API 提交 SOBJECT 树

要创建最多 200 个记录，包括父子关系，请使用 salesforce:composite-tree 操作。这需要一个 org.apache.camel.component.salesforce.api.dto.composite.SObjectTree 实例，并在输出消息中返回相同的对象树。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，对于某些记录操作可能会成功，对于某些记录操作可能会失败，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 DTO，但您也可以选择自定义识别树中每个对象的引用，用于来自您的数据库的实例主密钥。

我们来看一个例子：

```
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();
```

110.14. 使用 SALESFORCE COMPOSITE API 提交批处理中的多个请求

Composite API 批处理操作(comp-batch)允许您积累批处理中的多个请求，然后在一方提交一次，从而节省多个单个请求的往返成本。然后，每个响应都会在保留顺序的响应列表中接收，以便第 n 个请求响应处于响应的第 n 个位置。

注意

结果可能与 API 到 API 不同，因此请求的结果被指定为 java.lang.Object。在大多数情况下，结果将是 java.util.Map，其带有字符串键和值，或者其他 java.util.Map 作为值。请求以 JSON 格式发出并保存一些类型信息（例如，它知道哪些值为字符串）以及值是数字。

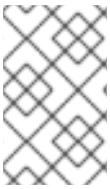
我们来看一个例子：

```
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", 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
```

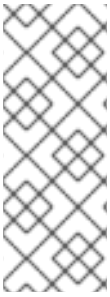
110.15. 使用 SALESFORCE COMPOSITE API 提交多个链请求

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，用于之前请求中生成的实例标识符。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，结果可能会与 API 不同，因此请求的结果是 java.lang.Object。在大多数情况下，结果将是 java.util.Map，其带有字符串键和值，或者其他 java.util.Map 作为值。以 JSON 格式发出的请求会保存一些类型信息（例如，它知道哪些值是字符串）以及值是数字。

我们来看一个例子：

```
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",
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()
```

110.16. 使用"原始" SALESFORCE 复合

由于 rawPayload 选项，可以在路由中准备 Salesforce JSON 请求，直接调用 Salesforce 复合。

例如，您可以有以下路由：

```
from("timer:fire?period=2000").setBody(constant("{\n" +
    " \"allOrNone\" : true,\n" +
    " \"records\" : [ { \n" +
    "   \"attributes\" : {\"type\" : \"FOO\"},\n" +
    "   \"Name\" : \"123456789\",\n" +
    "   \"FOO\" : \"XXXX\",\n" +
    "   \"ACCOUNT\" : 2100.0\n" +
    "   \"ExternalID\" : \"EXTERNAL\"\n"
    " }]\n" +
    "}")
  .to("salesforce:composite?rawPayload=true")
  .log("${body}");
```

路由直接以 JSON 创建正文，并使用 rawPayload=true 选项直接提交到 salesforce 端点。

使用这个方法，您可以完全控制 Salesforce 请求。

POST 是默认的 HTTP 方法，用于将原始复合请求发送到 salesforce。使用 compositeMethod 选项覆盖到其他支持的值 GET，这将返回其他可用复合资源的列表。

110.17. 使用 RAW OPERATION

将 HTTP 请求发送到 salesforce，并完全控制调用的所有方面。任何请求序列化或反序列化和响应正文都必须在路由中执行。Content-Type HTTP 标头将根据格式选项自动设置，但可以通过 rawHttpHeaders 选项覆盖。

参数	类型	描述	默认值	必填
请求正文	**string** 或 **InputStream**	HTTP 请求的正文		
rawPath	**字符串**	域名后端点 URL 的部分，如 '/services/data/v51.0/sobjects/Account/'		x
rawMethod	**字符串**	HTTP 方法		x

参数	类型	描述	默认值	必填
rawQueryParameters	**字符串**	以逗号分隔的消息标头列表，以作为查询参数包含。不要 url-encode 值，因为这将是自动进行的。		
rawHttpHeaders	**字符串**	以逗号分隔的消息标头列表，以包括为 HTTP 标头		

110.17.1. 查询示例

在本例中，我们将向 REST API 发送查询。查询必须在名为"q"的 URL 参数中传递，因此我们将创建一个名为 q 的消息标头，并告知原始操作将该消息标头包含为 URL 参数：

```
from("direct:queryExample")
  .setHeader("q", "SELECT Id, LastName FROM Contact")
  .to("salesforce:raw?
format=JSON&rawMethod=GET&rawQueryParameters=q&rawPath=/services/data/v51.0/query")
  // deserialize JSON results or handle in some other way
```

110.17.2. SObject 示例

在本例中，我们将在 create 操作中传递联系 REST API。由于 raw 操作不执行任何序列化，因此请确保在消息正文中传递 XML

```
from("direct:createAContact")
  .setBody(constant("<Contact><LastName>TestLast</LastName></Contact>"))
  .to("salesforce:raw?
format=XML&rawMethod=POST&rawPath=/services/data/v51.0/sobjects/Contact")
```

响应是：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Result>
    <id>0034x00000RnV6zAAF</id>
    <success>true</success>
</Result>
```

110.18. 使用 COMPOSITE SOBJECT COLLECTIONS

SObject Collections API 对一个请求中的多个记录执行操作。使用 sObject Collections 减少客户端和服务器之间的往返数量。整个请求将计为 API 限值的单个调用。此资源在 API 版本 42.0 及更高版本中提供。提供给这些操作的 SObject 记录（也称为 DTO）必须是 AbstractDescribedSObjectBase 的子类实例。有关生成这些 DTO 类的信息，请参阅 Maven 插件部分。这些操作序列化为 JSON 提供 DTO。

110.18.1. compositeRetrieveSObjectCollections

检索同一对象类型的一个或多个记录。

参数	类型	描述	默认值	必填
id	字符串列表或以逗号分隔的字符串	要返回的对象的一个或多个 ID 列表。所有 ID 必须属于同一对象类型。		x
fields	字符串列表或以逗号分隔的字符串	响应中要包含的字段列表。您指定的字段名称必须有效，且每个字段都必须具有读取权限。		x
sObjectName	字符串	SObject 的类型，如 **帐户**		x
sObjectClass	字符串	用于取消响应的 DTO 类的完全限定类名称。		如果 **sObjectName** 参数没有解析为 **软件包** 选项指定的软件包中存在的类，则需要此项。

110.18.2. compositeCreateSObjectCollections

最多添加 200 个记录，返回 SaveSObjectResult 对象列表。支持混合 SObject 类型。

参数	类型	描述	默认值	必填
请求正文	**SObject**列表	要创建的 SObjects 列表		x

参数	类型	描述	默认值	必填
allOrNone	布尔值	指明在创建任何对象失败时(true)还是继续独立创建请求时，是否回滚整个请求。	false	

110.18.3. compositeUpdateSObjectCollections

最多更新 200 个记录，返回 SaveSObjectResult 对象列表。支持混合 SObject 类型。

参数	类型	描述	默认值	必填
请求正文	**SObject**列表	要更新的 SObjects 列表		x
allOrNone	布尔值	指明在任何对象更新失败时(true)或继续独立更新请求时，是否回滚整个请求。	false	

110.18.4. compositeUpsertSObjectCollections

根据外部 ID 字段创建或更新(upsert) (upsert)，返回 UpsertSObjectResult 对象列表。不支持混合 SObject 类型。

参数	类型	描述	默认值	必填
请求正文	**SObject**列表	直到 upsert 的 SObjects 列表		x
allOrNone	布尔值	指明是否在任何对象的 upsert 失败(true)时回滚整个请求，还是继续对请求中的其他对象的独立 upsert。	false	
sObjectName	字符串	SObject 的类型，如 **帐户**		x
sObjectIdName	字符串	外部 ID 字段的名称		x

110.18.5. compositeDeleteSObjectCollections

删除最多 200 记录，返回 DeleteSObjectResult 对象列表。支持混合 SObject 类型。

参数	类型	描述	默认值	必填

参数	类型	描述	默认值	必填
sObjectIds 或请求正文	字符串列表或以逗号分隔的字符串	要删除最多 200 个 ID 的对象列表。		x
allOrNone	布尔值	指明在删除任何对象失败时(true)或继续独立删除请求时，是否回滚整个请求。	false	

110.19. 将 NULL 值发送到 SALESFORCE

默认情况下，带有 null 值的 SObject 字段不会发送到 salesforce。要将 null 值发送到 salesforce，请使用 fieldsToNull 属性，如下所示：

```
accountSObject.getFieldsToNull().add("Site");
```

110.20. 生成 SOQL 查询字符串

org.apache.camel.component.salesforce.api.utils.QueryHelper 包含用于生成 SOQL 查询的帮助方法。例如，要从 Account SObject 获取所有自定义字段，您可以调用以下内容来生成 SOQL SELECT：

```
String allCustomFieldsQuery = QueryHelper.queryToFetchFilteredFieldsOf(new Account(),
SObjectField::isCustom);
```

110.21. CAMEL SALESFORCE MAVEN 插件

此 Maven 插件为 Camel 生成 DTO。

出于明显的安全原因，建议不要在 pom.xml 中设置 clientId、clientSecret、userName 和 password 字段。该插件应该为其余属性配置，并可使用以下命令执行：

```
mvn camel-salesforce:generate -DcamelSalesforce.clientId=<clientid> -
DcamelSalesforce.clientSecret=<clientsecret> \
    -DcamelSalesforce.userName=<username> -DcamelSalesforce.password=<password>
```

生成的 DTOs 使用 Jackson 注解。所有 Salesforce 字段类型都支持。日期和时间字段默认映射到 java.time.ZonedDateTime，而 picklist 字段则映射到生成的 Java Enumerations。

有关如何生成 DTO 的详细信息，请参阅 README.md。

110.22. SPRING BOOT AUTO-CONFIGURATION

组件支持 91 个选项，如下所列。

Name	描述	default	类型
camel.component.salesforce.all-or-none	复合 API 选项指示在没有任何成功时回滚所有记录。	false	布尔值
camel.component.salesforce.apex-method	APEX 方法名称。		字符串
camel.component.salesforce.apex-query-params	APEX 方法的查询参数。		Map
camel.component.salesforce.apex-url	APEX 方法 URL。		字符串
camel.component.salesforce.api-version	Salesforce API 版本。	53.0	字符串
camel.component.salesforce.authentication-type	要使用的显式验证方法，USERNAME_PASSWORD、REFRESH_TOKEN 或 JWT 之一。Salesforce 组件可以自动决定使用属性集的身份验证方法，设置此属性以消除任何不确定性。		AuthenticationType
camel.component.salesforce.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.salesforce.backoff-increment	流连接重启尝试的 backoff 间隔递增超过 CometD auto-reconnect。选项是一个长类型。	1000	Long
camel.component.salesforce.batch-id	批量 API 批处理 ID。		字符串

Name	描述	default	类型
camel.component.salesforce.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.salesforce.client-id	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，需要配置一个连接的应用程序，但可以通过安装软件包来提供。		字符串
camel.component.salesforce.client-secret	在 Salesforce 实例设置中配置的连接应用程序的 OAuth 消费者 Secret。		字符串
camel.component.salesforce.composite-method	复合（原始）方法。		字符串
camel.component.salesforce.config	全局端点配置 - 用于设置适用于所有端点的值。选项是一个 org.apache.camel.component.salesforce.SalesforceEndpointConfig 类型。		SalesforceEndpointConfig
camel.component.salesforce.content-type	批量 API 内容类型, XML, CSV, ZIP_XML, ZIP_CSV.		ContentType
camel.component.salesforce.default-replay-id	如果 initialReplayIdMap 中没有值，则默认 replayId 设置。	-1	Long
camel.component.salesforce.enabled	是否启用 salesforce 组件的自动配置。这默认是启用的。		布尔值
camel.component.salesforce.fall-back-replay-id	ReplayId 在 Invalid Replay Id 响应后回退到。	-1	Long
camel.component.salesforce.format	用于 Salesforce API 调用的有效负载格式(JSON 或 XML)默认为 JSON。自 Camel 3.12 起，此选项仅适用于 Raw 操作。		PayloadFormat

Name	描述	default	类型
camel.component.salesforce.http-client	自定义 Jetty Http Client 用于连接到 Salesforce.选项是 org.apache.camel.component.salesforce.SalesforceHttpClient 类型。		SalesforceHttpClient
camel.component.salesforce.http-client-connection-timeout	连接到 Salesforce 服务器时 HttpClient 使用的连接超时。	60000	Long
camel.component.salesforce.http-client-idle-timeout	当等待来自 Salesforce 服务器的响应时，HttpClient 使用的超时。	10000	Long
camel.component.salesforce.http-client-properties	用于设置可在底层 HTTP 客户端上配置的任何属性。查看 SalesforceHttpClient 的属性，以及所有可用选项的 Jetty HttpClient。		Map
camel.component.salesforce.http-max-content-length	HTTP 响应的最大内容长度。		整数
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 httpProxyUsername 和 httpProxyPassword 用于身份验证。		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

Name	描述	default	类型
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 代理服务器抢占的 Basic/Digest 身份验证方法。		字符串
camel.component.salesforce.http-proxy-secure	如果设置为 false，则禁用访问 HTTP 代理时使用 TLS。	true	布尔值
camel.component.salesforce.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.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将使用，否则将使用基本授权方法。	false	布尔值
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.http-request-buffer-size	HTTP 请求缓冲区大小。对于大型 SOQL 查询，可能需要增加。	8192	整数
camel.component.salesforce.include-details	在 Salesforce1 Analytics 报告中包含详情，默认为 false。		布尔值
camel.component.salesforce.initial-replay-id-map	重播 ID 从每个频道名称开始。		Map
camel.component.salesforce.instance-id	Salesforce1 Analytics 报告执行实例 ID。		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 接收成功身份验证。		字符串

Name	描述	default	类型
camel.component.salesforce.job-id	批量 API 作业 ID。		字符串
camel.component.salesforce.jwt-audience	使用 OAuth JWT 流时用于 Audience 声明(aud)的值。如果没有设置，将使用登录 URL，这在大多数情形中都是合适的。		字符串
camel.component.salesforce.keystore	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 不验证证书链，因此这可轻松是自签名证书。确保您将证书上传到对应的连接应用程序。选项是 org.apache.camel.support.jsse.KeyStoreParameters 类型。		KeyStoreParameters
camel.component.salesforce.lazy-login	如果设置为 true，则组件可以在组件开始时向 Salesforce 进行身份验证。您通常会将其设置为 （默认）false，并立即了解任何身份验证问题。	false	布尔值
camel.component.salesforce.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.salesforce.limit	对返回的记录数量的限制。适用于一些 API，请查看 Salesforce 文档。		整数
camel.component.salesforce.locator	salesforce Bulk 2.0 API 提供的定位器用于获取 Query 作业的结果。		字符串
camel.component.salesforce.login-config	一个嵌套 Bean 中的所有身份验证配置，也会直接在组件上设置所有属性。选项是 org.apache.camel.component.salesforce.SalesforceLoginConfig 类型。		SalesforceLoginConfig
camel.component.salesforce.login-url	用于身份验证的 Salesforce 实例的 URL，默认设置为。		字符串
camel.component.salesforce.long-polling-transport-properties	用于设置由 streaming api 使用的 LongPollingTransport (CometD)使用的任何属性。		Map

Name	描述	default	类型
camel.component.salesforce.max-backoff	Streaming 连接重启尝试的最大 backoff 间隔，超过 CometD auto-reconnect。选项是一个长类型。	30000	Long
camel.component.salesforce.max-records	对于 Bulk 2.0 Query，用于检索每个结果的最大记录数。请求仍会受到大小限制。如果您正在处理大量查询结果，则在从 Salesforce 接收所有数据前可能会遇到超时。要防止超时，请在 maxRecords 参数中指定您的客户端期望接收的最大记录数。这会将结果分成较小的集合，这个值作为最大大小。		整数
camel.component.salesforce.not-found-behaviour	设置从 Salesforce API 接收的 404 not found 状态。如果正文被设置为 NULL NotFoundBehaviour#NULL，或者应该在交换上发送异常 NotFoundBehaviour#EXCEPTION - 默认值。		NotFoundBehaviour
camel.component.salesforce.notify-for-fields	通知字段，选项为 ALL, REFERENCED, SELECT, WHERE。		NotifyForFieldsEnum
camel.component.salesforce.notify-for-operation-create	通知创建操作，默认为 false (API version = 29.0)。		布尔值
camel.component.salesforce.notify-for-operation-delete	notify for delete 操作，默认为 false (API version = 29.0)。		布尔值
camel.component.salesforce.notify-for-operation-undelete	通知取消删除操作，默认为 false (API version = 29.0)。		布尔值
camel.component.salesforce.notify-for-operation-update	通知更新操作，默认为 false (API version = 29.0)。		布尔值
camel.component.salesforce.notify-for-operations	通知操作，选项为 ALL、CREATE、EXTENDED、UPDATE (API 版本 29.0)。		NotifyForOperationsEnum
camel.component.salesforce.object-mapper	自定义 Jackson ObjectMapper，以便在序列化/取消调试 Salesforce 对象时使用。选项是一个 com.fasterxml.jackson.databind.ObjectMapper 类型。		ObjectMapper

Name	描述	default	类型
camel.component.salesforce.packages	在哪些软件包中生成了 DTO 类。通常，使用 camel-salesforce-maven-plugin 生成类。如果使用生成的 DTOs 进行设置，以获取在 parameters/header 值中使用短 SObject 名称的好处。可以使用逗号分隔多个软件包。		字符串
camel.component.salesforce.password	OAuth 流中使用的密码，以获取访问令牌的访问权限。易于开始使用密码 OAuth 流，但通常应该避免它，因为它比其他流的安全要低。确保使用安全令牌时将安全令牌附加到密码末尾。		字符串
camel.component.salesforce.pk-chunking	使用 PK Chunking。仅用于原始 Bulk API。如果需要，批量 2.0 API 会自动执行 PK 块。		布尔值
camel.component.salesforce.pk-chunking-chunk-size	用于 PK Chunking 的块大小。如果未指定，salesforce 默认为 100,000。最大大小为 250,000。		整数
camel.component.salesforce.pk-chunking-parent	当您启用 PK 块对共享对象查询时，指定父对象。块基于父对象的记录，而不是共享对象的记录。例如，在 AccountShare 上查询时，将 Account 指定为父对象。只要支持父对象，支持 PK 块进行共享对象。		字符串
camel.component.salesforce.pk-chunking-start-row	指定用作第一个块的低边界的 15 个字符或 18 个字符记录 ID。在重启批处理间失败的作业时，使用此参数指定起始 ID。		字符串
camel.component.salesforce.query-locator	当查询结果超过单个调用中检索的记录数时，salesforce 提供的查询查找器可用。在后续调用中使用这个值来检索额外记录。		字符串
camel.component.salesforce.raw-http-headers	以逗号分隔的消息标头列表，以作为 Raw 操作的 HTTP 参数包含在内。		字符串
camel.component.salesforce.raw-method	用于 Raw 操作的 HTTP 方法。		字符串
camel.component.salesforce.raw-path	域名后端点 URL 的部分。E.g., '/services/data/v52.0/sobjects/Account/'.		字符串

Name	描述	default	类型
camel.component.salesforce.raw-payload	使用原始有效负载字符串进行请求和响应(JSON 或 XML，具体取决于格式)，而不是 DTOs，默认为 false。	false	布尔值
camel.component.salesforce.raw-query-parameters	以逗号分隔的消息标头列表，以作为 Raw 操作的查询参数包含。不要 url-encode 值，因为这将是自动进行的。		字符串
camel.component.salesforce.refresh-token	刷新令牌已在刷新令牌 OAuth 流中获取。需要设置 Web 应用并配置回调 URL 以接收刷新令牌，或使用位于 builtin 回调进行配置，然后从流程结束时从 URL 检索 refresh_token。请注意，在开发过程中，Salesforce 允许在 localhost 托管回调 Web 应用程序。		字符串
camel.component.salesforce.report-id	Salesforce1 Analytics 报告 Id.		字符串
camel.component.salesforce.report-metadata	Salesforce1 Analytics 报告元数据以进行过滤。选项是 org.apache.camel.component.salesforce.api.dto.analytics.reports.ReportMetadata 类型。		ReportMetadata
camel.component.salesforce.result-id	批量 API 结果 ID。		字符串
camel.component.salesforce.s-object-blob-field-name	SObject blob 字段名称。		字符串
camel.component.salesforce.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.s-object-fields	要检索的 SObject 字段。		字符串
camel.component.salesforce.s-object-id	API 需要 SObject ID。		字符串
camel.component.salesforce.s-object-id-name	SObject 外部 ID 字段名称。		字符串

Name	描述	default	类型
camel.component.salesforce.s-object-id-value	SObject external ID 字段值。		字符串
camel.component.salesforce.s-object-name	API 需要或支持 SObject 名称。		字符串
camel.component.salesforce.s-object-query	Salesforce SOQL 查询字符串。		字符串
camel.component.salesforce.s-object-search	salesforce SOSL 搜索字符串。		字符串
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.support.jsse.SSLContextParameters 类型。		SSLContextParameters
camel.component.salesforce.update-topic	在使用 Streaming API 时是否要更新现有的 Push 主题，默认为 false。	false	布尔值
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，以获取访问令牌的访问权限。易于开始使用密码 OAuth 流，但通常应该避免它，因为它比其他流的安全要低。		字符串
camel.component.salesforce.worker-pool-max-size	用于处理 HTTP 响应的线程池的最大大小。	20	整数
camel.component.salesforce.worker-pool-size	用于处理 HTTP 响应的线程池的大小。	10	整数

第 111 章 SAP 组件

SAP 组件是由十个不同 SAP 组件组成的软件包。有一些支持 sRFC、tRFC 和 qRFC 协议的远程函数调用(RFC)组件，并有一些 IDoc 组件用于使用 IDoc 格式的消息进行通信。组件使用 SAP Java Connector (SAP JCo)库来促进与 SAP 和 SAP IDoc 库双向通信，以 Intermediate Document (IDoc)格式传输文档。

111.1. 依赖项

将此组件的 pom.xml 添加以下依赖项：

```
<dependency>
  <groupId>org.fusesource</groupId>
  <artifactId>camel-sap-starter</artifactId>
  <version>4.4.0.redhat-00019</version>
</dependency>
```

111.1.1. SAP 组件的其他平台限制

因为 SAP 组件依赖于第三方 JCo 3 和 IDoc 3 库，所以它只能安装在这些库支持的平台上。

111.1.2. SAP JCo 和 SAP IDoc 库

使用 SAP 组件的前提条件是 SAP Java Connector (SAP JCo)库和 SAP IDoc 库被安装到 Java 运行时的 lib/ 目录中。您必须确保从 SAP Service Marketplace 下载目标操作系统的适当 SAP 库集合。

库文件的名称因目标操作系统而异，如下所示。

表 111.1. 所需的 SAP Libraries

SAP 组件	Linux 和 UNIX	Windows
SAP JCo 3	**sapjco3.jar** **libsapjco3.so**	**sapjco3.jar** **sapjco3.dll**
SAP IDoc	**sapidoc3.jar**	**sapidoc3.jar**

111.2. URI 格式

SAP 组件提供了两种不同的端点：remote Function Call (RFC)端点，以及 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 格式用于定义目标端点（换句话说，Camel producer 端点）和 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被命名和配置。

前缀为 sap-endpointKind-server 的 URI 格式用于定义服务器端点（换句话说，Camel 消费者端点）和 serverName 是来自 SAP 实例的特定入站连接的名称。在组件级别命名和配置入站连接。

RFC 端点 URI 的其他组件如下：

rfcName

（必需） 在目标端点 URI 中，是由连接的 SAP 实例中端点调用的 RFC 名称。在服务器端点 URI 中，是从连接的 SAP 实例调用时由端点处理的 RFC 名称。

queueName

指定此端点将 SAP 请求发送到的队列。

IDoc 端点 URI 的其他组件如下：

idocType

（必需） 指定此端点生成的 IDoc 类型的基本 IDoc 类型。

idocTypeExtension

指定此端点生成的 IDoc Type Extension （若有）。

systemRelease

指定此端点生成的 IDoc 的关联的 SAP Basis 版本（若有）。

applicationRelease

指定此端点生成的 IDoc 的关联的应用程序发行版本（若有）。

queueName

指定此端点将 SAP 请求发送到的队列。

111.2.1. 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	默认值	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

111.2.2. 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	默认值	描述
有状态	**false**	如果为 **true**，指定此端点启动 SAP 有状态会话。
propagateExceptions	**false**	（仅限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器。

111.2.3. IDoc List Server 端点的选项

SAP IDoc List Server 端点(sap-idoclist-server)支持以下 URI 选项：

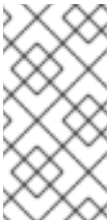
Name	默认值	描述
有状态	**false**	如果为 **true**，指定此端点启动 SAP 有状态会话。
propagateExceptions	**false**	如果为 **true**，指定此端点将异常传播到 SAP 中的调用者，而不是交换的异常处理程序。

111.2.4. 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

Camel SAP Synchronous Remote Function Call Destination Camel 组件.当 Camel 路由需要同步向来自 SAP 系统的请求和响应时，应使用此端点。



注意

此组件使用的 sRFC 协议为 SAP 系统提供请求和响应，并 尽力尽力。如果在发送请求时出现通信错误，接收 SAP 系统中的远程函数调用的完成状态将保持不变。

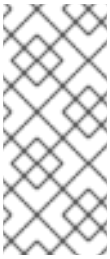
sap-trfc-destination

Camel SAP Transactional Remote Function Call Destination Camel 组件.如果请求必须发送

到 一次接收 SAP 系统，则应使用此端点。要达到此目的，组件会生成一个事务 ID tid，它包括了通过路由交换中组件发送的每个请求。接收 SAP 系统会在发送请求前记录与请求相关的 tid ；如果 SAP 系统使用相同的 tid 来再次接收请求，则不会发送请求。因此，如果路由在通过此组件的端点发送请求时遇到通信错误，它可以重试在同一交换中发送请求，知道它将只交付并执行一次。

注意

此组件使用的 tRFC 协议是异步的，且不会返回响应。因此，此组件的端点不会返回响应消息。

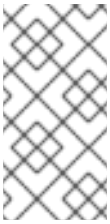


注意

此组件不保证一系列请求的顺序通过其端点，这些请求的交付和执行顺序可能会在接收 SAP 系统时有所不同，因为通信错误和重新发送请求。有关保证交付顺序，请参阅 Camel SAP Queued Remote Function Call Destination Camel 组件。

sap-qrfc-destination

Camel SAP Queued Remote Function Call Destination Camel 组件.此组件通过添加 Transactional Remote Function Call Destination camel 组件来扩展 Transactional Remote Function Call Destination camel 组件的功能，以便通过 端点向请求交付保证。当一系列请求相互依赖的情况下，应使用此端点，并且必须 最多 发送到接收 SAP 系统，并按顺序 发送到接收 SAP 系统。组件使用与 Camel SAP Transactional Remote Function Call Destination Camel 组件相同的机制完成 一次 交付保证。排序保证是通过将 SAP 系统收到的请求顺序序列化为 入站队列来完成的。入站队列由 SAP 中的 QIN 调度程序 处理。激活 入站队列后，QIN 调度程序将按顺序执行队列请求。



注意

此组件使用的 qRFC 协议是异步的，且不会返回响应。因此，此组件的端点不会返回响应消息。

sap-srfc-server

Camel SAP Synchronous Remote Function Call Server Camel 组件.当需要 Camel 路由同步处理来自 SAP 系统的请求并响应 SAP 系统时，应使用此组件及其端点。

sap-trfc-server

Camel SAP Transactional Remote Function Call Server Camel 组件。在发送 SAP 系统最多需要向 Camel 路由发送请求时，应使用此端点。为了达到此目的，发送 SAP 系统会生成事务 ID tid，它会在每次发送到组件的端点时发出一个事务 ID。发送 SAP 系统首先检查某个组件是否收到了给定的 tid，然后再发送与 tid 关联的一系列请求。组件将检查其维护的接收的 tid列表，如果该列表中没有记录所发送的 tid，然后响应发送 SAP 系统，指示是否已经记录了 tid。如果之前未记录 tid，则

发送 SAP 系统之后才会发送一系列请求。这可让发送 SAP 系统向 camel 路由可靠地发送一系列请求。

sap-idoc-destination

Camel SAP IDoc Destination Camel 组件.当 Camel 路由向 SAP 系统发送一个 Intermediate Documents (IDocs)列表时，应使用此端点。

sap-idoclist-destination

Camel SAP IDoc List Destination Camel 组件.如果 Camel 路由向 SAP 系统发送一个中间文档列表(IDocs)列表时，应使用此端点。

sap-qidoc-destination

Camel SAP Queued IDoc Destination Camel 组件.如果需要 Camel 路由列表向 SAP 系统发送一个中间文档(IDocs)，则应使用此组件及其端点。

sap-qidoclist-destination

Camel SAP Queued IDoc List Destination Camel 组件.当 camel 路由按顺序向 SAP 系统发送 Intermediate 文档(IDocs)列表时，会使用此组件及其端点。

sap-idoclist-server

Camel SAP IDoc List Server Camel 组件。在发送 SAP 系统需要向 Camel 路由提供 Intermediate Document 列表时，应使用此端点。这个组件使用 tRFC 协议与 SAP 通信，如 sap-trfc-server-standalone quick start 所述。

111.2.5. 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使 RFC 调用在 SAP 中的 ABAP 功能模块之外。RFC 目标端点配置为通过特定到 SAP 实例的连接对特定的 ABAP 功能调用特定的 ABAP 功能。RFC 目的地是出站连接的逻辑设计，具有唯一的名称。RFC 目的地由一组称为 目标数据 的连接参数指定。

RFC 目标端点将从其接收的 IN-OUT 交换的输入消息中提取 RFC 请求，并在向 SAP 的函数调用中分配该请求。函数调用的响应将在交换的输出消息中返回。由于 SAP RFC 目标端点只支持出站通信，所以 RFC 目标端点只支持创建制作者。

111.2.6. SAP RFC 服务器端点

RFC 服务器端点支持来自 SAP 的入站通信，它允许 SAP 中的 ABAP 应用程序在服务器端点中进行 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 服务器由一组连接参数指定，称为 服务器数据。

RFC 服务器端点将处理传入的 RFC 请求，并将其作为 IN-OUT 交换的输入消息进行分配。交换的输出消息将作为 RFC 调用的响应返回。由于 SAP RFC 服务器端点只支持入站通信，因此 RFC 服务器端点仅支持创建用户。

111.2.7. SAP IDoc 和 IDoc 列出目标端点

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 文档代表一个业务事务，可以轻松地与非 SAP 系统交换。IDoc 目的地由一组称为 目标数据 的连接参数指定。

IDoc 列表目的地端点与 IDoc 目的地端点类似，但其处理的消息由 IDoc 文档列表组成。

111.2.8. SAP IDoc list服务器端点

IDoc 列表服务器端点支持 SAP 的入站通信，使 Camel 路由能够从 SAP 系统接收 IDoc 文档列表。IDoc 列表服务器由一组连接参数指定，称为 服务器数据。

111.2.9. 元数据软件仓库

元数据存储库用于存储以下种类的元数据：

功能模块的接口描述

JCo 和 ABAP 运行时使用此元数据来检查 RFC 调用，以确保通信合作伙伴之间的数据安全传输，然后再分配这些调用。存储库填充了存储库数据。存储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及其输入信息的元数据，它们从功能模块传递，并且具有它描述的 function 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 IDoc 文档在发送到通信合作伙伴之前被正确格式化。基本 IDoc 类型由名称、允许片段列表以及网段之间的分层关系描述。可以在网段上实施一些额外的限制：段可以是强制或可选；可以为每个段指定最小/最大范围（定义该段的允许重复次数）。

因此，SAP 目标和服务器端点需要访问存储库，以发送和接收 RFC 调用并接收 IDoc 文档。对于 RFC 调用，由端点调用和处理的所有功能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。目标和服务器端点使用的存储库的位置在目标数据及其对应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位于 SAP 系统中，并且默认为它所连接的 SAP 系统。此默认值不需要目标数据中的显式配置。此外，目标端点进行的远程函数调用的元数据将已存在于其调用的任何现有功能模块的存储库中。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会驻留在 SAP 系统的存储库中，而是由驻留在 SAP 组件中的存储库提供。SAP 组件维护一个命名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。

111.3. CONFIGURATION

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 服务器数据存储在特殊的配置 对象 SapConnectionConfiguration 上配置，该对象自动注入到 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 必须直接在相关的 SAP 组件上配置。

111.3.1. 配置概述

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 存储目的地名称、属性 serverDataStore、按服务器名称密钥的服务器数据以及 repositoryDataStore，存储以存储库名称密钥的存储库数据。这些配置必须在初始化过程中传递给组件。

示例

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）将自动注入到此 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" />
        <!-- The following property values should not be changed -->
        <property name="progid" value="QUICKSTART" />
        <property name="repositoryDestination" value="quickstartDest" />
        <property name="connectionCount" value="2" />
    </bean>
</blueprint>
```

111.3.2. 目标配置

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到 SAP 实例的不同的出站连接。每个条目的密钥是出站连接的名称，用于目标端点 URI 的 destinationName 组件，如 URI 格式部分中所述。

每个条目的值是一个目标数据配置对象 - org.fusesource.camel.component.sap.model.rfc.impl.DestinationDataImpl - 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"
```

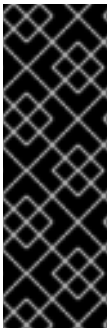
```
    class="org.fusesource.camel.component.sap.SapConnectionConfiguration">
    <property name="destinationDataStore">
      <map>
        <entry key="quickstartDest" value-ref="quickstartDestinationData" />
      </map>
    </property>
  </bean>

  <!-- 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，在配置目的地后，如前面的 Blueprint XML 文件所示，您可以使用以下 URI 在 quickstartDest 目的地上调用 BAPI_FLCUST_GETLIST 远程函数调用：

```
sap-srfc-destination:quickstartDest:BAPI_FLCUST_GETLIST
```

111.3.2.1. tRFC 和 qRFC 目的地的拦截器

前面的目标配置示例显示 CurrentProcessorDefinitionInterceptStrategy 对象的实例化。此对象在 Camel 运行时中安装拦截器，它允许 Camel SAP 组件在处理 RFC 事务时跟踪其位置。



重要

这个拦截器对于事务的 RFC 目标端点（如 sap-trfc-destination 和 sap-qrfc-destination）非常重要，且必须在 Camel 运行时中安装，以便对出站事务 RFC 通信进行正确管理。如果运行时找不到策略，则 Destination RFC Transaction Handlers 会在 Camel 日志中发出警告。在这种情况下，Camel 运行时需要重新置备并重启来正确地管理出站事务 RFC 通信。

111.3.2.2. 登录和验证选项

下表列出了在 SAP 目标数据存储中配置目的地的日志 和验证 选项：

Name	默认值	描述

client		SAP 客户端，在参数上强制记录.
user		在 user 上，登录基于密码的身份验证的参数。
aliasUser		可以使用登录用户别名，而不是登录 user。
userId		用于登录到 ABAP AS 的用户身份。JCo 运行时使用 SSO/assertion ticket、证书、当前用户、或 SNC 环境进行身份验证。如果用户 ID 是强制的，如果没有设置用户或用户别名。此 ID 从不会发送到 SAP 后端，它将被 JCo 运行时在本地使用。
passwd		在 password 上登录，请登录基于密码的身份验证的参数。
lang		如果未定义，则使用默认的用户语言。
mysapsso2		使用指定的 SAP Cookie Version 2 作为基于 SSO 验证的票据的日志。
x509cert		使用指定的 X509 证书进行基于证书的身份验证。
lcheck		在第一次调用 1（启用）前，验证才会生效。只在特殊情况下使用。
useSapGui		使用可见、隐藏或不使用 SAP GUI
codePage		其他 log on 参数，以定义用于转换日志参数的代码页面。只在特殊情况下使用。
getsso2		在登录后，目标属性中提供了获取的票据。
denyInitialPassword		如果设置为 **1**，则使用初始密码将导致异常（默认为 **0**）。

111.3.2.3. 连接选项

下表列出了在 SAP 目标数据存储中配置目标的 连接选项 ：

Name	默认值	描述
saprouter		用于连接到 SAP 路由器后面的系统的 SAP 路由器字符串。SAP Router 字符串包含 SAP 路由器链及其端口号，格式为：**(/H/<host>[/S/<port>])+**。
sysnr		SAP ABAP 应用服务器的系统数量，对于直接连接是必需的。
ashost		SAP ABAP 应用服务器，对于直接连接是必需的。
mshost		SAP 消息服务器，用于负载平衡连接的强制属性。
msserv		SAP 消息服务器端口，用于负载平衡连接的可选属性。为了解析服务名称 sapmsXXX，在 **etc/services** 中由操作系统的网络层执行查询。如果使用端口号而不是符号服务名称，则不会执行查找，且不需要额外的条目。
gwhost		允许指定一个 Concrete 网关，该网关用于建立与应用服务器的连接。如果没有指定，则使用应用服务器上的网关。
gwserv		使用 gwhost 时，应设置 gwhost。允许指定该网关上使用的端口。如果没有指定，则使用应用服务器上网关的端口。为了解析服务名称 sapgwXXX，在 etc/services 中由操作系统的网络层执行查询。如果使用端口号而不是符号服务名称，则不会执行查找，且不需要额外的条目。
r3name		SAP 系统的系统 ID，这是负载平衡连接的强制属性。
group		SAP 应用服务器组，用于负载平衡连接的强制属性。

network	**LAN**	根据 JCo 和您的目标系统之间的网络质量设置这个值，以优化性能。有效值为 **LAN** 或 **WAN**（仅与快速序列化）相关。如果将 **网络配置选项** 设置为 **WAN**，则使用较慢但效率更高的压缩算法，并分析数据以进一步压缩选项。如果将 **网络配置** 设置为 **LAN**，则使用非常快速的压缩算法，且数据分析仅在非常基本级别执行。当您设置 **LAN** 选项时，压缩率效率不高，但网络传输时间被视为非常显著。默认设置为 **LAN**。
serializationFormat	**rowBased**	有效值为 line **Based** 或 **columnbased**。对于快速序列化 **列，必须设置基于** 快速序列化列。默认序列化设置基于 **行**。

111.3.2.4. 连接池选项

下表列出了在 SAP 目标数据存储中配置目标的 连接池 选项：

Name	默认值	描述
peakLimit	**0**	可以同时为目标创建的最大活跃出站连接数。值 **0** 允许无限数量活跃的连接。否则，如果值小于 **jpoolCapacity** 的值，它将会自动增加到这个值。默认设置是 **poolCapacity** 的值，或者如果未指定 **poolCapacity**，则默认为 **0**（无限）。
poolCapacity	**1**	目标保持打开的最大空闲出站连接数。值 **0** 具有没有连接池的影响（默认为 **1**）。
expirationTime		以毫秒为单位，目标内部持有的空闲连接可以关闭。
expirationPeriod		目标检查已发布的连接过期的时间（以毫秒为单位）。
maxGetTime		如果应用程序已分配最大允许连接数，则等待连接的最大时间（以毫秒为单位）。

111.3.2.5. 安全网络连接选项

下表列出了在 SAP 目标数据存储中配置目标 的安全网络选项 ：

Name	默认值	描述
sncMode		安全网络连接(SNC)模式、**0** (off) 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		拥有的 SNC 名称。覆盖环境设置。
sncLibrary		提供 SNC 服务的库的路径。

111.3.2.6. 仓库选项

下表列出了在 SAP 目标数据存储中配置目标 的存储库 选项：

Name	默认值	描述
repositoryDest		指定用作存储库的目的地。
repositoryUser		如果没有设置存储库目的地，并且设置了此属性，它将用作存储库调用的用户。这可让您使用不同的用户进行存储库查找。
repositoryPasswd		存储库用户的密码。如果使用了存储库用户，则需要此项。
repositorySnc		**（可选）** 如果将 SNC 用于此目的地，如果此属性设置为 0，则可以为存储库连接关闭它。**默认设置为 jco.client.snc_mode 的值。只适用于特殊情况。**

repositoryRoundtripOptimization		启用 **RFC_METADATA_GET** API，该 API 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** 取消激活 ABAP System 中的 **RFC_METADATA_GET**。 如果没有设置属性，则目的地最初会进行远程调用来检查 **RFC_METADATA_GET** 是否可用。如果可用，目标将使用它。 **注意：** 如果存储库已经初始化（例如，因为它被其他目的地使用），则此属性没有任何效果。通常，此属性与 ABAP 系统相关，所有目的地中应该具有相同的值，指向同一 ABAP 系统。有关后端先决条件，请参阅备注 1456826。

111.3.2.7. 跟踪配置选项

下表列出了在 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 跟踪(**0** 或 **1**)。
cpicTrace		启用/禁用 CPIC 追踪 **[0..3]**。

111.3.3. 服务器配置

服务器配置在 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。每个条目的密钥是出站连接的名称，用于服务器端点 URI 的 serverName 组件，如 URI 格式部分中所述。

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" />
        <!-- The following property values should not be changed -->
        <property name="progid" value="QUICKSTART" />
        <property name="repositoryDestination" value="quickstartDest" />
        <property name="connectionCount" value="2" />
    </bean>
</blueprint>
```

请注意，本例还如何配置一个目标连接 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。如果没有配置这个选项，您必须创建一个本地元数据存储库。

例如，在配置目的地后，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 远程函数调用：

> *sap-srfc-server:quickstartServer:BAPI_FLCUST_GETLIST*

111.3.3.1. 所需的选项

服务器数据配置对象所需的选项如下：

Name	默认值	描述
gwhost		应该注册服务器连接的网关主机。
gwserv		网关服务，即可以在其上完成注册的端口。为了解析服务名称 **sapgwXXX**，**etc/services** 中的查询由操作系统的网络层执行。如果使用端口号而不是符号服务名称，则不会执行查找，且不需要额外的条目。
progid		完成注册的程序 ID。作为网关上的标识符，并在 ABAP 系统的目标中作为标识符。
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		应在网关上注册的连接数。

111.3.3.2. 安全网络连接选项

服务器数据配置对象的安全网络连接选项如下：

Name	默认值	描述
sncMode		安全网络连接(SNC)模式、**0** (off) 或 **1** (on)。
sncQop		SNC 安全级别，**1** 到 **9**。
sncMyname		服务器的 SNC 名称。覆盖默认的 SNC 名称。通常像 **p:CN=JCoServer, O=ACompany, C=EN** 类似。

sncLib		提供 SNC 服务的库路径。如果没有提供此属性，则改为使用 **jco.middleware.snc_lib** 属性的值。

111.3.3.3. 其他选项

服务器数据配置对象的其他选项如下：

Name	默认值	描述
saprouter		用于受防火墙保护的系统的 SAP 路由器字符串，因此仅在该 ABAP 系统网关上注册服务器时，才能通过 SAProuter 访问。一个典型的路由器字符串是 **/H/firewall.hostname/H/**。
maxStartupDelay		出现故障的两个启动尝试之间的最大时间（以秒为单位）。每次启动失败后，等待时间从最初 1 秒开始加倍，直到达到最大值或服务器可以成功启动。
trace		启用/禁用 RFC 跟踪(**0** 或 **1)**
workerThreadCount		服务器连接使用的最大线程数量。如果没有设置，则 **connectionCount** 的值将用作 **workerThreadCount**。最大线程数量不能超过 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**。

111.3.4. 仓库配置

存储库配置在 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，此键也对应于此存储库所附加的服务器的名称。

每个条目的值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.fusesource.camel.component.sap.model.rfc.impl.FunctionTemplateImpl)的映射。此映射中的每个条目都指定 function 模块的接口，每个条目的键是指定的 function 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>
```

111.3.4.1. 功能模板属性

功能模块的接口由四个参数列表组成，其数据被回传输至 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 参数列表包含在 RFC 调用中发送到功能模块的参数值 ；
- export 参数列表包含由 RFC 调用中的 function 模块返回的参数值 ；
- changing 参数列表包含由 RFC 调用中的 function 模块发送到和返回的参数值 ；

- *table 参数列表 包含由 RFC 调用中的 function 模块发送到和返回的内部表值。*

- *功能模块的接口还包括 ABAP 异常 异常列表，当模块在 RFC 调用中调用时，可能会引发该异常。*

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 RFC 调用中发送到功能模块的参数。
changingParameterList	列表字段元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.ListFieldMeataDataImpl**。指定 RFC 调用中与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 异常元数据对象列表 **org.fusesource.camel.component.sap.model.rfc.impl.AbapExceptionImpl**。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的概述：

```
<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>
```

111.3.4.2. 列出字段元数据属性

list 字段元数据对象 org.fusesource.camel.component.sap.model.rfc.impl.ListFieldMeataDataImpl，指定参数列表中的字段的名称和类型。对于元素参数字段 (CHAR,DATE,BCD,TIME,BYTE,NUM,FLOAT,INT,INT1,INT2,DECF16,DECF34,STRING,XSTRING)，下表列出了可在字段元数据对象上设置的配置属性：

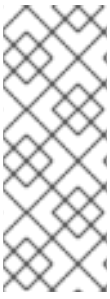
Name	默认值	描述
name	-	参数字段的名称。
type	-	字段的参数类型。
byteLength	-	非Unicode 布局的字段长度（以字节为单位）。这个值取决于参数类型。
unicodeByteLength	-	Unicode 布局的字段长度（以字节为单位）。这个值取决于参数类型。
十进制	**0**	字段值中的十进制数。参数类型 BCD 和 FLOAT 需要。

optional	**false**	如果为 **true**，则该字段是可选的，且不需要在 RFC 调用中设置。

*请**注意，所有元素参数字段都需要在字段 metadata 对象中指定名称,type,byteLength, 和 unicodeByteLength 属性。此外， BCD、FLOAT、DECF16 和 DECF34 字段需要字段元数据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， 下表列出了可在列表字段元数据对象中设置的配置属性：***

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。一个记录元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.RecordMetaDataImpl**，用于指定结构或表行中的字段。
optional	**false**	如果为 **true**，则该字段是可选的，且不需要在 RFC 调用中设置。



注意

所有复杂的参数字段都需要在字段 metadata 对象中指定 名称、type 和 recordMetaData 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，用于指定嵌套结构的结构或表行的结构。

Elementary list 字段元数据示例

以下元数据配置指定了一个可选的 24 位打包的 BCD 号参数，其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名为 **CONNINFO** 的必要 **TABLE** 参数，其中包含 **connectionInfo** 记录元数据对象指定的行结构：

```
<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>
```

111.3.4.3. 记录元数据属性

记录元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.RecordMetaDataImpl**，指定嵌套的 **STRUCTURE** 或 **TABLE** 参数所在的行。记录元数据对象维护记录字段元数据对象列表 **org.fusesource.camel.component.sap.model.rfc.impl.FieldMetaDataImpl**，用于指定驻留在嵌套结构或表行中的参数。

下表列出了可在记录元数据对象中设置的配置属性：

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>
```

111.3.4.4. 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**，指定结构中的参数字段的名称和类型。

记录字段元数据对象与参数字段元数据对象类似，但还必须额外指定嵌套结构或表行中各个字段位置的偏移量。单个字段的非Unicode 和 Unicode 偏移量必须从结构或行中前字段的非Unicode 和 Unicode 字节长度的总和指定。

注意

在嵌套结构和表行中正确指定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 调用中正确传输值。

对于元素参数字段 **(CHAR,DATE,BCD,TIME,BYTE,NUM,FLOAT,INT,INT1,INT2,DECF16,DECF34,STRING,XSTRING)**，下表列出了可在记录字段元数据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。
byteLength	-	非Unicode 布局的字段长度（以字节为单位）。这个值取决于参数类型。
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。

byteOffset	-	非Unicode 布局的字段偏移（以字节为单位）。这个偏移是括起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）。这个偏移是括起结构中字段的字节位置。
十进制	**0**	字段值中的十进制数；对于参数类型 **BCD** 和 **FLOAT** 才需要。

*对于类型为 **TABLE 或 STRUCTURE 的复杂参数字段，下表列出了可在记录字段元数据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）。这个偏移是括起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）。这个偏移是括起结构中字段的字节位置。
recordMetaData	-	结构或表的元数据。一个记录元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.RecordMetaDataImpl**，用于指定结构或表行中的字段。

Elementary record 字段元数据示例

以下元数据配置指定了名为 ARR DATE 的 DATE 字段参数，它位于 85 字节，当非Unicode 布局时，将 170 字节置于 Unicode 布局时，存在 170 字节。

```
<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名为 FLTINFO 的 STRUCTURE 字段参数，其结构由 flightInfo 记录元数据对象指定。参数位于出现非Unicode 和 Unicode 布局时的括起结构的开头。

```
<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>
```

111.4. 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的地名称。
CamelSap.serverName	处理消息的最后一个服务器端点的服务器名称。

CamelSap.queueName	处理消息的最后一个排队端点的队列名称。
CamelSap.rfcName	处理消息的最后一个 RFC 端点的 RFC 名称。
CamelSap.idocType	用于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）用于处理消息的最后一个 IDoc 端点。
CamelSap.applicationRelease	应用程序发行版本（若有）用于处理消息的最后一个 IDoc 端点。

111.5. EXCHANGE PROPERTIES

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 目标属性的映射。映射按目的地名称键，每个条目都是 **java.util.Properties** 对象，其中包含该目的地的配置属性。
CamelSap.serverPropertiesMap	包含交换遇到的每个 SAP 服务器属性的映射。该映射按服务器名称键，每个条目都是 **java.util.Properties** 对象，其中包含该服务器的配置属性。

111.6. RFC 的消息正文

111.6.1. 请求和响应对象

SAP 端点希望接收一条消息，其中包含 SAP 请求对象的消息正文，并将返回一个包含 SAP 响应对象的消息正文。SAP 请求和响应是固定映射数据结构，其中包含带有预定义数据类型的命名字段。

*请**注意，SAP 请求和响应中的指定字段特定于 SAP 端点，每个端点定义 SAP 请求中的参数并将其接受。SAP 端点提供工厂方法来创建特定于它的请求和响应对象。***

```
public class SAPEndpoint ... {
    ...
    public Structure getRequest() throws Exception;
```

```
    public Structure getResponse() throws Exception;
    ...
}
```

111.6.2. 结构对象

SAP 请求和响应对象在 Java 中都表示为支持 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通过映射界面中字段的 getter 方法访问。此外，结构接口提供了一种类型限制的方法来检索字段值。

结构对象使用 Eclipse Modeling Framework (EMF)在组件运行时实现，并支持该框架的 EObject 接口。结构对象的实例附加了元数据，用于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。详情请查看 EMF 文档。

注意

尝试获取结构对象上未定义的参数将返回 null。尝试设置在结构上未定义的参数将抛出异常，并尝试使用不正确的类型设置参数的值。

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 的值的字段。

注意

不需要创建这些类型的实例，并将它们添加到结构中。如果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。

111.6.3. 字段类型

驻留在 SAP 请求结构对象内或响应的字段可能是 元素 或复杂。elementary 字段包含单个 scalar

值，而复杂的字段将包含元素或复杂类型的一个或多个字段。

111.6.3.1. Elementary 字段类型

元素字段可以是字符、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的类型：

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 Type 'C': Fixed sized 字符字符串
DATE	**java.util.Date**	8	16	-	ABAP Type 'D': Date (format: YYYYMMDD)
BCD	**java.math.BigDecimal**	1 到 16	1 到 16	0 到 14	ABAP Type 'P': Packed BCD number.BCD 号为每个字节包含两个数字。
时间	**java.util.Date**	6	12	-	ABAP Type 'T': Time (format: HHMMSS)
BYTE	**byte[]**	1 到 65535	1 到 65535	-	ABAP Type 'X':Fixed sized byte array
NUM	**java.lang.String**	1 到 65535	1 到 65535	-	ABAP Type 'N': Fixed sized 数字字符字符串
浮点值	**java.lang.Double**	8	8	0 到 15	ABAP Type 'F': Floating point number
INT	**java.lang.Integer**	4	4	-	ABAP Type 'I': 4 字节 Integer
INT2	**java.lang.Integer**	2	2	-	ABAP Type 'S': 2 字节 Integer

INT1	**java.lang.Integer**	1	1	-	ABAP Type 'B': 1-byte Integer
DECF16	**java.match.BigDecimal**	8	8	16	ABAP Type 'decfloat16': 8-byte Decimal Floating Point Number
DECF34	**java.math.BigDecimal**	16	16	34	ABAP Type 'decfloat34': 16-byte Decimal Floating Point Number
字符串	**java.lang.String**	8	8	-	ABAP Type 'G': Variable length 字符字符串
XSTRING	**byte[]**	8	8	-	ABAP Type 'Y': Variable length byte array

111.6.3.2. 字符字段类型

character 字段包含一个固定大小的字符串，可在底层 JCo 和 ABAP 运行时中使用非Unicode 或 Unicode 字符编码。非统一字符字符串对每个字节一个字符进行编码。Unicode 字符字符串使用 UTF-16 编码以两个字节编码。字符字段值在 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 8 个字符日期字段，年份、月份和日期格式为 YYYYMMDD，对应于 ABAP 类型 D。

时间

TIME 字符字段是一个 6 个字符时间字段，其小时、分钟和秒格式为 HHMMSS，对应于 ABAP 类型 T。

111.6.3.3. 数字字段类型

一个数字字段包含一个数字。支持以下数字字段类型：

INT

INT 数字字段是以 4 字节整数值存储在底层 JCo 和 ABAP 运行时中的整数字段，对应于 ABAP 类型 I。INT 字段值以 Java 表示为 java.lang.Integer 对象。

INT2

INT2 数字字段是以 2 字节整数值存储在底层 JCo 和 ABAP 运行时中的整数字段，对应于 ABAP 类型 S。一个 INT2 字段值表示为 java.lang.Integer 对象。

INT1

INT1 字段是一个整数值，存储在底层 JCo 和 ABAP 运行时值中，并对应于 ABAP 类型 B。INT1 字段值以 Java 表示为 java.lang.Integer 对象。

浮点值

FLOAT 字段是一个二进制浮动点数字段，它存储为底层 JCo 和 ABAP 运行时中的 8 字节双值，对应于 ABAP 类型 F。FLOAT 字段声明了字段在相关十进制属性中包含的十进制数字。对于 FLOAT 字段，此十进制属性可以在 1 到 15 位之间具有值。FLOAT 字段值在 Java 中表示为 java.lang.Double 对象。

BCD

BCD 字段是一个二进制代码十进制字段，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。单位数字为每个字节存储两个十进制数字。BCD 字段在关联的 byteLength 和 unicodeByteLength 属性中声明其字段长度。对于 BCD 字段，这些属性可以在 1 到 16 字节之间具有值，并且两个属性都具有相同的值。BCD 字段声明字段值包含在其关联的十进制属性中的十进制数字。对于 BCD 字段，这个十进制属性可以在 1 到 14 位之间具有值。BCD 字段值以 Java 表示，作为 java.math.BigDecimal。

DECF16

DECF16 字段是以 8 字节 IEEE 754 decimal64 浮点值存储在底层 JCo 和 ABAP 运行时中的十进制浮点值，对应于 ABAP 类型 decfloat16。DECF16 字段的值有 16 个十进制数字。DECF16 字段的值以 Java 表示，以 java.math.BigDecimal 表示。

DECF34

DECF34 字段是以 16 字节 IEEE 754 decimal128 浮点值存储在底层 JCo 和 ABAP 运行时中的十进制浮动点，对应于 ABAP 类型 decfloat34。DECF34 字段的值有 34 个十进制数字。DECF34 字段的值以 Java 表示，以 java.math.BigDecimal 表示。

111.6.3.4. 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。一个 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，这些属性可以在 1 到 65535 字节之间具有值，并且两个属性都具有相同的值。BYTE 字段的值以 Java 表示，作为 byte[] 对象。

111.6.3.5. 字符串字段类型

字符串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修复。字符串值的存储会在底层 JCo 和 ABAP 运行时中动态创建。string 字段本身的存储已被修复，仅包含字符串标头。

字符串

STRING 字段将存储在底层 JCo 和 ABAP 运行时中的字符字符串引用为 8 字节值。它对应于 ABAP 类型 G。STRING 字段的值在 Java 中表示为 java.lang.String 对象。

XSTRING

XSTRING 字段引用存储在底层 JCo 和 ABAP 运行时中的字节字符串，作为 8 字节值。它对应于 ABAP 类型 Y。STRING 字段的值在 Java 中表示为 byte[] 对象。

111.6.3.6. 复杂的字段类型

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数	单个字段 Unicode 字节长度的总	-	ABAP Type 'u' & 'v': Heterogeneous Structure
表	**org.fusesource.camel.component.sap.model.rfc.Table**	行结构的字节长度	Unicode 行结构的字节长度	-	ABAP Type 'h': Table

111.6.3.7. 结构字段类型

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。

111.6.3.8. 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。字段的值由带有接口 org.fusesource.camel.component.sap.model.rfc.Table 的表对象来表示。

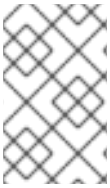
111.6.3.9. 表对象

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。此接口同时扩展了 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 a table row at the end of the row list
     */
    S add();

    /**
     * Creates and adds a table row at the index in the row list
     */
    S add(int index);

}
```

表对象中的行列表通过列表界面中定义的标准方法访问和管理。另外，表接口提供了两个工厂方法，用于创建和添加结构对象到行列表中。

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。表对象的实例附加了元数据，用于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。详情请查看 EMF 文档。



注意

尝试添加或设置错误类型的行结构值将抛出异常。

111.7. IDOC 的消息正文

111.7.1. idoc 消息类型

当使用其中一个 IDoc Camel SAP 端点时，消息正文的类型取决于您使用的特定端点。

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，消息正文是 DocumentList 类型：

```
org.fusesource.camel.component.sap.model.idoc.DocumentList
```

111.7.2. IDoc 文档模型

对于 Camel SAP 组件，IDoc 文档使用 Eclipse Modeling Framework (EMF)建模，它围绕底层 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 类型代表 IDoc 文档实例。简而言之，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();
}
```

以下方法由 Document 接口公开：

访问**控制**记录**的方法**

大多数方法都用于访问**或修改 IDoc 控制**记录**的字段**值。这**些方法是 AttributeName 的形式，其中 AttributeName 是字段**值**的名称。**

访问**文档内容的方法**

getRootSegment 方法提供对**文档内容(IDoc 数据**记录**)的**访问**，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片段，片段可以嵌套到任意程度。**

但请**注意，片段**层**次**结构**的精确布局由文档的特定 IDoc** 类**型定**义**。在**创**建（或**读**取）片段**层**次**结构时，您必**须**确保遵循 IDoc** 类**型定**义**的确切**结构**。**

分段类**型用于**访问 **IDoc 文档的数据**记录，其中片段根据文档的 **IDoc** 类**型定**义**的**结构**而定。**简**而言之，分段 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 an 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) 方法对于向段添加新（嵌套）子级特别有用。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 Java 代码，如下所示：

```
Segment rootSegment = document.getRootSegment();

Segment E1SCU_CRE_Segment = rootSegment.getChildren("E1SCU_CRE").add();
```

111.7.3. 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 对象上的属性表示。

数据记录

数据记录由 Segment 对象表示，后者作为片段的嵌套层次结构构建。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 status 值。

创建文档实例示例

以下示例演示了如何使用 Java 中的 IDoc 模型 API 创建 IDoc 类型 FLCUSTOMER_CREATEFROMDATA01。

例 111.1. 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 modeling 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");
```

111.8. 文档属性

idoc 文档属性表显示您可以在 Document 对象上设置的控制记录属性。

表 111.2. idoc 文档属性

属性	length	SAP 字段	描述
archiveKey	70	**ARCKEY**	EDI 归档密钥
client	3	**MANDT**	客户端
creationDate	8	**CREDAT**	日期 IDoc 被创建
creationTime	6	**CRETIM**	创建时间 IDoc
direction	1	**直接**	方向
eDIMessage	14	**REFMES**	引用消息
eDIMessageGroup	14	**REFGRP**	引用消息组

属性	length	SAP 字段	描述
eDIMessageType	6	**STDMES**	EDI 消息类型
eDIStandardFlag	1	**STD**	EDI 标准
eDIStandardVersion	6	**STDVRS**	EDI 标准版本
eDITransmissionFile	14	**INT**	对交换文件的引用
iDocCompoundType	8	**DOCTYP**	idoc 类型
iDocNumber	16	**文档NUM**	idoc number
iDocSAPRelease	4	**DOCREL**	SAP 版本的 IDoc
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**	接收器合作伙伴数量
recipientPartnerType	2	**RCVPRT**	接收器合作伙伴类型
recipientPort	10	**RCVPOR**	接收器端口(SAP 系统、EDI 子系统)
senderAddress		**SNDSAD**	发件人地址(SADR)

属性	length	SAP 字段	描述
senderLogicalAddress	70	**SNDLAD**	发件人的逻辑地址
senderPartnerFunction	2	**SNDPFC**	发件人合作伙伴功能
senderPartnerNumber	10	**SNDPRN**	发件人合作伙伴数量
senderPartnerType	2	**SNDPRT**	发件人合作伙伴类型
senderPort	10	**SNDPOR**	发件人端口(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

111.8.1. 在 Java 中设置文档属性

在 Java 中设置控制记录属性时，Java bean 属性的常见约定如下：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，iDocType,iDocTypeExtension, 和 messageType 属性可以在 Document 对象中设置，如下所示：

```
// Java
document.setIDocType("FLCUSTOMER_CREATEFROMDATA01");
document.setIDocTypeExtension("");
document.setMessageType("FLCUSTOMER_CREATEFROMDATA");
```

111.8.2. 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，iDocType,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>
```

111.9. 事务支持

111.9.1. BAPI 事务模型

SAP 组件支持 BAPI 事务模型与 SAP 进行出站通信。如果需要，一个包含 transacted 选项设为 true 的 URL 的目标端点，将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。此同步对象将调用 BAPI 服务方法 BAPI_TRANSACTION_COMMIT，并在消息交换完成时结束有状态会话。如果消息交换的处理失败，同步对象将调用 BAPI 服务器方法 BAPI_TRANSACTION_ROLLBACK 并终止有状态会话。

111.9.2. RFC 事务模型

tRFC 协议通过识别具有唯一事务标识符(TID)的每个事务请求来实现 AT-MOST-ONCE 交付和处理保证。TID 附带协议中发送的每个请求。使用 tRFC 协议发送应用程序必须在发送请求时标识带有唯一 TID 的请求的每个实例。应用程序可能会多次发送带有给定 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。应用程序可以选择在发送请求时遇到通信或系统错误时重新发送带有给定 TID 的请求，因此与在接收系统中发送和处理请求时的不确定性。通过在遇到通信错误时重新发送请求，使用 tRFC 协议的客户端应用可以确保 EXACTLY-ONCE 传输和处理其请求保证。

111.9.3. 要使用哪种事务模型？

BAPI 事务是一个应用程序级别的事务，它代表对 SAP 数据库中 BAPI 方法或 RFC 功能执行持久数据更改实施 ACID 保证。RFC 事务是一个通信事务，它对 BAPI 方法和/或 RFC 功能的请求实施交付保证 (AT-MOST-ONCE, EXACTLY-ONCE-IN-ORDER)。

111.9.4. 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 目标。这意味着 Camel SAP 组件可能需要跟踪通过路由的每个 Exchange 对象的事务 ID (TID)。现在，如果路由处理失败且必须重试，则这种情况会非常复杂。RFC 事务语义要求，每个 RFC 目的地 都必须使用相同的 TID 调用，该 ID 首次使用（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，以便可以按照正确顺序重新执行 TID。

默认情况下，Camel 不提供一种机制，使 Exchange 能够知道它在路由中的位置。要提供这样的机制，需要将 CurrentProcessorDefinitionInterceptStrategy 拦截器安装到 Camel 运行时中。此拦截器必须安装到 Camel 运行时中，以便 Camel SAP 组件在路由中跟踪 TID。

111.9.5. 事务的 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来处理处理错误。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 服务器端点会记录故障，并且发送 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 的事务请求来再次处理请求。

111.10. RFC 的 XML SERIALIZATION

SAP 请求和响应对象支持 XML 序列化格式，使这些对象能够序列化为 XML 文档，并从 XML 文档进行序列化。

111.10.1. XML 命名空间

仓库中的每个 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 的格式如下：

```
http://sap.fusesource.org/rfc/<Repository Name>/<RFC Name>
```

RFC 命名空间 URL 有一个常见的 http://sap.fusesource.org/rfc 前缀，后跟定义了 RFC 元数据的存储库的名称。URL 中的最终组件是 RFC 本身的名称。

111.10.2. 请求和响应 XML 文档

SAP 请求对象将序列化为 XML 文档，其中包含名为 Request 的文档的根元素，并通过请求 RFC 的命名空间范围。

```
<?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**象将序列化**为 **XML 文档，其中包含名**为 **Response 的文档的根元素，并通**过响应 **RFC 的命名空**间**范**围**。**

```
<?xml version="1.0" encoding="ASCII"?>
<BOOK_FLIGHT:Response
    xmlns:BOOK_FLIGHT="http://sap.fusesource.org/rfc/nplServer/BOOK_FLIGHT">
  ...
</BOOK_FLIGHT:Response>
```

111.10.3. 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**留的参数列表中**结构**的字段名称、**结构**或表行条目。**

```
<BOOK_FLIGHT:FLTINFO
    xmlns:BOOK_FLIGHT="http://sap.fusesource.org/rfc/nplServer/BOOK_FLIGHT">
  ...
</BOOK_FLIGHT:FLTINFO>
```

请**注意，RFC 命名空**间**中的 structure 元素的**类**型名称将与定**义结构**的**记录**元数据**对**象的名称**对应，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>
```

当指定 JAXB bean to marshal 和 unmarshal the 结构时，这**种区**别**非常重要。**

111.10.4. 表字段

参数列表或嵌套结构**中的表字段被序列化**为**元素。序列化**结构**的元素名称将**对应**于它所**驻**留的参数列表、**结构**或表行条目中表的字段名称。table 元素将包含一系列用于存放表行条目的序列化**值**的行元素。**

```
<BOOK_FLIGHT:CONNINFO
    xmlns:BOOK_FLIGHT="http://sap.fusesource.org/rfc/nplServer/BOOK_FLIGHT">
   <row ... > ... </row>
   ...
   <row ... > ... </row>
</BOOK_FLIGHT:CONNINFO>
```

请注意，RFC 命名空间中的 table 元素的类型名称对应于记录元数据对象的名称，用于定义由 _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>
```

当指定 JAXB bean to marshal 和 unmarshal the 结构时，这种区别非常重要。

111.10.5. Elementary 字段

参数列表或嵌套结构中的元素字段按照括起参数列表或结构的元素上的属性进行序列化。serialized 字段的属性名称对应于它所驻留的参数列表、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"/>
```

111.10.6. 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
```

日期字段将仅用年、月、天和时区组件集序列化：

```
DEPDATE="2014-03-19T00:00:00.000-0400"
```

时间字段只会序列化，且只设置小时、分钟、秒、毫秒和时区组件：

```
DEPTIME="1970-01-01T16:00:00.000-0500"
```

111.11. IDOC 的 XML SERIALIZATION

IDoc 消息正文可以序列化为 XML 字符串格式，并帮助内置类型转换器。

111.11.1. XML 命名空间

每个序列化 IDoc 都与 XML 命名空间关联，它有以下通用格式：

```
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 文档类型)都是必需的，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///
```

111.11.2. 内置类型转换器

Camel SAP 组件有一个内置类型转换器，它能够将 Document 对象或 DocumentList 对象转换为字符串类型，以及从 String 类型转换。

例如，要将 Document 对象序列化为 XML 字符串，只需在 XML DSL 中的路由中添加以下行：

```
<convertBodyTo type="java.lang.String"/>
```

您还可以使用这种方法将 XML 消息序列化到 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"/>
```

111.11.3. XML 格式的 IDoc 消息正文示例

当您将 IDoc 消息转换为 String 时，它将被序列化为 XML 文档，其中 root 元素是 idoc:Document（用于单个文档）或 idoc:DocumentList （用于文档列表）。它显示单个 IDoc 文档已被序列化为 idoc:Document 元素。

例 111.2. 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>
```

111.12. 示例 1：从 SAP 读取数据

本例展示了从 SAP 读取 FlightCustomer 业务对象数据的路由。路由调用 FlightCustomer BAPI 方法 BAPI_FLCUST_GETLIST，使用 SAP 同步 RFC 目标端点来检索数据。

111.12.1. 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");
```

111.12.2. 路由的 XML DSL

同一路由的 Spring DSL 如下：

```
<route>
    <from uri="direct:getFlightCustomerInfo"/>
    <to uri="bean:createFlightCustomerGetListRequest"/>
    <to uri="sap-srfc-destination:nplDest:BAPI_FLCUST_GETLIST"/>
    <to uri="bean:returnFlightCustomerInfo"/>
</route>
```

111.12.3. createFlightCustomerGetListRequest bean

createFlightCustomerGetListRequest bean 负责在后续 SAP 端点的 RFC 调用中构建 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);
}
```

111.12.4. returnFlightCustomerInfo bean

returnFlightCustomerInfo bean 负责从 **SAP** 响应对象提取数据，在其从以前的 **SAP** 端点接收的交换方法中提取数据。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);

}
```

111.13. 示例 2：将数据写入 SAP

本例演示了在 SAP 中创建 FlightTrip 业务对象实例的路由。路由调用 FlightTrip BAPI 方法 BAPI_FLTRIP_CREATE，使用目标端点来创建对象。

111.13.1. 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");
```

111.13.2. 路由的 XML DSL

同一路由的 Spring DSL 如下：

```
<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>
```

111.13.3. 事务支持

请注意，SAP 端点的 URL 将 transacted 选项设置为 true。启用此选项后，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，因此此选项需要在 SAP 中永久进行路由更改。

111.13.4. 填充请求参数

createFlightTripRequest 和 returnFlightTripResponse beans 负责将请求参数填充到 SAP 请求中，并根据上例中所示的相同操作序列来提取响应参数。

111.14. 示例 3：处理 SAP 的请求

本例演示了处理从 SAP 到 BOOK_FLIGHT RFC 的请求的路由，该路由由路由实施。此外，它演示了组件的 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 FlightCustomer 创建 FlightTrip 业务对象。路由首先将 SAP 服务器端点收到的 SAP 请求对象加载到自定义 JAXB bean 中。然后，这个自定义 Bean 在交换中被多播到三个子路由，它会收集创建动态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。最终的子路由还会创建并返回自定义 JAXB bean，该 Bean 会被 marshaled 到 SAP 响应对象，并由服务器端点返回。

111.14.1. 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);
```

111.14.2. 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>
```

111.14.3. 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;

    ...
}
```

111.14.4. BookFlightResponse bean

以下列表演示了一个 JAXB bean，它被处理到 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。

111.14.5. FlightInfo bean

以下列表演示了一个 JAXB bean，它被处理到复杂结构参数 FLTINFO 的序列化形式：

```
@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;

    ...
}
```

111.14.6. ConnectionInfoTable bean

以下列表演示了一个 JAXB bean，它将对复杂表参数 CONNINFO 的序列化形式 marshals。请注意，JAXB bean 的根元素类型的名称对应于类型为 _TABLE 的行结构的名称，bean 包含行元素列表。

```
@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;

    ...
}
```

111.14.7. ConnectionInfo bean

以下列表演示了一个 JAXB bean，它被处理到上述表行元素的序列化形式：

```
@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 112 章 XQUERY

Camel 支持 XQuery 以允许一个 Expression或 Predicate 在 DSL 中使用。

例如，您可以使用 XQuery 在 Message Filter 中创建一个 predicate，或作为 Recipient List 的表达式。

112.1. 依赖项

要在 camel 路由中使用 XQuery，您需要对实现 XQuery 语言的 camel-saxon 添加依赖项。当在 Red Hat build of Camel Spring Boot 中使用 xquery 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-saxon-starter</artifactId>
</dependency>
```

112.2. XQUERY 语言选项

XQuery 语言支持 4 个选项，在以下列出。

Name	默认值	Java 类型	描述
type		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。
headerName		字符串	作为输入的标头名称，而不是消息的正文。
configurationRef		字符串	到 registry 中的用于 xquery 的一个 saxon 配置实例的引用 （需要 camel-saxon）。这可能需要在 saxon 配置中添加自定义功能，因此这些自定义功能可以在 xquery 表达式中使用。
trim		布尔值	是否修剪值以移除前导和结尾的空格和换行符。

112.3. 变量

消息正文将设置为 contextItem。另外，以下变量也可用：

变量	类型	描述
交换	Exchange	当前的交换
in.body	对象	消息正文
out.body	对象	**弃用** OUT 消息正文（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	**弃用**，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
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**。

112.4. 示例

```
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，您需要明确的类型转换，或者您将收到一个 org.w3c.dom.DOMException: HIERARCHY_REQUEST_ERR。您需要传递函数的预期输出类型。例如，concat 函数返回一个 String，它按如下方式完成：

```
from("direct:start")
  .recipientList().xquery("concat('mock:foo.', /person/@city)", String.class);
```

在 XML DSL 中：

```
<route>
  <from uri="direct:start"/>
  <recipientList>
    <xquery type="java.lang.String">concat('mock:foo.', /person/@city</xquery>
  </recipientList>
</route>
```

112.4.1. 使用命名空间

如果您有一个标准的命名空间集，并且希望在多个 XQuery 表达式之间共享它们，您可以在使用 Java

DSL 时使用 org.apache.camel.support.builder.Namespaces，如下所示：

```
Namespaces ns = new Namespaces("c", "http://acme.com/cheese");

from("direct:start")
  .filter().xquery("/c:person[@name='James']", ns)
  .to("mock:result");
```

注意如何将命名空间提供给 xquery 以及作为第二参数传递的 ns 变量。

每个命名空间都是 key=value 对，前缀是键。在 XQuery 表达式中，命名空间被前缀使用，例如：

```
/c:person[@name='James']
```

命名空间构建器支持添加多个命名空间，如下所示：

```
Namespaces ns = new Namespaces("c", "http://acme.com/cheese")
                .add("w", "http://acme.com/wine")
                .add("b", "http://acme.com/beer");
```

在 XML DSL 中使用命名空间时，如您在 XML root 标签中设置命名空间（或 camelContext, routes, route tag 之一）。

在下面的 XML 示例中，我们使用 Spring XML，其中在 root 标签 Bean 中声明命名空间，在 xmlns:foo="http://example.com/person" 一行中：

```
<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>
```

这个命名空间使用 foo 作为前缀，因此 <xquery> 表达式使用 /foo: 来使用这个命名空间。

112.5. 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);
```

如果要使用类似标头的 Camel 变量，则必须在 XQuery 表达式中显式声明它们。

```
<transform>
  <xquery>
    declare variable $in.headers.foo external;
    element item {$in.headers.foo}
  </xquery>
</transform>
```

112.6. 从外部资源载入脚本

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" 等引用您可以进行的类路径中的文件：

```
.setHeader("myHeader").xquery("resource:classpath:myxquery.txt", String.class)
```

112.7. 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*

112.8. 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.xquery.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.xquery.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.xquery.configuration	使用自定义 Saxon 配置。选项是一个 net.sf.saxon.Configuration 类型。		Configuration
camel.component.xquery.configuration-properties	设置自定义 Saxon 配置属性。		Map
camel.component.xquery.enabled	是否启用 xquery 组件的自动配置。这默认是启用的。		布尔值
camel.component.xquery.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.xquery.module-u-r-i-resolver	使用自定义 ModuleURIResolver。选项是一个 net.sf.saxon.lib.ModuleURIResolver 类型。		ModuleURIResolver

Name	描述	默认值	类型
camel.language.xquery.configuration-ref	到 registry 中的用于 xquery 的一个 saxon 配置实例的引用 （需要 camel-saxon）。这可能需要在 saxon 配置中添加自定义功能，因此这些自定义功能可以在 xquery 表达式中使用。		字符串
camel.language.xquery.enabled	是否启用 xquery 语言的自动配置。这默认是启用的。		布尔值
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

第 113 章 SCHEDULER

仅支持消费者

调度程序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，此组件使用 JDK ScheduledExecutorService。其中，作为计时器使用 JDK Timer。

您只能消耗来自此端点的事件。

113.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 调度程序 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-scheduler-starter</artifactId>
</dependency>
```

113.2. 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。

注意

生成的交换的 IN 正文为 null。因此 exchange.getIn （）.getBody （） 返回 null。

113.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别

- 端点级别

113.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

113.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

113.4. 组件选项

Scheduler 组件支持 3 个选项，如下所列。

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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
poolSize (scheduler)	调度线程池中由调度线程池使用的核心线程数量。默认情况下使用单个线程。	1	int

113.5. 端点选项

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：

113.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
name (consumer)	**必需** 调度程序的名称。		字符串

113.5.2. 查询参数(21 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
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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
同步 (advanced)	设置是否应严格使用同步处理。	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）。	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。	1000	long
poolSize (scheduler)	调度线程池中由调度线程池使用的核心线程数量。默认情况下使用单个线程。	1	int
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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
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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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

113.6. 更多信息

此组件是一个调度程序 Polling Consumer，您可以在其中找到有关上述选项的更多信息，以及 Polling Consumer 页面的示例。

113.7. EXCHANGE PROPERTIES

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。

113.8. 示例

要设置一个路由，该路由每 60 秒生成事件：

```
from("scheduler://foo?delay=60000").to("bean:myBean?method=someMethodName");
```

以上路由将生成一个事件，然后在名为 myBean 的 bean 上调用 someMethodName 方法，如 JNDI 或 Spring。

和 Spring DSL 中的路由：

```
<route>
  <from uri="scheduler://foo?delay=60000"/>
  <to uri="bean:myBean?method=someMethodName"/>
</route>
```

113.9. 强制调度程序在完成后立即触发

要在上一任务完成后马上让调度程序触发，您可以设置 greedy=true 选项。但是，请注意，调度程序会持续触发所有时间。因此请谨慎使用它。

113.10. 强制调度程序闲置

有些用例中，您可能希望调度程序触发并仅显。但是，有时您希望"tell the scheduler"没有任务轮询，因此调度程序可以使用 backoff 选项切换到闲置模式。要做到这一点，您需要使用密钥 Exchange.SCHEDULER_POLLED_MESSAGES 在交换上设置属性 false。这将导致消费者表示没有轮询任何消息。

消费者将按其他方式返回轮询给调度程序的 1 消息，每次消费者完成交换处理时。

113.11. SPRING BOOT AUTO-CONFIGURATION

组件支持 4 个选项，如下所列。

Name	描述	默认值	类型

Name	描述	默认值	类型
camel.component.scheduler.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.scheduler.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.scheduler.enabled	是否启用调度程序组件的自动配置。这默认是启用的。		布尔值
camel.component.scheduler.pool-size	调度线程池中由调度线程池使用的核心线程数量。默认情况下使用单个线程。	1	整数

第 114 章 SEDA

支持生成者和消费者

SEDA 组件提供异步 SEDA 行为，以便在 BlockingQueue 和消费者上交换消息，并在与制作者独立的线程中调用消息。

请注意，队列只在 单个 CamelContext 中可见。如果要跨 CamelContext 实例通信（例如，在 Web 应用程序间进行通信），请查看组件。

如果虚拟机在处理消息时终止，则此组件不会实施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。

注意

同步

Direct 组件在生成者发送消息交换时提供任何消费者的同步调用。

114.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 seda 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-seda-starter</artifactId>
</dependency>
```

114.2. 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。

114.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

114.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

114.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

114.4. 组件选项

SEDA 组件支持 10 个选项，如下所列。

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)	设置默认并发线程处理交换数。	1	int
defaultPollTimeout (consumer (advanced))	轮询时使用的超时（以毫秒为单位）。发生超时时，消费者可以检查是否允许继续运行。设置较低值可让消费者在关闭时更快地响应。	1000	int
defaultBlockWhenFull (producer)	将消息发送到完整的 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，抛出异常表示队列已满。通过启用此选项，调用线程将阻止并等待消息被接受。	false	布尔值
defaultDiscardWhenFull (producer)	是否将消息发送到完整的 SEDA 队列的线程将被丢弃。默认情况下，抛出异常表示队列已满。通过启用此选项，调用线程将放弃发送并继续，这意味着消息没有发送到 SEDA 队列。	false	布尔值
defaultOfferTimeout (producer)	将消息发送到完整的 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，抛出异常表示队列已满。通过启用这个选项，可将配置的超时添加到块问题单中。利用布线 java 队列的 .offer (timeout)方法。		long
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
defaultQueueFactory (advanced)	设置默认队列工厂。		BlockingQueueFactory

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

114.5. 端点选项

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：

114.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
name (common)	**必需** 队列的名称。		字符串

114.5.2. 查询参数(18 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
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
limitConcurrentConsumers (consumer (advanced))	是否将 concurrentConsumers 的数量限制为最多 500 个。默认情况下，如果端点配置了更多数字，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 (advanced))	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 消息传递。也就是说，您可以向 SEDA 队列发送消息，并使每个消费者收到消息的副本。启用后，应在每个消费者端点上指定这个选项。	false	布尔值
pollTimeout (consumer (advanced))	轮询时使用的超时（以毫秒为单位）。发生超时时，消费者可以检查是否允许继续运行。设置较低值可让消费者在关闭时更快地响应。	1000	int
purgeWhenStopping (consumer (advanced))	在停止 consumer/route 时是否清除任务队列。这样可以更快地停止，因为队列中的任何待处理消息都会被丢弃。	false	布尔值
blockWhenFull (producer)	将消息发送到完整的 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，抛出异常表示队列已满。通过启用此选项，调用线程将阻止并等待消息被接受。	false	布尔值
discardIfNoConsumers (producer)	生产者是否应丢弃消息（不要将消息添加到队列中），当发送到没有活动消费者的队列时。只能同时启用其中一个选项 discardIfNoConsumers 和 failIfNoConsumers。	false	布尔值
discardWhenFull (producer)	是否将消息发送到完整的 SEDA 队列的线程将被丢弃。默认情况下，抛出异常表示队列已满。通过启用此选项，调用线程将放弃发送并继续，这意味着消息没有发送到 SEDA 队列。	false	布尔值
failIfNoConsumers (producer)	当发送到没有活跃消费者的队列时，生成者是否应该通过抛出异常失败。只能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
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
offerTimeout (producer)	当队列满时，可以将提供超时（以毫秒为单位）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 生成者将停止等待异步任务完成前的超时（毫秒）。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 任务完成，然后再继续。支持以下三个选项： Always, Never 或 IfReplyExpected。前两个值是 self-explanatory。最后的值(ifReplyExpected)将仅在消息是 Request Reply based 时等待。默认选项为 IfReplyExpected。 Enum 值： • Never • IfReplyExpected • Always	IfReplyExpected	WaitForTaskToComplete
queue (advanced)	定义端点将使用的队列实例。		BlockingQueue

114.6. 选择 BLOCKINGQUEUE 实现

默认情况下，SEDA 组件总是 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 个实现提供了 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>
```

114.7. 使用 REQUEST REPLY

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。请求被路由到 seda:input 队列。因为它是一个 Request Reply 消息，我们等待响应。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。

114.8. 并发消费者

默认情况下，SEDA 端点使用单个消费者线程，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("seda:stageName?concurrentConsumers=5").process(...)
```

对于两者之间的差别，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会被固定。

114.9. 线程池

请注意，通过执行以下操作将线程池添加到 SEDA 端点：

```
from("seda:stageName").thread(5).process(...)
```

可以通过两个 BlockQueues 获胜：来自 SEDA 端点，另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 Direct 端点，该端点可以同步和异步处理消息。例如：

```
from("direct:stageName").thread(5).process(...)
```

您还可以使用 concurrentConsumers 选项直接配置在 SEDA 端点上处理消息的线程数量。

114.10. 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，以便能够发送一个 fire-andforget 消息，以便在另一个线程中进一步处理，并将此线程中的恒定回复返回到原始调用者。

我们发送 Hello World 消息，并期望回复正常。

```
@Test
public void testSendAsync() throws Exception {
    MockEndpoint mock = getMockEndpoint("mock:result");
    mock.expectedBodiesReceived("Hello World");

    // START SNIPPET: e2
    Object out = template.requestBody("direct:start", "Hello World");
    assertEquals("OK", out);
    // END SNIPPET: e2

    assertMockEndpointsSatisfied();
}

@Override
protected RouteBuilder createRouteBuilder() throws Exception {
    return new RouteBuilder() {
        // START SNIPPET: e1
        public void configure() throws Exception {
            from("direct:start")
                // send it to the seda queue that is async
                .to("seda:next")
                // return a constant response
                .transform(constant("OK"));

            from("seda:next").to("mock:result");
        }
        // END SNIPPET: e1
    };
}
```

"Hello World"消息将从另一个线程的 SEDA 队列中使用，以便进一步处理。由于这是单元测试，因此它将被发送到 模拟 端点，我们在单元测试中执行断言。

114.11. 使用 MULTIPLECONSUMERS

在本例中，我们定义了两个消费者。

```
@Test
public void testSameOptionsProducerStillOkay() throws Exception {
    getMockEndpoint("mock:foo").expectedBodiesReceived("Hello World");
    getMockEndpoint("mock:bar").expectedBodiesReceived("Hello World");

    template.sendBody("seda:foo", "Hello World");

    assertMockEndpointsSatisfied();
}

@Override
protected RouteBuilder createRouteBuilder() throws Exception {
    return new RouteBuilder() {
        @Override
        public void configure() throws Exception {
            from("seda:foo?multipleConsumers=true").routeId("foo").to("mock:foo");
            from("seda:foo?multipleConsumers=true").routeId("bar").to("mock:bar");
        }
    };
}
```

由于我们在 seda foo 端点上指定了 multipleConsumers=true，因此这两个消费者可以拥有自己的消息副本作为 pub-sub 风格的消息传递。

由于 Bean 是单元测试的一部分，它们只是发送消息到模拟端点。

114.12. 提取队列信息

如果需要，可以在不使用 JMX 的情况下获取队列大小等信息：

```
SedaEndpoint seda = context.getEndpoint("seda:xxxx");
int size = seda.getExchanges().size();
```

114.13. 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.seda.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.seda.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.seda.concurrent-consumers	设置默认并发线程处理交换数。	1	整数
camel.component.seda.default-block-when-full	将消息发送到完整的 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，抛出异常表示队列已满。通过启用此选项，调用线程将阻止并等待消息被接受。	false	布尔值
camel.component.seda.default-discard-when-full	是否将消息发送到完整的 SEDA 队列的线程将被丢弃。默认情况下，抛出异常表示队列已满。通过启用此选项，调用线程将放弃发送并继续，这意味着消息没有发送到 SEDA 队列。	false	布尔值
camel.component.seda.default-offer-timeout	将消息发送到完整的 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，抛出异常表示队列已满。通过启用这个选项，可将配置的超时添加到块问题单中。利用布线 java 队列的 .offer (timeout)方法。		Long
camel.component.seda.default-poll-timeout	轮询时使用的超时（以毫秒为单位）。发生超时时，消费者可以检查是否允许继续运行。设置较低值可让消费者在关闭时更快地响应。	1000	整数
camel.component.seda.default-queue-factory	设置默认队列工厂。选项是一个 org.apache.camel.component.seda.BlockingQueueFactory<org.apache.camel.Exchange> 类型。		BlockingQueueFactory
camel.component.seda.enabled	是否启用 seda 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.seda.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.seda.queue-size	设置 SEDA 队列的默认最大容量（例如，它可以保存的消息数）。	1000	整数

第 115 章 SERVLET

仅支持消费者

Servlet 组件为消耗 HTTP 请求提供基于 HTTP 的端点，该端点到达绑定到公布的 Servlet 的 HTTP 端点。

注意

Stream

Servlet 基于流，这意味着它收到的输入作为流提交到 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，消息正文似乎为空，或者您需要多次访问数据（例如：执行多播或重新传送错误处理），您应该使用流缓存，或将消息正文转换为一个可安全地读取多次的字符串。

115.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 servlet 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-servlet-starter</artifactId>
</dependency>
```

115.2. URI 格式

```
servlet://relative_path[?options]
```

115.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别

- 端点级别

115.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

115.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

115.4. 组件选项

Servlet 组件支持 11 个选项，如下所列。

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消息时发生任何异常，现在将被作为消息进行处理，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
muteException (consumer)	如果对消费者启用并且交换失败，响应的正文不会包含异常的堆栈跟踪。	false	布尔值
servletName (consumer)	要使用的默认 servlet 名称。默认名称为 CamelServlet。	Camel Servlet	字符串
attachmentMultipartBinding (consumer (advanced))	是否自动将多部分/格式数据绑定为 Camel Exchange 上的附件。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，这会被关闭，因为在使用 Servlets 时，可能需要 servlet 特定的配置才能启用此功能。	false	布尔值
fileNameExtWhitelist (consumer (advanced))	接受已上传文件的已接受文件名扩展名的白名单。多个扩展可以用逗号分开，如 txt,xml。		字符串
httpRegistry (consumer (advanced))	使用自定义 org.apache.camel.component.servlet.HttpRegistry。		HttpRegistry
allowJavaSerializedObject (advanced)	当请求使用 context-type=application/x-java-serialized-object 时，是否允许 java 序列化。默认情况下是关闭的。如果启用此选项，则 Java 会将传入数据从请求反序列化到 Java，这可能会成为潜在的安全风险。	false	布尔值
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
httpBinding (advanced)	使用自定义 HttpBinding 来控制 Camel 消息和 HttpClient 之间的映射。		HttpBinding
httpConfiguration (advanced)	将共享的 HttpConfiguration 用作基础配置。		HttpConfiguration

Name	描述	默认值	类型
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头到 Camel 消息。		HeaderFilterStrategy

115.5. 端点选项

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：

115.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
contextPath (consumer)	**必需** 要使用的 context-path。		字符串

115.5.2. 查询参数(22 参数)

Name	描述	默认值	类型
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
disableStreamCache (common)	确定来自 Servlet 的原始输入流是否被缓存(Camel 会将流读取到内存/流到文件、流缓存)缓存。默认情况下，Camel 将缓存 Servlet 输入流来支持多次读取它，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，您可以将这个选项设置为 true，比如将其直接流传输到文件或其他持久性存储。DefaultHttpBinding 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，以支持多次读取流。如果您使用 Servlet 桥接/代理，则考虑启用这个选项以提高性能，以防您不需要多次读取消息有效负载。http producer 默认将缓存响应正文流。如果此选项设为 true，则制作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 过滤标头到 Camel 消息。		HeaderFilterStrategy

Name	描述	默认值	类型
httpBinding (common (advanced))	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 模式中工作。	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消息时发生任何异常，现在将被作为消息进行处理，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
httpMethodRestrict (consumer)	仅用于允许 HttpMethod 匹配，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
muteException (consumer)	如果对消费者启用并且交换失败，响应的正文不会包含异常的堆栈跟踪。	false	布尔值
responseBufferSize (consumer)	要在 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，且 Exchange 失败，如果作为 application/x-java-serialized-object 内容类型发送了导致的 Exception 被序列化。在生成者一侧，异常将被反序列化和抛出，而不是 HttpOperationFailedException。导致异常需要被序列化。默认情况下是关闭的。如果启用此选项，则 Java 会将传入数据从请求反序列化到 Java，这可能会成为潜在的安全风险。	false	布尔值
attachmentMultipartBinding (consumer (advanced))	是否自动将多部分/格式数据绑定为 Camel Exchange 上的附件。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，这会被关闭，因为在使用 Servlets 时，可能需要 servlet 特定的配置才能启用此功能。	false	布尔值

Name	描述	默认值	类型
eagerCheckContentAvailable (consumer (advanced))	是否要求检查 HTTP 请求是否有 content-length 标头是否为 0。如果 HTTP 客户端没有发送流数据，则可以打开它。	false	布尔值
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
fileNameExtWhitelist (consumer (advanced))	接受已上传文件的已接受文件名扩展名的白名单。多个扩展可以用逗号分开，如 txt,xml。		字符串
mapHttpMessageBody (consumer (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 将避免 HTTP 映射。	true	布尔值
mapHttpMessageFormUrlEncodedBody (consumer (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 将避免 HTTP Form Encoded body 映射。	true	布尔值
mapHttpMessageHeaders (consumer (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
optionsEnabled (consumer (advanced))	指定是否为此 Servlet 使用者启用 HTTP OPTIONS。默认情况下关闭 OPTIONS。	false	布尔值
traceEnabled (consumer (advanced))	指定是否为此 Servlet 使用者启用 HTTP TRACE。默认情况下关闭 TRACE。	false	布尔值

115.6. 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：如果客户端请求有 URL http://myserver/myserver?orderid=123，则交换将包含名为 orderid 的标头，值为 123。

115.7. 使用方法

您只能从 Servlet 组件生成的端点中使用。因此，它应该只用作 Camel 路由的输入。要针对其他 HTTP 端点发出 HTTP 请求，请使用 HTTP 组件。

115.8. 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.servlet.allow-java-serialized-object	当请求使用 context-type=application/x-java-serialized-object 时，是否允许 java 序列化。默认情况下是关闭的。如果启用此选项，则 Java 会将传入数据从请求反序列化到 Java，这可能会成为潜在的安全风险。	false	布尔值
camel.component.servlet.attachment-multipart-binding	是否自动将多部分/格式数据绑定为 Camel Exchange 上的附件。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，这会被关闭，因为在使用 Servlet 时，可能需要 servlet 特定的配置才能启用此功能。	false	布尔值
camel.component.servlet.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.servlet.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消息时发生任何异常，现在将被作为消息进行处理，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
camel.component.servlet.enabled	是否启用 servlet 组件的自动配置。这默认是启用的。		布尔值
camel.component.servlet.file-name-ext-whitelist	接受已上传文件的已接受文件名扩展名的白名单。多个扩展可以用逗号分开，如 txt,xml。		字符串
camel.component.servlet.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头到 Camel 消息。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		HeaderFilterStrategy
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。		HttpBinding
camel.component.servlet.http-configuration	将共享的 HttpConfiguration 用作基础配置。选项是 org.apache.camel.http.common.HttpConfiguration 类型。		HttpConfiguration
camel.component.servlet.http-registry	使用自定义 org.apache.camel.component.servlet.HttpRegistry。选项是 org.apache.camel.http.common.HttpRegistry 类型。		HttpRegistry
camel.component.servlet.mute-exception	如果对消费者启用并且交换失败，响应的正文不会包含异常的堆栈跟踪。	false	布尔值
camel.component.servlet.servlet-name	要使用的默认 servlet 名称。默认名称为 CamelServlet。	Camel Servlet	字符串
camel.servlet.mapping.context-path	servlet 组件用于自动映射的上下文路径。	/camel/*	字符串
camel.servlet.mapping.enabled	启用 servlet 组件的自动映射到 Spring web 上下文。	true	布尔值
camel.servlet.mapping.servlet-name	Camel servlet 的名称。	Camel Servlet	字符串

第 116 章 SIMPLE（简单）

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。它主要的设计目的是，使用一个非常小且简单的语言，用于评估 Expression 或 Predicate，而无需了解任何对其他脚本语言（如 Groovy）。

这个简单的语言被设计为在 Camel 路由中无需大量开发脚本而满足所有常见的用例。

但是，对于更复杂的用例，建议使用更强大的语言，例如：

- Groovy
- MVEL
- OGNL

注意

如果简单语言使用 OGNL 表达式，则简单语言需要 camel-bean JAR 作为类路径依赖项，例如在消息正文中调用名为 myMethod 的方法：${body.myMethod （）}。在运行时，简单语言将我们其内置的 OGNL 支持（需要 camel-bean 组件）。

简单语言使用 ${body} 占位符进行复杂的表达式或函数。

注意

另请参阅编译的 CSimple 语言。

注意

其他语法
您还可以使用替代语法，其使用 $simple{ } 作为占位符。当将 Spring 属性占位符与 Camel 一起使用时，这可用于避免冲突。

116.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 简单 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-core-starter</artifactId>
</dependency>
```

116.2. 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		布尔值	是否修剪值以移除前导和结尾的空格和换行符。

116.3. 变量

变量	类型	描述
camelId	字符串	CamelContext 名称
camelContext.OGNL	对象	使用 Camel OGNL 表达式调用的 CamelContext。
交换	Exchange	Exchange
Exchange.OGNL	对象	Exchange 使用 Camel OGNL 表达式调用。
exchangeId	字符串	Exchange id
id	字符串	消息 ID

变量	类型	描述
messageTimestamp	字符串	此消息的来源的消息时间戳(millis since epoc)。一些系统，如 JMS、Kafka、AWS 在 Camel 收到的事件/消息上有一个时间戳。如果存在时间戳，这个方法会返回时间戳。创建的消息时间戳和交换不同。交换始终具有创建的时间戳，这是 Camel 创建交换时的本地时间戳。只有当消费者能够从源事件中提取时间戳时，一些 Camel 组件中才有消息时间戳。如果消息没有时间戳，则返回 0。
正文（body）	对象	正文(body)
body.**OGNL**	对象	使用 Camel OGNL 表达式调用的正文。
bodyAs(*type*)	类型	将正文转换为由其 classname 确定的给定类型。转换的正文可以是 null。
bodyAs(*type*).**OGNL**	对象	将正文转换为由其 classname 确定的给定类型，然后使用 Camel OGNL 表达式调用方法。转换的正文可以是 null。
bodyOneLine	字符串	将正文转换为 String，并删除所有 line-breaks，以便字符串在一行中。
mandatoryBodyAs(*type*)	类型	将正文转换为由其 classname 确定的给定类型，并期望正文不是 null。
mandatoryBodyAs(*type*).**OGNL**	对象	将正文转换为由其 classname 确定的给定类型，然后使用 Camel OGNL 表达式调用方法。
header.foo	对象	请参阅 foo 标头
header[foo]	对象	请参阅 foo 标头
headers.foo	对象	请参阅 foo 标头
headers:foo	对象	请参阅 foo 标头
headers[foo]	对象	请参阅 foo 标头
header.foo[bar]	对象	关于 foo 标头作为映射，并在映射上以 bar 为键执行查找
header.foo.**OGNL**	对象	引用 foo 标头并使用 Camel OGNL 表达式调用其值。
headerAs(*key*,*type*)	类型	将标头转换为由其 classname 确定的给定类型
标头	Map	请参阅标头
exchangeProperty.foo	对象	请参阅交换上的 foo 属性

变量	类型	描述
exchangeProperty[foo]	对象	请参阅交换上的 foo 属性
exchangeProperty.foo.**OGNL**	对象	请参考交换上的 foo 属性，并使用 Camel OGNL 表达式调用其值。
sys.foo	字符串	请参阅 JVM 系统属性
sysenv.foo	字符串	请参阅系统环境变量
env.foo	字符串	请参阅系统环境变量
例外	对象	如果没有在交换上设置异常，请参阅交换上的异常对象为 **null**。如果 Exchange 有任何，将会回退和获取异常（**Exchange.EXCEPTION_CAUGHT**）。
例外。**OGNL**	对象	请参考使用 Camel OGNL 表达式对象调用的交换异常
exception.message	字符串	如果没有在交换上设置异常，请参阅 Exchange 上的 exception.message。**如果 Exchange 有任何，将会回退和获取异常（Exchange.EXCEPTION_CAUGHT）。**
exception.stacktrace	字符串	如果没有在交换上设置异常，请参阅交换上的 exception.stracktrace。**如果 Exchange 有任何，将会回退和获取异常（Exchange.EXCEPTION_CAUGHT）。**
date:_command_	Date	评估为 Date 对象。支持的命令是：当前的时间戳，在创建当前交换 时为时间戳创建交换，**header.xxx** 使用带有密钥 xxx 的 Long/Date 对象标头。**exchangeProperty.xxx，使用带有键 xxx 的 exchange 属性中的 Long/Date 对象。文件 的最后一个修改时间戳的文件（可通过文件消费者使用）。命令接受偏移，如：now-24h 或 header.xxx+1h，甚至 现在+1h30m-100。**
date:_command:pattern_	字符串	使用 **java.text.SimpleDateFormat** 模式进行日期格式化。
date-with-timezone:_command:timezone:pattern_	字符串	使用 **java.text.SimpleDateFormat** 时区和模式进行日期格式化。
bean:_bean expression_	对象	使用语言调用 bean 表达式.指定方法名称，必须使用点作为分隔符。我们还支持组件使用的 ?method=methodname 语法。默认情况下，Camel 将按指定名称查找 bean。但是，如果您需要引用 bean 类（如调用静态方法），那么您可以使用一个类型作为前缀，如 **bean:type:fqnClassName**。

变量	类型	描述
properties:key:default	字符串	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	返回 Exchange 正在路由的当前路由的 id。
stepId	字符串	返回 Exchange 正在路由的当前步骤的 id。
threadName	字符串	返回当前线程的名称。可用于日志目的。
hostname	字符串	返回本地主机名（如果无法解析，则为空）。
ref:xxx	对象	从具有给定 id 的 Registry 中查找 bean。
type:name.field	对象	根据 FQN 名称引用类型或字段。要引用某个字段，您可以附加 .FIELD_NAME。例如，您可以将 Exchange 中的 constant 字段引用为： **org.apache.camel.Exchange.FILE_NAME**
null	null	代表 **null**
random(value)	整数	返回一个随机的、在 0 (包括) 和 *value* (不包括) 之间的值
random(min,max)	整数	在 *min* (included)和 *max* (excluded)之间返回一个随机的整数
collate(group)	list	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。这可与 Splitter EIP 一起使用，将消息正文和 group/batch 分成一组 N 子列表。这个方法的工作方式与 Groovy 中的合作方法类似。
skip(number)	Ierator	skip 函数迭代消息正文并跳过第一个项目数。这可与 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	当前交换的消息历史记录如何路由。这类似于路由 stack-trace 消息历史记录，在出现未处理异常时的错误处理程序日志。
messageHistory(false)	字符串	作为 messageHistory，但没有交换详情（仅包含路由 stack-trace）。如果您不想从消息本身记录敏感数据，则可以使用此选项。

116.4. OGNL 表达式支持

使用 OGNL 时，需要 camel-bean JAR 才能在 classpath 上。

Camel 的 OGNL 支持仅用于调用方法。您不能访问字段。Camel 支持访问 Java 阵列的 length 字

段。

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，它有一个 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 getters 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，也可以使用 null 安全运算符(?.)来避免 NPE。

```
simple("${body?.address?.street}")
```

也可以用 Map 或 List 类型来索引，因此您可以：

```
simple("${body[foo].name}")
```

假设正文基于 Map，并使用 foo 作为键查找值，并调用该值上的 getName 方法。

如果键有空格，则必须 使用引号括起键，如 'foo bar'：

```
simple("${body['foo bar'].name}")
```

您可以使用其密钥名称（带有或没有点）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，那么您可以使用 null 安全运算符来避免 NPE，如下所示：

```
simple("${body[foo]?.name}")
```

您还可以访问 List 类型，例如从您可以进行的地址获取行：

```
simple("${body.address.lines[0]}")
simple("${body.address.lines[1]}")
simple("${body.address.lines[2]}")
```

有一个特殊的 最后一个 关键字，可用于从列表中获取最后一个值。

```
simple("${body.address.lines[last]}")
```

要获得最后 2 个信息，您可以减去一个数字，因此我们可以使用 last-1 来指示以下内容：

```
simple("${body.address.lines[last-1]}")
```

最后是课程的第 3 个问题：

```
simple("${body.address.lines[last-2]}")
```

您可以使用以下内容调用列表的大小方法

```
simple("${body.address.lines.size}")
```

Camel 也支持 Java 阵列的 length 字段，例如：

```
String[] lines = new String[]{"foo", "bar", "cat"};
exchange.getIn().setBody(lines);

simple("There are ${body.length} lines")
```

和 yes，您可以将此操作与 Operator 支持相结合，如下所示：

```
simple("${body.address.zip} > 1000")
```

116.5. OPERATOR 支持

解析器仅限于只支持单个 Operator。

要启用它，必须将左侧值包括在 $\{ } 中。语法为：

```
${leftValue} OP rightValue
```

其中 rightValue 可以是以 ' '、null、一个恒定值或其他包括在 ${ } 中的其他表达式的字符串。

注意

运算符必须有空格。

Camel 将自动将 rightValue 类型转换为 leftValue 类型，例如它可以将一个字符串转换为一个数字，因此可以使用 > 来比较数字值。

支持以下 Operator：

Operator	描述
==	等于
=~	等于忽略大小（在比较 String 值时忽略大小写）
>	大于
>=	大于 or equals
<	小于
<=	小于 or equals
!=	not equals
!=~	不等于忽略大小（在比较 String 值时忽略大小写）
contains	如果包含基于字符串的值，则用于测试

Operator	描述
!contains	如果基于字符串的值中没有包含测试
~~	在基于字符串的值中忽略大小敏感度，则用于测试
!~~	在基于字符串的值中忽略问题单敏感度，则用于测试
regex	对于与给定的正则表达式模式匹配，定义为 String 值
!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，每个元素必须用逗号分开。如果要包含空值，则必须使用双逗号（如 ',bronze,silver,gold' ）定义它，它是一组带有空值的四个值，然后是三个形形。
!in	若要匹配（如果没有在一组值中），则必须用逗号分隔每个元素。如果要包含空值，则必须使用双逗号（如 ',bronze,silver,gold' ）定义它，它是一组带有空值的四个值，然后是三个形形。
is	如果左侧 type 是值的实例，则进行匹配。
!is	如果左侧类型不是值的实例，则进行匹配。
range	如果左手位于定义为数字(**从.. 到 ..)的值范围内，** 则匹配。
!range	如果左手不在定义为数字(**从..到 .)的值范围之内，** 则匹配。
startsWith	测试左侧字符串是否以右手字符串开头。
从开始	与 startsWith 运算符相同。
endwith	用于测试左侧字符串是否以右手字符串结尾。
结束	与 endWith 运算符相同。

也可以使用以下元运算符：

Operator	描述
++	要按数字递增，请执行以下操作：左侧必须是函数，否则解析为 literal。
–	要减少一个数字，请执行以下操作：左侧必须是函数，否则解析为 literal。
\n	使用换行符。

Operator	描述
\t	使用制表符。
\r	使用 carriage 返回字符。
\}	要将 } 字符用作文本。使用简单语言构建 JSon 结构时需要这样做。

以下逻辑运算符可用于对表达式进行分组：

Operator	描述
&&	logical 和 运算符用于对两个表达式进行分组。
	logical or 运算符用于对两个表达式进行分组。

AND 的语法是：

```
${leftValue} OP rightValue && ${leftValue} OP rightValue
```

OR 的语法是：

```
${leftValue} OP rightValue || ${leftValue} OP rightValue
```

一些示例：

```
// exact equals match
simple("${header.foo} == 'foo'")

// ignore case when comparing, so if the header has value FOO this will match
simple("${header.foo} =~ 'foo'")

// here Camel will type convert '100' into the type of header.bar and if it is an Integer '100' will
also be converter to an Integer
simple("${header.bar} == '100'")

simple("${header.bar} == 100")

// 100 will be converter to the type of header.bar so we can do > comparison
simple("${header.bar} > 100")
```

-

116.5.1. 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 1 个优先级。如果这两个值都无法根据该类型进行比较，并回退到右侧类型。
这意味着您可以利用值来强制执行特定类型的类型。假设上面的 bar 值是一个 String。然后您可以重新利用它：

```
simple("100 < ${header.bar}")
```

然后，确保 int 类型被用作 1st 优先级。

如果 Camel 团队将二进制比较操作改进，以基于 String 的首选数字类型，则这可能会改变。最常见的 String 类型会导致与数字比较时出现问题。

```
// testing for null
simple("${header.baz} == null")

// testing for not null
simple("${header.baz} != null")
```

另一个更高级的示例，其中右值是另一个表达式

```
simple("${header.date} == ${date:now:yyyyMMdd}")

simple("${header.type} == ${bean:orderService?method=getOrderType}")
```

另一个包含的示例，测试标题是否包含 Camel 一词

```
simple("${header.title} contains 'Camel'")
```

另外，带有 regex 的示例，测试数字标头是否为 4 位值：

```
simple("${header.number} regex '\\d{4}'")
```

最后是一个示例，如果标头等于列表中的任何值。每个元素必须用逗号分开，且不能空格分开。
这也适用于数字，因为 Camel 会将每个元素转换为左侧的类型。

```
simple("${header.type} in 'gold,silver'")
```

对于最后的 3，我们也支持使用 negate 测试：

```
simple("${header.type} !in 'gold,silver'")
```

您可以测试类型是否是一个特定实例，例如字符串

```
simple("${header.type} is 'java.lang.String'")
```

我们为所有 java.lang 类型添加了一个简写，以便您可以将其写入：

```
simple("${header.type} is 'String'")
```

也支持范围。范围间隔需要数字，以及 from 和 end 都包括。例如，要测试值介于 100 到 199 之间：

```
simple("${header.number} range 100..199")
```

请注意，我们在没有空格的范围内使用 ..。它基于与 Groovy 相同的语法。

```
simple("${header.number} range '100..199'")
```

因为 XML DSL 没有所有具有其各种构建器方法的 Java DSL 电源，所以您必须利用其他语言通过简单运算符进行测试。现在，您可以使用简单的语言进行此操作。在以下示例中，我们需要测试标头是否为小部件顺序：

```
<from uri="seda:orders">
  <filter>
    <simple>${header.type} == 'widget'</simple>
    <to uri="bean:orderService?method=handleWidget"/>
  </filter>
</from>
```

116.5.2. 使用 和 / 或

如果您有两个表达式，您可以将它们与 & amp;& 或 || 运算符合并。

例如：

```
simple("${header.title} contains 'Camel' && ${header.type'} == 'gold'")
```

此外，还支持 ||。这个示例为：

```
simple("${header.title} contains 'Camel' || ${header.type'} == 'gold'")
```

116.6. 例子

在下面的 XML DSL 示例中，我们根据标头值过滤：

```
<from uri="seda:orders">
  <filter>
    <simple>${header.foo}</simple>
    <to uri="mock:fooOrders"/>
  </filter>
</from>
```

Simple 语言可用于消息过滤器模式的 predicate 测试，其中我们测试消息是否有 foo 标头（键为 foo 的标头存在）。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("${header.foo}")
    .to("seda:fooOrders");
```

您还可以将简单的语言用于简单的文本串联，例如：

```
from("direct:hello")
  .transform().simple("Hello ${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。

此示例使用 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，以调用要包含在返回字符串中的 bean 的方法：

```
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。

如果要声明在 ID 生成器 Bean 上调用哪些方法，我们必须前置 .method 名称，如我们调用 generateId 方法的位置。

```
from("direct:order")
    .transform().simple("OrderId: ${bean:orderIdGenerator.generateId}")
    .to("mock:reply");
```

我们可以使用我们熟悉 Bean 组件本身的 ?method=methodname 选项：

```
from("direct:order")
    .transform().simple("OrderId: ${bean:orderIdGenerator?method=generateId}")
    .to("mock:reply");
```

您还可以将正文转换为给定类型，例如，确保它是一个字符串，您可以：

```
<transform>
  <simple>Hello ${bodyAs(String)} how are you?</simple>
</transform>
```

有几个类型具有简写表示法，因此我们可以使用 String 而不是 java.lang.String。这些是：byte[]、String、Integer、Long。所有其他类型都必须使用其 FQN 名称，例如 org.w3c.dom.Document。

也可以从标头映射中查找值：

```
<transform>
  <simple>The gold value is ${header.type[gold]}</simple>
</transform>
```

在上面的代码中，我们查找名称 type 的标头并将其视为 java.util.Map，然后以键金级进行查找并返回值。如果标头不可转换为 Map 异常，则会抛出异常。如果名为 type 的标头不存在，则返回 null。

您可以嵌套功能，如下所示：

```
<setHeader name="myHeader">
  <simple>${properties:${header.someKey}}</simple>
</setHeader>
```

116.7. 设置结果类型

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。这可用于定义布尔值、整数等类型。

例如，要将标头设置为布尔值类型，您可以：

```
.setHeader("cool", simple("true", Boolean.class))
```

在 XML DSL 中

```
<setHeader name="cool">
  <!-- use resultType to indicate that the type should be a java.lang.Boolean -->
  <simple resultType="java.lang.Boolean">true</simple>
</setHeader>
```

116.8. 在 XML DSL 中使用新行或标签页

在 XML DSL 中指定新行或标签页变得更加容易，因为您可以立即转义值

```
<transform>
  <simple>The following text\nis on a new line</simple>
</transform>
```

116.9. 前导和结尾的空格处理

表达式的 trim 属性可用于控制是否删除或保留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>
```

116.10. 从外部资源载入脚本

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" 等引用您可以进行的类路径中的文件：

```
.setHeader("myHeader").simple("resource:classpath:mysimple.txt")
```

116.11. SPRING BOOT AUTO-CONFIGURATION

组件支持 147 选项，如下所列。

Name	描述	默认值	类型
camel.cloud.consul.service-discovery.acl-token	设置用于 Consul 的 ACL 令牌。		字符串
camel.cloud.consul.service-discovery.block-seconds	等待监视事件的秒数，默认为 10 秒。	10	整数
camel.cloud.consul.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.consul.service-discovery.connect-timeout-millis	OkHttpClient 的连接超时。		Long
camel.cloud.consul.service-discovery.datacenter	数据中心。		字符串
camel.cloud.consul.service-discovery.enabled	启用组件。	true	布尔值

Name	描述	默认值	类型
camel.cloud.consul.service-discovery.password	设置用于基本身份验证的密码。		字符串
camel.cloud.consul.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.consul.service-discovery.read-timeout-millis	OkHttpClient 的读取超时。		Long
camel.cloud.consul.service-discovery.url	Consul 代理 URL。		字符串
camel.cloud.consul.service-discovery.user-name	设置用于基本身份验证的用户名。		字符串
camel.cloud.consul.service-discovery.write-timeout-millis	OkHttpClient 的写入超时。		Long
camel.cloud.dns.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.dns.service-discovery.domain	域名；		字符串
camel.cloud.dns.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.dns.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map

Name	描述	默认值	类型
camel.cloud.dns.service-discovery.proto	所需服务的传输协议。	_tcp	字符串
camel.cloud.etcd.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.etcd.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.etcd.service-discovery.password	用于基本身份验证的密码。		字符串
camel.cloud.etcd.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.etcd.service-discovery.service-path	查找服务发现的路径。	/services/	字符串
camel.cloud.etcd.service-discovery.timeout	要设置操作可以采取的最长时间，请执行以下操作：		Long
camel.cloud.etcd.service-discovery.type	要设置发现类型，有效值为 on-demand 和 watch。	按需	字符串
camel.cloud.etcd.service-discovery.uris	客户端可以连接到的 URI。		字符串
camel.cloud.etcd.service-discovery.username	用于基本身份验证的用户名。		字符串

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.api-version	使用客户端查找时设置 API 版本。		字符串
camel.cloud.kubernetes.service-discovery.ca-cert-data	使用客户端查找时设置证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.ca-cert-file	在使用客户端查找时，设置从文件加载的证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-data	使用客户端查找时设置客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-file	在使用客户端查找时，设置从文件加载的客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-algo	设置客户端密钥存储算法，如使用客户端查找时 RSA。		字符串
camel.cloud.kubernetes.service-discovery.client-key-data	使用客户端查找时设置客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-file	在使用客户端查找时，设置从文件加载的客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-passphrase	使用客户端查找时设置客户端密钥存储密码短语。		字符串
camel.cloud.kubernetes.service-discovery.configurations	定义其他配置定义。		Map

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.dns-domain	设置用于 DNS 查找的 DNS 域。		字符串
camel.cloud.kubernetes.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.kubernetes.service-discovery.lookup	如何执行服务查找。可能的值有：client、dns、environment。在使用客户端时，客户端会查询 kubernetes master 来获取提供该服务的活跃 pod 列表，然后随机（或循环）选择一个 pod。当使用 dns 时，服务名称被解析为 name.namespace.svc.dnsDomain。当使用 dnssrv 时，服务名称使用 SRV 查询解析 ....svc... when using environment，环境变量用于查找服务。默认情况下使用环境。	环境	字符串
camel.cloud.kubernetes.service-discovery.master-url	在使用客户端查找时，将 URL 设置为 master。		字符串
camel.cloud.kubernetes.service-discovery.namespace	设置要使用的命名空间。默认情况下，将使用来自 ENV 变量 KUBERNETES_MASTER 的命名空间。		字符串
camel.cloud.kubernetes.service-discovery.oauth-token	在使用客户端查找时，为身份验证设置 OAUTH 令牌（而不是用户名/密码）。		字符串
camel.cloud.kubernetes.service-discovery.password	在使用客户端查找时设置用于身份验证的密码。		字符串
camel.cloud.kubernetes.service-discovery.port-name	设置用于 DNS/DNSSRV 查找的端口名称。		字符串
camel.cloud.kubernetes.service-discovery.port-protocol	设置用于 DNS/DNSSRV 查找的端口协议。		字符串

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.kubernetes.service-discovery.trust-certs	设置在使用客户端查找时是否打开信任证书检查。	false	布尔值
camel.cloud.kubernetes.service-discovery.username	在使用客户端查找时设置用于身份验证的用户名。		字符串
camel.cloud.ribbon.load-balancer.client-name	设置 Ribbon 客户端名称。		字符串
camel.cloud.ribbon.load-balancer.configurations	定义其他配置定义。		Map
camel.cloud.ribbon.load-balancer.enabled	启用组件。	true	布尔值
camel.cloud.ribbon.load-balancer.namespace	命名空间。		字符串
camel.cloud.ribbon.load-balancer.password	密码。		字符串
camel.cloud.ribbon.load-balancer.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map

Name	描述	默认值	类型
camel.cloud.ribbon.load-balancer.username	用户名。		字符串
camel.hystrix.allow-maximum-size-to-diverge-from-core-size	允许配置使 maximumSize 生效。然后该值可以等于或大于 coreSize。	false	布尔值
camel.hystrix.circuit-breaker-enabled	是否使用 HystrixCircuitBreaker。如果为 false，则不会使用 断路器逻辑，并且所有允许的请求。这与 circuitBreakerForceClosed （）的影响类似，除非继续跟踪指标，知道它是否应该是 open/closed，此属性即使实例化一个断路器。	true	布尔值
camel.hystrix.circuit-breaker-error-threshold-percentage	错误百分比阈值（如 50 ）指向断路器将打开和拒绝请求。它将在 circuitBreakerSleepWindowInMilliseconds 中定义的持续时间保持出差；与 HystrixCommandMetrics.getHealthCounts （）进行比较的错误百分比。	50	整数
camel.hystrix.circuit-breaker-force-closed	如果为 true，HystrixCircuitBreaker#allowRequest （）将始终返回 true 以允许请求，无论 HystrixCommandMetrics.getHealthCounts （）的错误百分比如何。如果设为 true，则 circuitBreakerForceOpen （）属性具有优先权。	false	布尔值
camel.hystrix.circuit-breaker-force-open	如果为 true，HystrixCircuitBreaker.allowRequest （）将始终返回 false，从而导致电路变为开路（接受），并拒绝所有请求。此属性优先于 circuitBreakerForceClosed （）;。	false	布尔值
camel.hystrix.circuit-breaker-request-volume-threshold	metricsRollingStatisticalWindowInMilliseconds （）中的最少请求数必须存在于 HystrixCircuitBreaker 之前。如果此数字低于这个数字，无论错误百分比如何，电路都不会被出差。	20	整数
camel.hystrix.circuit-breaker-sleep-window-in-milliseconds	HystrixCircuitBreaker trips 之后的时间（以毫秒为单位），它应该在尝试请求前等待。	5000	整数
camel.hystrix.configurations	定义其他配置定义。		Map

Name	描述	默认值	类型
camel.hystrix.core-pool-size	传递给 java.util.concurrent.ThreadPoolExecutor#setCorePoolSize (int)的核心 thread-pool 大小。	10	整数
camel.hystrix.enabled	启用组件。	true	布尔值
camel.hystrix.execution-isolation-semaphore-max-concurrent-requests	允许 HystrixCommand.run （）的并发请求数。超过并发限制的请求将被拒绝。仅在执行IsolationStrategy == SEMAPHORE 时使用。	20	整数
camel.hystrix.execution-isolation-strategy	将通过什么隔离策略 HystrixCommand.run （）执行。如果 THREAD，它将在单独的线程上执行，并且受 thread-pool 中的线程数量限制的并发请求。如果 SEMAPHORE，它将在调用线程上执行，并且受 semaphore 数限制的并发请求。	线程	字符串
camel.hystrix.execution-isolation-thread-interrupt-on-timeout	当线程超时时，执行线程是否应该尝试中断（使用 future#cancel）。仅在执行IsolationStrategy （）== THREAD 时才适用。	true	布尔值
camel.hystrix.execution-timeout-enabled	此命令是否启用了超时机制。	true	布尔值
camel.hystrix.execution-timeout-in-milliseconds	以毫秒为单位，将命令超时和停止执行的时间（以毫秒为单位）。如果 executionIsolationThreadInterruptOnTimeout == true 且命令是线程隔离，则执行线程将中断。如果命令是 semaphore-isolated 和 HystrixObservableCommand，则该命令将被取消订阅。	1000	整数
camel.hystrix.fallback-enabled	出现故障时，是否应尝试 HystrixCommand.getFallback （）。	true	布尔值
camel.hystrix.fallback-isolation-semaphore-max-concurrent-requests	允许 HystrixCommand.getFallback （）的并发请求数。超过并发限制的请求将快速失败，且不会尝试检索回退。	10	整数
camel.hystrix.group-key	设置要使用的 group 键。默认值为 CamelHystrix。	Camel Hystrix	字符串

Name	描述	默认值	类型
camel.hystrix.keep-alive-time	更长的时间（以分钟为单位）传递给 ThreadPoolExecutor#setKeepAliveTime (long,TimeUnit)。	1	整数
camel.hystrix.max-queue-size	在 HystrixConcurrencyStrategy.getBlockingQueue (int)中传递给 BlockingQueue 的最大队列大小应该只影响 threadpool 的实例化 - 它不会立即更改队列大小。为此，请使用 queueSizeRejectionThreshold （）。	-1	整数
camel.hystrix.maximum-size	传递给 ThreadPoolExecutor#setMaximumPoolSize (int)的最大 thread-pool 大小。这是可在不开始拒绝 HystrixCommands 的情况下支持的最大并发数量。请注意，只有在您也设置了 allowMaximumSizeToDivergeFromCoreSize 时，此设置才会生效。	10	整数
camel.hystrix.metrics-health-snapshot-interval-in-milliseconds	在允许计算成功和错误百分比时等待的时间（以毫秒为单位），并影响 HystrixCircuitBreaker.isOpen （）状态。在高容量电路上，错误百分比的连续计算可能会成为 CPU 密集型，从而控制其计算的频率。	500	整数
camel.hystrix.metrics-rolling-percentile-bucket-size	滚动百分比的每个存储桶中存储的最大值数。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	10	整数
camel.hystrix.metrics-rolling-percentile-enabled	是否应该使用 HystrixRollingPercentile 内部 HystrixCommandMetrics 来捕获百分比的指标。	true	布尔值
camel.hystrix.metrics-rolling-percentile-window-buckets	滚动窗口的存储桶数量被分成。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	6	整数
camel.hystrix.metrics-rolling-percentile-window-in-milliseconds	以毫秒为单位的滚动窗口的持续时间。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	10000	整数
camel.hystrix.metrics-rolling-statistical-window-buckets	滚动统计窗口划分为的 bucket 数量。这在 HystrixCommandMetrics 中被传递给 HystrixRollingNumber。	10	整数

Name	描述	默认值	类型
camel.hystrix.metrics-rolling-statistical-window-in-milliseconds	此属性设置统计滚动窗口的持续时间，以毫秒为单位。这是为线程池保留指标的时间。窗口被分成 bucket，按这些增量回滚。	10000	整数
camel.hystrix.queue-size-rejection-threshold	队列大小拒绝阈值是 artificial max size，即使尚未达到 maxQueueSize，也会发生拒绝。这是因为 BlockingQueue 的 maxQueueSize 无法动态更改，我们希望动态更改影响拒绝的队列大小。在排队线程以进行执行时，HystrixCommand 会使用它。	5	整数
camel.hystrix.request-log-enabled	HystrixCommand 执行和事件是否应记录到 HystrixRequestLog。	true	布尔值
camel.hystrix.thread-pool-key	设置要使用的线程池密钥。默认情况下，将使用与 groupKey 配置相同的值。	Camel Hystrix	字符串
camel.hystrix.thread-pool-rolling-number-statistical-window-buckets	滚动统计窗口划分为的 bucket 数量。这会传递到每个 HystrixThreadPoolMetrics 实例内的 HystrixRollingNumber。	10	整数
camel.hystrix.thread-pool-rolling-number-statistical-window-in-milliseconds	统计滚动窗口的持续时间（以毫秒为单位）。这会传递到每个 HystrixThreadPoolMetrics 实例内的 HystrixRollingNumber。	10000	整数
camel.language.constant.enabled	是否启用恒定语言的自动配置。这默认是启用的。		布尔值
camel.language.constant.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.csimple.enabled	是否启用 csimple 语言的自动配置。这默认是启用的。		布尔值
camel.language.csimple.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.exchangeproperty.enabled	是否启用 exchangeProperty 语言的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.language.exchangeproperty.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.file.enabled	是否启用文件语言的自动配置。这默认是启用的。		布尔值
camel.language.file.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.header.enabled	是否启用标头语言的自动配置。这默认是启用的。		布尔值
camel.language.header.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.ref.enabled	是否启用 ref 语言的自动配置。这默认是启用的。		布尔值
camel.language.ref.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.simple.enabled	是否启用简单语言的自动配置。这默认是启用的。		布尔值
camel.language.simple.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.tokenize.enabled	是否启用令牌化语言的自动配置。这默认是启用的。		布尔值
camel.language.tokenize.group-delimiter	设置在分组时要使用的分隔符。如果没有设置，则令牌将用作分隔符。		字符串
camel.language.tokenize.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.resilience4j.automatic-transition-from-open-to-half-open-enabled	在通过 waitDurationInOpenState 后，启用从 OPEN 自动过渡到 HALF_OPEN 状态。	false	布尔值

Name	描述	默认值	类型
camel.resilience4j.circuit-breaker-ref	代表现有的 io.github.resilience4j.circuitbreaker.CircuitBreaker 实例从 registry 中查找和使用。使用此选项时，不使用任何其他断路器选项。		字符串
camel.resilience4j.config-ref	指的是现有的 io.github.resilience4j.circuitbreaker.CircuitBreakerConfig 实例，以便从 registry 中查找和使用。		字符串
camel.resilience4j.configurations	定义其他配置定义。		Map
camel.resilience4j.enabled	启用组件。	true	布尔值
camel.resilience4j.failure-rate-threshold	以百分比为单位配置故障率阈值。如果失败率相等或大于阈值，则 CircuitBreaker 过渡到 open，并启动短路调用。阈值必须大于 0，而不是大于 100。默认值为 50 百分比。		æµ®ç‚¹å€¼
camel.resilience4j.minimum-number-of-calls	在 CircuitBreaker 可以计算错误率之前，配置所需的最少调用数（每个滑动期限）。例如，如果 minimumNumberOfCalls 为 10，则必须至少记录 10 个调用，然后才能计算失败率。如果只记录了 9 个调用，则 CircuitBreaker 不会过渡到 open，即使所有 9 调用都失败。默认 minimumNumberOfCalls 为 100。	100	整数
camel.resilience4j.permitted-number-of-calls-in-half-open-state	配置 CircuitBreaker 为一半打开时允许的调用数量。大小必须大于 0。默认大小为 10。	10	整数
camel.resilience4j.sliding-window-size	配置滑动窗口的大小，该窗口用于在 CircuitBreaker 关闭时记录调用的结果。slidingWindowSize 配置滑动窗口的大小。滑动窗口可以是基于计数或基于时间的窗口。如果 slidingWindowType 是 COUNT_BASED，则最后的 slidingWindowSize 调用会被记录并聚合。如果 slidingWindowType 是 TIME_BASED，则最后 slidingWindowSize 秒的调用会被记录并聚合。slidingWindowSize 必须大于 0。minimumNumberOfCalls 必须大于 0。如果 slidingWindowType 是 COUNT_BASED，则 minimumNumberOfCalls 不能大于 slidingWindowSize。如果 slidingWindowType 是 TIME_BASED，您可以选择任何您需要的。默认 slidingWindowSize 为 100。	100	整数

Name	描述	默认值	类型
camel.resilience4j.sliding-window-type	配置滑动窗口的类型，用于记录 CircuitBreaker 关闭时调用的结果。滑动窗口可以是基于计数或基于时间的窗口。如果 slidingWindowType 是 COUNT_BASED，则最后的 slidingWindowSize 调用会被记录并聚合。如果 slidingWindowType 是 TIME_BASED，则最后 slidingWindowSize 秒的调用会被记录并聚合。默认 slidingWindowType 是 COUNT_BASED。	COUNT_BASED	字符串
camel.resilience4j.slow-call-duration-threshold	配置上面的持续时间阈值（秒），调用被视为缓慢，并增加较慢的调用百分比。默认值为 60 秒。	60	整数
camel.resilience4j.slow-call-rate-threshold	以百分比为单位配置阈值。当调用持续时间大于 slowCallDurationThreshold Duration 时，CircuitBreaker 会将调用视为较慢。当较慢的调用百分比相等或大于阈值时，CircuitBreaker 过渡到 open，并启动短路调用。阈值必须大于 0，而不是大于 100。默认值为 100 百分比，这意味着所有记录的调用都必须比 slowCallDurationThreshold 慢。		æµ®ç‚¹â€¼
camel.resilience4j.wait-duration-in-open-state	配置等待持续时间（以秒为单位），指定 CircuitBreaker 应该保持打开的时间，然后再切换到半次。默认值为 60 秒。	60	整数
camel.resilience4j.writable-stack-trace-enabled	启用可写入堆栈跟踪。当设置为 false 时，Exception.getStackTrace 返回一个零长度数组。当断路器处于开路状态时，这可用于减少日志垃圾邮件，因为存在例外的原因（断路器是短路调用）。	true	布尔值
camel.rest.api-component	用作 REST API 的 Camel 组件名称（如 swagger）如果没有明确配置 API 组件，则 Camel 会查找负责服务并生成 REST API 文档的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestApiProcessorFactory。如果找到其中任何一个，则使用它。		字符串
camel.rest.api-context-path	设置领导的 API 上下文路径将使用的 REST API 服务。这在使用使用 context-path 部署 Web 应用程序的组件（如 camel-servlet ）时使用它。		字符串
camel.rest.api-context-route-id	设置用于服务 REST API 的路由的路由 ID。默认情况下，路由将使用自动分配的路由 ID。		字符串
camel.rest.api-host	要将特定主机名用于 API 文档（如 swagger），这可用于用这个配置的主机名覆盖生成的主机。		字符串

Name	描述	默认值	类型
camel.rest.api-property	允许为 api 文档配置任意数量的附加属性(swagger)。例如，将属性 api.title 设置为我的冷却。		Map
camel.rest.api-vendor-extension	是否在 Rest API 中启用供应商扩展。如果启用，Camel 将包含额外信息作为厂商扩展名（例如，以 x- 开头的键），如路由 ID、类名称等。在导入 API 文档时，并非所有第三方 API 网关和工具都支持 vendor-extensions。	false	布尔值
camel.rest.binding-mode	设置要使用的绑定模式。默认值为 off。		RestBindingMode
camel.rest.client-request-validation	是否启用客户端请求验证，以检查客户端的 Content-Type 和 Accept 标头是否受到其 consume/produces 设置的 Rest-DSL 配置的支持。这可以打开，以启用此检查。如果验证错误，则返回 HTTP Status code 415 或 406。默认值为 false。	false	布尔值
camel.rest.component	用于 REST 传输(consumer)的 Camel Rest 组件，如 netty-http, jetty, servlet, undertow。如果没有明确配置组件，则 Camel 会查找是否有与 Rest DSL 集成的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestConsumerFactory。如果找到其中任何一个，则使用它。		字符串
camel.rest.component-property	允许为正在使用的其他组件配置任意数量的附加属性。		Map
camel.rest.consumer-property	允许为使用中的其他使用者配置任意数量的附加属性。		Map
camel.rest.context-path	设置 REST 服务将使用的前导上下文路径。这在使用使用 context-path 部署 Web 应用程序的组件（如 camel-servlet ）时使用它。或者对于包含 HTTP 服务器的 camel-jetty 或 camel-netty-http 等组件。		字符串
camel.rest.cors-headers	允许配置自定义 CORS 标头。		Map
camel.rest.data-format-property	允许为使用的数据格式配置多个额外属性。例如，将属性 prettyPrint 设置为 true，以便以用户友善模式输出 json。属性可以加上前缀来表示选项仅适用于 JSON 或 XML，以及 IN 或 OUT。前缀为： json.in. json.out. xml.in. xml.out。例如，值为 xml.out.mustBeJAXBElement 的键仅用于传出的 XML 数据格式。没有前缀的密钥是所有情况的通用密钥。		Map

Name	描述	默认值	类型
camel.rest.enable-cors	是否在 HTTP 响应中启用 CORS 标头。默认值为 false。	false	布尔值
camel.rest.endpoint-property	允许为使用中的其他端点配置多个额外的属性。		Map
camel.rest.host	用于公开 REST 服务的主机名。		字符串
camel.rest.host-name-resolver	如果没有明确配置的主机名，这个 resolver 会用于计算 REST 服务将要使用的主机名。		RestHostNameResolver
camel.rest.json-data-format	要使用的特定 json 数据格式的名称。默认将使用 json-jackson。重要：此选项仅用于设置数据格式的自定义名称，而不是引用现有数据格式实例。		字符串
camel.rest.port	用于公开 REST 服务的主机名。请注意，如果您使用 servlet 组件，则此处配置的端口号不适用，因为使用中的端口号是 servlet 组件使用的实际端口号。例如，如果使用 Apache Tomcat，它的 tomcat http 端口，如果使用 Apache Karaf，它的在 Karaf 中的 HTTP 服务，它默认使用端口 8181。虽然在这些情况下，这里设置端口号，但允许工具和 JMX 知道端口号，因此建议将端口号设置为 servlet 引擎使用的数字。		字符串
camel.rest.producer-api-doc	设置 api 文档的位置，REST 生成者将根据这个文档来验证 REST uri 和查询参数是否有效。这需要将 camel-swagger-java 添加到 classpath 中，任何缺失的配置都会导致 Camel 在启动时失败并报告错误。默认情况下从 classpath 加载的 api 文档的位置，但您可以使用 file: 或 http: 引用从文件或 http url 加载的资源。		字符串
camel.rest.producer-component	设置要用作 REST 生成者的 Camel 组件的名称。		字符串
camel.rest.scheme	用于公开 REST 服务的方案。通常支持 http 或 https。默认值为 http。		字符串
camel.rest.skip-binding-on-error-code	如果存在自定义 HTTP 错误代码标头，是否跳过输出绑定。这允许构建没有绑定到 json / xml 等自定义错误消息，否则成功信息会这样做。	false	布尔值
camel.rest.use-x-forward-headers	是否将 X-Forward 标头用于主机和相关设置。默认值为 true。	true	布尔值
camel.rest.xml-data-format	要使用的特定 XML 数据格式的名称。默认情况下将使用 jaxb。重要：此选项仅用于设置数据格式的自定义名称，而不是引用现有数据格式实例。		字符串

Name	描述	默认值	类型
camel.rest.api-context-id-pattern	**弃用** 设置 CamelContext id 特征，以只允许 CamelContext 中名称与特征匹配的其他服务的 Rest API。特征 name 指的是 CamelContext 名称，仅匹配当前的 CamelContext。对于任何其他值，特征使用来自 PatternHelper#matchPattern (String,String)的规则。		字符串
camel.rest.api-context-listing	**弃用** 设置是否启用了 JVM 中带有 REST 服务的所有可用 CamelContext 的列表。如果启用，它将允许发现这些上下文，如果为 false，则只使用当前的 CamelContext。	false	布尔值

第 117 章 SLACK

支持生成者和消费者

Slack 组件允许您连接到 Slack 实例，并通过预先建立的 Slack 传入 Webhook 传递消息包含在消息正文中。

117.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 slack 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-slack-starter</artifactId>
</dependency>
```

117.2. URI 格式

向频道发送消息。

```
slack:#channel[?options]
```

向 slackuser 发送直接消息。

```
slack:@userID[?options]
```

117.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

117.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

117.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

117.4. 组件选项

Slack 组件支持 5 个选项，如下所列。

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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
令牌 （令牌）	要使用的令牌。		字符串
webhookUrl (webhook)	传入的 Webhook URL。		字符串

117.5. 端点选项

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：

117.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
channel (common)	**必需** 频道名称(syntax #name)或 slackuser (syntax userName)来直接向用户发送信息。		字符串

117.5.2. 查询参数(29 参数)

Name	描述	默认值	类型
token (common)	要使用的令牌。		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
conversationType (consumer)	对话类型。 Enum 值： • PUBLIC_CHANNEL • PRIVATE_CHANNEL • MPIM • IM	PUBLIC_CHANNEL	ConversationType
maxResults (consumer)	轮询的 Max Result。	10	字符串
naturalOrder (consumer)	以自然顺序（最旧的）或未创建交换。	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
serverUrl (consumer)	Slack 实例的服务器 URL。		字符串
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern

Name	描述	默认值	类型
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
iconEmoji (producer)	**弃用了** 使用 Slack emoji 作为 avatar。		字符串
iconUrl (producer)	**弃用了** 组件在向频道或用户发送消息时使用的 avatar。		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
username (producer)	**弃用**，这是 bot 将信息发送到频道或用户时具有的用户名。		字符串
webhookUrl (producer)	传入的 Webhook URL。		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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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

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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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

117.6. 在PRINT XML 中配置

带有 XML 的 Slack 组件必须配置为包含传入 webhook url 的 Spring 或 Blueprint bean，或包含集成的应用程序令牌作为参数。

```
<bean id="slack" class="org.apache.camel.component.slack.SlackComponent">
    <property name="webhookUrl"
value="https://hooks.slack.com/services/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"/>
    <property name="token" value="xoxb-12345678901-1234567890123-
xxxxxxxxxxxxxxxxxxxxxxxx"/>
</bean>
```

对于 Java，您可以使用 Java 代码进行配置。

117.7. 示例

带有 Blueprint 的 CamelContext 可能如下：

```
<?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>
```

117.8. 制作者

现在，您可以使用令牌来发送消息，而不是 WebhookUrl。

```
from("direct:test")
    .to("slack:#random?token=RAW(<YOUR_TOKEN>)");
```

现在，您可以使用 Slack API 模型来创建块。您可以在此阅读更多有关 https://api.slack.com/block-kit 的信息。

```
    public void testSlackAPIModelMessage() {
        Message message = new Message();
        message.setBlocks(Collections.singletonList(SectionBlock
                .builder()
                .text(MarkdownTextObject
                        .builder()
                        .text("*Hello from Camel!*")
                        .build())
                .build()));

        template.sendBody(test, message);
    }
```

117.9. 消费者

您还可以将消费者用于频道中的消息。

```
 from("slack://general?token=RAW(<YOUR_TOKEN>)&maxResults=1")
    .to("mock:result");
```

这样，您将从常规频道获得最后一条信息。消费者将跟踪使用的最后一个消息的时间戳，并在下一次轮询中从该时间戳检查。

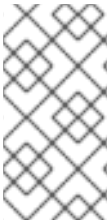
您需要创建一个 Slack 应用程序，并在您的工作区中使用它。

使用 'Bot User OAuth Access Token' 作为消费者端点的令牌。

注意

添加对应的历史 (channels:history 或 groups:history 或 mpim:history 或 im:history)，并读 (channels:read 或 groups:read 或 mpim:read 或 im:read) 用户令牌范围到您的应用程序，为它授予权限来查看相关频道中的消息。您需要使用 conversationType 选项设置它(PUBLIC_CHANNEL,PRIVATE_CHANNEL,MPIM,IM)

naturalOrder 选项允许使用从最旧的到最新消息。最初，您会得到最新的第一个信息，并会向后使用 (message 3 jpeg message 2 busybox message 1)



注意

您可以使用 conversationType 选项从没有公开的频道读取历史记录和消息 (PUBLIC_CHANNEL,PRIVATE_CHANNEL,MPIM,IM)

117.10. 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.slack.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.slack.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.slack.enabled	是否启用 slack 组件的自动配置。这默认是启用的。		布尔值
camel.component.slack.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.slack.token	要使用的令牌。		字符串
camel.component.slack.webhook-url	传入的 Webhook URL。		字符串

第 118 章 SMB

从 Camel 4.3 开始

仅支持消费者。

服务器消息块(SMB)组件提供了一种原生连接到 SMB 文件共享的方法，如 Microsoft Windows 或 Samba 提供的。

118.1. 依赖项

当在 Camel Spring Boot 中使用 camel-smb 时，请将以下 Maven 依赖项添加到 pom.xml 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-smb-starter</artifactId>
</dependency>
```

118.2. URI 格式

```
smb:address[:port]/shareName[?options]
```

118.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

118.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

118.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

118.4. 组件选项

SMB 组件支持 2 个选项，如下所列。

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 Camel 消费者试图获取传入的消息或类似信息时，会出现异常（如果可能），现在将被作为消息进行处理，并由路由 Error Handler 处理。重要：只有在第三方组件允许 Camel 抛出异常时，才能警报这一点。有些组件仅在内部处理，因此无法 bridgeErrorHandler。在其他情况下，我们可能会将 Camel 组件提高到第三方组件中的 hook，并使其可能用于将来的版本。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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

118.5. 端点选项

SMB 端点使用 URI 语法进行配置：

```
smb:hostname:port/shareName
```

使用以下 路径和 查询参数 ：

118.5.1. 路径参数(3 参数)

Name	描述	默认值	类型
hostname (consumer)	**必需** 共享主机名或 IP 地址。		字符串
port (consumer)	共享端口号。	445	int
sharename (consumer)	要连接的共享的名称。		字符串

118.6. 查询参数(26 参数)

Name	描述	默认值	类型
idempotentRepository (consumer)	可插拔存储库 org.apache.camel.spi.IdempotentRepository，如果未指定，则默认为 MemoryIdempotentRepository。		IdempotentRepository
path (consumer)	**必需** 共享中的路径，以使用其中的文件。		字符串
searchPattern (consumer)	用于列出文件的搜索模式。	*.txt	字符串

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
bridgeErrorHandler (consumer (advanced))	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当 Camel 消费者试图获取传入的消息或类似信息时，会出现异常（如果可能），现在将被作为消息进行处理，并由路由 Error Handler 处理。重要：只有在第三方组件允许 Camel 抛出异常时，才能警报这一点。有些组件仅在内部处理，因此无法 bridgeErrorHandler。在其他情况下，我们可能会将 Camel 组件提高到第三方组件中的 hook，并使其可能用于将来的版本。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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： ● InOnly ● InOut		ExchangePattern
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
smbIoBean (advanced)	可选的 SMB I/O bean，用于在读取/写入文件时设置文件访问属性。		SmbIOBean
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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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
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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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
域 （安全）	用户域。		字符串
password (security)	用于访问共享的密码。		字符串
用户名 （安全性）	访问共享所需的用户名。		字符串

118.7. 例子

以下示例演示了如何轮询 SMB 文件共享中的所有文件并读取这些文件的内容：

```
private void process(Exchange exchange) throws IOException {
   final File file = exchange.getMessage().getBody(File.class);
   try (InputStream inputStream = file.getInputStream()) {
      LOG.debug("Read exchange: {}, with contents: {}", file.getFileInformation(), new
String(inputStream.readAllBytes()));
   }
}

public void configure() {
   fromF("smb:%s/%s?username=%s&password=%s&path=/", service.address(),
service.shareName(), service.userName(), service.password())
      .process(this::process)
      .to("mock:result");
}
```

注意

提供的 File 对象不是 java.io.File 实例，而是一个 com.hierynomus.smbj.share.File 实例。

第 119 章 SNMP

自 Camel 2.1 起

支持生成者和消费者

SNMP 组件可让您轮询 SNMP 功能的设备或接收陷阱

119.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 snmp 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-snmp-starter</artifactId>
</dependency>
```

119.2. URI 格式

```
snmp://hostname[:port][?Options]
```

组件支持从 SNMP 启用的设备和接收陷阱中轮询 OID 值。

119.3. SNMP PRODUCER

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。

119.4. 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

119.4.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

119.4.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

119.5. 组件选项

SNMP 组件支持 5 个选项，如下所列。

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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
healthCheckConsumerEnabled (health)	用于从这个组件启用或禁用所有基于消费者的健康检查。	true	布尔值
healthCheckProducerEnabled (health)	用于从此组件启用或禁用所有基于制作者的健康检查。注意：默认情况下，Camel 禁用了所有基于健康检查的制作者。您可以通过设置 camel.health.producersEnabled=true 来全局打开制作者检查。	true	布尔值

119.6. 端点选项

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：

119.6.1. 路径参数(2 参数)

Name	描述	默认值	类型
host (common)	**必需** SNMP 启用的设备的主机名。		字符串
port (common)	SNMP 启用设备的 **所需** 端口号。		整数

119.6.2. 查询参数(36 参数)

Name	描述	默认值	类型
OIDs (common)	定义您感兴趣的值。请仔细查看 Wikipedia 以更好地了解。您可以提供单个 OID 或 OID 的独立列表。示例：oids=1.3.6.1.2.1.1.3.0,1.3.6.1.2.1.25.3.2.1.5.1,1.3.6.1.2.1.3.5.1.1.1,1.3.6.1.2.1.43.5.1.1.11.1.		OIDList
protocol (common)	您可以在此处选择要使用的协议。您可以使用 udp 或 tcp。 Enum 值： • tcp • udp	udp	字符串
retries (common)	定义在取消请求前重试的频率。	2	int
snmpCommunity (common)	为 snmp 请求设置社区八位字符串。	public	字符串
snmpContextEngineId (common)	设置范围 PDU 的上下文引擎 ID 字段。		字符串
snmpContextName (common)	设置此范围 PDU 的上下文名称字段。		字符串
snmpVersion (common)	为请求设置 snmp 版本。值 0 表示 SNMPv1，1 表示 SNMPv2c，值 3 表示 SNMPv3。 Enum 值： • 0 • 1 • 3	0	int
timeout (common)	在 millis 中设置请求的超时值。	1500	int

Name	描述	默认值	类型
type (common)	要执行的操作，如轮询、陷阱等。 Enum 值： • TRAP • POLL • GET_NEXT		SnmpActionType
delay (consumer)	设置更新率（以秒为单位）。	60000	long
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
treeList (consumer)	设置范围 PDU 的标记，如果其树中有子元素，则其列表将显示为列表。	false	布尔值
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut		ExchangePattern
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy

Name	描述	默认值	类型
lazyStartProducer (producer (advanced))	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
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
initialDelay (scheduler)	第一次轮询开始前的毫秒。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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
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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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 null，则 authenticationProtocol 还必须为 null。RFC3414 11.2 需要密码短语的最小长度为 8 字节。如果 authenticationPassphrase 的长度小于 8 字节，则抛出 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。可能的值为：MD5、SHA1。 Enum 值： • MD5 • SHA1		字符串

Name	描述	默认值	类型
PrivacyPassphrase (security)	隐私密码短语。如果没有 null，则 PrivacyProtocol 也不能为空。RFC3414 11.2 需要密码短语的最小长度为 8 字节。如果 authenticationPassphrase 的长度小于 8 字节，则抛出 IllegalArgumentException。		字符串
PrivacyProtocol (security)	与这个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
Securitylevel (security)	为这个目标设置安全级别。安全模型必须支持提供的安全级别，这要依赖于与此目标设置的安全名称相关的信息。值 1 表示：没有身份验证，且没有加密。任何人都可以使用此安全级别创建和读取消息。值 2 表示：身份验证且没有加密。只有具有正确身份验证密钥的用户可以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可以使用此安全级别创建消息，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。 Enum 值： • 1 • 2 • 3	3	int
securityName (security)	设置用于此目标的安全名称。		字符串

119.7. 轮询的结果

鉴于这种情况，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>
```

您可能会识别的结果多于请求....1.3.6.1.2.1.1.1.0。
这个设备会在这个特殊情况下自动填充。因此，可能绝对发生，您收到的不仅仅是您请求...be 准备。

OID 以点表示开头

```
.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）。确保明确设置了不是默认值的 snmpVersion，例如，在可以查询具有不同版本的 SNMP 表的情况下。其他可能的值有 version2c 和 version3。

119.8. 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
```

您可以获取 SNMP TRAP 的社区，其中包含消息标头 'securityName'，SNMP TRAP 的对等地址，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");
```

119.9. 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.snmp.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.snmp.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.snmp.enabled	是否启用 snmp 组件的自动配置。这默认是启用的。		布尔值
camel.component.snmp.health-check-consumer-enabled	用于从这个组件启用或禁用所有基于消费者的健康检查。	true	布尔值

Name	描述	默认值	类型
camel.component.snmp.health-check-producer-enabled	用于从此组件启用或禁用所有基于制作者的健康检查。注意：默认情况下，Camel 禁用了所有基于健康检查的制作者。您可以通过设置 camel.health.producersEnabled=true 来全局打开制作者检查。	true	布尔值
camel.component.snmp.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

第 120 章 SOAP

SOAP 是一种数据格式，它使用 JAXB2 和 JAX-WS 注释进行 marshal 和 unmarshal SOAP 有效负载。它提供了 Apache CXF 的基本功能，无需 CXF 堆栈。

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。

120.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 soap 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-soap-starter</artifactId>
</dependency>
```

120.2. SOAP 选项

SOAP dataformat 支持 6 个选项，如下所列。

Name	默认值	Java 类型	描述
contextPath		字符串	JAXB 类所在的 必需 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**	指的是从 registry 中查找一个元素策略。元素名称策略用于两个目的。第一种方法是查找给定对象的 xml 元素名称，并在将对象放入 SOAP 消息时执行 soap 操作。第二个是查找给定 soap 故障名称的 Exception 类。以下三元素策略类名称开箱即用。QNameStrategy - 使用实例化时配置的固定 qName。不支持 TypeNameStrategy - 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。一个 exception lookup is not supported ServiceInterfaceStrategy - 使用来自 webservice 接口的信息来确定类型名称，并查找 SOAP fault All three 类的异常类位于软件包名称 org.apache.camel.dataformat.soap.name if you generated the web service stub code with cxf-codegen 或类似工具，您可能需要使用 ServiceInterfaceStrategy。在这种情况下，您没有注解的服务接口，您应该使用 QNameStrategy 或 TypeNameStrategy。
version		**字符串**	SOAP 版本应为 1.1 或 1.2。默认为 1.1。
namespacePrefixRef		**字符串**	使用 JAXB 或 SOAP 总结时，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**	针对现有架构进行验证。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。

120.3. ELEMENTNAMESTRATEGY

元素名称策略用于两个目的。第一种方法是查找给定对象的 xml 元素名称，并在将对象放入 SOAP 消息时执行 soap 操作。第二个是查找给定 soap 故障名称的 Exception 类。

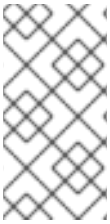
策略	使用方法
QNameStrategy	使用实例化时配置的固定 qName。不支持异常查找
TypeNameStrategy	使用给定类型的 @XMLType 注释中的名称和命名空间。如果没有设置命名空间，则使用 package-info。不支持异常查找
ServiceInterfaceStrategy	使用 webservice 界面中的信息来确定类型名称，并查找 SOAP 故障的异常类

如果您使用 cxf-codegen 或类似的工具生成 web 服务 stub 代码，您可能需要使用 ServiceInterfaceStrategy。在这种情况下，您没有注解的服务接口，您应该使用 QNameStrategy 或 TypeNameStrategy。

120.4. 使用 JAVA DSL

以下示例使用名为 DataFormat of soap，它使用软件包 com.example.customerservice 来配置来初始化 JAXBContext。第二个参数是 ElementNameStrategy。路由可以 marshal 常规对象和例外。（请注意，以下只是将 SOAP Envelope 发送到队列。Web 服务提供商实际上需要侦听用于实际发生 SOAP 调用的队列，在这种情况下，它将是一个 SOAP 请求。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");
```



注意

另请参阅
，因为 SOAP dataformat 会继承于此处的 JAXB dataformat 大多数设置。

120.4.1. 使用 SOAP 1.2

从 Camel 2.11 开始

```
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 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>
```

120.5. 多部分消息

ServiceInterfaceStrategy 支持多部分 SOAP 信息。ServiceInterfaceStrategy 必须使用服务接口定义进行初始化，该定义根据 JAX-WS 2.2 进行注解，并满足 Document Barestyle 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进行初始化，该参数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);
```

120.5.1. holder 对象映射

JAX-WS 指定对 In/Out 和 Out 参数使用类型参数 javax.xml.ws.Holder 对象。您可以直接使用 parameterized-type 实例。根据 Holder 的类的 JAXB 映射，camel-soap DataFormat marshals Holder 值。在 unmarshalled 响应中，没有为 \'Holder 对象提供映射。

120.6. 例子

120.6.1. WebService 客户端

以下路由支持 marshalling the request 和 unmarshalling a response or fault。

```
String WS_URI = "cxf://http://myserver/customerservice?
serviceClass=com.example.customerservice&dataFormat=RAW";
```

```
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，并使 SOAP 调用上述路由。

```
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());
```

120.6.2. WebService Server

使用以下路由设置侦听 jms 队列 customerServiceQueue 的 webservice 服务器，并使用类 CustomerServiceImpl 处理请求。该课程的 customerServiceImpl 应实施接口 CustomerService。无需直接实例化服务器类，它可以作为常规 Bean 在 spring 上下文中定义。

```
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);
```

120.7. 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.dataformat.soapjaxb.context-path	JAXB 类所在的软件包名称。		字符串
camel.dataformat.soapjaxb.element-name-strategy-ref	指的是从 registry 中查找一个元素策略。元素名称策略用于两个目的。第一种方法是查找给定对象的 xml 元素名称，并在将对象放入 SOAP 消息时执行 soap 操作。第二个是查找给定 soap 故障名称的 Exception 类。以下三元素策略类名称开箱即用。QNameStrategy - 使用实例化时配置的固定 qName。不支持 TypeNameStrategy - 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。一个 exception lookup is not supported ServiceInterfaceStrategy - 使用来自 webservice 接口的信息来确定类型名称，并查找 SOAP fault All three 类的异常类位于软件包名称 org.apache.camel.dataformat.soap.name if you generated the web service stub code with cxf-codegen 或类似工具，您可能需要使用 ServiceInterfaceStrategy。在这种情况下，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	是否启用 soapjaxb 数据格式的自动配置。这默认是启用的。		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 总结时，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	针对现有架构进行验证。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。默认为 1.1。	1.1	字符串

第 121 章 SPLUNK

从 Camel 2.13 开始

支持生成者和消费者

Splunk 组件使用 Splunk 提供的客户端 api 提供对 Splunk 的访问，并可让您在 Splunk 中发布和搜索事件。

121.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 mvapich 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-splunk-starter</artifactId>
</dependency>
```

121.2. URI 格式

```
splunk://[endpoint]?[options]
```

121.3. 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

121.3.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

121.3.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

121.4. 组件选项

Splunk 组件支持 6 个选项，如下所列。

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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
mvapichConfigurationFactory (advanced)	使用 SplunkConfigurationFactory。		SplunkConfigurationFactory
healthCheckConsumerEnabled (health)	用于从这个组件启用或禁用所有基于消费者的健康检查。	true	布尔值
healthCheckProducerEnabled (health)	用于从此组件启用或禁用所有基于制作者的健康检查。注意：默认情况下，Camel 禁用了所有基于健康检查的制作者。您可以通过设置 camel.health.producersEnabled=true 来全局打开制作者检查。	true	布尔值

121.5. 端点选项

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：

121.5.1. 路径参数(1 参数)

Name	描述	default	类型
name (common)	**必需** 的名称没有目的。		字符串

121.5.2. 查询参数(45 参数)

Name	描述	default	类型
app (common)	Splunk 应用程序.		字符串
connectionTimeout (common)	连接到 Splunk 服务器时的 MS 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
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
savedSearch (consumer)	要运行的 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，而不是在批处理中发送。	false	布尔值
bridgeErrorHandler (consumer (advanced))	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	default	类型
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut		ExchangePattern
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
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)	有效负载应插入 raw。	false	布尔值
source (producer)	Splunk 源参数.		字符串
sourceType (producer)	Splunk sourcetype 参数。		字符串
tcpReceiverLocalPort (producer)	Splunk tcp 接收器端口在 mvapich 服务器上本地定义。（例如，如果 mvapich 端口 9997 映射到 12345，tcpReceiverLocalPort 必须是 9997）。		整数
tcpReceiverPort (producer)	Splunk tcp 接收器端口。		int
lazyStartProducer (producer (advanced))	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	default	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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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
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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象

Name	描述	default	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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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
password (security)	Splunk 的密码。		字符串
SSLProtocol (security)	设置要使用的 ssl 协议。 Enum 值： • TLSv1.2 • TLSv1.1 • TLSv1 • SSLv3	TLSv1.2	SSLSecurityProtocol
令牌 （安全）	Splunk 的用户令牌。当两者都被设置时，这优先于密码。		字符串
用户名 （安全性）	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 时很有用，以避免 app. 服务器 https 处理。	false	布尔值

121.6. 生成者端点

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。当使用流模式时，Splunk 在事件进入索引前具有一些内部缓冲区（大约 1MB 或更多）。如果您需要实时，最好使用提交或 tcp 模式。
提交	提交模式.使用 Splunk rest api 将事件发布到指定索引中，如果未指定，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。

发布事件时，消息正文应包含 SplunkEvent。查看邮件正文下的注释。

Example

```
from("direct:start").convertBodyTo(SplunkEvent.class)
    .to("splunk://submit?
username=user&password=123&index=myindex&sourceType=someSourceType&source=my
Source")...
```

在本例中，需要转换器转换为 SplunkEvent 类。

121.7. 消费者端点：

端点	描述
normal	执行正常搜索，并在搜索选项中需要搜索查询。
realtime	对实时数据执行实时搜索，并在搜索选项中需要搜索查询。
savedsearch	根据 mvapich 中保存的搜索查询执行搜索，并在 savedSearch 选项中需要查询的名称。

Example

```
    from("splunk://normal?delay=5000&username=user&password=123&initEarliestTime=-
10s&search=search index=myindex sourcetype=someSourcetype")
        .to("direct:search-result");
```

camel-mvapich 组件为每个搜索结果创建一个路由交换，在正文中带有 SplunkEvent。

121.8. 消息正文

Splunk 对键/值对中的数据进行操作。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，它将在每个消费者的搜索结果中返回。

您可以通过在 producer 端点上设置 raw 选项，将原始数据发送到 Splunk。这可用于 json/xml 和其他 Splunk 在支持中构建的其他有效负载。

121.9. 使用案例

使用 music 为 tweets 搜索 Twitter，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;
    }
}
```

为 tweets 搜索 Splunk：

```
    from("splunk://normal?username=foo&password=bar&initEarliestTime=-
2m&search=search index=camel-tweets sourcetype=twitter")
        .log("${body}");
```

121.10. 其他评论

Splunk 附带各种选项，利用预构建的应用程序利用机器生成的数据进行分析和显示。例如，jmx 应用可用于发布 jmx 属性，如路由和 jvm 指标到 Splunk，并在仪表板中显示此属性。

121.11. 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.splunk.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.splunk.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.splunk.enabled	是否启用 mvapich 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.splunk.health-check-consumer-enabled	用于从这个组件启用或禁用所有基于消费者的健康检查。	true	布尔值
camel.component.splunk.health-check-producer-enabled	用于从此组件启用或禁用所有基于制作者的健康检查。注意：默认情况下，Camel 禁用了所有基于健康检查的制作者。您可以通过设置 camel.health.producersEnabled=true 来全局打开制作者检查。	true	布尔值
camel.component.splunk.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.splunk.splunk-configuration-factory	使用 SplunkConfigurationFactory。选项是 org.apache.camel.component.mvapich.SplunkConfigurationFactory 类型。		SplunkConfigurationFactory

第 122 章 SPRING BATCH

Since Camel 2.10

仅支持生成者

Spring Batch 组件和支持类提供 Camel 和 Spring 批处理 基础架构之间的集成桥接。

122.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 spring-batch 时，使用以下 Maven 依赖项来启用对自动配置的支持：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-spring-batch-starter</artifactId>
</dependency>
```

122.2. URI 格式

```
spring-batch:jobName[?options]
```

其中，jobName 代表 Camel registry 中的 Spring Batch 作业的名称。如果提供了 JobRegistry，则用于定位作业。

此组件仅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。

122.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

122.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

122.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

122.4. 组件选项

Spring Batch 组件支持 4 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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

122.5. 端点选项

Spring Batch 端点使用 URI 语法进行配置：

```
spring-batch:jobName
```

以下是 path 和 query 参数：

122.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
jobname (producer)	**必需** registry 中的 Spring Batch 作业的名称。		字符串

122.5.2. 查询参数(4 参数)

Name	描述	默认值	类型
jobFromHeader (producer)	显式定义是否应从标头而不是 URI 获取 jobName。	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
lazyStartProducer (producer (advanced))	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

122.6. 使用方法

当 Spring Batch 组件收到消息时，它会触发作业执行。作业是使用根据以下算法解析的 org.springframework.batch.core.launch.JobLaucher 实例执行的。

- ***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，则使用给定名称在 Camel Registry 中搜索 JobLauncher。***
- ***如果在 Camel Registry 中的 jobLauncher 名称下注册了 JobLauncher，则使用它。***
- ***如果以上任何步骤都无法解析 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会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，其他数据类型将转换为 Strings。

122.7. 例子

触发 Spring Batch 作业执行：

```
from("direct:startBatch").to("spring-batch:myJob");
```

使用明确设置的 JobLauncher 触发 Spring Batch 作业执行。

```
from("direct:startBatch").to("spring-batch:myJob?jobLauncherRef=myJobLauncher");
```

JobLauncher 返回的 JobExecution 实例由 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();
```

122.8. 支持类

除了组件外，Camel Spring Batch 还提供可用于 hook Spring Batch 基础架构的支持类。

122.8.1. 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 队列中读取数据：

```
<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>
```

122.8.2. CamelItemWriter

CamelItemWriter 与 CamelItemReader 类似，但专用于编写已处理的数据块。

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>
```

122.8.3. CamelItemProcessor

CamelItemProcessor 是 Spring Batch org.springframework.batch.item.ItemProcessor 接口的实现。后一种实施转发 Request Reply 模式，将批处理项目的处理委托给 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，以下代码片段使用直接端点和简单 表达式语言 对批处理项目执行简单的处理。 http://camel.apache.org/direct.html

```
<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>
```

122.8.4. 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。

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，设置为 BEFORE 或 AFTER 值。

以下示例片断将 Spring Batch 作业执行事件发送到 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>
```

122.9. 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.spring-batch.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.spring-batch.enabled	是否启用 spring-batch 组件的自动配置。这默认是启用的。		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。		JobLauncher
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。		JobRegistry
camel.component.spring-batch.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

第 123 章 SPRING JDBC

Since Camel 3.10

仅支持生成者

Spring JDBC 组件是 JDBC 组件的扩展，它具有一个额外的功能，可与 Spring Transaction Manager 集成。

123.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 spring-jdbc 时，使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-spring-jdbc-starter</artifactId>
</dependency>
```

版本使用 BOM 以下列方式指定：

```
<dependencyManagement>
      <dependencies>
         <dependency>
            <groupId>com.redhat.camel.springboot.platform</groupId>
            <artifactId>camel-spring-boot-bom</artifactId>
            <version>${camel-spring-boot-version}</version>
            <type>pom</type>
            <scope>import</scope>
         </dependency>
      </dependencies>
   </dependencyManagement>
```

123.2. 配置选项

Camel 组件在两个级别上配置：

- 组件级别

- 端点级别

123.2.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

123.2.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

123.3. 组件选项

Spring JDBC 组件支持下面列出的 4 个选项。

Name	描述	默认值	类型
datasource (producer)	使用 DataSource 实例，而不是根据 registry 中的名称查找数据源。		DataSource

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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
connectionStrategy (advanced)	使用自定义策略来使用连接。在使用 spring-jdbc 组件时不要使用自定义策略，因为默认使用特殊的 Spring ConnectionStrategy 支持 Spring Transactions。		ConnectionStrategy

123.4. 端点选项

Spring JDBC 端点使用 URI 语法进行配置：

```
spring-jdbc:dataSourceName
```

以下是 path 和 query 参数：

123.4.1. 路径参数(1 参数)

Name	描述	default	类型
dataSourceName (producer)	在 Registry 中查找所需的数据源名称。**如果名称是 dataSource 或 default，则 Camel 将尝试从 registry 中查找默认 DataSource，这意味着如果只有一个找到的 DataSource 实例，则会使用此 DataSource。**		字符串

123.4.2. 查询参数 (14 参数)

Name	描述	default	类型
allowNamedParameters (producer)	是否在查询中使用命名参数。	true	布尔值
outputClass (producer)	指定在 outputType=SelectOne 或 SelectList 时用作转换的完整软件包和类名称。		字符串
outputType (producer)	确定制作者应使用的输出。 Enum 值： • SelectOne • SelectList • StreamList	SelectList	JdbcOutputType
parameters (producer)	java.sql.Statement 的可选参数。例如，要设置 maxRows、fetchSize 等。		Map
readSize (producer)	轮询查询可以读取的默认最大行数。默认值为 0。		int

Name	描述	default	类型
resetAutoCommit (producer)	Camel 将 JDBC 连接上的 autoCommit 设置为 false，在执行声明后提交更改，并在末尾重置连接的 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，您很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)	是否使用事务。	false	布尔值
useGetBytesForBlob (producer)	以字节而不是字符串数据形式读取 BLOB 列。对于某些数据库（如 Oracle ）可能需要这个数据库，其中必须以字节形式读取 BLOB 列。	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。这允许使用指定占位符定义查询，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 或 JDBC 3.0 的旧语义。JDBC 4.0 使用 columnLabel 获取列名称，其中 JDBC 3.0 使用 columnName 或 columnLabel。不幸的是，JDBC 驱动程序的行为不同，如果您使用此组件解决了这个问题，则可以使用此选项来排除 JDBC 驱动程序的问题。	true	布尔值

Name	描述	default	类型
lazyStartProducer (producer (advanced))	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
beanRowMapper (advanced)	使用 outputClass 时使用自定义 org.apache.camel.component.jdbc.BeanRowMapper。默认实现会小写，行名称和跳过下划线和横线。例如 CUST_ID 被映射为 custId。		BeanRowMapper
connectionStrategy (advanced)	使用自定义策略来使用连接。在使用 spring-jdbc 组件时不要使用自定义策略，因为默认使用特殊的 Spring ConnectionStrategy 支持 Spring Transactions。		ConnectionStrategy
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。		JdbcPrepareStatementStrategy

123.5. SPRING BOOT AUTO-CONFIGURATION

*组**件支持下面列出的 4 个**选项。*

Name	描述	默认值	类型
camel.component.spring-jdbc.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.spring-jdbc.connection-strategy	使用自定义策略来使用连接。在使用 spring-jdbc 组件时不要使用自定义策略，因为默认使用特殊的 Spring ConnectionStrategy 支持 Spring Transactions。选项是一个 org.apache.camel.component.jdbc.ConnectionStrategy 类型。		ConnectionStrategy
camel.component.spring-jdbc.enabled	是否启用 spring-jdbc 组件的自动配置。这默认是启用的。		布尔值
camel.component.spring-jdbc.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

第 124 章 SPRING LDAP

从 Camel 2.11 开始

仅支持生成者

Spring LDAP 组件为 Spring LDAP 提供 Camel 包装器。

124.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 spring-ldap 时，使用以下 Maven 依赖项来启用对自动配置的支持：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-spring-ldap-starter</artifactId>
</dependency>
```

124.2. URI 格式

```
spring-ldap:springLdapTemplate[?options]
```

其中 springLdapTemplate 是 Spring LDAP 模板 bean 的名称。在此 bean 中，您要配置用于 LDAP 访问的 URL 和凭证。

124.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

124.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

124.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

124.4. 组件选项

Spring LDAP 组件支持 2 个选项，如下所列。

Name	描述	默认值	类型

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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

124.5. 端点选项

Spring LDAP 端点使用 URI 语法进行配置：

```
spring-ldap:templateName
```

以下是 path 和 query 参数：

124.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
templateName (producer)	Spring LDAP 模板 bean 所需的 名称。		字符串

124.5.2. 查询参数(3 参数)

Name	描述	默认值	类型
operation (producer)	**需要执行** LDAP 操作。 Enum 值： • 搜索 • BIND • UNBIND • 身份验证 • MODIFY_ATTRIBUTES • FUNCTION_DRIVEN		LdapOperation
scope (producer)	搜索操作的范围。 Enum 值： • object • onelevel • subtree	subtree	字符串
lazyStartProducer (producer (advanced))	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

124.6. 使用方法

组件仅支持制作者端点。尝试创建消费者端点可能会导致 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带有键 dn (function_driven 操作不需要)的条目，用于指定要执行的 LDAP 操作的根节点。映射的其他条目是特定于操作的。

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。

124.6.1. 搜索

消息正文必须具有带有键 过滤器 的条目。该值必须是 代表 有效 LDAP 过滤器的字符串，请参阅 http://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol#Search_and_Compare。

124.6.2. 绑定

消息正文必须具有带有键 属性 的条目。该值必须是 javax.naming.directory.Attributes 的一个实例，指定要创建的 LDAP 节点。

124.6.3. unbind

不需要其他条目，删除具有指定 dn 的节点。

124.6.4. 身份验证

消息正文必须具有带有键 过滤器 和密码 的条目。这些值必须是 String 实例，分别代表有效的 LDAP 过滤器和用户密码。

124.6.5. 修改属性

消息正文必须具有带有键 modify Items 的条目。该值必须是类型为 javax.naming.directory.ModificationItem的数组的实例

124.6.6. function-Driven

消息正文必须具有带有键 function 和 request 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。请求 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 方法预期的参数。 S type 参数表示被调用的 LdapTemplate 方法返回的响应类型。此操作允许动态调用上述操作没有涵盖的 LdapTemplate 方法。

重要定义

为了避免拼写错误，在 org.apache.camel.springldap.SpringLdapProducer 中定义了以下常数：

- **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";**

以下是 createMap 功能示例：

```
from(“direct:start”)
   .setBody(constant(createMap()))
   .to("spring-ldap:ldapTemplate?operation=BIND");
```

在这里，createMap 功能返回 Map 对象，其中包含有关 ldap 服务器属性和域名的信息。

```
private static Map<String, Object> createMap() {
   BasicAttributes basicAttributes = new BasicAttributes();
   basicAttributes.put("cn", "Name Surname");
   basicAttributes.put("sn", "Surname");
   basicAttributes.put("objectClass", "person");
   Map<String, Object> map = new HashMap<>();
   map.put(SpringLdapProducer.DN, "cn=LdapDN,dc=example,dc=org");
```

```
    map.put(SpringLdapProducer.ATTRIBUTES, basicAttributes);
    return map;
}
```

对于上例，还必须使用 Spring Boot 自动配置或 LdapTemplate Bean 配置 ldap 连接。

Spring Boot 自动配置示例：

```
spring.ldap.password=passwordforldapserver
spring.ldap.urls=urlForLdapServer
spring.ldap.username=usernameForLdapServer
```

124.7. SPRING BOOT AUTO-CONFIGURATION

组件支持下面列出的 3 个选项。

Name	描述	默认值	类型
camel.component.spring-ldap.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.spring-ldap.enabled	是否启用 spring-ldap 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.spring-ldap.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

第 125 章 SPRING RABBITMQ

从 Camel 3.8 开始

支持生成者和消费者

Spring RabbitMQ 组件允许您使用 Spring RabbitMQ 客户端从 RabbitMQ 实例生成和使用消息。

125.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 spring-rabbitmq 时，使用以下 Maven 依赖项来启用对自动配置的支持：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-spring-rabbitmq-starter</artifactId>
</dependency>
```

版本使用 BOM 以下列方式指定：

```
<dependencyManagement>
        <dependencies>
          <dependency>
            <groupId>com.redhat.camel.springboot.platform</groupId>
            <artifactId>camel-spring-boot-bom</artifactId>
            <version>${camel-spring-boot-version}</version>
            <type>pom</type>
            <scope>import</scope>
          </dependency>
        </dependencies>
    </dependencyManagement>
```

125.2. URI 格式

```
spring-rabbitmq:exchangeName?[options]
```

exchangeName 决定生成的消息发送到的交换。对于消费者，exchangeName 决定队列绑定到的交换。

125.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

125.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

125.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

125.4. 组件选项

Spring RabbitMQ 组件支持下面列出的 29 个选项。

Name	描述	默认值	类型
amqpAdmin (common)	**Autowired** Optional AMQP Admin 服务用于自动声明元素（队列、交换、绑定）。		AmqpAdmin
ConnectionFactory (common)	**Autowired** 要使用的连接工厂。连接工厂必须在组件或端点上配置。		ConnectionFactory
testConnectionOnStartup (common)	指定是否在启动时测试连接。这样可确保 Camel 启动所有 JMS 用户具有与 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 没有使用失败的连接启动。JMS 制作者也经过测试。	false	布尔值
autoDeclare (consumer)	指定消费者是否应该在启动时自动声明交换、队列和路由密钥之间的绑定。启用这对开发可能很好，方便代理上的交换、队列和绑定。	false	布尔值
autoStartup (consumer)	指定消费者容器是否应该自动启动。	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
deadLetterExchange (consumer)	死信交换的名称。		字符串

Name	描述	默认值	类型
deadLetterExchangeType (consumer)	死信交换的类型。 Enum 值： • 直接 • fanout • 标头 • topic	直接	字符串
deadLetterQueue (consumer)	死信队列的名称。		字符串
deadLetterRoutingKey (consumer)	死信交换的路由密钥。		字符串
maximumRetryAttempts (consumer)	如果 Camel 无法处理消息，则 Rabbitmq 使用者将重试相同的消息的次数。	5	int
rejectAndDontRequeue (consumer)	Rabbitmq 使用者是否应该拒绝消息，而无需重新排队。这可让配置了代理，将失败的消息发送到 Dead Letter Exchange/Queue。	true	布尔值
retryDelay (consumer)	在 msec 中，Rabbitmq 消费者将在 Camel 无法处理的消息前等待。	1000	int
concurrentConsumers (consumer (advanced))	用户数量。	1	int
errorHandler (consumer (advanced))	使用自定义 ErrorHandler 处理消息监听器 (consumer)中的异常。		ErrorHandler
listenerContainerFactory (consumer (advanced))	使用自定义工厂来创建和配置 ListenerContainer，供消费者用于接收消息。		ListenerContainerFactory
maxConcurrentConsumers (consumer (advanced))	使用者的最大数量（仅适用于 SMLC）。		整数

Name	描述	默认值	类型
messageListenerContainerType (consumer (advanced))	MessageListenerContainer 的类型。 Enum 值： • DMLC • SMLC	DMLC	字符串
prefetchCount (consumer (advanced))	告知代理在单个请求中发送到每个消费者的消息数量。通常，这可以设置得非常高，以提高吞吐量。	250	int
retry (consumer (advanced))	要使用的自定义重试配置。如果这被配置，则不会使用用于重试的 maximumRetryAttempts 等其他设置。		RetryOperationsInterceptor
shutdownTimeout (consumer (advanced))	容器停止后，等待 worker 的时间（以毫秒为单位）。如果有任何 worker 在关闭信号进入时处于活跃状态，只要这些程序可以在这个超时时间内结束，就可以完成处理。	5000	long
allowNullBody (producer)	是否允许发送不包含正文的消息。如果此选项为 false，且消息正文为 null，则会抛出 MessageConversionException。	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

Name	描述	默认值	类型
replyTimeout (producer)	在执行请求/回复消息时，指定在等待回复消息时使用的超时时间（以毫秒为单位）。默认值为 5 秒。负值表示一个不正确的超时。	5000	long
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
ignoreDeclarationExceptions (advanced)	在声明时切换忽略异常，如不匹配的属性。	false	布尔值
messageConverter (advanced)	要使用自定义的 MessageConverter，以便您可以控制如何映射到 org.springframework.amqp.core.Message。		MessageConverter
messagePropertiesConverter (advanced)	要使用自定义 MessagePropertiesConverter，以便您可以控制如何映射到 org.springframework.amqp.core.MessageProperties。		MessagePropertiesConverter
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头到 Camel 消息。		HeaderFilterStrategy

125.5. 端点选项

Spring RabbitMQ 端点使用 URI 语法进行配置：

```
spring-rabbitmq:exchangeName
```

以下是 path 和 query 参数：

125.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
exchangeName (common)	**必需** 交换名称，决定生成的消息要发送到的交换。如果是消费者，交换名称决定了队列将绑定到的交换。注意：要使用默认交换，则不要使用空名称，而是使用 default。		字符串

125.5.2. 查询参数(34 参数)

Name	描述	默认值	类型
ConnectionFactory (common)	要使用的连接工厂。连接工厂必须在组件或端点上配置。		ConnectionFactory
disableReplyTo (common)	指定 Camel 是否忽略消息中的 ReplyTo 标头。如果为 true，Camel 不会向 ReplyTo 标头中指定的目的地发送回复。如果您希望 Camel 消耗路由，且您不想 Camel 自动发送回复消息，您可以使用这个选项，因为代码中的另一个组件处理回复消息。如果要使用 Camel 作为不同消息代理之间的代理，而您想要将消息从一个系统路由到另一个系统，也可以使用此选项。	false	布尔值
routingKey (common)	要使用的路由键的值。默认为空，在使用默认（或任何直接）交换时很有用，但如果交换是实例的标头交换，则正常操作。		字符串

Name	描述	默认值	类型
testConnectionOnStartup (common)	指定是否在启动时测试连接。这样可确保 Camel 启动所有 JMS 用户具有与 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 没有使用失败的连接启动。JMS 制作者也经过测试。	false	布尔值
acknowledgeMode (consumer)	控制容器的行为与消息确认相关的标志。最常见的用法是让容器处理确认（因此，侦听器不需要了解频道或消息）。如果监听器将使用 Channel.basicAck (long, boolean)发送确认，则设置为 AcknowledgeMode.MANUAL。手动攻击与事务或非事务频道一致，但如果您没有其他工作，而不是接收单个消息，则事务可能是不必要的。设置为 AcknowledgeMode.NONE 以告知代理不会期望任何确认，它会假定所有消息在发送后马上被确认（这在原生 Rabbit 代理术语中自动确认）。如果 AcknowledgeMode.NONE，则频道无法事务处理（因此如果该标志意外设置了该标志，容器将失败）。 Enum 值： • NONE • 手动 • AUTO		AcknowledgeMode

Name	描述	默认值	类型
asyncConsumer (consumer)	消费者是否异步处理交换。如果启用，消费者可以从队列中获取下一个消息，而前面的消息会被异步处理（由异步路由引擎）。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），则在消费者从队列中获取下一个消息前完全处理交换。	false	布尔值
autoDeclare (consumer)	指定消费者是否应该在启动时自动声明交换、队列和路由密钥之间的绑定。	true	布尔值
autoStartup (consumer)	指定消费者容器是否应该自动启动。	true	布尔值
deadLetterExchange (consumer)	死信交换的名称。		字符串
deadLetterExchangeType (consumer)	死信交换的类型。 Enum 值： • 直接 • fanout • 标头 • topic	直接	字符串
deadLetterQueue (consumer)	死信队列的名称。		字符串
deadLetterRoutingKey (consumer)	死信交换的路由密钥。		字符串

Name	描述	默认值	类型
exchangeType (consumer)	交换的类型。 Enum 值： • 直接 • fanout • 标头 • topic	直接	字符串
exclusive (consumer)	对于专用消费者，设置为 true。	false	布尔值
maximumRetryAttempts (consumer)	如果 Camel 无法处理消息，则 Rabbitmq 使用者将重试相同的消息的次数。	5	int
noLocal (consumer)	对于非本地消费者，设置为 true。	false	布尔值
Queue (consumer)	用于消耗消息的队列。可以使用逗号分隔多个队列名称。如果尚未配置任何配置，则 Camel 将生成唯一 id 作为消费者的队列名称。		字符串
rejectAndDontRequeue (consumer)	Rabbitmq 使用者是否应该拒绝消息，而无需重新排队。这可让配置了代理，将失败的消息发送到 Dead Letter Exchange/Queue。	true	布尔值
retryDelay (consumer)	在 msec 中，Rabbitmq 消费者将在 Camel 无法处理的消息前等待。	1000	int

Name	描述	默认值	类型
bridgeErrorHandler (consumer (advanced))	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 (advanced))	用户数量。		整数
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
maxConcurrentConsumers (consumer (advanced))	使用者的最大数量（仅适用于 SMLC）。		整数
messageListenerContainerType (consumer (advanced))	MessageListenerContainer 的类型。 Enum 值： • DMLC • SMLC	DMLC	字符串

Name	描述	默认值	类型
prefetchCount (consumer (advanced))	告知代理在单个请求中发送多少消息。通常，这可以设置得非常高，以提高吞吐量。		整数
retry (consumer (advanced))	要使用的自定义重试配置。如果这被配置，则不会使用用于重试的 maximumRetryAttempts 等其他设置。		RetryOperationsInterceptor
replyTimeout (producer)	在执行请求/回复消息时，指定在等待回复消息时使用的超时时间（以毫秒为单位）。默认值为 5 秒。负值表示一个不正确的超时。	5000	long
usePublisherConnection (producer)	为发布者和消费者使用单独的连接。	false	布尔值
lazyStartProducer (producer (advanced))	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
args (advanced)	指定用于配置不同 RabbitMQ 概念的参数，每个元素都需要不同的前缀：arg.consumer. arg.exchange. arg.queue. arg.queue. arg.dlq.exchange. arg.dlq.queue. arg.dlq.binding。例如，使用消息 ttl 参数声明队列：args=arg.queue.x-message-ttl=60000。		Map

Name	描述	默认值	类型
messageConverter (advanced)	要使用自定义的 MessageConverter，以便您可以控制如何映射到 org.springframework.amqp.core.Message。		MessageConverter
messagePropertiesConverter (advanced)	要使用自定义 MessagePropertiesConverter，以便您可以控制如何映射到 org.springframework.amqp.core.MessageProperties。		MessagePropertiesConverter
同步 (advanced)	设置是否应严格使用同步处理。	false	布尔值

125.6. 消息标头

***Spring RabbitMQ** 组件支持以下列出的 2 个消息标头：*

Name	描述	默认值	类型
CamelSpringRabbitmqRoutingOverrideKey (common) 常数：**ROUTING_OVERRIDE_KEY**	Exchange 密钥。		字符串
CamelSpringRabbitmqExchangeOverrideName (common) 常量：**EXCHANGE_OVERRIDE_NAME**	交换名称。		字符串

125.7. 使用连接工厂

若要连接到 RabbitMQ，您必须设置 ConnectionFactory（与 JMS 相同）以及登录详细信息，如下所述。

建议使用来自 spring-rabbit 的 caching ConnectionFactory，因为它附带连接池。

```
<bean id="rabbitConnectionFactory"
class="org.springframework.amqp.rabbit.connection.CachingConnectionFactory">
  <property name="uri" value="amqp://localhost:5672"/>
</bean>
```

默认情况下，ConnectionFactory 会被自动探测到，因此您可以执行它。

```
<camelContext>
  <route>
    <from uri="direct:cheese"/>
    <to uri="spring-rabbitmq:foo?routingKey=cheese"/>
  </route>
</camelContext>
```

125.8. 默认交换名称

要使用默认交换名称（这是 RabbitMQ 中的空交换名称），您必须在 endpoint uri 中使用 default，如下所示：

```
to("spring-rabbitmq:default?routingKey=foo")
```

125.9. 自动声明交换、队列和绑定

在从 RabbitMQ 发送或接收消息之前，您必须首先设置交换、队列和绑定。

在开发模式中，Camel 可以自动执行此操作。您可以通过在 SpringRabbitMQComponent 中设置 autoDeclare=true 来启用它。

然后，Spring RabbitMQ 会自动声明元素，并设置交换、队列和路由密钥之间的绑定。

可以使用多值 args 选项来配置元素。

例如，要将队列指定为 durable 和 exclusive，您可以使用 arg.queue.durable=true&arg.queue.exclusive=true 配置 endpoint uri。

Exchanges

选项	类型	描述	default
autoDelete	布尔值	如果服务器在不再使用交换时应删除交换，则为 true （如果所有绑定都已被删除）。	false
durable	布尔值	在服务器重新启动后，持久化交换将保留下来。	true

您还可以配置任何其他 x 参数。请参阅 RabbitMQ 文档中的详细信息。

队列

选项	类型	描述	default
autoDelete	布尔值	如果服务器在不再使用交换时应删除交换，则为 true （如果所有绑定都已被删除）。	false
durable	布尔值	持久队列将在服务器重启后保留。	false
exclusive	布尔值	队列是 exclusive	false
x-dead-letter-exchange	字符串	死信交换的名称。如果没有配置，则使用组件配置的值。	
x-dead-letter-routing-key	字符串	死信交换的路由密钥。如果没有配置，则使用组件配置的值。	

您还可以配置任何其他 x- 参数，如使用 x-message-ttl 等其他参数进行实时消息。请参阅 RabbitMQ 文档中的详细信息。

125.10. 从 CAMEL 映射到 RABBITMQ

消息正文从 Camel Message body 映射到 byte[]，这是 RabbitMQ 用于消息正文的类型。Camel 使用其类型转换器将消息正文转换为字节数组。

Spring Rabbit 开箱即用，支持映射 Java 序列化对象，但 Camel Spring RabbitMQ 不支持此功能，因为存在安全漏洞，使用 Java 对象是一种不良的设计，因为它强制执行强大的耦合。

自定义消息标头从 Camel Message 标头映射到 RabbitMQ 标头。这可以通过在 Camel 组件上配置新的 HeaderFilterStrategy 实现来自定义。

125.11. 请求/恢复

使用 RabbitMQ 直接回复到，支持请求和回复 消息传递。

以下示例执行请求/回复，其中消息使用 cheese Exchange 名称和路由密钥 foo.bar （由第二代 Camel 路由使用）发送，该路由由第二代 Camel 路由使用，它将在消息前面加上 'Hello '，然后发回消息。

因此，如果将 World 作为消息正文发送到 direct:start，则可以查看正在记录的消息

- log:request jpeg World
- log:input especially World
- log:response TOKEN Hello World

```
from("direct:start")
   .to("log:request")
   .to(ExchangePattern.InOut, "spring-rabbitmq:cheese?routingKey=foo.bar")
   .to("log:response");

from("spring-rabbitmq:cheese?queues=myqueue&routingKey=foo.bar")
   .to("log:input")
   .transform(body().prepend("Hello "));
```

125.12. 重复使用端点并发送到运行时计算的不同目的地

如果您需要发送消息到大量不同的 RabbitMQ 交换，您必须重复使用端点并在消息标头中指定实际目

的地。这允许 Camel 重复使用同一端点，但发送到不同的交换。这大大减少了在内存和线程资源中创建的端点数量。

注意

使用 toD 比使用标头指定动态目的地要简单。

您可以使用以下标头指定：

标头	类型	描述
CamelSpringRabbitmqExchangeOverrideName	**字符串**	交换名称。
CamelSpringRabbitmqRoutingOverrideKey	**字符串**	路由密钥。

例如，以下路由演示了如何在运行时计算目的地，并使用它来覆盖端点 URL 中出现的交换：

```
from("file://inbox")
  .to("bean:computeDestination")
  .to("spring-rabbitmq:dummy");
```

交换名称 dummy 只是一个占位符。它必须作为 RabbitMQ 端点 URL 的一部分提供，但在本示例中会忽略它。

在 computeDestination bean 中，通过设置 CamelRabbitmqExchangeOverrideName 标头来指定实际目的地，如下所示：

```
public void setExchangeHeader(Exchange exchange) {
   String region = ....
   exchange.getIn().setHeader("CamelSpringRabbitmqExchangeOverrideName", "order-" +
region);
}
```

Camel 读取此标头，并将其用作交换名称，而不是端点上配置的名称。因此，在这个示例中，Camel 将消息发送到 spring-rabbitmq:order-emea，假设 region 值为 emea。

生产者同时从 交换中删除 CamelSpringRabbitmqName 和 CamelSpringRabbitmqRoutingOverrideKey 标头，且不会将它们传播到创建的 Rabbitmq 消息，以避免路由中的意外循环（当消息转发到另一个 RabbitMQ 端点时）。

125.13. 使用 TOD

如果您需要发送消息到大量不同的交换，您必须重复使用端点并指定动态目的地（使用 toD ）。

例如，您需要向交换发送带有订购类型的信息，然后您可以使用 toD，如下所示：

```
from("direct:order")
  .toD("spring-rabbit:order-${header.orderType}");
```

125.14. SPRING BOOT AUTO-CONFIGURATION

组件支持下列 30 个选项。

Name	描述	默认值	类型
camel.component.spring-rabbitmq.allow-null-body	是否允许发送不包含正文的消息。如果此选项为 false，且消息正文为 null，则会抛出 MessageConversionException。	false	布尔值
camel.component.spring-rabbitmq.amqp-admin	可选的 AMQP Admin 服务用于自动声明元素（队列、交换、绑定）。选项是一个 org.springframework.amqp.core.AmqpAdmin 类型。		AmqpAdmin
camel.component.spring-rabbitmq.auto-declare	指定消费者是否应该在启动时自动声明交换、队列和路由密钥之间的绑定。启用这对开发可能很好，方便代理上的交换、队列和绑定。	false	布尔值
camel.component.spring-rabbitmq.auto-startup	指定消费者容器是否应该自动启动。	true	布尔值

Name	描述	默认值	类型
camel.component.spring-rabbitmq.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.spring-rabbitmq.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.spring-rabbitmq.concurrent-consumers	用户数量。	1	整数
camel.component.spring-rabbitmq.connection-factory	要使用的连接工厂。连接工厂必须在组件或端点上配置。选项是 org.springframework.amqp.rabbit.connection.ConnectionFactory 类型。		ConnectionFactory
camel.component.spring-rabbitmq.dead-letter-exchange	死信交换的名称。		字符串
camel.component.spring-rabbitmq.dead-letter-exchange-type	死信交换的类型。	直接	字符串
camel.component.spring-rabbitmq.dead-letter-queue	死信队列的名称。		字符串

Name	描述	默认值	类型
camel.component.spring-rabbitmq.dead-letter-routing-key	死信交换的路由密钥。		字符串
camel.component.spring-rabbitmq.enabled	是否启用 spring-rabbitmq 组件的自动配置。这默认是启用的。		布尔值
camel.component.spring-rabbitmq.error-handler	使用自定义 ErrorHandler 处理消息监听器(consumer)中的异常。选项是一个 org.springframework.util.ErrorHandler 类型。		ErrorHandler
camel.component.spring-rabbitmq.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头到 Camel 消息。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		HeaderFilterStrategy
camel.component.spring-rabbitmq.ignore-declaration-exceptions	在声明时切换忽略异常，如不匹配的属性。	false	布尔值
camel.component.spring-rabbitmq.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

Name	描述	默认值	类型
camel.component.spring-rabbitmq.listener-container-factory	使用自定义工厂来创建和配置 ListenerContainer，供消费者用于接收消息。选项是 org.apache.camel.component.springrabbit.ListenerContainerFactory 类型。		ListenerContainerFactory
camel.component.spring-rabbitmq.max-concurrent-consumers	使用者的最大数量（仅适用于 SMLC）。		整数
camel.component.spring-rabbitmq.maximum-retry-attempts	如果 Camel 无法处理消息，则 Rabbitmq 使用者将重试相同的消息的次数。	5	整数
camel.component.spring-rabbitmq.message-converter	要使用自定义的 MessageConverter，以便您可以控制如何映射到 org.springframework.amqp.core.Message。选项是一个 org.springframework.amqp.support.converter.MessageConverter 类型。		MessageConverter
camel.component.spring-rabbitmq.message-listener-container-type	MessageListenerContainer 的类型。	DMLC	字符串
camel.component.spring-rabbitmq.message-properties-converter	要使用自定义 MessagePropertiesConverter，以便您可以控制如何映射到 org.springframework.amqp.core.MessageProperties。选项是 org.apache.camel.component.springrabbit.MessagePropertiesConverter 类型。		MessagePropertiesConverter

Name	描述	默认值	类型
camel.component.spring-rabbitmq.prefetch-count	告知代理在单个请求中发送到每个消费者的消息数量。通常，这可以设置得非常高，以提高吞吐量。	250	整数
camel.component.spring-rabbitmq.reject-and-dont-requeue	Rabbitmq 使用者是否应该拒绝消息，而无需重新排队。这可让配置了代理，将失败的消息发送到 Dead Letter Exchange/Queue。	true	布尔值
camel.component.spring-rabbitmq.reply-timeout	在执行请求/回复消息时，指定在等待回复消息时使用的超时时间（以毫秒为单位）。默认值为 5 秒。负值表示一个不正确的超时。选项是一个长类型。	5000	Long
camel.component.spring-rabbitmq.retry	要使用的自定义重试配置。如果这被配置，则不会使用用于重试的 maximumRetryAttempts 等其他设置。选项是一个 org.springframework.retry.interceptor.RetryOperationsInterceptor 类型。		RetryOperationsInterceptor
camel.component.spring-rabbitmq.retry-delay	在 msec 中，Rabbitmq 消费者将在 Camel 无法处理的消息前等待。	1000	整数
camel.component.spring-rabbitmq.shutdown-timeout	容器停止后，等待 worker 的时间（以毫秒为单位）。如果有任何 worker 在关闭信号进入时处于活跃状态，只要这些程序可以在这个超时时间内结束，就可以完成处理。选项是一个长类型。	5000	Long

Name	描述	默认值	类型
camel.component.spring-rabbitmq.test-connection-on-startup	指定是否在启动时测试连接。这样可确保 Camel 启动所有 JMS 用户具有与 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 没有使用失败的连接启动。JMS 制作者也经过测试。	false	布尔值

第 126 章 SPRING REDIS

支持生成者和消费者。

此组件允许从 Redis 发送和接收信息。Redis 是一种高级键值存储，其中键可以包含字符串、哈希、列表、集合和排序的集合。另外，Red Hatis 为应用程序间的通信提供 pub/sub 功能。Camel 提供了一个制作者，用于执行命令、订阅 pub/sub 消息的使用者和用于过滤重复消息的幂等存储库。

先决条件

要使用此组件，您必须有一个 Redis 服务器正在运行。

126.1. 依赖项

当在 Camel Spring Boot 中使用 spring-redis 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-spring-redis-starter</artifactId>
</dependency>
```

使用 BOM 获取版本。

```
<dependencyManagement>
   <dependencies>
      <dependency>
         <groupId>com.redhat.camel.springboot.platform</groupId>
         <artifactId>camel-spring-boot-bom</artifactId>
         <version>${camel-spring-boot-version}</version>
         <type>pom</type>
         <scope>import</scope>
      </dependency>
   </dependencies>
</dependencyManagement>
```

126.2. URI 格式

```
spring-redis://host:port[?options]
```

126.3. 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

126.3.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 URL。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常具有常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

126.3.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项也被归类为：端点是否用作消费者（来自）还是作为生成者(to)用于两者。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式 的安全 方法。

在配置选项时，最好使用 Property Placeholders，它允许您硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许您从代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

126.4. 组件选项

Spring Redis 组件支持 4 个选项，如下所列。

Name	描述	默认值	类型
redisTemplate (common)	对要使用的预先配置的 RedisTemplate 实例的 **Autowired** 参考。		RedisTemplate
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，使用者使用 org.apache.camel.spi.ExceptionHandler 处理异常，该处理程序将记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
lazyStartProducer (producer)	生成者是否应懒惰启动 （在第一个消息中）。通过启动 lazy，您可以使用它来允许 CamelContext 和路由在生成者启动期间启动，并导致路由启动失败。通过启动 lazy，Camel 的路由错误处理程序会在路由消息时处理任何启动失败。请注意，当第一个消息被处理、创建和启动时，生成者可能需要稍等片刻，并延长处理的总处理时间。	false	布尔值
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 连接工厂和 AWS 客户端。	true	布尔值

126.5. 端点选项

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：

126.5.1. 路径参数(2 参数)

Name	描述	默认值	类型
host (common)	**必需** 运行 Redis 服务器的主机。		字符串
port (common)	**所需的** Redis 服务器端口号。		整数

126.5.2. 查询参数(10 参数)

Name	描述	默认值	类型
Channels (common)	要订阅的主题名称或名称特征列表。可以使用逗号分隔多个名称。		字符串
command (common)	默认命令，该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。 Enum 值： • PING • SET • GET • QUIT • EXISTS • DEL • TYPE • FLUSHDB • KEYS • RANDOMKEY • RENAME • RENAMENX • RENAMEX • DBSIZE • EXPIRE • EXPIREAT • TTL • 选择 • MOVE • FLUSHALL • GETSET • MGET • SETNX • SETEX • MSET • MSETNX	SET	命令

Name	描述	默认值	类型
	• DECRBY • DECR • INCRBY • INCR • 附加 • SUBSTR • HSET • HGET • HSETNX • HMSET • HMGET • HINCRBY • HEXISTS • HDEL • HLEN • HKEYS • HVALS • HGETALL • RPUSH • LPUSH • LLEN • LRANGE • LTRIM • LINDEX • LSET • LREM • LPOP • RPOP • RPOPLPUSH • SADD • SMEMBERS • SREM • SPOP		

Name	描述	默认值	类型
	• SMOVE • SCARD • SISMEMBER • SINTER • SINTERSTORE • SUNION • SUNIONSTORE • SDIFF • SDIFFSTORE • SRANDMEMBER • ZADD • ZRANGE • ZREM • ZINCRBY • ZRANK • ZREVRANK • ZREVRANGE • ZCARD • ZSCORE • MULTI • DISCARD • EXEC • WATCH • UNWATCH • SORT • BLPOP • BRPOP • AUTH • 订阅 • PUBLISH • 取消订阅 • PSUBSCRIBE • PUNSUBSCRIBE		

Name	描述	默认值	类型
	• ZCOUNT • ZRANGEBYSCORE • ZREVRANGEBYSCORE • ZREMRANGEBYRANK • ZREMRANGEBYSCORE • ZUNIONSTORE • ZINTERSTORE • 保存 • BGSAVE • BGREWRITEAOF • LASTSAVE • SHUTDOWN • INFO • MONITOR • SLAVEOF • CONFIG • STRLEN • SYNC • LPUSHX • PERSIST • RPUSHX • ECHO • LINSERT • DEBUG • BRPOPLPUSH • SETBIT • GETBIT • SETRANGE • GETRANGE • PEXPIRE • PEXPIREAT • GEOADD • GEODIST		

Name	描述	默认值	类型
	• GEOHASH • GEOPOS • GEORADIUS • GEORADIUSBYMEMBER		
ConnectionFactory (common)	引用要使用的预先配置的 RedisConnectionFactory 实例。		RedisConnectionFactory
redisTemplate (common)	引用要使用的预先配置的 RedisTemplate 实例。		RedisTemplate
S erializer (common)	引用要使用的预先配置的 RedisSerializer 实例。		RedisSerializer
bridgeErrorHandler (consumer (advanced))	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获取传入消息或类似消息时发生的任何异常都会被处理为消息，并由路由 Error Handler 处理。消费者默认为使用 org.apache.camel.spi.ExceptionHandler 来处理异常。这些例外日志为 WARN 或 ERROR 级别，并忽略。	False	布尔值
exceptionHandler (consumer (advanced))	要让使用者使用自定义例外处理程序：如果启用 bridgeErrorHandler 选项，则不会使用这个选项。默认情况下，消费者将处理异常，该例外记录在 WARN 或 ERROR 级别，并忽略。		ExceptionHandler
exchangePattern (consumer (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
listenerContainer (consumer (advanced))	对预先配置的 RedisMessageListenerContainer 实例的引用。		RedisMessageListenerContainer
lazyStartProducer (Producer (advanced))	生成者是否应懒惰启动 （在第一个消息中）。通过启动 lazy，您可以使用它来允许 CamelContext 和路由在生产者启动期间启动，并导致路由启动失败。通过将这个启动延迟延迟，启动失败可以在路由消息期间通过 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

126.6. 消息标头

***Spring Redis** 组件支持 29 消息标头，如下所列：*

Name	描述	默认值	类型
CamelRedis.Command (producer) 常数：COMMAND	要执行的命令。		字符串
CamelRedis.Key (common) 常数：KEY	密钥。		字符串
CamelRedis.Keys (common) 常数 ：KEYS	密钥。		集合
CamelRedis.Field (common) 常数 ：FIELD	该字段。		字符串
CamelRedis.Fields (common) 常数：FIELDS	这些字段。		集合
CamelRedis.Value (common) constant: VALUE	该值。		对象
CamelRedis.Values (common) constant: VALUES	值。		Map
CamelRedis.Start (common) 常量 ：START	开始。		Long
CamelRedis.End (common) 常数 ：END	结束.		Long

Name	描述	默认值	类型
CamelRedis.Timeout (common) 常数 ： TIMEOUT	超时。		Long
CamelRedis.Offset (common) 常数 ： OFFSET	偏移。		Long
CamelRedis.Destination (common) 常量 ： DESTINATION	目的地。		字符串
CamelRedis.Channel (common) 常数 ： CHANNEL	频道。		byte[] 或 String
CamelRedis.Message (common) 恒定： MESSAGE	消息。		对象
CamelRedis.Index (common) 常数： INDEX	索引。		Long
CamelRedis.Position (common) 常数： POSITION	位置。		字符串
CamelRedis.Pivot (common) 常数 ： PIVOT	pivot。		字符串
CamelRedis.Count (common) 常数 ： COUNT	数量。		Long

Name	描述	默认值	类型
CamelRedis.Timestamp (common) 恒定：TIMESTAMP	时间戳。		Long
CamelRedis.Pattern (common) 常量 ： PATTERN	模式。		byte[] 或 String
CamelRedis.Db (common) 常数 ： DB	db.		整数
CamelRedis.Score (common) 常数： SCORE	分数。		å�Œ
CamelRedis.Min (common) 常数 ： MIN	分钟。		å�Œ
CamelRedis.Max (common) 常数 ： MAX	最大。		å�Œ
CamelRedis.Increment (common) 常数：INCREMENT	递增.		å�Œ
CamelRedis.WithScore (common) 常数 ：WITHSCORE	WithScore.		布尔值
CamelRedis.Latitude (common) 常数： LATITUDE	拉特语.		å�Œ

Name	描述	默认值	类型
CamelRedis.Longitude (common) 常数：LONGITUDE	拉特语.		å€Œ
CamelRedis.Radius (common) 常量：RADIUS	RADIUS。		å€Œ

126.7. 使用方法

另外，请参阅可用的单元测试。

Redis Producer

```
from("direct:start")
    .setHeader("CamelRedis.Key", constant(key))
    .setHeader("CamelRedis.Value", constant(value))
    .to("spring-redis://host:port?command=SET&redisTemplate=#redisTemplate");
```

redis Consumer

```
from("spring-redis://host:port?command=SUBSCRIBE&channels=myChannel")
    .log("Received message: ${body}");
```

注意

其中 '//host:port' 是运行 Redis 服务器的 URL 地址。

126.7.1. 由 Redis producer 评估的消息标头

制作者向服务器发出命令，每个命令都有一组具有特定类型的参数。命令执行的结果会在消息正文中返回。

哈希命令	描述	参数	结果
HSET	设置 hash 字段的字符串值	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.FIELD**/"CamelRedis.Field" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	void
HGET	获取 hash 字段的值	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.FIELD**/"CamelRedis.Field" (String)	字符串
HSETNX	仅在字段不存在时才设置 hash 字段的值	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.FIELD**/"CamelRedis.Field" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	void
HMSET	将多个散列字段设置为多个值	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUES**/"CamelRedis.Values" (Map<String, Object>)	void
HMGET	获取所有给定哈希字段的值	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.FIELDS**/"CamelRedis.Filds" (Collection<String>)	collection<Object>
HINCRBY	根据给定数字增加散列字段的整数值	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.FIELD**/"CamelRedis.Field" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Long)	Long
HEXISTS	确定是否存在 hash 字段	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.FIELD**/"CamelRedis.Field" (String)	布尔值

哈希命令	描述	参数	结果
HDEL	删除一个或多个散列字段	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.FIELD**/"CamelRedis.Field" (String)	void
HLEN	获取哈希中的字段数	**RedisConstants.KEY**/"CamelRedis.Key" (String)	Long
HKEYS	获取哈希中的所有字段	**RedisConstants.KEY**/"CamelRedis.Key" (String)	set<String>
HVALS	获取哈希中的所有值	**RedisConstants.KEY**/"CamelRedis.Key" (String)	collection<Object>
HGETALL	获取哈希中的所有字段和值	**RedisConstants.KEY**/"CamelRedis.Key" (String)	Map<String, Object>

列出命令	描述	参数	结果
RPUSH	将一个或多个值附加到列表	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	Long
RPUSHX	将值附加到列表中，只有在列表存在时	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	Long
LPUSH	向列表前添加一个或多个值	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	Long
LLEN	获取列表的长度	**RedisConstants.KEY**/"CamelRedis.Key" (String)	Long
LRANGE	从列表中获取一系列元素	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.START**/"CamelRedis.Start"Long），**RedisConstants.END**/"CamelRedis.End" (Long)	List<Object>

列出命令	描述	参数	结果
LTRIM	将列表修剪到指定的范围	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.START**/"CamelRedis.Start"Long），**RedisConstants.END**/"CamelRedis.End" (Long)	void
LINDEX	按其索引从列表中获取元素	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.INDEX**/"CamelRedis.Index" (Long)	字符串
LINSERT	在列表中的另一个元素前或之后插入一个元素	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object), **RedisConstants.PIVOT**/"CamelRedis.Pivot" (String), **RedisConstants.POSITION**/"CamelRedis.Position" (String)	Long
LSET	按其索引设置列表中的元素值	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object), **RedisConstants.INDEX**/"CamelRedis.Index" (Long)	void
LREM	从列表中删除元素	**RedisConstants.KEY**/**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object), **RedisConstants.COUNT**/"CamelRedis.Count" (Long)	Long
LPOP	删除并获取列表中的第一个元素	**RedisConstants.KEY**/"CamelRedis.Key" (String)	对象
RPOP	删除并获取列表中的最后一个元素	**RedisConstants.KEY**/"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中并返回它	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.DESTINATION**/"CamelRedis.Destination" (String)	对象

列出命令	描述	参数	结果
BRPOPLPUSH	从列表中弹出一个值，将其推送到另一个列表并返回它；或阻止到某个列表可用	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.DESTINATION**/"CamelRedis.Destination" (String), **RedisConstants.TIMEOUT**/"CamelRedis.Timeout" (Long)	对象
BLPOP	删除并获取列表中的第一个元素，或块直到可用为止	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.TIMEOUT**/"CamelRedis.Timeout" (Long)	对象
BRPOP	删除并获取列表中的最后一个元素，或块直到可用为止	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.TIMEOUT**/"CamelRedis.Timeout" (Long)	字符串

设置命令	描述	参数	结果
SADD	在集合中添加一个或多个成员	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	布尔值
SMEMBERS	获取集合中的所有成员	**RedisConstants.KEY**/"CamelRedis.Key" (String)	set<Object>
SREM	从集合中删除一个或多个成员	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	布尔值
SPOP	从集合中删除并返回随机成员	**RedisConstants.KEY**/"CamelRedis.Key" (String)	字符串
SMOVE	将成员从一个集合移动到另一个集合	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object), **RedisConstants.DESTINATION**/"CamelRedis.Destination" (String)	布尔值
SCARD	获取集合中的成员数量	**RedisConstants.KEY**/"CamelRedis.Key" (String)	Long

设置命令	描述	参数	结果
SISMEMBER	确定给定值是否是集合的成员	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	布尔值
SINTER	交集多个集合	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.KEYS**/"CamelRedis.Keys" (String)	set<Object>
SINTERSTORE	插入多个集合，并将生成的设置存储在键中	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.KEYS**/"CamelRedis.Keys" (String), **RedisConstants.DESTINATION**/"CamelRedis.Destination" (String)	void
SUNION	添加多个集合	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.KEYS**/"CamelRedis.Keys" (String)	set<Object>
SUNIONSTORE	添加多个集合，并在键中保存生成的集合	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.KEYS**/"CamelRedis.Keys" (String), **RedisConstants.DESTINATION**/"CamelRedis.Destination" (String)	void
SDIFF	减去多个集合	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.KEYS**/"CamelRedis.Keys" (String)	set<Object>
SDIFFSTORE	减去多个集合，并在键中保存生成的集合	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.KEYS**/"CamelRedis.Keys" (String), **RedisConstants.DESTINATION**/"CamelRedis.Destination" (String)	void
SRANDMEMBER	从集合中获取一个或多个随机成员	**RedisConstants.KEY**/"CamelRedis.Key" (String)	字符串

排序的集合命令	描述	参数	结果

排序的集合命令	描述	参数	结果
ZADD	将一个或多个成员添加到排序集，或者更新其分数（如果已存在）	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object), **RedisConstants.SCORE**/"CamelRedis.Score" (Double)	布尔值
ZRANGE	根据索引，在排序集中返回一系列成员	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.START**/"CamelRedis.Start"Long）, **RedisConstants.END**/"CamelRedis.End" (Long), **RedisConstants.WITHSCORE**/"CamelRedis.WithScore" (Boolean)	对象
ZREM	从排序的集合中删除一个或多个成员	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	布尔值
ZINCRBY	在排序的集合中递增成员分数	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object), **RedisConstants.INCREMENT**/"CamelRedis.Increment" (Double)	å�Œ
ZRANK	在排序的集合中确定成员的索引	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	Long
ZREVRANK	在排序的集合中确定成员的索引，分数从高到低排序	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	Long
ZREVRANGE	按索引在排序集中返回一系列成员，分数从高到低排序	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.START**/"CamelRedis.Start"Long）, **RedisConstants.END**/"CamelRedis.End" (Long), **RedisConstants.WITHSCORE**/"CamelRedis.WithScore" (Boolean)	对象
ZCARD	在排序的集合中获取成员数量	**RedisConstants.KEY**/"CamelRedis.Key" (String)	Long

排序的集合命令	描述	参数	结果
ZCOUNT	在给定值中使用分数设置中的成员计数	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.MIN**/"CamelRedis.Min" (Double), **RedisConstants.MAX**/"CamelRedis.Max" (Double)	Long
ZRANGEBYSCORE	按分数返回一系列成员（按分数排序）	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.MIN**/"CamelRedis.Min" (Double), **RedisConstants.MAX**/"CamelRedis.Max" (Double)	set<Object>
ZREVRANGEBYSCORE	返回按分数排序的一系列成员，分数从高到低排序	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.MIN**/"CamelRedis.Min" (Double), **RedisConstants.MAX**/"CamelRedis.Max" (Double)	set<Object>
ZREMRANGEBYRANK	删除给定索引中排序集中的所有成员	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.START**/"CamelRedis.Start" (Long), **RedisConstants.END**/"CamelRedis.End" (Long)	void
ZREMRANGEBYSCORE	删除在给定分数内排序集中的所有成员	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.START**/"CamelRedis.Start" (Long), **RedisConstants.END**/"CamelRedis.End" (Long)	void
ZUNIONSTORE	添加多个排序的集合，并将生成的排序集合存储在新键中	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.KEYS**/"CamelRedis.Keys" (String), **RedisConstants.DESTINATION**/"CamelRedis.Destination" (String)	void
ZINTERSTORE	插入多个排序的集合，并将生成的排序集合存储在新键中	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.KEYS**/"CamelRedis.Keys" (String), **RedisConstants.DESTINATION**/"CamelRedis.Destination" (String)	void

字符串命令	描述	参数	结果
SET	设置键的字符串值	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	void
GET	获取键值	**RedisConstants.KEY**/"CamelRedis.Key" (String)	对象
STRLEN	获取键中存储的值的长度	**RedisConstants.KEY**/"CamelRedis.Key" (String)	Long
附加	将值附加到键中	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (String)	整数
SETBIT	设置或清除存储在键的字符串值中的位	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.OFFSET**/"CamelRedis.Offset" (Long), **RedisConstants.VALUE**/"CamelRedis.Value" (Boolean)	void
GETBIT	在存储在键的字符串值中返回位值	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.OFFSET**/"CamelRedis.Offset" (Long)	布尔值
SETRANGE	覆盖从指定偏移开始的密钥时字符串的一部分	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object), **RedisConstants.OFFSET**/"CamelRedis.Offset" (Long)	void
GETRANGE	获取存储在键中的字符串的子字符串	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.START**/"CamelRedis.Start" (Long), **RedisConstants.END**/"CamelRedis.End" (Long)	字符串
SETNX	仅在键不存在时才设置键的值	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	布尔值

字符串命令	描述	参数	结果
SETEX	设置键的值和过期	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object), **RedisConstants.TIMEOUT**/"CamelRedis.Timeout" (Long), SECONDS	void
DECRBY	按给定数字减少键的整数值	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Long)	Long
DECR	逐一减少键的整数值	**RedisConstants.KEY**/"CamelRedis.Key" (String),	Long
INCRBY	以给定数量递增键的整数值	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Long)	Long
INCR	递增键的整数值	**RedisConstants.KEY**/"CamelRedis.Key" (String)	Long
MGET	获取所有给定键的值	**RedisConstants.FIELDS**/"CamelRedis.Filds" (Collection<String>)	List<Object>
MSET	将多个键设置为多个值	**RedisConstants.VALUES**/"CamelRedis.Values" (Map<String, Object>)	void
MSETNX	将多个键设置为多个值，只有在不存在键时	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	void
GETSET	设置键的字符串值，并返回其旧值	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	对象

关键命令	描述	参数	结果
EXISTS	确定键是否存在	**RedisConstants.KEY**/"CamelRedis.Key" (String)	布尔值

关键命令	描述	参数	结果
DEL	删除密钥	**RedisConstants.KEYS**/"CamelRedis.Keys" (String)	void
TYPE	确定存储在密钥中的类型	**RedisConstants.KEY**/"CamelRedis.Key" (String)	DataType
KEYS	查找与给定模式匹配的所有键	**RedisConstants.PATERN**/"CamelRedis.Pattern" (String)	collection<String>
RANDOMKEY	从 keyspace 返回随机密钥	**RedisConstants.PATERN**/"CamelRedis.Pattern" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (String)	字符串
RENAME	重命名密钥	**RedisConstants.KEY**/"CamelRedis.Key" (String)	void
RENAMENX	只有在新密钥不存在时才重命名密钥	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (String)	布尔值
EXPIRE	将密钥的时间设置为 live （以秒为单位）	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.TIMEOUT**/"CamelRedis.Timeout" (Long)	布尔值
SORT	对列表中的元素、设置或排序集中进行排序	**RedisConstants.KEY**/"CamelRedis.Key" (String)	List<Object>
PERSIST	从密钥中删除过期	**RedisConstants.KEY**/"CamelRedis.Key" (String)	布尔值
EXPIREAT	将密钥的过期时间设置为 UNIX 时间戳	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.TIMESTAMP**/"CamelRedis.Timestamp" (Long)	布尔值
PEXPIRE	以毫秒为单位设置密钥的生存时间	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.TIMEOUT**/"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	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.TIMESTAMP**/"CamelRedis.Timestamp" (Long)	布尔值

关键命令	描述	参数	结果
TTL	获取密钥实时的时间	**RedisConstants.KEY**/"CamelRedis.Key" (String)	Long
MOVE	将密钥移动到另一个数据库	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.DB**/"CamelRedis.Db" (Integer)	布尔值

地理命令	描述	参数	结果
GEOADD	将指定的 geospatial 项(latitude、yitude、name)添加到指定的键中	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.LATITUDE**/"CamelRedis.Latitude" (Double), **RedisConstants.LONGITUDE**/"CamelRedis.Longitude" (Double), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	Long
GEODIST	返回指定密钥的 geospatial 索引的两个成员之间的距离	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUES**/"CamelRedis.Values" (Object[])	distance
GEOHASH	返回代表指定键的 geospatial 索引中元素位置的有效 Geohash 字符串	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	List<String>
GEOPOS	返回指定键的 geospatial 索引中的元素的位置（长、latitude）	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object)	List<Point>
GEORADIUS	返回指定密钥的 geospatial 索引中的元素，它位于使用中央位置指定的区域以及中心(radius)的最大距离。	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.LATITUDE**/"CamelRedis.Latitude" (Double), **RedisConstants.LONGITUDE**/"CamelRedis.Longitude" (Double), **RedisConstants.RADIUS**/"CamelRedis.Radius" (Double), **RedisConstants.COUNT**/"CamelRedis.Count" (Integer)	GeoResults

地理命令	描述	参数	结果
GEORADIUSBYMEMBER	该命令与 GEORADIUS 完全不同，而是作为查询的区域中心、冗长和拉丁值，取指定密钥的 geospatial 索引中已存在的成员的名称	**RedisConstants.KEY**/"CamelRedis.Key" (String), **RedisConstants.VALUE**/"CamelRedis.Value" (Object), **RedisConstants.RADIUS**/"CamelRedis.Radius" (Double), **RedisConstants.COUNT**/"CamelRedis.Count" (Integer)	GeoResults

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 后发布的所有命令	none	void
EXEC	执行 MULTI 后发布的所有命令	none	void
WATCH	观察给定密钥，以确定 MULTI/EXEC 块的执行	**RedisConstants.KEYS**/"CamelRedis.Keys" (String)	void
UNWATCH	忘记所有监视的密钥	none	void
ECHO	回显给定字符串	**RedisConstants.VALUE**/"CamelRedis.Value" (String)	字符串
PING	对服务器发出 ping 命令	none	字符串
QUIT	关闭连接	none	void
PUBLISH	向频道发布消息	**RedisConstants.CHANNEL**/"CamelRedis.Channel" (String), **RedisConstants.MESSAGE**/"CamelRedis.Message" (Object)	void

126.8. 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.spring-redis.autowired-enabled	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
camel.component.spring-redis.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获取传入消息或类似消息时发生异常，将被作为消息进行处理，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.spring-redis.enabled	是否启用 spring-redis 组件的自动配置。这默认是启用的。		布尔值
camel.component.spring-redis.lazy-start-producer	生成者是否应懒惰启动（在第一个消息中）。通过启动 lazy，您可以允许 CamelContext 和路由在生成者启动期间启动，并导致路由启动失败。通过将这个启动延迟延迟，启动失败可以在路由消息期间通过 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.spring-redis.redis-template	引用要使用的预先配置的 RedisTemplate 实例。选项是一个 org.springframework.data.redis.core.RedisTemplate 类型。		RedisTemplate

第 127 章 SPRING SECURITY

Since Camel 2.3

Camel Spring Security 组件为 Camel 路由提供基于角色的授权。它利用 Spring Security 提供的身份验证和用户服务（以前称为 Acegi Security）并添加声明的基于角色的策略系统，以控制给定主体是否可以执行路由。

如果您不熟悉 Spring 安全身份验证和授权系统，请查看上面链接的 SpringSource 网站的当前参考文档。

127.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 spring-security 时，使用以下 Maven 依赖项来启用对自动配置的支持：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-spring-security-starter</artifactId>
</dependency>
```

127.2. 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 对象所需的 Spring Security authority (role)的名称，用于确定当前主体是否已被分配了该角色。策略对象可以被配置为 Spring Bean，或使用 Spring XML 中的 &lt ;authorizationPolicy& gt; 元素进行配置。

& lt;authorizationPolicy > 元素可以包含以下属性：

Name	默认值	描述
id	**null**	唯一的 Spring bean 标识符，用于引用路由中的策略（必需）
access	**null**	传递给访问决策管理器的 Spring Security authority 名称（必需）

Name	默认值	描述
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
authenticationAdapter	DefaultAuthenticationAdapter	用于在上下文中 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 Spring Security **SecurityContextHolder** 是否有 **Authentication** 对象。
alwaysReauthenticate	**false**	如果设置为 true，则每次访问策略时，**SpringSecurityAuthorizationPolicy** 始终都会调用 **AuthenticationManager.authenticate （）**。

127.3. 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：

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

  <bean id="accessDecisionManager"
class="org.springframework.security.access.vote.AffirmativeBased">
    <property name="allowIfAllAbstainDecisions" value="true"/>
    <property name="decisionVoters">
      <list>
```

```
        <bean class="org.springframework.security.access.vote.RoleVoter"/>
      </list>
    </property>
  </bean>

  <spring-security:authentication-manager alias="authenticationManager">
    <spring-security:authentication-provider user-service-ref="userDetailsService"/>
  </spring-security:authentication-manager>

  <spring-security:user-service id="userDetailsService">
    <spring-security:user name="jim" password="jimspassword" authorities="ROLE_USER,
ROLE_ADMIN"/>
    <spring-security:user name="bob" password="bobspassword"
authorities="ROLE_USER"/>
  </spring-security:user-service>

</beans>
```

现在设置了底层的安全对象，我们可以使用它们配置授权策略，并使用该策略控制对路由的访问：

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

   <!-- import the Spring security configuration  -->
   <import resource=
"classpath:org/apache/camel/component/spring/security/commonSecurity.xml"/>

   <authorizationPolicy id="admin" access="ROLE_ADMIN"
    authenticationManager="authenticationManager"
    accessDecisionManager="accessDecisionManager"
    xmlns="http://camel.apache.org/schema/spring-security"/>

   <camelContext id="myCamelContext" xmlns="http://camel.apache.org/schema/spring">
    <route>
      <from uri="direct:start"/>
      <!-- The exchange should be authenticated with the role -->
      <!-- of ADMIN before it is send to mock:endpoint -->
      <policy ref="admin">
        <to uri="mock:end"/>
      </policy>
    </route>
   </camelContext>
</beans>
```

在本例中，仅在以下情况下执行端点 mock:end ：

- admin SpringSecurityAuthorizationPolicy 可以找到 Spring Security Authentication 对象，如下所示：
  - 可以被验证或可以进行身份验证
  - 包含 ROLE_ADMIN 授权

127.4. 身份验证

获取用于授权的安全凭证的过程不由此组件指定。您可以根据自己的需要编写您自己的处理器或组件，从交换中获取身份验证信息。例如，您可以创建一个处理器，从源自 Jetty 组件的 HTTP 请求标头获取凭证。无论如何收集凭据，它们都需要放置在 Inmessage 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);
   }
}
```

如有必要，SpringSecurityAuthorizationPolicy 会自动验证 Authentication 对象。

注意

请注意，当您使用 SecurityContextHolder 而不是 或 除了 Exchange.AUTHENTICATION 标头外，请注意这两个问题：

1. 上下文拥有者使用 thread-local 变量来保存 Authentication 对象。任何跨线程边界的路由（如 seda 或 jms ）都会丢失 Authentication 对象。

2. Spring Security 系统要求上下文中的 Authentication 对象已经进行身份验证，并具有角色。

如需了解更多详细信息，请参阅 Spring 技术概述，第 5.3.1 节："Spring Security 中的身份验证是什么？"。

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。

如果您正在使用不使用 Spring Security 的组件，但不提供 Subject，则这很有用。

目前，只有 CXF 组件会填充 Exchange.AUTHENTICATION 标头。

127.5. 处理身份验证和授权错误

如果 SpringSecurityAuthorizationPolicy 中的身份验证或授权失败，则会抛出 CamelAuthorizationException。这可以通过 Camel 标准异常处理方法（如 Exception Clause）进行处理。CamelAuthorizationException 具有对策略 ID 的引用，该策略中解封异常，以便您可以根据策略以及例外类型处理错误。

```
<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>
```

127.6. SPRING BOOT AUTO-CONFIGURATION

组件没有 Spring Boot auto 配置选项。

第 128 章 SPRING WEBSERVICE

从 Camel 2.6 开始

支持生成者和消费者

Spring WS 组件允许您与 Spring Web Services 集成。它为访问 Web 服务以及创建您自己的合同第一 Web 服务提供客户端 - 端支持。

128.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 spring-ws 时，使用以下 Maven 依赖项来支持自动配置：

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

使用 BOM 获取版本。

```
<dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>com.redhat.camel.springboot.platform</groupId>
        <artifactId>camel-spring-boot-bom</artifactId>
        <version>${camel-spring-boot-version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
```

128.2. 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 元素的合格名称来映射 Web 服务请求的选项。
SOAPAction	用于根据消息标题中指定的 SOAP 操作来映射 Web 服务请求。
uri	要映射以特定 URI 为目标的 Web 服务请求。
xpathresult	用于根据 XPath **表达式** 对传入消息的评估来映射 Web 服务请求。评估的结果应与端点 URI 中指定的 XPath 结果匹配。
beanname	允许您引用 **org.apache.camel.component.spring.ws.bean.CamelEndpointDispatcher** 对象，以便与现有 (legacy) 端点映射 集成，如 **PayloadRootQNameEndpointMapping**,**SoapActionEndpointMapping**, 等

作为消费者，地址 应包含与指定 mapping-type 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。

128.3. 配置选项

Camel 组件在两个级别上配置：

- *组件级别*
- *端点级别*

128.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

128.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

128.4. 组件选项

Spring WebService 组件支持 4 个选项，如下所列。

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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
useGlobalSslContextParameters (security)	启用使用全局 SSL 上下文参数。	false	布尔值

128.5. 端点选项

Spring WebService 端点使用 URI 语法进行配置：

```
spring-ws:type:lookupKey:webServiceEndpointUri
```

以下是 path 和 query 参数：

128.5.1. 路径参数(4 参数)

Name	描述	默认值	类型

Name	描述	默认值	类型
type (consumer)	如果使用了端点映射，端点映射类型。使用 rootqname - 提供根据消息中包含的根元素的合格名称映射 Web 服务请求的选项。soapaction - 用于根据消息标头中指定的 SOAP 操作来映射 Web 服务请求。uri - 为映射以特定 URI. xpathresult - 用于根据针对传入邮件的 XPath 表达式评估的 Web 服务请求来映射 Web 服务请求。评估的结果应匹配端点 URI. beanname 中指定的 XPath 结果 - 允许您引用 org.apache.camel.component.spring.ws.bean.CamelEndpointDispatcher 对象，以便与 PayloadRootQNameEndpointMapping, SoapActionEndpointMapping 等现有（传统）端点映射集成。 Enum 值： • ROOT_QNAME • 操作 • TO • SOAP_ACTION • XPATHRESULT • URI • URI_PATH • BEANNAME		EndpointMappingType
lookupKey (consumer)	如果使用端点映射，端点映射密钥。		字符串
webServiceEndpointUri (producer)	用于制作者的默认 Web Service 端点 uri。		字符串

Name	描述	默认值	类型
expression (consumer)	用于 when 选项 type=xpathresult 的 XPath 表达式。然后需要配置这个选项。		字符串

128.5.2. 查询参数(21 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加时。		MessageFilter
messageIdStrategy (common)	提供自定义 MessageIdStrategy 来控制 WS-Addressing 唯一消息 ids 的生成选项。		MessageIdStrategy
endpointDispatcher (consumer)	Spring org.springframework.ws.server.endpoint.MessageEndpoint 将 Spring-WS 收到的消息分配给 Camel 端点，以与现有（传统）端点映射集成，如 PayloadRootQNameEndpointMapping, SoapActionEndpointMapping 等。		CamelEndpointDispatcher
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。这个 bean 由 MessageDispatcher 自动发现，用于根据端点上指定的特征（如 root QName、SOAP 操作等）将请求映射到 Camel 端点。		CamelSpringWSEndpointMapping

Name	描述	默认值	类型
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
allowResponseAttachmentOverride (producer)	通过来自实际服务层的附件覆盖跨外交换中的 soap 响应附件的选项。如果调用的服务在设为 true 时附加或重写 soap 附加，则允许在/out 消息附加中覆盖修改后的 soap attachments。	false	布尔值

Name	描述	默认值	类型
allowResponseHeaderOverride (producer)	使用来自实际服务层的标头信息覆盖/出站交换中的 soap 响应标头的选项。如果调用的服务在设为 true 时附加或重写 soap 标头，则允许在/out 消息标头中覆盖修改后的 soap 标头。	false	布尔值
faultAction (producer)	表示由方法提供的 faultAction 响应 WS-Addressing Fault Action 标头的值。如需了解更多详细信息，请参阅 org.springframework.ws.soap.addressing.server.annotation.Action 注解。		URI
faultTo (producer)	表示由方法提供的 faultAction 响应 WS-Addressing FaultTo 标头的值。如需了解更多详细信息，请参阅 org.springframework.ws.soap.addressing.server.annotation.Action 注解。		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。		WebServiceMessageFactory
messageSender (producer)	提供自定义 WebServiceMessageSender 的选项。例如，执行身份验证或使用替代传输。		WebServiceMessageSender
outputAction (producer)	表示由方法提供的响应 WS-Addressing Action 标头的值。如需了解更多详细信息，请参阅 org.springframework.ws.soap.addressing.server.annotation.Action 注解。		URI

Name	描述	默认值	类型
replyTo (producer)	表示由方法提供的 replyTo 响应 WS-Addressing ReplyTo 标头的值。如需了解更多详细信息，请参阅 org.springframework.ws.soap.addressing.server.annotation.Action 注解。		URI
SOAPAction (producer)	在访问远程 Web 服务时，SOAP 操作要包含在 SOAP 请求中。		字符串
timeout (producer)	在使用制作者调用 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 （）。此选项在使用内置消息发送器实现时工作：CommonsHttpMessageSender 和 HttpUrlConnectionMessageSender。除非自定义提供给组件的 Spring WS 配置选项，否则这些实现默认将用于基于 HTTP 的服务。如果您使用非标准发件人，则假设您将处理自己的超时配置。内置的消息发送者 HttpComponentsMessageSender 被认为是 CommonsHttpMessageSender，它已被弃用，请参阅 HttpComponentsMessageSender.setReadTimeout （）。		int
webServiceTemplate (producer)	提供自定义 WebServiceTemplate 的选项。这允许对客户端 Web 服务处理进行完全控制；例如添加自定义拦截器或指定故障解析器、邮件发送者或消息工厂。		WebServiceTemplate

Name	描述	默认值	类型
wsAddressingAction (producer)	在访问 Web 服务时包含 WS-Addressing 1.0 操作标头。To 标头设置为端点 URI （默认 Spring-WS 行为）中指定的 Web 服务的地址。		URI
lazyStartProducer (producer (advanced))	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
sslContextParameters (security)	使用 SSLContextParameters 配置安全性。		SSLContextParameters

128.6. 消息标头

***Spring WebService** 组件支持 7 个消息标头，如下所列：*

Name	描述	默认值	类型
CamelSpringWebserviceEndpointUri (producer) 常量：**SPRING_WS_ENDPOINT_URI**	端点 URI。		字符串

Name	描述	默认值	类型
CamelSpringWebserviceSoapAction (producer) 常量 ：**SPRING_WS_SOAP_ACTION**	在访问远程 Web 服务时，SOAP 操作要包含在 SOAP 请求中。		字符串
CamelSpringWebserviceSoapHeader (producer) 常量 ：**SPRING_WS_SOAP_HEADER**	soap 标头源。		Source
CamelSpringWebserviceAddressingAction (producer) 常数 ：**SPRING_WS_ADDRESSING_ACTION**	在访问 Web 服务时包含 WS-Addressing 1.0 操作标头。To 标头设置为端点 URI （默认 Spring-WS 行为）中指定的 Web 服务的地址。		URI
CamelSpringWebserviceAddressingFaultTo (producer) 恒定：**SPRING_WS_ADDRESSING_PRODUCER_FAULT_TO**	表示由方法提供的 faultAction 响应 WS-Addressing FaultTo 标头的值。如需了解更多详细信息，请参阅 org.springframework.ws.soap.addressing.server.annotation.Action 注解。		URI
CamelSpringWebserviceAddressingReplyTo (producer) 恒定：**SPRING_WS_ADDRESSING_PRODUCER_REPLY_TO**	表示由方法提供的 replyTo 响应 WS-Addressing ReplyTo 标头的值。如需了解更多详细信息，请参阅 org.springframework.ws.soap.addressing.server.annotation.Action 注解。		URI
breadcrumbId (consumer) 常数 ：**BREADCRUMB_ID**	面包屑导航栏 ID.		字符串

128.7. 访问 WEB 服务

要在 http://foo.com/bar 调用 web 服务，只需定义一个路由：

```
from("direct:example").to("spring-ws:http://foo.com/bar")
```

并发送消息：

```
template.requestBody("direct:example", "<foobar xmlns=\"http://foo.com\"><msg>test
message</msg></foobar>");
```

如果您要调用 SOAP 服务，则不得包含 SOAP 标签。Spring-WS 执行 XML-to-SOAP marshaling。

128.8. 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")
```

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");
```

128.9. 使用 SOAP 标头

在向 spring-ws 端点发送消息时，您可以提供 SOAP 标头作为 Camel 消息标头。例如，在 String 中给出以下 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 使用者还使用 SOAP 标头增强 Camel 消息。

有关示例，请参阅这个 单元测试。

128.10. 标头和附加传播

Spring WS Camel 支持将标头和附件传播到 Spring-WS WebServiceMessage 响应中。端点使用带有 MessageFilter 的"hook"（默认实现由 BasicMessageFilter 提供）将交换标头和附加传播到 WebServiceMessage 响应。

```
exchange.getOut().getHeaders().put("myCustom","myHeaderValue")
exchange.getIn().addAttachment("myAttachment", new DataHandler(...))
```

如果管道中的 Exchange 标头包含文本，它会在 soap 标头中生成 Qname (key)=value 属性。您必须直接创建一个 QName 类，并将任何键放在标头中。

128.11. 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.HeaderTransformationMessageFilte
r">
    <constructor-arg index="0" value="org/apache/camel/component/spring/ws/soap-header-
transform.xslt"/>
</bean>
```

使用在 camel 端点中定义的 bead

```
<route>
    <from uri="direct:stockQuoteWebserviceHeaderTransformation"/>
    <to uri="spring-ws:http://localhost?
```

```
webServiceTemplate=#webServiceTemplate&amp;soapAction=http://www.stockquotes.edu/G
etQuote&amp;messageFilter=#headerTransformationFilter"/>
</route>
```

128.12. 如何使用 MTOM ATTACHMENTS

BasicMessageFilter 提供 Apache Axiom 的所有所需信息，以便生成 MTOM 消息。如果要在 Apache Axiom 中使用 Apache Camel Spring WS，以下示例： - 定义 messageFactory，如下所示，Spring-WS 使用优化附件填充 SOAP 消息，并通过 MTOM 策略填充您的 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>
```

- 将附加添加到管道中，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，如下所示：

```
from("direct:send")
.process(new Attachement())
.to("spring-ws:http://localhost:8089/mySoapService?
soapAction=mySoap&messageFactory=axiomMessageFactory");
```

- 您的制作者现在生成 MTOM 信息，其中包含优化的附件。

128.13. 自定义标头和附加过滤

如果您需要提供标头或附加的自定义处理，请扩展现有的 BasicMessageFilter，并覆盖适当的方法或写入 MessageFilter 接口的品牌新实施。
要使用您的自定义过滤器，请在 spring 上下文中添加全局或本地消息过滤器。

- A）提供所有 Spring-WS 端点的全局自定义过滤器

```
<bean id="messageFilter" class="your.domain.myMessageFiler" scope="singleton" />
```

或者

- 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 中的默认 MessageFilter 实现中覆盖以下方法：

```
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*/ }
```

128.14. 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"/>
   </property>
</bean>
```

128.15. 公开 WEB 服务

要使用此组件公开 Web 服务，您必须首先设置一个 MessageDispatcher，以便在 Spring XML 文件中查找端点映射。如果要在 servlet 容器中运行，则必须使用 web.xml 中配置的 MessageDispatcherServlet。

默认情况下，MessageDispatcherServlet 将查找名为 /WEB-INF/spring-ws-servlet.xml 的 Spring XML。要将 Camel 与 Spring-WS 搭配使用，该 XML 文件中唯一的强制 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 Writing Contract-First Web Services。基本段落 3.6 "Implementing the Endpoint" 由此组件处理（特别是段落 3.6.2"向 Endpoint 进行"恢复消息"是 CamelEndpointMapping 所在的位置。请参阅 Camel 分发中包含的 Spring Web 服务示例。

128.16. 路由中的端点映射

使用 XML 配置原位，您现在可以使用 Camel 的 DSL 定义端点处理哪些 Web 服务请求：

以下路由接收 http://example.com/ 命名空间中具有名为"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)
```

以下路由接收包含 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)
```

以下路由收到包含消息（和默认命名空间） 中的元素 <foobar>abc </foobar> 的请求。

```
from("spring-ws:xpathresult:abc?
expression=//foobar&endpointMapping=#endpointMapping")
.convertBodyTo(String.class).to(mock:example)
```

128.16.1. 备用配置，使用现有端点映射

对于带有 mapping-type beanname 一个类型为 CamelEndpointDispatcher 的端点，Registry/ApplicationContext 中需要带有对应名称的 bean。此 bean 充当 Camel 端点和现有 端点映射 之间的桥接，如 PayloadRootQNameEndpointMapping。

beanname mapping-type 的使用主要用于（传统）您已使用 Spring-WS 并在 Spring XML 文件中定义端点映射。beanname mapping-type 允许您将 Camel 路由放入现有的端点映射中。从开始开始时，您必须将端点映射定义为 Camel URI （如上图所示），因为它需要较少的配置，且更加表达。您还可以在注解中使用 vanilla Spring-WS。

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.PayloadRootQNameEndpointMappi
ng">
  <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" />
```

128.17. POJO (UN) MARSHALLING

Camel 的可插拔数据格式支持使用 JAXB、XStream、JibX、Castor 和 XMLBeans 等库进行 pojo/xml marshall 支持。您可以在路由中使用这些数据格式，来向 Web 服务发送和接收 pojo。

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 POJOs unmarshal XML 请求，并将响应消息发送到 XML：

```
from("spring-ws:rootqname:{http://example.com/}GetFoo?
endpointMapping=#endpointMapping").unmarshal(jaxb)
.to("mock:example").marshal(jaxb);
```

128.18. SPRING BOOT AUTO-CONFIGURATION

组件支持下面列出的 5 个选项。

Name	描述	默认值	类型
camel.component.spring-ws.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.spring-ws.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.spring-ws.enabled	是否启用 spring-ws 组件的自动配置。这默认是启用的。		布尔值
camel.component.spring-ws.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。通过延迟启动，启动失败可以在路由消息期间使用 Camel 的路由错误处理程序进行处理。处理第一个消息后，创建并启动制作者需要花费时间和延长处理的总处理时间。	false	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

第 129 章 SQL

支持生成者和消费者

SQL 组件允许您使用 JDBC 查询处理数据库。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。

此组件使用 spring-jdbc 在 scenes 后面进行实际 SQL 处理。

SQL 组件还支持：

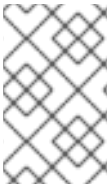
- 适用于 Idempotent Consumer EIP 模式的基于 JDBC 的存储库。请参见以下内容。
- 基于 JDBC 的存储库，用于聚合器 EIP 模式。请参见以下内容。

129.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 sql 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-sql-starter</artifactId>
</dependency>
```

129.2. URI 格式



注意

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]
```

您可以使用以下方法使用命名参数："#name_of_the_parameter' 样式，如下所示：

```
sql:select * from table where id=:#myId order by name[?options]
```

使用命名参数时，Camel 将在给定优先级中查找名称：

1. 来自消息正文（如果其 java.util.Map）
2. 来自消息标头

如果无法解析命名参数，则会抛出异常。

您可以使用简单表达式作为参数，如下所示：

```
sql:select * from table where id=:#${exchangeProperty.myId} order by name[?options]
```

请注意，标准 ? 符号表示 SQL 查询的参数被替换为 # 符号，因为 ? 符号用于指定端点的选项。可以基于端点配置 ? 符号替换。

您可以将 SQL 查询外部化到类路径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，只是一个纯文本

```
-- this is a comment
select *
from table
where
  id = :#${exchangeProperty.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 也可使用 - dash 行等注释。

129.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

129.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

129.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

129.4. 组件选项

***SQL** 组件支持 5 个选项，如下所列。*

Name	描述	默认值	类型
DataSource (common)	**Autowired** 设置用于与数据库通信的数据来源。		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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
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。这个选项默认为 true。	true	布尔值

129.5. 端点选项

***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：

129.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
query (common)	**必需** 设置要执行的 SQL 查询。您可以使用 file: 或 classpath: 作为前缀来外部化查询，并指定文件的位置。		字符串

129.5.2. 查询参数(45 参数)

Name	描述	默认值	类型
allowNamedParameters (common)	是否在查询中使用命名参数。	true	布尔值
DataSource (common)	**Autowired** 设置 DataSource，用来在端点级别与数据库通信。		DataSource
outputClass (common)	指定在 outputType=SelectOne 时用作转换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，则该值将用作标头的名称，以存储查询结果，并保留原始消息正文。		字符串
outputType (common)	将消费者或制作者的输出设置为 SelectList 作为 Map 列表，或选择以以下方式使用单个 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）将返回 Long 对象。如果查询有多个列，则它将返回该结果的 Map。c）如果设置了 outputClass，则输出为：然后，它将通过调用与列名称匹配的所有 setters 将查询结果转换为 Java bean 对象。它将假设您的类具有默认的构造器来创建实例。d）如果查询导致多个行，它会抛出一个非唯一结果异常。StreamList 使用 Iterator 来流传输查询的结果。这可以与流模式的 Splitter EIP 一起使用，以流传输方式处理 ResultSet。 Enum 值： • SelectOne • SelectList • StreamList	Select List	SqlOutputType

Name	描述	默认值	类型
分隔符 (common)	从消息正文获取参数值时使用的分隔符（如果正文是 String 类型），以在 # 占位符插入。注意如果您使用命名参数，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)	在使用 onConsume 时，设置要验证的预期更新数。	-1	int
maxMessagesPerPoll (consumer)	设置要轮询的最大消息数。		int
onConsume (consumer)	在处理完每行后，可以执行此查询，如果交换成功处理，例如将行标记为已处理。查询可以具有参数。		字符串
onConsumeBatchComplete (consumer)	处理整个批处理后，可以执行此查询来批量更新行等。查询不能有参数。		字符串
onConsumeFailed (consumer)	在处理完每行后，可以执行此查询，如果 Exchange 失败，例如，将行标记为失败。查询可以具有参数。		字符串
routeEmptyResultSet (consumer)	设置是否应该允许将空结果集发送到下一个跃点。默认为false。因此，空结果集会被过滤掉。	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，则消费者中断处理任何进一步的交换，从而导致回滚延迟。	false	布尔值
useIterator (consumer)	设置结果集应如何传送到路由。将 delivery 表示作为列表或单个对象。默认为 true。	true	布尔值

Name	描述	默认值	类型
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
processingStrategy (consumer (advanced))	允许插件使用自定义 org.apache.camel.component.sql.SqlProcessingStrategy 在消费者处理 rows/batch 时执行查询。		SqlProcessingStrategy
batch (producer)	启用或禁用批处理模式。	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
noop (producer)	如果设置，将忽略 SQL 查询的结果，并使用现有的 IN 消息作为 OUT 消息进行处理。	false	布尔值
useMessageBodyForSql (producer)	是否使用消息正文作为 SQL，然后对参数进行标头。如果启用了这个选项，则不会使用 uri 中的 SQL。请注意，消息正文中的查询参数由问号而不是 # 符号表示。	false	布尔值
alwaysPopulateStatement (advanced)	如果启用，则来自 org.apache.camel.component.sql.SqlPrepareStatementStrategy 的 populateStatement 方法总是被调用，如果没有预期的参数可以被准备。当这是 false 时，只有在要设置 1 个或更多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/headers。	false	布尔值

Name	描述	默认值	类型
parametersCount (advanced)	如果设置大于零，则 Camel 将使用此 count 值来替换，而不是通过 JDBC 元数据 API 进行查询。如果 JDBC 供应商无法返回正确的参数数，则这很有用，然后用户可能会覆盖。		int
占位符 (advanced)	指定将在 SQL 查询中替换的字符。请注意，它是简单的 String.replaceAll （）操作，且不会涉及 SQL 解析（加引号的字符串也会更改）。	#	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 prepared 语句的准备。		SqlPrepareStatementStrategy
templateOptions (advanced)	使用 Map 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。	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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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

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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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

129.6. 消息正文的处理

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（其中每个查询参数由端点 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，则转换会导致转换过程只迭代一个对象，即正文本身。

例如，如果消息正文是 java.util.List 的一个实例，则列表中的第一个项将被替换在 SQL 查询的第一个出现 # 的位置，则列表中的第二个项目将替换为 #，以此类推。

如果 batch 设为 true，则对入站消息正文的解释稍有变化 - 而非参数迭代器，则组件需要一个包含参数迭代器的迭代器；外迭代器的大小决定了批处理大小。

您可以使用选项 useMessageBodyForSql，允许使用消息正文作为 SQL 语句，然后在带有键 SqlConstants.SQL_PARAMETERS 的标头中提供 SQL 参数。这允许 SQL 组件动态地工作，因为 SQL 查询来自消息正文。使用模板（如 Velocity、freemarker）进行条件处理，例如，根据查询参数的存在来包含或排除 where 子句。

129.7. 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 正文返回，因为 更新操作 仅设置为标头，永远不会设置为正文。

注意

有关 update 操作的更多信息，请参阅标头值。https://camel.apache.org/components/4.0.x/sql-component.html#sql-component-header-values

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添加标头的替代选择，它提供了一个简洁的语法，用于查询序列或某些其他小值到标头中。最好将 outputHeader 和 outputType 搭配使用：

```
from("jms:order.inbox")
    .to("sql:select order_seq.nextval from dual?
outputHeader=OrderId&outputType=SelectOne")
    .to("jms:order.booking");
```

129.8. 使用 STREAMLIST

制作者支持使用 outputType=StreamList 来流传输查询的输出。这允许以流方式处理数据，例如，Splitter EIP 可一次处理每行，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();
```

129.9. 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	更新操作 **更新** 的行数，以 **整数** 对象返回。使用 outputType=StreamList 时不提供此标头。
CamelSqlRowCount	为 **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	要执行的查询。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 # 符号表示

在执行 插入 操作时，SQL 组件使用生成的键存储行，并将这些行的数量存储在以下消息标头中：

标头	描述
CamelSqlGeneratedKeysRowCount	包含生成的键的标头中的行数。
CamelSqlGeneratedKeyRows	包含生成的密钥的行（键映射列表）。

129.10. 生成的密钥

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，生成的密钥将以标头形式提供，其中包含上表中列出的键。

要指定应检索哪些生成的列，将标头 CamelSqlGeneratedColumns 设置为 String[] 或 int[]，分别代表列名称或索引。有些数据库需要此功能，如 Oracle。如果驱动程序无法正确确定参数数量，则可能需要使用 parametersCount 选项。

您可以在 单元测试 中看到更多详细信息。

129.11. DATASOURCE

您可以直接在 URI 中设置对 DataSource 的引用：

```
sql:select * from table where id=# order by name?dataSource=#myDS
```

129.12. 使用命名参数

在以下给定路由中，我们希望从 projects 表中获取所有项目。请注意，SQL 查询有 2 个命名的参数：：#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，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")
```

虽然消息正文是 java.util.Map，则 named 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")
```

129.13. 在制作者中使用表达式参数

在下面的给定路由中，我们希望从数据库获取所有项目。它使用交换的正文来定义许可证，并使用属性值作为第二个参数。

```
from("direct:projects")
```

```
  .setBody(constant("ASF"))
  .setProperty("min", constant(123))
  .to("sql:select * from projects where license = :#${body} and id > :#${exchangeProperty.min}
order by id")
```

129.13.1. 在消费者中使用表达式参数

当将 SQL 组件用作消费者时，您现在可以使用表达式参数（简单语言）来构建动态查询参数，如调用 bean 的方法来检索 id、日期或内容。

例如，在以下示例中，我们在 bean myIdGenerator 上调用 nextId 方法：

```
from("sql:select * from projects where id = :#${bean:myIdGenerator.nextId}")
   .to("mock:result");
```

bean 有以下方法：

```
public static class MyIdGenerator {

   private int id = 1;

   public int nextId() {
      return id++;
   }
```

请注意，没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。

129.14. 使用带有动态值的 IN 查询

SQL producer 允许将 SQL 查询与 IN 语句搭配使用，其中 IN 值是动态计算的。例如，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 添加前缀：
- 在参数外添加 ( )

示例说明了这一点。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 查询可以使用带有动态值的键名称的标头，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");
```

也可以在端点中指定查询，而不是外部化（注意外部化可以更轻松地维护 SQL 查询）

```
from("direct:query")
    .to("sql:select * from projects where project in (:#in:names) order by id")
    .to("log:query")
    .to("mock:query");
```

129.15. 使用基于 JDBC 的幂等存储库

在本节中，我们将使用基于 JDBC 的幂等存储库。

注意

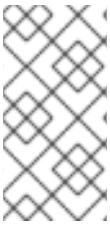
抽象类
有一个抽象类
org.apache.camel.processor.idempotent.jdbc.AbstractJdbcMessageIdRepository，
您可以扩展来构建自定义 JDBC 幂等存储库。

首先，我们必须创建将由幂等存储库使用的数据库表。我们使用以下模式：

```
CREATE TABLE CAMEL_MESSAGEPROCESSED ( processorName VARCHAR(255),
messageId VARCHAR(100) )
```

我们添加了 createdAt 列：

```
CREATE TABLE CAMEL_MESSAGEPROCESSED ( processorName VARCHAR(255),
messageId VARCHAR(100), createdAt TIMESTAMP )
```



注意

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。

在使用并发消费者时，对列 processorName 和 messageId 创建唯一约束至关重要。由于此约束的语法与数据库与数据库不同，因此这里不会显示它。

129.15.1. 自定义 JDBC 幂等性存储库

您有几个选项来调优 org.apache.camel.processor.idempotent.jdbc.JdbcMessageIdRepository 以满足您的需要：

参数	默认值	描述
createTableIfNotExists	true	定义 Camel 是否是否应该尝试创建表（如果不存在）。
tableName	CAMEL_MESSAGEPROCESSED	使用自定义表名称而不是默认名称：CAMEL_MESSAGEPROCESSED。

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。它必须抛出异常，以指示该表不存在。
createString	CREATE TABLE CAMEL_MESSAGEPROCESSED (processorName VARCHAR (255), messageId VARCHAR (100), createdAt TIMESTAMP)	用于创建表的声明。
queryString	SELECT COUNT(*) FROM CAMEL_MESSAGEPROCESSED WHERE processorName = ?AND messageId = ?	用于找出存储库中是否已存在消息的查询（结果不等于 '0'）。它采用两个参数。第一个是处理器名称 (**String**)，第二个是消息 id (**String**)。
insertString	INSERT INTO CAMEL_MESSAGEPROCESSED (processorName, messageId, createdAt) VALUES (?, ?, ?)	用于将条目添加到表中的语句。它采用三个参数。第一个是处理器名称(**String**)，第二个是消息 id (**String**)，第三个是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声明。它取两个参数。第一个是处理器名称(**String**)，第二个是消息 id (**String**)。

选项 ***tableName 可用于使用默认 SQL 查询，但使用不同的表名称。但是，如果要自定义 SQL 查询，您可以单独配置每个查询。***

129.15.2. orphan Lock aware Jdbc IdempotentRepository

org.apache.camel.processor.idempotent.jdbc.JdbcMessageIdRepository 的一个限制是它不会处理由 JVM 崩溃或非安全关闭导致的孤立锁定。如果您需要处理孤立锁定，则使用 org.apache.camel.processor.idempotent.jdbc.JdbcOrphanLockAwareIdempotentRepository。此存储库跟踪应用实例持有的锁定。对于持有的每个锁定，应用程序会将保留信号发送到锁定存储库，从而导致使用 current Timestamp 更新 createdAt 列。当应用程序实例尝试获取锁定时，如果存在 3 个可能：

- 锁定条目不存在，然后使用 JdbcMessageIdRepository 的基本实现提供锁定。

- lock already exists 和 createdAt < System.currentTimeMillis （）- lockMaxAgeMillis. 在这种情况下，假设一个活跃的实例具有锁定，并且未向请求锁定的新实例提供锁定

- lock already exists 和 createdAt > = System.currentTimeMillis （）- lockMaxAgeMillis. 在这种情况下，假设没有活跃的实例，该实例没有锁定，并且为请求实例提供了锁定。背后的原因是，如果原始实例已锁定，如果仍然在运行，它将使用其 keepAlive 机制更新 Timestamp on createdAt

这个软件仓库有两个额外的配置参数

参数	描述
lockMaxAgeMillis	这指的是锁定被视为孤立的持续时间，例如，如果 currentTimestamp - createdAt >= lockMaxAgeMillis，则锁定会被孤立。
lockKeepAliveIntervalMillis	为 createAt Timestamp 列进行保留 alive 更新的频率。

129.15.3. Caching Jdbc IdempotentRepository

有些 SQL 实现不会基于每个查询快速。JdbcMessageIdRepository 实现在 SQL 事务中单独执行其幂等检查。检查 100 个密钥可能需要几分钟。JdbcCachedMessageIdRepository preloads a in-memory 缓存以整个键列表开头。然后，在传递到原始实施之前首先检查此缓存。

与所有缓存实现一样，应该考虑陈旧数据和您的特定用途。

129.16. 使用基于 JDBC 的聚合存储库

Jdbc AggregationRepository 是一个聚合的Repository，它即时保留聚合的消息。这样可确保您不会松散消息，因为默认聚合器将仅使用内存 AggregationRepository。JdbcAggregationRepository 允许与 Camel 一起为聚合器提供持久的支持。

只有成功处理 Exchange 时，才会在 AggregationRepository 上调用 confirm 方法时将其标记为完成。这意味着，如果同一 Exchange 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 maximumRedeliveries 命中时发送交换的位置。

您可以在 camel-sql 的单元测试中看到一些示例，如 JdbcAggregateRecoverDeadLetterChannelTest.java

129.16.1. 数据库

要正常运行，每个聚合器使用两个表：聚合并完成一个。按照惯例，完成的名称与带有 "_COMPLETED" 后缀的聚合名称相同。名称必须在带有 RepositoryName 属性的 Spring bean 中进行配置。在以下示例中，将使用聚合。

两个表的表结构定义相同：如果一个 String 值都用作键(id)，而 Blob 包含字节阵列中的交换序列化。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，包含组件用来聚合消息的关联 Id。在完成的表中，id 保存在对应的 blob 字段中存储的交换的 id。

以下是用于创建表的 SQL 查询，只需将 "aggregation" 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,
 version BIGINT NOT NULL,
 constraint aggregation_pk PRIMARY KEY (id)
);
CREATE TABLE aggregation_completed (
 id varchar(255) NOT NULL,
 exchange blob NOT NULL,
 version BIGINT NOT NULL,
 constraint aggregation_completed_pk PRIMARY KEY (id)
);
```

129.17. 将正文和标头存储为文本

您可以将 JdbcAggregationRepository 配置为存储消息正文，并在单独的列中选择(ed)标头作为 String。例如，要存储正文，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,
 version BIGINT 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,
 version BIGINT NOT NULL,
 body varchar(1000),
 companyName varchar(1000),
 accountName varchar(1000),
 constraint aggregationRepo3_completed_pk PRIMARY KEY (id)
);
```

然后，配置存储库以启用此行为，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>
```

129.17.1. codec (Serialization)

由于它们可以包含任何类型的有效负载，因此交换的设计不适合。它转换为存储在数据库 BLOB 字段中的字节数组。所有这些转换都由 JdbcCodec 类处理。代码的一个详情需要注意：ClassLoadingAwareObjectInputStream。

ClassLoadingAwareObjectInputStream 已从 Apache ActiveMQ 项目中重复使用。它打包了一个 ObjectInputStream，并将其与 ContextClassLoader 而不是 currentThread 一起使用。好处是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。

129.17.2. Transactions

需要 Spring PlatformTransactionManager 来编配事务。

129.17.2.1. Service (Start/Stop)

start 方法验证数据库的连接并存在所需的表。如果出现错误，它将在启动过程中失败。

129.17.3. 聚合器配置

根据目标环境，聚合器可能需要一些配置。如您已经知道，每个聚合器应具有自己的存储库（在数据库中创建对应的表对）和数据源。如果默认的 lobHandler 没有根据您的数据库系统进行调整，则可以将其与 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>
```

129.17.4. Optimistic locking

您可以在集群环境中开启 optimisticLocking 并使用基于 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 驱动程序将抛出特定于供应商的异常，Jdb cAggregationRepository 可以响应。要了解将 JDBC 驱动程序中的异常视为 optimistick 锁定错误，我们需要一个映射器才能执行此操作。因此，有一个 org.apache.camel.processor.aggregate.jdbc.JdbcOptimisticLockingExceptionMapper 允许您实现自定义逻辑（如果需要）。有一个默认的 org.apache.camel.processor.aggregate.jdbc.DefaultJdbcOptimisticLockingExceptionMapper，它可以正常工作：

以下检查已完成：

- 如果导致异常是 SQLException，则如果以 23 开始，则会检查 SQLState。
- 如果原因异常是 DataIntegrityViolationException

- 如果原因异常类名称的名称中包含 "ConstraintViolation"。

- 如果配置了任何类名称，则对 FQN 类名称的可选检查都匹配。

您还可以添加 FQN 类名称，如果任何原因异常（或任何嵌套）等于任何 FQN 类名称，则其 optimistick 锁定错误。

例如，我们从 JDBC 供应商定义了 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>
```

129.17.5. 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，当在其自身使用 < transacted /> 的路由中使用 JdbcAggregationRepository 且使用了通用平台TransactionManager 时，可能需要配置 JdbcAggregationRepository 中的事务模板使用的 传播行为。

以下是进行该操作的方法：

```
<bean id="repo"
class="org.apache.camel.processor.aggregate.jdbc.JdbcAggregationRepository">
  <property name="propagationBehaviorName" value="PROPAGATION_NESTED" />
```

```
</bean>
```

propagation 由 org.springframework.transaction.TransactionDefinition 接口的常量指定，因此 propagationBehaviorName 方便地使用恒定名称。

129.17.6. PostgreSQL 问题单

有特殊的数据库可能会导致 JdbcAggregationRepository 使用的最佳锁定问题。在数据完整性违反异常时，PostgreSQL 会将连接标记为无效(SQLState 23505 之一)。这使得连接在嵌套的事务中有效无法使用。详情可在 文档 中找到。

org.apache.camel.processor.aggregate.jdbc.PostgresAggregationRepository 扩展 JdbcAggregationRepository，并使用特殊的 INSERT .ON CONFLICT .. 语句以提供最佳锁定行为。

此语句（具有默认聚合表定义）：

```
INSERT INTO aggregation (id, exchange) values (?, ?) ON CONFLICT DO NOTHING
```

详情请查看 PostgreSQL 文档。

当使用此 子句时，java.sql.PreparedStatement.executeUpdate （） 调用返回 0， 而不是用 SQLState=23505 丢弃 SQLException。进一步处理与通用 JdbcAggregationRepository 完全相同，但没有将 PostgreSQL 连接标记为无效。

129.18. CAMEL SQL STARTER

spring-boot 用户提供了一个初学者模块。在使用初学者时，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.springboot</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。

129.19. 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.sql-stored.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.sql-stored.enabled	是否启用 sql-stored 组件的自动配置。这默认是启用的。		布尔值
camel.component.sql-stored.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.sql.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.sql.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.sql.enabled	是否启用 sql 组件的自动配置。这默认是启用的。		布尔值
camel.component.sql.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.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。这个选项默认为 true。	true	布尔值

第 130 章 SQL 存储的步骤

从 Camel 2.17 开始

仅支持生成者

SQL Stored 组件允许您使用 JDBC 存储的步骤查询处理数据库。此组件是 SQL 组件的扩展，但专门调用存储的流程。

此组件使用 spring-jdbc 在 scenes 后面进行实际 SQL 处理。

130.1. 依赖项

当在 Camel Spring Boot 中使用 camel-sql 时，请将以下 Maven 依赖项添加到 pom.xml 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-sql-starter</artifactId>
</dependency>
```

130.2. 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或类路径的外部文件中的模板，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
)
```

130.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

130.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

130.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

130.4. 组件选项

SQL Stored 流程组件支持 3 个选项，如下所列。

Name	描述	默认值	类型
datasource (producer)	**Autowired** 设置用于与数据库通信的数据来源。		DataSource
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

130.5. 端点选项

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 ：

130.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
template (producer)	**必需** 设置要执行的存储的步骤模板。您可以使用 file: 或 classpath: 作为前缀来外部化模板，并指定文件的位置。		字符串

130.5.2. 查询参数(8 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
function (producer)	此调用是否为函数。	false	布尔值
noop (producer)	如果设置，将忽略存储的流程模板的结果，并使用现有的 IN 消息作为处理持续处理的 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)	是否使用消息正文作为存储的步骤模板，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
lazyStartProducer (producer (advanced))	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
templateOptions (advanced)	使用 Map 中的键/值配置 Spring JdbcTemplate。		Map

130.6. 消息标头

SQL Stored 流程组件支持 3 个消息标头，如下所列：

Name	描述	默认值	类型
CamelSqlStoredTemplate (producer) 常数：SQL_STORED_TEMPLATE	模板。		字符串
CamelSqlStoredParameters (producer) 常量：SQL_STORED_PARAMETERS	参数。		lerator
CamelSqlStoredUpdateCount (producer) 常量：SQL_STORED_UPDATE_COUNT	更新计数。		整数

130.7. 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。存储的流程的名称，然后是用括号括起的参数。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。在这个示例中，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，存储的流程可以声明为：

```
CREATE PROCEDURE STOREDSAMPLE(VALUE1 INTEGER, VALUE2 INTEGER, INOUT RESULT1 INTEGER, OUT RESULT2 INTEGER)
```

130.7.1. IN 参数

IN 参数采用以空格号分隔的四个部分：参数名称、SQL 类型（缩放）、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 的字段 INTEGER 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，简单表达式 ${header.val} 表示从标头 val 读取参数值。标头位置表达式 :#val 的作用相同。

```
<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，它的 SQL 类型是从类 org.example.Types 和 scale 范围的字段 INTEGER 读取的。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 上的 URI 相同，但 SQL-type 为 100，类型名称为 mytypename。

实际调用将使用 org.springframework.jdbc.core.SqlParameter 来完成。

130.7.2. OUT 参数

OUT 参数的工作方式类似，并包含三个部分：SQL 类型（带有 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，结果将是标头 outheader1。

```
<to uri="sql-stored:MYFUNC(OUT org.example.Types.NUMERIC(10) 'mytype' outheader1)"/>
```

这与前一个相同，但类型名称为 mytype。

实际调用将使用 org.springframework.jdbc.core.SqlOutParameter 来完成。

130.7.3. 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 的名称 定义了 SQL 参数名称和结果标头名称。

```
<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。

130.7.4. 查询超时

您可以在用于查询处理的语句上配置查询超时（通过 template.queryTimeout），如下所示：

```
<to uri="sql-stored:MYFUNC(INOUT DECIMAL(10) ${headers.inheader} outheader)?
template.queryTimeout=5000"/>
```

这在事务中执行时剩余的事务超时覆盖，该超时在事务级别指定超时。

130.8. CAMEL SQL STARTER

spring 引导用户可以使用一个初学者模块。在使用初学者时，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.springboot</groupId>
  <artifactId>camel-sql-starter</artifactId>
</dependency>

<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-jdbc</artifactId>
  <version>${spring-boot-version}</version>
</dependency>
```

如果需要，您还可以包含特定的数据库驱动程序。

130.9. 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.sql-stored.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.sql-stored.enabled	是否启用 sql-stored 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.sql-stored.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.sql.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.sql.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 Camel 消费者试图获取传入的消息或类似信息时，会出现异常（如果可能），现在将被作为消息进行处理，并由路由 Error Handler 处理。重要：只有在第三方组件允许 Camel 抛出异常时，才能警报这一点。有些组件仅在内部处理，因此无法 bridgeErrorHandler。在其他情况下，我们可能会将 Camel 组件提高到第三方组件中的 hook，并使其可能用于将来的版本。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.sql.enabled	是否启用 sql 组件的自动配置。这默认是启用的。		布尔值
camel.component.sql.health-check-consumer-enabled	用于从这个组件启用或禁用所有基于消费者的健康检查。	true	布尔值
camel.component.sql.health-check-producer-enabled	用于从此组件启用或禁用所有基于制作者的健康检查。注意：默认情况下，Camel 禁用了所有基于健康检查的制作者。您可以通过设置 camel.health.producersEnabled=true 来全局打开制作者检查。	true	布尔值
camel.component.sql.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

Name	描述	默认值	类型
camel.component.sql.row-mapper-factory	创建 RowMapper 工厂。选项是 org.apache.camel.component.sql.RowMapperFactory 类型。		RowMapperFactory
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。这个选项默认为 true。	true	布尔值

第 131 章 SSH

Since Camel 2.10

支持生成者和消费者

SSH 组件可让您访问 SSH 服务器，以便您可以发送 SSH 命令并处理响应。

131.1. 依赖项

当在 Camel Spring Boot 中使用 camel-ssh 时，请将以下 Maven 依赖项添加到 pom.xml 中，以支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-ssh-starter</artifactId>
</dependency>
```

131.2. URI 格式

```
ssh:[username[:password]@]host[:port][?options]
```

131.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

131.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

131.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

131.4. 组件选项

SSH 组件支持 25 个选项，如下所列。

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 knownHosts 属性时，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)	设置建立远程 SSH 服务器连接时要等待的超时时间（毫秒）。默认值为 30000 毫秒。	30000	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 Camel 消费者试图获取传入的消息或类似信息时，会出现异常（如果可能），现在将被作为消息进行处理，并由路由 Error Handler 处理。重要：只有在第三方组件允许 Camel 抛出异常时，才能警报这一点。有些组件仅在内部处理，因此无法 bridgeErrorHandler。在其他情况下，我们可能会将 Camel 组件提高到第三方组件中的 hook，并使其可能用于将来的版本。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)	设置在每次轮询周期期间要发送到远程 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://...)，您可能需要使用新行结束命令，且必须 URL 编码的 %0A。		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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
channelType (advanced)	设置要在命令执行过程中传递给频道的频道类型。默认为 exec。	exec	字符串
ClientBuilder (advanced)	生成者或消费者使用的 autowired ClientBuilder 实例，以创建新的 SshClient。		ClientBuilder
compressions (advanced)	是否要使用压缩，如果有。		字符串
configuration (advanced)	组件配置.		SshConfiguration
shellPrompt (advanced)	在命令执行后读取响应时，将 shellPrompt 设置为被丢弃。		字符串

Name	描述	默认值	类型
sleepForShellPrompt (advanced)	设置从 shell 提示符读取响应的睡眠周期（以毫秒为单位）。默认值为 100 毫秒。	100	long
healthCheckConsumerEnabled (health)	用于从这个组件启用或禁用所有基于消费者的健康检查。	true	布尔值
healthCheckProducerEnabled (health)	用于从此组件启用或禁用所有基于制作者的健康检查。  **注意** 默认情况下，所有基于制作者的健康检查都被禁用。您可以通过设置 **camel.health.producersEnabled=true** 来全局打开制作者检查。	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** 来解析基于文件的证书，并依赖于 keyType 设置。		字符串
certResourcePassword (security)	如果 certResource 是加密密钥，则设置要在加载 certResource 中使用的密码。		字符串
密码 （安全）	以逗号分隔的允许/支持的密码列表（按首选顺序排列）。		字符串
KEX （安全性）	以逗号分隔的允许/支持的密钥交换算法列表，按首选顺序排列。		字符串
keyPairProvider (security)	设置使用证书连接到远程 SSH 服务器时要使用的 KeyPairProvider 引用。		KeyPairProvider
keytype (security)	设置在身份验证过程中传递给 KeyPairProvider 的密钥类型。KeyPairProvider.loadKey (...)将传递这个值。在 Camel 3.0.0 / 2.25.0 中，默认情况下 Camel 将选择载入的第一个可用 KeyPair。在此之前，默认强制使用 'ssh-rsa' 的 KeyType。		字符串
MAC （安全性）	以逗号分隔的允许/支持的消息验证代码算法列表（按首选顺序排列）。MAC 算法用于数据完整性保护。		字符串
password (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

Name	描述	默认值	类型
签名 （安全性）	以逗号分隔的允许/支持的签名算法列表，按首选顺序排列。		字符串
用户名 （安全性）	设置用于登录远程 SSH 服务器的用户名。		字符串

131.5. 端点选项

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：

131.5.1. 路径参数(2 参数)

Name	描述	默认值	类型
host (common)	**必需** 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。	22	int

131.5.2. 查询参数(39 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 knownHosts 属性时，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)	设置建立远程 SSH 服务器连接时要等待的超时时间（毫秒）。默认值为 30000 毫秒。	30000	long
pollCommand (consumer)	设置在每次轮询周期期间要发送到远程 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://...)，您可能需要使用新行结束命令，且必须 URL 编码的 %0A。		字符串

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
bridgeErrorHandler (consumer (advanced))	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当 Camel 消费者试图获取传入的消息或类似信息时，会出现异常（如果可能），现在将被作为消息进行处理，并由路由 Error Handler 处理。重要：只有在第三方组件允许 Camel 抛出异常时，才能警报这一点。有些组件仅在内部处理，因此无法 bridgeErrorHandler。在其他情况下，我们可能会将 Camel 组件提高到第三方组件中的 hook，并使其可能用于将来的版本。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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut		ExchangePattern
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy
lazyStartProducer (producer (advanced))	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
channelType (advanced)	设置要在命令执行过程中传递给频道的频道类型。默认为 exec。	exec	字符串
ClientBuilder (advanced)	生成者或消费者使用的 autowired ClientBuilder 实例，以创建新的 SshClient。		ClientBuilder

Name	描述	默认值	类型
compressions (advanced)	是否要使用压缩，如果有。		字符串
shellPrompt (advanced)	在命令执行后读取响应时，将 shellPrompt 设置为被丢弃。		字符串
sleepForShellPrompt (advanced)	设置从 shell 提示符读取响应的睡眠周期（以毫秒为单位）。默认值为 100 毫秒。	100	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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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
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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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 来解析基于文件的证书，并依赖于 keyType 设置。		字符串
certResourcePassword (security)	如果 certResource 是加密密钥，则设置要在加载 certResource 中使用的密码。		字符串
密码 （安全）	以逗号分隔的允许/支持的密码列表（按首选顺序排列）。		字符串
KEX （安全性）	以逗号分隔的允许/支持的密钥交换算法列表，按首选顺序排列。		字符串
keyPairProvider (security)	设置使用证书连接到远程 SSH 服务器时要使用的 KeyPairProvider 引用。		KeyPairProvider

Name	描述	默认值	类型
keytype (security)	设置在身份验证过程中传递给 KeyPairProvider 的密钥类型。KeyPairProvider.loadKey (...)将传递这个值。在 Camel 3.0.0 / 2.25.0 中，默认情况下 Camel 将选择载入的第一个可用 KeyPair。在此之前，默认强制使用 'ssh-rsa' 的 KeyType。		字符串
MAC （安全性）	以逗号分隔的允许/支持的消息验证代码算法列表（按首选顺序排列）。MAC 算法用于数据完整性保护。		字符串
password (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
签名 （安全性）	以逗号分隔的允许/支持的签名算法列表，按首选顺序排列。		字符串
用户名 （安全性）	设置用于登录远程 SSH 服务器的用户名。		字符串

131.6. 消息标头

***SSH** 组件支持 **4** 个消息标头，如下所列：*

Name	描述	默认值	类型
CamelSshUsername (common) 常量 ：USERNAME_HEADER	用户名。		字符串
CamelSshPassword (common) 常量 ：PASSWORD_HEADER	密码。		字符串
CamelSshStderr (common) 常量 ：Duz al	此标头的值是 InputStream，它带有可执行文件的标准错误流。		InputStream

Name	描述	默认值	类型
CamelSshExitValue (common) constant: EXIT_VALUE	此标头的值是执行后返回的退出值。按照惯例非零状态退出值表示异常终止。请注意，退出值取决于操作系统。		整数

131.7. 使用作为 PRODUCER 端点

当 SSH 组件用作 Producer （'.to ("ssh://...")' 时，它将消息正文作为命令发送，以便在远程 SSH 服务器上执行。

XML DSL 示例

请注意，命令有一个 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>
```

131.8. 身份验证

SSH 组件可以使用以下两种机制之一对远程 SSH 服务器进行身份验证：

- 公钥证书
- 用户名/密码

根据如何设置选项，配置 SSH 组件如何进行身份验证。

1. 首先，它会查看 certResource 选项是否已设置，如果设置了，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，如果设置了 keyPairProvider，它将将其用于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。尽管 用户名和密码 是在通过 SshConstants.USERNAME_HEADER (CamelSshUsername)和 SshConstants.PASSWORD_HEADER (CamelSshPassword)设置的端点配置和标头设置的标头中提供的，也会使用标头中设置的凭证。

以下路由片段演示了使用 classpath 中的证书进行 SSH 轮询消费者。

XML DSL

```
<route>
  <from uri="ssh://scott@localhost:8101?
certResource=classpath:test_rsa&amp;useFixedDelay=true&amp;delay=5000&amp;pollCommand=feat
ures:list%0A"/>
  <log message="${body}"/>
</route>
```

Java DSL

```
from("ssh://scott@localhost:8101?
certResource=classpath:test_rsa&useFixedDelay=true&delay=5000&pollCommand=features:li
st%0A")
    .log("${body}");
```

例如，example /camel-example-ssh-security 中提供了使用公钥身份验证的示例。

131.9. 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。根据您使用的 Camel 版本，

您可能需要使用更新的版本。

组件使用 sshd-core 库，该库基于 bouncycastle 或 eddsa 安全提供程序。camel-ssh 被明确选择为安全供应商。

```
<dependency>
  <groupId>org.apache.sshd</groupId>
  <artifactId>sshd-core</artifactId>
  <version>2.8.0</version>
</dependency>
<dependency>
  <groupId>org.bouncycastle</groupId>
  <artifactId>bcpg-jdk18on</artifactId>
  <version>1.71</version>
</dependency>
<dependency>
  <groupId>org.bouncycastle</groupId>
  <artifactId>bcpkix-jdk18on</artifactId>
  <version>1.71</version>
</dependency>
```

131.10. 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.ssh.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.ssh.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 Camel 消费者试图获取传入的消息或类似信息时，会出现异常（如果可能），现在将被作为消息进行处理，并由路由 Error Handler 处理。重要：只有在第三方组件允许 Camel 抛出异常时，才能警报这一点。有些组件仅在内部处理，因此无法 bridgeErrorHandler。在其他情况下，我们可能会将 Camel 组件提高到第三方组件中的 hook，并使其可能用于将来的版本。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
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 来解析基于文件的证书，并依赖于 keyType 设置。		字符串
camel.component.ssh.cert-resource-password	如果 certResource 是加密密钥，则设置要在加载 certResource 中使用的密码。		字符串
camel.component.ssh.channel-type	设置要在命令执行过程中传递给频道的频道类型。默认为 exec。	exec	字符串
camel.component.ssh.ciphers	以逗号分隔的允许/支持的密码列表（按首选顺序排列）。		字符串
camel.component.ssh.client-builder	producer 或 consumer 用来创建一个新的 SshClient 的 ClientBuilder 实例。选项是 org.apache.sshd.client.ClientBuilder 类型。		ClientBuilder
camel.component.ssh.compressions	是否要使用压缩，如果有。		字符串
camel.component.ssh.configuration	组件配置.选项是 org.apache.camel.component.ssh.SshConfiguration 类型。		SshConfiguration
camel.component.ssh.enabled	是否启用 ssh 组件的自动配置。这默认是启用的。		布尔值
camel.component.ssh.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 knownHosts 属性时，才会检查这个值。	false	布尔值
camel.component.ssh.health-check-consumer-enabled	用于从这个组件启用或禁用所有基于消费者的健康检查。	true	布尔值
camel.component.ssh.health-check-producer-enabled	用于从此组件启用或禁用所有基于制作者的健康检查。注意：默认情况下，Camel 禁用了所有基于健康检查的制作者。您可以通过设置 camel.health.producersEnabled=true 来全局打开制作者检查。	true	布尔值
camel.component.ssh.kex	以逗号分隔的允许/支持的密钥交换算法列表，按首选顺序排列。		字符串

Name	描述	默认值	类型
camel.component.ssh.key-pair-provider	设置使用证书连接到远程 SSH 服务器时要使用的 KeyPairProvider 引用。选项是 org.apache.sshd.common.keyprovider.KeyPairProvider 类型。		KeyPairProvider
camel.component.ssh.key-type	设置在身份验证过程中传递给 KeyPairProvider 的密钥类型。KeyPairProvider.loadKey (...)将传递这个值。在 Camel 3.0.0 / 2.25.0 中，默认情况下 Camel 将选择载入的第一个可用 KeyPair。在此之前，默认强制使用 'ssh-rsa' 的 KeyType。		字符串
camel.component.ssh.known-hosts-resource	设置 known_hosts 文件的资源路径。		字符串
camel.component.ssh.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.ssh.macs	以逗号分隔的允许/支持的消息验证代码算法列表（按首选顺序排列）。MAC 算法用于数据完整性保护。		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	设置在每次轮询周期期间要发送到远程 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://...)，您可能需要使用新行结束命令，且必须 URL 编码的 %0A。		字符串
camel.component.ssh.shell-prompt	在命令执行后读取响应时，将 shellPrompt 设置为被丢弃。		字符串
camel.component.ssh.signatures	以逗号分隔的允许/支持的签名算法列表，按首选顺序排列。		字符串
camel.component.ssh.sleep-for-shell-prompt	设置从 shell 提示符读取响应的睡眠周期（以毫秒为单位）。默认值为 100 毫秒。	100	Long
camel.component.ssh.timeout	设置建立远程 SSH 服务器连接时要等待的超时时间（毫秒）。默认值为 30000 毫秒。	30000	Long

Name	描述	默认值	类型
camel.component.ssh.username	设置用于登录远程 SSH 服务器的用户名。		字符串

第 132 章 STUB

支持生成者和消费者

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。只需在任何端点 URI 前面添加 stub: 以根出端点。

Stub 组件在内部 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，因此在带有查询参数的典型 URI 前放置 vm: 通常会失败。然而，stub 基本上会忽略所有查询参数，以便您快速处理路由中的一个或多个端点。

132.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 stub 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-stub-starter</artifactId>
</dependency>
```

132.2. URI 格式

```
stub:someUri
```

其中 someUri 可以是任何带有任何查询参数的 URI。

132.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

132.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

132.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

132.4. 组件选项

Stub 组件支持 10 个选项，如下所列。

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)	设置默认并发线程处理交换数。	1	int

Name	描述	默认值	类型
defaultPollTimeout (consumer (advanced))	轮询时使用的超时（以毫秒为单位）。发生超时时，消费者可以检查是否允许继续运行。设置较低值可让消费者在关闭时更快地响应。	1000	int
defaultBlockWhenFull (producer)	将消息发送到完整的 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，抛出异常表示队列已满。通过启用此选项，调用线程将阻止并等待消息被接受。	false	布尔值
defaultDiscardWhenFull (producer)	是否将消息发送到完整的 SEDA 队列的线程将被丢弃。默认情况下，抛出异常表示队列已满。通过启用此选项，调用线程将放弃发送并继续，这意味着消息没有发送到 SEDA 队列。	false	布尔值
defaultOfferTimeout (producer)	将消息发送到完整的 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，抛出异常表示队列已满。通过启用这个选项，可将配置的超时添加到块问题单中。利用布线 java 队列的 .offer (timeout)方法。		long
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
defaultQueueFactory (advanced)	设置默认队列工厂。		BlockingQueueFactory
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

132.5. 端点选项

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：

132.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
name (common)	**必需** 队列的名称。		字符串

132.5.2. 查询参数(18 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
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
limitConcurrentConsumers (consumer (advanced))	是否将 concurrentConsumers 的数量限制为最多 500 个。默认情况下，如果端点配置了更多数字，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

Name	描述	默认值	类型
multipleConsumers (consumer (advanced))	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 消息传递。也就是说，您可以向 SEDA 队列发送消息，并使每个消费者收到消息的副本。启用后，应在每个消费者端点上指定这个选项。	false	布尔值
pollTimeout (consumer (advanced))	轮询时使用的超时（以毫秒为单位）。发生超时时，消费者可以检查是否允许继续运行。设置较低值可让消费者在关闭时更快地响应。	1000	int
purgeWhenStopping (consumer (advanced))	在停止 consumer/route 时是否清除任务队列。这样可以更快地停止，因为队列中的任何待处理消息都会被丢弃。	false	布尔值
blockWhenFull (producer)	将消息发送到完整的 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，抛出异常表示队列已满。通过启用此选项，调用线程将阻止并等待消息被接受。	false	布尔值
discardIfNoConsumers (producer)	生产者是否应丢弃消息（不要将消息添加到队列中），当发送到没有活动消费者的队列时。只能同时启用其中一个选项 discardIfNoConsumers 和 failIfNoConsumers。	false	布尔值
discardWhenFull (producer)	是否将消息发送到完整的 SEDA 队列的线程将被丢弃。默认情况下，抛出异常表示队列已满。通过启用此选项，调用线程将放弃发送并继续，这意味着消息没有发送到 SEDA 队列。	false	布尔值
failIfNoConsumers (producer)	当发送到没有活跃消费者的队列时，生成者是否应该通过抛出异常失败。只能同时启用其中一个选项 discardIfNoConsumers 和 failIfNoConsumers。	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
offerTimeout (producer)	当队列满时，可以将提供超时（以毫秒为单位）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 生成者将停止等待异步任务完成前的超时（毫秒）。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 任务完成，然后再继续。支持以下三个选项： Always, Never 或 IfReplyExpected。前两个值是 self-explanatory。最后的值(ifReplyExpected)将仅在消息是 Request Reply based 时等待。默认选项为 IfReplyExpected。 Enum 值： • Never • IfReplyExpected • Always	IfReplyExpected	WaitForTaskToComplete
queue (advanced)	定义端点将使用的队列实例。		BlockingQueue

132.6. 例子

以下是一些存根端点 uri 的示例

```
stub:smtp://somehost.foo.com?user=whatnot&something=else
stub:http://somehost.bar.com/something
```

132.7. 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.stub.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.stub.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

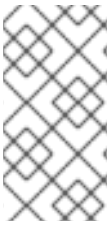
Name	描述	默认值	类型
camel.component.stub.concurrent-consumers	设置默认并发线程处理交换数。	1	整数
camel.component.stub.default-block-when-full	将消息发送到完整的 SEDA 队列的线程将阻止，直到队列的容量不再耗尽为止。默认情况下，抛出异常表示队列已满。通过启用此选项，调用线程将阻止并等待消息被接受。	false	布尔值
camel.component.stub.default-discard-when-full	是否将消息发送到完整的 SEDA 队列的线程将被丢弃。默认情况下，抛出异常表示队列已满。通过启用此选项，调用线程将放弃发送并继续，这意味着消息没有发送到 SEDA 队列。	false	布尔值
camel.component.stub.default-offer-timeout	将消息发送到完整的 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，抛出异常表示队列已满。通过启用这个选项，可将配置的超时添加到块问题单中。利用布线 java 队列的 .offer (timeout)方法。		Long
camel.component.stub.default-poll-timeout	轮询时使用的超时（以毫秒为单位）。发生超时时，消费者可以检查是否允许继续运行。设置较低值可让消费者在关闭时更快地响应。	1000	整数
camel.component.stub.default-queue-factory	设置默认队列工厂。选项是一个 org.apache.camel.component.seda.BlockingQueueFactory<org.apache.camel.Exchange> 类型。		BlockingQueueFactory
camel.component.stub.enabled	是否启用 stub 组件的自动配置。这默认是启用的。		布尔值
camel.component.stub.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.stub.queue-size	设置 SEDA 队列的默认最大容量（例如，它可以保存的消息数）。	1000	整数

第 133 章 TELEGRAM

支持生成者和消费者

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 来发送和接收信息，参与与普通用户、私有和公共组或频道的直接对话。

在使用此组件之前，必须先创建 Telegram Bot，按照 Telegram Bot developers 家的说明进行操作。创建新的 Bot 时，Bo tFather 提供了一个与 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。

133.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 telegram 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-telegram-starter</artifactId>
</dependency>
```

133.2. URI 格式

```
telegram:type[?options]
```

133.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别

- 端点级别

133.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

133.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

133.4. 组件选项

Telegram 组件支持 7 个选项，如下所列。

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
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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
baseURI (advanced)	可用于设置替代的基本 URI，例如，当您要针对模拟 Telegram API 测试组件时。		字符串
client (advanced)	使用自定义 AsyncHttpClient。		AsyncHttpClient
clientConfig (advanced)	将 AsyncHttpClient 配置为使用自定义 com.ning.http.client.AsyncHttpClientConfig 实例。		AsyncHttpClientConfig
authorizationToken (security)	端点中未提供信息时，要使用的默认 Telegram 授权令牌。		字符串

133.5. 端点选项

Telegram 端点使用 URI 语法进行配置：

```
telegram:type
```

使用以下路径和查询参数：

133.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
type (common)	**必需** 端点类型。目前，只支持 'bots' 类型。 Enum 值： • bots		字符串

133.5.2. 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
Limit (consumer)	限制单个轮询请求中可接收的更新数量。	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
timeout (consumer)	长时间轮询的超时时间（以秒为单位）。将 0 置于短轮询或更长的轮询，以进行长时间轮询。长时间轮询会生成较短的响应时间。	30	整数
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
pollStrategy (consumer (advanced))	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。		PollingConsumerPollStrategy

Name	描述	默认值	类型
chatId (producer)	将接收所生成消息的 chat 的标识符。chat id 可以首先从传入的消息中获取（例如，当电话用户开始与 bot 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
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
baseURI (advanced)	可用于设置替代的基本 URI，例如，当您要针对模拟 Telegram API 测试组件时。		字符串
bufferSize (advanced)	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
clientConfig (advanced)	将 AsyncHttpClient 配置为使用自定义 com.ning.http.client.AsyncHttpClientConfig 实例。		AsyncHttpClientConfig
proxyHost (proxy)	发送消息时可以使用的 HTTP 代理主机。		字符串
proxyPort (proxy)	发送消息时可以使用的 HTTP 代理端口。		整数
proxyType (proxy)	发送消息时可以使用的 HTTP 代理类型。 Enum 值： • HTTP • SOCKS4 • SOCKS5	HTTP	TelegramProxyType
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）。	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。	1000	long
repeatCount (scheduler)	指定触发的最大数量。因此，如果您将其设置为 1，调度程序将只触发一次。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
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。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	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。		ScheduledExecutorService
scheduler (scheduler)	要使用 camel-spring 或 camel-quartz 组件的 cron 调度程序。使用值 spring 或 quartz 用于内置在调度程序中。	none	对象
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。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	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
authorizationToken (security)	需要使用 bot 的授权令牌（如 BotFather）。		字符串

133.5.3. 消息标头

Name	描述
CamelTelegramChatId	此标头供 producer 端点用于解析将接收消息的 chat id。接收者 chat id 可以在消息正文（按优先级顺序）放置在 **CamelTelegramChatId** 标头或端点配置(**chatId** 选项)中。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枚举值，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	此标头用于使用 HTML 或 Markdown 格式化文本消息（请参阅 **org.apache.camel.component.telegram.TelegramParseMode**）。

133.6. 使用方法

Telegram 组件支持消费者和制作者端点。它还可用于 被动 chat-bot 模式 （使用，然后生成消息）。

133.7. 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。

在 Java DSL 中

```
from("direct:start").to("telegram:bots?
authorizationToken=123456789:insertYourAuthorizationTokenHere");
```

或在 Spring XML 中

```
<route>
    <from uri="direct:start"/>
    <to uri="telegram:bots?authorizationToken=123456789:insertYourAuthorizationTokenHere"/>
<route>
```

代码 123456789:insertYourAuthorizationTokenHere 是与 Bot 对应的 授权令牌。

在没有指定 chat id 选项的情况下使用制作者端点时，将使用消息正文或标头中包含的信息来标识目标 chat。以下消息正文可用于生成者端点（类型为 OutgoingXXXMessage 的消息属于软件包 org.apache.camel.component.telegram.model）

Java 类型	描述
OutgoingTextMessage	将文本消息发送到聊天
OutgoingPhotoMessage	向聊天发送照片(JPG、PNG)
OutgoingAudioMessage	将 mp3 音频发送到聊天
OutgoingVideoMessage	将 mp4 视频发送到聊天
OutgoingDocumentMessage	发送文件到聊天（任意介质类型）
OutgoingStickerMessage	发送粘滞者到聊天(WEBP)
OutgoingAnswerInlineQuery	向内联查询发送答案
EditMessageTextMessage	编辑文本和游戏消息(editMessageText)
EditMessageCaptionMessage	编辑消息的标题(editMessageCaption)

Java 类型	描述
EditMessageMediaMessage	编辑动画、音频、文档、照片或视频消息(editMessageMedia)
EditMessageReplyMarkupMessage	仅编辑消息的回复标记。(editMessageReplyMarkup)
EditMessageDelete	删除消息，包括服务消息。(deleteMessage)
SendLocationMessage	发送位置(setSendLocation)
EditMessageLiveLocationMessage	将更改发送到实时位置(editMessageLiveLocation)
StopMessageLiveLocationMessage	在 live_period 过期前，停止更新 bot 或通过 bot （用于内联 bots）发送的实时位置消息(stopMessageLiveLocation)
SendVenueMessage	发送有关场所的信息（发送）
byte[]	发送支持任何介质类型。它需要 **CamelTelegramMediaType** 标头设置为适当的介质类型
字符串	发送文本消息到聊天。它会自动转换为 **OutgoingTextMessage**

133.8. 消费者示例

以下是如何接收 telegram 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?authorizationToken=123456789:insertYourAuthorizationTokenHere")
.bean(ProcessorBean.class)
```

或在 Spring XML 中

```
<route>
  <from uri="telegram:bots?authorizationToken=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
    }

}
```

传入的消息支持的类型有

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，仅适用于文本消息

133.9. REACTIVE CHAT-BOT 示例

reactive chat-bot 模式是使用 Camel 组件构建一个简单的 chat bot， 它直接回复 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?authorizationToken=123456789:insertYourAuthorizationTokenHere")
.bean(ChatBotLogic.class)
.to("telegram:bots?authorizationToken=123456789:insertYourAuthorizationTokenHere");
```

或在 Spring XML 中

```
<route>
    <from uri="telegram:bots?authorizationToken=123456789:insertYourAuthorizationTokenHere"/>
    <bean ref="chatBotLogic" />
    <to uri="telegram:bots?authorizationToken=123456789:insertYourAuthorizationTokenHere"/>
<route>

<bean id="chatBotLogic" class="com.example.ChatBotLogic"/>
```

ChatBotLogic 是实施通用 String-to-String 方法的简单 bean。

```
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
    }

}
```

由 chatBotProcess 方法返回的每个非null 字符串会自动路由到源自请求的 chat （因为 CamelTelegramChatId 标头被用来路由消息）。

133.10. 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，然后运行类似以下路由：

```
from("telegram:bots?authorizationToken=123456789:insertYourAuthorizationTokenHere")
.to("log:INFO?showHeaders=true");
```

bot 收到的任何消息都将转储到您的日志，其中包含有关聊天(CamelTelegramChatId 标头)的信息。

获取 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?
authorizationToken=123456789:insertYourAuthorizationTokenHere&chatId=123456")
```

请注意，对应的 URI 参数只是 chatId。

133.11. 自定义键盘

您可以自定义用户键盘，而不是要求他编写选项。OutgoingTextMessage 具有属性 ReplyMarkup，可用于此类操作。

```
from("telegram:bots?authorizationToken=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.setReplyMarkup(replyMarkup);

        exchange.getIn().setBody(msg);
    })
    .to("telegram:bots?authorizationToken=123456789:insertYourAuthorizationTokenHere");
```

如果要禁用它，下一个消息必须在 ReplyKeyboardMarkup 对象上设置属性 removeKeyboard。

```
from("telegram:bots?authorizationToken=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?authorizationToken=123456789:insertYourAuthorizationTokenHere");
```

133.12. WEBHOOK 模式

Telegram 组件支持在 webhook mode 中使用 camel-webhook 组件。

要启用 Webhook 模式，用户首先需要在其应用中添加 REST 实现。Maven 用户，例如，可以将

netty-http 添加到其 pom.xml 文件中：

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

完成后，您需要将 webhook URI 添加到您要使用的 telegram URI。

Java DSL：

```
from("webhook:telegram:bots?
authorizationToken=123456789:insertYourAuthorizationTokenHere").to("log:info");
```

有些端点将由您的应用程序公开，并将 Telegram 配置为向它们发送消息。您需要确保服务器连接到互联网，并传递 camel.component.webhook.configuration.webhook-external-url 属性的正确值。

有关如何设置它的说明，请参阅 camel-webhook 组件文档。

133.13. 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.telegram.authorization-token	端点中未提供信息时，要使用的默认 Telegram 授权令牌。		字符串
camel.component.telegram.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.telegram.base-uri	可用于设置替代的基本 URI，例如，当您要针对模拟 Telegram API 测试组件时。		字符串

Name	描述	默认值	类型
camel.component.telegram.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.telegram.client	使用自定义 AsyncHttpClient。选项是 org.asynchttpclient.AsyncHttpClient 类型。		AsyncHttpClient
camel.component.telegram.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。		AsyncHttpClientConfig
camel.component.telegram.enabled	是否启用 telegram 组件的自动配置。这默认是启用的。		布尔值
camel.component.telegram.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

第 134 章 计时器

仅支持消费者

Timer 组件用于在计时器触发时生成消息交换，您只能使用来自此端点的事件。

134.1. 依赖项

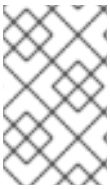
当在 Red Hat build of Camel Spring Boot 中使用 计时器 时，请确保使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-timer-starter</artifactId>
</dependency>
```

134.2. URI 格式

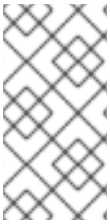
```
timer:name[?options]
```

其中 name 是 Timer 对象的名称，它跨端点创建并共享。因此，如果您对所有计时器端点使用相同的名称，则只使用一个 计时器 对象和线程。



注意

生成的交换的 IN 正文为 null。因此 exchange.getIn ().getBody () 返回 null。



注意

高级调度程序
另请参阅支持更多高级调度的 Quartz 组件。

134.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

134.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

134.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

134.4. 组件选项

Timer 组件支持 2 个选项，如下所列。

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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

134.5. 端点选项

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：

134.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
timerName (consumer)	**必需** 计时器的名称。		字符串

134.5.2. 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

Name	描述	默认值	类型
delay (consumer)	触发第一个事件前的延迟。	1000	long
fixedRate (consumer)	事件大约会定期进行，由指定周期分开。	false	布尔值
includeMetadata (consumer)	是否在交换中包含元数据，如触发的时间、计时器名称、计时器数等。此信息会被默认包括。	true	布尔值
period (consumer)	如果大于 0，请在每次期间都生成定期事件。	1000	long
repeatCount (consumer)	指定触发的最大数量。因此，如果您将其设置为 1，则计时器将只触发一次。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。		long
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern
daemon (advanced)	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)	允许您指定使用自定义日期模式，用于使用 URI 语法设置时间选项。		字符串
同步 (advanced)	设置是否应严格使用同步处理。	false	布尔值
time (advanced)	应生成第一个事件 java.util.Date。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:ss。		Date
timer (advanced)	使用自定义计时器。		计时器

134.6. EXCHANGE PROPERTIES

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**	当前触发计数器。从 1 开始.

134.7. 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");
```

以上路由将生成一个事件，然后在 Registry 中调用名为 myBean 的 bean 的 someMethodName 方法。

和 Spring DSL 中的路由：

```
<route>
  <from uri="timer://foo?fixedRate=true&amp;period=60000"/>
  <to uri="bean:myBean?method=someMethodName"/>
</route>
```

134.8. 尽快触发

从 Camel 2.17 开始

您可能需要尽快触发 Camel 路由中的消息，您可以使用负延迟：

```
<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。

您还可以结合使用 repeatCount 参数，并在达到固定数量后停止触发消息。

如果没有指定 repeatCount，则计时器将继续触发消息，直到路由停止为止。

134.9. 只触发一次

您可能只想在 Camel 路由中触发一条消息，例如在启动路由时。为此，您可以使用 repeatCount 选项，如下所示：

```
<route>
  <from uri="timer://foo?repeatCount=1"/>
  <to uri="bean:myBean?method=someMethodName"/>
</route>
```

134.10. 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.timer.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.timer.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

Name	描述	默认值	类型
camel.component.timer.enabled	是否启用计时器组件的自动配置。这默认是启用的。		布尔值

第 135 章 TOKENIZE

令牌器语言是 camel-core 中的内置语言，最常用于 Split EIP，以使用基于令牌的策略分割消息。

令牌器语言旨在使用指定的分隔符模式对文本文档进行令牌化。它还可用来通过一些有限的功能对 XML 文档进行令牌化。对于一个真正的 XML 感知令牌化，建议使用 XML 令牌 语言，因为它提供了一种更快、更有效的令牌，专门用于 XML 文档。

135.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用令牌时，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-core-starter</artifactId>
</dependency>
```

135.2. 令牌化选项

Tokenize 语言支持 11 个选项，如下所列。

Name	默认值	Java 类型	描述
token		字符串	必需 (start)令牌用作令牌工具，例如，您可以使用新行令牌。您可以使用简单语言作为令牌来支持动态令牌。
endToken		字符串	如果使用 start/end token 对，用作令牌器的末尾令牌。您可以使用简单语言作为令牌来支持动态令牌。
inheritNamespaceTagName		字符串	要使用 XML 时，要从 root/parent 标签名称继承命名空间，您可以使用简单语言作为标签名称来支持动态名称。
headerName		字符串	令牌化的标头名称，而不使用消息正文。
regex		布尔值	如果令牌是正则表达式模式。默认值为 false。
xml		布尔值	输入是否为 XML 消息。如果使用 XML 有效负载，则必须将此选项设置为 true。
includeTokens		布尔值	使用对时是否将令牌包含在部分中。默认值为 false。

Name	默认值	Java 类型	描述
group		字符串	将 N 部分分组在一起，例如将大型文件分成 1000 行的块。您可以使用简单语言作为组来支持动态组大小。
groupDelimiter		字符串	设置在分组时要使用的分隔符。如果没有设置，则令牌将用作分隔符。
skipFirst		布尔值	跳过第一个元素。
trim		布尔值	是否修剪值以移除前导和结尾的空格和换行符。

135.3. 示例

以下示例演示了如何从 direct:a 端点获取请求，然后使用 Expression 将其分成一些内容，然后将每个部分转发到 direct:b ：

```
<route>
  <from uri="direct:a"/>
  <split>
    <tokenize token="\n"/>
    <to uri="direct:b"/>
  </split>
</route>
```

在 Java DSL 中：

```
from("direct:a")
    .split(body().tokenize("\n"))
        .to("direct:b");
```

135.4. 另请参阅

如需更多示例，请参阅 Split EIP。

135.5. SPRING BOOT AUTO-CONFIGURATION

组件支持 147 选项，如下所列。

Name	描述	默认值	类型
camel.cloud.consul.service-discovery.acl-token	设置用于 Consul 的 ACL 令牌。		字符串
camel.cloud.consul.service-discovery.block-seconds	等待监视事件的秒数，默认为 10 秒。	10	整数
camel.cloud.consul.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.consul.service-discovery.connect-timeout-millis	OkHttpClient 的连接超时。		Long
camel.cloud.consul.service-discovery.datacenter	数据中心。		字符串
camel.cloud.consul.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.consul.service-discovery.password	设置用于基本身份验证的密码。		字符串
camel.cloud.consul.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.consul.service-discovery.read-timeout-millis	OkHttpClient 的读取超时。		Long
camel.cloud.consul.service-discovery.url	Consul 代理 URL。		字符串

Name	描述	默认值	类型
camel.cloud.consul.service-discovery.user-name	设置用于基本身份验证的用户名。		字符串
camel.cloud.consul.service-discovery.write-timeout-millis	OkHttpClient 的写入超时。		Long
camel.cloud.dns.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.dns.service-discovery.domain	域名；		字符串
camel.cloud.dns.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.dns.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.dns.service-discovery.proto	所需服务的传输协议。	_tcp	字符串
camel.cloud.etcd.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.etcd.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.etcd.service-discovery.password	用于基本身份验证的密码。		字符串

Name	描述	默认值	类型
camel.cloud.etcd.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.etcd.service-discovery.service-path	查找服务发现的路径。	/services/	字符串
camel.cloud.etcd.service-discovery.timeout	要设置操作可以采取的最长时间，请执行以下操作：		Long
camel.cloud.etcd.service-discovery.type	要设置发现类型，有效值为 on-demand 和 watch。	按需	字符串
camel.cloud.etcd.service-discovery.uris	客户端可以连接到的 URI。		字符串
camel.cloud.etcd.service-discovery.user-name	用于基本身份验证的用户名。		字符串
camel.cloud.kubernetes.service-discovery.api-version	使用客户端查找时设置 API 版本。		字符串
camel.cloud.kubernetes.service-discovery.ca-cert-data	使用客户端查找时设置证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.ca-cert-file	在使用客户端查找时，设置从文件加载的证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-data	使用客户端查找时设置客户端证书数据。		字符串

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.client-cert-file	在使用客户端查找时，设置从文件加载的客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-algo	设置客户端密钥存储算法，如使用客户端查找时 RSA。		字符串
camel.cloud.kubernetes.service-discovery.client-key-data	使用客户端查找时设置客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-file	在使用客户端查找时，设置从文件加载的客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-passphrase	使用客户端查找时设置客户端密钥存储密码短语。		字符串
camel.cloud.kubernetes.service-discovery.configurations	定义其他配置定义。		Map
camel.cloud.kubernetes.service-discovery.dns-domain	设置用于 DNS 查找的 DNS 域。		字符串
camel.cloud.kubernetes.service-discovery.enabled	启用组件。	true	布尔值
camel.cloud.kubernetes.service-discovery.lookup	如何执行服务查找。可能的值有：client、dns、environment。在使用客户端时，客户端会查询 kubernetes master 来获取提供该服务的活跃 pod 列表，然后随机（或循环）选择一个 pod。当使用 dns 时，服务名称被解析为 name.namespace.svc.dnsDomain。当使用 dnssrv 时，服务名称使用 SRV 查询解析 ....svc... when using environment，环境变量用于查找服务。默认情况下使用环境。	环境	字符串

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.master-url	在使用客户端查找时，将 URL 设置为 master。		字符串
camel.cloud.kubernetes.service-discovery.namespace	设置要使用的命名空间。默认情况下，将使用来自 ENV 变量 KUBERNETES_MASTER 的命名空间。		字符串
camel.cloud.kubernetes.service-discovery.oauth-token	在使用客户端查找时，为身份验证设置 OAUTH 令牌（而不是用户名/密码）。		字符串
camel.cloud.kubernetes.service-discovery.password	在使用客户端查找时设置用于身份验证的密码。		字符串
camel.cloud.kubernetes.service-discovery.port-name	设置用于 DNS/DNSSRV 查找的端口名称。		字符串
camel.cloud.kubernetes.service-discovery.port-protocol	设置用于 DNS/DNSSRV 查找的端口协议。		字符串
camel.cloud.kubernetes.service-discovery.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.kubernetes.service-discovery.trust-certs	设置在使用客户端查找时是否打开信任证书检查。	false	布尔值
camel.cloud.kubernetes.service-discovery.username	在使用客户端查找时设置用于身份验证的用户名。		字符串

Name	描述	默认值	类型
camel.cloud.ribbon.load-balancer.client-name	设置 Ribbon 客户端名称。		字符串
camel.cloud.ribbon.load-balancer.configurations	定义其他配置定义。		Map
camel.cloud.ribbon.load-balancer.enabled	启用组件。	true	布尔值
camel.cloud.ribbon.load-balancer.namespace	命名空间。		字符串
camel.cloud.ribbon.load-balancer.password	密码。		字符串
camel.cloud.ribbon.load-balancer.properties	设置要使用的客户端属性。这些属性特定于使用中的服务调用实现。例如，如果使用 ribbon，则客户端属性在 com.netflix.client.config.CommonClientConfigKey 中定义。		Map
camel.cloud.ribbon.load-balancer.username	用户名。		字符串
camel.hystrix.allow-maximum-size-to-diverge-from-core-size	允许配置使 maximumSize 生效。然后该值可以等于或大于 coreSize。	false	布尔值
camel.hystrix.circuit-breaker-enabled	是否使用 HystrixCircuitBreaker。如果为 false，则不会使用 断路器逻辑，并且所有允许的请求。这与 circuitBreakerForceClosed （）的影响类似，除非继续跟踪指标，知道它是否应该是 open/closed，此属性即使实例化一个断路器。	true	布尔值

Name	描述	默认值	类型
camel.hystrix.circuit-breaker-error-threshold-percentage	错误百分比阈值（如 50 ）指向断路器将打开和拒绝请求。它将在 circuitBreakerSleepWindowInMilliseconds 中定义的持续时间保持出差；与 HystrixCommandMetrics.getHealthCounts （）进行比较的错误百分比。	50	整数
camel.hystrix.circuit-breaker-force-closed	如果为 true，HystrixCircuitBreaker#allowRequest （）将始终返回 true 以允许请求，无论 HystrixCommandMetrics.getHealthCounts （）的错误百分比如何。如果设为 true，则 circuitBreakerForceOpen （）属性具有优先权。	false	布尔值
camel.hystrix.circuit-breaker-force-open	如果为 true，HystrixCircuitBreaker.allowRequest （）将始终返回 false，从而导致电路变为开路（接受），并拒绝所有请求。此属性优先于 circuitBreakerForceClosed （）;。	false	布尔值
camel.hystrix.circuit-breaker-request-volume-threshold	metricsRollingStatisticalWindowInMilliseconds （）中的最少请求数必须存在于 HystrixCircuitBreaker 之前。如果此数字低于这个数字，无论错误百分比如何，电路都不会被出差。	20	整数
camel.hystrix.circuit-breaker-sleep-window-in-milliseconds	HystrixCircuitBreaker trips 之后的时间（以毫秒为单位），它应该在尝试请求前等待。	5000	整数
camel.hystrix.configurations	定义其他配置定义。		Map
camel.hystrix.core-pool-size	传递给 java.util.concurrent.ThreadPoolExecutor#setCorePoolSize (int)的核心 thread-pool 大小。	10	整数
camel.hystrix.enabled	启用组件。	true	布尔值
camel.hystrix.execution-isolation-semaphore-max-concurrent-requests	允许 HystrixCommand.run （）的并发请求数。超过并发限制的请求将被拒绝。仅在执行IsolationStrategy == SEMAPHORE 时使用。	20	整数

Name	描述	默认值	类型
camel.hystrix.execution-isolation-strategy	将通过什么隔离策略 HystrixCommand.run （）执行。如果 THREAD，它将在单独的线程上执行，并且受 thread-pool 中的线程数量限制的并发请求。如果 SEMAPHORE，它将在调用线程上执行，并且受 semaphore 数限制的并发请求。	线程	字符串
camel.hystrix.execution-isolation-thread-interrupt-on-timeout	当线程超时时，执行线程是否应该尝试中断（使用 future#cancel）。仅在执行IsolationStrategy （）== THREAD 时才适用。	true	布尔值
camel.hystrix.execution-timeout-enabled	此命令是否启用了超时机制。	true	布尔值
camel.hystrix.execution-timeout-in-milliseconds	以毫秒为单位，将命令超时和停止执行的时间（以毫秒为单位）。如果 executionIsolationThreadInterruptOnTimeout == true 且命令是线程隔离，则执行线程将中断。如果命令是 semaphore-isolated 和 HystrixObservableCommand，则该命令将被取消订阅。	1000	整数
camel.hystrix.fallback-enabled	出现故障时，是否应尝试 HystrixCommand.getFallback （）。	true	布尔值
camel.hystrix.fallback-isolation-semaphore-max-concurrent-requests	允许 HystrixCommand.getFallback （）的并发请求数。超过并发限制的请求将快速失败，且不会尝试检索回退。	10	整数
camel.hystrix.group-key	设置要使用的 group 键。默认值为 CamelHystrix。	Camel Hystrix	字符串
camel.hystrix.keep-alive-time	更长的时间（以分钟为单位）传递给 ThreadPoolExecutor#setKeepAliveTime (long,TimeUnit)。	1	整数
camel.hystrix.max-queue-size	在 HystrixConcurrencyStrategy.getBlockingQueue (int)中传递给 BlockingQueue 的最大队列大小应该只影响 threadpool 的实例化 - 它不会立即更改队列大小。为此，请使用 queueSizeRejectionThreshold （）。	-1	整数

Name	描述	默认值	类型
camel.hystrix.maximum-size	传递给 ThreadPoolExecutor#setMaximumPoolSize(int)的最大 thread-pool 大小。这是可在不开始拒绝 HystrixCommands 的情况下支持的最大并发数量。请注意，只有在您也设置了 allowMaximumSizeToDivergeFromCoreSize 时，此设置才会生效。	10	整数
camel.hystrix.metrics-health-snapshot-interval-in-milliseconds	在允许计算成功和错误百分比时等待的时间（以毫秒为单位），并影响 HystrixCircuitBreaker.isOpen （）状态。在高容量电路上，错误百分比的连续计算可能会成为 CPU 密集型，从而控制其计算的频率。	500	整数
camel.hystrix.metrics-rolling-percentile-bucket-size	滚动百分比的每个存储桶中存储的最大值数。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	10	整数
camel.hystrix.metrics-rolling-percentile-enabled	是否应该使用 HystrixRollingPercentile 内部 HystrixCommandMetrics 来捕获百分比的指标。	true	布尔值
camel.hystrix.metrics-rolling-percentile-window-buckets	滚动窗口的存储桶数量被分成。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	6	整数
camel.hystrix.metrics-rolling-percentile-window-in-milliseconds	以毫秒为单位的滚动窗口的持续时间。这在 HystrixCommandMetrics 中被传递至 HystrixRollingPercentile。	10000	整数
camel.hystrix.metrics-rolling-statistical-window-buckets	滚动统计窗口划分为的 bucket 数量。这在 HystrixCommandMetrics 中被传递给 HystrixRollingNumber。	10	整数
camel.hystrix.metrics-rolling-statistical-window-in-milliseconds	此属性设置统计滚动窗口的持续时间，以毫秒为单位。这是为线程池保留指标的时间。窗口被分成 bucket，按这些增量回滚。	10000	整数
camel.hystrix.queue-size-rejection-threshold	队列大小拒绝阈值是 artificial max size，即使尚未达到 maxQueueSize，也会发生拒绝。这是因为 BlockingQueue 的 maxQueueSize 无法动态更改，我们希望动态更改影响拒绝的队列大小。在排队线程以进行执行时，HystrixCommand 会使用它。	5	整数

Name	描述	默认值	类型
camel.hystrix.request-log-enabled	HystrixCommand 执行和事件是否应记录到 HystrixRequestLog。	true	布尔值
camel.hystrix.thread-pool-key	设置要使用的线程池密钥。默认情况下，将使用与 groupKey 配置相同的值。	Camel Hystrix	字符串
camel.hystrix.thread-pool-rolling-number-statistical-window-buckets	滚动统计窗口划分为的 bucket 数量。这会传递到每个 HystrixThreadPoolMetrics 实例内的 HystrixRollingNumber。	10	整数
camel.hystrix.thread-pool-rolling-number-statistical-window-in-milliseconds	统计滚动窗口的持续时间（以毫秒为单位）。这会传递到每个 HystrixThreadPoolMetrics 实例内的 HystrixRollingNumber。	10000	整数
camel.language.constant.enabled	是否启用恒定语言的自动配置。这默认是启用的。		布尔值
camel.language.constant.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.csimple.enabled	是否启用 csimple 语言的自动配置。这默认是启用的。		布尔值
camel.language.csimple.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.exchangeproperty.enabled	是否启用 exchangeProperty 语言的自动配置。这默认是启用的。		布尔值
camel.language.exchangeproperty.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.file.enabled	是否启用文件语言的自动配置。这默认是启用的。		布尔值
camel.language.file.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.header.enabled	是否启用标头语言的自动配置。这默认是启用的。		布尔值

Name	描述	默认值	类型
camel.language.header.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.ref.enabled	是否启用 ref 语言的自动配置。这默认是启用的。		布尔值
camel.language.ref.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.simple.enabled	是否启用简单语言的自动配置。这默认是启用的。		布尔值
camel.language.simple.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.tokenize.enabled	是否启用令牌化语言的自动配置。这默认是启用的。		布尔值
camel.language.tokenize.group-delimiter	设置在分组时要使用的分隔符。如果没有设置，则令牌将用作分隔符。		字符串
camel.language.tokenize.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.resilience4j.automatic-transition-from-open-to-half-open-enabled	在通过 waitDurationInOpenState 后，启用从 OPEN 自动过渡到 HALF_OPEN 状态。	false	布尔值
camel.resilience4j.circuit-breaker-ref	代表现有的 io.github.resilience4j.circuitbreaker.CircuitBreaker 实例从 registry 中查找和使用。使用此选项时，不使用任何其他断路器选项。		字符串
camel.resilience4j.config-ref	指的是现有的 io.github.resilience4j.circuitbreaker.CircuitBreakerConfig 实例，以便从 registry 中查找和使用。		字符串
camel.resilience4j.configurations	定义其他配置定义。		Map
camel.resilience4j.enabled	启用组件。	true	布尔值

Name	描述	默认值	类型
camel.resilience4j.failure-rate-threshold	以百分比为单位配置故障率阈值。如果失败率相等或大于阈值，则 CircuitBreaker 过渡到 open，并启动短路调用。阈值必须大于 0，而不是大于 100。默认值为 50 百分比。		æµ®ç‚¹å€¼
camel.resilience4j.minimum-number-of-calls	在 CircuitBreaker 可以计算错误率之前，配置所需的最少调用数（每个滑动期限）。例如，如果 minimumNumberOfCalls 为 10，则必须至少记录 10 个调用，然后才能计算失败率。如果只记录了 9 个调用，则 CircuitBreaker 不会过渡到 open，即使所有 9 调用都失败。默认 minimumNumberOfCalls 为 100。	100	整数
camel.resilience4j.permitted-number-of-calls-in-half-open-state	配置 CircuitBreaker 为一半打开时允许的调用数量。大小必须大于 0。默认大小为 10。	10	整数
camel.resilience4j.sliding-window-size	配置滑动窗口的大小，该窗口用于在 CircuitBreaker 关闭时记录调用的结果。slidingWindowSize 配置滑动窗口的大小。滑动窗口可以是基于计数或基于时间的窗口。如果 slidingWindowType 是 COUNT_BASED，则最后的 slidingWindowSize 调用会被记录并聚合。如果 slidingWindowType 是 TIME_BASED，则最后 slidingWindowSize 秒的调用会被记录并聚合。slidingWindowSize 必须大于 0。minimumNumberOfCalls 必须大于 0。如果 slidingWindowType 是 COUNT_BASED，则 minimumNumberOfCalls 不能大于 slidingWindowSize。如果 slidingWindowType 是 TIME_BASED，您可以选择任何您需要的。默认 slidingWindowSize 为 100。	100	整数
camel.resilience4j.sliding-window-type	配置滑动窗口的类型，用于记录 CircuitBreaker 关闭时调用的结果。滑动窗口可以是基于计数或基于时间的窗口。如果 slidingWindowType 是 COUNT_BASED，则最后的 slidingWindowSize 调用会被记录并聚合。如果 slidingWindowType 是 TIME_BASED，则最后 slidingWindowSize 秒的调用会被记录并聚合。默认 slidingWindowType 是 COUNT_BASED。	COUNT_BASED	字符串
camel.resilience4j.slow-call-duration-threshold	配置上面的持续时间阈值（秒），调用被视为缓慢，并增加较慢的调用百分比。默认值为 60 秒。	60	整数

Name	描述	默认值	类型
camel.resilience4j.slow-call-rate-threshold	以百分比为单位配置阈值。当调用持续时间大于 slowCallDurationThreshold Duration 时，CircuitBreaker 会将调用视为较慢。当较慢的调用百分比相等或大于阈值时，CircuitBreaker 过渡到 open，并启动短路调用。阈值必须大于 0，而不是大于 100。默认值为 100 百分比，这意味着所有记录的调用都必须比 slowCallDurationThreshold 慢。		æµ®ç‚¹å€¼
camel.resilience4j.wait-duration-in-open-state	配置等待持续时间（以秒为单位），指定 CircuitBreaker 应该保持打开的时间，然后再切换到半次。默认值为 60 秒。	60	整数
camel.resilience4j.writable-stack-trace-enabled	启用可写入堆栈跟踪。当设置为 false 时，Exception.getStackTrace 返回一个零长度数组。当断路器处于开路状态时，这可用于减少日志垃圾邮件，因为存在例外的原因（断路器是短路调用）。	true	布尔值
camel.rest.api-component	用作 REST API 的 Camel 组件名称（如 swagger）如果没有明确配置 API 组件，则 Camel 会查找负责服务并生成 REST API 文档的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestApiProcessorFactory。如果找到其中任何一个，则使用它。		字符串
camel.rest.api-context-path	设置领导的 API 上下文路径将使用的 REST API 服务。这在使用使用 context-path 部署 Web 应用程序的组件（如 camel-servlet ）时使用它。		字符串
camel.rest.api-context-route-id	设置用于服务 REST API 的路由的路由 ID。默认情况下，路由将使用自动分配的路由 ID。		字符串
camel.rest.api-host	要将特定主机名用于 API 文档（如 swagger），这可用于用这个配置的主机名覆盖生成的主机。		字符串
camel.rest.api-property	允许为 api 文档配置任意数量的附加属性(swagger)。例如，将属性 api.title 设置为我的冷却。		Map
camel.rest.api-vendor-extension	是否在 Rest API 中启用供应商扩展。如果启用，Camel 将包含额外信息作为厂商扩展名（例如，以 x- 开头的键），如路由 ID、类名称等。在导入 API 文档时，并非所有第三方 API 网关和工具都支持 vendor-extensions。	false	布尔值
camel.rest.binding-mode	设置要使用的绑定模式。默认值为 off。		RestBindingMode

Name	描述	默认值	类型
camel.rest.client-request-validation	是否启用客户端请求验证，以检查客户端的 Content-Type 和 Accept 标头是否受到其 consume/produces 设置的 Rest-DSL 配置的支持。这可以打开，以启用此检查。如果验证错误，则返回 HTTP Status code 415 或 406。默认值为 false。	false	布尔值
camel.rest.component	用于 REST 传输(consumer)的 Camel Rest 组件，如 netty-http, jetty, servlet, undertow。如果没有明确配置组件，则 Camel 会查找是否有与 Rest DSL 集成的 Camel 组件，或者如果 registry 中注册了 org.apache.camel.spi.RestConsumerFactory。如果找到其中任何一个，则使用它。		字符串
camel.rest.component-property	允许为正在使用的其他组件配置任意数量的附加属性。		Map
camel.rest.consumer-property	允许为使用中的其他使用者配置任意数量的附加属性。		Map
camel.rest.context-path	设置 REST 服务将使用的前导上下文路径。这在使用使用 context-path 部署 Web 应用程序的组件（如 camel-servlet ）时使用它。或者对于包含 HTTP 服务器的 camel-jetty 或 camel-netty-http 等组件。		字符串
camel.rest.cors-headers	允许配置自定义 CORS 标头。		Map
camel.rest.data-format-property	允许为使用的数据格式配置多个额外属性。例如，将属性 prettyPrint 设置为 true，以便以用户友善模式输出 json。属性可以加上前缀来表示选项仅适用于 JSON 或 XML，以及 IN 或 OUT。前缀为： json.in. json.out. xml.in. xml.out。例如，值为 xml.out.mustBeJAXBElement 的键仅用于传出的 XML 数据格式。没有前缀的密钥是所有情况的通用密钥。		Map
camel.rest.enable-cors	是否在 HTTP 响应中启用 CORS 标头。默认值为 false。	false	布尔值
camel.rest.endpoint-property	允许为使用中的其他端点配置多个额外的属性。		Map
camel.rest.host	用于公开 REST 服务的主机名。		字符串
camel.rest.host-name-resolver	如果没有明确配置的主机名，这个 resolver 会用于计算 REST 服务将要使用的主机名。		RestHostNameResolver

Name	描述	默认值	类型
camel.rest.json-data-format	要使用的特定 json 数据格式的名称。默认将使用 json-jackson。重要：此选项仅用于设置数据格式的自定义名称，而不是引用现有数据格式实例。		字符串
camel.rest.port	用于公开 REST 服务的主机名。请注意，如果您使用 servlet 组件，则此处配置的端口号不适用，因为使用中的端口号是 servlet 组件使用的实际端口号。例如，如果使用 Apache Tomcat，它的 tomcat http 端口，如果使用 Apache Karaf，它的在 Karaf 中的 HTTP 服务，它默认使用端口 8181。虽然在这些情况下，这里设置端口号，但允许工具和 JMX 知道端口号，因此建议将端口号设置为 servlet 引擎使用的数字。		字符串
camel.rest.producer-api-doc	设置 api 文档的位置，REST 生成者将根据这个文档来验证 REST uri 和查询参数是否有效。这需要将 camel-swagger-java 添加到 classpath 中，任何缺失的配置都会导致 Camel 在启动时失败并报告错误。默认情况下从 classpath 加载的 api 文档的位置，但您可以使用 file: 或 http: 引用从文件或 http url 加载的资源。		字符串
camel.rest.producer-component	设置要用作 REST 生成者的 Camel 组件的名称。		字符串
camel.rest.scheme	用于公开 REST 服务的方案。通常支持 http 或 https。默认值为 http。		字符串
camel.rest.skip-binding-on-error-code	如果存在自定义 HTTP 错误代码标头，是否跳过输出绑定。这允许构建没有绑定到 json / xml 等自定义错误消息，否则成功信息会这样做。	false	布尔值
camel.rest.use-x-forward-headers	是否将 X-Forward 标头用于主机和相关设置。默认值为 true。	true	布尔值
camel.rest.xml-data-format	要使用的特定 XML 数据格式的名称。默认情况下将使用 jaxb。重要：此选项仅用于设置数据格式的自定义名称，而不是引用现有数据格式实例。		字符串
camel.rest.api-context-id-pattern	**弃用** 设置 CamelContext id 特征，以只允许 CamelContext 中名称与特征匹配的其他服务的 Rest API。特征 name 指的是 CamelContext 名称，仅匹配当前的 CamelContext。对于任何其他值，特征使用来自 PatternHelper#matchPattern (String,String)的规则。		字符串

Name	描述	默认值	类型
camel.rest.api-context-listing	**弃用** 设置是否启用了 JVM 中带有 REST 服务的所有可用 CamelContext 的列表。如果启用，它将允许发现这些上下文，如果为 false，则只使用当前的 CamelContext。	false	布尔值

第 136 章 验证器

仅支持生成者

Validation 组件使用 JAXP Validation API 执行消息正文的 XML 验证，并基于任何受支持的 XML 模式语言，默认为 XML Schema

请注意，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。

136.1. 依赖项

当在 Camel Spring Boot 中使用 验证器 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-validator-starter</artifactId>
</dependency>
```

136.2. URI 格式

```
validator:someLocalOrRemoteResource
```

其中 someLocalOrRemoteResource 是 classpath 上本地资源的一些 URL，或者包含要验证的 XSD 文件系统上的远程资源的完整 URL 或资源。例如：

- msv:org/foo/bar.xsd

- msv:file:../foo/bar.xsd
- msv:http://acme.com/cheese.xsd
- validator:com/mypackage/myschema.xsd

Validation 组件直接在 camel-core 中提供。

136.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

136.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

136.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

136.4. 组件选项

Validator 组件支持 3 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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
resourceResolverFactory (advanced)	使用自定义 LSResourceResolver，它依赖于动态端点资源 URI。		ValidatorResourceResolverFactory

136.5. 端点选项

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：

136.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
resourceUri (producer)	对于 classpath 上的本地资源，或引用在 Registry 中查找 bean 所需的 URL，或引用在包含要验证的 XSD 文件系统上的远程资源或资源的完整 URL。		字符串

136.5.2. 查询参数(10 参数)

Name	描述	默认值	类型
failOnNullBody (producer)	如果不存在正文，是否失败。	true	布尔值
failOnNullHeader (producer)	在针对标头验证时，是否不存在标头失败。	true	布尔值
headerName (producer)	对标头而不是邮件正文进行验证。		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
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。默认资源解析器工厂为资源解析器，可以从类路径和文件系统中读取文件。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

Name	描述	默认值	类型
useSharedSchema (advanced)	架构实例是否应共享。引进了这个选项，以临时处理 JDK 1.6.x 错误。Xerces 不应该出现这个问题。	true	布尔值

136.6. 示例

以下示例演示了如何配置来自端点 direct:start 的路由，然后进入两个端点之一，可以是 mock:valid 或 mock:invalid，具体取决于 XML 是否与给定架构（在 classpath 上提供）匹配。

136.7. 高级：JMX 方法 CLEARCACHEDSCHEMA

您可以强制清除验证器端点中的缓存模式，并使用 JMX 操作 clearCachedSchema 重新读取下一个进程调用。您还可以使用此方法以编程方式清除缓存。这个方法可用于 ValidatorEndpoint 类。

136.8. 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.validator.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.validator.enabled	是否启用验证器组件的自动配置。这默认是启用的。		布尔值
camel.component.validator.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.validator.resource-resolver-factory	使用自定义 LSResourceResolver，它依赖于动态端点资源 URI。选项是 org.apache.camel.component.validator.ValidatorResourceResolverFactory 类型。		ValidatorResourceResolverFactory

第 137 章 VELOCITY

自 Camel 1.2 开始

仅支持生成者

Velocity 组件允许您使用 Apache Velocity 模板处理消息。在使用模板生成请求的响应时，这是理想选择。

137.1. 依赖项

当在 Camel Spring Boot 中使用 velocity 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-velocity-starter</artifactId>
</dependency>
```

137.2. URI 格式

```
velocity: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 （例如 file://folder/myfile.vm）。

137.3. 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

137.3.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

137.3.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

137.4. 组件选项

Velocity 组件支持 5 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细信息。默认情况下，只能访问消息正文和标头。这个选项可以启用对当前 Exchange 和 CamelContext 的完整访问权限。这样做会造成潜在的安全风险，因为这将打开对 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（默认为 false）。启用此选项后，可以通过消息标头指定动态模板。但是，如果标头来自恶意用户，则可以将其视为潜在的安全漏洞，因此请谨慎使用它。	false	布尔值

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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
velocityEngine (advanced)	要使用 VelocityEngine，否则会创建一个新的引擎。		VelocityEngine

137.5. 端点选项

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：

137.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
resourceUri (producer)	**所需资源** 的路径。您可以为前缀：classpath, file, http, ref, 或 bean. classpath, file 和 http 使用这些协议加载资源(classpath 为 default)。ref 将查询 registry 中的资源。bean 将调用要用作资源的 bean 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

137.5.2. 查询参数(7 参数)

Name	描述	默认值	类型

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细信息。默认情况下，只能访问消息正文和标头。这个选项可以启用对当前 Exchange 和 CamelContext 的完整访问权限。这样做会造成潜在的安全风险，因为这将打开对 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（默认为 false）。启用此选项后，可以通过消息标头指定动态模板。但是，如果标头来自恶意用户，则可以将其视为潜在的安全漏洞，因此请谨慎使用它。	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 / 禁用默认情况下启用的 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
lazyStartProducer (producer (advanced))	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

137.6. 消息标头

***Velocity** 组件支持 4 个消息标头，如下所列：*

Name	描述	默认值	类型
CamelVelocityResourceUri (producer) 常量 ：VELOCITY_RESOURCE_URI	velocity 模板的名称。		字符串

Name	描述	默认值	类型
CamelVelocityTemplate (producer) 恒定：VELOCITY_TEMPLATE	velocity 模板的内容。		字符串
CamelVelocityContext (producer) 常数 ：VELOCITY_CONTEXT	要使用的 velocity 上下文。		Context
CamelVelocitySupplementalContext (producer) 常量 ：VELOCITY_SUPPLEMENTAL_CONTEXT	在使用的 VelocityContext 中添加额外的信息。此标头的值应当是将添加的键/值（用相同名称覆盖任何现有键）的映射。这可用于预设置要在 velocity 端点中重复使用的一些通用键/值。		Map

在 Velocity 评估期间设置的标头返回给消息，并作为标头添加。然后，可以将值从 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。

137.7. VELOCITY CONTEXT

Camel 将在 Velocity 上下文中提供交换信息（忽略 映射）。Exchange 被传输为：

key	value
交换	**Exchange** 本身。

key	value
exchange.properties	**Exchange** 属性。
标头	In 消息的标头。
camelContext	Camel 上下文实例。
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）。

您可以通过设置属性 allowTemplateFromHeader=true 并设置消息标头 CamelVelocityContext 来设置自定义 Velocity Context，如下所示

```
VelocityContext velocityContext = new VelocityContext(variableMap);
exchange.getIn().setHeader("CamelVelocityContext", velocityContext);
```

137.8. 热重新加载

默认情况下，Velociy 模板资源是 file 和 classpath 资源(expanded jar)的热重新加载。如果您设置了 contentCache=true，Camel 将只加载资源一次，因此无法热重新载入。当资源永不更改时，可以在生产环境中使用这种场景。

137.9. 动态模板

由于 Camel 2.1
Camel 提供了两个标头，因此您可以为模板或模板内容本身定义不同的资源位置。如果设置了这些标

头，则 Camel 会在端点配置的资源中使用它。这可让您在运行时提供动态模板。

标头	类型	描述
CamelVelocityResourceUri	字符串	要使用的模板资源的 URI，而不是配置的端点。
CamelVelocityTemplate	字符串	要使用的模板而不是配置的端点。

137.10. SAMPLES

例如，您可以使用：

```
from("activemq:My.Queue").
  to("velocity:com/acme/MyResponse.vm");
```

要使用 Velocity 模板来对 InOut 消息交换（其中有一个 JMSReplyTo 标头）的响应来建立一个响应。

如果要使用 InOnly 并使用消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");
```

使用内容缓存，例如在生产环境中使用，.vm 模板永远不会更改：

```
from("activemq:My.Queue").
  to("velocity:com/acme/MyResponse.vm?contentCache=true").
  to("activemq:Another.Queue");
```

以及基于文件的资源：

```
from("activemq:My.Queue").
  to("velocity:file://myfolder/MyResponse.vm?contentCache=true").
  to("activemq:Another.Queue");
```

可以指定组件应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"");
```

可以将模板直接指定为标头，则组件应通过标头来动态使用，例如：

```
from("direct:in").
  setHeader("CamelVelocityTemplate").constant("Hi this is a velocity template that can do
templating ${body}").
  to("velocity:dummy?allowTemplateFromHeader=true"");
```

137.11. 电子邮件示例

在本例中，要将 Velocity 模板用于一个订购确认电子邮件。电子邮件模板在 Velocity 中拉动，如下所示：

letter.vm

```
Dear ${headers.lastName}, ${headers.firstName}

Thanks for the order of ${headers.item}.

Regards Camel Riders Bookstore
${body}
```

和 java 代码（从单元测试）：

```
    private Exchange createLetter() {
        Exchange exchange = context.getEndpoint("direct:a").createExchange();
        Message msg = exchange.getIn();
        msg.setHeader("firstName", "Claus");
        msg.setHeader("lastName", "Ibsen");
        msg.setHeader("item", "Camel in Action");
        msg.setBody("PS: Next beer is on me, James");
        return exchange;
    }
```

```
    @Test
    public void testVelocityLetter() throws Exception {
        MockEndpoint mock = getMockEndpoint("mock:result");
        mock.expectedMessageCount(1);
        mock.message(0).body(String.class).contains("Thanks for the order of Camel in Action");

        template.send("direct:a", createLetter());

        mock.assertIsSatisfied();
    }

    @Override
    protected RouteBuilder createRouteBuilder() {
        return new RouteBuilder() {
            public void configure() {
                from("direct:a")
                    .to("velocity:org/apache/camel/component/velocity/letter.vm")
                    .to("mock:result");
            }
        };
    }
```

137.12. 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.velocity.allow-context-map-all	设置上下文映射是否应该允许访问所有详细信息。默认情况下，只能访问消息正文和标头。这个选项可以启用对当前 Exchange 和 CamelContext 的完整访问权限。这样做会造成潜在的安全风险，因为这将打开对 CamelContext API 的完整功能的访问。	false	布尔值
camel.component.velocity.allow-template-from-header	是否允许使用来自标头的资源模板（默认为 false）。启用此选项后，可以通过消息标头指定动态模板。但是，如果标头来自恶意用户，则可以将其视为潜在的安全漏洞，因此请谨慎使用它。	false	布尔值
camel.component.velocity.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.velocity.enabled	是否启用 velocity 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.velocity.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.velocity.velocity-engine	要使用 VelocityEngine，否则会创建一个新的引擎。选项是一个 org.apache.velocity.app.VelocityEngine 类型。		VelocityEngine

第 138 章 VERT.X HTTP CLIENT

从 Camel 3.5 开始

仅支持生成者

Vert.x HTTP 组件提供通过 Vert.x Web 客户端 向 HTTP 端点生成消息的功能。

138.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 vertx-http 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-vertx-http-starter</artifactId>
</dependency>
```

138.2. URI 格式

```
vertx-http:hostname[:port][/resourceUri][?options]
```

138.3. 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

138.3.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

138.3.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

138.4. 组件选项

Vert.x HTTP 客户端组件支持 19 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
responsePayloadAsByteArray (producer)	响应正文应该是 byte 或 io.vertx.core.buffer.Buffer。	true	布尔值

Name	描述	默认值	类型
allowJavaSerializedObject (advanced)	当请求具有 Content-Type application/x-java-serialized-object This is disabled 时，是否允许 java 序列化。如果启用此功能，请注意 Java 将反序列化来自请求的传入数据。这可能是潜在的安全风险。	false	布尔值
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
vertx (advanced)	使用现有的 vertx 而不是创建新实例。		Vertx
vertxHttpBinding (advanced)	自定义 VertxHttpBinding，它可以控制如何在 Vert.x 和 Camel 之间绑定。		VertxHttpBinding
vertxOptions (advanced)	为配置 vertx 提供一组自定义 vertx 选项。		VertxOptions
webClientOptions (advanced)	提供一组自定义选项来配置 vertx Web 客户端。		WebClientOptions
headerFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头到 Camel 消息。		HeaderFilterStrategy
proxyHost (proxy)	代理服务器主机地址。		字符串
proxyPassword (proxy)	如果需要身份验证，代理服务器密码。		字符串
proxyPort (proxy)	代理服务器端口。		整数
proxyType (proxy)	代理服务器类型。 Enum 值： • HTTP • SOCKS4 • SOCKS5		ProxyType
proxyUsername (proxy)	如果需要身份验证，代理服务器用户名。		字符串
basicAuthPassword (security)	用于基本身份验证的密码。		字符串

Name	描述	默认值	类型
basicAuthUsername (security)	用于基本身份验证的用户名。		字符串
BearerToken (security)	用于 bearer 令牌身份验证的 bearer 令牌。		字符串
sslContextParameters (security)	使用 SSLContextParameters 配置安全性。		SSLContextParameters
useGlobalSslContextParameters (security)	启用使用全局 SSL 上下文参数。	false	布尔值

138.5. 端点选项

Vert.x HTTP 客户端端点使用 URI 语法进行配置：

```
vertx-http:httpUri
```

使用以下路径和查询参数：

138.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
httpUri (producer)	**需要** 要连接的 HTTP URI。		URI

138.5.2. 查询参数(23 参数)

Name	描述	默认值	类型
connectTimeout (producer)	建立连接前的时间（毫秒）。超时值为零被解释为无限超时。	60000	int
cookieStore (producer)	启用会话管理时要使用的自定义 CookieStore。如果没有设置这个选项，则使用内存 CookieStore。	InMemoryCookieStore	CookieStore

Name	描述	默认值	类型
headerFilterStrategy (producer)	自定义 org.apache.camel.spi.HeaderFilterStrategy 来过滤标头到 Camel 消息。	VertxHttpHeaderFilterStrategy	HeaderFilterStrategy
httpMethod (producer)	要使用的 HTTP 方法。如果设置，HttpMethod 标头无法覆盖这个选项。		HttpMethod
okStatusCodeRange (producer)	被视为成功响应的状态代码。这些值包含为。可以定义多个范围，用逗号分开，如 200-204,209,301-304。每个范围都必须是一个数字或 from-to，包括横线。	200-299	字符串
responsePayloadAsByteArray (producer)	响应正文应该是 byte 或 io.vertx.core.buffer.Buffer。	true	布尔值
sessionManagement (producer)	通过 WebClientSession 启用会话管理。默认情况下，客户端配置为使用内存 CookieStore。cookieStore 选项可用于覆盖它。	false	布尔值
throwExceptionOnFailure (producer)	如果远程服务器中的响应失败，则禁用抛出 HttpOperationFailedException。	true	布尔值
timeout (producer)	如果请求在超时时间内没有返回任何数据的时间（以毫秒为单位），则 TimeoutException 会导致请求失败。设置零或负值会禁用超时。	-1	long
transferException (producer)	如果在消费者端启用并且 Exchange 失败，如果作为 application/x-java-serialized-object 内容类型发送了导致的 Exception 被序列化。在生成者一侧，异常将被反序列化和抛出，而不是 HttpOperationFailedException。导致异常需要被序列化。默认情况下是关闭的。如果您启用此项，则 Camel 会将传入数据从请求反序列化到 Java 对象，这可能会成为潜在的安全风险。	false	布尔值
useCompression (producer)	设置压缩是否启用来处理压缩（例如 gzipped）响应。	false	布尔值
vertxHttpBinding (producer)	自定义 VertxHttpBinding，它可以控制如何在 Vert.x 和 Camel 之间绑定。		VertxHttpBinding
webClientOptions (producer)	设置用于配置 Vert.x WebClient 的自定义选项。		WebClientOptions

Name	描述	默认值	类型
lazyStartProducer (producer (advanced))	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
proxyHost (proxy)	代理服务器主机地址。		字符串
proxyPassword (proxy)	如果需要身份验证，代理服务器密码。		字符串
proxyPort (proxy)	代理服务器端口。		整数
proxyType (proxy)	代理服务器类型。 Enum 值： • HTTP • SOCKS4 • SOCKS5		ProxyType
proxyUsername (proxy)	如果需要身份验证，代理服务器用户名。		字符串
basicAuthPassword (security)	用于基本身份验证的密码。		字符串
basicAuthUsername (security)	用于基本身份验证的用户名。		字符串
BearerToken (security)	用于 bearer 令牌身份验证的 bearer 令牌。		字符串
sslContextParameters (security)	使用 SSLContextParameters 配置安全性。		SSLContextParameters

138.6. 消息标头

Vert.x HTTP 客户端组件支持 8 个消息标头，如下所列：

Name	描述	默认值	类型
CamelHttpMethod (producer) 常数 ：HTTP_METHOD	http 方法。		HttpMethod
CamelHttpResponseCode (producer) 常量 ：HTTP_RESPONSE_CODE	来自外部服务器的 HTTP 响应代码。		整数
CamelHttpResponseText (producer) 常数：HTTP_RESPONSE_TEXT	来自外部服务器的 HTTP 响应文本。		字符串
content-Type (producer) 常数：CONTENT_TYPE	HTTP 内容类型。在 IN 和 OUT 消息上设置，以提供内容类型，如 text/html。		字符串
CamelHttpQuery (producer) 常数 ：HTTP_QUERY	URI 参数。将覆盖直接在端点上设置的现有 URI 参数。		字符串
CamelHttpUri (producer) 常数：HTTP_URI	要调用的 URI。将直接覆盖端点上设置的现有 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，它只是 http 服务器的 URI。		字符串
CamelHttpPath (producer) 常数 ：HTTP_PATH	请求 URI 的路径，标头将使用 HTTP_URI 构建请求 URI。		字符串
content-Encoding (producer) 常量 ：CONTENT_ENCODING	HTTP 内容编码。设置为提供内容编码，如 gzip。		字符串

138.7. 使用方法

以下示例演示了如何向 HTTP 端点发送请求。

您可以通过标头 Exchange.HTTP_URI 和 Exchange.HTTP_PATH 覆盖在 vertx-http producer 上配置的 URI。

```
from("direct:start")
    .to("vertx-http:https://camel.apache.org");
```

138.8. URI 参数

vertx-http producer 支持要发送到 HTTP 服务器的 URI 参数。URI 参数可以直接在端点 URI 上设置，也可以是消息上密钥 Exchange.HTTP_QUERY 的标头。

138.9. 响应代码

Camel 将根据 HTTP 响应代码处理：

- 响应代码范围为 100..299，Camel 会将它视为成功响应。
- 响应代码在范围 300..399中，Camel 会将它视为重定向响应，并将引发带有信息的 HttpOperationFailedException。
- 响应代码为 400+，Camel 会将它视为外部服务器失败，并将引发带有信息的 HttpOperationFailedException。

138.10. 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。

138.11. 例外

HttpOperationFailedException 异常包含以下信息：

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

138.12. HTTP 方法

以下算法决定了要使用的 HTTP 方法：

1. 使用作为端点配置(httpMethod)的方法。
2. 使用在标头中提供的方法(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 如果存在要发送的数据（用户不是 null）。
6. 否则 GET。

138.13. HTTP 表单参数

您可以通过两种方式之一发送 HTTP 表单参数：

1. 将 Exchange.CONTENT_TYPE 标头设置为值 application/x-www-form-urlencoded，并确保消息正文是一个 String 格式作为表单变量。例如，param1=value1&param2=value2。

2. 将消息正文设置为 MultiMap，允许您配置表单参数和值。

138.14. 多部分形式数据

您可以通过将消息正文设置为 MultipartForm 来上传文本或二进制文件。

138.15. 自定义 VERT.X WEB CLIENT 选项

当需要更精细地控制 Vert.x Web 客户端配置时，您可以将自定义 WebClientOptions 实例绑定到 registry。

```
WebClientOptions options = new WebClientOptions().setMaxRedirects(5)
  .setIdleTimeout(10)
  .setConnectTimeout(3);

camelContext.getRegistry.bind("clientOptions", options);
```

然后，引用 vertx-http producer 上的选项。

```
from("direct:start")
  .to("vertx-http:http://localhost:8080?webClientOptions=#clientOptions")
```

138.15.1. SSL

Vert.x HTTP 组件通过 Camel JSSE 配置实用程序 支持 SSL/TLS 配置。

也可以通过提供自定义 WebClientOptions 来配置 SSL 选项。

138.16. 会话管理

会话管理可以通过 sessionManagement URI 选项启用。启用后，使用内存中 Cookie 存储来跟踪 Cookie。这可以通过通过 CookieStore URI 选项提供自定义 CookieStore 来覆盖。

138.17. 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.vertx-http.allow-java-serialized-object	当请求具有 Content-Type application/x-java-serialized-object This is disabled 时，是否允许 java 序列化。如果启用此功能，请注意 Java 将反序列化来自请求的传入数据。这可能是潜在的安全风险。	false	布尔值
camel.component.vertx-http.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.vertx-http.basic-auth-password	用于基本身份验证的密码。		字符串
camel.component.vertx-http.basic-auth-username	用于基本身份验证的用户名。		字符串
camel.component.vertx-http.bearer-token	用于 bearer 令牌身份验证的 bearer 令牌。		字符串
camel.component.vertx-http.enabled	是否启用 vertx-http 组件的自动配置。这默认是启用的。		布尔值
camel.component.vertx-http.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 过滤标头到 Camel 消息。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		HeaderFilterStrategy
camel.component.vertx-http.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.vertx-http.proxy-host	代理服务器主机地址。		字符串

Name	描述	默认值	类型
camel.component.vertx-http.proxy-password	如果需要身份验证，代理服务器密码。		字符串
camel.component.vertx-http.proxy-port	代理服务器端口。		整数
camel.component.vertx-http.proxy-type	代理服务器类型。		ProxyType
camel.component.vertx-http.proxy-username	如果需要身份验证，代理服务器用户名。		字符串
camel.component.vertx-http.response-payload-as-byte-array	响应正文应该是 byte 或 io.vertx.core.buffer.Buffer。	true	布尔值
camel.component.vertx-http.ssl-context-parameters	使用 SSLContextParameters 配置安全性。选项是 org.apache.camel.support.jsse.SSLContextParameters 类型。		SSLContextParameters
camel.component.vertx-http.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值
camel.component.vertx-http.vertx	使用现有的 vertx 而不是创建新实例。选项是 io.vertx.core.Vertx 类型。		Vertx
camel.component.vertx-http.vertx-http-binding	自定义 VertxHttpBinding，它可以控制如何在 Vert.x 和 Camel 之间绑定。选项是 org.apache.camel.component.vertx.http.VertxHttpBinding 类型。		VertxHttpBinding
camel.component.vertx-http.vertx-options	为配置 vertx 提供一组自定义 vertx 选项。选项是 io.vertx.core.VertxOptions 类型。		VertxOptions

Name	描述	默认值	类型
camel.component.vertx-http.web-client-options	提供一组自定义选项来配置 vertx Web 客户端。选项是 io.vertx.ext.web.client.WebClientOptions 类型。		WebClientOptions

第 139 章 VERT.X WEBSOCKET

从 Camel 3.5 开始

支持生成者和消费者。

http://vertx.io/Vertx] WebSocket 组件提供 WebSocket 功能作为 WebSocket 服务器，或作为客户端来连接到现有的 WebSocket。

139.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 vertx-websocket 时，使用以下 Maven 依赖项来支持自动配置：

```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-vertx-websocket-starter</artifactId>
</dependency>
```

139.2. URI 格式

```
vertx-websocket://hostname[:port][/resourceUri][?options]
```

139.3. 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

139.3.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

139.3.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

139.4. 组件选项

Vert.x WebSocket 组件支持 11 个选项，如下所列。

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
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
allowOriginHeader (advanced)	WebSocket 客户端是否应该将 Origin 标头添加到 WebSocket 握手请求中。	true	布尔值
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
defaultHost (advanced)	WebSocket 应该绑定到的主机名的默认值。	0.0.0.0	字符串
defaultPort (advanced)	WebSocket 应该绑定到的端口的默认值。	0	int
originHeaderUrl (advanced)	WebSocket 客户端应在 WebSocket 握手请求中使用的 Origin 标头值。如果没有指定，WebSocket 客户端将自动从请求 URL 确定 Origin 的值。		字符串
路由器 (advanced)	提供要在 WebSocket 服务器中使用的自定义 vertx 路由器。		路由器
vertx (advanced)	使用现有的 vertx 而不是创建新实例。		Vertx
vertxOptions (advanced)	为配置 vertx 提供一组自定义 vertx 选项。		VertxOptions
useGlobalSslContextParameters (security)	启用使用全局 SSL 上下文参数。	false	布尔值

139.5. 端点选项

Vert.x WebSocket 端点使用 URI 语法进行配置：

```
vertx-websocket:host:port/path
```

使用以下路径和*查询*参数：

139.5.1. 路径参数(3 参数)

Name	描述	默认值	类型
host (common)	在客户端模式中时，**所需的** WebSocket 主机名，如 localhost 或远程主机。		字符串
port (common)	要使用 **所需的** WebSocket 端口号。		int
path (common)	要使用的 websocket 路径。		字符串

139.5.2. 查询参数(18 参数)

Name	描述	默认值	类型
allowedOriginPattern (consumer)	与 WebSocket 客户端发送的原始标头匹配的正则表达式模式。		字符串
allowOriginHeader (consumer)	WebSocket 客户端是否应该将 Origin 标头添加到 WebSocket 握手请求中。	true	布尔值
consumeAsClient (consumer)	当设置为 true 时，消费者充当 WebSocket 客户端，在每个收到的 WebSocket 事件上创建交换。	false	布尔值
fireWebSocketConnectionEvents (consumer)	当新的 WebSocket 对等点连接时，服务器消费者是否会创建一个消息交换。	false	布尔值
maxReconnectAttempts (consumer)	当 useAsClient 设置为 true 时，这会将允许重新连接的最大数量设置为之前关闭的 WebSocket。值 0 （默认值）将尝试无限期重新连接。	0	int
originHeaderUrl (consumer)	WebSocket 客户端应在 WebSocket 握手请求中使用的 Origin 标头值。如果没有指定，WebSocket 客户端将自动从请求 URL 确定 Origin 的值。		字符串
reconnectInitialDelay (consumer)	当 consumeAsClient 设置为 true 时，这会在尝试重新连接到之前关闭的 WebSocket 前设置初始延迟（毫秒）。	0	int
reconnectInterval (consumer)	当 consumeAsClient 设置为 true 时，这会以毫秒为单位设置间隔，这会重新连接到之前关闭的 WebSocket。	1000	int

Name	描述	默认值	类型
router (consumer)	将现有的 vertx 路由器用于 HTTP 服务器：		路由器
serverOptions (consumer)	设置用于为消费者配置托管 WebSocket 的 HTTP 服务器的自定义选项。		HttpServerOptions
bridgeErrorHandler (consumer (advanced))	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 (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： ● InOnly ● InOut		ExchangePattern
clientOptions (producer)	设置用于配置制作者中使用的 WebSocket 客户端的自定义选项。		HttpClientOptions
clientSubProtocols (producer)	以逗号分隔的 WebSocket 子协议列表，客户端用于 Sec-WebSocket-Protocol 标头。		字符串
sendToAll (producer)	发送到所有 websocket 订阅者。可用于在端点级别上配置，而不是在消息中提供 VertxWebsocketConstants.SEND_TO_ALL 标头。请注意，在使用此选项时，为 vertx-websocket producer URI 指定的主机名必须与用于现有 vertx-websocket consumer 的主机名匹配。请注意，只有在将消息生成到由 vertx-websocket 消费者托管的端点时，才会应用此选项，而不是外部托管的 WebSocket。	false	布尔值
lazyStartProducer (producer (advanced))	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
sslContextParameters (security)	使用 SSLContextParameters 配置安全性。		SSLContextParameters

139.6. 消息标头

Vert.x WebSocket 组件支持 4 个消息标头，如下所列：

Name	描述	默认值	类型
CamelVertxWebsocket.connectionKey (common) 常数：CONNECTION_KEY	使用给定连接密钥向客户端发送消息。您可以使用逗号分隔的键列表向多个客户端发送消息。请注意，只有在将消息生成到由 vertx-websocket 消费者托管的端点时，才会应用此选项，而不是外部托管的 WebSocket。		字符串
CamelVertxWebsocket.sendToAll (producer) 常数：SEND_TO_ALL	将消息发送到当前连接的所有客户端。您可以在端点中使用 sendToAll 选项，而不使用此标头。请注意，只有在将消息生成到由 vertx-websocket 消费者托管的端点时，才会应用此选项，而不是外部托管的 WebSocket。		布尔值
CamelVertxWebsocket.remoteAddress (consumer) 常数 ：REMOTE_ADDRESS	远程地址。		SocketAddress
CamelVertxWebsocket.event (consumer) 常量 ： EVENT	触发消息交换的 WebSocket 事件。 Enum 值： • 关闭 • ERROR • MESSAGE • OPEN		VertxWebsocketEvent

139.7. 使用方法

以下示例演示了如何在 http://localhost:8080/echo 上公开 WebSocket，并将"echo"响应返回给同一频道：

```
from("vertx-websocket:localhost:8080/echo")
    .transform().simple("Echo: ${body}")
    .to("vertx-websocket:localhost:8080/echo");
```

也可以使用 consumeAsClient 选项将消费者配置为远程地址上作为 WebSocket 客户端连接：

```
from("vertx-websocket:my.websocket.com:8080/chat?consumeAsClient=true")
    .log("Got WebSocket message ${body}");
```

139.8. PATH 和 QUERY 参数

WebSocket 服务器消费者支持配置参数化路径。path 参数值将设置为 Camel Exchange 标头：

```
from("vertx-websocket:localhost:8080/chat/{user}")
    .log("New message from ${header.user} >>> ${body}")
```

您还可以检索 WebSocket 客户端用来连接到服务器端点的任何查询参数值：

```
from("direct:sendChatMessage")
    .to("vertx-websocket:localhost:8080/chat/camel?role=admin");

from("vertx-websocket:localhost:8080/chat/{user}")
    .log("New message from ${header.user} (${header.role}) >>> ${body}")
```

139.9. 发送消息到连接到 VERTX-WEBSOCKET 服务器消费者的对等点

注意

本节仅在向由 camel-vertx-websocket 使用者托管的 WebSocket 生成消息时才适用。当向外部托管的 WebSocket 生成消息时，这并不相关。

要向连接到由 vertx-websocket 服务器消费者托管的 WebSocket 的所有对等点发送消息，请使用 sendToAll=true 端点选项，或 CamelVertxWebsocket.sendToAll 标头。

```
from("vertx-websocket:localhost:8080/chat")
    .log("Got WebSocket message ${body}");
```

```
from("direct:broadcastMessage")
    .setBody().constant("This is a broadcast message!")
    .to("vertx-websocket:localhost:8080/chat?sendToAll=true");
```

或者，您可以使用 CamelVertxWebsocket.connectionKey 标头将消息发送到特定的对等点。可将多个对等点指定为用逗号分开的列表。

当 peer 在 vertx-websocket 消费者上触发事件时，可以确定 connectionKey 的值，其中标识 peer 的唯一键将通过 CamelVertxWebsocket.connectionKey 标头传播。

```
from("vertx-websocket:localhost:8080/chat")
    .log("Got WebSocket message ${body}");

from("direct:broadcastMessage")
    .setBody().constant("This is a broadcast message!")
    .setHeader(VertxWebsocketConstants.CONNECTION_KEY).constant("key-1,key-2,key-3")
    .to("vertx-websocket:localhost:8080/chat");
```

139.10. SSL

默认情况下，使用 ws:// 协议，但通过 sslContextParameters URI 参数和 Camel JSSE 配置工具 配置消费者或生成者来支持具有 ws:// 协议 的安全连接。

139.11. 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.vertx-websocket.allow-origin-header	WebSocket 客户端是否应该将 Origin 标头添加到 WebSocket 握手请求中。	true	布尔值
camel.component.vertx-websocket.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值

Name	描述	默认值	类型
camel.component.vertx-websocket.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.vertx-websocket.default-host	WebSocket 应该绑定到的主机名的默认值。	0.0.0.0	字符串
camel.component.vertx-websocket.default-port	WebSocket 应该绑定到的端口的默认值。	0	整数
camel.component.vertx-websocket.enabled	是否启用 vertx-websocket 组件的自动配置。这默认是启用的。		布尔值
camel.component.vertx-websocket.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.vertx-websocket.origin-header-url	WebSocket 客户端应在 WebSocket 握手请求中使用的 Origin 标头值。如果没有指定，WebSocket 客户端将自动从请求 URL 确定 Origin 的值。		字符串
camel.component.vertx-websocket.router	提供要在 WebSocket 服务器中使用的自定义 vertx 路由器。选项是一个 io.vertx.ext.web.Router 类型。		路由器
camel.component.vertx-websocket.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.vertx-websocket.vertx	使用现有的 vertx 而不是创建新实例。选项是 io.vertx.core.Vertx 类型。		Vertx
camel.component.vertx-websocket.vertx-options	为配置 vertx 提供一组自定义 vertx 选项。选项是 io.vertx.core.VertxOptions 类型。		VertxOptions

第 140 章 WEBHOOK

仅支持消费者

Webhook meta 组件允许其他 Camel 组件在远程 Webhook 供应商上配置 Webhook 并侦听它们。

以下组件目前提供 Webhook 端点：

- telegram

通常，支持 Webhook 的其他组件会带来这个依赖项。

140.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 webhook 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-webhook-starter</artifactId>
</dependency>
```

140.2. URI 格式

```
webhook:endpoint[?options]
```

140.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

140.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

140.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

140.4. 组件选项

Webhook 组件支持 8 个选项，如下所列。

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
webhookAutoRegister (consumer)	在启动时自动注册 webhook，并在关闭时取消注册。	true	布尔值
webhookBasePath (consumer)	Webhook 将公开的第一个（基础）路径元素。最好将其设置为随机字符串，使其不能被未授权方猜测。		字符串
webhookComponentName (consumer)	用于 REST 传输的 Camel Rest 组件，如 netty-http。		字符串
webhookExternalUrl (consumer)	Webhook 提供程序看到的当前服务的 URL。		字符串
webhookPath (consumer)	Webhook 端点将公开的路径（相对于 basePath，若有）。		字符串
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
configuration (advanced)	设置 webhook meta-component 的默认配置。		WebhookConfiguration

140.5. 端点选项

Webhook 端点使用 URI 语法进行配置：

```
webhook:endpointUri
```

使用以下路径和查询参数：

140.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
endpointUri (consumer)	**必需**。delegate uri。必须属于支持 Webhook 的组件。		字符串

140.5.2. 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
webhookAutoRegister (consumer)	在启动时自动注册 webhook，并在关闭时取消注册。	true	布尔值
webhookBasePath (consumer)	Webhook 将公开的第一个（基础）路径元素。最好将其设置为随机字符串，使其不能被未授权方猜测。		字符串
webhookComponentName (consumer)	用于 REST 传输的 Camel Rest 组件，如 netty-http。		字符串
webhookExternalUrl (consumer)	Webhook 提供程序看到的当前服务的 URL。		字符串
webhookPath (consumer)	Webhook 端点将公开的路径（相对于 basePath，若有）。		字符串
exceptionHandler (consumer (advanced))	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 (advanced))	在消费者创建交换时设置交换模式。 Enum 值： • InOnly • InOut • InOptionalOut		ExchangePattern

140.6. 例子

Webhook 组件示例包括在支持它的委托组件文档中。

140.7. 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.webhook.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.webhook.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.webhook.configuration	设置 webhook meta-component 的默认配置。选项是一个 org.apache.camel.component.webhook.WebhookConfiguration 类型。		WebhookConfiguration
camel.component.webhook.enabled	是否启用 webhook 组件的自动配置。这默认是启用的。		布尔值
camel.component.webhook.webhook-auto-register	在启动时自动注册 webhook，并在关闭时取消注册。	true	布尔值
camel.component.webhook.webhook-base-path	Webhook 将公开的第一个（基础）路径元素。最好将其设置为随机字符串，使其不能被未授权方猜测。		字符串
camel.component.webhook.webhook-component-name	用于 REST 传输的 Camel Rest 组件，如 netty-http。		字符串
camel.component.webhook.webhook-external-url	Webhook 提供程序看到的当前服务的 URL。		字符串
camel.component.webhook.webhook-path	Webhook 端点将公开的路径（相对于 basePath，若有）。		字符串

第 141 章 XJ

Since Camel 3.0

仅支持生成者

XJ 组件允许您直接转换 XML 和 JSON 文档，无需中间 java 对象。您甚至可以指定 XSLT 风格表，直接转换为目标 JSON / XML （域）模型。

141.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 xj 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-xj-starter</artifactId>
</dependency>
```

141.2. URI 格式

```
xj:templateName?transformDirection=XML2JSON|JSON2XML[&options]
```

注意

XJ 组件扩展了 XSLT 组件，因此还支持 XSLT 组件提供的所有选项。至少查看 XSLT 组件文档如何配置 xsl 模板。

transformDirection 选项是必需的，必须是 XML2JSON 或 JSON2XML。templateName 参数允许通过指定名称 身份 来使用 识别转换。

141.3. 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

141.3.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

141.3.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

141.4. 组件选项

XJ 组件支持 11 个选项，如下所列。

Name	描述	默认值	类型
contentCache (producer)	加载时的资源内容（样式表文件）的缓存。如果设置为 false Camel，将在每次消息处理时重新加载风格表文件。这适用于开发。可以使用 clearCachedStylesheet 操作，强制缓存的风格表通过 JMX 在运行时重新加载。	true	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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
saxonConfiguration (advanced)	使用自定义 Saxon 配置。		配置
saxonConfigurationProperties (advanced)	设置自定义 Saxon 配置属性。		Map
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 中。该函数在 registry 中查找，您可以在其中使用逗号分隔多个值来查找。		字符串
secureProcessing (advanced)	XML 安全处理功能（请参阅 javax.xml.XMLConstants）。这默认是启用的。但是，在使用 Saxon 专业时，您可能需要关闭它，以允许 Saxon 能够使用 Java 扩展功能。	true	布尔值
transformerFactoryClass (advanced)	要使用自定义 XSLT 转换器工厂，请指定为 FQN 类名称。		字符串
transformerFactoryConfigurationStrategy (advanced)	应用到新创建的 TransformerFactory 实例的配置策略。		TransformerFactoryConfigurationStrategy
uriResolver (advanced)	使用自定义 UriResolver。不应与 'uriResolverFactory' 选项一同使用。		URIResolver

Name	描述	默认值	类型
uriResolverFactory (advanced)	使用自定义 UriResolver，它依赖于动态端点资源 URI。不应与 'uriResolver' 选项一同使用。		XsltUriResolverFactory

141.5. 端点选项

XJ 端点使用 URI 语法进行配置：

```
xj:resourceUri
```

使用以下路径和查询参数：

141.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
resourceUri (producer)	**必需的**模板的路径。默认 URIResolver 支持以下内容。您可以为前缀：classpath, file, http, ref, 或 bean. classpath, file 和 http 使用这些协议加载资源 (classpath 为 default)。ref 将查询 registry 中的资源。bean 将调用要用作资源的 bean 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

141.5.2. 查询参数(19 参数)

Name	描述	默认值	类型
allowStAX (producer)	是否允许使用 StAX 作为 javax.xml.transform.Source。如果 XSLT 库支持 StAX，如 Saxon 库(camel-saxon)，您可以启用此功能。Xalan 库(JVM 中的默认)不支持 StAXSource。	true	布尔值
contentCache (producer)	加载时的资源内容（样式表文件）的缓存。如果设置为 false Camel，将在每次消息处理时重新加载风格表文件。这适用于开发。可以使用 clearCachedStylesheet 操作，强制缓存的风格表通过 JMX 在运行时重新加载。	true	布尔值
deleteOutputFile (producer)	如果您有 output=file，则此选项指定在处理 Exchange 时是否应删除输出文件。例如，假设输出文件是一个临时文件，那么在使用后，最好将其删除。	false	布尔值

Name	描述	默认值	类型
failOnNullBody (producer)	如果输入正文为 null，是否抛出异常。	true	布尔值
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。前三个选项都基于内存，其中文件直接流传输到 java.io.File。对于文件，您必须使用键 XsltConstants.XSLT_FILE_NAME （也为 CamelXsltFileName）在 IN 标头中指定文件名。另外，任何前需要创建文件名的路径，否则会在运行时抛出异常。 Enum 值： ● 字符串 ● bytes ● DOM ● file	字符串	XsltOutput
transformDirection (producer)	**所需** 转换方向。XML2JSON 或 JSON2XML。 Enum 值： ● XML2JSON ● JSON2XML		TransformDirection
transformerCacheSize (producer)	已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
lazyStartProducer (producer (advanced))	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
entityResolver (advanced)	使用自定义 org.xml.sax.EntityResolver 和 javax.xml.transform.sax.SAXSource。		EntityResolver
errorListener (advanced)	允许配置为使用自定义 javax.xml.transform.ErrorListener。在执行此操作时，需要注意，默认错误监听程序捕获任何错误或致命错误，并将 Exchange 的信息存储为不使用属性。因此，仅将这个选项用于特殊用例。		ErrorListener

Name	描述	默认值	类型
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。		配置
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 中。该函数在 registry 中查找，您可以在其中用逗号分隔要查找的多个值。		字符串
secureProcessing (advanced)	XML 安全处理功能（请参阅 javax.xml.XMLConstants）。这默认是启用的。但是，在使用 Saxon 专业时，您可能需要关闭它，以允许 Saxon 能够使用 Java 扩展功能。	true	布尔值
transformerFactory (advanced)	使用自定义 XSLT 转换器工厂。		TransformerFactory
transformerFactoryClass (advanced)	要使用自定义 XSLT 转换器工厂，请指定为 FQN 类名称。		字符串
transformerFactoryConfigurationStrategy (advanced)	应用到新创建的 TransformerFactory 实例的配置策略。		TransformerFactoryConfigurationStrategy
uriResolver (advanced)	使用自定义 javax.xml.transform.URIResolver。		URIResolver
xsltMessageLogger (advanced)	XSLT 转换期间生成的消息的消费者。		XsltMessageLogger

141.6. 消息标头

XJ 组件支持 1 个消息标头，如下所列：

Name	描述	默认值	类型
CamelXsltFileName (producer) 常量 ：XSLT_FILE_NAME	XSLT 文件名。		字符串

141.7. 使用 XJ 端点

141.7.1. 将 JSON 转换为 XML

以下路由对消息进行"身份"转换，因为未给出 xslt 风格表。在 xml 到 xml 转换的情况下，"Identity"转换意味着输出文档只是输入文档的一个副本。如果是 XJ，这意味着它会将 json 文档转换为等同的 xml 表示。

```
from("direct:start").
  to("xj:identity?transformDirection=JSON2XML");
```

示例：

输入：

```
{
  "firstname": "camel",
  "lastname": "apache",
  "personalnumber": 42,
  "active": true,
  "ranking": 3.1415926,
  "roles": [
    "a",
    {
      "x": null
    }
  ],
  "state": {
    "needsWater": true
  }
}
```

将输出

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<object xmlns:xj="http://camel.apache.org/component/xj" xj:type="object">
   <object xj:name="firstname" xj:type="string">camel</object>
   <object xj:name="lastname" xj:type="string">apache</object>
   <object xj:name="personalnumber" xj:type="int">42</object>
   <object xj:name="active" xj:type="boolean">true</object>
   <object xj:name="ranking" xj:type="float">3.1415926</object>
   <object xj:name="roles" xj:type="array">
      <object xj:type="string">a</object>
      <object xj:type="object">
         <object xj:name="x" xj:type="null">null</object>
      </object>
   </object>
   <object xj:name="state" xj:type="object">
      <object xj:name="needsWater" xj:type="boolean">true</object>
   </object>
</object>
```

正如在上面的输出中所见，XJ 在生成的 xml 中写入一些元数据，这些元数据可用于进一步处理：

- XJ 元数据节点总是位于 http://camel.apache.org/component/xj 命名空间中。
- JSON 键名称放置在 xj:name 属性中。
- 解析的 JSON 类型可以在 xj:type 属性中找到。上例已包含所有可能的类型。
- 生成的 XML 元素始终命名为 "object"。

现在，我们可以应用风格表，例如：

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xj="http://camel.apache.org/component/xj"
                exclude-result-prefixes="xj">

   <xsl:output omit-xml-declaration="no" encoding="UTF-8" method="xml" indent="yes"/>

   <xsl:template match="/">
      <person>
         <xsl:apply-templates select="//object"/>
      </person>
   </xsl:template>

   <xsl:template match="object[@xj:type != 'object' and @xj:type != 'array' and string-
```

```
length(@xj:name) > 0]">
        <xsl:variable name="name" select="@xj:name"/>
        <xsl:element name="{$name}">
            <xsl:value-of select="text()"/>
        </xsl:element>
    </xsl:template>

    <xsl:template match="@*|node()"/>
</xsl:stylesheet>
```

在以上示例中指定端点上的模板：

```
from("direct:start").
  to("xj:com/example/json2xml.xsl?transformDirection=JSON2XML");
```

并获取以下输出：

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
    <firstname>camel</firstname>
    <lastname>apache</lastname>
    <personalnumber>42</personalnumber>
    <active>true</active>
    <ranking>3.1415926</ranking>
    <x>null</x>
    <needsWater>true</needsWater>
</person>
```

141.7.2. 将 XML 转换为 JSON

当未给出风格表时，将根据上述"身份"转换的解释执行：

```
from("direct:start").
  to("xj:identity?transformDirection=XML2JSON");
```

给出示例输入

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
    <firstname>camel</firstname>
    <lastname>apache</lastname>
    <personalnumber>42</personalnumber>
    <active>true</active>
    <ranking>3.1415926</ranking>
    <roles>
        <entry>a</entry>
```

```
        <entry>
          <x>null</x>
        </entry>
      </roles>
      <state>
        <needsWater>true</needsWater>
      </state>
    </person>
```

将导致

```
{
  "firstname": "camel",
  "lastname": "apache",
  "personalnumber": "42",
  "active": "true",
  "ranking": "3.1415926",
  "roles": [
    "a",
    {
      "x": "null"
    }
  ],
  "state": {
    "needsWater": "true"
  }
}
```

您可能注意到，当从 json 转换为 xml 完全没有特别特别时，输入 xml 和输出 json 与上面的示例非常相似。我们只将任意 XML 文档转换为 json。XJ 默认使用以下规则：

- XML root 元素可以命名某种方式，它将始终以 json root 对象声明 '\{}' 结尾
- json 键名称是 xml 元素的名称
- 如果上面有一个名称 clash as in "<roles>"，则会生成两个 "<entry>" 元素。
- 带有 text-only-child-nodes 的 XML 元素将导致通常的键/字符串-值对。混合内容元素会导致键/子对象对，如上面的 "<state>" 中所示。

现在，我们可以再次应用风格表，例如：

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xj="http://camel.apache.org/component/xj"
                exclude-result-prefixes="xj">

    <xsl:output omit-xml-declaration="no" encoding="UTF-8" method="xml" indent="yes"/>

    <xsl:template match="/">
        <xsl:apply-templates/>
    </xsl:template>

    <xsl:template match="personalnumber">
        <xsl:element name="{local-name()}">
            <xsl:attribute name="xj:type">
                <xsl:value-of select="'int'"/>
            </xsl:attribute>
            <xsl:apply-templates/>
        </xsl:element>
    </xsl:template>

    <xsl:template match="active|needsWater">
        <xsl:element name="{local-name()}">
            <xsl:attribute name="xj:type">
                <xsl:value-of select="'boolean'"/>
            </xsl:attribute>
            <xsl:apply-templates/>
        </xsl:element>
    </xsl:template>

    <xsl:template match="ranking">
        <xsl:element name="{local-name()}">
            <xsl:attribute name="xj:type">
                <xsl:value-of select="'float'"/>
            </xsl:attribute>
            <xsl:apply-templates/>
        </xsl:element>
    </xsl:template>

    <xsl:template match="roles">
        <xsl:element name="{local-name()}">
            <xsl:attribute name="xj:type">
                <xsl:value-of select="'array'"/>
            </xsl:attribute>
            <xsl:apply-templates/>
        </xsl:element>
    </xsl:template>

    <xsl:template match="*[normalize-space(text()) = 'null']">
        <xsl:element name="{local-name()}">
            <xsl:attribute name="xj:type">
                <xsl:value-of select="'null'"/>
            </xsl:attribute>
            <xsl:apply-templates/>
        </xsl:element>
    </xsl:template>
```

```
    <xsl:template match="@*|node()">
        <xsl:copy>
            <xsl:apply-templates select="@*|node()"/>
        </xsl:copy>
    </xsl:template>
</xsl:stylesheet>
```

通过指定端点上的模板来进入上例：

```
from("direct:start").
  to("xj:com/example/xml2json.xsl?transformDirection=XML2JSON");
```

并获取以下输出：

```
{
  "firstname": "camel",
  "lastname": "apache",
  "personalnumber": 42,
  "active": true,
  "ranking": 3.1415926,
  "roles": [
    "a",
    {
      "x": null
    }
  ],
  "state": {
    "needsWater": true
  }
}
```

请注意，这个转换会导致与对 json2xml 转换的输入完全相同的 json 文档。以下 XML 文档是在 xsl 转换后传递给 XJ 的内容：

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
    <firstname>camel</firstname>
    <lastname>apache</lastname>
    <personalnumber xmlns:xj="http://camel.apache.org/component/xj"
xj:type="int">42</personalnumber>
    <active xmlns:xj="http://camel.apache.org/component/xj" xj:type="boolean">true</active>
    <ranking xmlns:xj="http://camel.apache.org/component/xj" xj:type="float">3.1415926</ranking>
    <roles xmlns:xj="http://camel.apache.org/component/xj" xj:type="array">
        <entry>a</entry>
        <entry>
            <x xj:type="null">null</x>
        </entry>
    </roles>
```

```
    <state>
        <needsWater xmlns:xj="http://camel.apache.org/component/xj"
xj:type="boolean">true</needsWater>
    </state>
</person>
```

在风格表中，我们只提供了最小的所需类型提示，以获得同样的结果。从 json 转换为 xml 时，支持的类型提示与 XJ 写入 XML 文档完全相同。

在结尾处，我们可以返回 json 到 xml 转换示例的结果文档：

```
<?xml version="1.0" encoding="UTF-8"?>
<object xmlns:xj="http://camel.apache.org/component/xj" xj:type="object">
    <object xj:name="firstname" xj:type="string">camel</object>
    <object xj:name="lastname" xj:type="string">apache</object>
    <object xj:name="personalnumber" xj:type="int">42</object>
    <object xj:name="active" xj:type="boolean">true</object>
    <object xj:name="ranking" xj:type="float">3.1415926</object>
    <object xj:name="roles" xj:type="array">
        <object xj:type="string">a</object>
        <object xj:type="object">
            <object xj:name="x" xj:type="null">null</object>
        </object>
    </object>
    <object xj:name="state" xj:type="object">
        <object xj:name="needsWater" xj:type="boolean">true</object>
    </object>
</object>
```

再次获取相同的输出：

```
{
  "firstname": "camel",
  "lastname": "apache",
  "personalnumber": 42,
  "active": true,
  "ranking": 3.1415926,
  "roles": [
    "a",
    {
      "x": null
    }
  ],
  "state": {
    "needsWater": true
  }
}
```

如上例中所示： xj:type 可让您指定所需的输出类型 * xj:name 可让您覆盖 json 键名称。当您要生成包含在 XML 元素名称中不允许的字符的密钥名称时需要这样做。*

141.7.2.1. 可用类型提示

@xj:type=	描述
object	生成 json 对象
数组	生成 json 数组
string	生成 json 字符串
int	生成 json 号而无需部分部分
浮点值	使用部分部分生成 json 号
布尔值	生成 json 布尔值
null	使用 null 一词生成空值

141.8. 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.xj.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.xj.content-cache	加载时的资源内容（样式表文件）的缓存。如果设置为 false Camel，将在每次消息处理时重新加载风格表文件。这适用于开发。可以使用 clearCachedStylesheet 操作，强制缓存的风格表通过 JMX 在运行时重新加载。	true	布尔值
camel.component.xj.enabled	是否启用 xj 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.xj.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.xj.saxon-configuration	使用自定义 Saxon 配置。选项是一个 net.sf.saxon.Configuration 类型。		配置
camel.component.xj.saxon-configuration-properties	设置自定义 Saxon 配置属性。		Map
camel.component.xj.saxon-extension-functions	允许您使用自定义 net.sf.saxon.lib.ExtensionFunctionDefinition。您需要将 camel-saxon 添加到 classpath 中。该函数在 registry 中查找，您可以在其中使用逗号分隔多个值来查找。		字符串
camel.component.xj.secure-processing	XML 安全处理功能（请参阅 javax.xml.XMLConstants）。这默认是启用的。但是，在使用 Saxon 专业时，您可能需要关闭它，以允许 Saxon 能够使用 Java 扩展功能。	true	布尔值
camel.component.xj.transformer-factory-class	要使用自定义 XSLT 转换器工厂，请指定为 FQN 类名称。		字符串
camel.component.xj.transformer-factory-configuration-strategy	应用到新创建的 TransformerFactory 实例的配置策略。选项是一个 org.apache.camel.component.xslt.TransformerFactoryConfigurationStrategy 类型。		TransformerFactoryConfigurationStrategy
camel.component.xj.uri-resolver	使用自定义 UriResolver。不应与 'uriResolverFactory' 选项一同使用。选项是 javax.xml.transform.URIResolver 类型。		URIResolver
camel.component.xj.uri-resolver-factory	使用自定义 UriResolver，它依赖于动态端点资源 URI。不应与 'uriResolver' 选项一同使用。选项是一个 org.apache.camel.component.xslt.XsltUriResolverFactory 类型。		XsltUriResolverFactory

第 142 章 XML 令牌化

XML 令牌化语言是 camel-xml-jaxp 中的内置语言，它是一个真正的 XML 感知令牌工具，可与 Split EIP 用作高效、有效地 令牌化 XML 文档。

XML 令牌化不仅能够识别文档的 XML 命名空间和层次结构，还能够比传统令牌化语言更高效地对 XML 文档 进行令牌化。

142.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 xtokenize 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-xml-jaxp-starter</artifactId>
</dependency>
```

其他依赖项

要使用此组件，需要额外的依赖项，如下所示：

```
<dependency>
   <groupId>org.codehaus.woodstox</groupId>
   <artifactId>woodstox-core-asl</artifactId>
   <version>4.4.1</version>
</dependency>
```

或者

```
   <dependency>
     <groupId>org.apache.camel.springboot</groupId>
     <artifactId>camel-stax-starter</artifactId>
   </dependency>
```

142.2. XML TOKENIZER 选项

XML 令牌语言支持 4 个选项，如下所列。

Name	默认值	Java 类型	描述
headerName		字符串	令牌化的标头名称，而不使用消息正文。
模式		Enum	提取模式。可用的提取模式有：i - 将上下文命名空间绑定注入提取的令牌（默认） w - 将提取的令牌嵌套在其上级上下文 u 中 - 将提取的令牌解压缩到其子内容 t - 提取的元素文本内容。 Enum 值： • i • w • u • t
group		整数	将 N 部分分组在一起。
trim		布尔值	是否修剪值以移除前导和结尾的空格和换行符。

142.3. 示例

请参阅 Split EIP，其中包含使用 XML Tokenize 语言的示例。

142.4. 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.language.xtokenize.enabled	是否启用 xtokenize 语言的自动配置。这默认是启用的。		布尔值
camel.language.xtokenize.mode	提取模式。可用的提取模式有：i - 将上下文命名空间绑定注入提取的令牌（默认） w - 将提取的令牌嵌套在其上级上下文 u 中 - 将提取的令牌解压缩到其子内容 t - 提取的元素文本内容。		字符串
camel.language.xtokenize.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

第 143 章 XPATH

Camel 支持 XPath 允许在 DSL 中使用 Expression 或 Predicate。

例如，您可以使用 XPath 在 Message Filter 中创建 predicate，或作为 Recipient List 的表达式。

143.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 xpath 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-xpath-starter</artifactId>
</dependency>
```

143.2. XPATH LANGUAGE 选项

XPath 语言支持 10 个选项，如下所列。

Name	默认值	Java 类型	描述
documentType		字符串	文档类型的类名称是 org.w3c.dom.Document。
resultType		Enum	设置结果类型的类名称（输出中的类型）默认结果类型是 NodeSet。 Enum 值： • NUMBER • 字符串 • 布尔值 • NODESET • 节点
saxon		布尔值	是否使用 Saxon。
factoryRef		字符串	引用 registry 中要查找的自定义 XPathFactory。

Name	默认值	Java 类型	描述
objectModel		字符串	要使用的 XPath 对象模型。
logNamespaces		布尔值	是否记录在故障排除过程中协助的命名空间。
headerName		字符串	作为输入的标头名称，而不是消息的正文。
threadSafety		布尔值	是否为 xpath 表达式返回的结果启用 thread-safety。这适用于将 NODESET 用作结果类型，返回的集合具有多个元素。在这种情况下，如果您在并行处理模式中处理 NODESET （如从 Camel Splitter EIP）同时处理，则可能会出现线程安全的问题。这个选项通过执行节点强制副本来防止并发问题。如果您在应用程序中使用 camel-saxon 或 Saxon，则建议您打开这个选项。saxon 有 thread-safety 问题，这可以通过打开此选项来防止。
preCompile		布尔值	是否在初始化阶段启用预编译 xpath 表达式。默认启用预编译。这可用于关闭，例如在开始阶段需要编译阶段的情况，例如，如果应用程序在编译时（例如，camel-quarkus），这会加载构建操作系统的 xpath 工厂，而不是 JVM 运行时。
trim		布尔值	是否修剪值以移除前导和结尾的空格和换行符。

143.3. 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。

143.4. 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	消息
http://camel.apache.org/xml/out/	out	消息	**弃用** 输出消息（不要使用）
http://camel.apache.org/xml/function/	函数	对象	其他函数

命名空间 URI	本地部分	类型	描述
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

143.4.1. 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 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。

143.4.2. 没有给定的命名空间

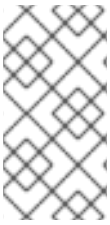
如果没有给定命名空间，则 Camel 仅根据本地部分解析。Camel 将在以下步骤中尝试解析变量：

- 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，如果有一个带有给定键的属性

143.5. 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	仅返回消息正文。
in:header	标头名称	对象	将返回消息正文。
out:body	none	对象	**弃用** 将返回 OUT 消息正文。
out:header	标头名称	对象	**弃用** 将返回 OUT 消息标头。
function:properties	属性的键	字符串	使用 a .
function:simple	简单表达式	对象	来评估一个语言。



注意

当返回类型是 NodeSet 时，不支持 function:properties 和 function:simple，比如与 Split EIP 一起使用时。

下面是一个示例，其中显示了一些正在使用的功能。

143.5.1. 功能示例

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，我们如何在 XPath 表达式中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。

143.6. 基于流的消息正文

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");
```

143.7. 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 字符串。为此，您必须指示要使用的结果类型的 XPath。

Java DSL：

```
xpath("/foo:person/@id", String.class)
```

在 XML 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>
```

注意

java.lang 中的类可以省略 FQN 名称，因此您可以使用 resultType="String"

使用 @XPath 注释：

```
@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，我们需要指定一个 String 作为结果类型，因此 concat 功能可以正常工作。

143.8. 在标头上使用 XPATH

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。

```
<xpath headerName="invoiceDetails">/invoice/@orderType = 'premium'</xpath>
```

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")
```

143.9. 示例

以下是在 Message Filter 中使用 XPath 表达式作为 predicate 的简单示例：

```
from("direct:start")
   .filter().xpath("/person[@name='James']")
      .to("mock:result");
```

在 XML 中

```
<route>
  <from uri="direct:start"/>
  <filter>
    <xpath>/person[@name='James']</xpath>
    <to uri="mock:result"/>
  </filter>
</route>
```

143.10. 使用命名空间

如果您有一个标准的命名空间集，并且希望在多个 XPath 表达式之间共享它们，您可以在使用 Java

DSL 时使用 org.apache.camel.support.builder.Namespaces，如下所示：

```
Namespaces ns = new Namespaces("c", "http://acme.com/cheese");

from("direct:start")
   .filter(xpath("/c:person[@name='James']", ns))
      .to("mock:result");
```

注意如何将命名空间提供给 xquery 以及作为第二参数传递的 ns 变量。

每个命名空间都是 key=value 对，前缀是键。在 XPath 表达式中，命名空间被前缀使用，例如：

```
/c:person[@name='James']
```

命名空间构建器支持添加多个命名空间，如下所示：

```
Namespaces ns = new Namespaces("c", "http://acme.com/cheese")
                .add("w", "http://acme.com/wine")
                .add("b", "http://acme.com/beer");
```

在 XML DSL 中使用命名空间时，如您在 XML root 标签中设置命名空间（或 camelContext, routes, route tag 之一）。

在下面的 XML 示例中，我们使用 Spring XML，其中在 root 标签 Bean 中声明命名空间，在 xmlns:foo="http://example.com/person" 一行中：

```
<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 xmlns="http://camel.apache.org/schema/spring">
    <route>
      <from uri="direct:start"/>
      <filter>
        <xpath logNamespaces="true">/foo:person[@name='James']</xpath>
        <to uri="mock:result"/>
      </filter>
    </route>
```

```
  </camelContext>

</beans>
```

这个命名空间使用 foo 作为前缀，因此 <xpath> 表达式使用 /foo: 来使用这个命名空间。

143.11. 使用 @XPATH ANNOTATION 用于 BEAN 集成

您可以使用 Bean 集成 在 bean 上调用方法，并使用各种语言（如 @XPath ）从消息中提取值并将其绑定到 method 参数。

注意

默认的 @XPath 注释具有 SOAP 和 XML 命名空间。

```
public class Foo {

   @Consume(uri = "activemq:my.queue")
   public void doSomething(@XPath("/person/@name") String name, String xml) {
      // process the inbound message here
   }
}
```

143.12. 在没有交换的情况下使用 XPATHBUILDER

现在，您可以使用 org.apache.camel.language.xpath.XPathBuilder，而无需 Exchange。如果您想将其用作进行自定义 XPath 评估的帮助程序，这很方便。

它要求您传递 CamelContext，因为 XPathBuilder 中的许多移动部分需要访问 Camel Type Converter，因此需要 CamelContext 的原因。

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。

您还可以按照以下三个示例所示评估：

```
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 String 结果评估是一个常见要求，它变得更为简单：

```
String name = XPathBuilder.xpath("foo/bar").evaluate(context, "<foo><bar>cheese</bar>
</foo>");
```

143.13. 使用带有 XPATHBUILDER 的 SAXON

您需要添加 camel-saxon 作为项目的依赖项。

现在，更易于将 Saxon 与 XPathBuilder 搭配使用，该Builder 可以通过几种方式完成，如下所示

- 使用自定义 XPathFactory
- 使用 ObjectModel

143.13.1. 使用系统属性设置自定义 XPathFactory

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。

这个单元测试演示了如何进行此操作来使用 Saxon：

如果使用非默认 XPathFactory，则 Camel 将以 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
```

143.13.2. 从 XML DSL 中启用 Saxon

与 Java DSL 类似，若要从 XML DSL 启用 Saxon，您有三个选项：

引用自定义工厂：

```
<xpath factoryRef="saxonFactory" resultType="java.lang.String">current-dateTime()</xpath>
```

并使用工厂声明 bean：

```
<bean id="saxonFactory" class="net.sf.saxon.xpath.XPathFactoryImpl"/>
```

指定对象模型：

```
<xpath objectModel="http://saxon.sf.net/jaxp/xpath/om" resultType="java.lang.String">current-dateTime()</xpath>
```

推荐的方法是设置 saxon=true，如下所示：

```
<xpath saxon="true" resultType="java.lang.String">current-dateTime()</xpath>
```

143.14. 命名空间审计以帮助调试

用户经常面临的许多与 XPath 相关的问题与命名空间的使用相关联。您的消息中的命名空间之间可能存在一些对齐，以及您的 XPath 表达式了解或引用的命名空间之间。当因为命名空间问题而无法找到 XML 元素和属性的 XPath predicates 或表达式可能像 它们无法正常工作 一样，实际上，它都缺少命名空间定义。

XML 中的命名空间是完全必要的，虽然我们希望通过自动将一些 magic 或 voodoo 到线命名空间来简化其使用情况，但实际上，任何操作都会与标准有关，并大大阻碍了互操作性。

因此，我们可以通过向 XPath Expression Language 添加两个新功能来帮助您调试此类问题，从而可

从 predicates 和 表达式访问。

143.14.1. 记录 XPath expression/predicate 的 Namespace 上下文

每次在内部池中创建新的 XPath 表达式时，Camel 将记录 org.apache.camel.language.xpath.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（父子关系）代表命名空间上下文，整个树都是以递归方式输出，且格式如下：

> [me: {prefix -> namespace}, {prefix -> namespace}], [parent: [me: {prefix -> namespace}, {prefix -> namespace}], [parent: [me: {prefix -> namespace}]]]

这些选项中的任何一个都可用于激活此日志记录：

- 在 org.apache.camel.language.xpath.XPathBuilder 日志记录器或一些父日志记录器（如 org.apache.camel 或 root 日志记录器）上启用 TRACE 日志记录
- 按照以下部分所示启用 logNamespaces 选项，在这种情况下，日志记录将在 INFO 级别上发生

143.14.2. 审计命名空间

Camel 可以在评估 XPath 表达式之前发现并转储每个传入消息上存在的所有命名空间，从而为您提供所需的所有丰富的信息，以帮助您分析和固定可能的命名空间问题。

为了达到此目的，它会在内部使用另一个定制的 XPath 表达式提取消息中显示的所有命名空间映射，显示每个映射的前缀和完整命名空间 URI。

需要考虑的一些点：

- 从输出中隐藏了隐式 XML 命名空间 (xmlns:xml="http://www.w3.org/XML/1998/namespace")，因为它没有添加值
- 默认命名空间列在输出中的 DEFAULT 关键字下
- 请记住，命名空间可以在不同的范围下重新映射。考虑可分配不同命名空间或默认命名空间

更改内部元素的顶级 'a' 前缀。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 XML DSL 中启用这个选项：

Java DSL：

```
XPathBuilder.xpath("/foo:person/@id", String.class).logNamespaces()
```

XML DSL:

```
<xpath logNamespaces="true" resultType="String">/foo:person/@id</xpath>
```

审计的结果将出现在 org.apache.camel.language.xpath.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]}
```

143.15. 从外部资源载入脚本

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)
```

143.16. 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.language.xpath.document-type	文档类型的类名称是 org.w3c.dom.Document。		字符串

Name	描述	默认值	类型
camel.language.xpath.enabled	是否启用 xpath 语言的自动配置。这默认是启用的。		布尔值
camel.language.xpath.factory-ref	引用 registry 中要查找的自定义 XPathFactory。		字符串
camel.language.xpath.log-namespaces	是否记录在故障排除过程中协助的命名空间。	false	布尔值
camel.language.xpath.object-model	要使用的 XPath 对象模型。		字符串
camel.language.xpath.pre-compile	是否在初始化阶段启用预编译 xpath 表达式。默认启用预编译。这可用于关闭，例如在开始阶段需要编译阶段的情况，例如，如果应用程序在编译时（例如，camel-quarkus），这会加载构建操作系统的 xpath 工厂，而不是 JVM 运行时。	true	布尔值
camel.language.xpath.saxon	是否使用 Saxon。	false	布尔值
camel.language.xpath.thread-safety	是否为 xpath 表达式返回的结果启用 thread-safety。这适用于将 NODESET 用作结果类型，返回的集合具有多个元素。在这种情况下，如果您在并行处理模式中处理 NODESET （如从 Camel Splitter EIP）同时处理，则可能会出现线程安全的问题。这个选项通过执行节点强制副本来防止并发问题。如果您在应用程序中使用 camel-saxon 或 Saxon，则建议您打开这个选项。saxon 有 thread-safety 问题，这可以通过打开此选项来防止。	false	布尔值
camel.language.xpath.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

第 144 章 XSLT

仅支持生成者

XSLT 组件允许您使用 XSLT 模板处理消息。在使用 Templating 生成请求响应时，这是理想选择。

144.1. 依赖项

当在 Camel Spring Boot 中使用 xslt 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-xslt-starter</artifactId>
</dependency>
```

144.2. URI 格式

```
xslt:templateName[?options]
```

URI 格式包含 templateName，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&...

表 144.1. 表 1.URI 示例

URI	描述
xslt:com/acme/mytransform.xsl	在 classpath 上引用文件 com/acme/mytransform.xsl
xslt:file:///foo/bar.xsl	引用文件 /foo/bar.xsl

URI	描述
xslt:http://acme.com/cheese/foo.xsl	指的是远程 http 资源

144.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

144.3.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

144.3.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

144.4. 组件选项

XSLT 组件支持 7 个选项，如下所列。

Name	描述	默认值	类型
contentCache (producer)	加载时的资源内容（样式表文件）的缓存。如果设置为 false Camel，将在每次消息处理时重新加载风格表文件。这适用于开发。可以使用 clearCachedStylesheet 操作，强制缓存的风格表通过 JMX 在运行时重新加载。	true	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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
transformerFactoryClass (advanced)	要使用自定义 XSLT 转换器工厂，请指定为 FQN 类名称。		字符串
transformerFactoryConfigurationStrategy (advanced)	应用到新创建的 TransformerFactory 实例的配置策略。		TransformerFactoryConfigurationStrategy
uriResolver (advanced)	使用自定义 UriResolver。不应与 'uriResolverFactory' 选项一同使用。		URIResolver
uriResolverFactory (advanced)	使用自定义 UriResolver，它依赖于动态端点资源 URI。不应与 'uriResolver' 选项一同使用。		XsltUriResolverFactory

144.5. 端点选项

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：

144.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
resourceUri (producer)	**必需的**模板的路径。默认 URIResolver 支持以下内容。您可以为前缀：classpath, file, http, ref, 或 bean. classpath, file 和 http 使用这些协议加载资源 (classpath 为 default)。ref 将查询 registry 中的资源。bean 将调用要用作资源的 bean 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

144.5.2. 查询参数(13 参数)

Name	描述	默认值	类型
contentCache (producer)	加载时的资源内容（样式表文件）的缓存。如果设置为 false Camel，将在每次消息处理时重新加载风格表文件。这适用于开发。可以使用 clearCachedStylesheet 操作，强制缓存的风格表通过 JMX 在运行时重新加载。	true	布尔值
deleteOutputFile (producer)	如果您有 output=file，则此选项指定在处理 Exchange 时是否应删除输出文件。例如，假设输出文件是一个临时文件，那么在使用后，最好将其删除。	false	布尔值
failOnNullBody (producer)	如果输入正文为 null，是否抛出异常。	true	布尔值
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
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。前三个选项都基于内存，其中文件直接流传输到 java.io.File。对于文件，您必须使用密钥 Exchange.XSLT_FILE_NAME （也是 CamelXsltFileName）在 IN 标头中指定文件名。另外，任何前需要创建文件名的路径，否则会在运行时抛出异常。 Enum 值： • 字符串 • bytes • DOM • file	字符串	XsltOutput
transformerCacheSize (producer)	已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
entityResolver (advanced)	使用自定义 org.xml.sax.EntityResolver 和 javax.xml.transform.sax.SAXSource。		EntityResolver
errorListener (advanced)	允许配置为使用自定义 javax.xml.transform.ErrorListener。在执行此操作时，需要注意，默认错误监听程序捕获任何错误或致命错误，并将 Exchange 的信息存储为不使用属性。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
transformerFactory (advanced)	使用自定义 XSLT 转换器工厂。		TransformerFactory
transformerFactoryClass (advanced)	要使用自定义 XSLT 转换器工厂，请指定为 FQN 类名称。		字符串
transformerFactoryConfigurationStrategy (advanced)	应用到新创建的 TransformerFactory 实例的配置策略。		TransformerFactoryConfigurationStrategy
uriResolver (advanced)	使用自定义 javax.xml.transform.URIResolver。		URIResolver

144.6. 使用 XSLT 端点

例如，使用 XSLT 模板为 InOut 消息交换（其中有一个 JMSReplyTo 标头）消息来计算响应。

```
from("activemq:My.Queue").
  to("xslt:com/acme/mytransform.xsl");
```

如果要使用 InOnly 并使用消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");
```

144.7. 在 XSLT 中使用的参数

默认情况下，所有标头都添加为参数，然后保存在 XSLT 中。
要使参数可以被使用，您需要声明它们。

```
<setHeader name="myParam"><constant>42</constant></setHeader>
<to uri="xslt:MyTransform.xsl"/>
```

参数还需要在 XSLT 的顶级声明，以便它可用：

```
<xsl: ..... >

  <xsl:param name="myParam"/>

   <xsl:template ...>
```

144.8. SPRING XML 版本

要使用以上 Spring XML 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>
```

144.9. 使用 XSL:INCLUDE

Camel 提供自己的 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 文件将位于相对于起始端点。

```
<xsl:include href="staff_template.xsl"/>
```

这意味着 Camel 将在 classpath 中查找文件，作为 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 路径中向后引用。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"/>
```

144.10. 使用 XSL:INCLUDE 和默认前缀

Camel 将使用端点配置中的前缀作为默认前缀。

您可以明确指定 file: 或 classpath: loading。如果需要，这两种加载类型可以在 XSLT 脚本中混合使用。

144.11. 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**。

144.12. 从 XSLT ERRORLISTENER 访问警告、错误和致命错误

任何 warning/error 或 fatalError 都作为属性存储在当前 Exchange 上，带有密钥 Exchange.XSLT_ERROR、Exchange.XSLT_FATAL_ERROR 或 Exchange.XSLT_WARNING （允许

最终用户在转换过程中出现任何错误）。

例如，在下面的样式表中，如果员工有空的 dob 字段，我们希望终止。和，使用 xsl:message 包含自定义错误消息。

```
<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>
```

例外存储在 Exchange 上，作为带有密钥 Exchange.XSLT_WARNING 的警告信息。

144.13. 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.xslt.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.xslt.content-cache	加载时的资源内容（样式表文件）的缓存。如果设置为 false Camel，将在每次消息处理时重新加载风格表文件。这适用于开发。可以使用 clearCachedStylesheet 操作，强制缓存的风格表通过 JMX 在运行时重新加载。	true	布尔值
camel.component.xslt.enabled	是否启用 xslt 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.xslt.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.xslt.transformer-factory-class	要使用自定义 XSLT 转换器工厂，请指定为 FQN 类名称。		字符串
camel.component.xslt.transformer-factory-configuration-strategy	应用到新创建的 TransformerFactory 实例的配置策略。选项是一个 org.apache.camel.component.xslt.TransformerFactoryConfigurationStrategy 类型。		TransformerFactoryConfigurationStrategy
camel.component.xslt.uri-resolver	使用自定义 UriResolver。不应与 'uriResolverFactory' 选项一同使用。选项是 javax.xml.transform.URIResolver 类型。		URIResolver
camel.component.xslt.uri-resolver-factory	使用自定义 UriResolver，它依赖于动态端点资源 URI。不应与 'uriResolver' 选项一同使用。选项是一个 org.apache.camel.component.xslt.XsltUriResolverFactory 类型。		XsltUriResolverFactory

第 145 章 XSLT SAXON

Since Camel 3.0

仅支持生成者

XSLT Saxon 组件允许您使用 Saxon 的 XSLT 模板处理消息。这是使用 Templating 生成请求的响应时的理想选择。

145.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 xslt-saxon 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-xslt-saxon-starter</artifactId>
</dependency>
```

145.2. URI 格式

```
xslt-saxon:templateName[?options]
```

URI 格式包含 templateName，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&…
```

表 145.1. URI 示例

URI	描述
xslt-saxon:com/acme/mytransform.xsl	在 classpath 上引用文件 com/acme/mytransform.xsl
xslt-saxon:file:///foo/bar.xsl	引用文件 /foo/bar.xsl
xslt-saxon:http://acme.com/cheese/foo.xsl	指的是远程 http 资源

145.3. 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

145.3.1. 配置组件选项

组件级别是最高级别，它包含端点继承的常规配置。例如，一个组件可能具有安全设置、用于身份验证的凭证、用于网络连接的 url 等等。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常已配置了常用的默认值，因此通常只需要在组件上配置几个选项，或者根本不需要配置任何选项。

可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，也可直接使用 Java 代码完成。

145.3.2. 配置端点选项

您发现自己在端点上配置了一个，因为端点通常有许多选项，允许您配置您需要的端点。这些选项被分别分类为：端点作为消费者（来自）被使用，和作为生成者（到）使用，或被两者使用。

配置端点通常在端点 URI 中作为路径和查询参数直接进行。您还可以使用 Endpoint DSL 作为配置端点的安全方法。

在配置选项时，最好使用 Property Placeholders，它不允许硬编码 URL、端口号、敏感信息和其他设置。换句话说，占位符允许从您的代码外部配置，并提供更多灵活性和重复使用。

以下两节列出了所有选项，首为于组件，后跟端点。

145.4. 组件选项

XSLT Saxon 组件支持 11 个选项，如下所列。

Name	描述	默认值	类型
contentCache (producer)	加载时的资源内容（样式表文件）的缓存。如果设置为 false Camel，将在每次消息处理时重新加载风格表文件。这适用于开发。可以使用 clearCachedStylesheet 操作，强制缓存的风格表通过 JMX 在运行时重新加载。	true	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
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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
saxonConfiguration (advanced)	使用自定义 Saxon 配置。		配置
saxonConfigurationProperties (advanced)	设置自定义 Saxon 配置属性。		Map
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 中。该函数在 registry 中查找，您可以在其中使用逗号分隔多个值来查找。		字符串

Name	描述	默认值	类型
secureProcessing (advanced)	XML 安全处理功能（请参阅 javax.xml.XMLConstants）。这默认是启用的。但是，在使用 Saxon 专业时，您可能需要关闭它，以允许 Saxon 能够使用 Java 扩展功能。	true	布尔值
transformerFactoryClass (advanced)	要使用自定义 XSLT 转换器工厂，请指定为 FQN 类名称。		字符串
transformerFactoryConfigurationStrategy (advanced)	应用到新创建的 TransformerFactory 实例的配置策略。		TransformerFactoryConfigurationStrategy
uriResolver (advanced)	使用自定义 UriResolver。不应与 'uriResolverFactory' 选项一同使用。		URIResolver
uriResolverFactory (advanced)	使用自定义 UriResolver，它依赖于动态端点资源 URI。不应与 'uriResolver' 选项一同使用。		XsltUriResolverFactory

145.5. 端点选项

XSLT Saxon 端点使用 URI 语法进行配置：

```
xslt-saxon:resourceUri
```

使用以下路径和查询参数：

145.5.1. 路径参数(1 参数)

Name	描述	默认值	类型
resourceUri (producer)	**必需的**模板的路径。默认 URIResolver 支持以下内容。您可以为前缀：classpath, file, http, ref, 或 bean. classpath, file 和 http 使用这些协议加载资源 (classpath 为 default)。ref 将查询 registry 中的资源。bean 将调用要用作资源的 bean 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

145.5.2. 查询参数(18 参数)

Name	描述	默认值	类型
allowStAX (producer)	是否允许使用 StAX 作为 javax.xml.transform.Source。如果 XSLT 库支持 StAX，如 Saxon 库(camel-saxon)，您可以启用此功能。Xalan 库(JVM 中的默认)不支持 StAXSource。	true	布尔值
contentCache (producer)	加载时的资源内容（样式表文件）的缓存。如果设置为 false Camel，将在每次消息处理时重新加载风格表文件。这适用于开发。可以使用 clearCachedStylesheet 操作，强制缓存的风格表通过 JMX 在运行时重新加载。	true	布尔值
deleteOutputFile (producer)	如果您有 output=file，则此选项指定在处理 Exchange 时是否应删除输出文件。例如，假设输出文件是一个临时文件，那么在使用后，最好将其删除。	false	布尔值
failOnNullBody (producer)	如果输入正文为 null，是否抛出异常。	true	布尔值
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。前三个选项都基于内存，其中文件直接流传输到 java.io.File。对于文件，您必须使用键 XsltConstants.XSLT_FILE_NAME （也为 CamelXsltFileName）在 IN 标头中指定文件名。另外，任何前需要创建文件名的路径，否则会在运行时抛出异常。 Enum 值： • 字符串 • bytes • DOM • file	字符串	XsltOutput
transformerCacheSize (producer)	已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
lazyStartProducer (producer (advanced))	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
entityResolver (advanced)	使用自定义 org.xml.sax.EntityResolver 和 javax.xml.transform.sax.SAXSource。		EntityResolver

Name	描述	默认值	类型
errorListener (advanced)	允许配置为使用自定义 javax.xml.transform.ErrorListener。在执行此操作时，需要注意，默认错误监听程序捕获任何错误或致命错误，并将 Exchange 的信息存储为不使用属性。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。		配置
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 中。该函数在 registry 中查找，您可以在其中用逗号分隔要查找的多个值。		字符串
secureProcessing (advanced)	XML 安全处理功能（请参阅 javax.xml.XMLConstants）。这默认是启用的。但是，在使用 Saxon 专业时，您可能需要关闭它，以允许 Saxon 能够使用 Java 扩展功能。	true	布尔值
transformerFactory (advanced)	使用自定义 XSLT 转换器工厂。		TransformerFactory
transformerFactoryClass (advanced)	要使用自定义 XSLT 转换器工厂，请指定为 FQN 类名称。		字符串
transformerFactoryConfigurationStrategy (advanced)	应用到新创建的 TransformerFactory 实例的配置策略。		TransformerFactoryConfigurationStrategy
uriResolver (advanced)	使用自定义 javax.xml.transform.URIResolver。		URIResolver
xsltMessageLogger (advanced)	XSLT 转换期间生成的消息的消费者。		XsltMessageLogger

145.6. 使用 XSLT 端点

***例如，使用 XSLT 模板**为 **InOut 消息交**换（**其中有一个 JMSReplyTo** 标头）**消息来**计**算**响应。*

```
from("activemq:My.Queue").
  to("xslt-saxon:com/acme/mytransform.xsl");
```

如果要使用 InOnly 并使用消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-saxon:com/acme/mytransform.xsl").
  to("activemq:Another.Queue");
```

145.7. 在 XSLT 中使用的参数

默认情况下，所有标头都添加为参数，然后保存在 XSLT 中。要使参数可以被使用，您需要声明它们。

```
<setHeader name="myParam"><constant>42</constant></setHeader>
<to uri="xslt:MyTransform.xsl"/>
```

参数还需要在 XSLT 的顶级声明，以便它可用：

```
<xsl: ..... >

  <xsl:param name="myParam"/>

   <xsl:template ...>
```

145.8. SPRING XML 版本

要在 Spring XML 中使用上述示例，请使用以下代码：

```
<camelContext xmlns="http://activemq.apache.org/camel/schema/spring">
  <route>
    <from uri="activemq:My.Queue"/>
    <to uri="xslt-saxon:org/apache/camel/spring/processor/example.xsl"/>
    <to uri="activemq:Another.Queue"/>
  </route>
</camelContext>
```

145.9. 使用 XSL:INCLUDE

Camel 提供自己的 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 文件将位于相对于起始端点。

```
<xsl:include href="staff_template.xsl"/>
```

这意味着 Camel 将在 classpath 中查找文件，作为 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 路径中向后引用。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"/>
```

145.10. 使用 XSL:INCLUDE 和默认前缀

Camel 使用端点配置中的前缀作为默认前缀。您可以明确指定 file: 或 classpath: loading。如果需要，这两种加载类型可以在 XSLT 脚本中混合使用。

145.11. 使用 SAXON 扩展功能

从 Saxon 9.2 开始，通过新机制补充了写扩展功能，称为 集成的扩展功能。现在，您可以轻松地使用 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-saxon: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-saxon:org/apache/camel/component/xslt/extensions/extensions.xslt?
saxonExtensionFunctions=#function1,#function2"/>
  </route>
</camelContext>
```

145.12. 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。

145.13. 从 XSLT ERRORLISTENER 访问警告、错误和致命错误

任何 warning/error 或 fatalError 都作为属性存储在当前 Exchange 上，带有密钥 Exchange.XSLT_ERROR、Exchange.XSLT_FATAL_ERROR 或 Exchange.XSLT_WARNING （允许最终用户在转换过程中出现任何错误）。

例如，在下面的样式表中，我们希望确定某个员工是否有空的 dob 字段。和，使用 xsl:message 包含自定义错误消息。

```
<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>
```

例外存储在 Exchange 上，作为带有密钥 Exchange.XSLT_WARNING 的警告信息。

145.14. 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.xslt-saxon.autowired-enabled	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。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.xslt-saxon.content-cache	加载时的资源内容（样式表文件）的缓存。如果设置为 false Camel，将在每次消息处理时重新加载风格表文件。这适用于开发。可以使用 clearCachedStylesheet 操作，强制缓存的风格表通过 JMX 在运行时重新加载。	true	布尔值
camel.component.xslt-saxon.enabled	是否启用 xslt-saxon 组件的自动配置。这默认是启用的。		布尔值
camel.component.xslt-saxon.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.xslt-saxon.saxon-configuration	使用自定义 Saxon 配置。选项是一个 net.sf.saxon.Configuration 类型。		配置
camel.component.xslt-saxon.saxon-configuration-properties	设置自定义 Saxon 配置属性。		Map
camel.component.xslt-saxon.saxon-extension-functions	允许您使用自定义 net.sf.saxon.lib.ExtensionFunctionDefinition。您需要将 camel-saxon 添加到 classpath 中。该函数在 registry 中查找，您可以在其中使用逗号分隔多个值来查找。		字符串
camel.component.xslt-saxon.secure-processing	XML 安全处理功能（请参阅 javax.xml.XMLConstants）。这默认是启用的。但是，在使用 Saxon 专业时，您可能需要关闭它，以允许 Saxon 能够使用 Java 扩展功能。	true	布尔值

Name	描述	默认值	类型
camel.component.xslt-saxon.transformer-factory-class	要使用自定义 XSLT 转换器工厂，请指定为 FQN 类名称。		字符串
camel.component.xslt-saxon.transformer-factory-configuration-strategy	应用到新创建的 TransformerFactory 实例的配置策略。选项是一个 org.apache.camel.component.xslt.TransformerFactoryConfigurationStrategy 类型。		TransformerFactoryConfigurationStrategy
camel.component.xslt-saxon.uri-resolver	使用自定义 UriResolver。不应与 'uriResolverFactory' 选项一同使用。选项是 javax.xml.transform.URIResolver 类型。		URIResolver
camel.component.xslt-saxon.uri-resolver-factory	使用自定义 UriResolver，它依赖于动态端点资源 URI。不应与 'uriResolver' 选项一同使用。选项是一个 org.apache.camel.component.xslt.XsltUriResolverFactory 类型。		XsltUriResolverFactory

第 146 章 YAML DSL

自 Camel 3.9 起

YAML DSL 提供了在 YAML 中定义 Camel 路由、路由模板和 REST DSL 配置的功能。

146.1. 定义路由

路由是定义的元素集合，如下所示：

```
- from: 1
    uri: "direct:start"
    steps: 2
      - filter:
          expression:
            simple: "${in.header.continue} == true"
          steps:
            - to:
                uri: "log:filtered"
      - to:
          uri: "log:original"
```

其中，

1

默认情况下，支持路由入口点和其余路由。

2

处理步骤

注意

每个步骤都代表了一个 YAML 映射，其中有一个条目，其中字段名称是 EIP 名称。

作为常规规则，每个步骤都提供相关定义声明的所有参数，但有一些细微的区别/隔离：

- output Aware 步骤

当交换与过滤器表达式或 分割 生成的项目匹配时，一些步骤（如 filter 和 split ）具有自己的管道。您可以在 steps 字段中定义这些管道：

```
filter:
  expression:
    simple: "${in.header.continue} == true"
    steps:
      - to:
          uri: "log:filtered"
```

- 表达式 Aware 步骤

有些 EIP （如 filter 和 split ）支持通过 expression 字段定义 表达式 ：

显式 Expression 字段

```
filter:
  expression:
    simple: "${in.header.continue} == true"
```

要使 DSL 不太详细，您可以省略 expression 字段。

隐式 Expression 字段

```
filter:
  simple: "${in.header.continue} == true"
```

通常，表达式可以定义内联，例如在上述示例中，但如果您需要提供更多信息，您可以"滚动"表达式定义并配置表达式定义的任何单个参数。

完整的表达式定义

```
filter:
  tokenize:
    token: "<"
    end-token: ">"
```

- 数据格式感知步骤

EIP marshal 和 unmarshal 支持定义数据格式：

```
marshal:
  json:
    library: Gson
```

注意

如果要使用 data-format 的默认设置，则需要将空块作为数据格式参数设置为数据格式参数，如 json: {}

146.2. 定义端点

要使用 YAML DSL 定义端点，有两个选项：

- 使用经典 Camel URI：

```
- from:
    uri: "timer:tick?period=1s"
    steps:
      - to:
          uri: "telegram:bots?authorizationToken=XXX"
```

- 使用 URI 和参数：

```
- from:
    uri: "timer://tick"
    parameters:
      period: "1s"
    steps:
      - to:
          uri: "telegram:bots"
          parameters:
            authorizationToken: "XXX"
```

146.3. 定义 BEAN

除了创建 Camel Main 提供的 Bean 的一般支持外，YAML DSL 提供了便捷的语法来定义和配置它们：

```
- beans:
  - name: beanFromMap (1)
    type: com.acme.MyBean (2)
    properties: (3)
      foo: bar
```

其中，

(1) 将实例绑定到 Camel Registry 的 bean 名称。

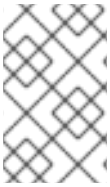
(2) bean 的完全限定类名称

(3) 要设置的 bean 的属性

可以使用映射或属性样式定义 bean 的属性，如下例所示：

```
- beans:
  # map style
  - name: beanFromMap
    type: com.acme.MyBean
    properties:
      field1: 'f1'
      field2: 'f2'
```

```
    nested:
      field1: 'nf1'
      field2: 'nf2'
  # properties style
  - name: beanFromProps
    type: com.acme.MyBean
    properties:
      field1: 'f1_p'
      field2: 'f2_p'
      nested.field1: 'nf1_p'
      nested.field2: 'nf2_p'
```



注意

Bean 元素仅用作 root 元素。

146.4. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

146.4.1. 组件级别选项

组件级别是最高级别。您在此级别上定义的配置由所有端点继承。例如，一个组件可以具有安全设置、用于身份验证的凭证、用于网络连接的 url，等等。

因为组件通常会为最常见的情况预先配置了默认值，因此您可能需要配置几个组件选项，或者根本都不需要配置任何组件选项。

您可以在配置文件(application.properties|yaml)中使用 组件 DSL 配置组件，或使用 Java 代码直接配置组件。

146.4.2. 端点级别选项

在 Endpoint 级别，您可以使用多个选项来配置您希望端点执行的操作。这些选项根据端点是否用作消费者（来自）或作为生成者(to)用于两者的分类。

您可以直接在端点 URI 中配置端点作为 路径和 查询参数。您还可以使用 Endpoint DSL 和 DataFormat DSL 作为在 Java 中配置端点和数据格式的安全 方法。

在配置选项时，对 urls、端口号、敏感信息和其他设置使用 Property Placeholders。

占位符允许您从代码外部化配置，为您提供更灵活且可重复使用的代码。

146.5. 在语言上配置选项

有些 语言 有可能需要使用的额外配置。

例如，可以将 JSONPath 配置为忽略 JSON 解析错误。当您使用基于内容的路由，并希望 将消息 路由到不同的端点时，这是目的。消息的 JSON 有效负载可以采用不同的形式，这意味着在某些情况下 JSonPath 表达式将失败，但其他时间不是。在这种情况下，您必须将 suppress-exception 设置为 true，如下所示：

```
- from:
    uri: "direct:start"
    steps:
      - choice:
          when:
          - jsonpath:
              expression: "person.middlename"
              suppress-exceptions: true
            steps:
            - to: "mock:middle"
          - jsonpath:
              expression: "person.lastname"
              suppress-exceptions: true
            steps:
            - to: "mock:last"
          otherwise:
            steps:
              - to: "mock:other"
```

在上面的路由中，以下消息会失败 JSonPath 表达式 person.middlename，因为 JSON 有效负载没有中间名称字段。为补救这一点，我们隐藏了例外。

```
{
  "person": {
    "firstname": "John",
```

```
    "lastname": "Doe"
  }
}
```

146.6. 外部示例

您可以在 Camel 示例中找到使用 main-yaml 的示例，以演示如何使用 YAML 创建 Camel 路由。您还可以引用使用 YAML 定义每个 Kamelet 的 Camel Kamelets。

第 147 章 ZIP 文件

Zip File Data Format 是消息压缩和解压缩格式。消息可以被放入包含单个条目的 Zip 文件，而包含单个条目的 Zip 文件可以编译到原始文件内容。只要使用 Java 7 或更高版本，这个数据格式支持 ZIP64。

147.1. 依赖项

当在 Red Hat build of Camel Spring Boot 中使用 zipfile 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-zipfile-starter</artifactId>
</dependency>
```

147.2. 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		布尔值	如果 zip 文件有更多条目，则此选项设置为 true，允许使用分割器 EIP，在流传输模式中使用迭代器来分割数据。
allowEmptyDirectory		布尔值	如果 zip 文件有更多条目，请将这个选项设置为 true，允许获取迭代器，即使目录为空。
preservePathElements		布尔值	如果文件名包含 path 元素，请将这个选项设置为 true，则允许在 zip 文件中维护路径。
maxDecompressedSize		整数	设置 zip 文件的最大解压缩大小（以字节为单位）。如果没有指定默认值，则默认值对应于 1GB。如果解压缩的大小超过这个数量，则会抛出 IOException。设置为 -1，以禁用设置最大解压缩大小。

147.3. MARSHAL

在本例中，我们使用 Zip 文件压缩将常规文本/XML 有效负载放入压缩的有效负载，并将其发送到名为 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 CamelFileName 消息标头的值，后缀为 ".zip"。例如，如果以下路由在输入目录中找到一个名为 "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 通常是一个唯一生成的 ID，因此您将以 ID-MACHINENAME-2443-1211718892437-1-0.zip 等文件名结束。如果要覆盖此行为，您可以在路由中明确设置 CamelFileName 标头的值：

```
from("direct:start")
    .setHeader(Exchange.FILE_NAME, constant("report.txt"))
    .marshal().zipFile()
    .to("file:output/directory");
```

此路由会在输出目录中生成名为 "report.txt.zip" 的 Zip 文件，其中包含一个名为 "report.txt" 的单个 Zip 条目。

147.4. UNMARSHAL

在本例中，我们将名为 MY_QUEUE 的 ActiveMQ 队列的 Zip 文件有效负载转发到其原始格式，并将其处理到 UnZippedMessageProcessor。

```
from("activemq:queue:MY_QUEUE")
    .unmarshal().zipFile()
    .process(new UnZippedMessageProcessor());
```

如果 zip 文件有更多条目，则 ZipFileDataFormat 的 useIterator 选项为 true，您可以使用 splitter 来进一步工作。

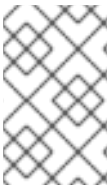
```
ZipFileDataFormat zipFile = new ZipFileDataFormat();
zipFile.setUsingIterator(true);

from("file:src/test/resources/org/apache/camel/dataformat/zipfile/?delay=1000&noop=true")
    .unmarshal(zipFile)
```

```
    .split(body(Iterator.class)).streaming()
        .process(new UnZippedMessageProcessor())
    .end();
```

或者您可以使用 ZipSplitter 作为分割器的表达式，如下所示

```
from("file:src/test/resources/org/apache/camel/dataformat/zipfile?delay=1000&noop=true")
    .split(new ZipSplitter()).streaming()
        .process(new UnZippedMessageProcessor())
    .end();
```

147.4.1. 聚合



注意

此聚合策略需要 eager 补全检查才能正常工作。

在本例中，我们将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");
```

传出的 CamelFileName 消息标头使用 java.io.File.createTempFile （带有 ".zip" 后缀）创建。如果要覆盖此行为，您可以在路由中明确设置 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");
```

147.5. 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.zipfile.allow-empty-directory	如果 zip 文件有更多条目，请将这个选项设置为 true，允许获取迭代器，即使目录为空。	false	布尔值
camel.dataformat.zipfile.enabled	是否启用 zipfile 数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.zipfile.max-decompressed-size	设置 zip 文件的最大解压缩大小（以字节为单位）。如果没有指定默认值，则默认值对应于 1GB。如果解压缩的大小超过这个数量，则会抛出 IOException。设置为 -1，以禁用设置最大解压缩大小。	1073741824	Long
camel.dataformat.zipfile.preserve-path-elements	如果文件名包含 path 元素，请将这个选项设置为 true，则允许在 zip 文件中维护路径。	false	布尔值
camel.dataformat.zipfile.using-iterator	如果 zip 文件有更多条目，则此选项设置为 true，允许使用分割器 EIP，在流传输模式中使用迭代器来分割数据。	false	布尔值