Red Hat build of Apache Camel for Spring Boot 3.20

Camel Spring Boot 参考

Camel Spring Boot 参考

Last Updated: 2024-06-14

Red Hat build of Apache Camel for Spring Boot 3.20 Camel Spring Boot 参考

Camel Spring Boot 参考

法律通告

摘要

本指南描述了 Camel Spring Boot 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。有关更多详情，请参阅我们的首席技术官 Chris Wright 提供的消息。

第 1 章 AMQP

从 Camel 1.2 开始

支持生成者和消费者

AMQP 组件支持使用 AMQP 项目的 JMS Client API 支持 AMQP 1.0 协议 。http://qpid.apache.org/

Maven 用户需要将以下依赖项添加到其 **pom.xml** 中。

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

1.1. URI 格式

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

1.2. 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

1.2.1. 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。

1.2.2. 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。

1.3. 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。它通常只适用于持久主题订阅。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。必须在组件或端点上配置连接工厂。		ConnectionFactory
disableReplyTo (common)	指定 Camel 是否忽略消息中的 JMSReplyTo 标头。如果为 true，Camel 不会向 JMSReplyTo 标头中指定的目的地发送回复。如果您希望 Camel 从路由中消耗，并且您不希望 Camel 自动发送回复消息，则可以使用此选项，因为代码中的另一个组件处理回复消息。如果要在不同的消息代理之间将 Camel 用作代理，并且希望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可配置订阅者名称。还必须配置 clientId 选项。		字符串
includeAmqpAnnotations (common)	从 AMQP 到 Camel 消息映射时是否包含 AMQP 注解。把它设置为 true 会映射包含 JMS_AMQP_MA_ 前缀的 AMQP 消息注解到消息标头。由于 Apacheqpid JMS API 中的限制，当前交付注解将被忽略。	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 会决定要从 In body 类型使用哪个 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（覆盖消费者中 Message.getJMSReplyTo （）的所有传入值）。		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 用户与 JMS 代理的有效连接。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 producer 也经过测试。	false	布尔值
acknowledgementModeName (consumer)	JMS 确认名称，其为： 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消息处于活跃状态时收到消息。通常，连接到队列的活动消费者以轮循方式从它接收消息。当使用消费者优先级时，如果有多个具有相同高优先级的活跃用户，则消息将进行循环发送。只有高优先级消费者没有可用的信用消息或高优先级消费者接受消息时，消息才会降低优先级较低的消费者（例如，它不符合与消费者关联的任何选择器的条件）。		int
asyncConsumer (consumer)	JmsConsumer 是否异步处理交换。如果启用，JmsConsumer 可以从 JMS 队列中提取下一个消息，而上一个消息则异步处理（通过 Asynchronous Routing Engine）。这意味着消息可能没有完全严格按照顺序进行处理。如果禁用（作为默认），则在 JmsConsumer 会从 JMS 队列中提取下一个消息前，会完全处理交换。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 必须支持 async 事务)。	false	布尔值
autoStartup (consumer)	指定消费者容器是否应自动启动。	true	布尔值
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)	根据底层 JMS 资源的名称设置缓存级别。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 消耗时的默认并发消费者数量（而不是通过 JMS 请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。当通过 JMS 执行 request/reply 时，选项 replyToConcurrentConsumers 用于控制回复消息监听器上的并发消费者数量。	1	int
maxConcurrentConsumers (consumer)	指定从 JMS 消耗时的最大并发消费者数（而不是通过 JMS 请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。当通过 JMS 执行请求/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交付进行回复。	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。可使用的 durable 订阅名称通过 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 域)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)	设置要创建的订阅名称。在带有共享或可升级订阅的主题（公共域）中应用。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： 每个订阅都只允许 1 个并发消费者（这是此消息侦听器容器的默认值），但一个共享订阅（需要 JMS 2.0）除外。		字符串
subscriptionShared (consumer)	设置是否共享订阅。可以使用的共享订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。请注意，共享的订阅也可能是危险的，因此此标志也可以与订阅相结合。仅在侦听主题(pub-sub 域)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，则可能会考虑启用此选项，同时仍然在队列中排队消息。如果此选项为 false，并且您停止 JMS 路由，则消息可能会被拒绝，并且 JMS 代理必须尝试 redeliveries （但可能会再次拒绝），最终消息可能会移到 JMS 代理上的死信队列中。要避免这种情况，建议启用这个选项。	false	布尔值
allowReplyManagerQuickStop (consumer (advanced))	是否启用请求管理器中使用的 DefaultMessageListenerContainer，允许 DefaultMessageListenerContainer.runningAllowed 标志在 JmsConfigurationVirtualMachineisAcceptMessagesWhileStopping 时快速停止，并且 org.apache.camel.CamelContext 当前已停止。在常规 JMS 用户中默认启用这种快速停止功能，但要为回复管理器启用这个标志。	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。消费者类型决定要使用的 Spring JMS 侦听器。默认将使用 org.springframework.jms.listener.DefaultMessageListenerContainer，Simple 将使用 org.springframework.jms.listener.SimpleMessageListenerContainer。指定 Custom 时，由 messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 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))	指定在 DefaultMessageListenerContainer 中使用哪些默认 TaskExecutor 类型，用于消费者端点和制作者端点的 ReplyTo consumer。可能的值： SimpleAsync（使用 Spring 的 SimpleAsyncTaskExecutor）或 ThreadPool （使用 Spring 的 ThreadPoolTaskExecutor 带有最佳值 - 缓存线程池）。如果没有设置，则默认为之前的行为，它将缓存线程池用于消费者端点，而 SimpleAsync 用于回复用户。建议使用 ThreadPool 来减少弹性配置中线程垃圾箱，同时动态增加和减少并发用户。 Enum 值： ● ThreadPool ● SimpleAsync		DefaultTaskExecutorType
eagerLoadingOfProperties (consumer (advanced))	加载消息时立即启用 JMS 属性和有效负载的 eager 加载，因为 JMS 属性可能并不是必需的，但有时可能会捕获与底层 JMS 提供程序和使用 JMS 属性的早期问题。另请参阅选项 eagerPoisonBody。	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)是 poison （无法读取/映射），然后将这个文本设置为消息正文，因此可以处理消息(poison 的原因)已作为交换异常保存。这可以通过设置 eagerPoisonBody=false 来关闭。另请参阅 eagerLoadingOfProperties 选项。	Poison JMS 消息，因为 $\{exception.message}	字符串

Name	描述	默认值	类型
exposeListenerSession (consumer (advanced))	指定在消耗消息时是否应公开监听程序会话。	false	布尔值
replyToConsumerType (consumer (advanced))	回复消费者的消费者类型（当执行请求/回复时），可以是 Simple、Default 或 Custom 之一。消费者类型决定要使用的 Spring JMS 侦听器。默认将使用 org.springframework.jms.listener.DefaultMessageListenerContainer，Simple 将使用 org.springframework.jms.listener.SimpleMessageListenerContainer。指定 Custom 时，由 messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 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))	JMS 使用者是否允许向消费者使用的同一目的地发送回复消息。这可防止出现无限循环，并通过消耗并向自己发送相同的消息。	false	布尔值
taskExecutor (consumer (advanced))	允许您指定自定义任务执行器以使用消息。		TaskExecutor
deliveryDelay (producer)	设置用于发送 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。可能的值是由 javax.jms.DeliveryMode 定义的值。NON_PERSISTENT = 1 和 PERSISTENT = 2。 Enum 值： ● 1 ● 2		整数
deliveryPersistent (producer)	指定默认使用持久性交付。	true	布尔值

Name	描述	默认值	类型
explicitQosEnabled (producer)	设置在发送消息时使用 deliveryMode、priority 或 timeToLive 数量服务。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用到当前的端点。这与 preserveMessageQos 选项（按消息粒度运行）相反，从 Camel In 消息标头中读取 QoS 属性。	false	布尔值
formatDateHeadersToIso8601 (producer)	根据 ISO 8601 标准设置 JMS 日期属性是否应格式化。	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)	如果要使用消息中指定的 QoS 设置来发送消息，而不是 JMS 端点上的 QoS 设置。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供全部或仅提供其中一些。如果没有提供，Camel 将回退到使用端点中的值。因此，在使用此选项时，标头会覆盖端点中的值。与之相反，clearQosEnabled 选项将仅使用端点上设置的选项，而不使用来自消息标头中的值。	false	布尔值
priority (producer)	大于 1 的值指定发送时的消息优先级（其中 1 是最低优先级，9 是最高）。必须启用 explicitQosEnabled 选项才能使此选项生效。 Enum 值： • 1 • 2 • 3 • 4 • 5 • 6 • 7 • 8 • 9	4	int

Name	描述	默认值	类型
replyToConcurrentConsumers (producer)	指定执行请求/通过 JMS 回复时的默认并发消费者数量。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。	1	int
replyToMaxConcurrentConsumers (producer)	指定在通过 JMS 使用请求/回复时的最大并发消费者数。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。		int
replyToOnTimeoutMaxConcurrentConsumers (producer)	指定使用请求/通过 JMS 时超时时继续路由的最大并发消费者数。	1	int
replyToOverride (producer)	在 JMS 消息中提供显式 ReplyTo 目的地，这将覆盖 replyTo 的设置。如果要将消息转发到远程队列，并从 ReplyTo 目的地接收回复消息，这非常有用。		字符串
replyToType (producer)	允许明确指定在执行 request/reply 时要用于 replyTo 队列的策略类型。可能的值有：Temporary、share 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的队列。如需了解更多详细信息，请参阅 Camel JMS 文档，特别是在集群环境中运行时的影响的备注，以及共享回复队列的性能低于其 alternatives Temporary 和 Exclusive。 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 来覆盖此端点配置的超时值，因此具有每个消息独立的超时值。另请参阅 requestTimeoutCheckerInterval 选项。	20000	long
timeToLive (producer)	发送消息时，指定消息的时间到时间（以毫秒为单位）。	-1	long
allowAdditionalHeaders (producer (advanced))	此选项用于允许其他标头，这些标头可能具有根据 JMS 规范无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 包含字节数组或其他无效类型的值来执行此操作。您可以用逗号指定多个标头名称，并用作通配符匹配的后缀。		字符串

Name	描述	默认值	类型
allowNullBody (producer (advanced))	是否允许在没有正文的情况下发送消息。如果此选项为 false，且消息正文为 null，则会抛出 JMSException。	true	布尔值
alwaysCopyMessage (producer (advanced))	如果为 true，则 Camel 始终会在消息传递给发送的制作者时生成 JMS 消息副本。在某些情况下需要复制消息，如设置 replyToDestinationSelectorName 时（如果设置了 replyToDestinationSelectorName，则 Camel 会将 alwaysCopyMessage 选项设置为 true）。	false	布尔值
correlationProperty (producer (advanced))	当使用 InOut 交换模式时，使用此 JMS 属性而不是 JMSCorrelationID JMS 属性来关联消息。如果设置消息仅针对此属性 JMSCorrelationID 属性的值关联，则将忽略且未由 Camel 设置。		字符串
disableTimeToLive (producer (advanced))	使用这个选项强制禁用时间。例如，当您通过 JMS 进行请求/回复时，Camel 默认使用 requestTimeout 值作为发送的消息的时间。问题是，发送者和接收器系统必须同步其时钟，因此它们正在同步。这并非始终易于归档。因此，您可以使用 disableTimeToLive=true 来不设置发送消息上的生存时间。然后，消息不会在接收器系统中过期。如需了解更多详细信息，请参见以下部分关于生存时间。	false	布尔值
forceSendOriginalMessage (producer (advanced))	当使用 mapJmsMessage=false Camel 时，如果在路由中涉及标头(get 或 set)，则会创建一个新的 JMS 消息来发送到新的 JMS 目的地。将此选项设置为 true，以强制 Camel 发送收到的原始 JMS 消息。	false	布尔值
includeSentJMSMessageID (producer (advanced))	仅在使用 InOnly 发送到 JMS 目的地时适用（例如触发和忘记）。启用此选项将增强 Camel Exchange 与 JMS 客户端在消息发送到 JMS 目的地时使用的实际 JMSMessageID。	false	布尔值

Name	描述	默认值	类型
replyToCacheLevelName (producer (advanced))	在执行请求/通过 JMS 时，按名称为回复消费者设置缓存级别。这个选项仅在使用固定回复队列（而非临时）时才适用。Camel 默认将使用：CACHE_CONSUMER 用于专用或共享的 w/ replyToSelectorName。和 CACHE_SESSION 用于没有 replyToSelectorName 的共享。IBM WebSphere 等 JMS 代理可能需要设置 replyToCacheLevelName=CACHE_NONE 才能正常工作。注：如果使用临时队列，则不允许使用更高的值，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，以便您可以在使用共享队列（也就是说，如果您不使用临时回复队列）时过滤来自其他回复的回复。		字符串
streamMessageTypeEnabled (producer (advanced))	设置 StreamMessage 类型是否已启用。流类型的消息有效负载（如 files、InStream 等）将通过作为 BytesMessage 或 StreamMessage 发送。此选项控制将使用哪种类型。默认情况下，使用 BytesMessage 来强制整个消息有效负载读取到内存中。通过启用此选项，消息有效负载以块的形式读取到内存中，然后每个块都会写入 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 自动发现 ConnectionFactory。如果只找到一个 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 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

Name	描述	默认值	类型
allowSerializedHeaders (advanced)	控制是否包含序列化标头。仅在 transferExchange 为 true 时才适用。这要求对象可以序列化。Camel 将排除任何不可序列化的对象，并将它记录在 WARN 级别。	false	布尔值
artemisStreamingEnabled (advanced)	是否针对 Apache Artemis 流模式进行优化。这可减少使用带有 JMS StreamMessage 类型的 Artemis 时的内存开销。只有在使用 Apache Artemis 时，才必须启用这个选项。	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 无法获得连接到远程 JMS 代理的连接，那么在重试和/或故障切换时可能会阻断它。这将使 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 使用异步模式的专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果没有建立连接，则会在 WARN 级别记录异常，使用者将无法接收消息；然后，您可以重启路由来重试。	false	布尔值
asyncStopListener (advanced)	在停止路由时，是否异步停止 JmsConsumer 消息监听程序。	false	布尔值
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
配置 （高级）	使用共享的 JMS 配置。		JmsConfiguration
destinationResolver (advanced)	一个可插拔的 org.springframework.jms.support.destination.DestinationResolver，供您使用自己的解析器（例如，在 JNDI registry 中查找实际目的地）。		DestinationResolver
errorHandler (advanced)	指定在处理消息时引发任何未发现异常时要调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置错误处理程序，则这些例外将在 WARN 级别记录。您可以配置日志记录级别，以及是否应使用 errorHandlerLoggingLevel 和 errorHandlerLogStack Trace 选项记录堆栈追踪。这样可以更容易配置，而不是需要对自定义错误处理程序进行代码。		ErrorHandler
exceptionListener (advanced)	指定针对任何底层 JMS 异常通知的 JMS Exception Listener。		ExceptionListener

Name	描述	默认值	类型
idleConsumerLimit (advanced)	指定任何给定时间允许闲置的用户数量的限值。	1	int
idleTaskExecutionLimit (advanced)	指定接收任务闲置执行的限制，不会在其执行中收到任何消息。如果达到这个限制，任务将关闭并将接收给其他执行任务（在动态调度时；请参阅 maxConcurrentConsumers 设置）。Spring 提供了额外的文档。	1	int
includeAllJMSXProperties (advanced)	在从 JMS 到 Camel Message 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，则不会应用这个选项。	false	布尔值
jmsKeyFormatStrategy (advanced)	编码和解码 JMS 密钥的可插拔策略，以便它们符合 JMS 规范。Camel 提供两个开箱即用的实现：default 和 passthrough。默认策略将安全地使用句点和连字符 (. 和 -)。passthrough 策略将密钥保留原样。可用于不负责 JMS 标头密钥是否包含非法字符的 JMS 代理。您可以自行提供 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 sVirt 表示法引用它。 Enum 值： • default • passthrough		JmsKeyFormatStrategy
mapJmsMessage (advanced)	指定 Camel 是否应该自动将收到的 JMS 消息映射到合适的有效负载类型，如 javax.jms.TextMessage 到 String 等。	true	布尔值
maxMessagesPerTask (advanced)	每个任务的消息数量。-1 代表没有限制。如果您为并发消费者使用范围（例如 min max），则可以使用此选项将值设为 100，以控制消费者在需要较少的工作时可以缩小的速度。	-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)	使用给定的 MessageCreatedStrategy，当 Camel 发送 JMS 消息时，Camel 创建 javax.jms.Message 对象的新实例。		MessageCreatedStrategy

Name	描述	默认值	类型
messageIdEnabled (advanced)	发送时，指定是否应添加消息 ID。这只是对 JMS 代理的提示。如果 JMS 提供程序接受此提示，则这些消息必须将消息 ID 设置为 null；如果提供程序忽略提示，则必须将消息 ID 设置为其普通唯一值。	true	布尔值
messageListenerContainerFactory (advanced)	MessageListenerContainerFactory 的 registry ID，用于决定要使用消息的 org.springframework.jms.listener.AbstractMessageListenerContainer。设置此项将自动将 consumerType 设置为 Custom。		MessageListenerContainerFactory
messageTimestampEnabled (advanced)	指定在发送消息时是否默认启用时间戳。这只是对 JMS 代理的提示。如果 JMS 提供程序接受此提示，则这些消息必须将时间戳设置为零；如果提供程序忽略提示，则必须将时间戳设置为其正常值。	true	布尔值
pubSubNoLocal (advanced)	指定是否禁止自己连接发布的消息的发送。	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，即当连接被刷新时，以毫秒为单位。默认值为 5000 ms，即 5 秒。	5000	long
requestTimeoutCheckerInterval (advanced)	配置 Camel 在执行请求/通过 JMS 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，如果发生超时时，您必须更快地响应，那么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
Sync (advanced)	设置是否应严格使用同步处理。	false	布尔值
transferException (advanced)	如果启用了且您使用 Request Reply messaging (InOut)，并且 Exchange 失败在消费者端，则原因例外将作为 javax.jms.ObjectMessage 发回的响应。如果客户端是 Camel，则返回的例外将重新箭头。这样，您可以使用 Camel JMS 作为路由中的桥接 - 例如，使用持久性队列来启用强大的路由。请注意，如果您也启用了 transferExchange，这个选项将具有优先权。caught 异常需要可以被序列化。消费者端的原始例外可以嵌套在外部异常中，如 org.apache.camel.RuntimeCamelException。请谨慎使用它，因为数据正在使用 Java 对象序列化，要求接收者在类级别反序列化数据，这会强制在生产者和消费者之间进行强校准。	false	布尔值

Name	描述	默认值	类型
transferExchange (advanced)	您可以在有线线上传输交换，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 header, Exchange properties, exchange exception.这要求对象可以序列化。Camel 将排除任何不可序列化的对象，并将它记录在 WARN 级别。您必须在制作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。请谨慎使用它，因为数据正在使用 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，这会强制在需要使用兼容 Camel 版本的生产者和消费者之间进行强大的协调。	false	布尔值
useMessageIDAsCorrelationID (advanced)	指定 JMSMessageID 是否应该始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值
waitForProvisionCorrelationToBeUpdatedCounter (advanced)	在执行 request/reply over JMS 以及启用了 useMessageIDAsCorrelationID 时，等待 provisional correlation id 被更新到实际关联 ID 的次数。	50	int
waitForProvisionCorrelationToBeUpdatedThreadSleepingTime (advanced)	等待置备关联 ID 时每次休眠的时间间隔（以秒为单位）。	100	long
HeaderFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 将标头过滤到或从 Camel 消息过滤。		HeaderFilterStrategy
errorHandlerLoggingLevel (logging)	允许为日志记录 uncaught 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)	允许通过默认错误处理程序来控制是否应记录 stacktraces。	true	布尔值

Name	描述	默认值	类型
密码 （安全）	与 ConnectionFactory 一起使用的密码。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
用户名 （安全）	与 ConnectionFactory 一起使用的用户名。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
Transacted (transaction)	指定是否使用转换模式。	false	布尔值
TransactedInOut (transaction)	指定 InOut 操作（请求回复）是否默认使用 transacted 模式，如果此标志被设置为 true，则 Spring JmsTemplate 将把 sessionTransacted 设置为 true，而 confirmMode 作为转换用于 InOut 操作。请注意：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，容器对这些值做出自己的决策。类似地，这些参数不会考虑本地管理的事务，因为 Spring JMS 在这种情况下在现有 JMS 会话上运行。在受管事务外运行时，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（除 XA 事务之外）时同步的本地 JMS 事务。这与主事务一起管理本地 JMS 事务（可能是原生 JDBC 事务）的影响，在主事务后 JMS 事务提交右侧的 JMS 事务。	false	布尔值
lazyCreateTransactionManager (transaction (advanced))	如果为 true，Camel 将创建一个 JmsTransactionManager，如果没有在选项 transacted=true 时注入 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))	如果使用转换模式，事务的超时值（以秒为单位）。	-1	int

1.4. 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：

1.4.1. 路径参数(2 参数)

Name	描述	默认值	类型
destinationType (common)	要使用的目标种类。 Enum 值： • queue • topic • temp-queue • temp-topic	queue	字符串
destinationName (common)	用作目的地的队列或主题 **所需的** 名称。		字符串

1.4.2. 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。它通常只适用于持久主题订阅。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。必须在组件或端点上配置连接工厂。		ConnectionFactory
disableReplyTo (common)	指定 Camel 是否忽略消息中的 JMSReplyTo 标头。如果为 true，Camel 不会向 JMSReplyTo 标头中指定的目的地发送回复。如果您希望 Camel 从路由中消耗，并且您不希望 Camel 自动发送回复消息，则可以使用此选项，因为代码中的另一个组件处理回复消息。如果要在不同的消息代理之间将 Camel 用作代理，并且希望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可配置订阅者名称。还必须配置 clientId 选项。		字符串

Name	描述	默认值	类型
jmsMessageType (common)	允许您强制使用特定的 javax.jms.Message 实现来发送 JMS 消息。可能的值有：Bytes, Map, Object, Stream, text。默认情况下，Camel 会决定要从 In body 类型使用哪个 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（覆盖消费者中 Message.getJMSReplyTo （）的所有传入值）。		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 用户与 JMS 代理的有效连接。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 producer 也经过测试。	false	布尔值
acknowledgementModeName (consumer)	JMS 确认名称，其为： 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消息处于活跃状态时收到消息。通常，连接到队列的活动消费者以轮循方式从它接收消息。当使用消费者优先级时，如果有多个具有相同高优先级的活跃用户，则消息将进行循环发送。只有高优先级消费者没有可用的信用消息或高优先级消费者接受消息时，消息才会降低优先级较低的消费者（例如，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 是否异步处理交换。如果启用，JmsConsumer 可以从 JMS 队列中提取下一个消息，而上一个消息则异步处理（通过 Asynchronous Routing Engine）。这意味着消息可能没有完全严格按照顺序进行处理。如果禁用（作为默认），则在 JmsConsumer 会从 JMS 队列中提取下一个消息前，会完全处理交换。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 必须支持 async 事务)。	false	布尔值
autoStartup (consumer)	指定消费者容器是否应自动启动。	true	布尔值
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)	根据底层 JMS 资源的名称设置缓存级别。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 消耗时的默认并发消费者数量（而不是通过 JMS 请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。当通过 JMS 执行 request/reply 时，选项 replyToConcurrentConsumers 用于控制回复消息监听器上的并发消费者数量。	1	int
maxConcurrentConsumers (consumer)	指定从 JMS 消耗时的最大并发消费者数（而不是通过 JMS 请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。当通过 JMS 执行请求/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

Name	描述	默认值	类型
replyToDeliveryPersistent (consumer)	指定是否默认使用持久性交付进行回复。	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。可使用的 durable 订阅名称通过 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 域)时有意义，因此此方法也会切换 pubSubDomain 标志。	false	布尔值
subscriptionName (consumer)	设置要创建的订阅名称。在带有共享或可升级订阅的主题（公共域）中应用。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： 每个订阅都只允许 1 个并发消费者（这是此消息侦听器容器的默认值），但一个共享订阅（需要 JMS 2.0）除外。		字符串
subscriptionShared (consumer)	设置是否共享订阅。可以使用的共享订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。请注意，共享的订阅也可能是危险的，因此此标志也可以与订阅相结合。仅在侦听主题(pub-sub 域)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，则可能会考虑启用此选项，同时仍然在队列中排队消息。如果此选项为 false，并且您停止 JMS 路由，则消息可能会被拒绝，并且 JMS 代理必须尝试 redeliveries （但可能会再次拒绝），最终消息可能会移到 JMS 代理上的死信队列中。要避免这种情况，建议启用这个选项。	false	布尔值
allowReplyManagerQuickStop (consumer (advanced))	是否启用请求管理器中使用的 DefaultMessageListenerContainer，允许 DefaultMessageListenerContainer.runningAllowed 标志在 JmsConfigurationVirtualMachineisAcceptMessagesWhileStopping 时快速停止，并且 org.apache.camel.CamelContext 当前已停止。在常规 JMS 用户中默认启用这种快速停止功能，但要为回复管理器启用这个标志。	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。消费者类型决定要使用的 Spring JMS 侦听器。默认将使用 org.springframework.jms.listener.DefaultMessageListenerContainer，Simple 将使用 org.springframework.jms.listener.SimpleMessageListenerContainer。指定 Custom 时，由 messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 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))	指定在 DefaultMessageListenerContainer 中使用哪些默认 TaskExecutor 类型，用于消费者端点和制作者端点的 ReplyTo consumer。可能的值： SimpleAsync （使用 Spring 的 SimpleAsyncTaskExecutor）或 ThreadPool （使用 Spring 的 ThreadPoolTaskExecutor 带有最佳值 - 缓存线程池）。如果没有设置，则默认为之前的行为，它将缓存线程池用于消费者端点，而 SimpleAsync 用于回复用户。建议使用 ThreadPool 来减少弹性配置中线程垃圾箱，同时动态增加和减少并发用户。 Enum 值： • ThreadPool • SimpleAsync		DefaultTaskExecutorType
eagerLoadingOfProperties (consumer (advanced))	加载消息时立即启用 JMS 属性和有效负载的 eager 加载，因为 JMS 属性可能并不是必需的，但有时可能会捕获与底层 JMS 提供程序和使用 JMS 属性的早期问题。另请参阅选项 eagerPoisonBody。	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)是 poison （无法读取/映射），然后将这个文本设置为消息正文，因此可以处理消息(poison 的原因)已作为交换异常保存。这可以通过设置 eagerPoisonBody=false 来关闭。另请参阅 eagerLoadingOfProperties 选项。	Poison JMS 消息，因为 $\{exception.message}	字符串

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
exposeListenerSession (consumer (advanced))	指定在消耗消息时是否应公开监听程序会话。	false	布尔值
replyToConsumerType (consumer (advanced))	回复消费者的消费者类型（当执行请求/回复时），可以是 Simple、Default 或 Custom 之一。消费者类型决定要使用的 Spring JMS 侦听器。默认将使用 org.springframework.jms.listener.DefaultMessageListenerContainer，Simple 将使用 org.springframework.jms.listener.SimpleMessageListenerContainer。指定 Custom 时，由 messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 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))	JMS 使用者是否允许向消费者使用的同一目的地发送回复消息。这可防止出现无限循环，并通过消耗并向自己发送相同的消息。	false	布尔值
taskExecutor (consumer (advanced))	允许您指定自定义任务执行器以使用消息。		TaskExecutor
deliveryDelay (producer)	设置用于发送 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。可能的值是由 javax.jms.DeliveryMode 定义的值。NON_PERSISTENT = 1 和 PERSISTENT = 2。 Enum 值： ● 1 ● 2		整数
deliveryPersistent (producer)	指定默认使用持久性交付。	true	布尔值
explicitQosEnabled (producer)	设置在发送消息时使用 deliveryMode、priority 或 timeToLive 数量服务。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用到当前的端点。这与 preserveMessageQos 选项（按消息粒度运行）相反，从 Camel In 消息标头中读取 QoS 属性。	false	布尔值
formatDateHeadersToIso8601 (producer)	根据 ISO 8601 标准设置 JMS 日期属性是否应格式化。	false	布尔值
preserveMessageQos (producer)	如果要使用消息中指定的 QoS 设置来发送消息，而不是 JMS 端点上的 QoS 设置。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供全部或仅提供其中一些。如果没有提供，Camel 将回退到使用端点中的值。因此，在使用此选项时，标头会覆盖端点中的值。与之相反，clearQosEnabled 选项将仅使用端点上设置的选项，而不使用来自消息标头中的值。	false	布尔值

Name	描述	默认值	类型
priority (producer)	大于 1 的值指定发送时的消息优先级（其中 1 是最低优先级，9 是最高）。必须启用 explicitQosEnabled 选项才能使此选项生效。 Enum 值： • 1 • 2 • 3 • 4 • 5 • 6 • 7 • 8 • 9	4	int
replyToConcurrentConsumers (producer)	指定执行请求/通过 JMS 回复时的默认并发消费者数量。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。	1	int
replyToMaxConcurrentConsumers (producer)	指定在通过 JMS 使用请求/回复时的最大并发消费者数。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。		int
replyToOnTimeoutMaxConcurrentConsumers (producer)	指定使用请求/通过 JMS 时超时时继续路由的最大并发消费者数。	1	int
replyToOverride (producer)	在 JMS 消息中提供显式 ReplyTo 目的地，这将覆盖 replyTo 的设置。如果要将消息转发到远程队列，并从 ReplyTo 目的地接收回复消息，这非常有用。		字符串

Name	描述	默认值	类型
replyToType (producer)	允许明确指定在执行 request/reply 时要用于 replyTo 队列的策略类型。可能的值有：Temporary、share 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的队列。如需了解更多详细信息，请参阅 Camel JMS 文档，特别是在集群环境中运行时的影响的备注，以及共享回复队列的性能低于其 alternatives Temporary 和 Exclusive。 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 来覆盖此端点配置的超时值，因此具有每个消息独立的超时值。另请参阅 requestTimeoutCheckerInterval 选项。	20000	long
timeToLive (producer)	发送消息时，指定消息的时间到时间（以毫秒为单位）。	-1	long
allowAdditionalHeaders (producer (advanced))	此选项用于允许其他标头，这些标头可能具有根据 JMS 规范无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 包含字节数组或其他无效类型的值来执行此操作。您可以用逗号指定多个标头名称，并用作通配符匹配的后缀。		字符串
allowNullBody (producer (advanced))	是否允许在没有正文的情况下发送消息。如果此选项为 false，且消息正文为 null，则会抛出 JMSException。	true	布尔值
alwaysCopyMessage (producer (advanced))	如果为 true，则 Camel 始终会在消息传递给发送的制作者时生成 JMS 消息副本。在某些情况下需要复制消息，如设置 replyToDestinationSelectorName 时（如果设置了 replyToDestinationSelectorName，则 Camel 会将 alwaysCopyMessage 选项设置为 true）。	false	布尔值
correlationProperty (producer (advanced))	当使用 InOut 交换模式时，使用此 JMS 属性而不是 JMSCorrelationID JMS 属性来关联消息。如果设置消息仅针对此属性 JMSCorrelationID 属性的值关联，则将忽略且未由 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时间。例如，当您通过 JMS 进行请求/回复时，Camel 默认使用 requestTimeout 值作为发送的消息的时间。问题是，发送者和接收器系统必须同步其时钟，因此它们正在同步。这并非始终易于归档。因此，您可以使用 disableTimeToLive=true 来不设置发送消息上的生存时间。然后，消息不会在接收器系统中过期。如需了解更多详细信息，请参见以下部分关于生存时间。	false	布尔值
forceSendOriginalMessage (producer (advanced))	当使用 mapJmsMessage=false Camel 时，如果在路由中涉及标头(get 或 set)，则会创建一个新的 JMS 消息来发送到新的 JMS 目的地。将此选项设置为 true，以强制 Camel 发送收到的原始 JMS 消息。	false	布尔值
includeSentJMSMessageID (producer (advanced))	仅在使用 InOnly 发送到 JMS 目的地时适用（例如触发和忘记）。启用此选项将增强 Camel Exchange 与 JMS 客户端在消息发送到 JMS 目的地时使用的实际 JMSMessageID。	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))	在执行请求/通过 JMS 时，按名称为回复消费者设置缓存级别。这个选项仅在使用固定回复队列（而非临时）时才适用。Camel 默认将使用：CACHE_CONSUMER 用于专用或共享的 w/ replyToSelectorName。和 CACHE_SESSION 用于没有 replyToSelectorName 的共享。IBM WebSphere 等 JMS 代理可能需要设置 replyToCacheLevelName=CACHE_NONE 才能正常工作。注：如果使用临时队列，则不允许使用更高的值，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，以便您可以在使用共享队列（也就是说，如果您不使用临时回复队列）时过滤来自其他回复的回复。		字符串
streamMessageTypeEnabled (producer (advanced))	设置 StreamMessage 类型是否已启用。流类型的消息有效负载（如 files、InStream 等）将通过作为 BytesMessage 或 StreamMessage 发送。此选项控制将使用哪种类型。默认情况下，使用 BytesMessage 来强制整个消息有效负载读取到内存中。通过启用此选项，消息有效负载以块的形式读取到内存中，然后每个块都会写入 StreamMessage，直到没有更多数据。	false	布尔值
allowSerializedHeaders (advanced)	控制是否包含序列化标头。仅在 transferExchange 为 true 时才适用。这要求对象可以序列化。Camel 将排除任何不可序列化的对象，并将它记录在 WARN 级别。	false	布尔值
artemisStreamingEnabled (advanced)	是否针对 Apache Artemis 流模式进行优化。这可减少使用带有 JMS StreamMessage 类型的 Artemis 时的内存开销。只有在使用 Apache Artemis 时，才必须启用这个选项。	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 无法获得连接到远程 JMS 代理的连接，那么在重试和/或故障切换时可能会阻断它。这将使 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 使用异步模式的专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果没有建立连接，则会在 WARN 级别记录异常，使用者将无法接收消息；然后，您可以重启路由来重试。	false	布尔值
asyncStopListener (advanced)	在停止路由时，是否异步停止 JmsConsumer 消息监听程序。	false	布尔值
destinationResolver (advanced)	一个可插拔的 org.springframework.jms.support.destination.DestinationResolver，供您使用自己的解析器（例如，在 JNDI registry 中查找实际目的地）。		DestinationResolver
errorHandler (advanced)	指定在处理消息时引发任何未发现异常时要调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置错误处理程序，则这些例外将在 WARN 级别记录。您可以配置日志记录级别，以及是否应使用 errorHandlerLoggingLevel 和 errorHandlerLogStack Trace 选项记录堆栈追踪。这样可以更容易配置，而不是需要对自定义错误处理程序进行代码。		ErrorHandler

Name	描述	默认值	类型
exceptionListener (advanced)	指定针对任何底层 JMS 异常通知的 JMS Exception Listener。		ExceptionListener
HeaderFilterStrategy (advanced)	使用自定义 HeaderFilterStrategy 将标头过滤到或从 Camel 消息过滤。		HeaderFilterStrategy
idleConsumerLimit (advanced)	指定任何给定时间允许闲置的用户数量的限值。	1	int
idleTaskExecutionLimit (advanced)	指定接收任务闲置执行的限制，不会在其执行中收到任何消息。如果达到这个限制，任务将关闭并将接收给其他执行任务（在动态调度时；请参阅 maxConcurrentConsumers 设置）。Spring 提供了额外的文档。	1	int
includeAllJMSXProperties (advanced)	在从 JMS 到 Camel Message 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，则不会应用这个选项。	false	布尔值
jmsKeyFormatStrategy (advanced)	编码和解码 JMS 密钥的可插拔策略，以便它们符合 JMS 规范。Camel 提供两个开箱即用的实现：default 和 passthrough。默认策略将安全地使用句点和连字符 (. 和 -)。passthrough 策略将密钥保留原样。可用于不负责 JMS 标头密钥是否包含非法字符的 JMS 代理。您可以自行提供 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 sVirt 表示法引用它。 Enum 值： • default • passthrough		JmsKeyFormatStrategy
mapJmsMessage (advanced)	指定 Camel 是否应该自动将收到的 JMS 消息映射到合适的有效负载类型，如 javax.jms.TextMessage 到 String 等。	true	布尔值
maxMessagesPerTask (advanced)	每个任务的消息数量。-1 代表没有限制。如果您为并发消费者使用范围（例如 min max），则可以使用此选项将值设为 100，以控制消费者在需要较少的工作时可以缩小的速度。	-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)	使用给定的 MessageCreatedStrategy，当 Camel 发送 JMS 消息时，Camel 创建 javax.jms.Message 对象的新实例。		MessageCreatedStrategy
messageIdEnabled (advanced)	发送时，指定是否应添加消息 ID。这只是对 JMS 代理的提示。如果 JMS 提供程序接受此提示，则这些消息必须将消息 ID 设置为 null；如果提供程序忽略提示，则必须将消息 ID 设置为其普通唯一值。	true	布尔值
messageListenerContainerFactory (advanced)	MessageListenerContainerFactory 的 registry ID，用于决定要使用消息的 org.springframework.jms.listener.AbstractMessageListenerContainer。设置此项将自动将 consumerType 设置为 Custom。		MessageListenerContainerFactory
messageTimestampEnabled (advanced)	指定在发送消息时是否默认启用时间戳。这只是对 JMS 代理的提示。如果 JMS 提供程序接受此提示，则这些消息必须将时间戳设置为零；如果提供程序忽略提示，则必须将时间戳设置为其正常值。	true	布尔值
pubSubNoLocal (advanced)	指定是否禁止自己连接发布的消息的发送。	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，即当连接被刷新时，以毫秒为单位。默认值为 5000 ms，即 5 秒。	5000	long
requestTimeoutCheckerInterval (advanced)	配置 Camel 在执行请求/通过 JMS 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，如果发生超时时，您必须更快地响应，那么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
Sync (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)	如果启用了且您使用 Request Reply messaging (InOut)，并且 Exchange 失败在消费者端，则原因例外将作为 javax.jms.ObjectMessage 发回的响应。如果客户端是 Camel，则返回的例外将重新箭头。这样，您可以使用 Camel JMS 作为路由中的桥接 - 例如，使用持久性队列来启用强大的路由。请注意，如果您也启用了 transferExchange，这个选项将具有优先权。caught 异常需要可以被序列化。消费者端的原始例外可以嵌套在外部异常中，如 org.apache.camel.RuntimeCamelException。请谨慎使用它，因为数据正在使用 Java 对象序列化，要求接收者在类级别反序列化数据，这会强制在生产者和消费者之间进行强校准。	false	布尔值
transferExchange (advanced)	您可以在有线线上传输交换，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 header, Exchange properties, exchange exception.这要求对象可以序列化。Camel 将排除任何不可序列化的对象，并将它记录在 WARN 级别。您必须在制作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。请谨慎使用它，因为数据正在使用 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，这会强制在需要使用兼容 Camel 版本的生产者和消费者之间进行强大的协调。	false	布尔值
useMessageIDAsCorrelationID (advanced)	指定 JMSMessageID 是否应该始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值
waitForProvisionCorrelationToBeUpdatedCounter (advanced)	在执行 request/reply over JMS 以及启用了 useMessageIDAsCorrelationID 时，等待 provisional correlation id 被更新到实际关联 ID 的次数。	50	int
waitForProvisionCorrelationToBeUpdatedThreadSleepingTime (advanced)	等待置备关联 ID 时每次休眠的时间间隔（以秒为单位）。	100	long

Name	描述	默认值	类型
errorHandlerLoggingLevel (logging)	允许为日志记录 uncaught 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)	允许通过默认错误处理程序来控制是否应记录 stacktraces。	true	布尔值
密码 （安全）	与 ConnectionFactory 一起使用的密码。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
用户名 （安全）	与 ConnectionFactory 一起使用的用户名。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
Transacted (transaction)	指定是否使用转换模式。	false	布尔值
TransactedInOut (transaction)	指定 InOut 操作（请求回复）是否默认使用 transacted 模式，如果此标志被设置为 true，则 Spring JmsTemplate 将把 sessionTransacted 设置为 true，而 confirmMode 作为转换用于 InOut 操作。请注意：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，容器对这些值做出自己的决策。类似地，这些参数不会考虑本地管理的事务，因为 Spring JMS 在这种情况下在现有 JMS 会话上运行。在受管事务外运行时，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（除 XA 事务之外）时同步的本地 JMS 事务。这与主事务一起管理本地 JMS 事务（可能是原生 JDBC 事务）的影响，在主事务后 JMS 事务提交右侧的 JMS 事务。	false	布尔值
lazyCreateTransactionManager (transaction (advanced))	如果为 true，Camel 将创建一个 JmsTransactionManager，如果没有在选项 transacted=true 时注入 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))	如果使用转换模式，事务的超时值（以秒为单位）。	-1	int

1.5. 使用方法

当 AMQP 组件继承自 JMS 组件时，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");
```

1.6. 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");
```

您还可以将 **org.apache.camel.component.amqp.amqpConnectionDetails 的实例** 添加到 registry 中，以便自动配置 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

CDI 的 AMQP 连接详情自动配置

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

您还可以依赖 来读取 AMQP 连接详情。工厂方法 **AMQPConnectionDetails.discoverAMQP （）** 会尝试以类似 Kubernetes 的约定读取 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**，您可以将选项附加到 URI 中，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。

1.7. 使用主题

要使用使用 **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在ComponentamqpComponent （）** 方法和 **AMQPConnectionDetails** 都预先配置带有主题前缀的组件，因此您不必显式配置它。

1.8. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 amqp 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-amqp-starter</artifactId>
</dependency>
```

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，则可能会考虑启用此选项，同时仍然在队列中排队消息。如果此选项为 false，并且您停止 JMS 路由，则消息可能会被拒绝，并且 JMS 代理必须尝试 redeliveries （但可能会再次拒绝），最终消息可能会移到 JMS 代理上的死信队列中。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，其为： 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，这些标头可能具有根据 JMS 规范无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 包含字节数组或其他无效类型的值来执行此操作。您可以用逗号指定多个标头名称，并用作通配符匹配的后缀。		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 自动发现 ConnectionFactory。如果只找到一个 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 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	是否允许在没有正文的情况下发送消息。如果此选项为 false，且消息正文为 null，则会抛出 JMSException。	true	布尔值

Name	描述	默认值	类型
camel.component.amqp.allow-reply-manager-quick-stop	是否启用请求管理器中使用的 DefaultMessageListenerContainer，允许 DefaultMessageListenerContainer.runningAllowed 标志在 JmsConfigurationVirtualMachineisAcceptMessages WhileStopping 时快速停止，并且 org.apache.camel.CamelContext 当前已停止。在常规 JMS 用户中默认启用这种快速停止功能，但要为回复管理器启用这个标志。	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 时才适用。这要求对象可以序列化。Camel 将排除任何不可序列化的对象，并将它记录在 WARN 级别。	false	布尔值
camel.component.amqp.always-copy-message	如果为 true，则 Camel 始终会在消息传递给发送的制作者时生成 JMS 消息副本。在某些情况下需要复制消息，如设置 replyToDestinationSelectorName 时（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消息处于活跃状态时收到消息。通常，连接到队列的活动消费者以轮循方式从它接收消息。当使用消费者优先级时，如果有多个具有相同高优先级的活跃用户，则消息将进行循环发送。只有高优先级消费者没有可用的信用消息或高优先级消费者接受消息时，消息才会降低优先级较低的消费者（例如，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。这可减少使用带有 JMS StreamMessage 类型的 Artemis 时的内存开销。只有在使用 Apache Artemis 时，才必须启用这个选项。	false	布尔值
camel.component.amqp.async-consumer	JmsConsumer 是否异步处理交换。如果启用，JmsConsumer 可以从 JMS 队列中提取下一个消息，而上一个消息则异步处理（通过 Asynchronous Routing Engine）。这意味着消息可能没有完全严格按照顺序进行处理。如果禁用（作为默认），则在 JmsConsumer 会从 JMS 队列中提取下一个消息前，会完全处理交换。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 必须支持 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 无法获得连接到远程 JMS 代理的连接，那么在重试和/或故障切换时可能会阻断它。这将使 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 使用异步模式的专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果没有建立连接，则会在 WARN 级别记录异常，使用者将无法接收消息；然后，您可以重启路由来重试。	false	布尔值
camel.component.amqp.async-stop-listener	在停止路由时，是否异步停止 JmsConsumer 消息监听程序。	false	布尔值
camel.component.amqp.auto-startup	指定消费者容器是否应自动启动。	true	布尔值
camel.component.amqp.autowired-enabled	是否启用自动关闭。这用于自动关闭选项（选项必须标记为 autowired），方法是在 registry 中查找查找是否有单个匹配类型实例，然后在组件上配置。这可以用于自动配置 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	根据底层 JMS 资源的名称设置缓存级别。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。它通常只适用于持久主题订阅。如果使用 Apache ActiveMQ，您可能更喜欢使用 Virtual Topics。		字符串
camel.component.amqp.concurrent-consumers	指定从 JMS 消耗时的默认并发消费者数量（而不是通过 JMS 请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。当通过 JMS 执行 request/reply 时，选项 replyToConcurrentConsumers 用于控制回复消息监听器上的并发消费者数量。	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。必须在组件或端点上配置连接工厂。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。消费者类型决定要使用的 Spring JMS 侦听器。默认将使用 org.springframework.jms.listener.DefaultMessageListenerContainer，Simple 将使用 org.springframework.jms.listener.SimpleMessageListenerContainer。指定 Custom 时，由 messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 将决定要使用的 org.springframework.jms.listener.AbstractMessageListenerContainer。		ConsumerType
camel.component.amqp.correlation-property	当使用 InOut 交换模式时，使用此 JMS 属性而不是 JMSCorrelationID JMS 属性来关联消息。如果设置消息仅针对此属性 JMSCorrelationID 属性的值关联，则将忽略且未由 Camel 设置。		字符串
camel.component.amqp.default-task-executor-type	指定在 DefaultMessageListenerContainer 中使用哪些默认 TaskExecutor 类型，用于消费者端点和制作者端点的 ReplyTo consumer。可能的值： SimpleAsync（使用 Spring 的 SimpleAsyncTaskExecutor）或 ThreadPool （使用 Spring 的 ThreadPoolTaskExecutor 带有最佳值 - 缓存线程池）。如果没有设置，则默认为之前的行为，它将缓存线程池用于消费者端点，而 SimpleAsync 用于回复用户。建议使用 ThreadPool 来减少弹性配置中线程垃圾箱，同时动态增加和减少并发用户。		DefaultTaskExecutorType
camel.component.amqp.delivery-delay	设置用于发送 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。可能的值是由 javax.jms.DeliveryMode 定义的值。NON_PERSISTENT = 1 和 PERSISTENT = 2。		整数
camel.component.amqp.delivery-persistent	指定默认使用持久性交付。	true	布尔值
camel.component.amqp.destination-resolver	一个可插拔的 org.springframework.jms.support.destination.DestinationResolver，供您使用自己的解析器（例如，在 JNDI registry 中查找实际目的地）。选项是一个 org.springframework.jms.support.destination.DestinationResolver 类型。		DestinationResolver

Name	描述	默认值	类型
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 从路由中消耗，并且您不希望 Camel 自动发送回复消息，则可以使用此选项，因为代码中的另一个组件处理回复消息。如果要在不同的消息代理之间将 Camel 用作代理，并且希望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时间。例如，当您通过 JMS 进行请求/回复时，Camel 默认使用 requestTimeout 值作为发送的消息的时间。问题是，发送者和接收器系统必须同步其时钟，因此它们正在同步。这并非始终易于归档。因此，您可以使用 disableTimeToLive=true 来不设置发送消息上的生存时间。然后，消息不会在接收器系统中过期。如需了解更多详细信息，请参见以下部分关于生存时间。	false	布尔值
camel.component.amqp.durable-subscription-name	用于指定持久主题订阅的可配置订阅者名称。还必须配置 clientId 选项。		字符串
camel.component.amqp.eager-loading-of-properties	加载消息时立即启用 JMS 属性和有效负载的 eager 加载，因为 JMS 属性可能并不是必需的，但有时可能会捕获与底层 JMS 提供程序和使用 JMS 属性的早期问题。另请参阅选项 eagerPoisonBody。	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)是 poison （无法读取/映射），然后将这个文本设置为消息正文，因此可以处理消息(poison 的原因)已作为交换异常保存。这可以通过设置 eagerPoisonBody=false 来关闭。另请参阅 eagerLoadingOfProperties 选项。	Poison JMS 消息，因为 $\{exception.message}	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
camel.component.amqp.error-handler	指定在处理消息时引发任何未发现异常时要调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置错误处理程序，则这些例外将在 WARN 级别记录。您可以配置日志记录级别，以及是否应使用 errorHandlerLoggingLevel 和 errorHandlerLogStack Trace 选项记录堆栈追踪。这样可以更容易配置，而不是需要对自定义错误处理程序进行代码。选项是一个 org.springframework.util.ErrorHandler 类型。		ErrorHandler

Name	描述	默认值	类型
camel.component.amqp.error-handler-log-stack-trace	允许通过默认错误处理程序来控制是否应记录 stacktraces。	true	布尔值
camel.component.amqp.error-handler-logging-level	允许为日志记录 uncaught 异常配置默认 errorHandler 日志记录级别。		LoggingLevel
camel.component.amqp.exception-listener	指定针对任何底层 JMS 异常通知的 JMS Exception Listener。选项是 javax.jms.ExceptionListener 类型。		ExceptionListener
camel.component.amqp.explicit-qos-enabled	设置在发送消息时使用 deliveryMode、priority 或 timeToLive 数量服务。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用到当前的端点。这与 preserveMessageQos 选项（按消息粒度运行）相反，从 Camel In 消息标头中读取 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监听程序会话。	false	布尔值
camel.component.amqp.force-send-original-message	当使用 mapJmsMessage=false Camel 时，如果在路由中涉及标头(get 或 set)，则会创建一个新的 JMS 消息来发送到新的 JMS 目的地。将此选项设置为 true，以强制 Camel 发送收到的原始 JMS 消息。	false	布尔值
camel.component.amqp.format-date-headers-to-iso8601	根据 ISO 8601 标准设置 JMS 日期属性是否应格式化。	false	布尔值
camel.component.amqp.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 将标头过滤到或从 Camel 消息过滤。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		HeaderFilterStrategy
camel.component.amqp.idle-consumer-limit	指定任何给定时间允许闲置的用户数量的限值。	1	整数

Name	描述	默认值	类型
camel.component.amqp.idle-task-execution-limit	指定接收任务闲置执行的限制，不会在其执行中收到任何消息。如果达到这个限制，任务将关闭并将接收给其他执行任务（在动态调度时；请参阅 maxConcurrentConsumers 设置）。Spring 提供了额外的文档。	1	整数
camel.component.amqp.include-all-jmsx-properties	在从 JMS 到 Camel Message 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，则不会应用这个选项。	false	布尔值
camel.component.amqp.include-amqp-annotations	从 AMQP 到 Camel 消息映射时是否包含 AMQP 注解。把它设置为 true 会映射包含 JMS_AMQP_MA_ 前缀的 AMQP 消息注解到消息标头。由于 Apacheqpid JMS API 中的限制，当前交付注解将被忽略。	false	布尔值
camel.component.amqp.include-sent-jms-message-id	仅在使用 InOnly 发送到 JMS 目的地时适用（例如触发和忘记）。启用此选项将增强 Camel Exchange 与 JMS 客户端在消息发送到 JMS 目的地时使用的实际 JMSMessageID。	false	布尔值
camel.component.amqp.jms-key-format-strategy	编码和解码 JMS 密钥的可插拔策略，以便它们符合 JMS 规范。Camel 提供两个开箱即用的实现：default 和 passthrough。默认策略将安全地使用句点和连字符 (. 和 -)。passthrough 策略将密钥保留原样。可用于不负责 JMS 标头密钥是否包含非法字符的 JMS 代理。您可以自行提供 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 sVirt 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 会决定要从 In body 类型使用哪个 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，Camel 将创建一个 JmsTransactionManager，如果没有在选项 transacted=true 时注入 transactionManager。	true	布尔值
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

Name	描述	默认值	类型
camel.component.amqp.map-jms-message	指定 Camel 是否应该自动将收到的 JMS 消息映射到合适的有效负载类型，如 javax.jms.TextMessage 到 String 等。	true	布尔值
camel.component.amqp.max-concurrent-consumers	指定从 JMS 消耗时的最大并发消费者数（而不是通过 JMS 请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。当通过 JMS 执行请求/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。如果您为并发消费者使用范围（例如 min max），则可以使用此选项将值设为 100，以控制消费者在需要较少的工作时可以缩小的速度。	-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。选项是一个 org.springframework.jms.support.converter.MessageConverter 类型。		MessageConverter
camel.component.amqp.message-created-strategy	使用给定的 MessageCreatedStrategy，当 Camel 发送 JMS 消息时，Camel 创建 javax.jms.Message 对象的新实例。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。这只是对 JMS 代理的提示。如果 JMS 提供程序接受此提示，则这些消息必须将消息 ID 设置为 null；如果提供程序忽略提示，则必须将消息 ID 设置为其普通唯一值。	true	布尔值
camel.component.amqp.message-listener-container-factory	MessageListenerContainerFactory 的 registry ID，用于决定要使用消息的 org.springframework.jms.listener.AbstractMessageListenerContainer。设置此项将自动将 consumerType 设置为 Custom。选项是 org.apache.camel.component.jms.MessageListenerContainerFactory 类型。		MessageListenerContainerFactory
camel.component.amqp.message-timestamp-enabled	指定在发送消息时是否默认启用时间戳。这只是对 JMS 代理的提示。如果 JMS 提供程序接受此提示，则这些消息必须将时间戳设置为零；如果提供程序忽略提示，则必须将时间戳设置为其正常值。	true	布尔值

Name	描述	默认值	类型
camel.component.amqp.password	与 ConnectionFactory 一起使用的密码。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
camel.component.amqp.preserve-message-qos	如果要使用消息中指定的 QoS 设置来发送消息，而不是 JMS 端点上的 QoS 设置。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供全部或仅提供其中一些。如果没有提供，Camel 将回退到使用端点中的值。因此，在使用此选项时，标头会覆盖端点中的值。与之相反，clearQosEnabled 选项将仅使用端点上设置的选项，而不使用来自消息标头中的值。	false	布尔值
camel.component.amqp.priority	大于 1 的值指定发送时的消息优先级（其中 1 是最低优先级，9 是最高）。必须启用 explicitQosEnabled 选项才能使此选项生效。	4	整数
camel.component.amqp.pub-sub-no-local	指定是否禁止自己连接发布的消息的发送。	false	布尔值
camel.component.amqp.queue-browse-strategy	在浏览队列时使用自定义 QueueBrowseStrategy。选项是一个 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，即当连接被刷新时，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
camel.component.amqp.reply-to	提供显式 ReplyTo 目的地（覆盖消费者中 Message.getJMSReplyTo （）的所有传入值）。		字符串
camel.component.amqp.reply-to-cache-level-name	在执行请求/通过 JMS 时，按名称为回复消费者设置缓存级别。这个选项仅在使用固定回复队列（而非临时）时才适用。Camel 默认将使用：CACHE_CONSUMER 用于专用或共享的 w/ replyToSelectorName。和 CACHE_SESSION 用于没有 replyToSelectorName 的共享。IBM WebSphere 等 JMS 代理可能需要设置 replyToCacheLevelName=CACHE_NONE 才能正常工作。注：如果使用临时队列，则不允许使用更高的值，如 CACHE_CONSUMER 或 CACHE_SESSION。		字符串

Name	描述	默认值	类型
camel.component.amqp.reply-to-concurrent-consumers	指定执行请求/通过 JMS 回复时的默认并发消费者数量。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。	1	整数
camel.component.amqp.reply-to-consumer-type	回复消费者的消费者类型（当执行请求/回复时），可以是 Simple、Default 或 Custom 之一。消费者类型决定要使用的 Spring JMS 侦听器。默认将使用 org.springframework.jms.listener.DefaultMessageListenerContainer，Simple 将使用 org.springframework.jms.listener.SimpleMessageListenerContainer。指定 Custom 时，由 messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 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交付进行回复。	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，以便您可以在使用共享队列（也就是说，如果您不使用临时回复队列）时过滤来自其他回复的回复。		字符串
camel.component.amqp.reply-to-max-concurrent-consumers	指定在通过 JMS 使用请求/回复时的最大并发消费者数。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。		整数
camel.component.amqp.reply-to-on-timeout-max-concurrent-consumers	指定使用请求/通过 JMS 时超时时继续路由的最大并发消费者数。	1	整数
camel.component.amqp.reply-to-override	在 JMS 消息中提供显式 ReplyTo 目的地，这将覆盖 replyTo 的设置。如果要将消息转发到远程队列，并从 ReplyTo 目的地接收回复消息，这非常有用。		字符串
camel.component.amqp.reply-to-same-destination-allowed	JMS 使用者是否允许向消费者使用的同一目的地发送回复消息。这可防止出现无限循环，并通过消耗并向自己发送相同的消息。	false	布尔值

Name	描述	默认值	类型
camel.component.amqp.reply-to-type	允许明确指定在执行 request/reply 时要用于 replyTo 队列的策略类型。可能的值有：Temporary、share 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的队列。如需了解更多详细信息，请参阅 Camel JMS 文档，特别是在集群环境中运行时的影响的备注，以及共享回复队列的性能低于其 alternatives Temporary 和 Exclusive。		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 来覆盖此端点配置的超时值，因此具有每个消息独立的超时值。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 在执行请求/通过 JMS 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，如果发生超时时，您必须更快地响应，那么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
camel.component.amqp.stream-message-type-enabled	设置 StreamMessage 类型是否已启用。流类型的消息有效负载（如 files、InStream 等）将通过作为 BytesMessage 或 StreamMessage 发送。此选项控制将使用哪种类型。默认情况下，使用 BytesMessage 来强制整个消息有效负载读取到内存中。通过启用此选项，消息有效负载以块的形式读取到内存中，然后每个块都会写入 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。可使用的 durable 订阅名称通过 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 域)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	设置要创建的订阅名称。在带有共享或可升级订阅的主题（公共域）中应用。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： 每个订阅都只允许 1 个并发消费者（这是此消息侦听器容器的默认值），但一个共享订阅（需要 JMS 2.0）除外。		字符串

Name	描述	默认值	类型
camel.component.amqp.subscription-shared	设置是否共享订阅。可以使用的共享订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。请注意，共享的订阅也可能是危险的，因此此标志也可以与订阅相结合。仅在侦听主题(pub-sub 域)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执行器以使用消息。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 用户与 JMS 代理的有效连接。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 producer 也经过测试。	false	布尔值
camel.component.amqp.time-to-live	发送消息时，指定消息的时间到时间（以毫秒为单位）。	-1	Long
camel.component.amqp.transacted	指定是否使用转换模式。	false	布尔值
camel.component.amqp.transacted-in-out	指定 InOut 操作（请求回复）是否默认使用 transacted 模式，如果此标志被设置为 true，则 Spring JmsTemplate 将把 sessionTransacted 设置为 true，而 confirmMode 作为转换用于 InOut 操作。请注意：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，容器对这些值做出自己的决策。类似地，这些参数不会考虑本地管理的事务，因为 Spring JMS 在这种情况下在现有 JMS 会话上运行。在受管事务外运行时，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（除 XA 事务之外）时同步的本地 JMS 事务。这与主事务一起管理本地 JMS 事务（可能是原生 JDBC 事务）的影响，在主事务后 JMS 事务提交右侧的 JMS 事务。	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

Name	描述	默认值	类型
camel.component.amqp.transaction-name	要使用的事务的名称。		字符串
camel.component.amqp.transaction-timeout	如果使用转换模式，事务的超时值（以秒为单位）。	-1	整数
camel.component.amqp.transfer-exception	如果启用了且您使用 Request Reply messaging (InOut)，并且 Exchange 失败在消费者端，则原因例外将作为 javax.jms.ObjectMessage 发回的响应。如果客户端是 Camel，则返回的例外将重新箭头。这样，您可以使用 Camel JMS 作为路由中的桥接 - 例如，使用持久性队列来启用强大的路由。请注意，如果您也启用了 transferExchange，这个选项将具有优先权。caught 异常需要可以被序列化。消费者端的原始例外可以嵌套在外部异常中，如 org.apache.camel.RuntimeCamelException。请谨慎使用它，因为数据正在使用 Java 对象序列化，要求接收者在类级别反序列化数据，这会强制在生产者和消费者之间进行强校准。	false	布尔值
camel.component.amqp.transfer-exchange	您可以在有线线上传输交换，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 header, Exchange properties, exchange exception.这要求对象可以序列化。Camel 将排除任何不可序列化的对象，并将它记录在 WARN 级别。您必须在制作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。请谨慎使用它，因为数据正在使用 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，这会强制在需要使用兼容 Camel 版本的生产者和消费者之间进行强大的协调。	false	布尔值
camel.component.amqp.use-message-id-as-correlation-id	指定 JMSMessageID 是否应该始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值
camel.component.amqp.username	与 ConnectionFactory 一起使用的用户名。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
camel.component.amqp.wait-for-provision-correlation-to-be-updated-counter	在执行 request/reply over JMS 以及启用了 useMessageIDAsCorrelationID 时，等待 provisional correlation id 被更新到实际关联 ID 的次数。	50	整数

Name	描述	默认值	类型
camel.component.amqp.wait-for-provision-correlation-to-be-updated-thread-sleeping-time	等待置备关联 ID 时每次休眠的时间间隔（以秒为单位）。选项是一个长类型。	100	Long

第 2 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。

2.1. 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&...**

2.2. 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

2.2.1. 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。

2.2.2. 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。

2.3. 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
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

Name	描述	默认值	类型
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

2.4. 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：

2.4.1. 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

2.4.2. 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

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 (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
value (producer)	指标值。		å�Œ

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 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。

2.5. 使用方法

2.5.1. 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

2.5.2. 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。
CamelAwsCwMetricTimestamp	**Date**	Amazon CW 指标时间戳。

标头	类型	描述
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.

2.5.3. 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**。

2.6. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-aws2-cw</artifactId>
    <version>${camel-version}</version>
</dependency>
```

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

2.7. 例子

2.7.1. 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");
```

2.8. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 aws2-cw 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-aws2-cw-starter</artifactId>
</dependency>
```

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
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

Name	描述	默认值	类型
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
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 3 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。

3.1. 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&...**

3.2. 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

3.2.1. 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。

3.2.2. 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。

3.3. 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

3.4. 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：

3.4.1. 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

3.4.2. 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。

3.5. 使用方法

3.5.1. 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

3.5.2. 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。

3.5.3. 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。

3.5.4. 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。

3.5.5. 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

3.5.6. 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。

3.5.7. 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。

3.5.8. 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。

3.5.9. 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。

3.5.10. 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。

3.5.11. 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。

3.5.12. 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**。

3.6. 支持的制作者操作

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

- UpdateTable

3.7. 例子

3.7.1. 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 **3.18.3** 必须替换为 Camel 的实际版本。

3.8. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 aws2-ddb 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-aws2-ddb-starter</artifactId>
</dependency>
```

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

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
camel.component.aws2-ddbstream.proxy-host	在实例化 DDBStreams 客户端时定义代理主机。		字符串
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 4 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。

4.1. 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&...**

4.2. 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

4.2.1. 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。

4.2.2. 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。

4.3. 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
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，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
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
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

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
accessKey (security)	Amazon AWS 访问密钥.		字符串
secretKey (security)	Amazon AWS Secret 密钥。		字符串

4.4. 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：

4.4.1. 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

4.4.2. 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

Name	描述	默认值	类型
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
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，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
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
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

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
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
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

Name	描述	默认值	类型
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
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	MILLISECONDS	TimeUnit

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

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。

4.5. BATCH CONSUMER

这个组件实现了 Batch Consumer。

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

4.6. 使用方法

4.6.1. 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

4.6.2. 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

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

4.6.3. 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");
```

4.6.4. 提供 AWS 凭证

建议您使用 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。

4.6.5. 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。

4.6.6. 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

4.7. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-aws2-kinesis</artifactId>
  <version>${camel-version}</version>
</dependency>
```

其中 **3.18.3** 必须替换为 Camel 的实际版本。

4.8. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 aws2-kinesis 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-aws2-kinesis-starter</artifactId>
</dependency>
```

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
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

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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 5 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。

5.1. 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&...**

5.2. 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

5.2.1. 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。

5.2.2. 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。

5.3. 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
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

Name	描述	默认值	类型
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
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

5.4. 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：

5.4.1. 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

5.4.2. 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。

5.5. 使用方法

5.5.1. 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

5.5.2. 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

5.6. 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

5.7. 例子

5.7.1. 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。

5.7.2. 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);
    }
});
```

5.8. 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。

5.9. 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 **3.18.3** 必须替换为 Camel 的实际版本。

5.10. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 aws2-lambda 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-aws2-lambda-starter</artifactId>
</dependency>
```

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

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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
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 6 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]。

6.1. 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&...

6.2. 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

6.2.1. 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。

6.2.2. 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。

6.3. 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
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

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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
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

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
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
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

6.4. 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：

6.4.1. 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

6.4.2. 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
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

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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。

6.5. BATCH CONSUMER

这个组件实现了 Batch Consumer。

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

6.6. 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");
```

6.6.1. 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。

6.6.2. 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。

6.6.3. 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。

6.6.4. 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。

6.6.5. 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");
```

6.6.6. 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。

6.6.7. 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

6.6.8. 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

6.7. 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。

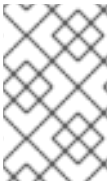
通过将 restartPolicy 设置为 lastPart，您将重启从制作者左侧最后一个部分编号上传文件和内容。

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。

6.8. 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。

6.9. 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。

6.10. 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。

6.11. 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);
```

6.12. 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。

6.13. 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。

6.14. 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 **3.18.3** 必须替换为 Camel 的实际版本。

6.15. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 aws2-s3 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-aws2-s3-starter</artifactId>
</dependency>
```

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

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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
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 7 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。

7.1. 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&...**

7.2. URI 选项

7.2.1. 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

7.2.1.1. 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。

7.2.1.2. 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。

7.3. 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
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
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

7.4. 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：

7.4.1. 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

7.4.2. 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

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
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
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

Name	描述	默认值	类型
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
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。

7.5. 使用方法

7.5.1. 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。

7.5.2. 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。

7.5.3. SNS producer 设置的消息标头

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

7.5.4. 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**。

7.5.5. 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(...);
```

7.6. 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。

7.7. 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。

7.7.1. 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。

7.8. 例子

7.8.1. 生成者示例

发送到主题

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

7.9. 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 **3.18.3** 必须替换为 Camel 的实际版本。

7.10. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 aws2-sns 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-aws2-sns-starter</artifactId>
</dependency>
```

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

Name	描述	默认值	类型
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 8 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。

8.1. 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&...

8.2. 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

8.2.1. 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。

8.2.2. 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。

8.3. 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
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

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)	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
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

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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
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 来确定要等待的时长。		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

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

8.4. 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：

8.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
queueNameOrArn (common)	**所需的** Queue name 或 ARN。		字符串

8.4.2. 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
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

Name	描述	默认值	类型
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
defaultVisibilityTimeout (consumer)	默认可见性超时（以秒为单位）。		整数
deleteAfterRead (consumer)	读取后，从 SQS 删除消息。	true	布尔值

Name	描述	默认值	类型
deleteIfFiltered (consumer)	如果交换具有键 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。详情请参阅 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。默认无限，但使用 0 或负数来禁用它（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
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

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
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 来确定要等待的时长。		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。

8.5. BATCH CONSUMER

这个组件实现了 Batch Consumer。

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

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 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

8.6.2. 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 checksum。
CamelAwsSqsMessageId	**字符串**	Amazon SQS 消息 ID。
CamelAwsSqsDelaySeconds	**整数**	Amazon SQS 消息可以被其他人查看的延迟秒数。

8.6.3. 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 checksum。
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。

8.6.4. 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** 实例配置拥有更多控制，您可以创建自己的实例：

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

8.6.5. 创建或更新 SQS 队列

在 SQS 组件中，当端点启动时，将执行检查以获取有关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（将 **autoCreateQueue** 选项设置为 true），它将使用 SQS 配置中的默认参数创建。如果它已在 AWS 上启动，则 SQS 配置选项将用于覆盖现有的 AWS 配置。

8.6.6. DelayQueue VS Delay 用于 Single 信息

当选项 **delayQueue** 设为 true 时，SQS Queue 将是一个 **DelayQueue**，带有 **DelaySeconds** 选项作为延迟。有关 **DelayQueue** 的更多信息，请参阅 AWS SQS 文档。要考虑的重要信息如下：

- 对于标准队列，每个队列延迟设置不会重新更改设置不会影响队列中已经的信息的延迟。
- 对于 FIFO 队列，每个队列延迟设置是重新引入的，该设置会影响队列中已存在于的消息的延迟。

如官方文档中所述。如果要在单个消息上指定延迟，可以忽略 **delayQueue** 选项，同时如果您需要向所有消息添加固定延迟，则可以将这个选项设置为 true。

8.6.7. 服务器同级加密

为队列有一组 Server Side Encryption 属性。相关的选项包括 **serverSideEncryptionEnabled**,**keyMasterKeyId** 和 **kmsDataKeyReusePeriod**。SSE 默认禁用。您需要明确将选项设置为 true，并将相关参数设置为 queue 属性。

8.7. JMS 风格的 SELECTORS

SQS 不允许选择器，但您可以使用 Camel Filter EIP 并设置适当的 **visibilityTimeout**。当 SQS 分配消息时，它将等待可见超时，然后尝试将消息发送到不同的消费者，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。要实现适当的过滤，在成功完成路由时也不会发送 **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 进行它，也不会从 SQS 队列中删除。5000 毫秒后，消息对其他消费者可见。

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

8.8. 可用的 PRODUCER 操作

- 单条消息（默认）
- sendBatchMessage
- deleteMessage
- listQueues

8.9. 发送消息

您可以设置 **SendMessageBatchRequest** 或 **Iterable**

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

8.10. 发送批处理消息

您可以设置 **SendMessageBatchRequest** 或 **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，您可以评估来检查哪些消息成功，以及什么信息不成功。批处理的每个消息上设置的 id 将是一个随机 UUID。

8.11. 删除单个消息

使用 **deleteMessage** 操作删除单个消息。您需要为您要删除的消息设置接收句柄标头。

```
from("direct:start")
  .setHeader(SqsConstants.SQS_OPERATION, constant("deleteMessage"))
  .setHeader(SqsConstants.RECEIPT_HANDLE, constant("123456"))
```

```
    .to("aws2-sqs://camel-1?accessKey=RAW(xxx)&secretKey=RAW(xxx)&region=eu-west-1");
```

因此，您将获得一个包含 **DeleteMessageResponse** 实例的交换，您可以使用它来检查消息是否已被删除。

8.12. LIST QUEUES

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，您将获得一个包含 **ListQueuesResponse** 实例的交换，您可以检查实际队列。

8.13. 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，您将获得一个包含 **PurgeQueueResponse** 实例的交换。

8.14. 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 对 AWS 中不存在队列的操作，则不会成功，并返回错误。

8.15. 发送批处理消息和消息重复数据删除策略

如果您使用 **SendBatchMessage** Operation，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，另一个将使用 **NullMessageDeduplicationIdStrategy**，它将使用 body 作为 deduplication 元素。

如果是发送批处理消息操作，您需要使用 **useContentBasedDeduplication** 和 Queue，您需要 **启用基于内容的 deduplication** 选项。

8.16. 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 **3.18.3** 必须替换为 Camel 的实际版本。

8.17. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 aws2-sqs 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-aws2-sqs-starter</artifactId>
</dependency>
```

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	将 AmazonSQS 用作客户端。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

Name	描述	默认值	类型
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
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。详情请参阅 Amazon 文档。	false	布尔值

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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
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 来确定要等待的时长。		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

Name	描述	默认值	类型
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
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 9 章 AZURE SERVICEBUS

从 Camel 3.12 开始

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。服务总线提供了一个可靠安全平台，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

您必须有一个有效的 Windows Azure Storage 帐户。如需更多信息，请参阅 Azure 文档门户。

将以下依赖项添加到此组件的 **pom.xml** 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-azure-servicebus</artifactId>
  <version>3.20.1.redhat-00050</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

9.1. 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

9.1.1. 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。

9.1.2. 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。

9.2. 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，则使用默认的重试选项。		AmqpRetryOptions
amqpTransportType (common)	设置发生与 Azure Service Bus 的所有通信的传输类型。默认值为 AmqpTransportType114AMQP。 Enum 值： • Amqp • AmqpWebSockets	AMQP	AmqpTransportType
clientOptions (common)	设置要从这个构建器构建的客户端发送的 ClientOptions，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。配置代理后，必须将 AmqpTransportTypetransAMQP_WEB_SOCKETS 用于传输类型。		ProxyOptions
serviceBusType (common)	**需要执行** 服务总线类型的连接类型。队列用于典型的队列选项，以及基于订阅模型的主题。 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)	禁用收到消息的自动完成和自动启用。默认情况下，成功处理的消息为 \\{link ServiceBusReceiverAsyncClientöcomplete (ServiceBusReceivedMessage) completed}。如果在处理消息时发生错误，它将是 \\{link ServiceBusReceiverAsyncClientöabandon (ServiceBusReceivedMessage) abandoned}。	false	布尔值
maxAutoLockRenewDuration (consumer)	设置继续自动更新锁定的时间。设置 Duration114ZERO 或 null 可禁用自动续订。对于 \\{link ServiceBusReceiveModeCEIVE_AND_DELETE RECEIVE_AND_DELETE} 模式，禁用自动续订。	5m	Duration
peekNumMaxMessages (consumer)	设置在 peek 操作期间要显示的最大消息数。		整数
prefetchCount (consumer)	设置接收方的 prefetch 计数。对于 \\{link ServiceBusReceiveModeHQPEEK_LOCK PEEK_LOCK} 和 \\{link ServiceBusReceiveModeDSRECEIVE_AND_DELETE RECEIVE_AND_DELETE} 模式，默认值为 1。Prefetch 通过模拟在应用使用 ServiceBusReceiverAsyncClient receiveMessages（）请求一个消息之前，为本地检索提供消息流速度。设置非零值将预先获取该消息的数量。将值设为零关闭。		int
receiverAsyncClient (consumer)	**Autowired** 设置 receiverAsyncClient，以便消耗消费者的消息。		ServiceBusReceiverAsyncClient
serviceBusReceiveMode (consumer)	为接收方设置接收模式。 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。还必须设置 topicOrQueueName 和 serviceBusType=topic。如果 serviceBusType=topic 和使用使用者，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域名。		字符串
tokenCredential (security)	用于 Azure AD 身份验证的 TokenCredential，在 com.azure.identity 中实现。		TokenCredential

9.3. 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：

9.3.1. 路径参数(1 参数)

Name	描述	默认值	类型
topicOrQueueName (common)	所选主题名称或队列名称，具体取决于 serviceBusType 配置。例如，如果 serviceBusType=queue，则这是队列名称，如果 serviceBusType=topic，则这是主题名称。		字符串

9.3.2. 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，则使用默认的重试选项。		AmqpRetryOptions
amqpTransportType (common)	设置发生与 Azure Service Bus 的所有通信的传输类型。默认值为 AmqpTransportType114AMQP。 Enum 值： • Amqp • AmqpWebSockets	AMQP	AmqpTransportType
clientOptions (common)	设置要从这个构建器构建的客户端发送的 ClientOptions，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。配置代理后，必须将 AmqpTransportTypetransAMQP_WEB_SOCKETS 用于传输类型。		ProxyOptions
serviceBusType (common)	**需要执行** 服务总线类型的连接类型。队列用于典型的队列选项，以及基于订阅模型的主题。 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)	禁用收到消息的自动完成和自动启用。默认情况下，成功处理的消息为 \\{link ServiceBusReceiverAsyncClientöcomplete (ServiceBusReceivedMessage) completed}。如果在处理消息时发生错误，它将是 \\{link ServiceBusReceiverAsyncClientöabandon (ServiceBusReceivedMessage) abandoned}。	false	布尔值
maxAutoLockRenewDuration (consumer)	设置继续自动更新锁定的时间。设置 Duration114ZERO 或 null 可禁用自动续订。对于 \\{link ServiceBusReceiveModeCEIVE_AND_DELETE RECEIVE_AND_DELETE} 模式，禁用自动续订。	5m	Duration
peekNumMaxMessages (consumer)	设置在 peek 操作期间要显示的最大消息数。		整数
prefetchCount (consumer)	设置接收方的 prefetch 计数。对于 \\{link ServiceBusReceiveModeHQPEEK_LOCK PEEK_LOCK} 和 \\{link ServiceBusReceiveModeDSRECEIVE_AND_DELETE RECEIVE_AND_DELETE} 模式，默认值为 1。Prefetch 通过模拟在应用使用 ServiceBusReceiverAsyncClient receiveMessages（）请求一个消息之前，为本地检索提供消息流速度。设置非零值将预先获取该消息的数量。将值设为零关闭。		int

Name	描述	默认值	类型
receiverAsyncClient (consumer)	**Autowired** 设置 receiverAsyncClient，以便消耗消费者的消息。		ServiceBusReceiverAsyncClient
serviceBusReceiveMode (consumer)	为接收方设置接收模式。 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。还必须设置 topicOrQueueName 和 serviceBusType=topic。如果 serviceBusType=topic 和使用使用者，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域名。		字符串
tokenCredential (security)	用于 Azure AD 身份验证的 TokenCredential，在 com.azure.identity 中实现。		TokenCredential

9.4. ASYNC CONSUMER 和 PRODUCER

此组件实现了 async Consumer 和 producer。这允许 camel 路由异步使用和生成事件，而不阻断任何线程。

9.5. 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) 常数：APPLICATION_PROPERTIES	生成者和消费者发送和接收的消息应用属性（也称为自定义属性）。		Map
CamelAzureServiceBusContentType (consumer) 常量：CONTENT_TYPE	获取消息的内容类型。		字符串
CamelAzureServiceBusCorrelationId (consumer) 常量：CORRELATION_ID	获取关联标识符。		字符串
CamelAzureServiceBusDeadLetterErrorDescription (consumer) 常量：DEAD_LETTER_ERROR_DESCRIPTION	获取已死字母消息的描述。		字符串
CamelAzureServiceBusDeadLetterReason (consumer) 常量：DEAD_LETTER_REASON	获取消息被死字母的原因。		字符串
CamelAzureServiceBusDeadLetterSource (consumer) 常数：DEAD_LETTER_SOURCE	在消息被排队之前，获取此消息排队的队列或订阅的名称。		字符串

Name	描述	默认值	类型
CamelAzureServiceBusDeliveryCount (consumer) 常量：DELIVERY_COUNT	获取此消息传送到客户端的次数。		long
CamelAzureServiceBusEnqueuedSequenceNumber (consumer) 常数：ENQUEUED_SEQUENCE_NUMBER	获取由服务总线分配给消息的排队序列号。		long
CamelAzureServiceBusEnqueuedTime (consumer) 常数：ENQUEUED_TIME	获取此消息在 Azure Service Bus 中排队的日期。		OffsetDateTime
CamelAzureServiceBusExpiresAt (consumer) 常数：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
CamelAzureServiceBusLockedUntil (consumer) 常数：LOCKED_UNTIL	获取此消息的锁定过期的日期。		OffsetDateTime
CamelAzureServiceBusMessageId (consumer) 常量：MESSAGE_ID	获取消息的标识符。		字符串

Name	描述	默认值	类型
CamelAzureServiceBusPartitionKey (consumer) 常量：PARTITION_KEY	获取向分区实体发送消息的分区密钥。		字符串
CamelAzureServiceBusRawAmqpMessage (consumer) 常量：RAW_AMQP_MESSAGE	AMQP 协议定义的消息表示。		AmqpAnnotatedMessage
CamelAzureServiceBusReplyTo (consumer) 常数：REPLY_TO	获取要发送回复的实体地址。		字符串
CamelAzureServiceBusReplyToSessionId (consumer) 常数：REPLY_TO_SESSION_ID	获取或设置增加 ReplyTo 地址的会话标识符。		字符串
CamelAzureServiceBusSequenceNumber (consumer) 常数：SEQUENCE_NUMBER	获取由服务总线分配给消息的唯一数字。		long
CamelAzureServiceBusSessionId (consumer) 常数：SESSION_ID	获取消息的会话 ID。		字符串

Name	描述	默认值	类型
CamelAzureServiceBusSubject (consumer) 常量：SUB JECT	获取邮件的主题。		字符串
CamelAzureServiceBusTimeToLive (consumer) 常数：TIME_TO_LIVE	获得此消息过期前的持续时间。		Duration
CamelAzureServiceBusTo (consumer) 常数： TO	获取地址。		字符串
CamelAzureServiceBusScheduledEnqueueTime (common) 常量：SCHEDULED_ENQUEUE_TIME	(producer)覆盖 OffsetDateTime，其中消息应出现在服务总线队列或主题中。(consumer)获取此消息的调度队列时间。		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
CamelAzureServiceBusProducerOperation (producer) 常数：PRODUCER_OPERATION	覆盖要在制作者中使用的所需操作。 Enum 值： • sendMessages • scheduleMessages		ServiceBusProducerOperationDefinition

9.5.1. 消息正文

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

在消费者中，返回的消息正文将是 'String。

9.5.2. Azure ServiceBus Producer 操作

操作	描述
sendMessages	使用批处理方法向服务总线队列或主题发送一组消息。
scheduleMessages	向此发送者连接的 Azure Service Bus 实体发送调度的消息。调度的消息已排队，且仅在调度的 enqueue 时间提供给接收器。

9.5.3. Azure ServiceBus Consumer 操作

操作	描述
receiveMessages	接收来自 Service Bus 实体的 infinite 流。
peekMessages	在不更改接收器或消息源的状态的情况下读取下一个活动消息批处理。

9.5.3.1. 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");
```

9.6. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 azure-servicebus 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-azure-servicebus-starter</artifactId>
</dependency>
```

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，则使用默认的重试选项。选项是一个 com.azure.core.amqp.AmqpRetryOptions 类型。		AmqpRetryOptions
camel.component.azure-servicebus.amqp-transport-type	设置发生与 Azure Service Bus 的所有通信的传输类型。默认值为 AmqpTransportType114AMQP。		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

Name	描述	默认值	类型
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	设置要从这个构建器构建的客户端发送的 ClientOptions，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。选项是一个 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	禁用收到消息的自动完成和自动启用。默认情况下，成功处理的消息为 \\{link ServiceBusReceiverAsyncClientöcomplete (ServiceBusReceivedMessage) completed}。如果在处理消息时发生错误，它将是 \\{link ServiceBusReceiverAsyncClientöabandon (ServiceBusReceivedMessage) abandoned}。	false	布尔值
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域名。		字符串

Name	描述	默认值	类型
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	设置继续自动更新锁定的时间。设置 Duration114ZERO 或 null 可禁用自动续订。对于 \\{link ServiceBusReceiveModeCEIVE_AND_DELETE RECEIVE_AND_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	设置在 peek 操作期间要显示的最大消息数。		整数
camel.component.azure-servicebus.prefetch-count	设置接收方的 prefetch 计数。对于 \\{link ServiceBusReceiveModeHQPEEK_LOCK PEEK_LOCK} 和 \\{link ServiceBusReceiveModeDSRECEIVE_AND_DELETE RECEIVE_AND_DELETE} 模式，默认值为 1。Prefetch 通过模拟在应用使用 ServiceBusReceiverAsyncClient receiveMessages （）请求一个消息之前，为本地检索提供消息流速度。设置非零值将预先获取该消息的数量。将值设为零关闭。		整数
camel.component.azure-servicebus.producer-operation	设置要在制作者中使用的所需操作。		ServiceBusProducerOperationDefinition
camel.component.azure-servicebus.proxy-options	设置用于 ServiceBusSenderAsyncClient 的代理配置。配置代理后，必须将 AmqpTransportTypetransAMQP_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，以便消费者使用消息。选项是一个 com.azure.messaging.servicebus.ServiceBusReceiverAsyncClient 类型。		ServiceBusReceiverAsyncClient

Name	描述	默认值	类型
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	设置在制作者中使用的 SenderAsyncClient。选项是一个 com.azure.messaging.servicebus.ServiceBusSenderAsyncClient 类型。		ServiceBusSenderAsyncClient
camel.component.azure-servicebus.service-bus-receive-mode	为接收方设置接收模式。		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	要执行的连接类型。队列用于典型的队列选项，以及基于订阅模型的主题。		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。还必须设置 topicOrQueueName 和 serviceBusType=topic。如果 serviceBusType=topic 和使用使用者，则需要此属性。		字符串
camel.component.azure-servicebus.token-credential	用于 Azure AD 身份验证的 TokenCredential，在 com.azure.identity 中实现。选项是一个 com.azure.core.credential.TokenCredential 类型。		TokenCredential

第 10 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，如果版本 v12，我们会看到此组件是否可以根据有问题的更改量而采用这些更改。

先决条件

您必须有一个有效的 Windows Azure Storage 帐户。如需更多信息，请参阅 Azure 文档门户。

Maven 用户需要将以下依赖项添加到其 **pom.xml** 中。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-azure-storage-blob</artifactId>
  <version>{CamelSBVersion}</version>
 <!-- use the same version as your Camel core version -->
</dependency>
```

10.1. URI 格式

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

如果是使用者，则需要 **accountName。如果是生成者，它取决于请求的操作，例如，如果操作位于容器级别上，例如： createContainer、accountName 和 containerName 只需要，但如果 blob 级别请求操作，例如 getBlob、accountName、containerName 和 blobName。**

如果 blob 尚不存在，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&...

10.2. 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**

10.2.1. 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。

10.2.2. 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。

10.3. 组件选项

Azure Storage Blob Service 组件支持 31 个选项，如下所列。

Name	描述	默认值	类型
blobName (common)	blob 名称，使用容器中的特定 blob。但是，在制作者上，只有 blob 级别上的操作才需要。		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
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，这包含重要的身份验证信息。		StorageSharedKeyCredential
dataCount (common)	范围内的字节数。如果指定，则必须大于或等于 0。		Long
fileDir (common)	下载的 Blob 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 Blob 数量上限，包括所有 BlobPrefix 元素。如果请求没有指定 maxResultsPerPage 或指定大于 5,000 的值，服务器将返回最多 5,000 个项目。		整数
maxRetryRequests (common)	指定从响应正文读取数据时将要进行的额外 HTTP 获取请求的最大数量。	0	int
prefix (common)	过滤结果，以仅返回名称以指定前缀开头的 Blob。可以是 null，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 prefix 和 regex，则正则表达式 可能是 null，则忽略 priority 和 prefix。		字符串
serviceClient (common)	**Autowired** Client 到存储帐户。此客户端不包含有关特定存储帐户的状态，而是便捷地将适当的请求发送到该服务上的资源。它还可用于将 URL 构造到 blob 和容器。此客户端包含服务帐户的操作。容器的操作可在 BlobContainerClient via BlobServiceClient114getBlobContainerClient (String) 上提供，并且 blob 上的操作可以通过 BlobContainerClient114getBlobClient (String)在 BlobClient 上提供。		BlobServiceClient

Name	描述	默认值	类型
timeout (common)	可选的超时值，超过 RuntimeException 会被引发。		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 到 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 ● 未提交 ● all	已提交	BlockListType
changeFeedContext (producer)	在使用 getChangeFeed producer 操作时，这提供了在服务调用期间通过 Http 管道传递的额外上下文。		Context
changeFeedEndTime (producer)	使用 getChangeFeed producer 操作时，这将过滤结果，以便在结束时间前返回事件。注： 也可以返回属于下一个小时的一些事件。属于此小时的几个事件可能会缺失；为了确保返回小时中的所有事件，以一小时结束时间。		OffsetDateTime
changeFeedStartTime (producer)	使用 getChangeFeed producer 操作时，这将过滤结果，以便在开始时间后大约返回事件。注： 也可以返回属于上一个小时的一些事件。属于此小时的几个事件可能会缺失；为了确保返回来自小时的所有事件，按一小时循环开始时间。		OffsetDateTime
closeStreamAfterWrite (producer)	在写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，暂存块不会直接提交。	true	布尔值
createAppendBlob (producer)	当设置为 true 时，在提交附加块时将创建附加块。	true	布尔值

Name	描述	默认值	类型
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时间(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
operation (producer)	可在制作者上与这个组件一起使用的 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

Name	描述	默认值	类型
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
accessKey (security)	用于连接 azure 帐户名称的访问密钥，用于使用 azure blob 服务进行身份验证。		字符串
sourceBlobAccessKey (security)	source Blob Access Key： 对于复制blob 操作，我们需要为源 Blob 有一个 accessKey，我们需要复制 accessKey 作为标头，因此我们可以设置为密钥。		字符串

10.4. 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：

10.4.1. 路径参数(2 参数)

Name	描述	默认值	类型
accountName (common)	用于使用 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

10.4.2. 查询参数(48 参数)

Name	描述	默认值	类型
blobName (common)	blob 名称，使用容器中的特定 blob。但是，在制作者上，只有 blob 级别上的操作才需要。		字符串

Name	描述	默认值	类型
blobOffset (common)	为上传或下载操作设置 blob 偏移，默认为 0。	0	long
blobServiceClient (common)	客户端到存储帐户。此客户端不包含有关特定存储帐户的状态，而是便捷地将适当的请求发送到该服务上的资源。它还可用于将 URL 构造到 blob 和容器。此客户端包含服务帐户的操作。容器的操作可以通过 getBlobContainerClient (String)在 BlobContainerClient (String)上提供，并且 BlobClient 上的操作可以通过 getBlobContainerClient (String).getBlobClient (String)获得。		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，这包含重要的身份验证信息。		StorageSharedKeyCredential
dataCount (common)	范围内的字节数。如果指定，则必须大于或等于 0。		Long
fileDir (common)	下载的 Blob 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 Blob 数量上限，包括所有 BlobPrefix 元素。如果请求没有指定 maxResultsPerPage 或指定大于 5,000 的值，服务器将返回最多 5,000 个项目。		整数
maxRetryRequests (common)	指定从响应正文读取数据时将要进行的额外 HTTP 获取请求的最大数量。	0	int
prefix (common)	过滤结果，以仅返回名称以指定前缀开头的 Blob。可以是 null，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 prefix 和 regex，则正则表达式 可能是 null，则忽略 priority 和 prefix。		字符串

Name	描述	默认值	类型
serviceClient (common)	**Autowired** Client 到存储帐户。此客户端不包含有关特定存储帐户的状态，而是便捷地将适当的请求发送到该服务上的资源。它还可用于将 URL 构造到 blob 和容器。此客户端包含服务帐户的操作。容器的操作可在 BlobContainerClient via BlobServiceClient114getBlobContainerClient (String) 上提供，并且 blob 上的操作可以通过 BlobContainerClient114getBlobClient (String)在 BlobClient 上提供。		BlobServiceClient
timeout (common)	可选的超时值，超过 RuntimeException 会被引发。		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
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 到 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 • 未提交 • all	已提交	BlockListType
changeFeedContext (producer)	在使用 getChangeFeed producer 操作时，这提供了在服务调用期间通过 Http 管道传递的额外上下文。		Context
changeFeedEndTime (producer)	使用 getChangeFeed producer 操作时，这将过滤结果，以便在结束时间前返回事件。注： 也可以返回属于下一个小时的一些事件。属于此小时的几个事件可能会缺失；为了确保返回小时中的所有事件，以一小时结束时间。		OffsetDateTime
changeFeedStartTime (producer)	使用 getChangeFeed producer 操作时，这将过滤结果，以便在开始时间后大约返回事件。注： 也可以返回属于上一个小时的一些事件。属于此小时的几个事件可能会缺失；为了确保返回来自小时的所有事件，按一小时循环开始时间。		OffsetDateTime
closeStreamAfterWrite (producer)	在写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，暂存块不会直接提交。	true	布尔值
createAppendBlob (producer)	当设置为 true 时，在提交附加块时将创建附加块。	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时间(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)	可在制作者上与这个组件一起使用的 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
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)	用于连接 azure 帐户名称的访问密钥，用于使用 azure blob 服务进行身份验证。		字符串
sourceBlobAccessKey (security)	source Blob Access Key： 对于复制blob 操作，我们需要为源 Blob 有一个 accessKey，我们需要复制 accessKey 作为标头，因此我们可以设置为密钥。		字符串

所需信息选项

要使用此组件，您需要 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 BlobServiceClient 实例，它可以提供给 blobServiceClient。注： 您不需要创建特定的客户端，如 BlockBlobClient，BlobServiceClient 代表顶层，可用于检索较低级别的客户端。**

10.5. 使用方法

例如，要从 camelazure 存储帐户中的 container1 上的块 blob hello.txt 下载 blob 内容，请使用以下代码片段：

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

10.5.1. 组件制作者评估的消息标头

标头	变量名称	类型	操作	描述
CamelAzureStorageBlobTimeout	**BlobConstants.TIMEOUT**	**Duration**	All	可选的超时值，超过 {@link RuntimeException} 将引发 {@link RuntimeException}。
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对公共使用的方式。如果无公共访问，传递 **null**。
CamelAzureStorageBlobRequestCondition	**BlobConstants.BLOB_REQUEST_CONDITION**	**BlobRequestConditions**	与容器和 Blob 相关的操作	它包含值，将限制各种请求的成功操作到存在的条件。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。可以是 null，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 Blob 数量上限，包括所有 BlobPrefix 元素。如果请求没有指定 maxResultsPerPage 或指定大于 5,000 的值，服务器将返回最多 5,000 个项目。
CamelAzureStorageBlobListBlobOptions	**BlobConstants.LIST_BLOB_OPTIONS**	**ListBlobsOptions**	**listBlobs**	定义可用于配置 {@link BlobContainerClient} 对象上 listBlobsFlatSegment 的调用行为的选项。
CamelAzureStorageBlobHttpHeaders	**BlobConstants.BLOB_HTTP_HEADERS**	**BlobHttpHeaders**	**uploadBlockBlob**, **commitBlobBlockList**, **createAppendBlob**, **createPageBlob**	一组操作的额外参数。
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，存储服务会比较与此标头值到达的内容的哈希值。请注意，这个 MD5 哈希不使用 blob 存储。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。假设页面必须与 512 字节界限一致，开始偏移必须是 512ulus 的 modulus，结束偏移必须是 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，暂存块不会直接提交。
CamelAzureStorageBlobCreateAppendBlob	**BlobConstants.CREATE_APPEND_BLOB**	**布尔值**	**commitAppendBlob**	当设置为 **true** 时，在提交附加块时将创建附加块。
CamelAzureStorageBlobCreatePageBlob	**BlobConstants.CREATE_PAGE_BLOB**	**布尔值**	**uploadPageBlob**	当设置为 **true** 时，在上传页面 blob 时将创建页面 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 到 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 上快照的行为。\{@code Include} 将删除基本 blob 和所有快照。\{@code Only} 将只删除快照。如果快照被删除，您必须传递 null。
CamelAzureStorageBlobListBlobContainersOptions	**BlobConstants.LIST_BLOB_CONTAINERS_OPTIONS**	**ListBlobContainersOptions**	**listBlobContainers**	{@link ListBlobContainersOptions}，用于指定该服务应返回哪些数据。
CamelAzureStorageBlobParallelTransferOptions	**BlobConstants.PARALLEL_TRANSFER_OPTIONS**	**ParallelTransferOptions**	**downloadBlobToFile**	用于下载到文件的 {@link ParallelTransferOptions}。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时间 (millis)。
CamelAzureStorageBlobBlobName	**BlobConstants.BLOB_NAME**	**字符串**	与 blob 相关的操作	覆盖/设置交换标头中的 blob 名称。
CamelAzureStorageBlobContainerName	**BlobConstants.BLOB_CONTAINER_NAME**	**字符串**	与容器和 Blob 相关的操作	覆盖/设置交换标头中的容器名称。
CamelAzureStorageBlobOperation	**BlobConstants.BLOB_OPERATION**	**BlobOperationsDefinition**	All	指定要执行的制作者操作，请参阅此页面上与制作者操作相关的 doc。

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。可以是 null 以返回所有。如果同时设置了 prefix 和 regex，则 regex 会忽略 priority 和 prefix。
CamelAzureStorageBlobChangeFeedStartTime	**BlobConstants.CHANGE_FEED_START_TIME**	**OffsetDateTime**	**getChangeFeed**	它过滤结果以在开始时间后大约返回事件。注： 也可以返回属于上一个小时的一些事件。属于此小时的几个事件可能会缺失；为了确保返回来自小时的所有事件，按一小时循环开始时间。
CamelAzureStorageBlobChangeFeedEndTime	**BlobConstants.CHANGE_FEED_END_TIME**	**OffsetDateTime**	**getChangeFeed**	它过滤结果以在结束时间之前返回事件。注： 也可以返回属于下一个小时的一些事件。属于此小时的几个事件可能会缺失；为了确保返回小时中的所有事件，以一小时结束时间。
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

10.5.2. 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**	访问 blob 的层。
CamelAzureStorageBlobAccessTierChangeTime	**BlobConstants.ACCESS_TIER_CHANGE_TIME**	**OffsetDateTime**	当 blob 的访问层最后一次更改时，日期时间。
CamelAzureStorageBlobArchiveStatus	**BlobConstants.ARCHIVE_STATUS**	**ArchiveStatus**	blob 的归档状态。
CamelAzureStorageBlobCreationTime	**BlobConstants.CREATION_TIME**	**OffsetDateTime**	blob 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**	提交到附加 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。

标头	变量名称	类型	描述
CamelAzureStorageBlobContentLanguage	**BlobConstants.CONTENT_LANGUAGE**	**字符串**	为 blob 指定的内容语言。
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 上的最后一次复制操作完成后，日期时间。
CamelAzureStorageBlobCopyDestinationSnapshot	**BlobConstants.COPY_DESTINATION_SNAPSHOT**	**字符串**	blob 最后一次增量副本快照的快照标识符。
CamelAzureStorageBlobCopyId	**BlobConstants.COPY_ID**	**字符串**	blob 上执行的最后复制操作的标识符。
CamelAzureStorageBlobCopyProgress	**BlobConstants.COPY_PROGRESS**	**字符串**	在 blob 上执行最后一次复制操作的进度。
CamelAzureStorageBlobCopySource	**BlobConstants.COPY_SOURCE**	**字符串**	在 blob 上执行最后一次复制操作的来源。
CamelAzureStorageBlobCopyStatus	**BlobConstants.COPY_STATUS**	**CopyStatusType**	blob 上执行的最后复制操作的状态。
CamelAzureStorageBlobCopyStatusDescription	**BlobConstants.COPY_STATUS_DESCRIPTION**	**字符串**	blob 上最后一个复制操作的描述。
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 的属性中推断出来的标志。

标头	变量名称	类型	描述
CamelAzureStorageBlobIsIncrementalCopy	**BlobConstants.IS_INCREMENTAL_COPY**	**布尔值**	表示 blob 是否已递增复制的标志。
CamelAzureStorageBlobIsServerEncrypted	**BlobConstants.IS_SERVER_ENCRYPTED**	**布尔值**	表示 blob 的内容是否在服务器上加密的标志。
CamelAzureStorageBlobLastModified	**BlobConstants.LAST_MODIFIED**	**OffsetDateTime**	当 Blob 最后一次修改时，日期时间。
CamelAzureStorageBlobLeaseDuration	**BlobConstants.LEASE_DURATION**	**LeaseDurationType**	blob 上的租期类型。
CamelAzureStorageBlobLeaseState	**BlobConstants.LEASE_STATE**	**LeaseStateType**	blob 上的租期状态。
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。
CamelAzureStorageBlobFileName	**BlobConstants.FILE_NAME**	**字符串**	从操作下载的文件名 **downloadBlobToFile**。
CamelAzureStorageBlobDownloadLink	**BlobConstants.DOWNLOAD_LINK**	**字符串**	由 **downloadLink** 操作生成的下载链接。
CamelAzureStorageBlobRawHttpHeaders	**BlobConstants.RAW_HTTP_HEADERS**	**httpHeaders**	返回用户可以使用的未解析 httpHeaders。

10.5.3. 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 实例配置拥有更多控制，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 组件配置中引用这个实例：

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

10.5.4. 在 registry 中自动检测 BlobServiceClient 客户端

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

10.5.5. Azure Storage Blob Producer 操作

Camel Azure Storage Blob 组件在制作者端提供广泛的操作：

对服务级别的操作

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	返回对存储帐户中的 blob 和 blob 元数据的所有更改的事务日志。更改源提供这些更改的排序、保证、持久性、不可变、只读日志。

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。如果存在具有相同名称的容器，则制作者将忽略它。
deleteBlobContainer	删除存储帐户中指定的容器。如果容器不存在，则操作会失败。
listBlobs	返回此容器中的 blob 列表，文件夹结构扁平化。

对 **blob 级别的操作**

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

操作	blob 类型	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	使用共享访问令牌(SAS)为指定的 blob 生成下载链接。默认情况下，仅允许 1 小时访问。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**	将指定块上传到块 blob 的"staging 区域"，以便稍后由对 commitBlobBlockList 的调用提交。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。

操作	blob 类型	描述
commitBlobBlockList	**BlockBlob**	通过指定要组成 blob 的块 ID 列表来写入 blob。为了作为 blob 的一部分编写，必须在之前的 **stageBlockBlobList** 操作中成功写入服务器。您可以调用 **commitBlobBlockList** 来仅上传更改的块来更新 Blob，然后将新和现有块提交在一起。任何没有在块列表中指定的块，并永久删除。
getBlobBlockList	**BlockBlob**	使用指定的块列表过滤器返回作为块 blob 的一部分上传的块列表。
createAppendBlob	**AppendBlob**	创建 0-length 附加 blob。调用 commitAppendBlo'b 操作，将数据附加到附加 Blob。
commitAppendBlob	**AppendBlob**	向现有附加 Blob 末尾提交一个新的数据块。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。
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 调整为指定大小（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 快照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，即使来自不同帐户。

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。

10.5.6. 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，您还可以直接在不使用文件组件的情况下写入文件，您需要指定 fileDir 文件夹路径，以便在您的机器中保存 blob。

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

另外，组件支持批处理消费者，因此您可以消耗多个 Blob，它只指定容器名称，使用者将根据容器中的 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");
```

10.5.7. 生成者操作示例

- 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:

我们都可以在交换正文中设置 outputStream，并将数据写入其中。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，以便进一步下游：

```
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");
```

- 下载BlobToFile ：

```
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' 的容器容器容器中的 file.txt 将被复制到同一帐户的容器容器blob2 中。

10.5.8. 开发备注(Important)

所有集成测试都使用 Testcontainers 并运行。需要获取 Azure accessKey 和 accountName 才能使用 Azure 服务运行所有集成测试。除了模拟单元测试外，还需要使用您进行的每个更改来运行集成测试，甚至客户端升级，因为 Azure 客户端可能会在次版本升级时破坏问题。要运行集成测试，在这个组件目录中运行以下 maven 命令：

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

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

10.6. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 azure-storage-blob 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-azure-storage-blob-starter</artifactId>
</dependency>
```

组件支持 32 个选项，如下所列。

Name	描述	默认值	类型
camel.component.azure-storage-blob.access-key	用于连接 azure 帐户名称的访问密钥，用于使用 azure blob 服务进行身份验证。		字符串
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，使用容器中的特定 blob。但是，在制作者上，只有 blob 级别上的操作才需要。		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 到 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

Name	描述	默认值	类型
camel.component.azure-storage-blob.change-feed-context	在使用 getChangeFeed producer 操作时，这提供了在服务调用期间通过 Http 管道传递的额外上下文。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，这将过滤结果，以便在结束时间前返回事件。注： 也可以返回属于下一个小时的一些事件。属于此小时的几个事件可能会缺失；为了确保返回小时中的所有事件，以一小时结束时间。选项是一个 java.time.OffsetDateTime 类型。		OffsetDateTime
camel.component.azure-storage-blob.change-feed-start-time	使用 getChangeFeed producer 操作时，这将过滤结果，以便在开始时间后大约返回事件。注： 也可以返回属于上一个小时的一些事件。属于此小时的几个事件可能会缺失；为了确保返回来自小时的所有事件，按一小时循环开始时间。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，暂存块不会直接提交。	true	布尔值
camel.component.azure-storage-blob.configuration	组件配置。选项是一个 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，在提交附加块时将创建附加块。	true	布尔值

Name	描述	默认值	类型
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.credentials	可以注入 StorageSharedKeyCredential 来创建 azure 客户端，这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-blob.data-count	范围内的字节数。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时间(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 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.lazy-start-producer	生成者是否应懒惰启动 （在第一个消息中）。通过懒惰启动，您可以使用此选项来允许 CamelContext 和路由在生成者启动期间启动，并导致路由启动失败。通过懒惰启动，启动失败可以在路由信息时通过 Camel 的路由错误处理程序进行处理。请注意，在处理第一个消息时，创建并启动生成者可能需要稍等时间，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 Blob 数量上限，包括所有 BlobPrefix 元素。如果请求没有指定 maxResultsPerPage 或指定大于 5,000 的值，服务器将返回最多 5,000 个项目。		整数
camel.component.azure-storage-blob.max-retry-requests	指定从响应正文读取数据时将要进行的额外 HTTP 获取请求的最大数量。	0	整数
camel.component.azure-storage-blob.operation	可在制作者上与这个组件一起使用的 blob 操作。		BlobOperationsDefinition

Name	描述	默认值	类型
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。可以是 null，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 prefix 和 regex，则正则表达式 可能是 null，则忽略 priority 和 prefix。		字符串
camel.component.azure-storage-blob.service-client	客户端到存储帐户。此客户端不包含有关特定存储帐户的状态，而是便捷地将适当的请求发送到该服务上的资源。它还可用于将 URL 构造到 blob 和容器。此客户端包含服务帐户的操作。容器的操作可在 BlobContainerClient via BlobServiceClient114getBlobContainerClient (String) 上提供，并且 blob 上的操作可以通过 BlobContainerClient114getBlobClient (String)在 BlobClient 上提供。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： 对于复制blob 操作，我们需要为源 Blob 有一个 accessKey，我们需要复制 accessKey 作为标头，因此我们可以设置为密钥。		字符串
camel.component.azure-storage-blob.timeout	可选的超时值，超过 RuntimeException 会被引发。选项是一个 java.time.Duration 类型。		Duration

第 11 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，如果版本 v12，我们会看到此组件是否可以根据有问题的更改量而采用这些更改。

先决条件

您必须有一个有效的 Windows Azure Storage 帐户。如需更多信息，请参阅 Azure 文档门户。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-azure-storage-queue</artifactId>
  <version>{CamelSBVersion}</version>
 <!-- use the same version as your Camel core version -->
</dependency>
```

11.1. URI 格式

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

如果是 consumer，则需要 accountName 和 queueName。如果是生成者，它取决于请求的操作，例如，如果操作位于服务级别上，例如：listQueues，仅需要 accountName，但在队列级别上请求 operationName，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&...
```

11.2. 配置选项

Camel 组件在两个独立级别上配置：

- 组件级别
- 端点级别

11.2.1. 配置组件选项

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。

11.2.2. 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。

11.3. 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。此客户端不包含有关特定存储帐户的状态，而是便捷地将适当的请求发送到该服务上的资源。此客户端包含与 Azure Storage 中的队列帐户交互的所有操作。客户端允许的操作正在创建、列出和删除队列、检索和更新帐户的属性，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，在发送消息到队列时会自动创建队列。	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
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

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
maxMessages (queue)	如果队列中存在比请求所有消息少的消息数量，则需要获取的最大消息数。如果只检索空 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的 ID。		字符串
popReceipt (queue)	必须匹配的唯一标识符才能删除或更新消息。		字符串
Timeout (queue)	应用到操作的可选超时。如果在超时结束前未返回响应，则会抛出 RuntimeException。		Duration
timeToLive (queue)	消息将在队列中保持活跃的时长。如果未设置该值，则默认值为 7 天，如果 -1 传递，则消息不会过期。生存时间必须是 -1 或任意正数。格式应为 20.345 秒：PnDTnHnMn.nS.，例如：PT20.345S 345- parses 为 20.345，因为这些 Java API 是类型safe。		Duration
visibilityTimeout (queue)	消息在队列中不可见的超时时间。超时时间必须在 1 秒和 7 天之间。格式应为 20.345 秒：PnDTnHnMn.nS.，例如：PT20.345S 345- parses 为 20.345，因为这些 Java API 是类型safe。		Duration
accessKey (security)	用于连接 azure 帐户名称的访问密钥，用于使用 azure 队列服务进行身份验证。		字符串
凭证 （安全）	可以注入 StorageSharedKeyCredential 来创建 azure 客户端，这包含重要的身份验证信息。		StorageSharedKeyCredential

11.4. 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：

11.4.1. 路径参数(2 参数)

Name	描述	默认值	类型
accountName (common)	用于使用 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

11.4.2. 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。此客户端不包含有关特定存储帐户的状态，而是便捷地将适当的请求发送到该服务上的资源。此客户端包含与 Azure Storage 中的队列帐户交互的所有操作。客户端允许的操作正在创建、列出和删除队列、检索和更新帐户的属性，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
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
createQueue (producer)	当设置为 true 时，在发送消息到队列时会自动创建队列。	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
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)	如果队列中存在比请求所有消息少的消息数量，则需要获取的最大消息数。如果只检索空 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的 ID。		字符串
popReceipt (queue)	必须匹配的唯一标识符才能删除或更新消息。		字符串

Name	描述	默认值	类型
Timeout (queue)	应用到操作的可选超时。如果在超时结束前未返回响应，则会抛出 RuntimeException。		Duration
timeToLive (queue)	消息将在队列中保持活跃的时长。如果未设置该值，则默认值为 7 天，如果 -1 传递，则消息不会过期。生存时间必须是 -1 或任意正数。格式应为 20.345 秒：PnDTnHnMn.nS.，例如：PT20.345S 345- parses 为 20.345，因为这些 Java API 是类型safe。		Duration
visibilityTimeout (queue)	消息在队列中不可见的超时时间。超时时间必须在 1 秒和 7 天之间。格式应为 20.345 秒：PnDTnHnMn.nS.，例如：PT20.345S 345- parses 为 20.345，因为这些 Java API 是类型safe。		Duration
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)	用于连接 azure 帐户名称的访问密钥，用于使用 azure 队列服务进行身份验证。		字符串
凭证 （安全）	可以注入 StorageSharedKeyCredential 来创建 azure 客户端，这包含重要的身份验证信息。		StorageSharedKeyCredential

所需信息选项

要使用此组件，您需要 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： 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。

11.5. 使用方法

例如，若要从 storageAccount 存储帐户中的队列 消息Queue 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");
```

11.5.1. 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	可选的超时值，超过 \{@link RuntimeException} 将引发。

标头	变量名称	类型	操作	描述
CamelAzureStorageQueueMetadata	**QueueConstants.METADATA**	**Map<String,String>**	**createQueue**	与队列关联的元数据
CamelAzureStorageQueueTimeToLive	**QueueConstants.TIME_TO_LIVE**	**Duration**	**sendMessage**	消息将在队列中保持活跃的时长。如果未设置该值，则默认值为 7 天，如果 -1 传递，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，则消息将立即可见。超时时间必须在 0 秒和 7 天之间。
CamelAzureStorageQueueCreateQueue	**QueueConstants.CREATE_QUEUE**	**布尔值**	**sendMessage**	当设置为 **true** 时，在发送消息到队列时会自动创建队列。
CamelAzureStorageQueuePopReceipt	**QueueConstants.POP_RECEIPT**	**字符串**	**DeleteMessage**, **updateMessage**	必须匹配的唯一标识符才能删除或更新消息。
CamelAzureStorageQueueMessageId	**QueueConstants.MESSAGE_ID**	**字符串**	**DeleteMessage**, **updateMessage**	要删除或更新的消息的 ID。
CamelAzureStorageQueueMaxMessages	**QueueConstants.MAX_MESSAGES**	**整数**	**receiveMessages**,**peekMessages**	如果队列中存在比请求所有消息少的消息数量，则需要获取的最大消息数。如果只检索空 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，请参阅此页面上与制作者操作相关的 doc。

标头	变量名称	类型	操作	描述
CamelAzureStorageQueueName	QueueConstants.QUEUE_NAME	字符串	All	覆盖队列名称。

11.5.2. 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	消息插入到 Queue 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，则此弹出信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	消息被取消队列的次数。
CamelAzureStorageQueueRawHttpHeaders	QueueConstants.RAW_HTTP_HEADERS	httpHeaders	返回用户可以使用的未解析 httpHeaders。

11.5.3. 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 实例配置拥有更多控制，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 组件配置中引用这个实例：

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

11.5.4. 在 registry 中自动检测 QueueServiceClient 客户端

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

11.5.5. Azure Storage Queue Producer 操作

Camel Azure Storage Queue 组件在制作者端提供广泛的操作：

对服务级别的操作

对于这些操作，需要 accountName。

操作	描述
listQueues	列出存储帐户中从指定标记开始的过滤器的队列。

对队列级别的操作

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。

操作	描述
sendMessage	**默认 Producer 操作** 会发送一个带有给定生存时间的消息，以及消息在队列中不可见的超时周期。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** 或标头 **CamelAzureStorageQueueCreateQueue** 设置为 **false**。
deleteMessage	删除队列中指定的消息。
receiveMessages	最多从队列检索最大消息数，并从超时时间的其他操作中隐藏它们。但是，由于可靠性的原因，它不会将消息从队列中分离。
peekMessages	来自队列前面的消息，最高到消息的最大数量。
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。

11.5.6. 消费者示例

要在一个批处理中最多 5 个消息的文件中消耗队列，如下所示：

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

11.5.7. 生成者操作示例

- **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");
```

- 清除Queue ：

```
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");
```

11.5.8. 开发备注(Important)

在此组件上开发时，您需要获取 Azure accessKey 才能运行集成测试。除了模拟单元测试外，还需要使用您进行的每个更改来运行集成测试，甚至客户端升级，因为 Azure 客户端可能会在次版本升级时破坏问题。要运行集成测试，在这个组件目录中运行以下 maven 命令：

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

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

11.6. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 azure-storage-queue 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-azure-storage-queue-starter</artifactId>
</dependency>
```

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	用于连接 azure 帐户名称的访问密钥，用于使用 azure 队列服务进行身份验证。		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

Name	描述	默认值	类型
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。选项是一个 org.apache.camel.component.azure.storage.queue.QueueConfiguration 类型。		QueueConfiguration
camel.component.azure-storage-queue.create-queue	当设置为 true 时，在发送消息到队列时会自动创建队列。	false	布尔值
camel.component.azure-storage-queue.credentials	可以注入 StorageSharedKeyCredential 来创建 azure 客户端，这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
camel.component.azure-storage-queue.max-messages	如果队列中存在比请求所有消息少的消息数量，则需要获取的最大消息数。如果只检索空 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的 ID。		字符串
camel.component.azure-storage-queue.operation	队列服务操作提示到制作者。		QueueOperationDefinition

Name	描述	默认值	类型
camel.component.azure-storage-queue.pop-receipt	必须匹配的唯一标识符才能删除或更新消息。		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。此客户端不包含有关特定存储帐户的状态，而是便捷地将适当的请求发送到该服务上的资源。此客户端包含与 Azure Storage 中的队列帐户交互的所有操作。客户端允许的操作正在创建、列出和删除队列、检索和更新帐户的属性，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	消息将在队列中保持活跃的时长。如果未设置该值，则默认值为 7 天，如果 -1 传递，则消息不会过期。生存时间必须是 -1 或任意正数。格式应为 20.345 秒：PnDTnHnMn.nS.，例如：PT20.345S 345- parses 为 20.345，因为这些 Java API 是类型safe。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。如果在超时结束前未返回响应，则会抛出 RuntimeException。选项是一个 java.time.Duration 类型。		Duration
camel.component.azure-storage-queue.visibility-timeout	消息在队列中不可见的超时时间。超时时间必须在 1 秒和 7 天之间。格式应为 20.345 秒：PnDTnHnMn.nS.，例如：PT20.345S 345- parses 为 20.345，因为这些 Java API 是类型safe。选项是一个 java.time.Duration 类型。		Duration

第 12 章 BEAN

仅支持生成者

Bean 组件将 Bean 绑定到 Camel 消息交换。

12.1. URI 格式

```
bean:beanName[?options]
```

其中 beanID 可以是用于在 Registry 中查找 bean 的任何字符串

12.2. 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

12.2.1. 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。

12.2.2. 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。

12.3. 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** 使用单例选项。	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） 时，仅创建或只查找一次，并在端点生命周期内重复使用一次。如果并发线程同时调用 bean，则 bean 应该是 thread-safe。在使用请求范围时，根据请求创建或查找一次（交换）。如果您要在处理请求时将状态存储在 bean 中，且您希望在处理请求时多次调用同一 bean 实例，则可以使用此项。bean 不必线程安全，因为实例仅从同一请求调用。在使用委派范围时，将根据调用查找或创建 bean。但是，如果查找，则这会被委派给 bean registry，如 Spring 或 CDI （如果使用）,这依赖于它们的配置，也可以作为单例范围或与之分配的范围。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

12.4. 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：

12.4.1. 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

12.4.2. 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） 时，仅创建或只查找一次，并在端点生命周期内重复使用一次。如果并发线程同时调用 bean，则 bean 应该是 thread-safe。在使用请求范围时，根据请求创建或查找一次（交换）。如果您要在处理请求时将状态存储在 bean 中，且您希望在处理请求时多次调用同一 bean 实例，则可以使用此项。bean 不必线程安全，因为实例仅从同一请求调用。使用过期范围时，将根据调用查找或创建 bean。但是，如果查找，则这会被委派给 bean registry，如 Spring 或 CDI （如果使用）,这依赖于它们的配置，也可以作为单例范围或与之分配的范围。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 上配置附加属性。		Map

12.5. 使用

用于使用消息的对象实例必须使用 Registry 显式注册。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。

注册端点后，您可以构建用于处理交换的 Camel 路由。

bean： 端点不能定义为路由的输入；例如，您无法从其中使用，您只能从某些入站消息端点路由到 bean 端点。因此请考虑使用 direct: 或 queue: 端点作为输入。

您可以使用 ProxyHelper 上的 createProxy （） 方法创建一个代理，该代理将生成交换并将其发送到任何端点：

以及使用 XML DSL 的同一路由：

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

12.6. BEAN 作为端点

Camel 还支持将 Bean 作为端点调用。这是因为当交换路由到 myBean Camel 时，使用 Bean Binding 来调用 bean。bean 的源仅是一个普通 POJO。

Camel 将使用 Bean Binding 调用声明 Hello 方法，方法是将 Exchange 的 In body 转换为 String 类型，并将方法的输出存储在 Exchange Out 正文上。

12.7. JAVA DSL BEAN 语法

Java DSL 附带组件的总语法。您可以使用以下语法，而不是将 bean 显式指定为端点（即 （即 ("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，而不是将名称传递给 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);
```

12.8. BEAN 绑定

如何选择 bean 方法（如果通过 method 参数明确指定）以及如何从 Message 构建参数值由 Bean Binding 机制定义，这些机制在 Camel 中的所有 Bean 集成机制中使用。

12.9. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

组件支持 13 个选项，如下所列。

Name	描述	默认	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

Name	描述	默认	类型
camel.component.bean.scope	bean 范围。使用单例范围（默认） 时，仅创建或只查找一次，并在端点生命周期内重复使用一次。如果并发线程同时调用 bean，则 bean 应该是 thread-safe。在使用请求范围时，根据请求创建或查找一次（交换）。如果您要在处理请求时将状态存储在 bean 中，且您希望在处理请求时多次调用同一 bean 实例，则可以使用此项。bean 不必线程安全，因为实例仅从同一请求调用。在使用委派范围时，将根据调用查找或创建 bean。但是，如果查找，则这会被委派给 bean registry，如 Spring 或 CDI （如果使用）,这依赖于它们的配置，也可以作为单例范围或与之分配的范围。因此，这取决于委派的 registry。		BeanScope
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） 时，仅创建或只查找一次，并在端点生命周期内重复使用一次。如果并发线程同时调用 bean，则 bean 应该是 thread-safe。在使用请求范围时，根据请求创建或查找一次（交换）。如果您要在处理请求时将状态存储在 bean 中，且您希望在处理请求时多次调用同一 bean 实例，则可以使用此项。bean 不必线程安全，因为实例仅从同一请求调用。在使用委派范围时，将根据调用查找或创建 bean。但是，如果查找，则这会被委派给 bean registry，如 Spring 或 CDI （如果使用）,这依赖于它们的配置，也可以作为单例范围或与之分配的范围。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

Name	描述	默认	类型
camel.language.bean.scope	bean 范围。使用单例范围（默认） 时，仅创建或只查找一次，并在端点生命周期内重复使用一次。如果并发线程同时调用 bean，则 bean 应该是 thread-safe。在使用请求范围时，根据请求创建或查找一次（交换）。如果您要在处理请求时将状态存储在 bean 中，且您希望在处理请求时多次调用同一 bean 实例，则可以使用此项。bean 不必线程安全，因为实例仅从同一请求调用。使用过期范围时，将根据调用查找或创建 bean。但是，如果查找，则这会被委派给 bean registry，如 Spring 或 CDI （如果使用）,这取决于它们的配置可以充当单例范围，也可以充当其他范围。因此，在使用过期范围时，这取决于 bean registry 实现。	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
camel.component.bean.cache	**弃用了** 使用单例选项。	true	布尔值
camel.component.class.cache	**弃用了** 使用单例选项。	true	布尔值

第 13 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 使用参考实施，即 Hibernate Validator。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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

13.1. 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&…

13.2. 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

13.2.1. 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。

13.2.2. 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。

13.3. 组件选项

Bean Validator 组件支持 8 个选项，如下所列。

Name	描述	默认	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	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 TraversableResolver。		TraversableResolver
validationProviderResolver (advanced)	使用自定义 ValidationProviderResolver。		ValidationProviderResolver
ValidatorFactory (advanced)	**Autowired** 使用自定义 ValidatorFactory。		ValidatorFactory

13.4. 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：

13.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
label (producer)	**必需** 的位置标签是描述端点的任意文本值。		字符串

13.4.2. 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 TraversableResolver。		TraversableResolver
validationProviderResolver (advanced)	使用自定义 ValidationProviderResolver。		ValidationProviderResolver
ValidatorFactory (advanced)	使用自定义 ValidatorFactory。		ValidatorFactory

13.5. 过程部署

要在 IaaS 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 一样。以下片段演示了这种方法。您还可以使用 HibernateValidationProviderResolver。

Using 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 sVirt 环境中，则会自动使用 HibernateValidationProviderResolver。

13.6. 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，只有属性制造商和 licensePlate 的 @NotNull 约束才会被验证(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 中检查约束，则必须定义路由，如下所示

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

如果要从这两个组中检查约束，您必须首先定义一个新接口

AllChecks.java

```
@GroupSequence({Default.class, OptionalChecks.class})
public interface AllChecks {
}
```

然后您的路由定义应类似如下

```
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工厂，您必须编写如下路由：

```
<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")
```

也可以将约束描述为 XML，而不是作为 Java 注解描述。在这种情况下，您必须提供文件 META-INF/validation.xml，该文件可能如下所示

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 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>
```

13.7. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

-

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。选项是一个 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 TraversableResolver。选项是一个 javax.validation.TraversableResolver 类型。		TraversableResolver
camel.component.bean-validator.validation-provider-resolver	使用自定义 ValidationProviderResolver。选项是一个 javax.validation.ValidationProviderResolver 类型。		ValidationProviderResolver

Name	描述	默认值	类型
camel.component.bean-validator.validator-factory	使用自定义 ValidatorFactory。选项是 javax.validation.ValidatorFactory 类型。		ValidatorFactory

第 14 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用于浏览。

14.1. URI 格式

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

其中 someName 可以是唯一标识端点的任意字符串。

14.2. 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

14.2.1. 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。

14.2.2. 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。

14.3. 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
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

14.4. 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：

14.4.1. 路径参数(1 参数)

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

14.4.2. 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
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

14.5. 示例

在以下路由中，我们插入一个 browse: 组件，以便浏览要传递的交换：

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

现在，我们可以检查来自 Java 代码中的接收的交换：

```
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
   }
}
```

14.6. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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

Name	描述	默认值	类型
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 15 章 CASSANDRA CQL

支持生成者和消费者

Apache Cassandra 是一个开源 IaaS 数据库，旨在处理商业硬件上的大量数据。与 Amazon 的 DynamoDB 一样，Casasa 具有对等和 master-less 架构，以避免单一故障点和高可用性。与 Google 的 TriTable 一样，Casasra 数据是使用列系列的结构化，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 DataStax 提供的 Cassandra Java 驱动程序。

15.1. 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

15.1.1. 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。

15.1.2. 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。

15.2. 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
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

15.3. 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：

15.3.1. 路径参数(4 参数)

Name	描述	默认值	类型
beanRef (common)	beanRef 使用 bean:id 来定义。		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
键空间 (common)	要使用的键空间。		字符串

15.3.2. 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 • QUORUM • ALL • LOCAL_ONE • LOCAL_QUORUM • EACH_QUORUM • SERIAL • LOCAL_SERIAL		DefaultConsistencyLevel
cql (common)	要执行的 CQL 查询。可以使用键 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)	要使用实施 ResultSet 的逻辑将 ResultSet 转换为消息 body ALL, ONE, LIMIT_10, LIMIT_100... 的自定义类		ResultSetConversionStrategy
session (common)	使用 Session 实例（通常不要使用这个选项）。		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

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

15.4. 端点连接语法

端点可以启动 Cassandra 连接或使用现有的连接。

URI	描述
cql:localhost/keyspace	单个主机、默认端口、测试通常
cql:host1,host2/keyspace	多主机，默认端口
cql:host1,host2:9042/keyspace	多主机、自定义端口
cql:host1,host2	默认端口和键空间
cql:bean:sessionRef	提供的会话参考
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。

15.5. MESSAGES

15.5.1. 传入消息

Camel Cassandra 端点需要一个简单的对象(Object 或 Object[] 或 Collection<Object>)，它将绑定到 CQL 语句作为查询参数。如果消息正文为空或为空，那么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 以普通字符串形式查询或使用 QueryBuilder 构建。

15.5.2. 传出消息

Camel Cassandra 端点根据 resultSetConversionStrategy 生成一个或多个 Cassandra Row 对象：

- list<Row > 如果 resultSetConversionStrategy 为 ALL 或 LIMIT_[0-9]+
- 如果 resultSetConversionStrategy 是 ONE，则单数为 Row'
- 否则，如果 resultSetConversionStrategy 是 ResultSetConversionStrategy的自定义实现

15.6. 软件仓库

Cassandra 可用于存储用于幂等和聚合 EIP 的消息。

Cassandra 可能尚未是排队用例的最佳工具，请阅读 Cassandra 反模式队列和队列，如数据集。建议使用这些表的 LeveledCompaction 和一个小的 GC grace 设置，以便快速删除 tombstoned 行。

15.7. 幂等软件仓库

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，并可扩展来使用不同的数据模型。

选项	默认	描述
table	CAMEL_IDEMPOTENT	表名称
pkColumns	NAME,' KEY'	主密钥列
name		软件仓库名称，用于 NAME 列的值
ttl		生存时间
writeConsistencyLevel		用于插入/删除密钥的一致性级别：ANY、ONE、TWO、QUORUM、LOCAL_QUORUM...
readConsistencyLevel		用于读取/检查键的一致性级别：ONE、TWO、QUORUM、LOCAL_QUORUM...

15.8. 聚合存储库

NamedCassandraAggregationRepository 将交换由关联密钥存储在 Cassandra 表中，如下所示：

CAMEL_AGGREGATION.cql

```
CREATE TABLE CAMEL_AGGREGATION (
  NAME varchar,        -- Repository name
  KEY varchar,         -- Correlation id
  EXCHANGE_ID varchar, -- Exchange id
  EXCHANGE blob,       -- Serialized exchange
  PRIMARY KEY (NAME, KEY)
) WITH compaction = {'class':'LeveledCompactionStrategy'}
  AND gc_grace_seconds = 86400;
```

或者，CassandraAggregationRepository 没有 NAME 列，并可扩展为使用不同的数据模型。

选项	默认	描述
table	CAMEL_AGGREGATION	表名称
pkColumns	名称,KEY	主密钥列
exchangeIdColumn	EXCHANGE_ID	Exchange Id 列
exchangeColumn	交换	交换内容列
name		软件仓库名称，用于 NAME 列的值
ttl		交换时间
writeConsistencyLevel		用于插入/删除交换的一致性级别：ANY、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...

15.9. 例子

要在表中插入一些内容，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，您应该能够使用列表作为正文插入数据

```
Arrays.asList("davsclaus", "Claus", "Ibsen")
```

相同的方法可用于更新或查询表。

15.10. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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 16 章 控制总线

仅支持生成者

EIP 模式的控制 总线 允许从框架内监控和管理集成系统。

使用控制总线来管理企业集成系统。控制总线使用与应用程序数据使用的相同消息传递机制，但使用单独的频道来传输与消息流中涉及的组件管理相关的数据。

在 Camel 中，您可以使用 JMX 管理和监控，或使用 CamelContext 中的 Java API 或 org.apache.camel.api.management 软件包，或使用此处具有示例的事件通知程序。

ControlBus 组件可根据控制总线 EIP 模式轻松管理 Camel 应用程序。例如，通过向端点发送消息来控制路由的生命周期，或收集性能统计信息。

```
controlbus:command[?options]
```

其中 command 可以是任何字符串，用于识别要使用的命令类型。

16.1. 命令

命令	描述
route	使用 routeId 和 action 参数控制路由。
语言	允许您指定用于评估消息正文的。如果评估有任何结果，则结果将放入消息正文中。

16.2. 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

16.2.1. 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。

16.2.2. 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。

16.3. 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

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

16.4. 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：

16.4.1. 路径参数(2 参数)

Name	描述	默认值	类型
command (producer)	**所需的命令** 可以是 route 或 language。 Enum 值： • route • 语言		字符串

Name	描述	默认值	类型
language (producer)	允许您指定用于评估消息正文的语言名称。如果评估有任何结果，则结果将放入消息正文中。 Enum 值： • bean • 常数 • el • exchangeProperty • file • groovy • header • jsonPath • MVEL • ognl • ref • simple • spel • sql • terser • 令牌化 • XPath • xquery • xtokenize		语言

16.4.1.1. 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)	要表示可以是 start、stop 或 status 的操作。要启动或停止路由，或者作为消息正文中的输出获取路由的状态。您可以使用挂起并从 Camel 2.11.1 中恢复来挂起或恢复路由。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整个 CamelContext 的性能统计。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，则任务的任何结果都不会在 Exchange 上设置。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，或者在处理任务期间发生任何异常。 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 millis 中的延迟。	1000	int
routeId (producer)	按其 id 指定路由：special 关键字 current 表示当前的路由。		字符串

16.5. 使用 ROUTE 命令

通过 route 命令，您可以在给定路由上执行常见的任务，例如启动路由，您可以向这个端点发送空消息：

```
template.sendBody("controlbus:route?routeId=foo&action=start", null);
```

要获取路由的状态，您可以执行以下操作：

```
String status = template.requestBody("controlbus:route?routeId=foo&action=status", null,
String.class);
```

16.6. 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的性能统计信息，或用于 CamelContext。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。您可以使用 ManagedRouteMBean 上的 dumpRouteStatsAsXml 操作从 JMX 获取的相同数据。

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);
```

16.7. 使用简单语言

您可以将 简单 语言与控制总线搭配使用，例如停止特定路由，您可以将消息发送到包含以下信息的 "controlbus:language:simple" 端点：

```
template.sendBody("controlbus:language:simple",
"${camelContext.getRouteController().stopRoute('myRoute')}");
```

因为这是一个 void 操作，因此不会返回任何结果。但是，如果要进行路由状态：

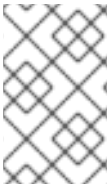
```
String status = template.requestBody("controlbus:language:simple",
"${camelContext.getRouteStatus('myRoute')}", String.class);
```

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，您可以执行以下操作：

```
template.sendBody("controlbus:language:simple?async=true", "${camelContext.stop()}");
```

我们使用 async=true 异步停止 Camel，否则我们将尝试在被发送到控制总线组件时尝试停止 Camel。



注意

您还可以使用其他语言，如 Groovy 等等。

16.8. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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
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 17 章 CRON

仅支持消费者

Cron 组件是一个通用接口组件，允许在使用 Unix cron 语法指定的特定间隔触发事件（例如： 0/2945 DS ？？），每两秒钟触发事件。

作为接口组件，Cron 组件不包含默认的实现，而是要求用户插入其选择的实施。

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，它可在 cron 表达式需要时使用 Kubernetes 调度程序触发路由。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。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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

可能需要其他库来插入特定的实现。

17.1. 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

17.1.1. 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。

17.1.2. 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。

17.2. 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)	提供了多个实现时要使用的 CamelCronService 的 id。		字符串

17.3. 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：

17.3.1. 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

17.3.2. 查询参数(4 参数)

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值

Name	描述	默认值	类型
schedule (consumer)	**必需的** A cron 表达式，用于生成事件。		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

17.4. 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+114 +? 也可以写为 0/3 10945?，并在每小时的 十分钟内每三个秒触发一次事件。

调度表达式中的部分表示（按顺序排列）：

- **秒（可选）**
- **minutes**
- **几小时**
- **月份的日期**

- 月
- 周天
- 年（可选）

计划表达式可由 5 个到 7 个部分组成。当表达式由 6 个部分组成时，第一个项是"秒"部分（和年份）。

调度表达式的其他有效示例如下：

- 0/2 * * * ?(5 部分，每两分钟事件一次)
- 0 0/2 200200 MON-FRI 2030 (7 部分，每两分钟，每 2 月 2030 日)

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>
```

17.5. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	提供了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 18 章 CXF

支持生成者和消费者

CXF 组件提供与 Apache CXF 集成，用于连接在 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），然后读取流缓存。

Maven 用户必须在其 pom.xml 中为这个组件添加以下依赖项：

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

18.1. URI 格式

此端点有两种 URI 格式： cxfEndpoint 和 someAddress。

```
cxf:bean:cxfEndpoint[?options]
```

其中 cxfEndpoint 代表引用 Spring bean registry 中的 bean 的 bean ID。使用这个 URI 格式，大多数端点详情都在 bean 定义中指定。

```
cxf://someAddress[?options]
```

其中 someAddress 指定 CXF 端点的地址。使用这个 URI 格式，大多数端点详情都会使用选项来指定。

对于以上任一风格，您可以在 URI 中附加选项，如下所示：

```
cxf:bean:cxfEndpoint?wsdlURL=wsdl/hello_world.wsdl&dataFormat=PAYLOAD
```

18.2. 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

18.2.1. 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。

18.2.2. 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。

18.3. 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 模式下运行 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 将标头过滤到或从 Camel 消息过滤。		HeaderFilterStrategy
useGlobalSslContextParameters (security)	启用使用全局 SSL 上下文参数。	false	布尔值

18.4. 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：

18.4.1. 路径参数(2 参数)

Name	描述	默认值	类型
beanId (common)	查找现有配置的 CxfEndpoint。必须使用 bean: 作为前缀。		字符串
地址 （服务）	服务发布地址。		字符串

18.4.2. 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 • 消息 • CXF_MESSAGE • POJO	POJO	dataformat
wrappedStyle (common)	描述 SOAP 正文中如何表示参数的 WSDL 风格。如果值为 false，则 CXF 将选择 Documentation-literal unwrapped 风格，如果值为 true，则 CXF 将选择文档粘贴风格。		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)	此选项将设置默认 operationName，它将由调用远程服务的 CxfProducer 使用。		字符串
defaultOperationNamespace (producer)	此选项将设置默认 operationNamespace，它将由调用远程服务的 CxfProducer 使用。		字符串
hostnameVerifier (producer)	要使用的主机名验证器。使用 sVirt 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 DNAT 表示法引用 SSL 上下文。		SSLContextParameters
wrap (producer)	CXF 端点制作者将调用的操作类型。	false	布尔值
Sync (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 模式下运行 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
总线 (advanced)	使用自定义配置的 CXF 总线。		总线

Name	描述	默认值	类型
continuationTimeout (advanced)	这个选项用于设置 CXF continuation 超时，当 CXF 服务器使用 Jetty 或 Servlet 传输时，默认可在 CxfConsumer 中使用。	30000	long
cxfBinding (advanced)	使用自定义 CxfBinding 来控制 Camel Message 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，它们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 端点自行创建总线时，将设置默认总线。	false	布尔值
HeaderFilterStrategy (advanced)	使用自定义 HeaderFilterStrategy 将标头过滤到或从 Camel 消息过滤。		HeaderFilterStrategy
mergeProtocolHeaders (advanced)	是否合并协议标头。如果启用，则 Camel 和 CXF 之间的标头会变得更加一致且类似。如需了解更多详细信息，请参阅 CAMEL-6393。	false	布尔值
mtomEnabled (advanced)	启用 MTOM (attachment)。这需要使用 POJO 或 PAYLOAD 数据格式模式。	false	布尔值
properties (advanced)	使用映射中的键/值对来设置额外的 CXF 选项。例如，要在 SOAP 错误中打开 stacktraces, properties.faultStack TraceEnabled=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)	要限制在启用日志记录功能时日志记录器输出的字节数，-1 代表没有限制。	49152	int
skipFaultLogging (logging)	这个选项控制 PhaseInterceptorChain 是否跳过它捕获的 Fault。	false	布尔值
密码 （安全）	此选项用于为 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）	此选项用于为 CXF 客户端设置用户名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)	此服务的端点名称，它映射到 wsdl:portname。以 ns:PORT_NAME 格式，其中 ns 是在这个范围内有效的命名空间前缀。		字符串
publishedEndpointUrl (service)	此选项可覆盖从 WSDL 发布的 endpointUrl，可通过服务地址 url 和 wsd 访问。		字符串
serviceClass (service)	SEI (Service Endpoint Interface)类的类名称，该类可以具有 JSR181 注解或不。		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 classpath、文件系统或远程托管。		字符串

serviceName 和 portName 是 QNames，因此如果您提供它们带有其 {namespace} 前缀，如上例所示。

18.4.3. 数据格式的描述

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 组件创建一个 CXF 端点，该端点可使用以下方法之一使用：

- **consumer - （在路由开始时）代表 Web 服务实例，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。**
- **producer - （路由中的其他点）代表 WS 客户端代理，它将当前的交换对象转换为远程 Web 服务上的操作调用。当前交换的格式必须与端点的 dataFormat 设置匹配。**

dataformat	描述

dataformat	描述
POJO	POJO （旧 Java 对象）是目标服务器上调用的方法的 Java 参数。支持协议和逻辑 JAX-WS 处理程序。
PAYLOAD	**PAYLOAD** 是应用 CXF 端点中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。如果您使用这类 DataFormat，则无法触动或更改流，因此您无法在 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** 允许通过将消息从传输层转换为原始 SOAP 消息来调用 CXF 拦截器的完整功能

您可以通过检索交换属性 CamelCXFDataFormat 来确定交换的数据格式模式。Exchange key 常量在 org.apache.camel.component.cxf.common.message.CxfConstants.DATA_FORMAT_PROPERTY 中定义。

18.4.4. 如何在 RAW 模式中启用 CXF 的 LoggingOutInterceptor

CXF 的 LoggingOutInterceptor 输出在有线到日志记录系统(Java Util Logging)上的出站消息。由于 LoggingOutInterceptor 处于 PRE_STREAM 阶段（但 PRE_STREAM 阶段以 RAW 模式删除），所以您必须将 LoggingOutInterceptor 配置为在 WRITE 阶段运行。以下是以下示例。

```
@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;
}
```

18.4.5. 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 标头是明确定义为端点的 WSDL 绑定合同（如 SOAP 标头）的标头。

带外 标头是通过线序列化的标头，但不明确是 WSDL 绑定合同的一部分。

标头中继/过滤是双向的。

当路由具有 CXF 端点并且开发人员需要具有在线标头（如 SOAP 标头）时，会在由另一个 JAXWS 端点使用的路由间进行转发，然后将 relayHeaders 设置为 true，这是默认值。

18.4.6. 仅适用于 POJO 模式

relayHeaders=true 表示转发标头的意图。对给定标头是否转发的实际决定将委派给实施 MessageHeadersRelay 接口的可插拔实例。将参考 MessageHeadersRelay 的共识实施，以确定是否需要转发标头。已实施 SoapMessageHeadersRelay，它将自身绑定到已知的 SOAP 名字空间。目前，只过滤带外标头，并在 relayHeaders=true 时始终转发带外标头。如果线上有一个标头，其名称空间对于运行时未知，则使用回退 DefaultMessageHeadersRelay，它只允许转发所有标头。

relayHeaders=false 设置指定所有标头带外和带外设置都应丢弃。

您可以插入您自己的 MessageHeadersRelay 实现，或向中继列表添加额外的 MessageHeadersRelay。要覆盖预加载的中继实例，请确保您的 MessageHeadersRelay 实现服务与您希望覆盖的名称相同。另请注意，覆盖中继必须服务所有要覆盖的名称空间，否则路由启动时的运行时异常将被抛出，因为这会给转发实例映射引入名字空间中的不确定性。

```
<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"/>
```

查看显示如何转发/过滤标头的测试：

CxfMessageHeadersRelayTest

- 支持 POJO 和 PAYLOAD 模式。在 POJO 模式中，只有带外消息标头才可以进行过滤，因为 cxf 从标头列表中移除。in-band 标头合并到 POJO 模式的 MessageContentList 中。camel-cxf 组件会使任何尝试从 MessageContentList 中删除 in-band 标头。如果需要过滤带头标头，请在 CXF 端点中使用 PAYLOAD 模式或插入(pretty 拦截器) CXF 拦截器/DemoWS Handler。

- Message Header Relay 机制已合并到 CxfHeaderFilterStrategy 中。relayHeaders 选项、其语义和默认值保持不变，但它是 CxfHeaderFilterStrategy 的属性。以下是配置它的示例。

```
@Bean
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.cl
ass);

cxfEndpoint.setAddress("/CxfMessageHeadersRelayTest/HeaderService/routerNoRelayEndpoi
nt");
    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 接口稍有变化，并被重命名为 MessageHeaderFilter。它是 CxfHeaderFilterStrategy 的属性。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
relayAllMessage Headers	否	所有消息标头都将传播（不由 Message Header Filters 处理） *类型*:**boolean***Default*:**false**

Name	必填	描述
allowFilterNamespaceClash	否	如果激活命名空间中的两个过滤器重叠，则属性控制如何处理它。如果值为 **true**，则最后一个 wins。如果值为 **false**，它将抛出一个 exception *Type*:**boolean** *Default*:**false**

18.5. 使用 SPRING 配置 CXF 端点

您可以使用下面显示的 Spring 配置文件配置 CXF 端点，您还可以将端点嵌入到 camelContext 标签中。在调用服务端点时，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 Bean 元素上指定的 JAX-WS 模式Location 属性。这允许 CXF 验证文件，且是必需的。另请注意 < cxf:cxfEndpoint/& gt; 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当前不支持组合 {namespace}localName 语法。

cxf:cxfEndpoint 元素支持许多附加属性：

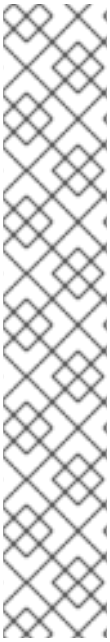
Name	值
portName	此服务的端点名称，它映射到 **wsdl:port@name**。以 **ns:PORT_NAME** 格式，其中 **ns** 是在这个范围内有效的命名空间前缀。
serviceName	此服务正在实现的服务名称，它映射到 **wsdl:service@name**。以 **ns:SERVICE_NAME** 格式，其中 **ns** 是在这个范围内有效的命名空间前缀。
wsdlURL	WSDL 的位置。可以位于 classpath、文件系统或远程托管。
bindingId	要使用的服务模型的 **bindingId**。
address	服务发布地址。
总线	JAX-WS 端点中使用的总线名称。
serviceClass	SEI (Service Endpoint Interface)类的类名称，该类可以具有 JSR181 注解或不。

它还支持多个子元素：

Name	值
cxf:inInterceptors	此端点的传入拦截器。< **bean> 或 &lt**; **ref>** 列表。
cxf:inFaultInterceptors	此端点的传入错误拦截器。< **bean> 或 &lt**; **ref>** 列表。
cxf:outInterceptors	此端点传出的拦截器。< **bean> 或 &lt**; **ref>** 列表。
cxf:outFaultInterceptors	此端点传出的故障拦截器。< **bean> 或 &lt**; **ref>** 列表。
cxf:properties	应该向 JAX-WS 端点提供的属性映射。请参阅以下。
cxf:handlers	应该向 JAX-WS 端点提供 JAX-WS 处理程序列表。请参阅以下。
cxf:dataBinding	您可以指定在端点中使用的 **DataBinding**。这可以通过 Spring < **bean class="MyDataBinding"/>** 语法来提供。
cxf:binding	您可以为这个端点指定 **BindingFactory**。这可以通过 Spring < **bean class="MyBindingFactory"/>** 语法来提供。
cxf:features	保存此端点拦截器的功能。Bean 或 refs 列表

Name	值
cxf:schemaLocations	要使用的端点的模式位置。schemaLocations 列表
cxf:serviceFactory	此端点要使用的服务工厂。这可以通过 Spring < **bean class="MyServiceFactory"/>** 语法提供

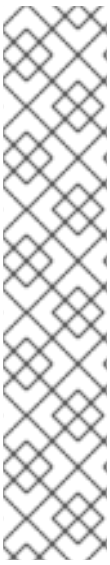
您可以找到更高级的示例，其中显示了如何在 CXF JAX-WS Configuration 页面上 提供拦截器、属性和处理程序。



注意

您可以使用 cxf:properties 设置 camel-cxf 端点的 dataFormat，并从 spring 配置文件中设置DefaultBus 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，使用配置了 Bean 的 Java 代码（如其他示例所示）是在 SpringBoot 中最佳实践。

18.6. 如何使 CAMEL-CXF 组件使用 LOG4J 而不是 JAVA.UTIL.LOGGING

CXF 的默认日志记录器为 java.util.logging。如果要将它更改为 log4j，请按如下所示操作：在

classpath 中创建一个名为 META-INF/cxf/org.apache.cxf.logger 的文件。此文件应包含类 org.apache.cxf.common.logging.Log4jLogger 的完全限定名称，且不含注释。

18.7. 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，则会出现这类错误，因为 CXF 没有添加 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，如以下 WriteXmlDeclarationInterceptor 所述：

```
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);
    }

}
```

另外，您还可以为它添加一个消息标头，如 CxfConsumerTest 所示：

```
// 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);
```

18.8. 如何从消息标头中覆盖 CXF PRODUCER 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()));
```

18.9. 如何以 POJO 数据格式使用 CAMEL-CXF 端点的消息

camel-cxf 端点消费者 POJO 数据格式基于 CXF 调用器，因此消息标头具有名为 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 });
    }

}
```

18.10. 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，然后将方法参数添加到列表中，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。

如果要从消息正文中获取对象数组，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");
```

18.11. 如何以 PAYLOAD 数据格式处理 CAMEL-CXF 端点的消息

PAYLOAD 意味着您将 SOAP envelope 中的有效负载作为原生 CxfPayload 处理。message .getBody （） 将返回 org.apache.camel.component.cxf.CxfPayload 对象，其中包含 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);
            }
         });
      }
   };
}
```

18.12. 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，数据格式是一个"Java 对象列表"。虽然 Camel 在此模式中将消息正文公开为 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 in-band SOAP 标头，因此只有带外 SOAP 标头可用于 POJO 模式的 camel-cxf。

以下示例演示了如何 get/set SOAP 标头。假设我们有一个从 Camel-cxf 端点转发到另一个 Camel-cxf 端点的路由。也就是说，SOAP Client → Camel → CXF 服务。在响应回 SOAP 客户端之前，我们可以将两个处理器附加到(1)来获取/插入 SOAP 标头，然后再向 CXF 服务获取/插入 SOAP 标头。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 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 Message 标头。Camel 消息标头名称为 "org.apache.cxf.headers.Header.list"，它在 CXF (org.apache.cxf.headers.Header.HEADER_LIST) 中定义。标头值是一个 CXF SoapHeader 对象列表(org.apache.cxf.binding.soap.SoapHeader)。以下片段是 InsertResponseOutHeaderProcessor （它会在响应消息中插入新的 SOAP 标头）。访问 InsertResponseOutHeaderProcessor 和 InsertRequestOutHeaderProcessor 中的 SOAP 标头的方式实际上相同。两个处理器之间的唯一区别是设置插入 SOAP 标头的方向。

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);

    }

}
```

18.13. 如何在 PAYLOAD 模式中获取和设置 SOAP 标头

我们已展示了如何在 PAYLOAD 模式中以 PAYLOAD 模式访问 SOAP 消息作为 CxfPayload 对象，该部分 是 How to handle the camel-cxf endpoint in PAYLOAD data format。

获取 CxfPayload 对象后，您可以调用 CxfPayload.getHeaders （）方法，该方法返回 DOM Elements (SOAP 标头)列表。

例如，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());
```

您还可以使用与子选择 " How to get and set SOAP headers in POJO 模式"中所述的方法一样设置或获取 SOAP 标头。因此，您可以使用标头 "org.apache.cxf.headers.Header.list" 来获取和设置 SOAP 标头列表。这也意味着，如果您有一个从一个 Camel-cxf 端点转发到另一个 Camel-cxf 端点的路由 (SOAP Client → Camel → CXF 服务)，现在也会将 SOAP 客户端发送的 SOAP 标头转发到 CXF 服务。如果您不想转发这些标头，则必须在 Camel 标头 "org.apache.cxf.headers.Header.list" 中删除它们。

18.14. 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。

18.15. 如何从 CAMEL 中抛出 SOAP FAULT

如果您使用 camel-cxf 端点来消耗 SOAP 请求，您可能需要从 camel 上下文抛出 SOAP Fault。基本上，您可以使用 throwFault DSL 来执行此操作；它适用于 POJO、PAYLOAD 和 MESSAGE 数据格式。
您可以定义 soap 错误，如 CxfCustomizedExceptionTest 所示：

```
SOAP_FAULT = new SoapFault(EXCEPTION_MESSAGE, SoapFault.FAULT_CODE_CLIENT);
Element detail = SOAP_FAULT.getOrCreateDetail();
Document doc = detail.getOwnerDocument();
Text tn = doc.createTextNode(DETAIL_TEXT);
detail.appendChild(tn);
```

然后，按照您需要抛出它

```
from(routerEndpointURI).setFaultBody(constant(SOAP_FAULT));
```

如果您的 CXF 端点以 MESSAGE 数据格式工作，您可以在消息正文中设置 SOAP Fault 消息，并在消息标头中设置响应代码，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));
   }

});
```

与使用 POJO 数据格式相同。您可以在外部正文上设置 SOAPFault。

18.16. 如何传播 CAMEL-CXF 端点的请求和响应上下文

CXF 客户端 API 提供了一种通过请求和响应上下文调用操作的方法。如果您使用 camel-cxf 端点制作者来调用外部 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());
```

18.17. 附加支持

POJO 模式： 支持带有 Attachment 和 MTOM 的 SOAP （请参阅启用 MTOM 的 Payload 模式）。但是，没有测试带有 Attachment 的 SOAP。由于附件被分为 marshalled 和 unmarshalled into POJO，因此用户通常不需要处理他们自己的附件。如果没有启用 MTOM，则会将附件传播到 Camel 消息的附加。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。附加可通过上述 Camel Message API 检索。支持具有附加(SwA)的 SOAP，并且可以检索附件。SwA 是默认值（与将 CXF 端点属性 "mtom-enabled" 设置为 false）。

要启用 MTOM，请将 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 attachment 的 Camel 消息，以便在 Payload 模式中发送到 CXF 端点。

```
Exchange exchange = context.createProducerTemplate().send("direct:testEndpoint", new
Processor() {

   public void process(Exchange exchange) throws Exception {
      exchange.setPattern(ExchangePattern.InOut);
      List<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());
```

您还可以以 Payload 模式使用从 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：不支持过期，因为它根本不处理消息。

CXF_RAW Mode ：支持 MTOM，并且前面提到的 Camel Message API 可以检索过期。请注意，当收到多部分（即 MTOM）消息时，默认的 SOAPMessage 到 String converter 将提供正文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。

18.18. PAYLOAD 模式中的流支持

camel-cxf 组件现在在使用 PAYLOAD 模式时支持流传输传入的消息。在以前的版本中，传入的信息会被完全解析。对于大型消息，这会消耗时间，并使用大量内存。传入消息可以在路由时保留为 javax.xml.transform.Source，如果没有修改有效负载，则可以直接流传输到目标目的地。对于常见的"简单代理"用例（例如： from ("cxf:...").to ("cxf:...")），这可提供显著的性能增加，并显著降低的内存要求。

然而，在有些情况下，流可能不合适或需要。由于流性质，在处理链中稍后才会发现无效的传入的 XML。此外，某些操作可能需要消息成为 DOM 解析任何方法（如 WS-Security 或消息追踪等），在这种情况下，流的优点有限。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 streaming on/off。
- 组件属性： CxfComponent 对象也有一个 allowStreaming 属性，可为从该组件创建的端点设置默认。

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，但设置上述 endpoint 属性将覆盖该端点的值。

18.19. 使用通用 CXF DISPATCH 模式

camel-cxf 组件支持通用 CXF 分配模式，该模式可以传输任意结构的消息（例如，不绑定到特定的 XML 模式）。要使用此模式，只需省略指定 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 key SOAPAction 在 Camel 标头中提供它（不区分大小写）。

18.20. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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 模式下运行 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 将标头过滤到或从 Camel 消息过滤。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

Name	描述	默认值	类型
camel.component.cxf.use-global-ssl-context-parameters	启用使用全局 SSL 上下文参数。	false	布尔值
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 将标头过滤到或从 Camel 消息过滤。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 19 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。

19.1. URI 格式

```
dataformat:name:(marshal|unmarshal)[?options]
```

其中 name 是数据格式的名称。然后后根一个操作，需要是 marshal 或 unmarshal。这些选项用于配置正在使用的 数据格式。有关它支持哪些选项，请参阅数据格式文档。

19.2. 数据格式选项

19.2.1. 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

19.2.1.1. 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。

19.2.1.2. 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。

19.3. 组件选项

数据格式组件支持 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

19.4. 端点选项

数据格式端点使用 URI 语法进行配置：

```
dataformat:name:operation
```

使用以下路径和查询参数：

19.4.1. 路径参数(2 参数)

Name	描述	默认值	类型
name (producer)	数据格式 **必需** 名称。		字符串
operation (producer)	**需要使用** marshal 或 unmarshal 的操作。 Enum 值： • marshal • unmarshal		字符串

19.4.2. 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

19.5. 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，您要进行：

```
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="activemq:My.Queue"/>
    <to uri="dataformat:jaxb:unmarshal?contextPath=com.acme.model"/>
```

```
    <to uri="mqseries:Another.Queue"/>
  </route>
</camelContext>
```

19.6. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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 20 章 DATASET

支持生成者和消费者

对分布式和异步处理的测试非常困难。Mock、Test 和 DataSet 端点与 Camel 测试框架协同工作，从而通过使用 企业集成模式 和 Camel 的大量组件以及强大的 Bean 集成来简化您的单元和集成测试。

DataSet 组件提供了一种机制，可轻松对系统执行负载和 soak 测试。它的工作原理是允许您创建 DataSet 实例作为 消息源，并作为接收数据集的方法。

Camel 将在发送数据集 时使用吞吐量日志记录器。

20.1. URI 格式

```
dataset:name[?options]
```

其中 name 用于在 Registry 中查找 DataSet 实例

Camel 附带 org.apache.camel.component.dataset.DataSet 的支持实施，org.apache.camel.component.dataset.DataSetSupport 类，可用作实施您自己的 DataSet 的基础。Camel 还附带一些实现，可用于测试：org.apache.camel.component.dataset.SimpleDataSet、org.apache.camel.component.dataset.ListDataSet 和 org.apache.camel.component.dataset.FileDataSet，它们扩展 DataSetSupport。

20.2. 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

20.2.1. 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。

20.2.2. 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。

20.3. 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

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)	在模拟收到传入的消息时打开日志记录。这将仅记录传入消息的 INFO 级别一次。如需更详细的日志记录，请将 org.apache.camel.component.mock.MockEndpoint 类的日志记录器设置为 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，则默认为 DefaultExchangeFormatter。		ExchangeFormatter

20.4. 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：

20.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
name (common)	要在 registry 中查找 DataSet 所需的 名称。		DataSet

20.4.2. 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 = 标头不会设置 - strict/lenient = 标头将被设置为 For Producers: - off = 标头值不会被验证，如果没有包括标头值，则不会设置它。如果标头值不存在，标头值必须存在，并将验证 = lenient = 标头值（如果不存在）。 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)	在开始发送消息前，millis 中的时间段。	1000	long
minRate (consumer)	等待 DataSet 至少包含这个数量的消息。	0	int
preloadSize (consumer)	设置在路由完成初始化前应预加载(sent)的消息数量。	0	long
produceDelay (consumer)	可以指定延迟，这会在消费者发送消息时造成延迟（模拟速度较慢的处理）。	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
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
assertPeriod (producer)	设置一个宽限期，模拟端点将重新分配，以确保初始断言仍然有效。例如，这用于精确有多个消息到达的声明。例如，如果 expectedMessageCount (int)设为 5，则在 5 个或更多消息到达时满足断言。为确保完全 5 个消息到达，您需要等待少量周期以确保没有进一步的消息到达。这是您可以使用此方法。默认情况下禁用这个周期。		long
consumeDelay (producer)	可以指定延迟，在生成者消耗消息时（模拟速度较慢）时会导致延迟。	0	long
expectedCount (producer)	指定此端点应接收的消息交换数量。beware：如果要期望 0 个消息，然后在测试启动时进行额外的操作，因为测试启动时需要设置一个断言周期，以便测试可以在一段时间内运行，以确保仍没有到达的消息；对于使用 setAssertPeriod (long)，您需要设置一个指定时间。另一种方法是使用 NotifyBuilder，并使用 notifier 知道 Camel 在对模拟调用 assertIsSatisfied （）方法之前，在路由某些消息前知道何时进行路由。这可让您不使用固定的断言周期来加快测试时间。如果您想成为完全 n 个消息到达这个模拟端点，请参阅 setAssertPeriod (long)方法以了解更多详细信息。	-1	int
failFast (producer)	设置是否 assertIsSatisfied （）是否应该在第一次检测到的失败时快速失败，同时可能会等待所有预期消息到达，然后再执行预期验证。默认为 true。设置为 false，以使用与 Camel 2.x 中一样的行为。	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)	在模拟收到传入的消息时打开日志记录。这将仅记录传入消息的 INFO 级别一次。如需更详细的日志记录，请将 org.apache.camel.component.mock.MockEndpoint 类的日志记录器设置为 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 assertIsSatisfied （）的最短预期时间（以 millis 为单位）。		long

Name	描述	默认值	类型
resultWaitTime (producer)	设置 assertIsSatisfied （）在达到前等待的最大时间（以 millis 为单位）。		long
retainFirst (producer)	指定只保留前 n 个接收的交换数。这在使用大量数据进行测试时，通过不存储每个交换端点接收的每个交换的副本来减少内存消耗。重要：在使用这个限制时，getReceivedCounter （）仍会返回实际收到的交换数量。例如，如果我们收到 5000 Exchanges，并且配置为只保留前 10 个交换，则 getReceivedCounter （）仍会返回 5000，但 getExchanges （）中只有前 10 个交换，并且 getReceivedExchanges （）方法。使用此方法时，不支持一些其他预期方法，例如 expectedBodiesReceived (Object...)在收到的第一个正文数量上设置预期。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方法。	-1	int
retainLast (producer)	指定只保留最后 n 个接收的交换数。这在使用大量数据进行测试时，通过不存储每个交换端点接收的每个交换的副本来减少内存消耗。重要：在使用这个限制时，getReceivedCounter （）仍会返回实际收到的交换数量。例如，如果我们收到 5000 Exchanges，并且配置为只保留最后 20 个交换，则 getReceivedCounter （）仍会返回 5000，但 getExchanges （）中只有最后 20 个交换，并且 getReceivedExchanges （）方法。使用此方法时，不支持一些其他预期方法，例如 expectedBodiesReceived (Object...)在收到的第一个正文数量上设置预期。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方法。	-1	int
sleepForEmptyTest (producer)	当 expectedMessageCount (int)调用零时，允许指定 sleep 来等待此端点确实为空。		long
copyOnExchange (producer (advanced))	设置是否在这个模拟端点收到传入交换的深度副本。默认为 true。	true	布尔值

20.5. 配置 DATASET

Camel 将在 Registry 中查找用于实施 DataSet 接口的 bean。以便您可以将自己的 DataSet 注册为：

```
<bean id="myDataSet" class="com.mycompany.MyDataSet">
  <property name="size" value="100"/>
</bean>
```

20.6. 示例

例如，要测试将一组消息发送到队列，然后从队列中消耗，而不丢失任何消息：

```
// send the dataset to a queue
from("dataset:foo").to("activemq:SomeQueue");

// now lets test that the messages are consumed correctly
from("activemq:SomeQueue").to("dataset:foo");
```

以上会在 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，例如使用 SimpleDataSet，配置数据集的大事项，以及消息样子等。

20.7. DATASETSUPPORT (ABSTRACT 类)

DataSetSupport 抽象类是新 DataSets 的过期起点，为派生类提供一些有用的功能。

20.7.1. DataSetSupport 的属性

属性	类型	默认值	描述
defaultHeaders	**Map<String,Object>**	**null**	指定默认消息正文。对于 SimpleDataSet，它是一个恒定的有效负载；但是，如果要为每个消息创建自定义有效负载，请创建自己的 **DataSetSupport**。
outputTransformer	**org.apache.camel.Processor**	null	
size	**long**	**10**	指定要发送/恢复的消息数量。
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 **大小为** / 5，如果 **大小为** 0，否则设为 **reportCount** 值。

20.8. SIMPLEDATASET

SimpleDataSet 扩展 DataSetSupport，并添加默认正文。

20.8.1. SimpleDataSet 的额外属性

属性	类型	默认值	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恒定有效负载。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。

20.9. LISTDATASET

List'DataSet' 扩展 DataSetSupport，并添加默认正文列表。

20.9.1. ListDataSet 的额外属性

属性	类型	默认值	描述
defaultBodies	**list<Object>**	**empty LinkedList<Object>**	指定默认消息正文。默认情况下，**ListDataSet** 使用 **CamelDataSetIndex** 从 **defaultBodies** 列表中选择一个恒定有效负载。如果要自定义有效负载，创建一个 Camel **Processor**，并通过设置 **outputTransformer** 属性将 **ListDataSet** 配置为使用它。
size	**long**	defaultBodies 列表的大小	指定要发送/恢复的消息数量。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，则会使用 **CamelDataSetIndex** 和 **defaultBodies.size ()** 的 modulus 选择交换的有效负载

20.10. FILEDATASET

FileDataSet 扩展 ListDataSet，并添加从文件中加载正文的支持。

20.10.1. FileDataSet 中的其他属性

属性	类型	默认值	描述
sourceFile	**File**	null	指定有效负载的源文件
delimiter	**字符串**	\z	指定 **java.util.Scanner** 用来将文件分成多个有效负载的分隔符模式。

20.11. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

组件支持 11 个选项，如下所列。

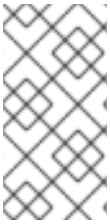
Name	描述	默认值	类型
camel.component.dataset-test.autowired-enabled	是否启用自动关闭。这用于自动关闭选项（选项必须标记为 autowired），方法是在 registry 中查找查找是否有单个匹配类型实例，然后在组件上配置。这可以用于自动配置 JDBC 数据源、JMS 连接工厂、AWS 客户端等。	true	布尔值
camel.component.dataset-test.enabled	是否启用 dataset-test 组件的自动配置。这默认是启用的。		布尔值
camel.component.dataset-test.exchange-formatter	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，则默认为 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

Name	描述	默认值	类型
camel.component.dataset-test.log	在模拟收到传入的消息时打开日志记录。这将仅记录传入消息的 INFO 级别一次。如需更详细的日志记录，请将 org.apache.camel.component.mock.MockEndpoint 类的日志记录器设置为 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
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，则默认为 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	在模拟收到传入的消息时打开日志记录。这将仅记录传入消息的 INFO 级别一次。如需更详细的日志记录，请将 org.apache.camel.component.mock.MockEndpoint 类的日志记录器设置为 DEBUG 级别。	false	布尔值

第 21 章 DIRECT

支持生成者和消费者

当制作者发送消息交换时，Direct 组件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 组件在发送消息交换时提供任何消费者的异步调用。

21.1. URI 格式

```
direct:someName[?options]
```

其中 someName 可以是唯一标识端点的任意字符串

21.2. 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

21.2.1. 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。

21.2.2. 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。

21.3. 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
block (producer)	如果向没有活跃消费者的直接端点发送消息，则我们可以告知制作者阻止并等待消费者变为 active。	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
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

21.4. 端点选项

直接端点使用 URI 语法进行配置：

```
direct:name
```

使用以下路径和查询参数：

21.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
name (common)	直接端点 **必需** 的名称。		字符串

21.4.2. 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
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
block (producer)	如果向没有活跃消费者的直接端点发送消息，则我们可以告知制作者阻止并等待消费者变为 active。	true	布尔值
failIfNoConsumers (producer)	当发送到没有活跃消费者的 DIRECT 端点时，生成者是否应该失败。	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
Sync (advanced)	是否强制进行同步处理。如果启用，则制作者线程将强制等待消息完成，直到同一线程继续处理。如果禁用（默认），则制作者线程可能会释放，并在消息继续由其他线程（活跃）处理时执行其他工作。	false	布尔值

21.5. SAMPLES

在以下路由中，我们使用直接组件将两个路由连接在一起：

```
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");
```

和使用 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，它们如何一起使用。

21.6. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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，则我们可以告知制作者阻止并等待消费者变为 active。	true	布尔值
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

Name	描述	默认值	类型
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 22 章 ELASTICSEARCH

从 Camel 3.18.3 开始

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 进行接口。

将以下依赖项添加到此组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-elasticsearch</artifactId>
  <version>3.20.1.redhat-00050</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

22.1. URI 格式

```
elasticsearch://clusterName[?options]
```

22.2. 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

22.2.1. 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。

22.2.2. 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。

22.3. 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(ms)。	30000	int
hostAddresses (producer)	以逗号分隔的带有要使用的 ip:port 格式的远程传输地址的列表。ip 和 port 选项必须留空，才能考虑 hostAddresses。		字符串
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 ms 的时间。	30000	int

Name	描述	默认值	类型
socketTimeout (producer)	套接字超时前要等待的超时时间(ms)。	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** 使用现有配置的 Elasticsearch 客户端，而不是为每个端点创建客户端。这允许使用特定设置自定义客户端。		RestClient
enableSniffer (advanced)	启用从正在运行的 Elasticsearch 集群中自动发现节点。如果将这个选项与 Spring Boot 结合使用，则由 Spring Boot 配置管理（请参阅：Spring Boot 中禁用 Sniffer）。	false	布尔值
sniffAfterFailureDelay (advanced)	失败后（以毫秒为单位）调度 sniff 执行的延迟。	60000	int
snifferInterval (advanced)	以毫秒为单位连续执行间隔。当禁用 sniffOnFailure 时，或连续的 sniffOnFailure 之间没有故障时，将休眠。	300000	int
certificatePath (security)	用于访问 Elasticsearch 的自签名证书的路径。		字符串
启用SSL （安全）	启用 SSL。	false	布尔值
密码 （安全）	进行身份验证的密码。		字符串
用户 （安全）	基本身份验证用户。		字符串

22.4. 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：

22.4.1. 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

22.4.2. 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(ms)。	30000	int
disconnect (producer)	在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)	以逗号分隔的带有要使用的 ip:port 格式的远程传输地址的列表。		字符串
indexName (producer)	要操作的索引的名称。		字符串
maxRetryTimeout (producer)	重试前 ms 的时间。	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跃状态的时间(ms)。	60000	int
size (producer)	响应的大小。		整数
socketTimeout (producer)	套接字超时前要等待的超时时间(ms)。	30000	int
useScroll (producer)	启用滚动使用。	false	布尔值
waitForActiveShards (producer)	索引创建会等待写入一致性的分片数量可用。	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)	反序列化文档时要使用的类。	Object Node	类

Name	描述	默认值	类型
enableSniffer (advanced)	启用从正在运行的 Elasticsearch 集群中自动发现节点。如果将这个选项与 Spring Boot 结合使用，则由 Spring Boot 配置管理（请参阅：Spring Boot 中禁用 Sniffer）。	false	布尔值
sniffAfterFailureDelay (advanced)	失败后（以毫秒为单位）调度 sniff 执行的延迟。	60000	int
snifferInterval (advanced)	以毫秒为单位连续执行间隔。当禁用 sniffOnFailure 时，或连续的 sniffOnFailure 之间没有故障时，将休眠。	300000	int
certificatePath (security)	用于访问 Elasticsearch 的自签名证书的路径。		字符串
启用SSL （安全）	启用 SSL。	false	布尔值

22.5. 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) 常数： PARAM_DOCUMENT_CLASS	文档类的全限定名称到 unmarshall。	Object Node	类
waitForActiveShards (producer) 常量： PARAM_WAIT_FOR_ACTIVE_SHARDS	索引创建会等待写入一致性的分片数量可用。		整数
scrollKeepAliveMs (producer) 常量： PARAM_SCROLL_KEEP_ALIVE_MS	响应的起始索引。		整数
useScroll (producer) 常数： PARAM_SCROLL	设置为 true 以启用滚动使用。		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

22.6. 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** or **IndexRequest.Builder** content to index	将内容添加到索引中，并在正文中返回内容的 indexId。您可以使用键 "indexName" 设置消息标头来设置目标索引的名称。您可以使用 "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，并在正文中返回一个 GetResponse 对象。您可以使用键 "indexName" 设置消息标头来设置目标索引的名称。您可以使用 "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，并在正文中返回一个 Result 对象。您可以使用键 "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。您可以使用键 "indexName" 设置消息标头来设置目标索引的名称。
批量	已接受的任何类型的 Serable 或 **BulkRequest.Builder** (DeleteOperation.Builder 用于删除操作，UpdateOperation.Builder 用于 update 操作，CreateOperation.Builder 用于 create operation, byte[], InputStream, String, Reader, Map 或 index 操作的任何文档类型)	将/更新/删除内容从/添加到索引中，并在正文中返回一个 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。

operation	消息正文	description
搜索	**map**、string 或 **SearchRequest.Builder**	使用查询字符串映射搜索内容。您可以使用键 "indexName" 设置消息标头来设置目标索引的名称。您可以通过将消息标头设置为"size"来设置要返回的点击数。您可以通过将消息标头设置为键"from"来设置起始文档偏移。
MultiSearch	**MsearchRequest.Builder**	一个搜索
MultiGet	**iterable<String>** 或 **MgetRequest.Builder** 要检索的文档 ID	多次获取 您可以使用键 "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，并在正文中返回布尔值标志。 您必须使用键 "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** 或要更新的任何文档类型内容	将内容更新为索引，并在正文中返回内容的 indexId。您可以使用键 "indexName" 设置消息标头来设置目标索引的名称。您可以使用 "indexId" 设置消息标头来设置 indexId。
ping	None	对 Elasticsearch 集群进行 ping 操作并返回 true （如果 ping 成功），否则返回 false

22.7. 配置组件并启用基本身份验证

要使用 Elasticsearch 组件，必须使用最低配置进行配置。

```
ElasticsearchComponent elasticsearchComponent = new ElasticsearchComponent();
elasticsearchComponent.setHostAddresses("myelkhost:9200");
camelContext.addComponent("elasticsearch", elasticsearchComponent);
```

对于使用 elasticsearch 的基本身份验证或在 elasticsearch 集群前面使用反向 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);
```

22.8. 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，您需要指定一个 indexId 标头。

客户端只需要将包含映射的正文消息传递给路由。结果正文包含创建的 indexId。

```
Map<String, String> map = new HashMap<String, String>();
map.put("content", "test");
String indexId = template.requestBody("direct:index", map, String.class);
```

22.9. 搜索示例

搜索特定字段和值使用 Operation 'Search'。传递查询 JSON 字符串或映射

```
from("direct:search")
  .to("elasticsearch://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 在特定字段中搜索。

```
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 搜索 api 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
}
```

还可以使用。

```
from("direct:search")
  .to("elasticsearch://elasticsearch?
operation=Search&indexName=twitter&useScroll=true&scrollKeepAliveMs=30000")
  .split()
  .body()
```

```
    .streaming()
    .to("mock:output")
    .end();
```

22.10. 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>
```

特定字段上的 MultiSearch

```
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);
```

22.11. 文档类型

对于所有搜索操作，可以指定检索的文档类型，以便结果已经与预期类型一起取消合并。

可使用标头 "documentClass" 或相同名称的 uri 参数来设置文档类型。

22.12. 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 提供了 jakarta.json-api:1.1.6，Elasticsearch 需要使用 json-api v2。

22.12.1. 使用 Spring Boot 提供的 RestClient

默认情况下，Spring Boot 将自动配置 camel 将使用的 Elasticsearch RestClient，它可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。

22.12.2. 在使用 Spring Boot 时禁用 Sniffer

当 Spring Boot 位于 classpath 时，Elasticsearch 的 Sniffer 客户端会被默认启用。这个选项可以在 Spring Boot Configuration 中禁用：

```
spring:
  autoconfigure:
    exclude:
org.springframework.boot.autoconfigure.elasticsearch.ElasticsearchRestClientAutoConfiguration
```

22.13. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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
camel.component.elasticsearch.client	要使用现有配置的 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。		RestClient
camel.component.elasticsearch.connection-timeout	连接超时前等待的时间(ms)。	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 集群中自动发现节点。如果将这个选项与 Spring Boot 结合使用，则由 Spring Boot 配置管理（请参阅：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	以逗号分隔的带有要使用的 ip:port 格式的远程传输地址的列表。ip 和 port 选项必须留空，才能考虑 hostAddresses。		字符串
camel.component.elasticsearch.lazy-start-producer	生成者是否应懒惰启动 （在第一个消息中）。通过懒惰启动，您可以使用此选项来允许 CamelContext 和路由在生成者启动期间启动，并导致路由启动失败。通过懒惰启动，启动失败可以在路由信息时通过 Camel 的路由错误处理程序进行处理。请注意，在处理第一个消息时，创建并启动生成者可能需要稍等时间，并延长处理的总处理时间。	false	布尔值
camel.component.elasticsearch.max-retry-timeout	重试前 ms 的时间。	30000	整数

Name	描述	默认值	类型
camel.component.elasticsearch.password	进行身份验证的密码。		字符串
camel.component.elasticsearch.sniff-after-failure-delay	失败后（以毫秒为单位）调度 sniff 执行的延迟。	60000	整数
camel.component.elasticsearch.sniffer-interval	以毫秒为单位连续执行间隔。当禁用 sniffOnFailure 时，或连续的 sniffOnFailure 之间没有故障时，将休眠。	300000	整数
camel.component.elasticsearch.socket-timeout	套接字超时前要等待的超时时间(ms)。	30000	整数
camel.component.elasticsearch.user	基本身份验证用户。		字符串

第 23 章 FHIR

支持生成者和消费者

FHIR 组件与 HAPI-FHIR 库集成，这是 Java 中 FHIR (Fast Healthcare Interoperability Resources) 规范的开源实现。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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

23.1. URI 格式

FHIR 组件使用以下 URI 格式：

```
fhir://endpoint-prefix/endpoint?[options]
```

端点前缀可以是以下之一：

- 功能
- create
- delete
- history
- load-page

- meta
- operation
- patch
- 读取
- search
- 事务
- update
- validate

23.2. 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

23.2.1. 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。

23.2.2. 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。

23.3. 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
encoding (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)	用户友善打印所有请求。	false	布尔值
serverUrl (common)	FHIR 服务器基础 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
压缩 （高级）	将传出(POST/PUT)内容压缩到 GZIP 格式。	false	布尔值
配置 （高级）	使用共享配置。		FhirConfiguration
connectionTimeout (advanced)	尝试和建立初始 TCP 连接的时间（以 ms 为单位）。	10000	整数
deferModelScanning (advanced)	当设置这个选项时，模型类不会扫描子类，直到实际访问给定类型的子列表。	false	布尔值
fhirContext (advanced)	FhirContext 是一个昂贵的对象，可以创建。为避免创建多个实例，可以直接设置它。		FhirContext
forceConformanceCheck (advanced)	强制一致性检查。	false	布尔值
sessionCookie (advanced)	要添加到每个请求的 HTTP 会话 Cookie。		字符串
socketTimeout (advanced)	单个读/写操作阻止的时间（以 ms 为单位）。	10000	整数
Summary (advanced)	请求服务器使用 _summary 参数修改响应。 Enum 值： • 数量 • 文本 • DATA • TRUE • FALSE		字符串
validationMode (advanced)	当应 Camel 验证 FHIR 服务器的一致声明时。 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

Name	描述	默认值	类型
proxyPort (proxy)	代理端口。		整数
proxyUser (proxy)	代理用户名。		字符串
accessToken (security)	OAuth 访问令牌。		字符串
密码 （安全）	用于基本身份验证的用户名。		字符串
用户名 （安全）	用于基本身份验证的用户名。		字符串

23.4. 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：

23.4.1. 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** 采取什么操作。 Enum 值： • 功能 • 创建 • DELETE • HISTORY • LOAD_PAGE • META • 操作 • PATCH • READ • 搜索 • 事务 • 更新 • VALIDATE		FhirApiName
methodName (common)	**必需的**所选操作需要哪些子操作。		字符串

23.4.2. 查询参数(44 参数)

Name	描述	默认值	类型
encoding (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 In Body 中传递的参数名称。		字符串
log (common)	将记录每个请求和响应。	false	布尔值
prettyPrint (common)	用户友善打印所有请求。	false	布尔值
serverUrl (common)	FHIR 服务器基础 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
压缩 （高级）	将传出(POST/PUT)内容压缩到 GZIP 格式。	false	布尔值
connectionTimeout (advanced)	尝试和建立初始 TCP 连接的时间（以 ms 为单位）。	10000	整数
deferModelScanning (advanced)	当设置这个选项时，模型类不会扫描子类，直到实际访问给定类型的子列表。	false	布尔值
fhirContext (advanced)	FhirContext 是一个昂贵的对象，可以创建。为避免创建多个实例，可以直接设置它。		FhirContext
forceConformanceCheck (advanced)	强制一致性检查。	false	布尔值
sessionCookie (advanced)	要添加到每个请求的 HTTP 会话 Cookie。		字符串
socketTimeout (advanced)	单个读/写操作阻止的时间（以 ms 为单位）。	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 • 文本 • DATA • TRUE • FALSE		字符串
validationMode (advanced)	当应 Camel 验证 FHIR 服务器的一致声明时。 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
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
accessToken (security)	OAuth 访问令牌。		字符串
密码 （安全）	用于基本身份验证的用户名。		字符串
用户名 （安全）	用于基本身份验证的用户名。		字符串

23.5. API 参数(13 API)

@FHIR 端点是基于 API 的组件，具有额外的参数，它基于使用的 API 名称和 API 方法。API 名称和 API 方法位于端点 URI 中，作为 apiName/methodName 路径参数：

```
fhir:apiName/methodName
```

下表中列出了 13 个 API 名称：

API 名称	类型	描述
功能	两者	API 为 Fetch 服务器的 capabilities 语句
create	两者	创建操作的 API，它会在服务器上创建新资源实例
delete	两者	删除操作的 API，它会在服务器资源上执行逻辑删除

API 名称	类型	描述
history	两者	历史记录方法的 API
load-page	两者	使用 atom 捆绑包中的链接 type=next 标签中指定的链接，从页面集中加载先前/下一步捆绑包的 API
meta	两者	meta 操作的 API，可用于从资源或服务器中获取、添加和删除标签和其他 Meta 元素
operation	两者	用于扩展 FHIR 操作的 API
patch	两者	补丁操作的 API，它会在服务器资源上执行逻辑补丁
读取	两者	读取操作的 API 方法
search	两者	API 搜索与给定条件匹配的资源
事务	两者	用于将事务（资源集合）发送到要作为单个单元执行的服务器的 API
update	两者	更新操作的 API，它会在服务器资源上执行逻辑删除
validate	两者	用于验证资源的 API

每个 API 都记录在以下部分中。

23.5.1. API: capabilities

支持生成者和消费者

capabilities API 采用语法定义，如下所示：

```
fhir:capabilities/methodName?[parameters]
```

该方法列在下表中，后跟每种方法的详细语法。(API 方法可以有一个简短的别名名称，可在语法而不是名称中使用)

æ–¹æ³•	描述
ofType	使用给定的模型类型检索 conformance 语句

23.5.1.1. Type 方法

签名：

- **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 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
type	模型类型	类

除了以上参数外，fhir API 也可以使用任何 查询参数。

任何参数都可以在端点 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 格式。inBody 参数覆盖消息标头，即 Body=myParameterNameHere 中的端点参数将覆盖 CamelFhir.myParameterNameHere 标头。

23.5.2. API: create

支持生成者和消费者

create API 采用语法定义，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 方法可以有一个简短的别名名称，可在语法而不是名称中使用)

æ–¹æ³•	描述
resource	在服务器上创建一个 IBaseResource

23.5.2.1. 方法资源

签名：

- **ca.uhn.fhir.rest.api.MethodOutcome 资源(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 资源 (org.hl7.fhir.instance.model.api.IBaseResource 资源, String url, ca.uhn.fhir.rest.api.PreferReturnEnum preferReturn, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/resource API 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
preferReturn	在请求中添加 Prefer 标头，其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 MethodOutcome114getResource （）解析客户端可访问，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。这个 URL 的格式应为 ResourceTypeParameters，例如：Patientname=Smith&identifier=13.2.4.11.4%7C847366，可能是 null	字符串

除了以上参数外，fhir API 也可以使用任何 查询参数。

任何参数都可以在端点 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 格式。inBody 参数覆盖消息标头，即 Body=myParameterNameHere 中的端点参数将覆盖 CamelFhir.myParameterNameHere 标头。

23.5.3. API: delete

支持生成者和消费者

删除 API 采用语法定义，如下所示：

```
fhir:delete/methodName?[parameters]
```

下表中列出了 3 方法，后跟每种方法的详细语法。(API 方法可以有一个简短的别名名称，可在语法而不是名称中使用)

æ–¹æ³•	描述
resource	删除给定资源
resourceById	根据资源类型删除资源，例如
resourceConditionalByUrl	指定应作为条件删除对给定搜索 URL 执行删除

23.5.3.1. 方法资源

签名：

- org.hl7.fhir.instance.model.api.IBaseOperationOutcome resource (org.hl7.fhir.instance.model.api.IBaseResource 资源, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);

fhir/resource API 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
resource	要删除的 IBaseResource	IBaseResource

23.5.3.2. method resourceById

签名：

- **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 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
id	IIdType 引用资源	IIdType
stringId	它 ID	字符串
type	资源类型，如 Patient	字符串

23.5.3.3. method resourceConditionalByUrl

签名：

- **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 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
url	要使用的搜索 URL。这个 URL 的格式应为 ResourceTypeParameters，例如：Patientname=Smith&identifier=13.2.4.11.4%7C847366	字符串

除了以上参数外，fhir API 也可以使用任何 查询参数。

任何参数都可以在端点 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 格式。inBody 参数覆盖消息标头，即 Body=myParameterNameHere 中的端点参数将覆盖 CamelFhir.myParameterNameHere 标头。

23.5.4. api: history

支持生成者和消费者

history API 采用语法定义，如下所示：

```
fhir:history/methodName?[parameters]
```

下表中列出了 3 方法，后跟每种方法的详细语法。(API 方法可以有一个简短的别名名称，可在语法而不是名称中使用)

æ–¹æ³•	描述
onInstance	在服务器中特定资源的所有版本（按 ID 和类型）执行操作
onServer	在服务器上所有类型的所有版本中执行操作
onType	在服务器中给定类型的所有版本中执行操作

23.5.4.1. 对Instance 的方法

签名：

- **org.hl7.fhir.instance.model.api.IBaseBundle onInstance（org.hl7.fhir.instance.model.api.IIdType id, 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/onInstance API 方法包含下表中列出的参数：

参数	描述	类型
æ·°é‡	请求服务器仅返回到Count 个资源数量，可以是 NULL	整数
cutoff	请求服务器只返回在给定时间（包括）时创建的资源版本，可能是 NULL	Date
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
iCutoff	请求服务器只返回在给定时间（包括）时创建的资源版本，可能是 NULL	IPrimitiveType
id	IIdType，它必须同时填充资源类型和资源 ID	IIdType
returnType	请求该方法会返回 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

23.5.4.2. Server 的方法

签名：

- **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 方法有下表中列出的参数：

参数	描述	类型
æ·°é‡	请求服务器仅返回到Count 个资源数量，可以是 NULL	整数
cutoff	请求服务器只返回在给定时间（包括）时创建的资源版本，可能是 NULL	Date
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
iCutoff	请求服务器只返回在给定时间（包括）时创建的资源版本，可能是 NULL	IPrimitiveType
returnType	请求该方法会返回 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

23.5.4.3. Type 的方法

签名：

- **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, 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 方法有下表中列出的参数：

参数	描述	类型
æ·°é‡	请求服务器仅返回到Count 个资源数量，可以是 NULL	整数
cutoff	请求服务器只返回在给定时间（包括）时创建的资源版本，可能是 NULL	Date

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
iCutoff	请求服务器只返回在给定时间（包括）时创建的资源版本，可能是 NULL	IPrimitiveType
resourceType	要搜索的资源类型	类
returnType	请求该方法会返回 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

除了以上参数外，fhir API 也可以使用任何 查询参数。

任何参数都可以在端点 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 格式。inBody 参数覆盖消息标头，即 Body=myParameterNameHere 中的端点参数将覆盖 CamelFhir.myParameterNameHere 标头。

23.5.5. api: load-page

支持生成者和消费者

load-page API 在语法中定义，如下所示：

```
fhir:load-page/methodName?[parameters]
```

下表中列出了 3 方法，后跟每种方法的详细语法。(API 方法可以有一个简短的别名名称，可在语法而不是名称中使用)

æ–¹æ³•	描述
byUrl	使用给定的 URL 和捆绑包类型加载结果页面，并返回 DSTU1 Atom 捆绑包
next	使用捆绑包中关系的链接加载下一个结果页面
previous	使用捆绑包中关系 prev 的链接加载前面的结果页面

23.5.5.1. 方法 byUrl

签名：

- **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 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
returnType	返回类型	类
url	搜索 url	字符串

23.5.5.2. Next 方法

签名：

- **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 方法有下表中列出的参数：

参数	描述	类型
bundle	IBaseBundle	IBaseBundle
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map

23.5.5.3. 前一个方法

签名：

- 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 方法有下表中列出的参数：

参数	描述	类型
bundle	IBaseBundle	IBaseBundle
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map

除了以上参数外，fhir API 也可以使用任何 查询参数。

任何参数都可以在端点 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 格式。inBody 参数覆盖消息标头，即 Body=myParameterNameHere 中的端点参数将覆盖 CamelFhir.myParameterNameHere 标头。

23.5.6. api: meta

支持生成者和消费者

meta API 在语法中定义，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 方法可以有一个简短的别名名称，可在语法而不是名称中使用)

æ–¹æ³•	描述
add	将给定元数据中的元素添加到现有集合（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	从特定类型的获取当前元数据

23.5.6.1. 方法添加

签名：

- **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 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
id	id	IIdType
meta	IBaseMetaType 类	IBaseMetaType

23.5.6.2. method delete

签名：

- **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 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
id	id	IIdType
meta	IBaseMetaType 类	IBaseMetaType

23.5.6.3. method getFromResource

签名：

- **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 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
id	id	IIdType
metaType	IBaseMetaType 类	类

23.5.6.4. method getFromServer

签名：

- **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 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
metaType	给定 FHIR 模型版本的 meta 数据类型的类型（应该为 MetaDt.class 或 MetaType.class）	类

23.5.6.5. method getFromType

签名：

- **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 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
metaType	IBaseMetaType 类	类
resourceType	资源类型，如 Patient	字符串

除了以上参数外，fhir API 也可以使用任何 查询参数。

任何参数都可以在端点 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 格式。inBody 参数覆盖消息标头，即 Body=myParameterNameHere 中的端点参数将覆盖 CamelFhir.myParameterNameHere 标头。

23.5.7. api: operation

支持生成者和消费者

操作 API 采用语法定义，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 方法可以有一个简短的别名名称，可在语法而不是名称中使用)

æ–¹æ³•	描述
onInstance	在服务器中特定资源的所有版本（按 ID 和类型）执行操作
onInstanceVersion	此操作在资源的特定版本上运行
onServer	在服务器上所有类型的所有版本中执行操作
onType	在服务器中给定类型的所有版本中执行操作
processMessage	此操作称为 $process-message，如 FHIR 规格定义

23.5.7.1. 对Instance 的方法

签名：

- **org.hl7.fhir.instance.model.api.IBaseResource onInstance（org.hl7.fhir.instance.model.api.IIdType id, String name, org.hl7.fhir.instance.model.api.IBaseParameters parameters, Class<org.hl7.fhir.instance.model.api.IBaseParameters> outputParameterType, 布尔值使用HttpGet, Class<org.hl7.fhir.instance.model.api.IBaseResource> returnType, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**

fhir/onInstance API 方法包含下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，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	输出参数使用的类型（这应该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，则可能是 null。	IBaseParameters
returnType	如果此操作将单个资源正文返回类型而不是 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/NNN/$everything）非常有用，可以是 NULL	类
useHttpGet	使用 HTTP GET 动词	布尔值

23.5.7.2. 方法 onInstanceVersion

签名：

- **org.hl7.fhir.instance.model.api.IBaseResource onInstanceVersion（org.hl7.fhir.instance.model.api.IIdType id, String name, org.hl7.fhir.instance.model.api.IBaseParameters parameters, Class<org.hl7.fhir.instance.model.api.IBaseParameters> outputParameterType, 布尔值使用HttpGet, Class<org.hl7.fhir.instance.model.api.IBaseResource> returnType, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**

fhir/onInstanceVersion API 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，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	输出参数使用的类型（这应该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，则可能是 null。	IBaseParameters
returnType	如果此操作将单个资源正文返回类型而不是 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/NNN/$everything）非常有用，可以是 NULL	类
useHttpGet	使用 HTTP GET 动词	布尔值

23.5.7.3. Server 的方法

签名：

- **org.hl7.fhir.instance.model.api.IBaseResource onServer （String name, org.hl7.fhir.instance.model.api.IBaseParameters parameters, Class<org.hl7.fhir.instance.model.api.IBaseParameters> outputParameterType, boolean useHttpGet, Class<org.hl7.fhir.instance.model.api.IBaseResource> returnType, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**

fhir/onServer API 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
name	操作名称	字符串
outputParameterType	输出参数使用的类型（这应该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，则可能是 null。	IBaseParameters

参数	描述	类型
returnType	如果此操作将单个资源正文返回类型而不是 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/NNN/$everything）非常有用，可以是 NULL	类
useHttpGet	使用 HTTP GET 动词	布尔值

23.5.7.4. Type 的方法

签名：

- **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> outputParameterType, 布尔值使用HttpGet, Class<org.hl7.fhir.instance.model.api.IBaseResource> returnType, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**

fhir/onType API 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
name	操作名称	字符串
outputParameterType	输出参数使用的类型（这应该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，则可能是 null。	IBaseParameters
resourceType	要操作的资源类型	类
returnType	如果此操作将单个资源正文返回类型而不是 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/NNN/$everything）非常有用，可以是 NULL	类

参数	描述	类型
useHttpGet	使用 HTTP GET 动词	布尔值

23.5.7.5. method processMessage

签名：

- **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, Object> extraParameters);**

fhir/processMessage API 方法有下表中列出的参数：

参数	描述	类型
异步	是否异步处理消息还是同步，默认为 sync。	布尔值
extraParameters	如需可传递的完整参数列表，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此 URI，可能是 NULL	字符串
responseClass	响应类	类

除了以上参数外，fhir API 也可以使用任何 查询参数。

任何参数都可以在端点 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 格式。inBody 参数覆盖消息标头，即 Body=myParameterNameHere 中的端点参数将覆盖 CamelFhir.myParameterNameHere 标头。

23.5.8. API: patch

支持生成者和消费者

patch API 采用语法定义，如下所示：

```
fhir:patch/methodName?[parameters]
```

下表中列出了 2 方法，后跟每种方法的详细语法。(API 方法可以有一个简短的别名名称，可在语法而不是名称中使用)

æ–¹æ³•	描述
patchById	将补丁应用到给定资源 ID
patchByUrl	指定应作为条件创建对给定搜索 URL 执行更新

23.5.8.1. method patchById

签名：

- **ca.uhn.fhir.rest.api.MethodOutcome patchById (String patchBody, 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 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 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，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	补丁文档的正文在 XML 或 JSON 中进行序列化，它们符合	字符串
preferReturn	在请求中添加 Prefer 标头，其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 MethodOutcome114getResource （）解析客户端可访问	PreferReturnEnum
stringId	要修补的资源 ID	字符串

23.5.8.2. method patchByUrl

签名：

- **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 方法包含下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
patchBody	补丁文档的正文在 XML 或 JSON 中进行序列化，它们符合	字符串
preferReturn	在请求中添加 Prefer 标头，其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 MethodOutcome114getResource （）解析客户端可访问	PreferReturnEnum
url	要使用的搜索 URL。这个 URL 的格式应为 ResourceTypeParameters，例如：Patientname=Smith&identifier=13.2.4.11.4%7C847366	字符串

除了以上参数外，fhir API 也可以使用任何 查询参数。

任何参数都可以在端点 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 格式。inBody 参数覆盖消息标头，即 Body=myParameterNameHere 中的端点参数将覆盖 CamelFhir.myParameterNameHere 标头。

23.5.9. API: read

支持生成者和消费者

read API 采用语法定义，如下所示：

```
fhir:read/methodName?[parameters]
```

下表中列出了 2 方法，后跟每种方法的详细语法。(API 方法可以有一个简短的别名名称，可在语法而不是名称中使用)

æ–¹æ³•	描述
resourceById	通过 id 读取服务器上的 IBaseResource
resourceByUrl	通过 url 读取服务器上的 IBaseResource

23.5.9.1. method resourceById

签名：

- org.hl7.fhir.instance.model.api.IBaseResource resourceById（Class<org.hl7.fhir.instance.model.api.IBaseResource> resource, Long longId, String ifVersionMatches, 布尔值 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（Class<org.hl7.fhir.instance.model.api.IBaseResource> resource, String stringId, String version, String ifVersionMatches, 布尔值 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 （Class<org.hl7.fhir.instance.model.api.IBaseResource> resource, org.hl7.fhir.instance.model.api.IIdType id, String ifVersionMatches, 布尔值 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 (String resourceClass, Long longId, 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 (String resourceClass, String stringId, String ifVersionMatches, String version, 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 （String resourceClass, org.hl7.fhir.instance.model.api.IIdType id, String ifVersionMatches, 布尔值 returnNull, org.hl7.fhir.instance.model.api.IBaseResource returnResource, boolean throwError, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；**

fhir/resourceById API 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
id	IIdType 引用资源	IIdType
ifVersionMatches	与服务器中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

参数	描述	类型
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
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，抛出错误	布尔值
version	资源版本	字符串

23.5.9.2. method resourceByUrl

签名：

- **org.hl7.fhir.instance.model.api.IBaseResource resourceByUrl（Class<org.hl7.fhir.instance.model.api.IBaseResource> resource, String url, String ifVersionMatches, 布尔值 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（Class<org.hl7.fhir.instance.model.api.IBaseResource> resource, org.hl7.fhir.instance.model.api.IIdType iUrl, String ifVersionMatches, 布尔值 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 resourceByUrl (String resourceClass, String url, 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 resourceByUrl （String resourceClass, org.hl7.fhir.instance.model.api.IIdType iUrl, String ifVersionMatches, 布尔值 returnNull, org.hl7.fhir.instance.model.api.IBaseResource returnResource, boolean**

throwError, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；

fhir/resourceByUrl API 方法包含下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
iUrl	IIdType 通过绝对 url 引用资源	IIdType
ifVersionMatches	与服务器中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，抛出错误	布尔值
url	通过绝对 url 引用资源	字符串

除了以上参数外，fhir API 也可以使用任何 查询参数。

任何参数都可以在端点 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 格式。inBody 参数覆盖消息标头，即 Body=myParameterNameHere 中的端点参数将覆盖 CamelFhir.myParameterNameHere 标头。

23.5.10. api: search

支持生成者和消费者

搜索 API 的语法定义，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 方法可以有一个简短的别名名称，可在语法而不是名称中使用)

æ–¹æ³•	描述
searchByUrl	通过 URL 直接执行搜索

23.5.10.1. method searchByUrl

签名：

- 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 方法包含下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
url	要搜索的 URL。请注意，此 URL 可以完成（例如，），在这种情况下，客户端的基本 URL 将被忽略。或者，它可以被相对（例如，Patientname=foo），在这种情况下，客户端的基本 URL 将会被使用。	字符串

除了以上参数外，fhir API 也可以使用任何 查询参数。

任何参数都可以在端点 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 格式。inBody 参数覆盖消息标头，即 Body=myParameterNameHere 中的端点参数将覆盖 CamelFhir.myParameterNameHere 标头。

23.5.11. API: transaction

支持生成者和消费者

transaction API 的语法定义，如下所示：

```
fhir:transaction/methodName?[parameters]
```

下表中列出了 2 方法，后跟每种方法的详细语法。(API 方法可以有一个简短的别名名称，可在语法而不是名称中使用)

æ–¹æ³•	描述
withBundle	使用给定的原始文本（应该是 Bundle 资源）作为事务输入
withResources	使用资源列表作为事务输入

23.5.11.1. 带有Bundle 的方法

签名：

- **带有Bundle (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 方法包含下表中列出的参数：

参数	描述	类型
bundle	在事务中使用的捆绑包	IBaseBundle
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
stringBundle	在事务中使用的捆绑包	字符串

23.5.11.2. 带有Resources 的方法

签名：

- 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 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
资源	在事务中使用的资源	list

除了以上参数外，fhir API 也可以使用任何 查询参数。

任何参数都可以在端点 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 格式。inBody 参数覆盖消息标头，即 Body=myParameterNameHere 中的端点参数将覆盖 CamelFhir.myParameterNameHere 标头。

23.5.12. API: update

支持生成者和消费者

更新 API 的语法定义，如下所示：

```
fhir:update/methodName?[parameters]
```

下表中列出了 2 方法，后跟每种方法的详细语法。(API 方法可以有一个简短的别名名称，可在语法而不是名称中使用)

æ–¹æ³•	描述
resource	通过 id 更新服务器上的 IBaseResource

æ–¹æ³•	描述
resourceBySearchUrl	通过搜索 url 更新服务器上的 IBaseResource

23.5.12.1. 方法资源

签名：

- ca.uhn.fhir.rest.api.MethodOutcome 资源(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 资源(String resourceAsString, 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);
- CA.uhn.fhir.rest.api.MethodOutcome 资源 (org.hl7.fhir.instance.model.api.IBaseResource 资源, 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 （org.hl7.fhir.instance.model.api.IBaseResource 资源, org.hl7.fhir.instance.model.api.IIdType id, ca.uhn.fhir.rest.api.PreferReturn, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters）；

fhir/resource API 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
id	IIdType 引用资源	IIdType
preferReturn	在结果中，服务器是否包含或阻止资源正文	PreferReturnEnum

参数	描述	类型
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

23.5.12.2. method resourceBySearchUrl

签名：

- **ca.uhn.fhir.rest.api.MethodOutcome resourceBySearchUrl (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 resourceBySearchUrl (org.hl7.fhir.instance.model.api.IBaseResource 资源, String url, ca.uhn.fhir.rest.api.PreferReturnEnum preferReturn, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);**

fhir/resourceBySearchUrl API 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
preferReturn	在结果中，服务器是否包含或阻止资源正文	PreferReturnEnum
resource	要更新的资源（如 Patient）	IBaseResource
resourceAsString	要更新的资源正文	字符串
url	指定应作为条件创建对给定搜索 URL 执行更新	字符串

除了以上参数外，fhir API 也可以使用任何 查询参数。

任何参数都可以在端点 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 格式。inBody 参数覆盖消息标头，即 Body=myParameterNameHere 中的端点参数将覆盖 CamelFhir.myParameterNameHere 标头。

23.5.13. api: validate

支持生成者和消费者

validate API 采用语法定义，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 方法可以有一个简短的别名名称，可在语法而不是名称中使用)

æ–¹æ³•	描述
resource	验证资源

23.5.13.1. 方法资源

签名：

- ca.uhn.fhir.rest.api.MethodOutcome 资源(String resourceAsString, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);
- ca.uhn.fhir.rest.api.MethodOutcome 资源 (org.hl7.fhir.instance.model.api.IBaseResource 资源, java.util.Map<org.apache.camel.component.fhir.api.ExtraParameters, Object> extraParameters);

fhir/resource API 方法有下表中列出的参数：

参数	描述	类型
extraParameters	如需可传递的完整参数列表，请参阅 ExtraParameters，可以是 NULL	Map
resource	用于验证的 IBaseResource	IBaseResource
resourceAsString	验证的原始资源	字符串

除了以上参数外，fhir API 也可以使用任何 查询参数。

任何参数都可以在端点 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 格式。inBody 参数覆盖消息标头，即 Body=myParameterNameHere 中的端点参数将覆盖 CamelFhir.myParameterNameHere 标头。

23.6. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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

Name	描述	默认值	类型
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)内容压缩到 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 连接的时间（以 ms 为单位）。	10000	整数
camel.component.fhir.defer-model-scanning	当设置这个选项时，模型类不会扫描子类，直到实际访问给定类型的子列表。	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 是一个昂贵的对象，可以创建。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	强制一致性检查。	false	布尔值

Name	描述	默认值	类型
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	用户友善打印所有请求。	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
camel.component.fhir.proxy-user	代理用户名。		字符串
camel.component.fhir.server-url	FHIR 服务器基础 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	单个读/写操作阻止的时间（以 ms 为单位）。	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

Name	描述	默认值	类型
camel.component.fhir.validation-mode	当应 Camel 验证 FHIR 服务器的一致声明时。	ONCE	字符串
camel.dataformat.fhirjson.content-type-header	数据格式是否应使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 例如，marshalling 到 XML，对于数据格式为 JSON，application/json 用于数据格式。	true	布尔值
camel.dataformat.fhirjson.dont-encode-elements	如果提供，请指定不应编码的元素。此字段的有效值包括： 德语 - Don't encode the the all sub Patient.name - Don't encode the the Patient.name.family - Don't encode the the family's family name .text - Don't encode the text element on any resource （只有最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 上的子元素的值（如 Patient.meta.lastUpdated ）只能在 DSTU3 模式中工作。		Set
camel.dataformat.fhirjson.dont-strip-versions-from-references-at-paths	如果提供的值，指定路径上的任何资源引用都会编码其资源版本，而不是在编码过程中自动剥离。此设置对解析过程没有影响。这个方法提供了比 setStripVersionsFromReferences (String)和此方法指定的任何路径更精细的控制级别，即使 setStripVersionsFromReferences (String)已设置为 true （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
camel.dataformat.fhirjson.encode-elements	如果提供，请指定应编码的元素，以排除所有其他元素。此字段的有效值包括：Patient - Encode the and all child Patient.name - Encode only the Patient.name.family - Encode only the good's 系列名称 .text - Encode - 对任何资源上的文本元素进行编码（仅限第一个位置可能包含通配符）-（必需）- 这个特殊情况，它导致任何资源的系列名称（最小 0）被编码。		Set
camel.dataformat.fhirjson.encode-elements-applies-to-child-resources-only	如果设置为 true （默认为 false），则提供给 setEncodeElements (Set)的值不会应用到根资源（通常是 Bundle），但将应用到包含在其中的任何子资源（例如，搜索捆绑包中的资源）。	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

Name	描述	默认值	类型
camel.dataformat.fhirjson.omit-resource-id	如果设置为 true （默认为 false），则编码的任何资源的 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，如果这设为 true，则包含的资源仍会具有本地 ID，但外部/包含 ID 将不会有 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），如果定义了 fullUrl，Bundle.entry.fullUrl 将覆盖 Bundle.entry.resource 的资源 ID。在将源数据解析到 Bundle 对象时，会发生此行为。如果这不是所需行为，则将其设置为 false （例如，客户端代码希望在 fullUrl 和资源 ID 之间执行额外的验证检查）。	false	布尔值
camel.dataformat.fhirjson.pretty-print	设置用户友善的 print 标志，这意味着解析器将使用人类可读和元素之间的新行编码资源，而不是尽可能高的输出。	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，如果资源引用以绝对 URL 提供，但具有与给定基础匹配的基础，则资源引用将转换为相对引用。		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编码资源时删除版本。这通常很好，因为在大多数情况下，从一个资源到另一个资源的引用应该是按 ID 到资源，而不是按 ID 和版本对资源。然而，在某些情况下可能需要在资源链接中保留版本。在这种情况下，这个值应设为 false。这个方法提供了全局禁用参考编码的功能。如果需要精细控制，请使用 setDontStripVersionsFromReferencesAtPaths (List)。	false	布尔值
camel.dataformat.fhirjson.summary-mode	如果设置为 true （默认为 false），则仅包含 FHIR 规格标记为概述元素的元素。	false	布尔值
camel.dataformat.fhirjson.suppress-narratives	如果设置为 true （默认为 false），则 narratives 不会包含在编码的值中。	false	布尔值
camel.dataformat.fhirxml.content-type-header	数据格式是否应使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 例如，marshalling 到 XML，对于数据格式为 JSON，application/json 用于数据格式。	true	布尔值

Name	描述	默认值	类型
camel.dataformat.fhirxml.dont-encode-elements	如果提供，请指定不应编码的元素。此字段的有效值包括： 德语 - Don't encode the the all sub Patient.name - Don't encode the the Patient.name.family - Don't encode the the family's family name .text - Don't encode the text element on any resource （只有最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 上的子元素的值（如 Patient.meta.lastUpdated ）只能在 DSTU3 模式中工作。		Set
camel.dataformat.fhirxml.dont-strip-versions-from-references-at-paths	如果提供的值，指定路径上的任何资源引用都会编码其资源版本，而不是在编码过程中自动剥离。此设置对解析过程没有影响。这个方法提供了比 setStripVersionsFromReferences (String)和此方法指定的任何路径更精细的控制级别，即使 setStripVersionsFromReferences (String)已设置为 true （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，请指定应编码的元素，以排除所有其他元素。此字段的有效值包括：Patient - Encode the and all child Patient.name - Encode only the Patient.name.family - Encode only the good's 系列名称 .text - Encode - 对任何资源上的文本元素进行编码（仅限第一个位置可能包含通配符）-（必需）- 这个特殊情况，它导致任何资源的系列名称（最小 0）被编码。		Set
camel.dataformat.fhirxml.encode-elements-applies-to-child-resources-only	如果设置为 true （默认为 false），则提供给 setEncodeElements (Set)的值不会应用到根资源（通常是 Bundle），但将应用到包含在其中的任何子资源（例如，搜索捆绑包中的资源）。	false	布尔值
camel.dataformat.fhirxml.fhir-version	要使用的 FHIR 版本。可能的值有：DSTU2,DSTU2_HL7ORG,DSTU2_1,DSTU3,R4。	DSTU3	字符串
camel.dataformat.fhirxml.omit-resource-id	如果设置为 true （默认为 false），则编码的任何资源的 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，如果这设为 true，则包含的资源仍会具有本地 ID，但外部/包含 ID 将不会有 ID。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.fhirxml.override-resource-id-with-bundle-entry-full-url	如果设置为 true （默认值），如果定义了 fullUrl，Bundle.entry.fullUrl 将覆盖 Bundle.entry.resource 的资源 ID。在将源数据解析到 Bundle 对象时，会发生此行为。如果这不是所需行为，则将其设置为 false （例如，客户端代码希望在 fullUrl 和资源 ID 之间执行额外的验证检查）。	false	布尔值
camel.dataformat.fhirxml.pretty-print	设置用户友善的 print 标志，这意味着解析器将使用人类可读和元素之间的新行编码资源，而不是尽可能高的输出。	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，如果资源引用以绝对 URL 提供，但具有与给定基础匹配的基础，则资源引用将转换为相对引用。		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编码资源时删除版本。这通常很好，因为在大多数情况下，从一个资源到另一个资源的引用应该是按 ID 到资源，而不是按 ID 和版本对资源。然而，在某些情况下可能需要在资源链接中保留版本。在这种情况下，这个值应设为 false。这个方法提供了全局禁用参考编码的功能。如果需要精细控制，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 FHIR 规格标记为概述元素的元素。	false	布尔值
camel.dataformat.fhirxml.suppress-narratives	如果设置为 true （默认为 false），则 narratives 不会包含在编码的值中。	false	布尔值

第 24 章 FILE

支持生成者和消费者

File 组件提供对文件系统的访问，允许由其他组件的任何其他 Camel 组件或来自其他组件的消息保存到磁盘中处理文件。

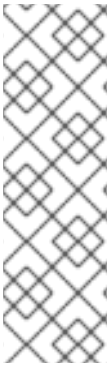
24.1. URI 格式

```
file:directoryName[?options]
```

其中 directoryName 代表底层文件目录。

只有目录

Camel 仅支持使用启动目录配置的端点。因此，directoryName 必须是目录。如果只使用单个文件，您可以使用 fileName 选项，例如通过设置 fileName=thefilename。另外，启动目录不得包含带有 ${ } 占位符的动态表达式。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
成为JDK File IO API 在检测另一个应用程序当前是否写入/复制文件时有一定限制。此外，实施也可以因操作系统平台而异。这可能会导致 Camel 认为该文件不会被其他进程锁定，并开始消耗它。因此，您必须自己调查您的环境套件。为了帮助获得此 Camel，可以使用不同的 readLock 选项和 doneFileName 选项。另请参阅 节，其中存储了其他人直接丢弃文件的文件夹。

24.2. 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

24.2.1. 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。

24.2.2. 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。

24.3. 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

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

24.4. 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：

24.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
directoryName (common)	**需要** 启动目录。		File

24.4.2. 查询参数(94 参数)

Name	描述	默认值	类型
charset (common)	此选项用于指定文件编码。您可以在消费者上使用此功能，以指定文件的编码，它允许 Camel 在访问文件内容时加载文件内容。在编写文件时，您可以使用此选项指定写入该文件的 charset。请注意，当编写 Camel 文件时，可能需要将消息内容读取在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

Name	描述	默认值	类型
doneFileName (common)	producer ：如果提供，则 Camel 将在编写原始文件时写入第二次完成的文件。done 文件为空。这个选项配置要使用的文件名。您可以指定固定名称。或者，您可以使用动态占位符。done 文件将始终写在与原始文件相同的文件夹中。consumer ：如果提供，Camel 仅在文件存在时使用文件。这个选项配置要使用的文件名。您可以指定固定名称。或者，您可以使用动态占位符。done 文件始终预期在与原始文件相同的文件夹中。仅支持 $\\{file.name} 和 $\\{file.name.next} 作为动态占位符。		字符串
fileName (common)	使用 Expression （如文件语言）来动态设置文件名。对于消费者，它用作文件名过滤器。对于生成者，它用于评估要写入的文件名。如果设置了表达式，它将优先于 CamelFileName 标头。（注意：标头本身也可以是 Expression）。表达式选项支持 String 和 Expression 类型。如果表达式是字符串类型，则始终使用 File Language 来评估。如果表达式是 Expression 类型，则会使用指定的 Expression 类型 - 例如，您可以使用 OGNL 表达式。对于消费者，您可以使用它来过滤文件名，因此您可以使用 File Language 语法：mydata-$\\{date:now:yyyyMMdd}.txt 来使用现在使用的文件。producers 支持 CamelOverruleFileName 标头，其优先于任何现有 CamelFileName 标头；CamelOverruleFileName 是一个仅一次的标头，并便于避免临时存储 CamelFileName 并在以后恢复它。		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，则会在成功处理文件后删除该文件。	false	布尔值
moveFailed (consumer)	根据简单语言设置移动失败表达式。例如，要将文件移到 .error 子目录使用：.error。注意：将文件移到失败位置 Camel 将处理错误时，不会再次获取该文件。		字符串

Name	描述	默认值	类型
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 ETL 类型要求。如果 noop=true，Camel 也设置幂等=true，以避免再次消耗相同的文件。	false	布尔值
preMove (consumer)	在处理前移动文件名的表达式（如文件语言）。例如，要将 in-progress 文件移到 order 目录中，将此值设置为 order。		字符串
preSort (consumer)	启用预排序后，消费者将在轮询过程中对文件和目录名称进行排序，这从文件系统检索。如果您需要按照排序的顺序对文件进行操作，您可能需要执行此操作。预排序是在消费者开始过滤前执行的，并接受由 Camel 处理的文件。这个选项是 default=false 表示禁用。	false	布尔值
recursion (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，但这在轮询过程中适用（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 in-progress 存储库 org.apache.camel.spi.IdempotentRepository。in-progress 存储库用于考虑当前正在使用的进程文件中。默认使用基于内存的存储库。		IdempotentRepository
localWorkDirectory (consumer (advanced))	消耗时，本地工作目录可用于直接将远程文件内容存储在本地文件中，以避免将内容加载到内存中。这很有用，如果您消耗非常大的远程文件，因此可以节省内存。		字符串
onCompletionExceptionHandler (consumer (advanced))	要使用自定义 org.apache.camel.spi.ExceptionHandler 来处理在消费者进行提交或回滚的完成过程中发生的任何抛出异常。默认实现将在 WARN 级别和忽略时记录任何异常。		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 Files114probeContentType (java.nio.file.Path)来确定文件的 content-type，并将该类型存储为 Message 上带有密钥 ExchangeFILE_CONTENT_TYPE 的标头。	false	布尔值
processStrategy (consumer (advanced))	一个可插拔的 org.apache.camel.component.file.GenericFileProcessStrategy，允许您实现自己的 readLock 选项或类似。在消耗文件之前，也可以使用特殊条件，如存在特殊的就绪文件。如果设置了这个选项，则不会应用 readLock 选项。		GenericFileProcessStrategy
resumeStrategy (consumer (advanced))	为文件设置恢复策略。这样便可定义一个策略，以便在最后一次点后恢复文件，然后再停止应用程序。有关实现详情，请参阅 FileConsumerResumeStrategy。		FileConsumerResumeStrategy
startingDirectoryMustExist (consumer (advanced))	起始目录必须存在。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，则通常会自动创建。您可以禁用 autoCreate 并启用此功能以确保启动目录必须存在。如果目录不存在，将抛出异常。	false	布尔值
startingDirectoryMustHaveAccess (consumer (advanced))	启动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)	用于在写入文件后附加字符（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。要指定新行(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)	如果文件已存在具有相同名称的文件，则该怎么办。覆盖（这是默认设置）替换现有文件。- Append - 将内容添加到现有文件中。- Fail - 抛出一个 GenericFileOperationException，表示已有的文件。- Ignore - 静默忽略问题，且不会覆盖现有的文件，但假设所有内容正常。- Move - 选项需要使用 moveExisting 选项。选项 eagerDeleteTargetFile 可以用来控制移动文件时要执行的操作，并且已存在现有文件，否则会导致移动操作失败。Move 选项将在编写目标文件前移动任何现有文件。- 只有在使用 tempFileName 选项时才适用 TryRename。这允许尝试将文件从临时名称重命名为实际名称，而无需执行任何存在检查。对于某些文件系统，这个检查可能会更快，特别是 FTP 服务器。 Enum 值： • 覆盖 • 附加 • Fail • Ignore • Move • TryRename	覆盖	GenericFileExist
flatten (producer)	扁平化用于扁平化任何前导路径的文件名路径，因此它只是文件名。这样，您可以递归地将文件写入子目录中，但当您将文件写入另一个目录中时，它们将写入到单个目录中。在制作者上将其设置为 true 可强制在 CamelFileName 标头中为任何前导路径去除任何文件名。	false	布尔值
jailStartingDirectory (producer)	仅用于 jailing （限制）将文件写入启动目录（和子）。默认情况下，这不允许 Camel 将文件写入外部目录（其开箱即用）。您可以将其关闭，以允许将文件写入启动目录之外的目录，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)	配置 fileExist=Move 时使用的表达式（如文件语言）用于计算文件名。将文件移到备份子目录中，只需输入备份。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至基于当前 dir 作为基础的相对目录中。		字符串
tempFileName (producer)	与 tempPrefix 选项相同，但提供对临时文件名命名更加精细的控制，因为它使用 File 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 包含一个目录前缀：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，并避免消费者（不使用专用读锁定）读取进度文件中。FTP 通常在上传大型文件时使用。		字符串
allowNullBody (producer (advanced))	用于指定在文件写入过程中是否允许 null 正文。如果设置为 true，则会创建一个空文件，设置为 false，并尝试向文件组件发送 null 正文，将抛出 'Cannot write null body 的 GenericFileWriteException。'如果 fileExist 选项设为 '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；如果存在前导数字，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；如果存在前导位，如 0755，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。您可以使用它来禁用（将其设置为 false）在写入临时文件前删除目标文件。例如，您可以编写大型文件，并希望在编写临时文件过程中存在目标文件。这样可确保目标文件仅在最后一次时间之前被删除，只需在临时文件重命名为目标文件名之前。此选项还用于控制在启用 fileExist=Move 时是否删除任何现有文件，并且存在现有的文件。如果此选项 copyAndDeleteOnRenameFails false，那么如果现有文件存在，则会抛出异常（如果为 true），则在移动操作前删除现有文件。	true	布尔值
forceWrites (producer (advanced))	是否强制同步到文件系统。如果您不希望此级别的保证，您可以关闭此级别，例如，如果写入日志 / 审计日志等；这会产生更好的性能。	true	布尔值
keepLastModified (producer (advanced))	将保留源文件的最后修改的时间戳（若有）。将使用 Exchange.FILE_LAST_MODIFIED 标头来定位时间戳。此标头可以包含 java.util.Date 或带有时间戳的长度。如果时间戳存在，并且启用了 选项，它将在写入的文件中设置此时间戳。注：此选项仅适用于文件制作者。您不能将此选项与任何 ftp producer 一起使用。	false	布尔值
moveExistingFileStrategy (producer (advanced))	用于在配置 fileExist=Move 时使用特殊命名令牌的策略（自定义策略）。默认情况下，如果没有提供自定义策略，则使用实施。		FileMoveExistingStrategy
auto create (advanced)	在文件的路径中自动创建缺少的目录。对于文件消费者，这意味着创建启动目录。对于文件制作者，这意味着应将文件写入的目录。	true	布尔值
BufferSize (advanced)	用于编写文件的缓冲区大小（或者 FTP 用于下载和上传文件）。	131072	int
copyAndDeleteOnRenameFail (advanced)	是否在无法直接重命名文件时，回退以及执行复制和删除文件。FTP 组件无法使用这个选项。	true	布尔值
renameUsingCopy (advanced)	使用复制和删除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中）。这个选项优先于 copyAndDeleteOnRenameFail 参数，该参数将自动回退到复制和删除策略，但只有在额外的延迟后。	false	布尔值

Name	描述	默认值	类型
Sync (advanced)	设置是否应严格使用同步处理。	false	布尔值
antExclude (filter)	ant 风格的过滤器排除。如果同时使用 antInclude 和 antExclude，则 antExclude 优先于 antInclude。可以使用以逗号分隔的格式指定多个排除项。		字符串
antFilterCaseSensitive (filter)	在 ant 过滤器上设置区分大小写标志。	true	布尔值
antInclude (filter)	ant 风格的过滤器包括。可使用以逗号分隔的格式指定多个 include。		字符串
eagerMaxMessagesPerPoll (filter)	允许控制 maxMessagesPerPoll 的限制是 eager。如果 eager，则在扫描文件的过程中会进行限制。其中为 false 将扫描所有文件，然后执行排序。将这个选项设置为 false 允许首先排序所有文件，然后限制轮询。请注意，这需要更高的内存用量，因为所有文件详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）（匹配不区分大小写）。请注意，如果您使用加号等符号，如果将其配置为端点 uri，则需要使用 RAW（）语法进行配置。有关配置端点 uri 的更多信息，请参阅。		字符串
excludeExt (filter)	用于排除与文件扩展名名称匹配的文件（不区分大小写）。例如，要排除 bak 文件，则使用 excludeExt=bak。可以使用逗号分隔多个扩展，例如排除 bak 和 dat 文件，使用 excludeExt=bak,dat。请注意，文件扩展名包含所有部分，例如，有一个名为 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 accept （）方法中返回 false，则会跳过文件。		GenericFileFilter
filterDirectory (filter)	根据简单语言过滤目录。例如，要过滤当前日期，您可以使用简单的日期模式，如 $\\{date:now: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
幂等 （过滤）	使用 Idempotent Consumer EIP 模式的选项，以便 Camel 跳过已经处理的文件。默认情况下，将使用基于内存的 LRUCache，其中包含 1000 个条目。如果 noop=true 也会启用，以避免再次消耗相同的文件。	false	布尔值

Name	描述	默认值	类型
idempotentKey (filter)	使用自定义幂等键，请执行以下操作：默认情况下，使用该文件的绝对路径。您可以使用 File Language，例如使用文件名和文件大小，您可以执行以下操作：OnlydKey=$\\{file:name}-$\\{file:size}。		字符串
idempotentRepository (filter)	可插拔存储库 org.apache.camel.spi.IdempotentRepository，它默认使用 MemoryIdempotentRepository （如果没有指定），且幂等性为 true。		IdempotentRepository
include (filter)	用于包含文件，如果文件名与正则表达式模式匹配（匹配不区分大小写）。请注意，如果您使用加号等符号，如果将其配置为端点 uri，则需要使用 RAW （）语法进行配置。有关配置端点 uri 的更多信息，请参阅。		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，则使用 includeExt=txt。可以使用逗号分隔多个扩展，例如包含 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，有一个名为 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)	定义每次轮询收集的最大消息。默认情况下不设置最大值。可用于设置限制，例如 1000，以避免在启动有数千个文件的服务器时避免。将值设为 0 或负数设置为禁用它。注意：如果使用这个选项，则文件和 FTP 组件将在任何排序前受到限制。例如，如果您有 100000 文件并使用 maxMessagesPerPoll=500，则仅获取前 500 个文件，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，然后对文件进行排序。		int
minDepth (filter)	递归处理目录时开始处理的最小深度。使用 minDepth=1 表示主目录。使用 minDepth=2 表示第一个子目录。		int
Move (filter)	在处理后移动文件名的表达式（如简单语言）。将文件移到 .done 子目录中，只需输入 .done。		字符串
exclusiveReadLockStrategy (lock)	可插拔读锁定作为 org.apache.camel.component.file.GenericFileExclusiveReadLockStrategy 实现。		GenericFileExclusiveReadLockStrategy

Name	描述	默认值	类型
readLock (lock)	消费者使用，仅在文件有独占的 read-lock 时轮询文件（例如，该文件不是 in-progress 或 in-progress 或正在写入）。Camel 将等到赋予文件锁定。此选项在策略中提供构建： - none - No read lock is use - markFile - Camel 创建标记文件 (fileName.camelLock)，然后在其上保存锁定。FTP 组件无法使用这个选项 - Changed - Changed 使用文件长度/修改时间戳来检测文件当前是否复制。至少将使用 1 sec 来确定这一点，因此此选项无法像其他进程一样快速使用文件，但可能会更可靠，因为 JDK IO API 并不总是确定当前由其他进程使用的文件。选项 readLockCheckInterval 可用于设置检查频率。 - fileLock - 使用 java.nio.channels.FileLock。这个选项不适用于 Windows OS 和 FTP 组件。在通过 mount/share 访问远程文件系统时，应该避免这种方法，除非该文件系统支持分布式文件锁定。- 重命名是使用尝试将文件重命名为测试（如果我们可以获得专用 read-lock. - 幂等的 - （仅用于文件组件）用于使用幂等结构作为读锁定。如果幂等存储库实现支持集群，这允许使用支持集群的读锁定。- idempotent-changed - （仅限文件组件）幂等性更改是使用幂等的 Repository，并作为组合的 read-lock 进行更改。如果幂等存储库实现支持集群，这允许使用支持集群的读锁定。- idempotent-rename - （仅用于文件组件）幂等名称使用幂等权限，并重命名为组合的 read-lock。如果幂等存储库实现支持集群，这允许使用支持集群的读锁定：各种读锁定并不是在集群模式下工作的，其中不同节点上的并发消费者对共享文件系统中的同一文件都是竞争的。使用一个接近原子操作的 markFile 来创建空标记文件，但无法保证在集群中工作。fileLock 可能更好，但文件系统需要支持分布式文件锁定等。如果幂等存储库支持集群，如 Hazelcast 组件或 Infinispan，则使用幂等读锁定支持集群。 Enum 值： • none • markerFile • fileLock • rename • changed • idempotent • idempotent-changed • idempotent-rename	none	字符串

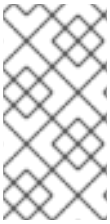
Name	描述	默认值	类型
readLockCheckInterval (lock)	read-lock 的 millis （如果读锁定支持）。这个间隔用于在尝试获取读锁定之间休眠。例如，在使用更改的读锁定时，您可以为较慢的写入设置更高的间隔周期。如果生成者非常慢，则默认值 1 sec. 可能太快。注意：对于 FTP，默认的 readLockCheckInterval 为 5000。readLockTimeout 值必须大于 readLockCheckInterval，但 thumb 的规则是具有至少 2 个或大于 readLockCheckInterval 的超时。这需要确保读锁定进程允许冻结时间，以便在超时命中前尝试获取锁定。	1000	long
readLockDeleteOrphanLockFiles (lock)	如果 Camel 没有正确关闭，则是否应在启动时读取带有标记文件（比如 JVM 崩溃）删除任何孤立的读锁定文件（如果 Camel 没有被正确关闭）。如果将此选项设置为 false，则任何孤立的锁定文件将导致 Camel 不尝试获取该文件，这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还是异步。请参阅 readLockIdempotentReleaseDelay 选项的详情。	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。在几乎所有用例中，使用默认的 1 个内核线程应该足够了，只有在更新幂等存储库缓慢或者有大量文件可以处理时，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，则此选项不会被使用。请参阅 readLockIdempotentReleaseDelay 选项的详情。		int
readLockIdempotentReleaseDelay (lock)	是否在 millis 期间延迟发行任务。当文件被视为读锁定时，这可用于延迟发行任务来扩展窗口，在使用共享幂等存储库的主动/主动集群场景中，以确保其他节点无法扫描并获取同一文件，因为竞争条件。通过延长发行任务的时间时间有助于防止这些情况。请注意，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。这个选项仅适用于 readLock 类型： changed, fileLock, idempotent, idempotent-changed, idempotent-rename, rename。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	DEBUG	LoggingLevel
readLockMarkerFile (lock)	是否将标记文件与更改、重命名或专用读取锁定类型一起使用。默认情况下，使用标记文件来保护其他进程获取同一文件。通过将这个选项设置为 false，可以关闭此行为。例如，如果您不想由 Camel 应用程序将标记文件写入文件系统。	true	布尔值
readLockMinAge (lock)	这个选项仅适用于 readLock=changed。它允许在尝试获取读取锁定前指定该文件的最短期限。例如，使用 readLockMinAge=300s 来要求文件持续 5 分钟。这可加快更改的读锁定速度，因为它将尝试获取至少给定时间的文件。	0	long
readLockMinLength (lock)	这个选项仅适用于 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置为零，以允许消耗零长度文件。	1	long
readLockRemoveOnCommit (lock)	这个选项仅适用于 readLock=idempotent。它允许在处理文件成功并发生提交时指定是否从幂等存储库中删除文件名条目。默认情况下，该文件不会被删除，这样可确保不会发生任何竞争条件，因此另一个活跃节点可能会尝试获取该文件。相反，幂等存储库可以支持驱除策略，您可以在 X 分钟后驱除文件名条目 - 这样可确保出现竞争条件的问题。请参阅 readLockIdempotentReleaseDelay 选项的详情。	false	布尔值
readLockRemoveOnRollback (lock)	这个选项仅适用于 readLock=idempotent。它允许在处理文件失败时指定是否从幂等存储库中删除文件名条目，并且进行回滚。如果此选项为 false，则确认文件名条目（就像文件进行了提交一样）。	true	布尔值

Name	描述	默认值	类型
readLockTimeout (lock)	可选的 timeout （如果 read-lock 支持）用于读锁定。如果无法授予 read-lock 并触发超时，则 Camel 将跳过该文件。下一次轮询 Camel 时，将再次尝试 文件，这一次可能会授予读锁定。使用 0 或较低值来指示永久值。当前 fileLock、change 和 rename 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 的规则是具有至少 2 个或大于 readLockCheckInterval 的超时。这需要确保读锁定进程允许冻结时间，以便在超时命中前尝试获取锁定。	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)	影响文件列表（按随机顺序排列）。	false	布尔值
sortBy (sort)	使用文件语言内置排序。支持嵌套排序，以便您可以按文件名排序，并作为按修改日期排序的第二个组。		字符串
排序 者（排序）	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。

24.5. 移动和删除操作

任何移动或删除操作都会在(post 命令)完成后执行；因此在处理 交换 过程中，该文件仍然位于 inbox 文件夹中。

通过一个示例来演示这一点：

```
from("file://inbox?move=.done").to("bean:handleOrder");
```

当文件在 inbox 文件夹中丢弃时，文件消费者会注意到这一点，并创建一个新的 FileExchange，它路由到 handleOrder bean。然后 bean 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（但是，如果您使用一个表达式（如 文件 语言）或 Simple，则表达式评估的结果是要使用的文件名。例如，如果设置了：

```
move=../backup/copy-of-${file:name}
```

然后，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_integration/2023.q2/html-single/camel_spring_boot_reference/index#csb-camel-file-language-starter如果相对，则该目录创建为使用该文件的文件夹的子文件夹。

默认情况下，Camel 会将消耗的文件移到相对于文件消耗的目录的 .camel 子文件夹。

如果要在处理后删除该文件，路由应该是：

```
from("file://inbox?delete=true").to("bean:handleOrder");
```

在处理文件 前，我们引入了一个 预 移动操作来移动文件。这可让您在处理前标记哪些文件被扫描，因为这些文件将移到这个子文件夹中。

```
from("file://inbox?preMove=inprogress").to("bean:handleOrder");
```

您可以组合 预 移动和常规移动：

```
from("file://inbox?preMove=inprogress&move=.done").to("bean:handleOrder");
```

因此，在这种情况下，文件在处理时位于 inprogress 文件夹中，并在处理后移至 .done 文件夹。

24.6. 对 MOVE 和 PREMOVE 选项进行精细控制

move 和 preMove 选项基于表达式，因此我们拥有 文件 语言的完整功能，以执行目录和名称模式的高级配置。
实际上，Camel 会在内部将您输入的目录名称转换为 文件 语言表达式。因此，当我们输入 move=.done Camel 时，会将其转换为 ：${file:parent}/.done/${file:onlyname}。这只有在 Camel 检测到您没有自行在选项值中提供 $\{ } 时才完成。因此，当您输入 $\{ } Camel 时，不会 转换它，因此您有完整的电源。

因此，如果我们想将文件移到备份文件夹中，且现在作为模式，我们可以：

```
move=backup/${date:now:yyyyMMdd}/${file:name}
```

24.7. 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，要移动带有时间戳的错误文件夹中的文件，您可以使用 moveFailed=/error/${file:name.noext}-${date:now:yyyyMMddHHmmssSSS}.${\'\'file:ext}。

请参阅更多示例

24.8. 消息标头

此组件支持以下标头：

24.8.1. 仅限文件制作者

标头	描述
CamelFileName	指定要写入的文件的名称（相对于端点目录）。此名称可以是 **字符串** ；包含文件语言或 简单 语言表达式的字符串；或一个 Expression 对象。 **https://access.redhat.com/documentation/zh-cn/red_hat_integration/2023.q2/html-single/camel_spring_boot_reference/index#csb-camel-file-language-starter如果为空，则 Camel 将根据消息唯一 ID 自动生成文件名。**
CamelFileNameProduced	所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，其用途是为最终用户提供所写入文件的名称。
CamelOverruleFileName	用于过度检测 **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）。该值只能是一个字符串。请注意，如果配置了 **fileName** 选项，则仍然会被评估。

24.8.2. 仅限文件消费者

标头	描述
CamelFileName	使用的文件名称作为相对文件路径，从端点上配置的起始目录中偏移。
CamelFileNameOnly	仅包括文件名（没有前导路径的名称）。
CamelFileAbsolute	指定消耗的文件是否表示绝对路径 **的布尔值** 选项。对于相对路径，通常是 **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。但也可在其他位置使用。
CamelFileAbsolutePath	文件的绝对路径。对于相对文件，这个路径保存相对路径。
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 **Long** 值。

24.9. BATCH CONSUMER

这个组件实现了 **Batch Consumer**。

24.10. 交换属性，仅限文件消费者

由于文件消费者实施 BatchConsumer，它支持批处理它轮询的文件。通过批处理，我们意味着 Camel 将在交换中添加以下额外属性，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	在此批处理中轮询的文件总数。
CamelBatchIndex	批处理的当前索引。从 0 开始。
CamelBatchComplete	指示批处理中最后一个交换的布尔值。仅对最后一个条目为 **true**。

这可让您让实例知道此批处理中存在多少个文件，例如，让聚合器2 聚合此数量的文件。

24.11. 使用 CHARSET

charset 选项允许在消费者和制作者端点上配置文件编码。例如，如果您读取 utf-8 文件，并希望将文件转换为 iso-8859-1，您可以执行以下操作：

```
from("file:inbox?charset=utf-8")
  .to("file:outbox?charset=iso-8859-1")
```

您还可以在路由中使用 convertBodyTo。在以下示例中，我们仍然以 utf-8 格式输入文件，但我们希望将文件内容转换为 iso-8859-1 格式的字节数组。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");
```

如果您省略了消费者端点上的 charset，则 Camel 不知道文件的 charset，默认情况下会使用 "UTF-8"。但是，您可以配置 JVM 系统属性来覆盖和使用带有 org.apache.camel.default.charset 键的不同默认编码。

在以下示例中，如果文件不在 UTF-8 编码中，这可能是一个问题，这是读取文件的默认编码。
在这个示例中，在编写文件时，内容已转换为字节数组，因此会将内容直接写入（无需进一步的编码）。

```
from("file:inbox")
  .convertBodyTo(byte[].class, "iso-8859-1")
  .to("bean:myBean")
  .to("file:outbox");
```

您还可以通过使用密钥 Exchange.CHARSET_NAME 在交换上设置属性来覆盖和控制编码动态。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");
```

我们建议保持简单操作，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，那么最好在端点上使用 charset 选项。

请注意，如果您在端点上明确配置了 charset 选项，则无论 Exchange.CHARSET_NAME 属性是什么，都会使用该配置。

如果您有一些问题，您可以在 org.apache.camel.component.file 上启用 DEBUG 日志记录，并在其使用特定的 charset 读取/写入文件时 Camel 日志。
例如，以下示例将记录以下内容：

```
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
```

24.12. 常用带有文件夹和文件名的 GETCHAS

当 Camel 生成文件时（写入文件）有一些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，因为消息 ID 通常是唯一的 ID，因此您将以文件名结尾，例如： ID-MACHINENAME-2443-1211718892437-1-0。如果不需要这样的文件名，则必须在 CamelFileName 消息标头中提供一个文件名。也可以使用 constant (Exchange.FILE_NAME)。

以下示例代码使用消息 ID 作为文件名生成文件：

```
from("direct:report").to("file:target/reports");
```

要使用 report.txt 作为您必须做的文件名：

```
from("direct:report").setHeader(Exchange.FILE_NAME, constant("report.txt")).to(
"file:target/reports");
```

- 与上述相同，但使用 CamelFileName ：

```
from("direct:report").setHeader("CamelFileName", constant("report.txt")).to(
"file:target/reports");
```

和语法，我们使用 fileName URI 选项在端点上设置文件名。

```
from("direct:report").to("file:target/reports/?fileName=report.txt");
```

24.13. 文件名表达式

文件名可以使用 expression 选项或作为 Camel File Name 标头中的基于文件的文件语言表达式进行设置。有关语法和示例，请参阅 文件 语言。

24.14. 从其他文件直接丢弃的文件夹中消耗文件

如果您使用其他应用程序将文件直接写入的文件夹中的文件，则 beware。查看不同的 readLock 选项以查看您的用例。最好方法是写入另一个文件夹，并在写入移动 drop 文件夹中的文件后写入。但是，如果您直接将文件写入 drop 文件夹，则选项更改可能会更好地检测文件当前是否被写入/记录，因为它使用文件更改的算法来查看文件大小/修改更改。其他 readLock 选项依赖于 Java 文件 API，这些文件在检测时并不好。您可能还希望查看 doneFileName 选项，该选项在文件完成后使用标志文件（解压文件）来信号，并可供使用。

24.15. 使用 DONE 文件

另请参阅下面 *编写 done 文件* 的部分。

如果您只想在文件存在时消耗文件，您可以在端点上使用 doneFileName 选项。

```
from("file:bar?doneFileName=done");
```

如果在目标文件的同一个目录中存在一个 done *文件*，将仅使用 bar 文件夹中的文件。在完成使用文件或，Camel 将自动删除 *done 文件*。如果配置了 noop=true，Camel 不会删除 *done 文件*。

但是，每个目标文件有一个 *完整的文件* 更为常见。这意味着有一个 1:1 个关联。要做到这一点，您必须在 doneFileName 选项中使用动态占位符。目前，Camel 支持以下动态令牌： file:name 和 file:name.noext，这些令牌必须用 $\{ } 括起来。消费者只支持以前缀或后缀（不支持两者） *完成的文件名* 的静态部分。

```
from("file:bar?doneFileName=${file:name}.done");
```

在本例中，只有在存在名为 .done 的文件时，仅轮询 *文件*。例如：

- hello.txt - 是要使用的文件
- 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 - 是要使用的文件
- ready-hello.txt - 是关联的 done 文件

24.16. 编写完成的文件

编写一个文件后，您可能希望将额外的 *done文件* 写为一种标记，以指示该文件已完成并已写入的其他人。为此，您可以在文件制作者端点上使用 doneFileName 选项。

```
.to("file:bar?doneFileName=done");
```

将简单地在与目标文件相同的目录中创建名为 done 的文件。

但是，每个目标文件有一个完整的文件更为常见。这意味着有一个 1:1 个关联。要做到这一点，您必须在 doneFileName 选项中使用动态占位符。目前，Camel 支持以下动态令牌： file:name 和 file:name.noext，这些令牌必须用 $\{ } 括起来。

```
.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。

24.17. SAMPLES

24.17.1. 从目录读取和写入到另一个目录

```
from("file://inputdir/?delete=true").to("file://outputdir")
```

24.17.2. 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，并为在那里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。

24.17.3. 从目录中递归读取并写入另一个目录

```
from("file://inputdir/?recursive=true&delete=true").to("file://outputdir")
```

侦听目录，并为在那里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中。将把文件放在与 input dir 相同 的目录结构中，包括任何子目录。

```
inputdir/foo.txt
inputdir/sub/bar.txt
```

将导致以下输出布局：

```
outputdir/foo.txt
outputdir/sub/bar.txt
```

24.18. 使用扁平化

如果要将文件存储在同一目录中的 outputdir 目录中，请忽略源目录布局（例如扁平化路径），只需在文件制作者侧添加 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
```

24.19. 从目录和默认移动操作读取

默认情况下，Camel 将任何处理的文件移到文件使用的目录中的 .camel 子目录中。

```
from("file://inputdir/?recursive=true&delete=true").to("file://outputdir")
```

按如下所示影响布局：
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
```

24.20. 从目录读取并处理 JAVA 中的消息

```
from("file://inputdir/").process(new Processor() {
  public void process(Exchange exchange) throws Exception {
    Object body = exchange.getIn().getBody();
    // do some business logic with the input body
  }
});
```

正文将是一个 File 对象，它指向刚刚放入到 inputdir 目录中的文件。

24.21. 写入文件

Camel 也能够编写文件，即生成文件。在以下示例中，我们将在将进程写入目录前收到有关 SEDA 队列的一些报告。

24.21.1. 使用 Exchange.FILE_NAME写入子目录

通过使用单个路由，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 header 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来将文件写入多个目的地。

24.21.2. 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是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，并在进行数据传输后，将以原子方式移到' /var/myapp/finalDirectory 'directory。

```
from("direct:start").
  to("file:///var/myapp/finalDirectory?tempPrefix=/../filesInProgress/");
```

24.22. 将表达式用于文件名

在这个示例中，我们希望将消耗的文件移到备份文件夹中，使用当前的日期作为子文件夹名称：

```
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。

24.23. 避免多次读取同一文件（隐藏消费者）

Camel 直接在组件内支持 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键，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，将名称和文件大小用作键

```
<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，它使用最近保留 1000 个条目的缓存。您可以使用值中的 explicit Repository 选项来插件您自己的存储实现，以指示它在 Registry 中引用具有指定 id 的 Registry 中的 bean。

```
<!-- 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 跳过文件，则 Camel 将在 DEBUG 级别记录，因为它之前已被消耗：

```
DEBUG FileConsumer is idempotent and the file has been consumed before. Will skip this
file: target\idempotent\report.txt
```

24.24. 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，而不是默认基于内存的内存中。
此仓库使用第一级缓存以避免读取文件存储库。它将仅使用文件存储库来存储第一级别缓存的内容。因此，存储库可以在服务器重启后保留。它将在启动时将文件的内容加载到第一级缓存中。文件结构非常简单，因为它将密钥存储在文件中的单独行中。默认情况下，文件存储的大小限制为 1mb。当文件增长较大的 Camel 会截断文件存储时，通过将 1st 级别缓存刷新到一个新的空文件来重建内容。

我们使用 Spring XML 创建我们的存储库来创建我们的文件幂等存储库，并定义我们的文件消费者使用我们的存储库与 幂等 权限一起使用，以表示 Registry 查找：

24.25. 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，而不是默认使用的内存中。

首先，我们需要在 META-INF/persistence.xml 中有一个 persistence-unit，其中需要使用类 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，也可以在 spring XML 文件中创建 JPA 幂等存储库：

```
<!-- 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，这是的，只需要使用 sVirt 语法选项引用文件消费者端点中的 jpaStore bean：

```
<route>
  <from uri="file://inbox?idempotent=true&amp;idempotentRepository=#jpaStore"/>
  <to uri="bean:processInbox"/>
</route>
```

24.26. 使用 ORG.APACHE.CAMEL.COMPONENT.FILE.GENERICFILEFILTER 进行过滤

Camel 支持可插拔过滤策略。然后，您可以使用此类过滤器配置端点，以跳过被处理的某些文件。

在示例中，我们构建了自己的过滤器，它会跳过其文件名是以 skip 开始的文件：

然后，我们可以使用 filter 属性配置路由，以引用我们在 spring XML 文件中定义的过滤器（使用 spring 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>
```

24.27. 使用 ANT 路径匹配程序进行过滤

ANT 路径匹配程序基于 AntPathMatcher。

文件路径与以下规则匹配：

- ? 匹配一个字符
- DNAT 匹配零个或多个字符
- Tailoring 匹配路径中的零个或多个目录

antInclude 和 antExclude 选项可以轻松地指定 ANT 风格的 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：

24.27.1. 使用 Comparator 进行排序

Camel 支持可插拔排序策略。此策略在 Java 中的 java.util.Comparator 中使用构建。然后，您可以使用这样的比较器配置端点，并在处理前对文件进行排序。

在示例中，我们构建了自己的比较器，只需按照文件名排序：

然后，我们可以使用 sorter 选项配置路由，以引用我们在 spring XML 文件中定义的排序器 (mySorter)：

```
<!-- 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 sorter=114mySorter 将指示 Camel 在 Registry 中查找 ID 为 mySorter 的 bean。

24.27.2. 使用 sortBy 排序

Camel 支持可插拔排序策略。此策略使用 文件 语言来配置排序。sortBy 选项配置如下：

```
sortBy=group 1;group 2;group 3;...
```

其中，每个组都用分号分开。在简单的情形中，您只使用一个组，因此一个简单的示例可以是：

```
sortBy=file:name
```

这将根据文件名排序，您可以通过前缀 reverseing reverse: 到组来撤销顺序，因此排序现在是 Z..A：

```
sortBy=reverse:file:name
```

因为我们拥有完整的 文件 语言功能，我们可以使用一些额外的参数，因此如果我们希望根据文件大小排序：

```
sortBy=file:length
```

您可以将配置为忽略问题单，使用 ignoreCase: 进行字符串比较，因此如果您想要使用文件排序但忽略这种情况，则我们这样做：

```
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，我们希望按上次修改的文件排序，因此我们这样做：

```
sortBy=file:modified
```

然后，我们希望将名称作为 2 个选项进行分组，以便具有相同修改的文件按照名称排序：

```
sortBy=file:modified;file:name
```

现在，此处存在一个问题，您可以发现它吗？对文件的修改的时间戳太正常，但如果我们只想根据日期排序，那么根据名称对子组进行排序，那么什么？
另外，我们拥有 文件语言的真实 功能，我们还可以使用其支持模式的 date 命令。因此，您可以通过以下方法解决：

```
sortBy=date:file:yyyyMMdd;file:name
```

Yeah 是非常强大的，根据您还可为每个组使用反向方式，因此我们可以撤销文件名：

```
sortBy=date:file:yyyyMMdd;reverse:file:name
```

24.28. 使用 GENERICFILEPROCESSSTRATEGY

选项 processStrategy 可用于使用自定义 GenericFileProcessStrategy，允许您实现自己的 *开始、提交 和回滚* 逻辑。
例如，假设系统在应使用的文件夹中写入文件。但是，在写另一个 *就绪* 文件之前，您不应该开始使用文件。

因此，通过实施自己的 GenericFileProcessStrategy，我们可以根据以下方法实现它：

- 在 start （） 方法中，我们可以测试是否存在特殊的 *就绪* 文件。start 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，我们可以移动实际的文件，同时删除 *可用的* 文件。

24.29. 使用过滤器

filter 选项允许您通过实施 org.apache.camel.component.file.GenericFileFilter 接口在 Java 代码中

实施自定义过滤器。此接口具有返回布尔值的 accept 方法。返回 true 以包含该文件，使用 false 来跳过该文件。GenericFile 有 isDirectory 方法，无论是目录。这可让您过滤不需要的目录，以避免遍历不需要的目录。

例如，要跳过名称中以 "skip" 开头的任何目录，如下所示：

24.30. 使用 BRIDGEERRORHANDLER

如果要使用 Camel Error Handler 处理文件消费者中发生的任何异常，您可以启用 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，您要做的只是启用此选项，路由中的错误处理程序将从那里获取它。

重要

当使用 bridgeErrorHandler 时，在使用 bridgeErrorHandler 时，OnCompletions 不会被应用。Exchange 由 Camel Error Handler 直接处理，不允许之前的操作（如拦截器）执行操作。

24.31. 调试日志记录

此组件具有日志级别 TRACE，如果您出现问题，这非常有用。

24.32. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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
camel.cluster.file.acquire-lock-interval	尝试获取锁定之间等待的时间。		字符串
camel.cluster.file.attributes	自定义服务属性。		Map
camel.cluster.file.enabled	设置是否应启用集群集群服务，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	根路径。		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

Name	描述	默认值	类型
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 25 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。

当从远程 FTP 服务器消耗时，请确保在进一步 *消耗文件时读取名为 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 的所有前导斜杠来将绝对路径转换为相对。日志中会打印 WARN 消息。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-ftp</artifactId>
  <version>{CamelSBVersion}</version>See the documentation of the Apache Commons
  <!-- use the same version as your Camel core version -->
</dependency>
```

25.1. URI 格式

```
ftp://[username@]hostname[:port]/directoryname[?options]
sftp://[username@]hostname[:port]/directoryname[?options]
ftps://[username@]hostname[:port]/directoryname[?options]
```

其中 directoryname 代表底层目录。目录名称是相对路径。不支持 绝对路径。相对路径可以包含嵌套文件夹，如 /inbox/us。

支持 autoCreate 选项。当消费者启动时，在轮询前执行额外的 FTP 操作来创建端点配置的目录。autoCreate 的默认值为 true。

如果没有提供 用户名，则尝试 使用任何密码的匿名 登录。
如果没有提供 端口号，Camel 将根据协议提供默认值(ftp = 21, sftp = 22, ftps = 2222)。

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

这个组件使用两个不同的库进行实际 FTP 工作。FTP 和 FTPS 使用 Apache Commons Net，而 SFTP 使用 JCraft JSCH。

FTPS （也称为 FTP 安全）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。

25.2. 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

25.2.1. 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。

25.2.2. 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。

25.3. 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

25.4. 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：

25.4.1. 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)	启动目录。		字符串

25.4.2. 查询参数(111 参数)

Name	描述	默认值	类型
二进制 (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编码。您可以在消费者上使用此功能，以指定文件的编码，它允许 Camel 在访问文件内容时加载文件内容。在编写文件时，您可以使用此选项指定写入该文件的 charset。请注意，当编写 Camel 文件时，可能需要将消息内容读取在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)	使用后是否断开与远程 FTP 服务器的连接。断开连接将仅断开当前与 FTP 服务器的连接。如果您有要停止的消费者，则需要停止 consumer/route。	false	布尔值
doneFileName (common)	producer ：如果提供，则 Camel 将在编写原始文件时写入第二次完成的文件。done 文件为空。这个选项配置要使用的文件名。您可以指定固定名称。或者，您可以使用动态占位符。done 文件将始终写在与原始文件相同的文件夹中。consumer ：如果提供，Camel 仅在文件存在时使用文件。这个选项配置要使用的文件名。您可以指定固定名称。或者，您可以使用动态占位符。done 文件始终预期在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 Expression （如文件语言）来动态设置文件名。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。（注意：标头本身也可以是 Expression）。表达式选项支持 String 和 Expression 类型。如果表达式是字符串类型，则始终使用 File Language 来评估。如果表达式是 Expression 类型，则会使用指定的 Expression 类型 - 例如，您可以使用 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 来使用现在使用的文件。producers 支持 CamelOverruleFileName 标头，其优先于任何现有 CamelFileName 标头；CamelOverruleFileName 是一个仅一次的标头，并便于避免临时存储 CamelFileName 并在以后恢复它。		字符串
passiveMode (common)	设置被动模式连接。默认为活动模式连接。	false	布尔值
分隔符 (common)	设置要使用的路径分隔符。unix = 使用 unix 风格的路径分隔符 Windows = 使用窗口风格路径分隔符 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)	在记录上传和下载操作的进度时，配置使用的时间间隔（以秒为单位）。这可用于在操作需要更长的时间时记录进度。	5	int
transferLoggingLevel (common)	配置日志记录在记录上传和下载操作进度时要使用的日志级别。 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执行详细（粒度）日志记录。	false	布尔值
fastExistsCheck (common (advanced))	如果将此选项设为 true，则 camel-ftp 将直接使用列表文件来检查是否存在该文件。由于某些 FTP 服务器可能不支持直接列出文件，如果选项为 false，则 camel-ftp 将使用旧方法列出目录并检查该文件是否存在。此选项还会影响 readLock=changed，以控制它是否执行快速检查来更新文件信息。如果 FTP 服务器有很多文件，则这可用于加快进程。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
delete (consumer)	如果为 true，则会在成功处理文件后删除该文件。	false	布尔值
moveFailed (consumer)	根据简单语言设置移动失败表达式。例如，要将文件移到 .error 子目录使用：.error。注意：将文件移到失败位置 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 ETL 类型要求。如果 noop=true，Camel 也设置幂等=true，以避免再次消耗相同的文件。	false	布尔值
preMove (consumer)	在处理前移动文件名的表达式（如文件语言）。例如，要将 in-progress 文件移到 order 目录中，将此值设置为 order。		字符串
preSort (consumer)	启用预排序后，消费者将在轮询过程中对文件和目录名称进行排序，这从文件系统检索。如果您需要按照排序的顺序对文件进行操作，您可能需要执行此操作。预排序是在消费者开始过滤前执行的，并接受由 Camel 处理的文件。这个选项是 default=false 表示禁用。	false	布尔值
recursion (consumer)	如果某个目录，也会在所有子目录中查找文件。	false	布尔值

Name	描述	默认值	类型
resumeDownload (consumer)	配置是否启用了恢复下载。这必须被 FTP 服务器支持（大多数 FTP 服务器都支持它）。此外，还必须配置本地WorkDirectory 选项，以便下载的文件存储在本地目录中，且必须启用选项二进制文件，这是支持恢复下载所必需的。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
streamDownload (consumer)	设置不使用本地工作目录时要使用的下载方法。如果设置为 true，则在读取时将远程文件流传输到路由。当设置为 false 时，远程文件会在发送到路由前被加载到内存中。如果启用这个选项，则必须设置 stepwise=false，因为无法同时启用这两个步骤。	false	布尔值
下载 (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，但消费者仍会触发具有文件详情的 Camel Exchange，如文件名、文件大小等。只是不下载该文件。	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))	如果目录解析器以绝对路径导致，则允许设置消费者如何处理路径中的子文件夹和文件。因此，有些 FTP 服务器可能会用绝对路径返回文件名，那么 FTP 组件需要通过将返回的路径转换为相对路径来应对这一点。	false	布尔值
ignoreFileNotFoundOrPermissionError (consumer (advanced))	是否忽略（尝试列出目录中的文件还是下载文件时），这些文件不存在还是因为权限错误。默认情况下，当某个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 允许忽略它。	false	布尔值

Name	描述	默认值	类型
inProgressRepository (consumer (advanced))	可插拔 in-progress 存储库 org.apache.camel.spi.IdempotentRepository。in-progress 存储库用于考虑当前正在使用的进程文件中。默认使用基于内存的存储库。		IdempotentRepository
localWorkDirectory (consumer (advanced))	消耗时，本地工作目录可用于直接将远程文件内容存储在本地文件中，以避免将内容加载到内存中。这很有用，如果您消耗非常大的远程文件，因此可以节省内存。		字符串
onCompletionExceptionHandler (consumer (advanced))	要使用自定义 org.apache.camel.spi.ExceptionHandler 来处理在消费者进行提交或回滚的完成过程中发生的任何抛出异常。默认实现将在 WARN 级别和忽略时记录任何异常。		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))	一个可插拔的 org.apache.camel.component.file.GenericFileProcessStrategy，允许您实现自己的 readLock 选项或类似。在消耗文件之前，也可以使用特殊条件，如存在特殊的就绪文件。如果设置了这个选项，则不会应用 readLock 选项。		GenericFileProcessStrategy
useList (consumer (advanced))	在下载文件时，是否使用 LIST 命令。默认值为 true。在某些情况下，您可能想要下载特定的文件，且不允许使用 LIST 命令，因此您可以将这个选项设置为 false。请注意，使用这个选项时，要下载的具体文件不包括 meta-data 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)	如果文件已存在具有相同名称的文件，则该怎么办。覆盖（这是默认设置）替换现有文件。- Append - 将内容添加到现有文件中。- Fail - 抛出一个 GenericFileOperationException，表示已有的文件。- Ignore - 静默忽略问题，且不会覆盖现有的文件，但假设所有内容正常。- Move - 选项需要使用 moveExisting 选项。选项 eagerDeleteTargetFile 可以用来控制移动文件时要执行的操作，并且已存在现有文件，否则会导致移动操作失败。Move 选项将在编写目标文件前移动任何现有文件。- 只有在使用 tempFileName 选项时才适用 TryRename。这允许尝试将文件从临时名称重命名为实际名称，而无需执行任何存在检查。对于某些文件系统，这个检查可能会更快，特别是 FTP 服务器。 Enum 值： • 覆盖 • 附加 • Fail • Ignore • Move • TryRename	覆盖	GenericFileExist
flatten (producer)	扁平化用于扁平化任何前导路径的文件名路径，因此它只是文件名。这样，您可以递归地将文件写入子目录中，但当您将文件写入另一个目录中时，它们将写入到单个目录中。在制作者上将其设置为 true 可强制在 CamelFileName 标头中为任何前导路径去除任何文件名。	false	布尔值
jailStartingDirectory (producer)	仅用于 jailing （限制）将文件写入启动目录（和子）。默认情况下，这不允许 Camel 将文件写入外部目录（其开箱即用）。您可以将其关闭，以允许将文件写入启动目录之外的目录，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)	配置 fileExist=Move 时使用的表达式（如文件语言）用于计算文件名。将文件移到备份子目录中，只需输入备份。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至基于当前 dir 作为基础的相对目录中。		字符串
tempFileName (producer)	与 tempPrefix 选项相同，但提供对临时文件名命名更加精细的控制，因为它使用 File 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 包含一个目录前缀：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，并避免消费者（不使用专用读锁定）读取进度文件中。FTP 通常在上传大型文件时使用。		字符串
allowNullBody (producer (advanced))	用于指定在文件写入过程中是否允许 null 正文。如果设置为 true，则会创建一个空文件，设置为 false，并尝试向文件组件发送 null 正文，将抛出 'Cannot write null body 的 GenericFileWriteException。'如果 fileExist 选项设为 '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))	Batch 上传完成后是否与远程 FTP 服务器断开连接。disconnectOnBatchComplete 只会断开当前与 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。您可以使用它来禁用（将其设置为 false）在写入临时文件前删除目标文件。例如，您可以编写大型文件，并希望在编写临时文件过程中存在目标文件。这样可确保目标文件仅在最后一次时间之前被删除，只需在临时文件重命名为目标文件名之前。此选项还用于控制在启用 fileExist=Move 时是否删除任何现有文件，并且存在现有的文件。如果此选项 copyAndDeleteOnRenameFails false，那么如果现有文件存在，则会抛出异常（如果为 true），则在移动操作前删除现有文件。	true	布尔值

Name	描述	默认值	类型
keepLastModified (producer (advanced))	将保留源文件的最后修改的时间戳（若有）。将使用 Exchange.FILE_LAST_MODIFIED 标头来定位时间戳。此标头可以包含 java.util.Date 或带有时间戳的长度。如果时间戳存在，并且启用了 选项，它将在写入的文件中设置此时间戳。注：此选项仅适用于文件制作者。您不能将此选项与任何 ftp producer 一起使用。	false	布尔值
moveExistingFileStrategy (producer (advanced))	用于在配置 fileExist=Move 时使用特殊命名令牌的策略（自定义策略）。默认情况下，如果没有提供自定义策略，则使用实施。		FileMoveExistingStrategy
sendNoop (producer (advanced))	在将文件上传到 FTP 服务器前，是否将 noop 命令作为预写检查发送。这默认是启用的，因为连接的验证仍有效，这样可以静默地重新连接可以上传该文件。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)	在活跃模式下设置客户端端口范围。语法为：minPort-maxPort Both 端口号，包含 10000-19999，使其包含所有 1xxxx 端口。		字符串
auto create (advanced)	在文件的路径中自动创建缺少的目录。对于文件消费者，这意味着创建启动目录。对于文件制作者，这意味着应将文件写入的目录。	true	布尔值
BufferSize (advanced)	用于编写文件的缓冲区大小（或者 FTP 用于下载和上传文件）。	131072	int
connectTimeout (advanced)	设置等待连接建立由 FTPClient 和 JSCH 使用的连接超时。	10000	int
ftpClient (advanced)	使用 FTPClient 的自定义实例。		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 提供额外的参数。		Map
ftpClientParameters (advanced)	FtpComponent 用来为 FTPClient 提供额外的参数。		Map
maximumReconnectAttempts (advanced)	指定当 Camel 尝试连接到远程 FTP 服务器时的最大重新连接尝试。使用 0 禁用此行为。		int

Name	描述	默认值	类型
reconnectDelay (advanced)	在执行重新连接尝试前，millis Camel 将等待的时间。	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。建议将其设置为 300000，因为没有挂起的连接。在 SFTP 上，此选项被设置为 JSCH Session 实例上的超时。	300000	int
步骤(高级)	设置在下载文件时，还是在将文件上传到目录时，是否应先更改目录结构。例如，当您因为安全原因而无法更改 FTP 服务器上的目录时，您可以禁用此设置。步骤不能与 streamDownload 一起使用。	true	布尔值
Sync (advanced)	设置是否应严格使用同步处理。	false	布尔值
throwExceptionOnConnectFailed (advanced)	如果连接失败(dhausted) By 默认异常，则不会抛出异常，并记录 WARN。您可以使用它来启用异常，并处理 org.apache.camel.spi.PollingConsumerPollStrategy 回滚方法的抛出异常。	false	布尔值
timeout (advanced)	为等待 FTPClient 使用的回复设置数据超时。	30000	int
antExclude (filter)	ant 风格的过滤器排除。如果同时使用 antInclude 和 antExclude，则 antExclude 优先于 antInclude。可以使用以逗号分隔的格式指定多个排除项。		字符串
antFilterCaseSensitive (filter)	在 ant 过滤器上设置区分大小写标志。	true	布尔值
antInclude (filter)	ant 风格的过滤器包括。可使用以逗号分隔的格式指定多个 include。		字符串
eagerMaxMessagesPerPoll (filter)	允许控制 maxMessagesPerPoll 的限制是 eager。如果 eager，则在扫描文件的过程中会进行限制。其中为 false 将扫描所有文件，然后执行排序。将这个选项设置为 false 允许首先排序所有文件，然后限制轮询。请注意，这需要更高的内存用量，因为所有文件详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）（匹配不区分大小写）。请注意，如果您使用加号等符号，如果将其配置为端点 uri，则需要使用 RAW（）语法进行配置。有关配置端点 uri 的更多信息，请参阅。		字符串
excludeExt (filter)	用于排除与文件扩展名名称匹配的文件（不区分大小写）。例如，要排除 bak 文件，则使用 excludeExt=bak。可以使用逗号分隔多个扩展，例如排除 bak 和 dat 文件，使用 excludeExt=bak,dat。请注意，文件扩展名包含所有部分，例如，有一个名为 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 accept （）方法中返回 false，则会跳过文件。		GenericFileFilter
filterDirectory (filter)	根据简单语言过滤目录。例如，要过滤当前日期，您可以使用简单的日期模式，如 $\\{date:now: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 $\\{file:size} 5000。		字符串
幂等 （过滤）	使用 Idempotent Consumer EIP 模式的选项，以便 Camel 跳过已经处理的文件。默认情况下，将使用基于内存的 LRUCache，其中包含 1000 个条目。如果 noop=true 也会启用，以避免再次消耗相同的文件。	false	布尔值
idempotentKey (filter)	使用自定义幂等键，请执行以下操作：默认情况下，使用该文件的绝对路径。您可以使用 File Language，例如使用文件名和文件大小，您可以执行以下操作：OnlydKey=$\\{file:name}-$\\{file:size}。		字符串
idempotentRepository (filter)	可插拔存储库 org.apache.camel.spi.IdempotentRepository，它默认使用 MemoryIdempotentRepository （如果没有指定），且幂等性为 true。		IdempotentRepository
include (filter)	用于包含文件，如果文件名与正则表达式模式匹配（匹配不区分大小写）。请注意，如果您使用加号等符号，如果将其配置为端点 uri，则需要使用 RAW（）语法进行配置。有关配置端点 uri 的更多信息，请参阅。		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，则使用 includeExt=txt。可以使用逗号分隔多个扩展，例如包含 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，有一个名为 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)	定义每次轮询收集的最大消息。默认情况下不设置最大值。可用于设置限制，例如 1000，以避免在启动有数千个文件的服务器时避免。将值设为 0 或负数设置为禁用它。注意：如果使用这个选项，则文件和 FTP 组件将在任何排序前受到限制。例如，如果您有 100000 文件并使用 maxMessagesPerPoll=500，则仅获取前 500 个文件，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，然后对文件进行排序。		int
minDepth (filter)	递归处理目录时开始处理的最小深度。使用 minDepth=1 表示主目录。使用 minDepth=2 表示第一个子目录。		int
Move (filter)	在处理后移动文件名的表达式（如简单语言）。将文件移到 .done 子目录中，只需输入 .done。		字符串
exclusiveReadLockStrategy (lock)	可插拔读锁定作为 org.apache.camel.component.file.GenericFileExclusiveReadLockStrategy 实现。		GenericFileExclusiveReadLockStrategy

Name	描述	默认值	类型
readLock (lock)	消费者使用，仅在文件有独占的 read-lock 时轮询文件（例如，该文件不是 in-progress 或 in-progress 或正在写入）。Camel 将等到赋予文件锁定。此选项在策略中提供构建： - none - No read lock is use - markFile - Camel 创建标记文件 (fileName.camelLock)，然后在其上保存锁定。FTP 组件无法使用这个选项 - Changed - Changed 使用文件长度/修改时间戳来检测文件当前是否复制。至少将使用 1 sec 来确定这一点，因此此选项无法像其他进程一样快速使用文件，但可能会更可靠，因为 JDK IO API 并不总是确定当前由其他进程使用的文件。选项 readLockCheckInterval 可用于设置检查频率。 - fileLock - 使用 java.nio.channels.FileLock。这个选项不适用于 Windows OS 和 FTP 组件。在通过 mount/share 访问远程文件系统时，应该避免这种方法，除非该文件系统支持分布式文件锁定。- 重命名是使用尝试将文件重命名为测试（如果我们可以获得专用 read-lock. - 幂等的 - （仅用于文件组件）用于使用幂等结构作为读锁定。如果幂等存储库实现支持集群，这允许使用支持集群的读锁定。- idempotent-changed - （仅限文件组件）幂等性更改是使用幂等的 Repository，并作为组合的 read-lock 进行更改。如果幂等存储库实现支持集群，这允许使用支持集群的读锁定。- idempotent-rename - （仅用于文件组件）幂等名称使用幂等权限，并重命名为组合的 read-lock。如果幂等存储库实现支持集群，这允许使用支持集群的读锁定：各种读锁定并不是在集群模式下工作的，其中不同节点上的并发消费者对共享文件系统中的同一文件都是竞争的。使用一个接近原子操作的 markFile 来创建空标记文件，但无法保证在集群中工作。fileLock 可能更好，但文件系统需要支持分布式文件锁定等。如果幂等存储库支持集群，如 Hazelcast 组件或 Infinispan，则使用幂等读锁定支持集群。 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)	read-lock 的 millis （如果读锁定支持）。这个间隔用于在尝试获取读锁定之间休眠。例如，在使用更改的读锁定时，您可以为较慢的写入设置更高的间隔周期。如果生成者非常慢，则默认值 1 sec. 可能太快。注意：对于 FTP，默认的 readLockCheckInterval 为 5000。readLockTimeout 值必须大于 readLockCheckInterval，但 thumb 的规则是具有至少 2 个或大于 readLockCheckInterval 的超时。这需要确保读锁定进程允许冻结时间，以便在超时命中前尝试获取锁定。	1000	long
readLockDeleteOrphanLockFiles (lock)	如果 Camel 没有正确关闭，则是否应在启动时读取带有标记文件（比如 JVM 崩溃）删除任何孤立的读锁定文件（如果 Camel 没有被正确关闭）。如果将此选项设置为 false，则任何孤立的锁定文件将导致 Camel 不尝试获取该文件，这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。这个选项仅适用于 readLock 类型： changed, fileLock, idempotent, idempotent-changed, idempotent-rename, rename。 Enum 值： ● TRACE ● DEBUG ● INFO ● WARN ● ERROR ● OFF	DEBUG	LoggingLevel
readLockMarkerFile (lock)	是否将标记文件与更改、重命名或专用读取锁定类型一起使用。默认情况下，使用标记文件来保护其他进程获取同一文件。通过将这个选项设置为 false，可以关闭此行为。例如，如果您不想由 Camel 应用程序将标记文件写入文件系统。	true	布尔值
readLockMinAge (lock)	这个选项仅适用于 readLock=changed。它允许在尝试获取读取锁定前指定该文件的最短期限。例如，使用 readLockMinAge=300s 来要求文件持续 5 分钟。这可加快更改的读锁定速度，因为它将尝试获取至少给定时间的文件。	0	long

Name	描述	默认值	类型
readLockMinLength (lock)	这个选项仅适用于 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置为零，以允许消耗零长度文件。	1	long
readLockRemoveOnCommit (lock)	这个选项仅适用于 readLock=idempotent。它允许在处理文件成功并发生提交时指定是否从幂等存储库中删除文件名条目。默认情况下，该文件不会被删除，这样可确保不会发生任何竞争条件，因此另一个活跃节点可能会尝试获取该文件。相反，幂等存储库可以支持驱除策略，您可以在 X 分钟后驱除文件名条目 - 这样可确保出现竞争条件的问题。请参阅 readLockIdempotentReleaseDelay 选项的详情。	false	布尔值
readLockRemoveOnRollback (lock)	这个选项仅适用于 readLock=idempotent。它允许在处理文件失败时指定是否从幂等存储库中删除文件名条目，并且进行回滚。如果此选项为 false，则确认文件名条目（就像文件进行了提交一样）。	true	布尔值
readLockTimeout (lock)	可选的 timeout （如果 read-lock 支持）用于读锁定。如果无法授予 read-lock 并触发超时，则 Camel 将跳过该文件。下一次轮询 Camel 时，将再次尝试 文件，这一次可能会授予读锁定。使用 0 或较低值来指示永久值。当前 fileLock、change 和 rename 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 的规则是具有至少 2 个或大于 readLockCheckInterval 的超时。这需要确保读锁定进程允许冻结时间，以便在超时命中前尝试获取锁定。	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）	用于登录的帐户。		字符串
密码 （安全）	用于登录的密码。		字符串
用户名 （安全）	用于登录的用户名。		字符串
shuffle (sort)	影响文件列表（按随机顺序排列）。	false	布尔值
sortBy (sort)	使用文件语言内置排序。支持嵌套排序，以便您可以按文件名排序，并作为按修改日期排序的第二个组。		字符串
排序 者（排序）	可插拔排序器作为 java.util.Comparator 类。		比较器

25.5. FTPS 组件默认信任存储

当将与 SSL 相关的 ftpClient. 属性与 FTPS 组件搭配使用时，信任存储会接受所有证书。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 来配置信任存储。

在使用 sslContextParameters 时，信任存储由提供的 SSLContextParameters 实例的配置来管理。

您可以使用 ftpClient. 或 ftpClientConfig. 前缀直接在 URI 上配置 ftpClient 和 ftpClientConfig 上的附加选项。

例如，要将 FTPClient 上的 setDataTimeout 设置为 30 秒，您可以执行以下操作：

```
from("ftp://foo@myserver?password=secret&ftpClient.dataTimeout=30000").to("bean:foo");
```

您可以混合和匹配，并使用这两个前缀，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");
```

您可以根据需要拥有尽可能多的选项。

有关可能的选项以及更多详情，请参阅 Apache Commons FTPClientConfig 文档。另外，对于 Apache Commons FTPClient。

如果您不想在 url 中有多个和较长的配置，您可以通过让 Registry 中的 Camel 查找供 registry 引用要使用的 ftpClient 或 ftpClientConfig。

例如：

```
<bean id="myConfig" class="org.apache.commons.net.ftp.FTPClientConfig">
   <property name="lenientFutureDates" value="true"/>
   <property name="serverLanguageCode" value="fr"/>
</bean>
```

然后，当您在 url 中使用 sVirt 表示法时，请让 Camel 查找此 bean。

```
from("ftp://foo@myserver?password=secret&ftpClientConfig=#myConfig").to("bean:foo");
```

25.6. 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
```

25.7. 并发

FTP Consumer 不支持并发

FTP 使用者（具有相同端点）不支持并发（后备 FTP 客户端不安全）。
您可以使用多个 FTP 用户从不同的端点轮询。它只是不支持并发消费者的单一端点。

FTP 制作者 没有 这个问题，它支持并发。

25.8. 更多信息

这个组件是 File 组件的扩展。因此，File 组件页面中还有更多样本和详情。

25.9. 使用文件时的默认

FTP 使用者默认会将消耗的文件留在远程 FTP 服务器上。如果您希望删除文件或将它们移动到其他位置，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。

常规文件消费者与默认情况下，该文件将文件移到 .camel 子目录。对于 FTP 消费者，Camel 默认 不会 执行此操作的原因是，默认情况下可能会缺少权限来移动或删除文件。

25.9.1. 限制

选项 **readLock 可用于强制 Camel 不 消耗当前正在写入中的文件。但是，此选项默认关闭，因为它要求用户具有写入访问权限。有关读取锁定的详情，请查看 File2 中的选项表。**
其他解决方案可以避免当前通过 FTP 写入的文件；例如，您可以写入一个临时目的地并在文件被写入后移动该文件。

使用 move 或 preMove 选项移动文件时，文件仅限于 FTP_ROOT 文件夹。这可防止您在 FTP 区域外移动文件。如果要将文件移动到另一个区域，您可以使用软链接并将文件移到软链接文件夹中。

25.10. 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）。如果这不存在且没有表达式，则生成的消息 ID 将用作文件名。
CamelFileNameProduced	所写入的输出文件的实际文件路径（路径 + 名称）。此标头由 Camel 设置，其用途是为最终用户提供所写入文件的名称。
CamelFileNameConsumed	消耗的文件的文件名
CamelFileHost	远程主机名。
CamelFileLocalWorkPath	使用本地工作目录的路径（如果使用本地工作目录）。

此外，FTP/FTPS 使用者和制作者还将增强带有以下标头的 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

25.10.1. 交换属性

Camel 设置以下交换属性

标头	描述
CamelBatchIndex	当前索引在这个批处理中消耗的文件总数中。
CamelBatchSize	在这个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。

25.11. 关于超时

两组库（请参阅 top）有不同的 API 来设置超时。您可以使用这两者的 connectTimeout 选项在 millis 中设置超时来建立网络连接。也可以在 FTP/FTPS 上设置单独的 soTimeout，它与使用 ftpClient.soTimeout 对应。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。所有超时值都为 millis。

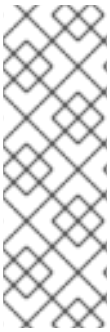
25.12. 使用本地工作目录

Camel 支持从远程 FTP 服务器消耗，并将文件直接下载到本地工作目录中。这可避免将整个远程文件内容读取在内存中，因为它使用 FileOutputStream 直接流传输到本地文件。

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 与扩展名。之后，文件被重命名为删除 .inprogress 后缀。最后，当 Exchange 完成后，将删除本地文件。

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

以上路由效率更高，因为它可避免将整个文件内容读取到内存中。它将直接将远程文件下载到本地文件流。java.io.File 处理随后用作交换正文。文件制作者利用此事实，可以直接用于工作文件 java.io.File handle，并对目标文件名执行 java.io.File.rename。由于 Camel 知道它是一个本地的工作文件，它可以优化并使用重命名而不是文件副本，因为工作文件旨在删除。

25.13. 步骤更改目录

在消耗文件（例如下载）或生成文件（例如上传）时，Camel FTP 可以在两个模式下运行。

- stepwise
- 没有步骤

您可能希望根据您的情况和安全问题选择一个。有些 Camel 最终用户只能在使用步骤时下载文件，而其他一些用户只能下载这些文件。

您可以使用 stepwise 选项来控制行为。

请注意，当大多数情况下，只有在用户仅限于主目录以及将主目录报告为 "/" 时，才会更改目录。

这两者之间的区别最好在示例中演示。假设我们需要在远程 FTP 服务器上有以下目录结构，我们需要遍历并下载文件：

```
/
/one
/one/two
/one/two/sub-a
/one/two/sub-b
```

我们在每个子 A (a.txt)和 sub-b (b.txt)文件夹中都有一个文件。

25.14. 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.
```

如在启用步骤时可以看到，它会遍历使用 CD xxx 的目录结构。

25.15. 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.
```

如您在不使用步骤时可以看到，根本不调用 CD 操作。

25.16. SAMPLES

在以下示例中，我们设置 Camel 以每小时(60 分钟)从 FTP 服务器下载所有报告作为 BINARY 内容，并将它存储为本地文件系统中的文件。

和使用 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>
```

25.16.1. 使用远程 FTPS 服务器（指示 SSL）和客户端身份验证

```
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");
```

25.16.2. 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");
```

25.17. 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，以在处理前跳过某些过滤器。

在示例中，我们构建了自己的过滤器，该过滤器仅接受文件名中以 report 开头的文件。

然后，我们可以使用 filter 属性配置路由，以引用我们在 spring XML 文件中定义的过滤器（使用 spring 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>
```

25.18. 使用 ANT 路径匹配程序进行过滤

ANT 路径匹配程序是在 camel-spring jar 中提供的开箱即用的过滤器。因此，如果您使用 Maven，则需要依赖于 camel-spring。
我们利用 Spring 的 AntPathMatcher 进行实际匹配的原因。

文件路径与以下规则匹配：

- ? 匹配一个字符
- DNAT 匹配零个或多个字符
- Tailoring 匹配路径中的零个或多个目录

以下示例演示了如何使用它：

25.19. 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。请参考 com.jcraft.jsch.Proxy 文档来发现所有选项。

25.20. 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，如果您希望 Camel 尝试使用私有/公共 SSH 密钥进行身份验证，并在没有公钥可用时回退到用户/密码身份验证，请使用以下路由配置：

```
from("sftp://localhost:9999/root?
username=admin&password=admin&preferredAuthentications=publickey,password").
  to("bean:processFile");
```

25.21. 使用固定名称消耗单个文件

当您想下载单个文件并知道文件名时，您可以使用 fileName=myFileName.txt 告知 Camel 要下载的文

件的名称。默认情况下，消费者仍会执行 FTP LIST 命令来执行目录列表，然后根据 fileName 选项过滤这些文件。虽然在这种情况下，可能需要通过设置 useList=false 来关闭目录列表。例如，用于登录到 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，然后提供要下载的文件的固定名称 fileName=myFileName.txt，然后 FTP 使用者仍然可以下载该文件。如果由于某种原因的文件不存在，则 Camel 默认抛出异常，您可以关闭此异常，并通过设置 ignoreFileNotFoundOrPermissionError=true 来忽略此文件。

例如，要有一个 Camel 路由来提取单个文件，并在使用后将其删除

```
from("ftp://admin@localhost:21/nolist/?
password=admin&stepwise=false&useList=false&ignoreFileNotFoundOrPermissionError=true
&fileName=report.txt&delete=true")
  .to("activemq:queue:report");
```

请注意，我们使用了上面讨论的所有选项。

您还可以将其用于 ConsumerTemplate。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);
```

25.22. 调试日志记录

此组件具有日志级别 TRACE，如果您出现问题，这非常有用。

25.23. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 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>
```

组件支持 13 个选项，如下所列。

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 26 章 GOOGLE BIGQUERY

从 Camel 2.20 开始

仅支持生成者。

Google Bigquery 组件通过 link:https://developers.google.com/api-client-library/java/apis/bigquery/v2 [Google Client Services API] 提供对 Cloud BigQuery Infrastructure 的访问。

当前实施不使用 gRPC。

当前实施不支持查询 BigQuery，它只是一个制作者。

将以下依赖项添加到此组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-google-bigquery</artifactId>
  <version>3.20.1.redhat-00050</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

26.1. 身份验证配置

Google BigQuery 组件身份验证主要用于 GCP 服务帐户。如需更多信息，请参阅 Google Cloud Platform Auth Guide。

Google 安全凭证可以通过提供到 GCP 凭证文件位置的路径来明确设置。

或者它们被隐式设置，其中连接工厂回退到 应用程序默认凭据。

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>";
```

或者设置环境变量 GOOGLE_APPLICATION_CREDENTIALS ：

```
export GOOGLE_APPLICATION_CREDENTIALS="/home/user/Downloads/my-key.json"
```

26.2. URI 格式

```
google-bigquery://project-id:datasetId[:tableId]?[options]
```

26.3. 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

26.3.1. 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。

26.3.2. 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。

26.4. 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，则使用默认值。		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

26.5. 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：

26.5.1. 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)	**所需的** BigQuery Dataset Id。		字符串
tableid (common)	BigQuery 表 ID。		字符串

26.5.2. 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，则使用默认值。		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)	使用 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

26.6. 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	插入数据时使用的表后缀。		字符串
CamelGoogleBigQueryTableId (producer) 常数：TABLE_ID	表 ID，数据将被提交。如果指定将覆盖端点配置。		字符串
CamelGoogleBigQueryInsertId (producer) 常量： INSERT_ID	插入数据时要使用的 InsertId。		字符串
CamelGoogleBigQueryPartitionDecorator (producer) 常量：PARTITION_DECORATOR	分区 decorator 来指示插入数据时要使用的分区。		字符串

26.7. 制作者端点

生产者端点可以接受并交付对 个人，并分组交换。组交换设置了 Exchange.GROUPED_EXCHANGE 属性集。

Google BigQuery producer 将在单个 api 调用中发送一个分组交换，除非指定了不同的表后缀或分区修饰符，这样可将其中断，以确保数据使用正确的后缀或分区 decorator 编写。

Google BigQuery 端点预期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，包含映射列表的有效负载将在列表中为每个条目插入一行。

26.8. 模板表

模板表可以使用 GoogleBigQueryConstants.TABLE_SUFFIX 标头来指定。例如，以下路由将创建表并插入每天分片的记录：

```
from("direct:start")
  .header(GoogleBigQueryConstants.TABLE_SUFFIX, "_${date:now:yyyyMMdd}")
  .to("google-bigquery:sampleDataset:sampleTable")
```

注意

建议您将此分区用于这个用例。

有关模板表的更多信息，请参阅 模板表。

26.9. 分区

在创建表时指定分区，如果设置的数据将自动分区到单独的表中。在插入数据时，可通过设置交换上的 GoogleBigQueryConstants.PARTITION_DECORATOR 标头来指定特定分区。

有关分区的更多信息，请参阅创建分区表。

26.10. 确保数据一致性

可以在带有标头 GoogleBigQueryConstants.INSERT_ID 的交换上设置插入 id，或者指定查询参数 useAsInsertId。因为插入 id 需要每行指定，当有效负载是一个列表时，将不会使用插入的交换标头。如果有效负载是一个列表，则将忽略 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

26.11. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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
camel.component.google-bigquery-sql.connection-factory	ConnectionFactory 获取与 Bladequery 服务的连接。如果没有提供默认值，则使用默认值。选项是一个 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 获取与 Bladequery 服务的连接。如果没有提供默认值，则使用默认值。选项是一个 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

Name	描述	默认值	类型
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
camel.component.google-bigquery.project-id	Google Cloud Project Id。		字符串

第 27 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 Infrastructure 的访问。

将以下依赖项添加到此组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-google-pubsub</artifactId>
  <version>3.20.1.redhat-00050</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

27.1. URI 格式

Google Pubsub 组件使用以下 URI 格式：

```
google-pubsub://project-id:destinationName?[options]
```

目的地名称可以是主题或订阅名称。

27.2. 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

27.2.1. 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。

27.2.2. 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。

27.3. 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)	与 PubSub 服务交互时使用凭证（在使用仿真器时不需要身份验证）。	true	布尔值
endpoint (common)	用于本地 Pub/Sub 模拟器的端点。		字符串

Name	描述	默认值	类型
serviceAccountKey (common)	Service account key，可用作 PubSub publisher/subscriber 的凭证。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)	要缓存的最大制作者数。如果您有许多不同主题的制作者，则可以增加。		int
publisherCacheTimeout (producer)	每个制作者在缓存中保持活跃的毫秒。		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)	允许生成者终止的毫秒数。		int

27.4. 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：

27.4.1. 路径参数(2 参数)

Name	描述	默认值	类型
projectID (common)	**需要** Google Cloud PubSub Project Id。		字符串
destinationName (common)	**必需** 目的地名称。对于消费者，这是订阅名称，而对于生成者而言，这是主题名称。		字符串

27.4.2. 查询参数(15 参数)

Name	描述	默认值	类型
Authentication (common)	与 PubSub 服务交互时使用凭证（在使用仿真器时不需要身份验证）。	true	布尔值
loggerId (common)	与父路由匹配的日志记录器 ID。		字符串
serviceAccountKey (common)	Service account key，可用作 PubSub publisher/subscriber 的凭证。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 = 交换在完成后被 ack'ed/nack'ed。NONE = 下游进程必须明确 ack/nack。 Enum 值： • AUTO • NONE	AUTO	AckMode
concurrentConsumers (consumer)	来自订阅的并行流数量。	1	整数
maxAckExtensionPeriod (consumer)	设置消息 ack deadline 将扩展的最大周期。值（以秒为单位）。	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)	同步拉取批处理消息。	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))	应该启用消息排序。	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 自定义 GooglePubsubSerializer，用于在制作者中序列化消息有效负载。		GooglePubsubSerializer

27.5. 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) 常量：MESSAGE_ID	服务器发布消息时分配的消息的 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) 常数：PUBLISH_TIME	发布消息的时间。		Timestamp
CamelGooglePubsubAttributes (common) 常量：ATTRIBUTES	消息的属性。		Map
CamelGooglePubsubOrderingKey (producer) 常量：ORDERING_KEY	如果非空，请识别应遵守发布订购的相关消息。		字符串

27.6. 制作者端点

制作者端点可以接受并传送到 PubSub 个人并分组交换。组交换设置了 Exchange.GROUPED_EXCHANGE 属性集。

Google PubSub 期望有效负载为 byte[] 数组，Producer 端点将发送：

- **字符串正文作为 byte[] 编码，格式为 UTF-8**

- byte[] body，如下所示
- 其他所有内容将序列化为 byte[] 数组

将 Map 设置为消息标头 GooglePubsubConstants.ATTRIBUTES 将作为 PubSub 属性发送。

Google PubSub 支持排序消息发送。

要启用此设置，将选项 messageOrderingEnabled 设置为 true，并将 pubsubEndpoint 设置为 GCP 区域。

生成消息时，设置消息标头 GooglePubsubConstants.ORDERING_KEY。这将设置为消息的 PubSub sortKey。

有关 订购消息 的更多信息。

交换传送到 PubSub 后，PubSub 消息 ID 将分配给标头 GooglePubsubConstants.MESSAGE_ID。

27.7. 消费者端点

如果在订阅上的配置选项内未确认，Google PubSub 将重新发送消息。

在交换处理完成后，组件将确认消息。

如果路由抛出异常，则交换将标记为失败，组件将 NACK 消息 - 它将立即重新连接。

要对消息进行 ack/nack 操作，组件使用 Acknowledgement ID 作为标头 GooglePubsubConstants.ACK_ID。如果删除了或篡改了标头，ack 将失败，并在 ack 截止时间后再次通知消息。

27.8. 消息正文

消费者端点将消息的内容返回为 byte[] - 与底层系统发送一样。它是转换/退出所有内容的路由。

27.9. 身份验证配置

默认情况下，此组件使用 GoogleCredentials.getApplicationDefault （） 获取凭证。通过将 authentication 选项设置为 false 来禁用此行为，在这种情况下，将在没有身份验证详情的情况下对 Google API 的请求进行。*这只在针对模拟器开发时才需要。可以通过提供服务帐户密钥文件的路径来更改此行为。*

27.10. 回滚和重新发送

Google PubSub 的回滚取决于 Acknowledgement Deadline 的概念 - Google PubSub 希望接收确认的时间周期。如果未收到确认，则会通知消息。

Google 提供了一个 API 来扩展消息的截止时间。

Google PubSub 文档.

因此，回滚实际上是一个带有零值的截止扩展 API 调用 - i.e. 期限现在就达到，消息可以恢复到下一个消费者。

通过将消息标头 GooglePubsubConstants.ACK_DEADLINE 设置为值（以秒为单位），可以将消息标头 GooglePubsubConstants.ACK_DEADLINE 设置为值来延迟消息重新发送。

27.11. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	与 PubSub 服务交互时使用凭证（在使用仿真器时不需要身份验证）。	true	布尔值
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	用于本地 Pub/Sub 模拟器的端点。		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	要缓存的最大制作者数。如果您有许多不同主题的制作者，则可以增加。		整数
camel.component.google-pubsub.publisher-cache-timeout	每个制作者在缓存中保持活跃的毫秒。		整数
camel.component.google-pubsub.publisher-termination-timeout	允许生成者终止的毫秒数。		整数

Name	描述	默认值	类型
camel.component.google-pubsub.service-account-key	Service account key，可用作 PubSub publisher/subscriber 的凭证。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.google-pubsub.synchronous-pull-retryable-codes	用于同步拉取的额外可重试错误代码列表。默认情况下，PubSub 客户端库重试 ABORTED、UNAVAILABLE、UNKNOWN。		字符串

第 28 章 HTTP

仅支持生成者

HTTP 组件提供基于 HTTP 的端点，用于调用外部 HTTP 资源（作为客户端使用 HTTP 调用外部服务器）。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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

28.1. URI 格式

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

将默认使用端口 80 作为 HTTP，443 用于 HTTPS。

28.2. 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

28.2.1. 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。

28.2.2. 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。

28.3. 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，这是仅内存的 Cookie 存储。请注意，如果 bridgeEndpoint=true，则 Cookie 存储被强制成为 noop cookie 存储，因为 Cookie 不应存储，因为我们只是桥接（例如作为代理）。		CookieStore
copyHeaders (producer)	如果此选项为 true，则 IN 交换标头将根据复制策略复制到 OUT 交换标头中。把它设置为 false，允许仅包含 HTTP 响应中的标头（不传播 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)	这种阈值（以字节为单位）控制响应有效负载是否应该以字节阵列形式存储在内存中，还是基于流。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，这可以避免为 JVM 垃圾收集器解析许多对象分配的开销。	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。如果没有 HTTP 标头需要的数据，这可以避免为 JVM 垃圾收集器解析许多对象分配的开销。	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。默认情况下关闭。如果您启用此功能，则 Java 会将传入的数据从请求反序列化到 Java，这可能会存在潜在的安全风险。	false	布尔值
authCachingDisabled (advanced)	禁用身份验证方案缓存。	false	布尔值
automaticRetriesDisabled (advanced)	禁用自动请求恢复和重新执行。	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。如果已经进行了配置，则这总是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)	连接实时的时间（时间单位为毫秒），默认值始终保持处于活动状态。		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，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 控制 Camel 消息和 HttpClient 之间的映射。		HttpBinding
httpClientConfigurer (advanced)	使用自定义 HttpClientConfigurer 执行要使用的 HttpClientConfigurer 的配置。		HttpClientConfigurer
httpConfiguration (advanced)	使用共享 HttpConfiguration 作为基础配置。		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 将标头过滤到或从 Camel 消息过滤。		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)	用于 NTML 的代理身份验证域（工作站名称）。		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 SSLContext 参数配置安全性：重要： 每个 HttpComponent 仅支持一个 org.apache.camel.support.jsse.SSLContextParameters 的实例。如果您需要使用 2 个或更多不同的实例，则需要定义您需要的每个实例的新 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。负值被解释为未定义（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。负值被解释为未定义（系统默认值）。	-1	int
socketTimeout (timeout)	以毫秒为单位定义套接字超时，这是等待数据的超时时间，不同，在两个连续数据包之间有最长不活跃周期。超时值为零被解释为无限超时。负值被解释为未定义（系统默认值）。	-1	int

28.4. 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：

28.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
httpUri (common)	**需要** 要调用的 HTTP 端点的 url。		URI

28.4.2. 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 的原始输入流是否缓存(Camel 会将流读取到内存/出口流到文件中，流缓存)缓存中。默认情况下，Camel 将缓存 Servlet 输入流，以支持多次读取它，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（如将其直接流传输到文件或其他持久性存储）时，您可以将此选项设置为 true。DefaultHttpBinding 将请求输入流复制到流缓存中，如果此选项为 false，则将其放在消息正文中，以支持多次读取流。如果您使用 Servlet 网桥/代理端点，请考虑启用此选项来提高性能，以防不需要多次读取消息有效负载。默认情况下，http producer 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，而是将响应流用作消息正文。	false	布尔值
HeaderFilterStrategy (common)	使用自定义 HeaderFilterStrategy 将标头过滤到或从 Camel 消息过滤。		HeaderFilterStrategy
httpBinding (common (advanced))	使用自定义 HttpBinding 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 进行请求。您还可以将 throwExceptionOnFailure 选项设置为 false，以便 HttpProducer 发送所有错误响应。	false	布尔值

Name	描述	默认值	类型
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 在过期时被删除。如果组件禁用了 Cookie 管理，这个选项也会禁用。	true	布尔值
connectionClose (producer)	指定是否必须将 Connection Close 标头添加到 HTTP 请求中。默认情况下，connectionClose 为 false。	false	布尔值
copyHeaders (producer)	如果此选项为 true，则 IN 交换标头将根据复制策略复制到 OUT 交换标头中。把它设置为 false，允许仅包含 HTTP 响应中的标头（不传播 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，将被忽略。当设置将覆盖从 url 中派生的主机标头。		字符串
httpMethod (producer)	配置要使用的 HTTP 方法。如果设置，HttpMethod 标头将无法覆盖这个选项。 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
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
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，在您希望下游服务器接收的 Host 标头来反映上游客户端调用的 URL，这将允许使用 Host 标头为代理服务生成准确的 URL。	false	布尔值
throwExceptionOnFailure (producer)	在从远程服务器的响应失败时禁用抛出 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和交换失败的处理，如果原因的例外在响应中以 application/x-java-serialized-object 内容类型发送回序列化。在制作者一侧，异常会被反序列化并抛出，而不是 HttpOperationFailedException。需要对原因的异常进行序列化。默认情况下关闭。如果您启用此功能，则 Java 会将传入的数据从请求反序列化到 Java，这可能会存在潜在的安全风险。	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，这是仅内存的 Cookie 存储。请注意，如果 bridgeEndpoint=true，则 Cookie 存储被强制成为 noop cookie 存储，因为 Cookie 不应存储，因为我们只是桥接（例如作为代理）。如果设置了 CookieHandler，则 Cookie 存储也会强制成为 noop cookie 存储，因为 Cookie 处理由 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，在一些个别情况下，用户可能需要包含消息正文。	false	布尔值
getWithBody (producer (advanced))	HTTP GET 是否应该包含消息正文。默认情况下，HTTP GET 不包含任何 HTTP 正文。但是，在一些个别情况下，用户可能需要包含消息正文。	false	布尔值
okStatusCodeRange (producer (advanced))	被视为成功响应的状态代码。该值包含。可以定义多个范围，用逗号分开，例如 200-204,209,301-304。每个范围都必须是一个数字或 from-to，其中包含横线。	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，这可以避免为 JVM 垃圾收集器解析许多对象分配的开销。	false	布尔值

Name	描述	默认值	类型
skipResponseHeaders (producer (advanced))	是否跳过将所有 HTTP 响应标头映射到 Camel 标头。如果没有 HTTP 标头需要的数据，这可以避免为 JVM 垃圾收集器解析许多对象分配的开销。	false	布尔值
userAgent (producer (advanced))	设置自定义 HTTP User-Agent 请求标头：		字符串
ClientBuilder (advanced)	提供对此端点的制作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 管理连接。		HttpClientConnectionManager
connectionsPerRoute (advanced)	每个路由的最大连接数。	20	int
httpClient (advanced)	设置制作者要使用的自定义 HttpClient。		HttpClient
httpClientConfigurer (advanced)	为生成者或消费者（如配置身份验证机制等）创建的新 HttpClient 实例注册自定义配置策略。		HttpClientConfigurer
httpClientOptions (advanced)	使用映射中的键/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)	用于 NTML 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)	用于 NTML 的代理身份验证域（工作站名称）。		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)	要使用的代理身份验证方案。 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)	允许用作以逗号分隔的值列表的身份验证方法 Basic、Digest 或 NTLM。		字符串

Name	描述	默认值	类型
authMethodPriority (security)	要优先使用哪个身份验证方法，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 SSLContext 参数配置安全性：重要： 每个 HttpComponent 仅支持一个 org.apache.camel.util.jsse.SSLContextParameters 的实例。如果您需要使用 2 个或更多不同的实例，则需要定义您需要的每个实例的新 HttpComponent。		SSLContextParameters
x509HostnameVerifier (security)	使用自定义 X509HostnameVerifier，如 DefaultHostnameVerifier 或 NoopHostnameVerifier。		HostnameVerifier

28.5. 消息标头

Name	类型	描述
Exchange.HTTP_URI	**字符串**	要调用的 URI。将覆盖直接在端点上设置的现有 URI。此 uri 是要调用的 http 服务器的 uri。它与 Camel 端点 uri 不同，您可以在其中配置端点选项，如安全性等。此标头不支持，它只支持 http 服务器的 uri。
Exchange.HTTP_PATH	**字符串**	请求 URI 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。对于 OK，为 200。
Exchange.HTTP_RESPONSE_TEXT	**字符串**	来自外部服务器的 HTTP 响应文本。

Name	类型	描述
Exchange.HTTP_CHARACTER_ENCODING	**字符串**	字符编码。
Exchange.CONTENT_TYPE	**字符串**	HTTP 内容类型。在 IN 和 OUT 消息上设置，以提供内容类型，如 **text/html**。
Exchange.CONTENT_ENCODING	**字符串**	HTTP 内容编码。在 IN 和 OUT 消息上设置，以提供内容编码，如 **gzip**。

28.6. 消息正文

Camel 将从外部服务器的 HTTP 响应存储在 OUT 正文上。IN 消息中的所有标头都会被复制到 OUT 消息，因此在路由过程中保留标头。此外，Camel 会将 HTTP 响应标头和 OUT 消息标头添加到 OUT 消息标头中。

28.7. 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*

28.8. 响应代码

Camel 将根据 HTTP 响应代码进行处理：

- *响应代码位于 100..299 之间，Camel 会将它视为成功响应。*
- *响应代码位于 300..399 之间，Camel 会将它视为重定向响应，并将抛出带有信息的 HttpOperationFailedException。*
- *响应代码为 400+，Camel 会将它视为外部服务器故障，并将抛出一个带有信息的 HttpOperationFailedException。*

throwExceptionOnFailure 选项 throwExceptionOnFailure 可以设置为 false，以防止 HttpOperationFailedException 抛出失败响应代码。这可让您从远程服务器获取任何响应。下面是一个示例。

28.9. 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，则以 java.lang.String 形式响应正文*

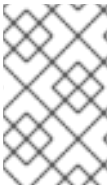
28.10. 使用哪个 HTTP 方法

以下算法用于决定应使用什么 HTTP 方法：
1。使用作为端点配置提供的方法(httpMethod)。
2.使用标头中提供的方法(Exchange.HTTP_METHOD)。
3.GET 如果标头中提供了查询字符串。
4.如果端点配置了查询字符串，则 GET。
5.如果有 要发送的数据（任何人不是 null）。
6.否则，GET。

28.11. 如何访问 HTTPSERVLETREQUEST 和 HTTPSERVLETRESPONSE

您可以使用 Camel 类型转换器系统访问这两个

```
HttpServletRequest request = exchange.getIn().getBody(HttpServletRequest.class);
HttpServletResponse response = exchange.getIn().getBody(HttpServletResponse.class);
```



注意

您可以在 camel-jetty 或 camel-cxf 端点后从处理器获取请求和响应。

28.12. 配置 URI 来调用

您可以直接组成端点 URI 设置 HTTP producer 的 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");
```

以及等效的 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键 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");
```

在上例中，Camel 会调用 http://newhost/，尽管端点配置了 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。

28.13. 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 或消息中使用键 Exchange.HTTP_QUERY 的标头设置。

```
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");
```

28.14. 如何将 HTTP 方法(GET/PATCH/POST/PUT/DELETE/HEAD/OPTIONS/TRACE)设置为 HTTP 生成者

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");
```

使用字符串常量可以编写方法稍短：

```
.setHeader("CamelHttpMethod", constant("POST"))
```

以及等效的 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>
```

28.15. 使用客户端超时 - SO_TIMEOUT

请参阅 HttpSOTimeoutTest 单元测试。

28.16. 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。

28.16.1. 使用 URI 之外的代理设置

为避免系统属性冲突，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 首先从 Java System 或 CamelContext 属性设置设置，然后设置端点代理选项（如果提供）。
因此，您可以使用端点选项覆盖系统属性。

另外，您还可以设置 http.proxyScheme 属性来明确配置要使用的方案。

28.17. 配置 CHARSET

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

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

28.17.1. 带有调度的轮询示例

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");
```

28.17.2. 来自端点 URI 的 URI 参数

在本例中，我们有完整的 URI 端点，它只是您在 Web 浏览器中输入的内容。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 不会在此处造成混淆。

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

28.17.3. 来自消息的 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 带有前缀 ?， 您可以像 & amp; char 一样分隔参数。

28.17.4. 获取响应代码

您可以通过使用 Exchange.HTTP_RESPONSE_CODE 从 Out 消息标头中获取 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);
```

28.18. 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
```

28.19. 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

28.20. 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。

28.20.1. 为 HTTP 客户端设置 SSL

使用 JSSE 配置工具

HTTP 组件通过 Camel JSSE 配置实用程序支持 SSL/TLS 配置。这个工具可显著减少您需要编写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 的端点配置

```
<camel:sslContextParameters
    id="sslContextParameters">
  <camel:keyManagers
      keyPassword="keyPassword">
    <camel:keyStore
```

```
        resource="/users/home/server/keystore.jks"
        password="keystorePassword"/>
    </camel:keyManagers>
  </camel:sslContextParameters>

  <to uri="https://127.0.0.1/mail/?sslContextParameters=#sslContextParameters"/>
```

直接配置 Apache HTTP 客户端

基本上 camel-http 组件基于 Apache HttpClient 构建。详情请参阅 SSL/TLS 自定义，或查看 org.apache.camel.component.http.HttpsServerTestSupport 单元测试基本类。
您还可以实施自定义 org.apache.camel.component.http.HttpClientConfigurer，以便在需要完全控制 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，并注册前一个密钥存储或信任存储的 https 协议。然后，从 camel 路由构建器类进行 hook：

```
HttpComponent httpComponent = getContext().getComponent("http", HttpComponent.class);
httpComponent.setHttpClientConfigurer(new MyHttpClientConfigurer());
```

如果使用 Spring DSL 进行此操作，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，并配置密钥存储和信任存储，它将可以正常工作。

使用 HTTPS 验证 getchas

最终用户报告他在使用 HTTPS 进行身份验证时存在问题。最终，通过提供自定义配置的 org.apache.http.protocol.HttpContext 来解决这个问题：

- 1.为 HttpContexts 创建(Spring)工厂：

```
public class HttpContextFactory {

  private String httpHost = "localhost";
  private String httpPort = 9001;

  private BasicHttpContext httpContext = new BasicHttpContext();
  private BasicAuthCache authCache = new BasicAuthCache();
  private BasicScheme basicAuth = new BasicScheme();

  public HttpContext getObject() {
    authCache.put(new HttpHost(httpHost, httpPort), basicAuth);

    httpContext.setAttribute(ClientContext.AUTH_CACHE, authCache);

    return httpContext;
  }

  // getter and setter
}
```

- 2.在 Spring 应用程序上下文文件中声明 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 SSLContext 参数

HTTP 组件只支持每个组件的一个 org.apache.camel.support.jsse.SSLContextParameters 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。在这里，我们有 2 个组件，各自使用自己的 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>
```

28.21. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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。默认情况下关闭。如果您启用此功能，则 Java 会将传入的数据从请求反序列化到 Java，这可能会存在潜在的安全风险。	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	禁用自动请求恢复和重新执行。	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。如果已经进行了配置，则这总是用于此组件创建的所有端点。选项是 org.apache.http.conn.HttpClientConnectionManager 类型。		HttpClientConnectionManager

Name	描述	默认值	类型
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。负值被解释为未定义（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。负值被解释为未定义（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
camel.component.http.connection-time-to-live	连接实时的时间（时间单位为毫秒），默认值始终保持处于活动状态。		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，这是仅内存的 Cookie 存储。请注意，如果 bridgeEndpoint=true，则 Cookie 存储被强制成为 noop cookie 存储，因为 Cookie 不应存储，因为我们只是桥接（例如作为代理）。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 交换标头将根据复制策略复制到 OUT 交换标头中。把它设置为 false，允许仅包含 HTTP 响应中的标头（不传播 IN 标头）。	true	布尔值
camel.component.http.default-user-agent-disabled	如果用户未提供，则禁用此构建器设置的默认用户代理。	false	布尔值

Name	描述	默认值	类型
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 将标头过滤到或从 Camel 消息过滤。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		HeaderFilterStrategy
camel.component.http.http-binding	使用自定义 HttpBinding 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.http.http-client-configurer	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	使用共享 HttpConfiguration 作为基础配置。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。选项是一个 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

Name	描述	默认值	类型
camel.component.http.proxy-auth-nt-host	用于 NTML 的代理身份验证域（工作站名称）。		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
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	这种阈值（以字节为单位）控制响应有效负载是否应该以字节阵列形式存储在内存中，还是基于流。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，这可以避免为 JVM 垃圾收集器解析许多对象分配的开销。	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。如果没有 HTTP 标头需要的数据，这可以避免为 JVM 垃圾收集器解析许多对象分配的开销。	false	布尔值
camel.component.http.socket-timeout	以毫秒为单位定义套接字超时，这是等待数据的超时时间，不同，在两个连续数据包之间有最长不活跃周期。超时值为零被解释为无限超时。负值被解释为未定义（系统默认值）。	-1	整数
camel.component.http.ssl-context-parameters	使用 SSLContext 参数配置安全性：重要： 每个 HttpComponent 仅支持一个 org.apache.camel.support.jsse.SSLContextParameters 的实例。如果您需要使用 2 个或更多不同的实例，则需要定义您需要的每个实例的新 HttpComponent。选项是 org.apache.camel.support.jsse.SSLContextParameters 类型。		SSLContextParameters

Name	描述	默认值	类型
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 29 章 INFINISPAN

支持生成者和消费者

此组件允许您使用 Hot Rod procol 与 Infinispan 分布式数据平面/缓存交互。Infinispan 是一个高度可扩展的、高度可用的键/值数据存储和使用 Java 编写的数据网格平台。

如果使用 Maven，您必须在 pom.xml 中添加以下依赖项：

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

29.1. URI 格式

```
infinispan://cacheName?[options]
```

制作者允许使用 HotRod 协议向远程缓存发送消息。消费者允许使用 HotRod 协议从远程缓存中侦听事件。

29.2. 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

29.2.1. 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。

29.2.2. 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。

29.3. 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)	定义我们是否连接到安全的 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，返回使用中的自定义监听程序。		InfinispanRemoteCustomListener
eventTypes (consumer)	指定要由 consumer.Multiple 事件注册的事件类型集合，可以使用逗号分隔。可能的事件类型有：CLIENT_CACHE_ENTRY_CREATED, CLIENT_CACHE_ENTRY_MODIFIED, CLIENT_CACHE_ENTRY_REMOVED, CLIENT_CACHE_ENTRY_EXPIRED, CLIENT_CACHE_FAILOVER。		字符串
defaultValue (producer)	为某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)	为某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 • QUERY • 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 SASL 机制来访问 infinispan 实例。		字符串
securityRealm (security)	定义访问 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** CacheContainer 配置。如果没有定义 cacheContainer，则使用。		Configuration
configurationProperties (advanced)	为 CacheManager 实施特定属性。		Map
ConfigurationUri (advanced)	CacheManager 的实施特定 URI。		字符串
标记 (advanced)	默认情况下，在每个缓存调用时应用以逗号分隔的 org.infinispan.client.hotrod.Flag 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而不是消息正文中。默认情况下，结果 == 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，则该值将用作存储查询结果的标头名称，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

29.4. 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：

29.4.1. 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 来使用当前配置的缓存管理器中的现有缓存名称。或者，将 default 用于默认缓存管理器名称。		字符串

29.4.2. 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)	定义我们是否连接到安全的 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，返回使用中的自定义监听程序。		InfinispanRemoteCustomListener
eventTypes (consumer)	指定要由 consumer.Multiple 事件注册的事件类型集合，可以使用逗号分隔。可能的事件类型有：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)	为某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)	为某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 • QUERY • 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 SASL 机制来访问 infinispan 实例。		字符串
securityRealm (security)	定义访问 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** CacheContainer 配置。如果没有定义 cacheContainer，则使用。		Configuration
configurationProperties (advanced)	为 CacheManager 实施特定属性。		Map
ConfigurationUri (advanced)	CacheManager 的实施特定 URI。		字符串
标记 (advanced)	默认情况下，在每个缓存调用时应用以逗号分隔的 org.infinispan.client.hotrod.Flag 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而不是消息正文中。默认情况下，结果 == 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，则该值将用作存储查询结果的标头名称，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

29.5. CAMEL OPERATIONS

本节列出所有可用的操作及其标头信息。

表 29.1. 表 1。Put 操作

操作名称	描述
InfinispanOperation.PUT	将键/值对放在缓存中，可选使用过期
InfinispanOperation.PUTASYNC	异步将键/值对放在缓存中，可选地放置过期

操作名称	描述
InfinispanOperation.PUTIFABSENT	如果不存在，在缓存中放置一个键/值对，可以选择使用过期
InfinispanOperation.PUTIFABSENTASYNC	如果一个键/值对（如果不存在）在缓存中异步放置一个键/值对，可以选择使用过期

- *所需的标头 ：*
  - ***CamelInfinispanKey***
  - ***CamelInfinispanValue***
- *可选标头 ：*
  - ***CamelInfinispanLifespanTime***
  - ***CamelInfinispanLifespanTimeUnit***
  - ***CamelInfinispanMaxIdleTime***
  - ***CamelInfinispanMaxIdleTimeUnit***
- *结果标头 ：*
  - ***CamelInfinispanOperationResult***

表 29.2. 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，可选过期

操作名称	描述
CamelInfinispanOperation.PUTALLASYNC	异步向缓存添加多个条目，可选使用过期

- *所需的标头：*
  - ***CamelInfinispanMap***
- *可选标头：*
  - ***CamelInfinispanLifespanTime***
  - ***CamelInfinispanLifespanTimeUnit***
  - ***CamelInfinispanMaxIdleTime***
  - ***CamelInfinispanMaxIdleTimeUnit***

表 29.3. 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 29.4. 表 4包含密钥操作

操作名称	描述
InfinispanOperation.CONTAINSKEY	确定缓存是否包含特定密钥

- ***所需的标头***
  - ***CamelInfinispanKey***
- ***结果标头***
  - ***CamelInfinispanOperationResult***

表 29.5. 表 5包含 Value 操作

操作名称	描述
InfinispanOperation.CONTAINSVALUE	确定缓存是否包含特定值

- ***所需的标头 ：***
  - ***CamelInfinispanKey***

表 29.6. 表 6。删除操作

操作名称	描述
InfinispanOperation.REMOVE	从缓存中删除条目，只有值与给定值匹配时（可选）
InfinispanOperation.REMOVEASYNC	异步从缓存中删除条目，只有值与给定值匹配时（可选）

- ***所需的标头 ：***
  - ***CamelInfinispanKey***

- *可选标头 ：*
  - ***CamelInfinispanValue***
- *结果标头 ：*
  - ***CamelInfinispanOperationResult***

表 29.7. 表 7。替换操作

操作名称	描述
InfinispanOperation.REPLACE	有条件地替换缓存中的条目，可选地替换为过期
InfinispanOperation.REPLACEASYNC	异步地替换缓存中的条目，可以选择使用过期

- *所需的标头 ：*
  - ***CamelInfinispanKey***
  - ***CamelInfinispanValue***
  - ***CamelInfinispanOldValue***
- *可选标头 ：*
  - ***CamelInfinispanLifespanTime***
  - ***CamelInfinispanLifespanTimeUnit***
  - ***CamelInfinispanMaxIdleTime***

- ***CamelInfinispanMaxIdleTimeUnit***

- *结果标头 ：*
  - ***CamelInfinispanOperationResult***

表 29.8. 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 29.9. 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 29.10. 表 10。stats Operation

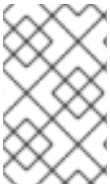
操作名称	描述
InfinispanOperation.STATS	返回有关缓存的统计信息

- *结果标头 ：*
  - ***CamelInfinispanOperationResult***

表 29.11. 表 11。查询操作

操作名称	描述
InfinispanOperation.QUERY	对缓存执行查询

- *所需的标头：*
  - ***CamelInfinispanQueryBuilder***
- *结果标头：*
  - ***CamelInfinispanOperationResult***



注意

写入方法，如 put (key, value)和 remove (key)默认不会返回之前的值。

29.6. 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 Map
CamelInfinispanKey	**null**	对象	共享	执行操作的密钥或生成事件的密钥。
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	缓存内某个值的 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	条目生命周期的时间范围。
CamelInfinispanMaxIdleTime	**null**	long	制作者	在将条目被视为过期前允许闲置的最长时间。
CamelInfinispanMaxIdleTimeUnit	**null**	字符串	制作者	条目 Max Idle Time 的时间单位。
CamelInfinispanQueryBuilder	null	InfinispanQueryBuilder	制作者	用于 QUERY 命令的 QueryBuilde，如果没有显示命令默认为 InifinispanConfiguration's。
CamelInfinispanOperationResultHeader	null	字符串	制作者	将操作结果存储在标头而不是消息正文中

29.7. 例子

- *将键/值放入命名缓存中：*

```
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键*
- *3 - 使用 registry 中配置的 cache manager 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，并从缓存中被驱除，例如：

```
from("direct:start")
   .setHeader(InfinispanConstants.OPERATION).constant(InfinispanOperation.GET)
```

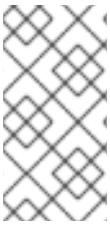
```
    .setHeader(InfinispanConstants.KEY).constant("123")
    .setHeader(InfinispanConstants.LIFESPAN_TIME).constant(100L) (1)

.setHeader(InfinispanConstants.LIFESPAN_TIME_UNIT.constant(TimeUnit.MILLISECO
NDS.toString()) (2)
    .to("infinispan:myCacheName");
```

其中,

- 1 - 设置条目的生命周期
- 2 - 为生命周期设置时间单位

查询

```
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 Data Grid 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 实例必须存在，Camel 应该可以从 Registry 中查找它。建议用户扩展 org.apache.camel.component.infinispan.remote.InfinispanRemoteCustomListener 类，并使用 @ClientListener 注解生成的类，该类可以在软件包 org.infinispan.client.hotrod.annotation 中找到。

29.8. 使用基于 INFINISPAN 的幂等存储库

在本小节中，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 - 设置仓库要使用的缓存名称
- 2 - 配置存储库 Bean
- 3 - 将存储库设置为路由

29.9. 使用基于 INFINISPAN 的聚合存储库

在本小节中，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*

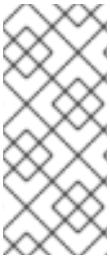
XML 示例

```
<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 - 设置仓库要使用的缓存名称
- 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。这对于消耗事件也至关重要。如需更多信息，请参阅官方 Infinispan 文档中的 Data Encoding 和 MediaTypes。

29.10. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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
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。选项是一个 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。选项是一个 org.infinispan.client.hotrod.configuration.Configuration 类型。		Configuration
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 实施特定属性。		Map
camel.component.infinispan.configuration-uri	CacheManager 的实施特定 URI。		字符串
camel.component.infinispan.custom-listener	如果提供，返回使用中的自定义监听程序。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

Name	描述	默认值	类型
camel.component.infinispan.event-types	指定要由 consumer.Multiple 事件注册的事件类型集合，可以使用逗号分隔。可能的事件类型有：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	默认情况下，在每个缓存调用时应用以逗号分隔的 org.infinispan.client.hotrod.Flag 列表。		字符串
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而不是消息正文中。默认情况下，结果 == 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，则该值将用作存储查询结果的标头名称，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.sasl-mechanism	定义 SASL 机制来访问 infinispan 实例。		字符串
camel.component.infinispan.secure	定义我们是否连接到安全的 Infinispan 实例。	false	布尔值
camel.component.infinispan.security-realm	定义访问 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
camel.component.infinispan.username	定义用于访问 infinispan 实例的用户名。		字符串

第 30 章 JIRA

支持生成者和消费者

JIRA 组件通过封装 Atlassian 的 REST Java Client for JIRA 来与 JIRA API 交互。它目前为新的问题和新评论提供轮询。它还能够创建新问题、添加评论、更改问题、添加/删除监视者、添加附加并转换问题的状态。

此端点依赖于简单的轮询，而不是 webhook。原因包括：

- 可靠性/稳定问题
- 我们轮询的有效负载类型通常并不大（另外，API 中提供了分页）
- 需要支持在 Webhook 失败时无法公开访问的应用程序

请注意，JIRA API 非常频繁地加快。因此，此组件可轻松扩展以提供额外的交互。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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

30.1. URI 格式

```
jira://type[?options]
```

JIRA 类型接受以下操作：

对于消费者：

- *newIssues：仅检索路由启动后的新问题*
- *newComments：仅检索路由启动后的新注释*
- *watchUpdates ：仅根据提供的 jql 检索更新的字段/发布*

对于制作者：

- *addIssue：添加一个问题*
- *addComment：在给定问题中添加注释*
- *attach ：在给定问题中添加附加*
- *deleteIssue：删除给定的问题*
- *updateIssue: 更新给定问题的字段*
- *ConvertIssue：转换给定问题的状态*
- *watchers：给定问题的添加/删除监视者*

因为 JIRA 完全自定义，您必须确保项目和工作流存在字段 ID，因为它们可以在不同的 JIRA 服务器之间有所变化。

30.2. 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

30.2.1. 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。

30.2.2. 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。

30.3. 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)	下一次轮询的时间（毫秒）。	6000	整数
jiraUrl (common)	**需要** 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
配置 （高级）	使用共享基础 jira 配置。		JiraConfiguration
accessToken (security)	（仅限OAuth）由 JIRA 服务器生成的访问令牌。		字符串
consumerKey (security)	（仅限OAuth）来自 JIRA 设置的消费者密钥。		字符串
密码 （安全）	（仅限基本身份验证）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）	（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 时使用。如果同时设置了 username 和 OAuth 令牌参数，则不要设置用户名和 OAuth 令牌参数，则用户名基本身份验证将具有优先权。		字符串
ValidationCode (security)	（仅限OAuth）从授权过程的第一个步骤中生成的 JIRA 验证代码。		字符串

30.4. 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：

30.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
type (common)	**需要** 执行的操作。消费者：newIssues、NewComments.producers: AddIssue, AttachFile, DeleteIssue, TransitionIssue, UpdateIssue, Watchers. 如需更多信息，请参阅本类 javadoc 描述。 Enum 值： • ADDCOMMENT • ADDISSUE • ATTACH • DELETEISSUE • NEWISSUES • NEWCOMMENTS • WATCHUPDATES • UPDATEISSUE • TRANSITIONISSUE • WATCHERS • ADDISSUELINK • ADDWORKLOG • FETCHISSUE • FETCHCOMMENTS		JiraType

30.4.2. 查询参数 (16 参数)

Name	描述	默认值	类型
delay (common)	下一次轮询的时间（毫秒）。	6000	整数
jiraUrl (common)	**需要** 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 的查询语言，允许您检索您想要的数据。例如，jql=project=MyProject where MyProject 是 JIRA 中的产品密钥。使用 RAW （）并设置 JQL 以防止 camel 解析它非常重要，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更改的字段的指标。默认情况下，使用者仅发送更改的字段。	true	布尔值
watchedFields (consumer)	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
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
accessToken (security)	（仅限OAuth）由 JIRA 服务器生成的访问令牌。		字符串
consumerKey (security)	（仅限OAuth）来自 JIRA 设置的消费者密钥。		字符串
密码 （安全）	（仅限基本身份验证）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）	（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 时使用。如果同时设置了 username 和 OAuth 令牌参数，则不要设置用户名和 OAuth 令牌参数，则用户名基本身份验证将具有优先权。		字符串
ValidationCode (security)	（仅限OAuth）从授权过程的第一个步骤中生成的 JIRA 验证代码。		字符串

30.5. 客户端工厂

您可以将 JIRARestClientFactory 与 registry 中名为 JIRARestClientFactory 的绑定，使其自动在 JIRA 端点中设置。

30.6. 身份验证

Camel-jira 支持 基本身份验证 和 OAuth 3 委托身份验证。

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。

30.6.1. 基本身份验证要求：

- *用户名和密码*

30.6.2. 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。
- 一个验证代码，由 JIRA 服务器生成。
- consumer 键，在 JIRA 服务器设置中设置。
- 由 JIRA 服务器生成的访问令牌。

30.7. JQL

JQL URI 选项都由两个使用者端点使用。理论上，"project key"等项目可以是 URI 选项本身。但是，通过要求使用 JQL，消费者变得更灵活且强大。

至少，消费者需要以下内容：

```
jira://[type]?[required options]&jql=project=[project key]
```

需要注意的是 newIssues 消费者将自动将 JQL 设置为：

- 将 ORDER BY 键 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，而不必对项目中的每一 个问题进行索引。

另一个备注是，newComments 消费者也必须索引项目中每个单个问题 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 插件在查询中包含一个 "Number of comment" 自定义字段，用于您的查询中的 '"Number of comments" > 0'。另外，还会尝试根据状态 (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)"*

30.8. 操作

请参阅在使用 JIRA 操作时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，则会抛出 IllegalArgumentException。

对于字段需要 id，如问题类型、优先级、转换。检查 jira 项目上的有效 id，因为它们可能会在 jira 安装和项目工作流上有所不同。

30.9. ADDISSUE

必需：

- *ProjectKey ：项目键，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 ：问题摘要。*

可选：

- *IssueAssignee ：分配用户*
- *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 ：带有有效组件名称的字符串列表。*

- *IssueWatchersAdd ：带有要添加到 watcher 列表中的用户名的字符串列表。*
- *IssueDescription ：问题的描述。*

30.10. ADDCOMMENT

必需：

- *IssueKey ：问题密钥标识符。*
- *交换的正文是描述。*

30.11. ATTACH

每个调用只能有一个文件。

必需：

- *IssueKey ：问题密钥标识符。*
- *交换的正文应该是 文件类型*

30.12. DELETEISSUE

必需：

- *IssueKey ：问题密钥标识符。*

30.13. TRANSITIONISSUE

必需：

- *IssueKey ：问题密钥标识符。*

- *IssueTransitionId ：问题转换 id。*
- *交换的正文是描述。*

30.14. UPDATEISSUE

- *IssueKey ：问题密钥标识符。*
- *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 ：问题摘要。*
- *IssueAssignee ：分配用户*
- *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 ：带有有效组件名称的字符串列表。*
- *IssueDescription ：问题的描述。*

30.15. WATCHER

- *IssueKey ：问题密钥标识符。*
- *IssueWatchersAdd ：带有要添加到 watcher 列表中的用户名的字符串列表。*
- *IssueWatchersRemove ：带有要从观察列表中删除的用户名的字符串列表。*

30.16. WATCHUPDATES (CONSUMER)

- watchedFields Comma 分隔的字段列表，以监视更改，如 Status、Priority、Assignee、Components 等。
- sendOnlyUpdatedField By 默认仅发送更改的字段作为正文。

所有消息还包含以下标头，用于添加有关更改的额外信息：

- issueKey ：更新的问题的密钥
- 已更改的: 更新字段的名称（例如 Status）
- watchedIssues: 列出更新时监视的所有问题密钥

30.17. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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）由 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	（仅限OAuth）来自 JIRA 设置的消费者密钥。		字符串
camel.component.jira.delay	下一次轮询的时间（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）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 时使用。如果同时设置了 username 和 OAuth 令牌参数，则不要设置用户名和 OAuth 令牌参数，则用户名基本身份验证将具有优先权。		字符串
camel.component.jira.verification-code	（仅限OAuth）从授权过程的第一个步骤中生成的 JIRA 验证代码。		字符串

第 31 章 JMS

支持生成者和消费者

此组件允许消息发送到（或从中消耗） JMS Queue 或 Topic。它使用 Spring 的 JMS 支持声明事务，包括用于发送的 Spring 的 JmsTemplate 和 MessageListenerContainer 以供使用。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

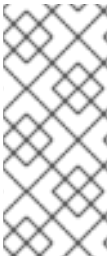
```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jms</artifactId>
  <version>{CamelSBVersion}</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

注意

使用 ActiveMQ，如果您使用 Apache ActiveMQ，您应该首选使用 ActiveMQ 组件，因为它已针对 ActiveMQ 进行了优化。此页面上的所有选项和示例也适用于 ActiveMQ 组件。

注意

如果您在使用 JMS 的事务时，请参阅以下的事务和缓存级别，因为它可能会影响性能。



注意

请求/通过 JMS，确保阅读本页面上 Request-reply 一节，以了解有关请求/回复的重要备注，因为 Camel 提供了很多选项来配置性能，以及集群环境。

31.1. URI 格式

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

其中 destinationName 是 JMS 队列或主题名称。默认情况下，destinationName 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
```

要连接到某个主题，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&…

31.1.1. 使用 ActiveMQ

JMS 组件重复使用 Spring 2 的 JmsTemplate 来发送消息。这不是在非J2EE 容器中使用的理想的选择，通常需要 JMS 供应商中的一些缓存以避免 性能不佳。

如果要使用 Apache ActiveMQ 作为您的消息代理，建议您进行以下操作之一：

- 使用 ActiveMQ 组件，该组件已优化为高效地使用 ActiveMQ
- 在 ActiveMQ 中使用 PoolingConnectionFactory。

31.1.2. 事务和缓存级别

如果您使用消息并使用事务(transacted=true)，缓存级别的默认设置可能会影响性能。

如果您使用 XA 事务，则无法缓存，因为它可能会导致 XA 事务无法正常工作。

如果您不使用 XA，那么您应该考虑缓存以加快性能，例如设置 cacheLevelName=CACHE_CONSUMER。

cacheLevelName 的默认设置是 CACHE_AUTO。这个默认自动检测模式，并相应地设置缓存级别：

- 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。

31.1.3. 可写订阅

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，且只能供整个网络中的单个 JMS 连接实例使用。您可能希望使用 Virtual Topics 来避免这个限制。此处 有关持久消息传递的更背景。

31.1.4. 消息标头映射

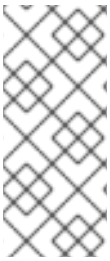
在使用消息标头时，JMS 规格指出标头名称必须是有效的 Java 标识符。因此，请尝试将您的标头命名为有效的 Java 标识符。执行此操作的一个优点是，您可以在 JMS Selector 中使用您的标头（其 SQL92 语法强制使用标头的 Java 标识符语法）。

默认使用映射标头名称的简单策略。策略是替换标头名称中的任何点和连字符，如下所示，并在标头名称从通过线发送的 JMS 消息中恢复时反向替换。这意味着什么？在 bean 组件上没有更多调用的方法名称，不再丢失 File 组件的文件名标头，以此类推。

接受 Camel 中的标头名称的当前标头名称策略如下：

- 点被 'DOT' 替换，当 Camel 使用消息时，替换将被撤销
- 连字符由 'HYPHEN' 替代，并在 Camel 使用消息时撤销替换

您可以在 JMS 端点上配置许多不同的属性，该端点映射到 JMSConfiguration 对象上的属性。



注意

映射到 Spring JMS 的许多属性映射到 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 文档来获取有关这些属性的更多信息。

31.2. 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

31.2.1. 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。

31.2.2. 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。

31.3. 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。它通常只适用于持久主题订阅。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。必须在组件或端点上配置连接工厂。		ConnectionFactory
disableReplyTo (common)	指定 Camel 是否忽略消息中的 JMSReplyTo 标头。如果为 true，Camel 不会向 JMSReplyTo 标头中指定的目的地发送回复。如果您希望 Camel 从路由中消耗，并且您不希望 Camel 自动发送回复消息，则可以使用此选项，因为代码中的另一个组件处理回复消息。如果要在不同的消息代理之间将 Camel 用作代理，并且希望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可配置订阅者名称。还必须配置 clientId 选项。		字符串
jmsMessageType (common)	允许您强制使用特定的 javax.jms.Message 实现来发送 JMS 消息。可能的值有：Bytes, Map, Object, Stream, text。默认情况下，Camel 会决定要从 In body 类型使用哪个 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（覆盖消费者中 Message.getJMSReplyTo （）的所有传入值）。		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 用户与 JMS 代理的有效连接。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 producer 也经过测试。	false	布尔值
acknowledgementModeName (consumer)	JMS 确认名称，其为： 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消息处于活跃状态时收到消息。通常，连接到队列的活动消费者以轮循方式从它接收消息。当使用消费者优先级时，如果有多个具有相同高优先级的活跃用户，则消息将进行循环发送。只有高优先级消费者没有可用的信用消息或高优先级消费者接受消息时，消息才会降低优先级较低的消费者（例如，它不符合与消费者关联的任何选择器的条件）。		int
asyncConsumer (consumer)	JmsConsumer 是否异步处理交换。如果启用，JmsConsumer 可以从 JMS 队列中提取下一个消息，而上一个消息则异步处理（通过 Asynchronous Routing Engine）。这意味着消息可能没有完全严格按照顺序进行处理。如果禁用（作为默认），则在 JmsConsumer 会从 JMS 队列中提取下一个消息前，会完全处理交换。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 必须支持 async 事务)。	false	布尔值
autoStartup (consumer)	指定消费者容器是否应自动启动。	true	布尔值
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)	根据底层 JMS 资源的名称设置缓存级别。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 消耗时的默认并发消费者数量（而不是通过 JMS 请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。当通过 JMS 执行 request/reply 时，选项 replyToConcurrentConsumers 用于控制回复消息监听器上的并发消费者数量。	1	int
maxConcurrentConsumers (consumer)	指定从 JMS 消耗时的最大并发消费者数（而不是通过 JMS 请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。当通过 JMS 执行请求/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交付进行回复。	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。可使用的 durable 订阅名称通过 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 域)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)	设置要创建的订阅名称。在带有共享或可升级订阅的主题（公共域）中应用。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： 每个订阅都只允许 1 个并发消费者（这是此消息侦听器容器的默认值），但一个共享订阅（需要 JMS 2.0）除外。		字符串
subscriptionShared (consumer)	设置是否共享订阅。可以使用的共享订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。请注意，共享的订阅也可能是危险的，因此此标志也可以与订阅相结合。仅在侦听主题(pub-sub 域)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，则可能会考虑启用此选项，同时仍然在队列中排队消息。如果此选项为 false，并且您停止 JMS 路由，则消息可能会被拒绝，并且 JMS 代理必须尝试 redeliveries （但可能会再次拒绝），最终消息可能会移到 JMS 代理上的死信队列中。要避免这种情况，建议启用这个选项。	false	布尔值
allowReplyManagerQuickStop (consumer (advanced))	是否启用请求管理器中使用的 DefaultMessageListenerContainer，允许 DefaultMessageListenerContainer.runningAllowed 标志在 JmsConfigurationVirtualMachineisAcceptMessagesWhileStopping 时快速停止，并且 org.apache.camel.CamelContext 当前已停止。在常规 JMS 用户中默认启用这种快速停止功能，但要为回复管理器启用这个标志。	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。消费者类型决定要使用的 Spring JMS 侦听器。默认将使用 org.springframework.jms.listener.DefaultMessageListenerContainer，Simple 将使用 org.springframework.jms.listener.SimpleMessageListenerContainer。指定 Custom 时，由 messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 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))	指定在 DefaultMessageListenerContainer 中使用哪些默认 TaskExecutor 类型，用于消费者端点和制作者端点的 ReplyTo consumer。可能的值： SimpleAsync（使用 Spring 的 SimpleAsyncTaskExecutor）或 ThreadPool （使用 Spring 的 ThreadPoolTaskExecutor 带有最佳值 - 缓存线程池）。如果没有设置，则默认为之前的行为，它将缓存线程池用于消费者端点，而 SimpleAsync 用于回复用户。建议使用 ThreadPool 来减少弹性配置中线程垃圾箱，同时动态增加和减少并发用户。 Enum 值： ● ThreadPool ● SimpleAsync		DefaultTaskExecutorType
eagerLoadingOfProperties (consumer (advanced))	加载消息时立即启用 JMS 属性和有效负载的 eager 加载，因为 JMS 属性可能并不是必需的，但有时可能会捕获与底层 JMS 提供程序和使用 JMS 属性的早期问题。另请参阅选项 eagerPoisonBody。	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)是 poison （无法读取/映射），然后将这个文本设置为消息正文，因此可以处理消息(poison 的原因)已作为交换异常保存。这可以通过设置 eagerPoisonBody=false 来关闭。另请参阅 eagerLoadingOfProperties 选项。	Poison JMS 消息，因为 $\{exception.message}	字符串

Name	描述	默认值	类型
exposeListenerSession (consumer (advanced))	指定在消耗消息时是否应公开监听程序会话。	false	布尔值
replyToSameDestinationAllowed (consumer (advanced))	JMS 使用者是否允许向消费者使用的同一目的地发送回复消息。这可防止出现无限循环，并通过消耗并向自己发送相同的消息。	false	布尔值
taskExecutor (consumer (advanced))	允许您指定自定义任务执行器以使用消息。		TaskExecutor
deliveryDelay (producer)	设置用于发送 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。可能的值是由 javax.jms.DeliveryMode 定义的值。NON_PERSISTENT = 1 和 PERSISTENT = 2。 Enum 值： ● 1 ● 2		整数
deliveryPersistent (producer)	指定默认使用持久性交付。	true	布尔值
explicitQosEnabled (producer)	设置在发送消息时使用 deliveryMode、priority 或 timeToLive 数量服务。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用到当前的端点。这与 preserveMessageQos 选项（按消息粒度运行）相反，从 Camel In 消息标头中读取 QoS 属性。	false	布尔值
formatDateHeadersToIso8601 (producer)	根据 ISO 8601 标准设置 JMS 日期属性是否应格式化。	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

Name	描述	默认值	类型
preserveMessageQos (producer)	如果要使用消息中指定的 QoS 设置来发送消息，而不是 JMS 端点上的 QoS 设置。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供全部或仅提供其中一些。如果没有提供，Camel 将回退到使用端点中的值。因此，在使用此选项时，标头会覆盖端点中的值。与之相反，clearQosEnabled 选项将仅使用端点上设置的选项，而不使用来自消息标头中的值。	false	布尔值
priority (producer)	大于 1 的值指定发送时的消息优先级（其中 1 是最低优先级，9 是最高）。必须启用 explicitQosEnabled 选项才能使此选项生效。 Enum 值： • 1 • 2 • 3 • 4 • 5 • 6 • 7 • 8 • 9	4	int
replyToConcurrentConsumers (producer)	指定执行请求/通过 JMS 回复时的默认并发消费者数量。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。	1	int
replyToMaxConcurrentConsumers (producer)	指定在通过 JMS 使用请求/回复时的最大并发消费者数。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。		int
replyToOnTimeoutMaxConcurrentConsumers (producer)	指定使用请求/通过 JMS 时超时时继续路由的最大并发消费者数。	1	int
replyToOverride (producer)	在 JMS 消息中提供显式 ReplyTo 目的地，这将覆盖 replyTo 的设置。如果要将消息转发到远程队列，并从 ReplyTo 目的地接收回复消息，这非常有用。		字符串

Name	描述	默认值	类型
replyToType (producer)	允许明确指定在执行 request/reply 时要用于 replyTo 队列的策略类型。可能的值有：Temporary、share 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的队列。如需了解更多详细信息，请参阅 Camel JMS 文档，特别是在集群环境中运行时的影响的备注，以及共享回复队列的性能低于其 alternatives Temporary 和 Exclusive。 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 来覆盖此端点配置的超时值，因此具有每个消息独立的超时值。另请参阅 requestTimeoutCheckerInterval 选项。	20000	long
timeToLive (producer)	发送消息时，指定消息的时间到时间（以毫秒为单位）。	-1	long
allowAdditionalHeaders (producer (advanced))	此选项用于允许其他标头，这些标头可能具有根据 JMS 规范无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 包含字节数组或其他无效类型的值来执行此操作。您可以用逗号指定多个标头名称，并用作通配符匹配的后缀。		字符串
allowNullBody (producer (advanced))	是否允许在没有正文的情况下发送消息。如果此选项为 false，且消息正文为 null，则会抛出 JMSException。	true	布尔值
alwaysCopyMessage (producer (advanced))	如果为 true，则 Camel 始终会在消息传递给发送的制作者时生成 JMS 消息副本。在某些情况下需要复制消息，如设置 replyToDestinationSelectorName 时（如果设置了 replyToDestinationSelectorName，则 Camel 会将 alwaysCopyMessage 选项设置为 true）。	false	布尔值
correlationProperty (producer (advanced))	当使用 InOut 交换模式时，使用此 JMS 属性而不是 JMSCorrelationID JMS 属性来关联消息。如果设置消息仅针对此属性 JMSCorrelationID 属性的值关联，则将忽略且未由 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时间。例如，当您通过 JMS 进行请求/回复时，Camel 默认使用 requestTimeout 值作为发送的消息的时间。问题是，发送者和接收器系统必须同步其时钟，因此它们正在同步。这并非始终易于归档。因此，您可以使用 disableTimeToLive=true 来不设置发送消息上的生存时间。然后，消息不会在接收器系统中过期。如需了解更多详细信息，请参见以下部分关于生存时间。	false	布尔值
forceSendOriginalMessage (producer (advanced))	当使用 mapJmsMessage=false Camel 时，如果在路由中涉及标头(get 或 set)，则会创建一个新的 JMS 消息来发送到新的 JMS 目的地。将此选项设置为 true，以强制 Camel 发送收到的原始 JMS 消息。	false	布尔值
includeSentJMSMessageID (producer (advanced))	仅在使用 InOnly 发送到 JMS 目的地时适用（例如触发和忘记）。启用此选项将增强 Camel Exchange 与 JMS 客户端在消息发送到 JMS 目的地时使用的实际 JMSMessageID。	false	布尔值
replyToCacheLevelName (producer (advanced))	在执行请求/通过 JMS 时，按名称为回复消费者设置缓存级别。这个选项仅在使用固定回复队列（而非临时）时才适用。Camel 默认将使用：CACHE_CONSUMER 用于专用或共享的 w/ replyToSelectorName。和 CACHE_SESSION 用于没有 replyToSelectorName 的共享。IBM WebSphere 等 JMS 代理可能需要设置 replyToCacheLevelName=CACHE_NONE 才能正常工作。注：如果使用临时队列，则不允许使用更高的值，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，以便您可以在使用共享队列（也就是说，如果您不使用临时回复队列）时过滤来自其他回复的回复。		字符串

Name	描述	默认值	类型
streamMessageTypeEnabled (producer (advanced))	设置 StreamMessage 类型是否已启用。流类型的消息有效负载（如 files、InStream 等）将通过作为 BytesMessage 或 StreamMessage 发送。此选项控制将使用哪种类型。默认情况下，使用 BytesMessage 来强制整个消息有效负载读取到内存中。通过启用此选项，消息有效负载以块的形式读取到内存中，然后每个块都会写入 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 自动发现 ConnectionFactory。如果只找到一个 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 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 时才适用。这要求对象可以序列化。Camel 将排除任何不可序列化的对象，并将它记录在 WARN 级别。	false	布尔值
artemisStreamingEnabled (advanced)	是否针对 Apache Artemis 流模式进行优化。这可减少使用带有 JMS StreamMessage 类型的 Artemis 时的内存开销。只有在使用 Apache Artemis 时，才必须启用这个选项。	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 无法获得连接到远程 JMS 代理的连接，那么在重试和/或故障切换时可能会阻断它。这将使 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 使用异步模式的专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果没有建立连接，则会在 WARN 级别记录异常，使用者将无法接收消息；然后，您可以重启路由来重试。	false	布尔值
asyncStopListener (advanced)	在停止路由时，是否异步停止 JmsConsumer 消息监听程序。	false	布尔值
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
配置 （高级）	使用共享的 JMS 配置。		JmsConfiguration

Name	描述	默认值	类型
destinationResolver (advanced)	一个可插拔的 org.springframework.jms.support.destination.DestinationResolver，供您使用自己的解析器（例如，在 JNDI registry 中查找实际目的地）。		DestinationResolver
errorHandler (advanced)	指定在处理消息时引发任何未发现异常时要调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置错误处理程序，则这些例外将在 WARN 级别记录。您可以配置日志记录级别，以及是否应使用 errorHandlerLoggingLevel 和 errorHandlerLogStack Trace 选项记录堆栈追踪。这样可以更容易配置，而不是需要对自定义错误处理程序进行代码。		ErrorHandler
exceptionListener (advanced)	指定针对任何底层 JMS 异常通知的 JMS Exception Listener。		ExceptionListener
idleConsumerLimit (advanced)	指定任何给定时间允许闲置的用户数量的限值。	1	int
idleTaskExecutionLimit (advanced)	指定接收任务闲置执行的限制，不会在其执行中收到任何消息。如果达到这个限制，任务将关闭并将接收给其他执行任务（在动态调度时；请参阅 maxConcurrentConsumers 设置）。Spring 提供了额外的文档。	1	int
includeAllJMSXProperties (advanced)	在从 JMS 到 Camel Message 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，则不会应用这个选项。	false	布尔值
jmsKeyFormatStrategy (advanced)	编码和解码 JMS 密钥的可插拔策略，以便它们符合 JMS 规范。Camel 提供两个开箱即用的实现：default 和 passthrough。默认策略将安全地使用句点和连字符 (. 和 -)。passthrough 策略将密钥保留原样。可用于不负责 JMS 标头密钥是否包含非法字符的 JMS 代理。您可以自行提供 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 sVirt 表示法引用它。 Enum 值： • default • passthrough		JmsKeyFormatStrategy
mapJmsMessage (advanced)	指定 Camel 是否应该自动将收到的 JMS 消息映射到合适的有效负载类型，如 javax.jms.TextMessage 到 String 等。	true	布尔值

Name	描述	默认值	类型
maxMessagesPerTask (advanced)	每个任务的消息数量。-1 代表没有限制。如果您为并发消费者使用范围（例如 min max），则可以使用此选项将值设为 100，以控制消费者在需要较少的工作时可以缩小的速度。	-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)	使用给定的 MessageCreatedStrategy，当 Camel 发送 JMS 消息时，Camel 创建 javax.jms.Message 对象的新实例。		MessageCreatedStrategy
messageIdEnabled (advanced)	发送时，指定是否应添加消息 ID。这只是对 JMS 代理的提示。如果 JMS 提供程序接受此提示，则这些消息必须将消息 ID 设置为 null；如果提供程序忽略提示，则必须将消息 ID 设置为其普通唯一值。	true	布尔值
messageListenerContainerFactory (advanced)	MessageListenerContainerFactory 的 registry ID，用于决定要使用消息的 org.springframework.jms.listener.AbstractMessageListenerContainer。设置此项将自动将 consumerType 设置为 Custom。		MessageListenerContainerFactory
messageTimestampEnabled (advanced)	指定在发送消息时是否默认启用时间戳。这只是对 JMS 代理的提示。如果 JMS 提供程序接受此提示，则这些消息必须将时间戳设置为零；如果提供程序忽略提示，则必须将时间戳设置为其正常值。	true	布尔值
pubSubNoLocal (advanced)	指定是否禁止自己连接发布的消息的发送。	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，即当连接被刷新时，以毫秒为单位。默认值为 5000 ms，即 5 秒。	5000	long
requestTimeoutCheckerInterval (advanced)	配置 Camel 在执行请求/通过 JMS 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，如果发生超时时，您必须更快地响应，那么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

Name	描述	默认值	类型
Sync (advanced)	设置是否应严格使用同步处理。	false	布尔值
transferException (advanced)	如果启用了且您使用 Request Reply messaging (InOut)，并且 Exchange 失败在消费者端，则原因例外将作为 javax.jms.ObjectMessage 发回的响应。如果客户端是 Camel，则返回的例外将重新箭头。这样，您可以使用 Camel JMS 作为路由中的桥接 - 例如，使用持久性队列来启用强大的路由。请注意，如果您也启用了 transferExchange，这个选项将具有优先权。caught 异常需要可以被序列化。消费者端的原始例外可以嵌套在外部异常中，如 org.apache.camel.RuntimeCamelException。请谨慎使用它，因为数据正在使用 Java 对象序列化，要求接收者在类级别反序列化数据，这会强制在生产者和消费者之间进行强校准。	false	布尔值
transferExchange (advanced)	您可以在有线线上传输交换，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 header, Exchange properties, exchange exception.这要求对象可以序列化。Camel 将排除任何不可序列化的对象，并将它记录在 WARN 级别。您必须在制作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。请谨慎使用它，因为数据正在使用 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，这会强制在需要使用兼容 Camel 版本的生产者和消费者之间进行强大的协调。	false	布尔值
useMessageIDAsCorrelationID (advanced)	指定 JMSMessageID 是否应该始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值
waitForProvisionCorrelationToBeUpdatedCounter (advanced)	在执行 request/reply over JMS 以及启用了 useMessageIDAsCorrelationID 时，等待 provisional correlation id 被更新到实际关联 ID 的次数。	50	int
waitForProvisionCorrelationToBeUpdatedThreadSleepingTime (advanced)	等待置备关联 ID 时每次休眠的时间间隔（以秒为单位）。	100	long
HeaderFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 将标头过滤到或从 Camel 消息过滤。		HeaderFilterStrategy

Name	描述	默认值	类型
errorHandlerLoggingLevel (logging)	允许为日志记录 uncaught 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)	允许通过默认错误处理程序来控制是否应记录 stacktraces。	true	布尔值
密码 （安全）	与 ConnectionFactory 一起使用的密码。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
用户名 （安全）	与 ConnectionFactory 一起使用的用户名。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
Transacted (transaction)	指定是否使用转换模式。	false	布尔值
TransactedInOut (transaction)	指定 InOut 操作（请求回复）是否默认使用 transacted 模式，如果此标志被设置为 true，则 Spring JmsTemplate 将把 sessionTransacted 设置为 true，而 confirmMode 作为转换用于 InOut 操作。请注意：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，容器对这些值做出自己的决策。类似地，这些参数不会考虑本地管理的事务，因为 Spring JMS 在这种情况下在现有 JMS 会话上运行。在受管事务外运行时，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（除 XA 事务之外）时同步的本地 JMS 事务。这与主事务一起管理本地 JMS 事务（可能是原生 JDBC 事务）的影响，在主事务后 JMS 事务提交右侧的 JMS 事务。	false	布尔值
lazyCreateTransactionManager (transaction (advanced))	如果为 true，Camel 将创建一个 JmsTransactionManager，如果没有在选项 transacted=true 时注入 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))	如果使用转换模式，事务的超时值（以秒为单位）。	-1	int

31.4. 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：

31.4.1. 路径参数(2 参数)

Name	描述	默认值	类型
destinationType (common)	要使用的目标种类。 Enum 值： • queue • topic • temp-queue • temp-topic	queue	字符串
destinationName (common)	用作目的地的队列或主题 **所需的** 名称。		字符串

31.4.2. 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。它通常只适用于持久主题订阅。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。必须在组件或端点上配置连接工厂。		ConnectionFactory
disableReplyTo (common)	指定 Camel 是否忽略消息中的 JMSReplyTo 标头。如果为 true，Camel 不会向 JMSReplyTo 标头中指定的目的地发送回复。如果您希望 Camel 从路由中消耗，并且您不希望 Camel 自动发送回复消息，则可以使用此选项，因为代码中的另一个组件处理回复消息。如果要在不同的消息代理之间将 Camel 用作代理，并且希望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可配置订阅者名称。还必须配置 clientId 选项。		字符串
jmsMessageType (common)	允许您强制使用特定的 javax.jms.Message 实现来发送 JMS 消息。可能的值有：Bytes, Map, Object, Stream, text。默认情况下，Camel 会决定要从 In body 类型使用哪个 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（覆盖消费者中 Message.getJMSReplyTo （）的所有传入值）。		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 用户与 JMS 代理的有效连接。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 producer 也经过测试。	false	布尔值

Name	描述	默认值	类型
acknowledgementModeName (consumer)	JMS 确认名称，其为：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消息处于活跃状态时收到消息。通常，连接到队列的活动消费者以轮循方式从它接收消息。当使用消费者优先级时，如果有多个具有相同高优先级的活跃用户，则消息将进行循环发送。只有高优先级消费者没有可用的信用消息或高优先级消费者接受消息时，消息才会降低优先级较低的消费者（例如，它不符合与消费者关联的任何选择器的条件）。		int
asyncConsumer (consumer)	JmsConsumer 是否异步处理交换。如果启用，JmsConsumer 可以从 JMS 队列中提取下一个消息，而上一个消息则异步处理（通过 Asynchronous Routing Engine）。这意味着消息可能没有完全严格按照顺序进行处理。如果禁用（作为默认），则在 JmsConsumer 会从 JMS 队列中提取下一个消息前，会完全处理交换。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 必须支持 async 事务)。	false	布尔值
autoStartup (consumer)	指定消费者容器是否应自动启动。	true	布尔值
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)	根据底层 JMS 资源的名称设置缓存级别。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 消耗时的默认并发消费者数量（而不是通过 JMS 请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。当通过 JMS 执行 request/reply 时，选项 replyToConcurrentConsumers 用于控制回复消息监听器上的并发消费者数量。	1	int
maxConcurrentConsumers (consumer)	指定从 JMS 消耗时的最大并发消费者数（而不是通过 JMS 请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。当通过 JMS 执行请求/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交付进行回复。	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。可使用的 durable 订阅名称通过 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 域)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)	设置要创建的订阅名称。在带有共享或可升级订阅的主题（公共域）中应用。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： 每个订阅都只允许 1 个并发消费者（这是此消息侦听器容器的默认值），但一个共享订阅（需要 JMS 2.0）除外。		字符串
subscriptionShared (consumer)	设置是否共享订阅。可以使用的共享订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。请注意，共享的订阅也可能是危险的，因此此标志也可以与订阅相结合。仅在侦听主题(pub-sub 域)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，则可能会考虑启用此选项，同时仍然在队列中排队消息。如果此选项为 false，并且您停止 JMS 路由，则消息可能会被拒绝，并且 JMS 代理必须尝试 redeliveries （但可能会再次拒绝），最终消息可能会移到 JMS 代理上的死信队列中。要避免这种情况，建议启用这个选项。	false	布尔值
allowReplyManagerQuickStop (consumer (advanced))	是否启用请求管理器中使用的 DefaultMessageListenerContainer，允许 DefaultMessageListenerContainer.runningAllowed 标志在 JmsConfigurationVirtualMachineisAcceptMessagesWhileStopping 时快速停止，并且 org.apache.camel.CamelContext 当前已停止。在常规 JMS 用户中默认启用这种快速停止功能，但要为回复管理器启用这个标志。	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。消费者类型决定要使用的 Spring JMS 侦听器。默认将使用 org.springframework.jms.listener.DefaultMessageListenerContainer，Simple 将使用 org.springframework.jms.listener.SimpleMessageListenerContainer。指定 Custom 时，由 messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 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))	指定在 DefaultMessageListenerContainer 中使用哪些默认 TaskExecutor 类型，用于消费者端点和制作者端点的 ReplyTo consumer。可能的值： SimpleAsync（使用 Spring 的 SimpleAsyncTaskExecutor）或 ThreadPool （使用 Spring 的 ThreadPoolTaskExecutor 带有最佳值 - 缓存线程池）。如果没有设置，则默认为之前的行为，它将缓存线程池用于消费者端点，而 SimpleAsync 用于回复用户。建议使用 ThreadPool 来减少弹性配置中线程垃圾箱，同时动态增加和减少并发用户。 Enum 值： ● ThreadPool ● SimpleAsync		DefaultTaskExecutorType
eagerLoadingOfProperties (consumer (advanced))	加载消息时立即启用 JMS 属性和有效负载的 eager 加载，因为 JMS 属性可能并不是必需的，但有时可能会捕获与底层 JMS 提供程序和使用 JMS 属性的早期问题。另请参阅选项 eagerPoisonBody。	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)是 poison （无法读取/映射），然后将这个文本设置为消息正文，因此可以处理消息(poison 的原因)已作为交换异常保存。这可以通过设置 eagerPoisonBody=false 来关闭。另请参阅 eagerLoadingOfProperties 选项。	Poison JMS 消息，因为 $\{exception.message}	字符串

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
exposeListenerSession (consumer (advanced))	指定在消耗消息时是否应公开监听程序会话。	false	布尔值
replyToSameDestinationAllowed (consumer (advanced))	JMS 使用者是否允许向消费者使用的同一目的地发送回复消息。这可防止出现无限循环，并通过消耗并向自己发送相同的消息。	false	布尔值
taskExecutor (consumer (advanced))	允许您指定自定义任务执行器以使用消息。		TaskExecutor
deliveryDelay (producer)	设置用于发送 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。可能的值是由 javax.jms.DeliveryMode 定义的值。NON_PERSISTENT = 1 和 PERSISTENT = 2。 Enum 值： • 1 • 2		整数
deliveryPersistent (producer)	指定默认使用持久性交付。	true	布尔值

Name	描述	默认值	类型
explicitQosEnabled (producer)	设置在发送消息时使用 deliveryMode、priority 或 timeToLive 数量服务。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用到当前的端点。这与 preserveMessageQos 选项（按消息粒度运行）相反，从 Camel In 消息标头中读取 QoS 属性。	false	布尔值
formatDateHeadersToIso8601 (producer)	根据 ISO 8601 标准设置 JMS 日期属性是否应格式化。	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)	如果要使用消息中指定的 QoS 设置来发送消息，而不是 JMS 端点上的 QoS 设置。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供全部或仅提供其中一些。如果没有提供，Camel 将回退到使用端点中的值。因此，在使用此选项时，标头会覆盖端点中的值。与之相反，clearQosEnabled 选项将仅使用端点上设置的选项，而不使用来自消息标头中的值。	false	布尔值
priority (producer)	大于 1 的值指定发送时的消息优先级（其中 1 是最低优先级，9 是最高）。必须启用 explicitQosEnabled 选项才能使此选项生效。 Enum 值： • 1 • 2 • 3 • 4 • 5 • 6 • 7 • 8 • 9	4	int

Name	描述	默认值	类型
replyToConcurrentConsumers (producer)	指定执行请求/通过 JMS 回复时的默认并发消费者数量。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。	1	int
replyToMaxConcurrentConsumers (producer)	指定在通过 JMS 使用请求/回复时的最大并发消费者数。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。		int
replyToOnTimeoutMaxConcurrentConsumers (producer)	指定使用请求/通过 JMS 时超时时继续路由的最大并发消费者数。	1	int
replyToOverride (producer)	在 JMS 消息中提供显式 ReplyTo 目的地，这将覆盖 replyTo 的设置。如果要将消息转发到远程队列，并从 ReplyTo 目的地接收回复消息，这非常有用。		字符串
replyToType (producer)	允许明确指定在执行 request/reply 时要用于 replyTo 队列的策略类型。可能的值有：Temporary、share 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的队列。如需了解更多详细信息，请参阅 Camel JMS 文档，特别是在集群环境中运行时的影响的备注，以及共享回复队列的性能低于其 alternatives Temporary 和 Exclusive。 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 来覆盖此端点配置的超时值，因此具有每个消息独立的超时值。另请参阅 requestTimeoutCheckerInterval 选项。	20000	long
timeToLive (producer)	发送消息时，指定消息的时间到时间（以毫秒为单位）。	-1	long

Name	描述	默认值	类型
allowAdditionalHeaders (producer (advanced))	此选项用于允许其他标头，这些标头可能具有根据 JMS 规范无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 包含字节数组或其他无效类型的值来执行此操作。您可以用逗号指定多个标头名称，并用作通配符匹配的后缀。		字符串
allowNullBody (producer (advanced))	是否允许在没有正文的情况下发送消息。如果此选项为 false，且消息正文为 null，则会抛出 JMSException。	true	布尔值
alwaysCopyMessage (producer (advanced))	如果为 true，则 Camel 始终会在消息传递给发送的制作者时生成 JMS 消息副本。在某些情况下需要复制消息，如设置 replyToDestinationSelectorName 时（如果设置了 replyToDestinationSelectorName，则 Camel 会将 alwaysCopyMessage 选项设置为 true）。	false	布尔值
correlationProperty (producer (advanced))	当使用 InOut 交换模式时，使用此 JMS 属性而不是 JMSCorrelationID JMS 属性来关联消息。如果设置消息仅针对此属性 JMSCorrelationID 属性的值关联，则将忽略且未由 Camel 设置。		字符串
disableTimeToLive (producer (advanced))	使用这个选项强制禁用时间。例如，当您通过 JMS 进行请求/回复时，Camel 默认使用 requestTimeout 值作为发送的消息的时间。问题是，发送者和接收器系统必须同步其时钟，因此它们正在同步。这并非始终易于归档。因此，您可以使用 disableTimeToLive=true 来不设置发送消息上的生存时间。然后，消息不会在接收器系统中过期。如需了解更多详细信息，请参见以下部分关于生存时间。	false	布尔值
forceSendOriginalMessage (producer (advanced))	当使用 mapJmsMessage=false Camel 时，如果在路由中涉及标头(get 或 set)，则会创建一个新的 JMS 消息来发送到新的 JMS 目的地。将此选项设置为 true，以强制 Camel 发送收到的原始 JMS 消息。	false	布尔值
includeSentJMSMessageID (producer (advanced))	仅在使用 InOnly 发送到 JMS 目的地时适用（例如触发和忘记）。启用此选项将增强 Camel Exchange 与 JMS 客户端在消息发送到 JMS 目的地时使用的实际 JMSMessageID。	false	布尔值

Name	描述	默认值	类型
replyToCacheLevelName (producer (advanced))	在执行请求/通过 JMS 时，按名称为回复消费者设置缓存级别。这个选项仅在使用固定回复队列（而非临时）时才适用。Camel 默认将使用：CACHE_CONSUMER 用于专用或共享的 w/ replyToSelectorName。和 CACHE_SESSION 用于没有 replyToSelectorName 的共享。IBM WebSphere 等 JMS 代理可能需要设置 replyToCacheLevelName=CACHE_NONE 才能正常工作。注：如果使用临时队列，则不允许使用更高的值，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，以便您可以在使用共享队列（也就是说，如果您不使用临时回复队列）时过滤来自其他回复的回复。		字符串
streamMessageTypeEnabled (producer (advanced))	设置 StreamMessage 类型是否已启用。流类型的消息有效负载（如 files、InStream 等）将通过作为 BytesMessage 或 StreamMessage 发送。此选项控制将使用哪种类型。默认情况下，使用 BytesMessage 来强制整个消息有效负载读取到内存中。通过启用此选项，消息有效负载以块的形式读取到内存中，然后每个块都会写入 StreamMessage，直到没有更多数据。	false	布尔值
allowSerializedHeaders (advanced)	控制是否包含序列化标头。仅在 transferExchange 为 true 时才适用。这要求对象可以序列化。Camel 将排除任何不可序列化的对象，并将它记录在 WARN 级别。	false	布尔值
artemisStreamingEnabled (advanced)	是否针对 Apache Artemis 流模式进行优化。这可减少使用带有 JMS StreamMessage 类型的 Artemis 时的内存开销。只有在使用 Apache Artemis 时，才必须启用这个选项。	false	布尔值

Name	描述	默认值	类型
asyncStartListener (advanced)	在启动路由时，是否异步启动 JmsConsumer 消息监听程序。例如，如果 JmsConsumer 无法获得连接到远程 JMS 代理的连接，那么在重试和/或故障切换时可能会阻断它。这将使 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 使用异步模式的专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果没有建立连接，则会在 WARN 级别记录异常，使用者将无法接收消息；然后，您可以重启路由来重试。	false	布尔值
asyncStopListener (advanced)	在停止路由时，是否异步停止 JmsConsumer 消息监听程序。	false	布尔值
destinationResolver (advanced)	一个可插拔的 org.springframework.jms.support.destination.DestinationResolver，供您使用自己的解析器（例如，在 JNDI registry 中查找实际目的地）。		DestinationResolver
errorHandler (advanced)	指定在处理消息时引发任何未发现异常时要调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置错误处理程序，则这些例外将在 WARN 级别记录。您可以配置日志记录级别，以及是否应使用 errorHandlerLoggingLevel 和 errorHandlerLogStack Trace 选项记录堆栈追踪。这样可以更容易配置，而不是需要对自定义错误处理程序进行代码。		ErrorHandler
exceptionListener (advanced)	指定针对任何底层 JMS 异常通知的 JMS Exception Listener。		ExceptionListener
HeaderFilterStrategy (advanced)	使用自定义 HeaderFilterStrategy 将标头过滤到或从 Camel 消息过滤。		HeaderFilterStrategy
idleConsumerLimit (advanced)	指定任何给定时间允许闲置的用户数量的限值。	1	int
idleTaskExecutionLimit (advanced)	指定接收任务闲置执行的限制，不会在其执行中收到任何消息。如果达到这个限制，任务将关闭并将接收给其他执行任务（在动态调度时；请参阅 maxConcurrentConsumers 设置）。Spring 提供了额外的文档。	1	int
includeAllJMSXProperties (advanced)	在从 JMS 到 Camel Message 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，则不会应用这个选项。	false	布尔值

Name	描述	默认值	类型
jmsKeyFormatStrategy (advanced)	编码和解码 JMS 密钥的可插拔策略，以便它们符合 JMS 规范。Camel 提供两个开箱即用的实现：default 和 passthrough。默认策略将安全地使用句点和连字符 (. 和 -)。passthrough 策略将密钥保留原样。可用于不负责 JMS 标头密钥是否包含非法字符的 JMS 代理。您可以自行提供 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 sVirt 表示法引用它。 Enum 值： • default • passthrough		JmsKeyFormatStrategy
mapJmsMessage (advanced)	指定 Camel 是否应该自动将收到的 JMS 消息映射到合适的有效负载类型，如 javax.jms.TextMessage 到 String 等。	true	布尔值
maxMessagesPerTask (advanced)	每个任务的消息数量。-1 代表没有限制。如果您为并发消费者使用范围（例如 min max），则可以使用此选项将值设为 100，以控制消费者在需要较少的工作时可以缩小的速度。	-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)	使用给定的 MessageCreatedStrategy，当 Camel 发送 JMS 消息时，Camel 创建 javax.jms.Message 对象的新实例。		MessageCreatedStrategy
messageIdEnabled (advanced)	发送时，指定是否应添加消息 ID。这只是对 JMS 代理的提示。如果 JMS 提供程序接受此提示，则这些消息必须将消息 ID 设置为 null；如果提供程序忽略提示，则必须将消息 ID 设置为其普通唯一值。	true	布尔值
messageListenerContainerFactory (advanced)	MessageListenerContainerFactory 的 registry ID，用于决定要使用消息的 org.springframework.jms.listener.AbstractMessageListenerContainer。设置此项将自动将 consumerType 设置为 Custom。		MessageListenerContainerFactory
messageTimestampEnabled (advanced)	指定在发送消息时是否默认启用时间戳。这只是对 JMS 代理的提示。如果 JMS 提供程序接受此提示，则这些消息必须将时间戳设置为零；如果提供程序忽略提示，则必须将时间戳设置为其正常值。	true	布尔值

Name	描述	默认值	类型
pubSubNoLocal (advanced)	指定是否禁止自己连接发布的消息的发送。	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，即当连接被刷新时，以毫秒为单位。默认值为 5000 ms，即 5 秒。	5000	long
requestTimeoutCheckerInterval (advanced)	配置 Camel 在执行请求/通过 JMS 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，如果发生超时时，您必须更快地响应，那么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
Sync (advanced)	设置是否应严格使用同步处理。	false	布尔值
transferException (advanced)	如果启用了且您使用 Request Reply messaging (InOut)，并且 Exchange 失败在消费者端，则原因例外将作为 javax.jms.ObjectMessage 发回的响应。如果客户端是 Camel，则返回的例外将重新箭头。这样，您可以使用 Camel JMS 作为路由中的桥接 - 例如，使用持久性队列来启用强大的路由。请注意，如果您也启用了 transferExchange，这个选项将具有优先权。caught 异常需要可以被序列化。消费者端的原始例外可以嵌套在外部异常中，如 org.apache.camel.RuntimeCamelException。请谨慎使用它，因为数据正在使用 Java 对象序列化，要求接收者在类级别反序列化数据，这会强制在生产者和消费者之间进行强校准。	false	布尔值
transferExchange (advanced)	您可以在有线线上传输交换，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 header, Exchange properties, exchange exception.这要求对象可以序列化。Camel 将排除任何不可序列化的对象，并将它记录在 WARN 级别。您必须在制作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。请谨慎使用它，因为数据正在使用 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，这会强制在需要使用兼容 Camel 版本的生产者和消费者之间进行强大的协调。	false	布尔值
useMessageIDAsCorrelationID (advanced)	指定 JMSMessageID 是否应该始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值

Name	描述	默认值	类型
waitForProvisionCorrelationToBeUpdatedCounter (advanced)	在执行 request/reply over JMS 以及启用了 useMessageIDAsCorrelationID 时，等待 provisional correlation id 被更新到实际关联 ID 的次数。	50	int
waitForProvisionCorrelationToBeUpdatedThreadSleepingTime (advanced)	等待置备关联 ID 时每次休眠的时间间隔（以秒为单位）。	100	long
errorHandlerLoggingLevel (logging)	允许为日志记录 uncaught 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)	允许通过默认错误处理程序来控制是否应记录 stacktraces。	true	布尔值
密码 （安全）	与 ConnectionFactory 一起使用的密码。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
用户名 （安全）	与 ConnectionFactory 一起使用的用户名。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
Transacted (transaction)	指定是否使用转换模式。	false	布尔值

Name	描述	默认值	类型
TransactedInOut (transaction)	指定 InOut 操作（请求回复）是否默认使用 transacted 模式，如果此标志被设置为 true，则 Spring JmsTemplate 将把 sessionTransacted 设置为 true，而 confirmMode 作为转换用于 InOut 操作。请注意：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，容器对这些值做出自己的决策。类似地，这些参数不会考虑本地管理的事务，因为 Spring JMS 在这种情况下在现有 JMS 会话上运行。在受管事务外运行时，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（除 XA 事务之外）时同步的本地 JMS 事务。这与主事务一起管理本地 JMS 事务（可能是原生 JDBC 事务）的影响，在主事务后 JMS 事务提交右侧的 JMS 事务。	false	布尔值
lazyCreateTransactionManager (transaction (advanced))	如果为 true，Camel 将创建一个 JmsTransactionManager，如果没有在选项 transacted=true 时注入 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))	如果使用转换模式，事务的超时值（以秒为单位）。	-1	int

31.5. SAMPLES

JMS 也在许多示例中用于其他组件。但是，我们提供了几个示例来开始。

31.5.1. 从 JMS 接收

在以下示例中，我们将配置一个路由，该路由接收 JMS 信息并将消息路由到 POJO：

```
from("jms:queue:foo").
  to("bean:myBusinessLogic");
```

您可以使用任何 EIP 模式，以便路由基于上下文。例如，这里的如何为大消耗器过滤订购主题：

```
from("jms:topic:OrdersTopic").
  filter().method("myBean", "isGoldCustomer").
  to("jms:queue:BigSpendersQueue");
```

31.5.2. 发送到 JMS

在以下示例中，我们轮询文件文件夹，并将文件内容发送到 JMS 主题。当我们希望将文件的内容作为 TextMessage 而不是 BytesMessage 一样，我们需要将正文转换为 字符串 ：

```
from("file://orders").
  convertBodyTo(String.class).
  to("jms:topic:OrdersTopic");
```

31.5.3. 使用注解

Camel 还具有注释，因此您可以使用 POJO Consuming 和 POJO Producing。

31.5.4. 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 的大消耗的示例：

```
<route>
  <from uri="jms:topic:OrdersTopic"/>
  <filter>
    <method ref="myBean" method="isGoldCustomer"/>
    <to uri="jms:queue:BigSpendersQueue"/>
  </filter>
</route>
```

31.5.5. 其他示例

JMS 出现在其他组件和 EIP 模式的许多示例中，以及此 Camel 文档。因此，可以浏览文档。

31.5.6. 使用 JMS 作为 Dead Letter Queue Store Exchange

通常，当使用 JMS 作为传输时，它仅传输正文和标头作为有效负载。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，您可以在 JMS 死信队列中使用 transferExchange 选项来指示 Camel 将整个 Exchange 存储在队列中，作

为包含 org.apache.camel.support.DefaultExchangeHolder 的 javax.jms.ObjectMessage。这可让您从 Dead Letter Queue 中使用，并使用密钥 Exchange.EXCEPTION_CAUGHT 从 Exchange 属性中检索原因异常。以下演示展示了这一点：

```
// setup error handler to use JMS as queue and store the entire Exchange
errorHandler(deadLetterChannel("jms:queue:dead?transferExchange=true"));
```

然后，您可以从 JMS 队列中消耗并分析问题：

```
from("jms:queue:dead").to("bean:myErrorAnalyzer");

// and in our bean
String body = exchange.getIn().getBody();
Exception cause = exchange.getProperty(Exchange.EXCEPTION_CAUGHT, Exception.class);
// the cause message is
String problem = cause.getMessage();
```

31.5.7. 使用 JMS 作为只存储错误的 Dead Letter Channel

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 在移到 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，我们只会将原始原因错误消息存储在转换中。但是，您可以使用任何表达式来发送您需要的任何表达式。例如，您可以在 Bean 上调用方法或使用自定义处理器。

31.6. JMS 和 CAMEL 之间的消息映射

Camel 自动映射 javax.jms.Message 和 org.apache.camel.Message 之间的消息。

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 Message	注释
字符串	javax.jms.TextMessage	

正文类型	JMS Message	注释
org.w3c.dom.Node	**javax.jms.TextMessage**	DOM 将转换为 **字符串**。
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 信息转换为以下正文类型：

JMS Message	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**

31.6.1. 禁用自动映射 JMS 消息

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，而是直接将其用作有效负载。这可让您避免映射开销，并使 Camel 只是通过 JMS 消息传递。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。

31.6.2. 使用自定义 MessageConverter

您可以使用 messageConverter 选项在 Spring org.springframework.jms.support.converter.MessageConverter 类中进行映射。

例如，在下面的路由中，在向 JMS order 队列发送消息时，使用自定义消息转换器：

```
from("file://inbox/order").to("jms:queue:order?messageConverter=#myMessageConverter");
```

在从 JMS 目的地消耗时，您还可以使用自定义消息转换器。

31.6.3. 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，从文件夹中轮询文件，并将其作为 javax.jms.TextMessage 发送，因为我们已强制 JMS producer 端点使用文本消息：

```
from("file://inbox/order").to("jms:queue:order?jmsMessageType=Text");
```

您还可以通过使用键 CamelJmsMessageType 设置标头来指定要用于每个消息的消息类型。例如：

```
from("file://inbox/order").setHeader("CamelJmsMessageType",
JmsMessageType.Text).to("jms:queue:order");
```

可能的值在 enum 类 org.apache.camel.jms.JmsMessageType 中定义。

31.7. 发送时的消息格式

通过 JMS 有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 ：

- 从 JMS 或 JMSX 开头的密钥将被保留。
- exchange.in.headers 键必须是字面数，并且全部都是有效的 Java 标识符（不要在密钥名称中使用点）。
-

Camel 在消耗 JMS 消息时替换句点和连字符，在 Camel 使用消息时被 'DOT' 替代，并在 Camel 使用消息时替换反向替换。
- 被 'HYPHEN' 替换，并在 Camel 消耗消息时替换反向替换。

- *另请参阅 jmsKeyFormatStrategy 选项，该选项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 primitives 或其计数器对象（如 Integer、Long、Character）。类型、字符串、CharSequence、日期、MuterDecimal 和 BigInteger 都转换为其 toString （） 表示。所有其他类型都会丢弃。*

如果它丢弃给定标头值，Camel 会在 DEBUG 级别使用类别 org.apache.camel.component.jms.JmsBinding 记录。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}
```

31.8. 接收时的消息格式

Camel 在收到消息时向 交换 添加以下属性：

属性	类型	描述
org.apache.camel.jms.replyDestination	**javax.jms.Destination**	回复目的地。

Camel 在收到 JMS 消息时，在 In 消息标头中添加以下 JMS 属性：

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 唯一的消息 ID。
JMSPriority	**int**	JMS 优先级(0 作为最低优先级，9 为最高)。
JMSRedelivered	**布尔值**	是 JMS 消息，redelivered。
JMSReplyTo	**javax.jms.Destination**	JMS 回复到目的地。
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的 JMS，您可以检查 JMS 文档 以了解更多详情。

31.9. 关于使用 CAMEL 来发送和接收消息，以及 JMSREPLYTO

JMS 组件比较复杂，您必须特别注意它在某些情况下如何工作。因此，这是要查找的一些 Region/pitfalls 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，*
- *在端点或消息标头中设置了 JMSReplyTo，*
- *是否在 JMS 端点上设置了以下选项：disableReplyTo、preserveMessageQos,explicitQosEnabled。*

所有这些都很复杂，理解并配置来支持您的用例。

31.9.1. JmsProducer

JmsProducer 的行为如下，具体取决于配置：

交换模式	其他选项	描述
InOut	-	Camel 将期望回复，设置临时 **JMSReplyTo**，并在发送消息后，它将开始侦听临时队列上的回复消息。
InOut	**JMSReplyTo** 被设置	Camel 将期望回复，并在发送消息后，它将开始侦听指定的 **JMSReplyTo** 队列上的回复消息。
InOnly	-	Camel 将发送邮件，**而不是** 预期回复。
InOnly	**JMSReplyTo** 被设置	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** 期望回复。Camel 会在 **WARN** 级别的日志中记录它（从 **Camel 2.6** 改为 **DEBUG** 级别）。您可以使用 **preserveMessageQuo=true** 来指示 Camel 保留 **JMSReplyTo**。在所有情况下，**JmsProducer 并不** 期望任何回复，因此在发送消息后继续。

31.9.2. JmsConsumer

JmsConsumer 的行为如下，具体取决于配置：

交换模式	其他选项	描述
InOut	-	Camel 将回复发送回 **JMSReplyTo** 队列。
InOnly	-	Camel 不会发送回复回来，因为模式是 *InOnly*。
-	**disableReplyTo =true**	这个选项会阻止回复。

请注意您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 信息，这非常有用：

```
from("activemq:queue:in")
  .to("bean:validateOrder")
  .to(ExchangePattern.InOnly, "activemq:topic:order")
  .to("bean:handleOrder");
```

31.10. 重复使用端点并发送到在运行时计算的不同目的地

如果您需要发送消息到许多不同的 JMS 目的地，可以重复利用 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。这可显著减少内存和线程资源上创建的端点数量。

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**	目标名称。

例如，以下路由演示了如何在运行时计算目的地，并使用它来覆盖 JMS URL 中显示的目的地：

```
from("file://inbox")
  .to("bean:computeDestination")
  .to("activemq:queue:dummy");
```

队列名称 dummy 只是一个占位符。它必须作为 JMS 端点 URL 的一部分提供，但本例中将被忽略。

在 computeDestination bean 中，设置 CamelJmsDestinationName 标头来指定实际目的地，如下所示：

```
public void setJmsHeader(Exchange exchange) {
  String id = ....
  exchange.getIn().setHeader("CamelJmsDestinationName", "order:" + id");
}
```

然后 Camel 将读取此标头并将其用作目的地，而不是在端点上配置的标头。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。

如果同时设置了 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）。

31.11. 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。前面的示例配置了 activemq 组件。您可以执行同样的方法来配置 MQSeries、TibCo、TibCo、Sonic 等。

命名的 JMS 组件后，您可以使用 URI 引用该组件中的端点。例如，对于组件名称 activemq，您可以使用 URI 格式引用目的地，activemq:[queue:|topic:]destinationName。您可以将相同的方法用于所有其他 JMS 提供程序。

这可通过 Spring CamelContext 许可，从用于 Endpoint URI 的方案名称的 spring 上下文获取组件，并让组件解析端点 URI。

31.11.1. 使用 JNDI 查找 ConnectionFactory

如果您使用 J2EE 容器，您可能需要查找 JNDI 以查找 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。

31.12. 并发消耗

JMS 的常见要求是在多个线程中同时使用消息，以便应用程序更快响应。您可以设置

concurrentConsumers 选项，以指定为 JMS 端点提供服务的线程数量，如下所示：

```
from("jms:SomeQueue?concurrentConsumers=20").
  bean(MyClass.class);
```

您可以使用以下方法之一配置这个选项：

- *在 JmsComponent 上，*
- *在端点 URI 或.*
- *通过直接在 JmsEndpoint 上调用 setConcurrentConsumers （）。*

31.12.1. 使用 async 使用者并发恢复

请注意，当当前消息被完全处理后，每个并发消费者将仅从 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 队列中提取下一个消息，而前面的消息则异步处理（通过 Asynchronous Routing Engine）。有关 asyncConsumer 选项的信息，请参阅页面顶部的更多详情。

```
from("jms:SomeQueue?concurrentConsumers=20&asyncConsumer=true").
  bean(MyClass.class);
```

31.13. 通过 JMS 请求代表

Camel 支持通过 JMS 重新请求请求。在向 JMS 队列发送消息时，交换的 MEP 应该为 InOut。

Camel 提供了多个选项来配置请求/对 JMS 的影响，它们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，并由 Camel 自动创建。要使用它，**不要指定** replyTo 队列名称。另外，您还可以配置 **replyToType=Temporary**，使其认为临时队列正在使用。

选项	性能	集群	描述
共享	slow	是	共享持久性队列用作回复队列。必须事先创建队列，但有些代理可以在实时（如 Apache ActiveMQ）上创建它们。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。一个共享队列可以在有同时运行此 Camel 应用程序的多个节点的集群环境中使用。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；这会影响性能。JMS 消息选择器较慢，因此不会快为 **Temporary** 或 **Exclusive** 队列。请参阅下面的 如何调整此功能以提高性能。
exclusive	速度快	否（是）	专用持久队列用作回复队列。必须事先创建队列，但有些代理可以在实时（如 Apache ActiveMQ）上创建它们。要使用此功能，您必须指定 replyTo 队列名称。**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，JMS 消息选择器 **不会被使用**，因此其他应用也不得使用此队列。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。但是，如果您使用每个节点的唯一名称配置每个 Exclusive 回复队列，您可以在集群环境中运行它。**因此，回复消息将发送到给定节点的该队列，从而等待回复消息。**
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：请注意**，使用共享回复队列可能也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：请注意**，使用共享回复队列可能也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，并提供带有要使用的回复目的地的 JMSReplyTo 标头。默认情况下，Camel 使用临时队列，但您可以使用端点上的 replyTo 选项指定固定的回复队列（请参阅以下有关固定回复队列的更多信息）。

Camel 将自动设置侦听回复队列的消费者，因此 您不应该 执行任何操作。
此消费者是一个 Spring DefaultMessageListenerContainer，它侦听回复。但是，它被固定到 1 个并发消费者。
这意味着回复将按顺序处理，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。这可让您更轻松地在 Camel 中配置此功能，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 使用具有 5 个线程的线程池异步路由回复。

31.13.1. 请求通过 JMS 并使用共享的固定回复队列

如果您在执行 Request Reply over JMS 时使用固定回复队列，如以下示例所示，请小心。

```
from(xxx)
.inOut().to("activemq:queue:foo?replyTo=bar")
.to(yyy)
```

在本例中，使用名为"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是共享的，因此它使用 JMSSelector 来仅获取预期的回复消息（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 为回复消息拉取频率。默认情况下，其 1000 millis。因此，您可以将它设置为 250 millis 以每秒拉取 4 次，如下所示：

```
from(xxx)
.inOut().to("activemq:queue:foo?replyTo=bar&receiveTimeout=250")
.to(yyy)
```

请注意，这会导致 Camel 更频繁地向消息代理发送拉取请求，因此需要更多网络流量。
通常建议您尽可能使用临时队列。

31.13.2. 请求通过 JMS，并使用一个专用固定回复队列

在上例中，Camel 将预期名为"bar"的固定回复队列被共享，因此它使用 JMSSelector 来仅消耗期望的回复消息。但是，这个问题有缺陷，因为 JMS 选择器会较慢。另外，回复队列上的消费者使用新的 JMS 选择器 ID 更新速度较慢。实际上，只有在 receiveTimeout 选项超时时才会更新，默认为 1 秒。因此，在回复消息中，可能会检测到大约 1 秒。另一方面，如果固定回复队列专用于 Camel 回复消费者，则我们可以避免使用 JMS 选择器，因此更为高性能。实际上，使用临时队列就快。您可以将 ReplyToType 选项配置为 Exclusive
，以告知 Camel 回复队列是独占的，如下例所示：

```
from(xxx)
.inOut().to("activemq:queue:foo?replyTo=bar&replyToType=Exclusive")
.to(yyy)
```

请记住，队列必须专用于每个端点和每个端点。因此，如果您有两个路由，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，则适用相同。然后，集群中的每个节点都必须使用唯一的回复队列名称。否则，集群中的每个节点可能会获取旨在作为另一节点上的回复的信息。对于集群环境，建议您使用共享回复队列。

31.14. 在发送方和接收器之间同步时钟

在系统间进行消息传递时，最好系统具有同步时钟。例如，发送 JMS 消息时，您可以在消息上设置生存时间。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和处理。但是，这需要发送方和接收器都同步时钟。如果使用 ActiveMQ，您可以使用 timestamp 插件来 同步时钟。

31.15. 关于生存时间

首先阅读有关同步时钟的上面。

当您使用 Camel 通过 JMS 进行请求/回复(InOut)时，Camel 会在发送方端使用超时，这是 requestTimeout 选项的默认 20 秒。您可以通过设置更高的/低值来控制这个值。但是，在正在发送的消息上仍然设置了实时值的时间。因此，这需要在系统之间同步时钟。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，如果您对 JMS 进行请求/回复，并且禁用了生存时间，则 Camel 仍然将

使用 20 秒的超时(requestTimeout 选项)。也可以配置该选项。因此，两个选项 requestTimeout 和 disableTimeToLive 可让您在执行请求/回复时进行精细控制。

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，我们有一个端点配置的 requestTimeout 为 30 秒。因此，Camel 将等待 30 秒，以便该回复消息回到栏队列中。如果没有收到回复消息，则在 Exchange 上设置 org.apache.camel.ExchangeTimedOutException，Camel 会继续路由消息，然后因为异常而失败，Camel 的错误处理程序响应。

如果要使用每个消息超时值，您可以使用键 org.apache.camel.component.jms.JmsConstants114JMS_REQUEST_TIMEOUT 设置标头，其值为 "CamelJmsRequestTimeout "。

例如，我们可以使用 bean 计算每个消息的超时值，例如在服务 bean 上调用 "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 通过 JMS 触发和忘记(InOut)时，默认情况下 Camel 不会将 任何时间设置为消息上的实时值。您可以使用 timeToLive 选项配置值。例如，要指定 5 秒，您可以设置 timeToLive=5000。选项 disableTimeToLive 可用于强制禁用生存时间，也用于 InOnly messaging。requestTimeout 选项不用于 InOnly messaging。

31.16. 启用 TRANSACTED CONSUMPTION

常见要求是从事务中的队列消耗，然后使用 Camel 路由处理消息。要做到这一点，只要确保在组件/端点上设置以下属性：

- transacted = true

- transactionManager = Transsaction Manager - 通常是 JmsTransactionManager

详情请查看 Transactional Client EIP 模式。

与 JMS 相比，事务和 [Request Reply]

当通过 JMS 使用 Request Reply 时，您不能使用单个事务；在执行提交前，JMS 不会发送任何消息，因此服务器端不会完全接收任何内容，直到事务提交为止。因此，要使用 Request Reply，您必须在发送请求后提交事务，然后使用单独的事务接收响应。

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)。

您可以使用组件/端点中的以下属性利用 DMLC 转换的会话 API ：

- transacted = true
- lazyCreateTransactionManager = false

这样做的好处在于，在使用没有配置的 TransactionManager 的本地事务时，会遵循 cacheLevel 设置。配置 TransactionManager 时，在 DMLC 级别不需要缓存，需要依赖于池的连接工厂。有关此类设置的详情，请查看 此处 和 此处。

31.17. 使用 JMSREPLYTO 进行 LATE 回复

当使用 Camel 作为 JMS 侦听器时，它会使用 ReplyTo javax.jms.Destination 对象的值设置 Exchange 属性，其键为 ReplyTo。您可以按如下方式 获取 此目标：

```
Destination replyDestination =
exchange.getIn().getHeader(JmsConstants.JMS_REPLY_DESTINATION, Destination.class);
```

然后，之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 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.");
  }
}
```

31.18. 使用请求超时

在以下示例中，我们发送了一个 Request Reply 风格的消息 Exchange （我们使用 requestBody method = InOut）到 slow 队列以便在 Camel 中进一步处理，然后等待返回回复：

31.19. 发送 INONLY 消息并保留 JMSREPLYTO 标头

当使用 camel-jms 发送到 JMS 目的地时，生成者将使用 MEP 检测其 InOnly 或 InOut 消息传递。但是，在有些情况下，您可以发送 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 信息发送到 foo 队列，但带有带有 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。

31.20. 在目的地上设置 JMS 供应商选项

有些 JMS 提供程序，如 IBM 的 WebSphere MQ 需要在 JMS 目的地上设置选项。例如，您可能需要指定 targetClient 选项。由于 targetClient 是 WebSphere MQ 选项，而不是 Camel URI 选项，您需要在 JMS 目标名称上设置它，如：

```
// ...
.setHeader("CamelJmsDestinationName", constant("queue:///MY_QUEUE?targetClient=1"))
.to("wmq:queue:MY_QUEUE?useMessageIDAsCorrelationID=true");
```

一些 WMQ 版本在目的地名称中不接受这个选项，您会得到如下例外：

```
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");
    }
});
```

31.21. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 jms 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>
```

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，则可能会考虑启用此选项，同时仍然在队列中排队消息。如果此选项为 false，并且您停止 JMS 路由，则消息可能会被拒绝，并且 JMS 代理必须尝试 redeliveries （但可能会再次拒绝），最终消息可能会移到 JMS 代理上的死信队列中。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，其为 ： 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，这些标头可能具有根据 JMS 规范无效的值。例如，一些消息系统（如 WMQ）使用前缀 JMS_IBM_MQMD_ 包含字节数组或其他无效类型的值来执行此操作。您可以用逗号指定多个标头名称，并用作通配符匹配的后缀。		字符串
camel.component.jms.allow-auto-wired-connection-factory	如果没有配置连接工厂，是否从 registry 自动发现 ConnectionFactory。如果只找到一个 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 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	是否允许在没有正文的情况下发送消息。如果此选项为 false，且消息正文为 null，则会抛出 JMSException。	true	布尔值
camel.component.jms.allow-reply-manager-quick-stop	是否启用请求管理器中使用的 DefaultMessageListenerContainer，允许 DefaultMessageListenerContainer.runningAllowed 标志在 JmsConfigurationVirtualMachineisAcceptMessages WhileStopping 时快速停止，并且 org.apache.camel.CamelContext 当前已停止。在常规 JMS 用户中默认启用这种快速停止功能，但要为回复管理器启用这个标志。	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 时才适用。这要求对象可以序列化。Camel 将排除任何不可序列化的对象，并将它记录在 WARN 级别。	false	布尔值

Name	描述	默认值	类型
camel.component.jms.always-copy-message	如果为 true，则 Camel 始终会在消息传递给发送的制作者时生成 JMS 消息副本。在某些情况下需要复制消息，如设置 replyToDestinationSelectorName 时（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消息处于活跃状态时收到消息。通常，连接到队列的活动消费者以轮循方式从它接收消息。当使用消费者优先级时，如果有多个具有相同高优先级的活跃用户，则消息将进行循环发送。只有高优先级消费者没有可用的信用消息或高优先级消费者接受消息时，消息才会降低优先级较低的消费者（例如，它不符合与消费者关联的任何选择器的条件）。		整数
camel.component.jms.artemis-streaming-enabled	是否针对 Apache Artemis 流模式进行优化。这可减少使用带有 JMS StreamMessage 类型的 Artemis 时的内存开销。只有在使用 Apache Artemis 时，才必须启用这个选项。	false	布尔值
camel.component.jms.async-consumer	JmsConsumer 是否异步处理交换。如果启用，JmsConsumer 可以从 JMS 队列中提取下一个消息，而上一个消息则异步处理（通过 Asynchronous Routing Engine）。这意味着消息可能没有完全严格按照顺序进行处理。如果禁用（作为默认），则在 JmsConsumer 会从 JMS 队列中提取下一个消息前，会完全处理交换。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 必须支持 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 无法获得连接到远程 JMS 代理的连接，那么在重试和/或故障切换时可能会阻断它。这将使 Camel 在启动路由时阻止。通过将此选项设置为 true，您将让路由启动，而 JmsConsumer 使用异步模式的专用线程连接到 JMS 代理。如果使用这个选项，请注意，如果没有建立连接，则会在 WARN 级别记录异常，使用者将无法接收消息；然后，您可以重启路由来重试。	false	布尔值
camel.component.jms.async-stop-listener	在停止路由时，是否异步停止 JmsConsumer 消息监听程序。	false	布尔值
camel.component.jms.auto-startup	指定消费者容器是否应自动启动。	true	布尔值

Name	描述	默认值	类型
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	根据底层 JMS 资源的名称设置缓存级别。可能的值有：CACHE_AUTO、CACHE_CONNECTION、CACHE_CONSUMER、CACHE_NONE 和 CACHE_SESSION。默认设置为 CACHE_AUTO。如需更多信息，请参阅 Spring 文档和事务缓存级别。	CACHE_AUTO	字符串
camel.component.jms.client-id	设置要使用的 JMS 客户端 ID。请注意，如果指定，这个值必须是唯一的，且只能被单个 JMS 连接实例使用。它通常只适用于持久主题订阅。如果使用 Apache ActiveMQ，您可能更喜欢使用 Virtual Topics。		字符串
camel.component.jms.concurrent-consumers	指定从 JMS 消耗时的默认并发消费者数量（而不是通过 JMS 请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。当通过 JMS 执行 request/reply 时，选项 replyToConcurrentConsumers 用于控制回复消息监听器上的并发消费者数量。	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。必须在组件或端点上配置连接工厂。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。消费者类型决定要使用的 Spring JMS 侦听器。默认将使用 org.springframework.jms.listener.DefaultMessageListenerContainer，Simple 将使用 org.springframework.jms.listener.SimpleMessageListenerContainer。指定 Custom 时，由 messageListenerContainerFactory 选项定义的 MessageListenerContainerFactory 将决定要使用的 org.springframework.jms.listener.AbstractMessageListenerContainer。		ConsumerType

Name	描述	默认值	类型
camel.component.jms.correlation-property	当使用 InOut 交换模式时，使用此 JMS 属性而不是 JMSCorrelationID JMS 属性来关联消息。如果设置消息仅针对此属性 JMSCorrelationID 属性的值关联，则将忽略且未由 Camel 设置。		字符串
camel.component.jms.default-task-executor-type	指定在 DefaultMessageListenerContainer 中使用哪些默认 TaskExecutor 类型，用于消费者端点和制作者端点的 ReplyTo consumer。可能的值： SimpleAsync（使用 Spring 的 SimpleAsyncTaskExecutor）或 ThreadPool （使用 Spring 的 ThreadPoolTaskExecutor 带有最佳值 - 缓存线程池）。如果没有设置，则默认为之前的行为，它将缓存线程池用于消费者端点，而 SimpleAsync 用于回复用户。建议使用 ThreadPool 来减少弹性配置中线程垃圾箱，同时动态增加和减少并发用户。		DefaultTaskExecutorType
camel.component.jms.delivery-delay	设置用于发送 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。可能的值是由 javax.jms.DeliveryMode 定义的值。NON_PERSISTENT = 1 和 PERSISTENT = 2。		整数
camel.component.jms.delivery-persistent	指定默认使用持久性交付。	true	布尔值
camel.component.jms.destination-resolver	一个可插拔的 org.springframework.jms.support.destination.DestinationResolver，供您使用自己的解析器（例如，在 JNDI registry 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 从路由中消耗，并且您不希望 Camel 自动发送回复消息，则可以使用此选项，因为代码中的另一个组件处理回复消息。如果要在不同的消息代理之间将 Camel 用作代理，并且希望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
camel.component.jms.disable-time-to-live	使用这个选项强制禁用时间。例如，当您通过 JMS 进行请求/回复时，Camel 默认使用 requestTimeout 值作为发送的消息的时间。问题是，发送者和接收器系统必须同步其时钟，因此它们正在同步。这并非始终易于归档。因此，您可以使用 disableTimeToLive=true 来不设置发送消息上的生存时间。然后，消息不会在接收器系统中过期。如需了解更多详细信息，请参见以下部分关于生存时间。	false	布尔值
camel.component.jms.durable-subscription-name	用于指定持久主题订阅的可配置订阅者名称。还必须配置 clientId 选项。		字符串
camel.component.jms.eager-loading-of-properties	加载消息时立即启用 JMS 属性和有效负载的 eager 加载，因为 JMS 属性可能并不是必需的，但有时可能会捕获与底层 JMS 提供程序和使用 JMS 属性的早期问题。另请参阅选项 eagerPoisonBody。	false	布尔值
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)是 poison （无法读取/映射），然后将这个文本设置为消息正文，因此可以处理消息(poison 的原因)已作为交换异常保存。这可以通过设置 eagerPoisonBody=false 来关闭。另请参阅 eagerLoadingOfProperties 选项。	Poison JMS 消息，因为 $\{exception.message}	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	指定在处理消息时引发任何未发现异常时要调用的 org.springframework.util.ErrorHandler。默认情况下，如果没有配置错误处理程序，则这些例外将在 WARN 级别记录。您可以配置日志记录级别，以及是否应使用 errorHandlerLoggingLevel 和 errorHandlerLogStack Trace 选项记录堆栈追踪。这样可以更容易配置，而不是需要对自定义错误处理程序进行代码。选项是一个 org.springframework.util.ErrorHandler 类型。		ErrorHandler
camel.component.jms.error-handler-log-stack-trace	允许通过默认错误处理程序来控制是否应记录 stacktraces。	true	布尔值
camel.component.jms.error-handler-logging-level	允许为日志记录 uncaught 异常配置默认 errorHandler 日志记录级别。		LoggingLevel

Name	描述	默认值	类型
camel.component.jms.exception-listener	指定针对任何底层 JMS 异常通知的 JMS ExceptionListener。选项是 javax.jms.ExceptionListener 类型。		ExceptionListener
camel.component.jms.explicit-qos-enabled	设置在发送消息时使用 deliveryMode、priority 或 timeToLive 数量服务。这个选项基于 Spring 的 JmsTemplate。deliveryMode、priority 和 timeToLive 选项应用到当前的端点。这与 preserveMessageQos 选项（按消息粒度运行）相反，从 Camel In 消息标头中读取 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监听程序会话。	false	布尔值
camel.component.jms.force-send-original-message	当使用 mapJmsMessage=false Camel 时，如果在路由中涉及标头(get 或 set)，则会创建一个新的 JMS 消息来发送到新的 JMS 目的地。将此选项设置为 true，以强制 Camel 发送收到的原始 JMS 消息。	false	布尔值
camel.component.jms.format-date-headers-to-iso8601	根据 ISO 8601 标准设置 JMS 日期属性是否应格式化。	false	布尔值
camel.component.jms.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 将标头过滤到或从 Camel 消息过滤。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		HeaderFilterStrategy
camel.component.jms.idle-consumer-limit	指定任何给定时间允许闲置的用户数量的限值。	1	整数
camel.component.jms.idle-task-execution-limit	指定接收任务闲置执行的限制，不会在其执行中收到任何消息。如果达到这个限制，任务将关闭并将接收给其他执行任务（在动态调度时；请参阅 maxConcurrentConsumers 设置）。Spring 提供了额外的文档。	1	整数
camel.component.jms.include-all-j-m-s-x-properties	在从 JMS 到 Camel Message 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，则不会应用这个选项。	false	布尔值

Name	描述	默认值	类型
camel.component.jms.include-sent-j-m-s-message-id	仅在使用 InOnly 发送到 JMS 目的地时适用（例如触发和忘记）。启用此选项将增强 Camel Exchange 与 JMS 客户端在消息发送到 JMS 目的地时使用的实际 JMSMessageID。	false	布尔值
camel.component.jms.jms-key-format-strategy	编码和解码 JMS 密钥的可插拔策略，以便它们符合 JMS 规范。Camel 提供两个开箱即用的实现：default 和 passthrough。默认策略将安全地使用句点和连字符 (. 和 -)。passthrough 策略将密钥保留原样。可用于不负责 JMS 标头密钥是否包含非法字符的 JMS 代理。您可以自行提供 org.apache.camel.component.jms.JmsKeyFormatStrategy 的实现，并使用 sVirt 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 会决定要从 In body 类型使用哪个 JMS 消息类型。这个选项允许您指定它。		JmsMessageType
camel.component.jms.lazy-create-transaction-manager	如果为 true，Camel 将创建一个 JmsTransactionManager，如果没有在选项 transacted=true 时注入 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 是否应该自动将收到的 JMS 消息映射到合适的有效负载类型，如 javax.jms.TextMessage 到 String 等。	true	布尔值
camel.component.jms.max-concurrent-consumers	指定从 JMS 消耗时的最大并发消费者数（而不是通过 JMS 请求/回复）。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。当通过 JMS 执行请求/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。如果您为并发消费者使用范围（例如 min max），则可以使用此选项将值设为 100，以控制消费者在需要较少的工作时可以缩小的速度。	-1	整数

Name	描述	默认值	类型
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。选项是一个 org.springframework.jms.support.converter.MessageConverter 类型。		MessageConverter
camel.component.jms.message-created-strategy	使用给定的 MessageCreatedStrategy，当 Camel 发送 JMS 消息时，Camel 创建 javax.jms.Message 对象的新实例。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。这只是对 JMS 代理的提示。如果 JMS 提供程序接受此提示，则这些消息必须将消息 ID 设置为 null；如果提供程序忽略提示，则必须将消息 ID 设置为其普通唯一值。	true	布尔值
camel.component.jms.message-listener-container-factory	MessageListenerContainerFactory 的 registry ID，用于决定要使用消息的 org.springframework.jms.listener.AbstractMessageListenerContainer。设置此项将自动将 consumerType 设置为 Custom。选项是 org.apache.camel.component.jms.MessageListenerContainerFactory 类型。		MessageListenerContainerFactory
camel.component.jms.message-timestamp-enabled	指定在发送消息时是否默认启用时间戳。这只是对 JMS 代理的提示。如果 JMS 提供程序接受此提示，则这些消息必须将时间戳设置为零；如果提供程序忽略提示，则必须将时间戳设置为其正常值。	true	布尔值
camel.component.jms.password	与 ConnectionFactory 一起使用的密码。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
camel.component.jms.preserve-message-qos	如果要使用消息中指定的 QoS 设置来发送消息，而不是 JMS 端点上的 QoS 设置。以下三个标头被视为 JMSPriority、JMSDeliveryMode 和 JMSExpiration。您可以提供全部或仅提供其中一些。如果没有提供，Camel 将回退到使用端点中的值。因此，在使用此选项时，标头会覆盖端点中的值。与之相反，clearQosEnabled 选项将仅使用端点上设置的选项，而不使用来自消息标头中的值。	false	布尔值
camel.component.jms.priority	大于 1 的值指定发送时的消息优先级（其中 1 是最低优先级，9 是最高）。必须启用 explicitQosEnabled 选项才能使此选项生效。	4	整数

Name	描述	默认值	类型
camel.component.jms.pub-sub-no-local	指定是否禁止自己连接发布的消息的发送。	false	布尔值
camel.component.jms.queue-browse-strategy	在浏览队列时使用自定义 QueueBrowseStrategy。选项是一个 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，即当连接被刷新时，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
camel.component.jms.reply-to	提供显式 ReplyTo 目的地（覆盖消费者中 Message.getJMSReplyTo （）的所有传入值）。		字符串
camel.component.jms.reply-to-cache-level-name	在执行请求/通过 JMS 时，按名称为回复消费者设置缓存级别。这个选项仅在使用固定回复队列（而非临时）时才适用。Camel 默认将使用：CACHE_CONSUMER 用于专用或共享的 w/ replyToSelectorName。和 CACHE_SESSION 用于没有 replyToSelectorName 的共享。IBM WebSphere 等 JMS 代理可能需要设置 replyToCacheLevelName=CACHE_NONE 才能正常工作。注：如果使用临时队列，则不允许使用更高的值，如 CACHE_CONSUMER 或 CACHE_SESSION。		字符串
camel.component.jms.reply-to-concurrent-consumers	指定执行请求/通过 JMS 回复时的默认并发消费者数量。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。	1	整数
camel.component.jms.reply-to-delivery-persistent	指定是否默认使用持久性交付进行回复。	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，以便您可以在使用共享队列（也就是说，如果您不使用临时回复队列）时过滤来自其他回复的回复。		字符串

Name	描述	默认值	类型
camel.component.jms.reply-to-max-concurrent-consumers	指定在通过 JMS 使用请求/回复时的最大并发消费者数。另请参阅 maxMessagesPerTask 选项来控制线程的动态扩展/缩减。		整数
camel.component.jms.reply-to-on-timeout-max-concurrent-consumers	指定使用请求/通过 JMS 时超时时继续路由的最大并发消费者数。	1	整数
camel.component.jms.reply-to-override	在 JMS 消息中提供显式 ReplyTo 目的地，这将覆盖 replyTo 的设置。如果要将消息转发到远程队列，并从 ReplyTo 目的地接收回复消息，这非常有用。		字符串
camel.component.jms.reply-to-same-destination-allowed	JMS 使用者是否允许向消费者使用的同一目的地发送回复消息。这可防止出现无限循环，并通过消耗并向自己发送相同的消息。	false	布尔值
camel.component.jms.reply-to-type	允许明确指定在执行 request/reply 时要用于 replyTo 队列的策略类型。可能的值有：Temporary、share 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的队列。如需了解更多详细信息，请参阅 Camel JMS 文档，特别是在集群环境中运行时的影响的备注，以及共享回复队列的性能低于其 alternatives Temporary 和 Exclusive。		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 来覆盖此端点配置的超时值，因此具有每个消息独立的超时值。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 在执行请求/通过 JMS 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，如果发生超时时，您必须更快地响应，那么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

Name	描述	默认值	类型
camel.component.jms.stream-message-type-enabled	设置 StreamMessage 类型是否已启用。流类型的消息有效负载（如 files、InStream 等）将通过作为 BytesMessage 或 StreamMessage 发送。此选项控制将使用哪种类型。默认情况下，使用 BytesMessage 来强制整个消息有效负载读取到内存中。通过启用此选项，消息有效负载以块的形式读取到内存中，然后每个块都会写入 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。可使用的 durable 订阅名称通过 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 域)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	设置要创建的订阅名称。在带有共享或可升级订阅的主题（公共域）中应用。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： 每个订阅都只允许 1 个并发消费者（这是此消息侦听器容器的默认值），但一个共享订阅（需要 JMS 2.0）除外。		字符串
camel.component.jms.subscription-shared	设置是否共享订阅。可以使用的共享订阅名称可以通过 subscriptionName 属性指定。默认值为 false。把它设置为 true 以注册共享订阅，通常与 subscriptionName 值结合使用（除非您的消息监听程序类名称足以满足订阅名称）。请注意，共享的订阅也可能是危险的，因此此标志也可以与订阅相结合。仅在侦听主题(pub-sub 域)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执行器以使用消息。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 用户与 JMS 代理的有效连接。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 producer 也经过测试。	false	布尔值
camel.component.jms.time-to-live	发送消息时，指定消息的时间到时间（以毫秒为单位）。	-1	Long

Name	描述	默认值	类型
camel.component.jms.transacted	指定是否使用转换模式。	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，如果此标志被设置为 true，则 Spring JmsTemplate 将把 sessionTransacted 设置为 true，而 confirmMode 作为转换用于 InOut 操作。请注意：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，容器对这些值做出自己的决策。类似地，这些参数不会考虑本地管理的事务，因为 Spring JMS 在这种情况下在现有 JMS 会话上运行。在受管事务外运行时，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（除 XA 事务之外）时同步的本地 JMS 事务。这与主事务一起管理本地 JMS 事务（可能是原生 JDBC 事务）的影响，在主事务后 JMS 事务提交右侧的 JMS 事务。	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
camel.component.jms.transaction-name	要使用的事务的名称。		字符串
camel.component.jms.transaction-timeout	如果使用转换模式，事务的超时值（以秒为单位）。	-1	整数
camel.component.jms.transfer-exception	如果启用了且您使用 Request Reply messaging (InOut)，并且 Exchange 失败在消费者端，则原因例外将作为 javax.jms.ObjectMessage 发回的响应。如果客户端是 Camel，则返回的例外将重新箭头。这样，您可以使用 Camel JMS 作为路由中的桥接 - 例如，使用持久性队列来启用强大的路由。请注意，如果您也启用了 transferExchange，这个选项将具有优先权。caught 异常需要可以被序列化。消费者端的原始例外可以嵌套在外部异常中，如 org.apache.camel.RuntimeCamelException。请谨慎使用它，因为数据正在使用 Java 对象序列化，要求接收者在类级别反序列化数据，这会强制在生产者和消费者之间进行强校准。	false	布尔值

Name	描述	默认值	类型
camel.component.jms.transfer-exchange	您可以在有线线上传输交换，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 header, Exchange properties, exchange exception.这要求对象可以序列化。Camel 将排除任何不可序列化的对象，并将它记录在 WARN 级别。您必须在制作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。请谨慎使用它，因为数据正在使用 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，这会强制在需要使用兼容 Camel 版本的生产者和消费者之间进行强大的协调。	false	布尔值
camel.component.jms.use-message-i-d-as-correlation-i-d	指定 JMSMessageID 是否应该始终用作 InOut 消息的 JMSCorrelationID。	false	布尔值
camel.component.jms.username	与 ConnectionFactory 一起使用的用户名。您还可以直接在 ConnectionFactory 上配置用户名/密码。		字符串
camel.component.jms.wait-for-provision-correlation-to-be-updated-counter	在执行 request/reply over JMS 以及启用了 useMessageIDAsCorrelationID 时，等待 provisional correlation id 被更新到实际关联 ID 的次数。	50	整数
camel.component.jms.wait-for-provision-correlation-to-be-updated-thread-sleeping-time	等待置备关联 ID 时每次休眠的时间间隔（以秒为单位）。选项是一个长类型。	100	Long

第 32 章 JPA

从 Camel 1.0 开始

支持生成者和消费者。

JPA 组件可让您使用 EJB 3 的 Java Persistence 架构(DSL)从持久性存储存储和检索 Java 对象。Java Persistence 架构(DSL)是一种标准接口层，它打包 Object/Relational Mapping (ORM)产品，如 Open the Hibernate, Hibernate, TopLink。

将以下依赖项添加到此组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jpa</artifactId>
  <version>3.20.1.redhat-00050</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

32.1. 发送到端点

您可以通过将 Java 实体 Bean 发送到 JPA producer 端点，将其存储在数据库中。In 消息正文假定为实体 Bean （即，包含 @Entity 注解的 POJO）或实体 Bean 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。

如果正文不包含以上列出的类型之一，请在端点首先执行必要的转换前放置 Message Translator。

您还可以将名为Query 的查询、名为Query 或 nativeQuery 用于生成者。对于 参数的值，您可以使用简单表达式从 Message body、标头等检索参数值。这些查询可用于通过 SELECT JPQL/SQL 语句来检索一组数据，并使用 UPDATE/DELETE JPQL/SQL 语句执行批量更新/删除。请注意，如果您执行 名为 Query 的 UPDATE/DELETE，则您需要将 useExecuteUpdate 指定为 true，因为 camel 不会查看命名的 查询 与查询和 原生Query 不同。

32.2. 从端点消耗

从 JPA consumer 端点消耗消息会删除（或更新）数据库中的实体 Bean。这可让您将数据库表用作

逻辑队列：消费者从队列中获取消息，然后删除/更新它们以逻辑地将其从队列中删除。

如果您不想在处理实体 Bean 时删除它（在路由完成后），您可以在 URI 中指定 consumeDelete=false。这将导致实体处理每个轮询。

如果您想对实体执行一些更新，将其标记为已处理（例如，从将来的查询中排除它），您可以使用 @Consumed 注解方法，它会在实体 Bean 被处理时调用它。

您可以使用 @PreConsumed，它会在处理前在实体 bean 上调用（在路由之前）。

如果您消耗了大量行(100K+)，且遇到 OutOfMemory 问题，您应该将 maxResults 设置为 sensable 值。

32.3. URI 格式

```
jpa:entityClassName[?options]
```

对于发送到端点，entityClassName 是可选的。如果指定，它可以帮助 Type Converter来确保正文是正确的类型。

为消耗，entityClassName 是必须的。

32.4. 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

32.4.1. 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。

32.4.2. 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 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。

32.4.3. 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
alias (common)	将别名映射到 JPA 实体类。然后可在端点 URI 中使用别名（而不是完全限定的类名称）。		Map
entityManagerFactory (common)	使用 EntityManagerFactory。强烈建议您进行配置。		EntityManagerFactory
加入事务 (common)	camel-jpa 组件默认将加入事务。您可以使用此选项关闭此选项，例如，如果您使用 LOCAL_RESOURCE 并加入事务无法与您的 JPA 供应商一起使用。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。请注意，在大多数情况下，加入事务应设为 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

32.4.4. 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：

32.4.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
entityType (common)	**所需的** 实体类名称。		类

32.4.4.2. 查询参数(44 参数)

Name	描述	默认值	类型
加入事务 (common)	camel-jpa 组件默认将加入事务。您可以使用此选项关闭此选项，例如，如果您使用 LOCAL_RESOURCE 并加入事务无法与您的 JPA 供应商一起使用。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)	使用命名的查询：		字符串
nativeQuery (common)	使用自定义原生查询。在使用原生查询时，您可能还希望使用 resultClass 选项。		字符串
persistenceUnit (common)	**必需** 默认使用的 JPA 持久性单元。	camel	字符串
query (common)	使用自定义查询，请执行以下操作：		字符串
resultClass (common)	定义返回的有效负载的类型（we will call entityManager.createNativeQuery (nativeQuery, resultClass)而不是 entityManager.createNativeQuery (nativeQuery)。如果没有这个选项，我们将返回对象数组。在消耗数据时，仅与原生查询结合使用时才有影响。		类
consumeDelete (consumer)	如果为 true，实体会在被使用后删除；如果为 false，则不会删除实体。	true	布尔值
consumeLockEntity (consumer)	指定在处理轮询结果时，是否在每个实体 bean 上设置专用锁定。	true	布尔值
deleteHandler (consumer)	使用自定义 DeleteHandler 在消费者处理交换后删除行。		DeleteHandler

Name	描述	默认值	类型
lockModeType (consumer)	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，用于定义每个轮询要收集的最大消息数。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许多数千个消息。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)	要配置是否在锁定时使用 NOWAIT，并静默跳过实体。	false	布尔值
Transacted (consumer)	是否以转换模式运行消费者，在处理整个批处理时，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。它应该是通用类型 java.util.Map，其中键是给定 JPA 查询的命名参数，值是您要选择它们的对应有效值。当它用于制作者时，可将简单表达式用作参数值。它允许您从消息正文、标头等检索参数值。		Map
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，则制作者将使用消息正文作为 key 和 entityType 作为类类型来查找单个实体。这可用而不是查询来查找单个实体。	false	布尔值
flushOnSend (producer)	在实体 Bean 保留后清除 EntityManager。	true	布尔值
remove (producer)	表示使用 entityManager.remove (entity)。	false	布尔值
useExecuteUpdate (producer)	配置在制作者执行查询时使用 executeUpdate （）。当使用 INSERT、UPDATE 或 DELETE 语句作为命名查询时，您需要将这个选项指定为 'true'。		布尔值
usePersist (producer)	表示使用 entityManager.persist (entity)而不是 entityManager.merge (entity)。注意：EntityManager.persist (entity)不适用于分离实体（其中 EntityManager 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 将使用标头 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哪些实体管理器。	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。请注意，在大多数情况下，加入事务应设为 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

32.5. 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) 常量：link: JPA_PARAMETERS_HEADER	将查询参数作为 Exchange 标头传递的替代方法。		Map

32.6. 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，这通常是您想要的。

例如，您可以实例化一个 JPA 组件来引用 myEMFactory 实体管理器工厂，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 上配置它，如上所示。只有在存在不确定的情况时，您只需要这样做，在这种情况下，Camel 将记录一个 WARN。

32.7. 配置 TRANSACTIONMANAGER

JpaComponent 会自动从 Registry 中查找 TransactionManager。如果 Camel 找不到注册的任何 TransactionManager 实例，它将查找 TransactionTemplate，并尝试从中提取 TransactionManager。

如果 registry 中没有 TransactionTemplate，JpaEndpoint 将自动创建自己的 TransactionManager 实例，这通常是您想要的。

如果找到了超过 TransactionManager 的单个实例，Camel 将记录 WARN。在这种情况下，您可能想要实例化和明确配置引用 myTransactionManager 事务管理器的 JPA 组件，如下所示：

```
<bean id="jpa" class="org.apache.camel.component.jpa.JpaComponent">
  <property name="entityManagerFactory" ref="myEMFactory"/>
  <property name="transactionManager" ref="myTransactionManager"/>
</bean>
```

32.8. 使用带有命名查询的消费者

对于仅使用所选实体，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");
```

32.9. 使用带有查询的消费者

对于仅使用所选实体，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");
```

32.10. 使用带有原生查询的消费者

对于仅使用所选实体，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 查询选项，您将收到消息正文中的对象数组。

32.11. 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，您需要将 useExecuteUpdate 选项指定为 true 来执行 UPDATE/DELETE 语句作为命名的查询。

32.12. 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");
```

32.13. 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");
```

如果您使用原生查询选项而不指定 resultClass，您将在消息正文中收到对象数组。

32.14. 使用 JPA-BASED IDEMPOTENT 仓库

EIP 模式中的 Idempotent Consumer 用于过滤掉重复的消息。提供了基于 JPA 的幂等存储库。

使用基于 JPA 的幂等存储库。

流程

1. 在 persistence.xml 文件中设置 persistence-unit。
2. 设置 org.springframework.orm.jpa.JpaTemplate，供 org.apache.camel.processor.idempotent.jpa.JpaMessageIdRepository 使用。
3. 配置错误格式宏： snippet: java.lang.IndexOutOfBoundsException: Index: 20, Size: 20
4. 将幂等存储库配置为 org.apache.camel.processor.idempotent.jpa.JpaMessageIdRepository。
5. 在 Spring XML 文件中创建 JPA 幂等存储库，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

如果您直接在 IDE 中运行 此组件 的测试，而不是通过 Maven，您可以看到如下例外：

```
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。要克服这一点，您需要启用 Open Liberty 的动态字节代码增强。例如，假设 Camel 中使用的当前 Open the 版本为 2.2.1，要在 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
```

32.15. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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。然后可在端点 URI 中使用别名（而不是完全限定的类名称）。		Map

Name	描述	默认值	类型
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
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 组件默认将加入事务。您可以使用此选项关闭此选项，例如，如果您使用 LOCAL_RESOURCE 并加入事务无法与您的 JPA 供应商一起使用。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。请注意，在大多数情况下，加入事务应设为 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

Name	描述	默认值	类型
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 33 章 JSLT

从 Camel 3.1 开始

仅支持生成者

JSLT 组件允许您使用 JSLT 表达式处理 JSON 消息。在对 JSON 进行 JSON 转换或查询数据时，这可能是理想的选择。

将以下依赖项添加到此组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jslt</artifactId>
  <version>3.20.1.redhat-00050</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

33.1. URI 格式

```
jslt:specName[?options]
```

其中 specName 是要调用的规格的 classpath-local URI，或者远程规格的完整 URL （例如 file://folder/myfile.vm）。

33.2. 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

33.2.1. 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。

33.2.2. 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。

33.2.3. 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)	是否允许从标头使用资源模板（默认为 false）。启用此功能允许通过消息标头指定动态模板。但是，如果标头来自恶意用户，则可能会被视为潜在的安全漏洞，因此请小心使用。	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
功能 （高级）	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

33.2.4. 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：

33.2.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
resourceUri (producer)	资源 **所需的** 路径。您可以使用前缀：classpath, file, http, ref, 或 bean. classpath, 文件和 http 使用这些协议 (classpath 为 default)。ref 将查找 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

33.2.4.2. 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。可以启用此选项以完全访问当前 Exchange 和 CamelContext。这样做会产生潜在的安全风险，因为这会打开对 CamelContext API 完整功能的访问。	false	布尔值
allowTemplateFromHeader (producer)	是否允许从标头使用资源模板（默认为 false）。启用此功能允许通过消息标头指定动态模板。但是，如果标头来自恶意用户，则可能会被视为潜在的安全漏洞，因此请小心使用。	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，则映射程序将使用 USE_BIG_DECIMAL_FOR_FLOATS in serialization 功能。	false	布尔值
ObjectMapper (producer)	设置要使用的自定义 JSON 对象映射程序。		ObjectMapper
prettyPrint (common)	如果为 true，则输出消息中的 JSON 是用户友善打印的。	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

33.3. 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) 常数：HEADER_JSLT_STRING	JSLT 模板作为字符串。		字符串

Name	描述	默认值	类型
CamelJsltResourceUri (producer) 常数：HEADER_JSLT_RESOURCE_URI	资源 URI。		字符串

33.4. 将值传递给 JSLT

在对正文应用 JSLT 表达式时，Camel 可以以变量形式提供交换信息。Exchange 中的可用变量有：

name	value
标头	**In** message 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。

所有不能使用 Jackson 转换为 json 的值都会被拒绝，且不会在 jslt 表达式中可用。

例如，名为"type"的标头和交换属性"instance"可以访问，如下所示

```
{
  "type": $headers.type,
  "instance": $exchange.properties.instance
}
```

33.5. SAMPLES

示例如下所示。

```
from("activemq:My.Queue").
  to("jslt:com/acme/MyResponse.json");
```

和基于文件的资源：

```
from("activemq:My.Queue").
  to("jslt:file://myfolder/MyResponse.json?contentCache=true").
  to("activemq:Another.Queue");
```

您还可以指定组件通过标头动态使用哪些 JSLT 表达式，例如：

```
from("direct:in").
  setHeader("CamelJsltResourceUri").constant("path/to/my/spec.json").
  to("jslt:dummy?allowTemplateFromHeader=true");
```

或者通过标头发送整个 jslt 表达式：（用于查询的理解）

```
from("direct:in").
  setHeader("CamelJsltString").constant(".published").
  to("jslt:dummy?allowTemplateFromHeader=true");
```

将交换属性传递给 jslt 表达式，如下所示

```
from("direct:in").
  to("jslt:com/acme/MyResponse.json?allowContextMapAll=true");
```

33.6. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	是否允许从标头使用资源模板（默认为 false）。启用此功能允许通过消息标头指定动态模板。但是，如果标头来自恶意用户，则可能会被视为潜在的安全漏洞，因此请小心使用。	false	布尔值

Name	描述	默认值	类型
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
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 34 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。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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

34.1. URI 格式

```
kafka:topic[?options]
```

34.2. 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

34.2.1. 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。

34.2.2. 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。

34.3. 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（例如：新的 Kafka 属性没有反映在 Camel 配置中），则必须为 kafka consumer 或 kafka producer 设置额外的属性，属性必须使用 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，也可以是指向代理子集的 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的字符串，以帮助追踪调用。它应该以逻辑方式识别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 将标头过滤到或从 Camel 消息过滤。		HeaderFilterStrategy
reconnectBackoffMaxMs (common)	当重新连接到重复无法连接的代理时，等待的最大时间（毫秒）。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。计算 backoff 后，会添加 20% 随机 jitter 以避免连接停滞。	1000	整数
shutdownTimeout (common)	超时时间（毫秒）以毫秒为单位等待消费者或生成者关闭并终止其 worker 线程。	30000	int
allowManualCommit (consumer)	是否允许通过 KafkaManualCommit 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，这将允许最终用户访问这个 API，并通过 Kafka 使用者执行手动偏移提交。	false	布尔值
autoCommitEnable (consumer)	如果为 true，请定期提交到 ZooKeeper，以偏移已由消费者获取的信息。当进程失败时，将使用此提交偏移，作为新消费者开始的位置。	true	布尔值
autoCommitIntervalMs (consumer)	消费者偏移提交到 zookeeper 的频率。	5000	整数
autoCommitOnStop (consumer)	消费者停止时是否执行显式自动提交，以确保代理有来自最近使用的消息的提交。这需要打开选项 autoCommitEnable。可能的值有： sync、syncsync 或 none。sync 是默认值。 Enum 值： • 同步 • async • none	同步	字符串
autoOffsetReset (consumer)	当 ZooKeeper 中没有初始偏移量时，或者偏移没有范围： earliest : 自动将偏移重置为最早的偏移 latest：自动将偏移重置为最新的偏移失败：抛出异常。 Enum 值： • 最新 • earliest • none	最新	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且失败时会发生什么。如果选项为 false，则消费者将继续到下一个消息并处理它。如果选项为 true，则消费者会发现出导致故障的消息的偏移，然后重新尝试处理此消息。但是，如果每次绑定都失败，这可能会导致意外处理同一消息，例如一个 poison 消息。因此，建议处理这一点，例如使用 Camel 的错误处理程序。	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)	自动检查所消耗的记录的 CRC32。这样可确保不会发生在线或磁盘崩溃信息。此检查增加了一些开销，因此在出现极端性能的情况下可能会禁用它。	true	布尔值
commitTimeoutMs (consumer)	代码将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。如果在超时超时前未收到响应，如果需要，或者如果重试耗尽，则请求将重新发送。	40000	整数
consumersCount (consumer)	连接到 kafka 服务器的消费者数量。每个消费者都在一个单独的线程上运行，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中的第一个消息大于这个值，则服务器为 fetch 请求返回的最大数据量并非绝对的最大值。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，使用者并行执行多个获取。	52428800	整数
fetchMinBytes (consumer)	服务器为获取请求返回的最小数据量。如果数据不足，请求将在回答请求前等待该数量的数据累积。	1	整数
fetchWaitMaxMs (consumer)	如果没有足够的数据立即满足 fetch.min.bytes，服务器将在回答获取请求前阻止的最大时间。	500	整数
GroupId (consumer)	标识此消费者所属的消费者进程组的字符串。通过设置相同的组 id 多个进程表示它们都是同一消费者组的一部分。消费者需要这个选项。		字符串

Name	描述	默认值	类型
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，则消费者被视为静态成员，这意味着在任何消费者组中都只允许具有此 ID 的实例。这可以与更大的会话超时结合使用，以避免因为临时不可用（如进程重启）导致组重新平衡。如果没有设置，则消费者将作为动态成员加入组，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)	在使用 Kafka 的组管理功能时，心跳到消费者协调器的预期时间。心跳用于确保消费者的会话保持活动状态，并在新消费者加入或离开组时促进重新平衡。该值必须小于 session.timeout.ms，但通常不应设置高于该值的 1/3。可以调整它，以控制正常重新平衡的预期时间。	3000	整数
keyDeserializer (consumer)	实施 Deserializer 接口的密钥反序列化类。	org.apache.kafka.common.serialization.StringDeserializer	字符串
maxPartitionFetchBytes (consumer)	服务器将返回的最大每个分区的数据量。用于请求的最大内存总量为 192.168.1.0/24partitions max.partition.fetch.bytes。这个大小必须至少与服务器允许的最大消息大小相同，否则生成者可以发送大于消费者的消息。如果发生这种情况，使用者可能会卡住尝试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调用 poll （）的最大延迟。这会在获取更多记录前，在消费者闲置的时间上放置上限。如果在超时过期前没有调用 poll （），则消费者被视为失败，组将重新平衡，以便将分区重新分配给另一个成员。		Long
maxPollRecords (consumer)	单个调用中返回到 poll （）中返回的最大记录数。	500	整数
offsetRepository (consumer)	用于本地存储主题的每个分区的偏移程序库。定义一个将禁用自动提交。		StateRepository

Name	描述	默认值	类型
partitionAssignor (consumer)	使用组管理时，客户端将使用分区分配策略的类名称，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 在轮询新消息时异常，则该怎么办。除非在端点级别上配置了显式值，否则默认使用组件配置中的值。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，然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 RETRY，以便消费者再次重试同一消息，STOP 将停止消费者（如果消费者应该再次消耗消息，则应手动启动/重新启动）。 Enum 值： • 丢弃 • ERROR_HANDLER • RECONNECT • RETRY • STOP	ERROR_HANDLER	PollOnError
pollTimeoutMs (consumer)	轮询 KafkaConsumer 时使用的超时。	5000	Long
resumeStrategy (consumer)	这个选项允许用户设置自定义恢复策略。恢复策略会在分配分区时执行（例如：连接或重新连接）。它允许实现自定义如何恢复操作，并更灵活替代 seekTo 和 offsetRepository 机制。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使用手动提交选项进行评估。		KafkaConsumerResumeStrategy
seekTo (consumer)	设置 KafkaConsumer 将在启动时从开始或结束读取：start : read from end : read from end this is replace the previous attributes seekToBeginning。 Enum 值： • 开始 • end		字符串

Name	描述	默认值	类型
sessionTimeoutMs (consumer)	使用 Kafka 组管理功能时检测失败的超时。	10000	整数
specificAvroReader (consumer)	这可让特定的 Avro reader 与 Confluent Platform schema registry 和 io.confluent.kafka.serializers.KafkaAvroDeserializer 搭配使用。这个选项仅适用于 Confluent Platform （不适用于标准 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)	用于实现 Deserializer 接口的值反序列化类。	org.apache.kafka.common.serialization.StringDeserializer	字符串
kafkaManualCommitFactory (consumer (advanced))	用于创建 KafkaManualCommit 实例的 **Autowired** Factory。当执行与开箱即用的默认实现中分离的手动提交时，如果需要插入自定义 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从 Kafka 代理抛出的异常。		PollExceptionStrategy
bufferMemorySize (producer)	生成者可用于缓冲区等待发送到服务器的内存总量字节。如果发送记录比服务器发送的速度快，则生成者会根据 block.on.buffer.full 指定的首选项阻止或抛出异常。此设置应该与制作者将使用的总内存对应，而不是硬绑定，因为不是生成者使用的所有内存进行缓冲。一些额外的内存将用于压缩（如果启用了压缩），以及维护动态请求。	33554432	整数

Name	描述	默认值	类型
compressionCodec (producer)	这个参数允许您为这个制作者生成的所有数据指定压缩 codec。有效值为 none、gzip 和 snappy。 Enum 值： • none • gzip • snappy • lz4	none	字符串
connectionMaxIdleMs (producer)	在此配置指定的毫秒数后关闭闲置连接。	540000	整数
deliveryTimeoutMs (producer)	在调用 send （）后报告成功或失败的时间上限。这限制了在发送前记录延迟、从代理等待确认的时间（如预期）以及可重新检索失败所需的时间。	120000	整数
enableIdempotence (producer)	如果设置为 'true'，则制作者将确保在流中写入每个消息的一个副本。如果 'false'，则制作者重试可能会在流中写入重试的消息的副本。如果设置为 true，则此选项需要 max.in.flight.requests.per.connection 设置为 1，并且重试不能为零，且其他 acks 必须设为 'all'。	false	布尔值
headerSerializer (producer)	使用自定义 KafkaHeaderSerializer 来序列化 kafka 标头值。		KafkaHeaderSerializer
key (producer)	记录键（如果没有指定密钥，则为 null）。如果配置了这个选项，它将优先于标头 KafkaConstants114KEY。		字符串
keySerializer (producer)	键的序列化类（如果没有给出任何信息，则默认为消息）。	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的任何记录分组到单个批处理请求中。通常，只有在记录到达比发送的速度相比，才会在负载下发生。然而，在某些情况下，客户端可能希望减少请求数，即使在负载下也是如此。此设置通过添加少量人类延迟来实现这一目的，而不是立即发送记录，让生成者最多等待给定延迟，以允许将发送其他记录，以便将发送发送。这可以象 TCP 中的 Nagle 算法类似。此设置在批处理延迟上提供了上限：一旦为分区获取 batch.size，无论此设置如何，都会立即发送它。但是，如果我们对这个分区的总字节少于这个分区，我们将"闲置"等待更多记录显示。此设置默认为 0 （例如，无延迟）。例如，设置 linger.ms=5 可减少发送的请求数量，但对负载中发送的记录增加最多 5ms 的延迟数。	0	整数
maxBlockMs (producer)	配置控制发送到 kafka 的时长将阻断。这些方法可能会因为多个原因而被阻止。例如：缓冲区已满，元数据不可用。此配置对获取元数据、键和值序列化、在执行 send （）时分区和分配缓冲区内存的总时间实施最大限制。如果是 partitionsFor （），此配置在等待元数据时强制实施最长时间阈值。	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，如果此设置设定为大于 1，且有失败，则可能会因为重试重试而重新排序消息（例如，如果启用了重试）。	5	整数
maxRequestSize (producer)	请求的最大大小。这也在最大记录大小上有效上限。请注意，服务器对记录大小有自己的上限，它们可能与这个值不同。此设置将限制生成者将在单个请求中发送的记录数量，以避免发送大量请求。	1048576	整数
metadataMaxAgeMs (producer)	我们强制刷新元数据的时间（以毫秒为单位），即使我们没有看到任何分区领导更改来主动发现任何新的代理或分区。	300000	整数
metricReporters (producer)	用作指标报告器的类列表。实施 MetricReporter 接口允许插入将创建新指标创建通知的类。JmxReporter 始终被包含以注册 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)	在子主题间分区消息的 partitioner 类。默认分区程序基于密钥的哈希。	org.apache.kafka.clients.producer.internals.DefaultPartitioner	字符串
partitionKey (producer)	将记录发送到的分区（如果没有指定分区，则为 null）。如果配置了这个选项，它将优先于标头 KafkaConstants114PARTITION_KEY。		整数
producerBatchSize (producer)	每当将多个记录发送到同一分区时，生产者会尝试将记录批处理到较少的请求中。这有助于客户端和服务器的性能。此配置以字节为单位控制默认批处理大小。不尝试批处理大于这个大小的批处理记录。发送到代理的请求将包含多个批处理，每个带有可用数据的分区都会进行批处理。小批处理大小会减少吞吐量，并可能会降低吞吐量（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，因为我们始终以额外的记录为指定批处理大小分配缓冲区。	16384	整数
queueBufferingMaxMessages (producer)	在使用 async 模式时可以排队生成者的最大未消息数量，然后才能阻止生成者，或者必须丢弃数据。	10000	整数
receiveBufferBytes (producer)	读取数据时使用的 TCP 接收缓冲区(SO_RCVBUF)的大小。	65536	整数
reconnectBackoffMs (producer)	尝试重新连接到给定主机前等待的时间。这可避免在严格的循环中重复连接到主机。此 backoff 适用于消费者向代理发送的所有请求。	50	整数
recordMetadata (producer)	生成者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。该列表存储在带有键 KafkaConstantsHQKAFKA_RECORDMETA 的标头中。	true	布尔值

Name	描述	默认值	类型
requestRequiredAcks (producer)	确认生成者要求接收领导数量，然后才能考虑请求完成。这控制发送的记录的持久性。以下设置比较常见：acks=0 如果设为零，则制作者将根本不等待服务器的任何确认。记录将立即添加到套接字缓冲区中并被视为发送。不保证服务器在这种情况下收到记录，重试的配置不会生效（因为客户端通常不知道任何故障）。为每个记录的偏移量始终设置为 -1。这意味着领导会将记录写入本地日志，但不会等待所有后续者的确认。在这种情况下，领导会在确认记录后立即失败，但在后续者复制前它会丢失。acks=all 表示领导将等待完整的 in-sync 副本集确认记录。这样可保证，只要至少有一个同步副本保持活跃状态，就不会丢失记录。这是最强的保证。 Enum 值： ● -1 ● 0 ● 1 ● all	1	字符串
requestTimeoutMs (producer)	在将错误发送到客户端前，代理将等待尝试满足 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任何记录，并显示潜在的临时错误。请注意，这个重试与客户端在收到错误时重新处理记录不同。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，第一个失败且被重试，但第二个成功，则可能会首先出现第二个记录。	0	整数
retryBackoffMs (producer)	每次重试前，制作者会刷新相关主题的元数据，以查看是否已选择新的领导。由于领导选举机制需要一些时间，此属性指定制作者在刷新元数据前等待的时间。	100	整数
sendBufferBytes (producer)	套接字写入缓冲区大小。	131072	整数
valueSerializer (producer)	消息的序列化类。	org.apache.kafka.common.serialization.StringSerializer	字符串

Name	描述	默认值	类型
workerpool (producer)	要在 kafka 服务器确认使用异步非阻塞处理从 KafkaProducer 发送的消息后，使用自定义 worker 池继续路由交换。如果使用这个选项，则必须处理线程池的生命周期，以便在不再需要时关闭池。		ExecutorService
workerPoolCoreSize (producer)	kafka 服务器后用于继续路由交换的 worker 池的核心线程数量确认使用异步非阻塞处理从 KafkaProducer 发送的消息。	10	整数
workerPoolMaxSize (producer)	kafka 服务器后，用于继续路由交换的 worker 池的最大线程数量确认使用异步非阻塞处理从 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，以使用扩展 vanilla Kafka 客户端的逻辑创建实例。		KafkaClientFactory
Sync (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 文档中称为 schema.registry.url。这个选项仅适用于 Confluent Platform （不适用于标准 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 Consumer interceptors 的类，则需要类实施 org.apache.kafka.clients.consumer.ConsumerInterceptor 请注意，如果您在消费者上使用 Producer 拦截器，它将在运行时抛出类多播异常。		字符串
kerberosBeforeReloginMinTime (security)	在刷新尝试之间登录线程睡眠时间。	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，第一个匹配主体名称的规则被用来将其映射到短名称。列表中的任何后续规则都会被忽略。默认情况下，形式为 {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 百分比。	0.05	å�Œ
kerberosRenewWindowFactor (security)	登录线程将休眠，直到达到最后刷新到票据的过期时间的窗口因子，此时它将尝试续订票据。	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。	明文	字符串
sslCipherSuites (security)	密码套件列表。这是用于使用 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。如果配置了，则在其他 SSL 端点参数之前应用它。注意： Kafka 只支持从文件位置加载密钥存储，因此在 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 会被默认启用。		字符串
sslEndpointAlgorithm (security)	端点识别算法，使用服务器证书验证服务器主机名。	https	字符串
sslKeymanagerAlgorithm (security)	SSL 连接的密钥管理器工厂使用的算法。默认值为为 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)	密钥存储文件中私钥的密码。对于客户端，这是可选的。		字符串
sslKeystoreLocation (security)	密钥存储文件的位置。这对客户端是可选的，可用于客户端的双向身份验证。		字符串
sslKeystorePassword (security)	密钥存储文件的存储密码。这对客户端是可选的，且仅在配置了 ssl.keystore.location 时才需要。		字符串
sslKeystoreType (security)	密钥存储文件的文件格式。对于客户端，这是可选的。默认值为 JKS。	JKS	字符串
SSLProtocol (security)	用于生成 SSLContext 的 SSL 协议。默认设置为 TLS，对于大多数情况来说是理想的选择。最近的 JVM 中允许的值是 TLS、TLSv1.1 和 TLSv1.2。SSL、SSLv2 和 SSLv3 可能在较旧的 JVM 中被支持，但由于已知的安全漏洞，不建议使用它们的使用。		字符串
sslProvider (security)	用于 SSL 连接的安全提供程序的名称。默认值为 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。默认值为为 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

34.4. 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：

34.4.1. 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。在消费者上，您可以使用逗号分隔多个主题。生产者只能向单个主题发送消息。		字符串

34.4.2. 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（例如：新的 Kafka 属性没有反映在 Camel 配置中），则必须为 kafka consumer 或 kafka producer 设置额外的属性，属性必须使用 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，也可以是指向代理子集的 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的字符串，以帮助追踪调用。它应该以逻辑方式识别发出请求的应用程序。		字符串
HeaderFilterStrategy (common)	使用自定义 HeaderFilterStrategy 将标头过滤到或从 Camel 消息过滤。		HeaderFilterStrategy
reconnectBackoffMaxMs (common)	当重新连接到重复无法连接的代理时，等待的最大时间（毫秒）。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。计算 backoff 后，会添加 20% 随机 jitter 以避免连接停滞。	1000	整数
shutdownTimeout (common)	超时时间（毫秒）以毫秒为单位等待消费者或生成者关闭并终止其 worker 线程。	30000	int
allowManualCommit (consumer)	是否允许通过 KafkaManualCommit 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，这将允许最终用户访问这个 API，并通过 Kafka 使用者执行手动偏移提交。	false	布尔值
autoCommitEnable (consumer)	如果为 true，请定期提交到 ZooKeeper，以偏移已由消费者获取的信息。当进程失败时，将使用此提交偏移，作为新消费者开始的位置。	true	布尔值

Name	描述	默认值	类型
autoCommitIntervalMs (consumer)	消费者偏移提交到 zookeeper 的频率。	5000	整数
autoCommitOnStop (consumer)	消费者停止时是否执行显式自动提交，以确保代理有来自最近使用的消息的提交。这需要打开选项 autoCommitEnable。可能的值有： sync、syncsync 或 none。sync 是默认值。 Enum 值： • 同步 • async • none	同步	字符串
autoOffsetReset (consumer)	当 ZooKeeper 中没有初始偏移量时，或者偏移没有范围： earliest : 自动将偏移重置为最早的偏移 latest：自动将偏移重置为最新的偏移失败：抛出异常。 Enum 值： • 最新 • earliest • none	最新	字符串
breakOnFirstError (consumer)	此选项控制消费者处理交换且失败时会发生什么。如果选项为 false，则消费者将继续到下一个消息并处理它。如果选项为 true，则消费者会发现出导致故障的消息的偏移，然后重新尝试处理此消息。但是，如果每次绑定都失败，这可能会导致意外处理同一消息，例如一个 poison 消息。因此，建议处理这一点，例如使用 Camel 的错误处理程序。	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)	自动检查所消耗的记录的 CRC32。这样可确保不会发生在线或磁盘崩溃信息。此检查增加了一些开销，因此在出现极端性能的情况下可能会禁用它。	true	布尔值
commitTimeoutMs (consumer)	代码将等待同步提交完成的最长时间（以毫秒为单位）。	5000	Long

Name	描述	默认值	类型
consumerRequestTimeoutMs (consumer)	配置控制客户端等待请求响应的最长时间。如果在超时超时前未收到响应，如果需要，或者如果重试耗尽，则请求将重新发送。	40000	整数
consumersCount (consumer)	连接到 kafka 服务器的消费者数量。每个消费者都在一个单独的线程上运行，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中的第一个消息大于这个值，则服务器为 fetch 请求返回的最大数据量并非绝对的最大值。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，使用者并行执行多个获取。	52428800	整数
fetchMinBytes (consumer)	服务器为获取请求返回的最小数据量。如果数据不足，请求将在回答请求前等待该数量的数据累积。	1	整数
fetchWaitMaxMs (consumer)	如果没有足够的数据立即满足 fetch.min.bytes，服务器将在回答获取请求前阻止的最大时间。	500	整数
GroupId (consumer)	标识此消费者所属的消费者进程组的字符串。通过设置相同的组 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，则消费者被视为静态成员，这意味着在任何消费者组中都只允许具有此 ID 的实例。这可以与更大的会话超时结合使用，以避免因为临时不可用（如进程重启）导致组重新平衡。如果没有设置，则消费者将作为动态成员加入组，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)	在使用 Kafka 的组管理功能时，心跳到消费者协调器的预期时间。心跳用于确保消费者的会话保持活动状态，并在新消费者加入或离开组时促进重新平衡。该值必须小于 session.timeout.ms，但通常不应设置高于该值的 1/3。可以调整它，以控制正常重新平衡的预期时间。	3000	整数

Name	描述	默认值	类型
keyDeserializer (consumer)	实施 Deserializer 接口的密钥反序列化类。	org.apache.kafka.common.serialization.StringDeserializer	字符串
maxPartitionFetchBytes (consumer)	服务器将返回的最大每个分区的数据量。用于请求的最大内存总量为 192.168.1.0/24partitions max.partition.fetch.bytes。这个大小必须至少与服务器允许的最大消息大小相同，否则生成者可以发送大于消费者的消息。如果发生这种情况，使用者可能会卡住尝试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调用 poll （）的最大延迟。这会在获取更多记录前，在消费者闲置的时间上放置上限。如果在超时过期前没有调用 poll （），则消费者被视为失败，组将重新平衡，以便将分区重新分配给另一个成员。		Long
maxPollRecords (consumer)	单个调用中返回到 poll （）中返回的最大记录数。	500	整数
offsetRepository (consumer)	用于本地存储主题的每个分区的偏移程序库。定义一个将禁用自动提交。		StateRepository
partitionAssignor (consumer)	使用组管理时，客户端将使用分区分配策略的类名称，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 在轮询新消息时异常，则该怎么办。除非在端点级别上配置了显式值，否则默认使用组件配置中的值。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，然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 RETRY，以便消费者再次重试同一消息，STOP 将停止消费者（如果消费者应该再次消耗消息，则应手动启动/重新启动）。 Enum 值： • 丢弃 • ERROR_HANDLER • RECONNECT • RETRY • STOP	ERROR_HANDLER	PollOnError
pollTimeoutMs (consumer)	轮询 KafkaConsumer 时使用的超时。	5000	Long
resumeStrategy (consumer)	这个选项允许用户设置自定义恢复策略。恢复策略会在分配分区时执行（例如：连接或重新连接）。它允许实现自定义如何恢复操作，并更灵活替代 seekTo 和 offsetRepository 机制。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使用手动提交选项进行评估。		KafkaConsumerResumeStrategy
seekTo (consumer)	设置 KafkaConsumer 将在启动时从开始或结束读取：start : read from end : read from end this is replace the previous attributes seekToBeginning。 Enum 值： • 开始 • end		字符串
sessionTimeoutMs (consumer)	使用 Kafka 组管理功能时检测失败的超时。	10000	整数
specificAvroReader (consumer)	这可让特定的 Avro reader 与 Confluent Platform schema registry 和 io.confluent.kafka.serializers.KafkaAvroDeserializer 搭配使用。这个选项仅适用于 Confluent Platform （不适用于标准 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)	用于实现 Deserializer 接口的值反序列化类。	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。当执行与开箱即用的默认实现中分离的手动提交时，如果需要插入自定义 KafkaManualCommit 实例。		KafkaManualCommitFactory
bufferMemorySize (producer)	生成者可用于缓冲区等待发送到服务器的内存总量字节。如果发送记录比服务器发送的速度快，则生成者会根据 block.on.buffer.full 指定的首选项阻止或抛出异常。此设置应该与制作者将使用的总内存对应，而不是硬绑定，因为不是生成者使用的所有内存进行缓冲。一些额外的内存将用于压缩（如果启用了压缩），以及维护动态请求。	33554432	整数

Name	描述	默认值	类型
compressionCodec (producer)	这个参数允许您为这个制作者生成的所有数据指定压缩 codec。有效值为 none、gzip 和 snappy。 Enum 值： • none • gzip • snappy • lz4	none	字符串
connectionMaxIdleMs (producer)	在此配置指定的毫秒数后关闭闲置连接。	540000	整数
deliveryTimeoutMs (producer)	在调用 send （）后报告成功或失败的时间上限。这限制了在发送前记录延迟、从代理等待确认的时间（如预期）以及可重新检索失败所需的时间。	120000	整数
enableIdempotence (producer)	如果设置为 'true'，则制作者将确保在流中写入每个消息的一个副本。如果 'false'，则制作者重试可能会在流中写入重试的消息的副本。如果设置为 true，则此选项需要 max.in.flight.requests.per.connection 设置为 1，并且重试不能为零，且其他 acks 必须设为 'all'。	false	布尔值
headerSerializer (producer)	使用自定义 KafkaHeaderSerializer 来序列化 kafka 标头值。		KafkaHeaderSerializer
key (producer)	记录键（如果没有指定密钥，则为 null）。如果配置了这个选项，它将优先于标头 KafkaConstants114KEY。		字符串
keySerializer (producer)	键的序列化类（如果没有给出任何信息，则默认为消息）。	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的任何记录分组到单个批处理请求中。通常，只有在记录到达比发送的速度相比，才会在负载下发生。然而，在某些情况下，客户端可能希望减少请求数，即使在负载下也是如此。此设置通过添加少量人类延迟来实现这一目的，而不是立即发送记录，让生成者最多等待给定延迟，以允许将发送其他记录，以便将发送发送。这可以象 TCP 中的 Nagle 算法类似。此设置在批处理延迟上提供了上限：一旦为分区获取 batch.size，无论此设置如何，都会立即发送它。但是，如果我们对这个分区的总字节少于这个分区，我们将"闲置"等待更多记录显示。此设置默认为 0 （例如，无延迟）。例如，设置 linger.ms=5 可减少发送的请求数量，但对负载中发送的记录增加最多 5ms 的延迟数。	0	整数
maxBlockMs (producer)	配置控制发送到 kafka 的时长将阻断。这些方法可能会因为多个原因而被阻止。例如：缓冲区已满，元数据不可用。此配置对获取元数据、键和值序列化、在执行 send （）时分区和分配缓冲区内存的总时间实施最大限制。如果是 partitionsFor （），此配置在等待元数据时强制实施最长时间阈值。	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，如果此设置设定为大于 1，且有失败，则可能会因为重试重试而重新排序消息（例如，如果启用了重试）。	5	整数
maxRequestSize (producer)	请求的最大大小。这也在最大记录大小上有效上限。请注意，服务器对记录大小有自己的上限，它们可能与这个值不同。此设置将限制生成者将在单个请求中发送的记录数量，以避免发送大量请求。	1048576	整数
metadataMaxAgeMs (producer)	我们强制刷新元数据的时间（以毫秒为单位），即使我们没有看到任何分区领导更改来主动发现任何新的代理或分区。	300000	整数
metricReporters (producer)	用作指标报告器的类列表。实施 MetricReporter 接口允许插入将创建新指标创建通知的类。JmxReporter 始终被包含以注册 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)	在子主题间分区消息的 partitioner 类。默认分区程序基于密钥的哈希。	org.apache.kafka.clients.producer.internals.DefaultPartitioner	字符串
partitionKey (producer)	将记录发送到的分区（如果没有指定分区，则为 null）。如果配置了这个选项，它将优先于标头 KafkaConstants114PARTITION_KEY。		整数
producerBatchSize (producer)	每当将多个记录发送到同一分区时，生产者会尝试将记录批处理到较少的请求中。这有助于客户端和服务器的性能。此配置以字节为单位控制默认批处理大小。不尝试批处理大于这个大小的批处理记录。发送到代理的请求将包含多个批处理，每个带有可用数据的分区都会进行批处理。小批处理大小会减少吞吐量，并可能会降低吞吐量（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，因为我们始终以额外的记录为指定批处理大小分配缓冲区。	16384	整数
queueBufferingMaxMessages (producer)	在使用 async 模式时可以排队生成者的最大未消息数量，然后才能阻止生成者，或者必须丢弃数据。	10000	整数
receiveBufferBytes (producer)	读取数据时使用的 TCP 接收缓冲区(SO_RCVBUF)的大小。	65536	整数
reconnectBackoffMs (producer)	尝试重新连接到给定主机前等待的时间。这可避免在严格的循环中重复连接到主机。此 backoff 适用于消费者向代理发送的所有请求。	50	整数
recordMetadata (producer)	生成者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。该列表存储在带有键 KafkaConstantsHQKAFKA_RECORDMETA 的标头中。	true	布尔值

Name	描述	默认值	类型
requestRequiredAcks (producer)	确认生成者要求接收领导数量，然后才能考虑请求完成。这控制发送的记录的持久性。以下设置比较常见：acks=0 如果设为零，则制作者将根本不等待服务器的任何确认。记录将立即添加到套接字缓冲区中并被视为发送。不保证服务器在这种情况下收到记录，重试的配置不会生效（因为客户端通常不知道任何故障）。为每个记录的偏移量始终设置为 -1。这意味着领导会将记录写入本地日志，但不会等待所有后续者的确认。在这种情况下，领导会在确认记录后立即失败，但在后续者复制前它会丢失。acks=all 表示领导将等待完整的 in-sync 副本集确认记录。这样可保证，只要至少有一个同步副本保持活跃状态，就不会丢失记录。这是最强的保证。 Enum 值： ● -1 ● 0 ● 1 ● all	1	字符串
requestTimeoutMs (producer)	在将错误发送到客户端前，代理将等待尝试满足 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任何记录，并显示潜在的临时错误。请注意，这个重试与客户端在收到错误时重新处理记录不同。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，第一个失败且被重试，但第二个成功，则可能会首先出现第二个记录。	0	整数
retryBackoffMs (producer)	每次重试前，制作者会刷新相关主题的元数据，以查看是否已选择新的领导。由于领导选举机制需要一些时间，此属性指定制作者在刷新元数据前等待的时间。	100	整数
sendBufferBytes (producer)	套接字写入缓冲区大小。	131072	整数
valueSerializer (producer)	消息的序列化类。	org.apache.kafka.common.serialization.StringSerializer	字符串

Name	描述	默认值	类型
workerpool (producer)	要在 kafka 服务器确认使用异步非阻塞处理从 KafkaProducer 发送的消息后，使用自定义 worker 池继续路由交换。如果使用这个选项，则必须处理线程池的生命周期，以便在不再需要时关闭池。		ExecutorService
workerPoolCoreSize (producer)	kafka 服务器后用于继续路由交换的 worker 池的核心线程数量确认使用异步非阻塞处理从 KafkaProducer 发送的消息。	10	整数
workerPoolMaxSize (producer)	kafka 服务器后，用于继续路由交换的 worker 池的最大线程数量确认使用异步非阻塞处理从 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，以使用扩展 vanilla Kafka 客户端的逻辑创建实例。		KafkaClientFactory
Sync (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 文档中称为 schema.registry.url。这个选项仅适用于 Confluent Platform （不适用于标准 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 Consumer interceptors 的类，则需要类实施 org.apache.kafka.clients.consumer.ConsumerInterceptor 请注意，如果您在消费者上使用 Producer 拦截器，它将在运行时抛出类多播异常。		字符串
kerberosBeforeReloginMinTime (security)	在刷新尝试之间登录线程睡眠时间。	60000	整数
kerberosInitCmd (security)	Kerberos kinit 命令路径。默认为 /usr/bin/kinit。	/usr/bin/kinit	字符串
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，第一个匹配主体名称的规则被用来将其映射到短名称。列表中的任何后续规则都会被忽略。默认情况下，形式为 {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

Name	描述	默认值	类型
kerberosRenewJitter (security)	添加到续订时间的随机 jitter 百分比。	0.05	å€Œ
kerberosRenewWindowFactor (security)	登录线程将休眠，直到达到最后刷新到票据的过期时间的窗口因子，此时它将尝试续订票据。	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。	明文	字符串
sslCipherSuites (security)	密码套件列表。这是用于使用 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。如果配置了，则在其他 SSL 端点参数之前应用它。注意： Kafka 只支持从文件位置加载密钥存储，因此在 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 会被默认启用。		字符串
sslEndpointAlgorithm (security)	端点识别算法，使用服务器证书验证服务器主机名。	https	字符串
sslKeymanagerAlgorithm (security)	SSL 连接的密钥管理器工厂使用的算法。默认值为为 Java 虚拟机配置的密钥管理器工厂算法。	SunX509	字符串
sslKeyPassword (security)	密钥存储文件中私钥的密码。对于客户端，这是可选的。		字符串
sslKeystoreLocation (security)	密钥存储文件的位置。这对客户端是可选的，可用于客户端的双向身份验证。		字符串

Name	描述	默认值	类型
sslKeystorePassword (security)	密钥存储文件的存储密码。这对客户端是可选的，且仅在配置了 ssl.keystore.location 时才需要。		字符串
sslKeystoreType (security)	密钥存储文件的文件格式。对于客户端，这是可选的。默认值为 JKS。	JKS	字符串
SSLProtocol (security)	用于生成 SSLContext 的 SSL 协议。默认设置为 TLS，对于大多数情况来说是理想的选择。最近的 JVM 中允许的值是 TLS、TLSv1.1 和 TLSv1.2。SSL、SSLv2 和 SSLv3 可能在较旧的 JVM 中被支持，但由于已知的安全漏洞，不建议使用它们的使用。		字符串
sslProvider (security)	用于 SSL 连接的安全提供程序的名称。默认值为 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。默认值为为 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***

34.5. 消息标头

34.5.1. 消费者标头

使用 Kafka 的信息时可使用以下标头。

标头常量	标头值	类型	描述
KafkaConstants.TOPIC	**"kafka.TOPIC"**	**字符串**	消息源自的主题
kafkaConstants.PARTITION	**"kafka.PARTITION"**	**整数**	存储消息的分区
KafkaConstants.OFFSET	**"kafka.OFFSET"**	**Long**	消息的偏移
KafkaConstants.KEY	**"kafka.KEY"**	**对象**	消息的密钥（如果已配置）
KafkaConstants.HEADERS	**"kafka.HEADERS"**	**org.apache.kafka.common.header.Headers**	记录标头
KafkaConstants.LAST_RECORD_BEFORE_COMMIT	**"kafka.LAST_RECORD_BEFORE_COMMIT"**	**布尔值**	提交前的最后一条记录（仅在 **autoCommitEnable** endpoint 参数为 **false**时可用）
KafkaConstants.LAST_POLL_RECORD	**"kafka.LAST_POLL_RECORD"**	**布尔值**	表示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**时可用）
KafkaConstants.MANUAL_COMMIT	**"CamelKafkaManualCommit"**	**KafkaManualCommit**	在使用 Kafka 消费者时，可用于强制手动偏移提交。

34.5.2. 生成者标头

在向 Kafka 发送消息前，您可以配置以下标头。

标头常量	标头值	类型	描述
KafkaConstants.KEY	**"kafka.KEY"**	**对象**	**必需** 消息的密钥，以确保所有相关消息都位于同一分区中
KafkaConstants.OVERRIDE_TOPIC	**"kafka.OVERRIDE_TOPIC"**	**字符串**	发送消息的主题(override 和 take priority)，标头不会被保留。

标头常量	标头值	类型	描述
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，则制作者将调整与提供的时间戳的记录，并且不会保留标头。
KafkaConstants.PARTITION_KEY	**"kafka.PARTITION_KEY"**	**整数**	明确指定分区

如果要向动态主题发送消息，则使用 KafkaConstants.OVERRIDE_TOPIC 用作未与消息一起发送的一次性标头，因为它在制作者中删除。

在消息发送到 Kafka 后，以下标头将可用

标头常量	标头值	类型	描述
KafkaConstants.KAFKA_RECORDMETA	**"org.apache.kafka.clients.producer.RecordMetadata"**	**List<RecordMetadata>**	元数据（仅在 **recordMetadata** 端点参数为 **true**时配置

34.6. 消费者错误处理

虽然 kafka 使用者轮询来自 kafka 代理的消息，但可能会出现错误。本节描述了什么情况以及您可以配置的内容。

在调用 Kafka poll API 时，消费者可能会抛出异常。例如，如果因为无效数据而无法反序列化消息，以及许多其他错误。这些错误采用 KafkaException 的形式，它们可以被重试 或不可 重试。可以重试的例外(RetriableException)将再次重试（之间有一个轮询超时）。所有其他异常都根据 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，然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 将使消费者再次重试轮询相同的消息
- STOP 将停止消费者（如果消费者应该再次消耗消息，则需要手动启动/重新启动）。

默认为 ERROR_HANDLER，它将让 Camel 的错误处理程序（若有配置）处理导致异常。然后继续轮询下一个消息。这个行为与 Camel 组件具有的 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，这允许控制上述策略的异常原因。

34.7. SAMPLES

34.7.1. 使用 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]}")
```

如果您需要使用来自多个主题的消息，您可以使用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，为主题名称提供模式，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]}")
```

使用 Kafka 中的消息时，您可以使用自己的偏移管理，且不会将这个管理委派给 Kafka。为了保持偏移，组件需要 StateRepository 实施，如 FileStateRepository。此 bean 应该在 registry 中提供。在这里如何使用它：

```
// Create the repository in which the Kafka offsets will be persisted
FileStateRepository repository = FileStateRepository.fileStateRepository(new
File("/path/to/repo.dat"));

// Bind this repository into the Camel registry
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");
    }
});
```

34.7.2. 生成信息到 Kafka

以下是向 Kafka 写入信息所需的最小路由。

```
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");
```

34.8. 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");
    }
});
```

34.9. 使用 KAFKA 幂等存储库

camel-kafka 库提供了一个基于 Kafka 主题的幂等存储库。

此存储库在 Kafka 主题中将所有更改广播到幂等状态(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是唯一的。

机制没有任何有关主题分区数量的要求；因为存储库同时从所有分区消耗。它还没有任何有关主题复制

因素的要求。

使用主题（例如，并行运行的不同机器上）的每个存储库实例控制其自身的消费者组，因此在使用相同主题的 10 个 Camel 进程集群中，控制其自身偏移。

在启动时，实例订阅该主题，并将偏移回开始，将缓存重建为最新状态。在一次轮询 轮询持续时间M前，缓存不会被认为是 0 个记录。在缓存热上或 30 秒发生之前，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，直到消费者捕获到主题的末尾为止。

请注意用于唯一检查的标头格式。默认情况下，它使用 Strings 作为数据类型。使用原语数字格式时，必须相应地对标头进行反序列化。例如，请检查以下示例。

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 生成者和消费者上的 **bootstrap.servers** 属性。如果没有设置 **consumerConfig** 和 **producerConfig**，则将其用作缩写。如果使用，此组件将为生成者和消费者应用可配置默认配置。
producerConfig	设置 Kafka producer 将用于广播更改的属性。覆盖 **bootstrapServers**，因此必须定义 Kafka **bootstrap.servers** 属性本身
consumerConfig	设置 Kafka 消费者将使用的属性，该消费者从主题填充缓存。覆盖 **bootstrapServers**，因此必须定义 Kafka **bootstrap.servers** 属性本身
maxCacheSize	最近使用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对等点后如何相对于发送缓存操作消息的幂等消费者实例。默认值为 100 ms。 如果明确设置这个值，请注意，远程缓存存活度和此存储库消费者和 Kafka 代理之间的网络流量卷之间有一个权衡。缓存温过程还依赖于一个轮询，它获取任何内容 - 这表示该流已消耗到当前点。如果轮询持续时间对于主题上发送消息的速率过长，则缓存可能无法热热，并且会相对于对等点运行处于不一致的状态，直到捕获为止。

可以通过定义 主题和 bootstrapServers 来实例化存储库，或者明确定义 producerConfig 和 consumerConfig 属性集来启用 SSL/SASL 等功能。要使用，此存储库必须手动放置在 Camel 注册表中，也可以作为 bean 在 Spring/Blueprint 中进行注册，因为它为 CamelContext 感知。

用法示例如下：

```
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>
```

在使用带有数字标识符的 idempotency 时，可以选择 3 个替代方案。第一个是使用来自 org.apache.camel.component.kafka.serde.KafkaSerdeHelper 的静态方法 number Header 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程序来执行转换：

```
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，也可以在处理器中执行此操作：

```
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);
```

34.10. 使用 KAFKA 使用者手动提交

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。这需要在 KafkaComponent 或端点上将 allowManualCommit 选项设置为 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();
}
```

这将强制同步提交，该提交将在 Kafka 上确认提交，或者抛出异常失败。您可以通过配置带有 'DefaultKafkaManualAsyncCommitFactory' 实现的 KafkaManualCommitFactory 来使用异步提交。

然后，提交将使用 kafka 异步提交 api 在下一个消费者循环中进行。请注意，分区中的记录必须由唯一的线程处理并提交。如果没有，这会导致非一致的行为。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 上配置自定义 KafkaManualCommitFactory，以创建自定义实现的实例。

34.11. KAFKA 标头传播

使用 Kafka 中的消息时，标头将自动传播到 camel 交换标头。生成由同一行为 - 特定交换的 camel 标头支持的流将被传播到 kafka 消息标头。

由于 kafka 标头只允许 byte[] 值，以便 camel Exchange 标头被序列化为 bytes[]，否则将跳过标头。支持以下标头值类型：string ,Integer,Long, Double, 布尔值,byte[]. 注： 所有标头生成的 从 kafka 到 camel 的交换默认都会包括值 byte[]。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 进行过滤。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，必须手动放置在 Camel registry 中，也可以作为 bean 在 Spring/Blueprint 中进行注册，因为它为 CamelContext 感知。

34.12. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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（例如：新的 Kafka 属性没有反映在 Camel 配置中），则必须为 kafka consumer 或 kafka producer 设置额外的属性，属性必须使用 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 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，这将允许最终用户访问这个 API，并通过 Kafka 使用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。当进程失败时，将使用此提交偏移，作为新消费者开始的位置。	true	布尔值
camel.component.kafka.auto-commit-interval-ms	消费者偏移提交到 zookeeper 的频率。	5000	整数
camel.component.kafka.auto-commit-on-stop	消费者停止时是否执行显式自动提交，以确保代理有来自最近使用的消息的提交。这需要打开选项 autoCommitEnable。可能的值有： sync、syncsync 或 none。sync 是默认值。	同步	字符串
camel.component.kafka.auto-offset-reset	当 ZooKeeper 中没有初始偏移量时，或者偏移没有范围： earliest : 自动将偏移重置为最早的偏移 latest : 自动将偏移重置为最新的偏移失败：抛出异常。	最新	字符串

Name	描述	默认值	类型
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且失败时会发生什么。如果选项为 false，则消费者将继续到下一个消息并处理它。如果选项为 true，则消费者会发现出导致故障的消息的偏移，然后重新尝试处理此消息。但是，如果每次绑定都失败，这可能会导致意外处理同一消息，例如一个 poison 消息。因此，建议处理这一点，例如使用 Camel 的错误处理程序。	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，也可以是指向代理子集的 VIP。这个选项在 Kafka 文档中称为 bootstrap.servers。		字符串
camel.component.kafka.buffer-memory-size	生成者可用于缓冲区等待发送到服务器的内存总量字节。如果发送记录比服务器发送的速度快，则生成者会根据 block.on.buffer.full 指定的首选项阻止或抛出异常。此设置应该与制作者将使用的总内存对应，而不是硬绑定，因为不是生成者使用的所有内存进行缓冲。一些额外的内存将用于压缩（如果启用了压缩），以及维护动态请求。	33554432	整数
camel.component.kafka.check-crcs	自动检查所消耗的记录的 CRC32。这样可确保不会发生在线或磁盘崩溃信息。此检查增加了一些开销，因此在出现极端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的字符串，以帮助追踪调用。它应该以逻辑方式识别发出请求的应用程序。		字符串
camel.component.kafka.commit-timeout-ms	代码将等待同步提交完成的最长时间（以毫秒为单位）。选项是 java.lang.Long 类型。	5000	Long
camel.component.kafka.compression-codec	这个参数允许您为这个制作者生成的所有数据指定压缩 codec。有效值为 none、gzip 和 snappy。	none	字符串

Name	描述	默认值	类型
camel.component.kafka.configuration	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	在此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。如果在超时超时前未收到响应，如果需要，或者如果重试耗尽，则请求将重新发送。	40000	整数
camel.component.kafka.consumers-count	连接到 kafka 服务器的消费者数量。每个消费者都在一个单独的线程上运行，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 （）后报告成功或失败的时间上限。这限制了在发送前记录延迟、从代理等待确认的时间（如预期）以及可重新检索失败所需的时间。	120000	整数
camel.component.kafka.enable-idempotence	如果设置为 'true'，则制作者将确保在流中写入每个消息的一个副本。如果 'false'，则制作者重试可能会在流中写入重试的消息的副本。如果设置为 true，则此选项需要 max.in.flight.requests.per.connection 设置为 1，并且重试不能为零，且其他 acks 必须设为 '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中的第一个消息大于这个值，则服务器为 fetch 请求返回的最大数据量并非绝对的最大值。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，使用者并行执行多个获取。	52428800	整数
camel.component.kafka.fetch-min-bytes	服务器为获取请求返回的最小数据量。如果数据不足，请求将在回答请求前等待该数量的数据累积。	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，服务器将在回答获取请求前阻止的最大时间。	500	整数

Name	描述	默认值	类型
camel.component.kafka.group-id	标识此消费者所属的消费者进程组的字符串。通过设置相同的组 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，则消费者被视为静态成员，这意味着在任何消费者组中都只允许具有此 ID 的实例。这可以与更大的会话超时结合使用，以避免因为临时不可用（如进程重启）导致组重新平衡。如果没有设置，则消费者将作为动态成员加入组，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 将标头过滤到或从 Camel 消息过滤。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	在使用 Kafka 的组管理功能时，心跳到消费者协调器的预期时间。心跳用于确保消费者的会话保持活动状态，并在新消费者加入或离开组时促进重新平衡。该值必须小于 session.timeout.ms，但通常不应设置高于该值的 1/3。可以调整它，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 Consumer interceptors 的类，则需要类实施 org.apache.kafka.clients.consumer.ConsumerInterceptor 请注意，如果您在消费者上使用 Producer 拦截器，它将在运行时抛出类多播异常。		字符串
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，以使用扩展 vanilla Kafka 客户端的逻辑创建实例。选项是一个 org.apache.camel.component.kafka.KafkaClientFactory 类型。		KafkaClientFactory

Name	描述	默认值	类型
camel.component.kafka.kafka-manual-commit-factory	用于创建 KafkaManualCommit 实例的工厂。当执行与开箱即用的默认实现中分离的手动提交时，如果需要插入自定义 KafkaManualCommit 实例。选项是一个 org.apache.camel.component.kafka.KafkaManualCommitFactory 类型。		KafkaManualCommitFactory
camel.component.kafka.kerberos-before-relogin-min-time	在刷新尝试之间登录线程睡眠时间。	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，第一个匹配主体名称的规则被用来将其映射到短名称。列表中的任何后续规则都会被忽略。默认情况下，形式为 {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 百分比。		å�Œ
camel.component.kafka.kerberos-renew-window-factor	登录线程将休眠，直到达到最后刷新到票据的过期时间的窗口因子，此时它将尝试续订票据。		å�Œ
camel.component.kafka.key	记录键（如果没有指定密钥，则为 null）。如果配置了这个选项，它将优先于标头 KafkaConstants114KEY。		字符串
camel.component.kafka.key-deserializer	实施 Deserializer 接口的密钥反序列化类。	org.apache.kafka.common.serialization.StringDeserializer	字符串

Name	描述	默认值	类型
camel.component.kafka.key-serializer	键的序列化类（如果没有给出任何信息，则默认为消息）。	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的任何记录分组到单个批处理请求中。通常，只有在记录到达比发送的速度相比，才会在负载下发生。然而，在某些情况下，客户端可能希望减少请求数，即使在负载下也是如此。此设置通过添加少量人类延迟来实现这一目的，而不是立即发送记录，让生成者最多等待给定延迟，以允许将发送其他记录，以便将发送发送。这可以象 TCP 中的 Nagle 算法类似。此设置在批处理延迟上提供了上限：一旦为分区获取 batch.size，无论此设置如何，都会立即发送它。但是，如果我们对这个分区的总字节少于这个分区，我们将"闲置"等待更多记录显示。此设置默认为 0 （例如，无延迟）。例如，设置 linger.ms=5 可减少发送的请求数量，但对负载中发送的记录增加最多 5ms 的延迟数。	0	整数
camel.component.kafka.max-block-ms	配置控制发送到 kafka 的时长将阻断。这些方法可能会因为多个原因而被阻止。例如：缓冲区已满，元数据不可用。此配置对获取元数据、键和值序列化、在执行 send （）时分区和分配缓冲区内存的总时间实施最大限制。如果是 partitionsFor （），此配置在等待元数据时强制实施最长时间阈值。	60000	整数
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，如果此设置设定为大于 1，且有失败，则可能会因为重试重试而重新排序消息（例如，如果启用了重试）。	5	整数
camel.component.kafka.max-partition-fetch-bytes	服务器将返回的最大每个分区的数据量。用于请求的最大内存总量为 192.168.1.0/24partitions max.partition.fetch.bytes。这个大小必须至少与服务器允许的最大消息大小相同，否则生成者可以发送大于消费者的消息。如果发生这种情况，使用者可能会卡住尝试在某个分区中获取大量消息。	1048576	整数

Name	描述	默认值	类型
camel.component.kafka.max-poll-interval-ms	使用消费者组管理时调用 poll （）的最大延迟。这会在获取更多记录前，在消费者闲置的时间上放置上限。如果在超时过期前没有调用 poll （），则消费者被视为失败，组将重新平衡，以便将分区重新分配给另一个成员。选项是 java.lang.Long 类型。		Long
camel.component.kafka.max-poll-records	单个调用中返回到 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。这也在最大记录大小上有效上限。请注意，服务器对记录大小有自己的上限，它们可能与这个值不同。此设置将限制生成者将在单个请求中发送的记录数量，以避免发送大量请求。	1048576	整数
camel.component.kafka.metadata-max-age-ms	我们强制刷新元数据的时间（以毫秒为单位），即使我们没有看到任何分区领导更改来主动发现任何新的代理或分区。	300000	整数
camel.component.kafka.metric-reporters	用作指标报告器的类列表。实施 MetricReporter 接口允许插入将创建新指标创建通知的类。JmxReporter 始终被包含以注册 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
camel.component.kafka.offset-repository	用于本地存储主题的每个分区的偏移程序库。定义一个将禁用自动提交。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，客户端将使用分区分配策略的类名称，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
camel.component.kafka.partition-key	将记录发送到的分区（如果没有指定分区，则为 null）。如果配置了这个选项，它将优先于标头 KafkaConstants114PARTITION_KEY。		整数
camel.component.kafka.partitioner	在子主题间分区消息的 partitioner 类。默认分区程序基于密钥的哈希。	org.apache.kafka.clients.producer.internals.DefaultPartitioner	字符串
camel.component.kafka.poll-exception-strategy	要将自定义策略与使用者一起使用，以控制如何在池信息时处理从 Kafka 代理抛出的异常。选项是一个 org.apache.camel.component.kafka.PollExceptionStrategy 类型。		PollExceptionStrategy
camel.component.kafka.poll-on-error	如果 kafka 在轮询新消息时异常，则该怎么办。除非在端点级别上配置了显式值，否则默认使用组件配置中的值。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，然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 RETRY，以便消费者再次重试同一消息，STOP 将停止消费者（如果消费者应该再次消耗消息，则应手动启动/重新启动）。		PollOnError
camel.component.kafka.poll-timeout-ms	轮询 KafkaConsumer 时使用的超时。选项是 java.lang.Long 类型。	5000	Long
camel.component.kafka.producer-batch-size	每当将多个记录发送到同一分区时，生产者会尝试将记录批处理到较少的请求中。这有助于客户端和服务器的性能。此配置以字节为单位控制默认批处理大小。不尝试批处理大于这个大小的批处理记录。发送到代理的请求将包含多个批处理，每个带有可用数据的分区都会进行批处理。小批处理大小会减少吞吐量，并可能会降低吞吐量（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，因为我们始终以额外的记录为指定批处理大小分配缓冲区。	16384	整数
camel.component.kafka.queue-buffering-max-messages	在使用 async 模式时可以排队生成者的最大未消息数量，然后才能阻止生成者，或者必须丢弃数据。	10000	整数

Name	描述	默认值	类型
camel.component.kafka.receive-buffer-bytes	读取数据时使用的 TCP 接收缓冲区(SO_RCVBUF)的大小。	65536	整数
camel.component.kafka.reconnect-backoff-max-ms	当重新连接到重复无法连接的代理时，等待的最大时间（毫秒）。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。计算 backoff 后，会添加 20% 随机 jitter 以避免连接停滞。	1000	整数
camel.component.kafka.reconnect-backoff-ms	尝试重新连接到给定主机前等待的时间。这可避免在严格的循环中重复连接到主机。此 backoff 适用于消费者向代理发送的所有请求。	50	整数
camel.component.kafka.record-metadata	生成者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。该列表存储在带有键 KafkaConstantsHQKAFKA_RECORDMETA 的标头中。	true	布尔值
camel.component.kafka.request-required-acks	确认生成者要求接收领导数量，然后才能考虑请求完成。这控制发送的记录的持久性。以下设置比较常见：acks=0 如果设为零，则制作者将根本不等待服务器的任何确认。记录将立即添加到套接字缓冲区中并被视为发送。不保证服务器在这种情况下收到记录，重试的配置不会生效（因为客户端通常不知道任何故障）。为每个记录的偏移量始终设置为 -1。这意味着领导会将记录写入本地日志，但不会等待所有后续者的确认。在这种情况下，领导会在确认记录后立即失败，但在后续者复制前它会丢失。acks=all 表示领导将等待完整的 in-sync 副本集确认记录。这样可保证，只要至少有一个同步副本保持活跃状态，就不会丢失记录。这是最强的保证。	1	字符串
camel.component.kafka.request-timeout-ms	在将错误发送到客户端前，代理将等待尝试满足 request.required.acks 要求的时间。	30000	整数
camel.component.kafka.resume-strategy	这个选项允许用户设置自定义恢复策略。恢复策略会在分配分区时执行（例如：连接或重新连接）。它允许实现自定义如何恢复操作，并更灵活替代 seekTo 和 offsetRepository 机制。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使用手动提交选项进行评估。选项是一个 org.apache.camel.component.kafka.consumer.support.KafkaConsumerResumeStrategy 类型。		KafkaConsumerResumeStrategy

Name	描述	默认值	类型
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任何记录，并显示潜在的临时错误。请注意，这个重试与客户端在收到错误时重新处理记录不同。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，第一个失败且被重试，但第二个成功，则可能会首先出现第二个记录。	0	整数
camel.component.kafka.retry-backoff-ms	每次重试前，制作者会刷新相关主题的元数据，以查看是否已选择新的领导。由于领导选举机制需要一些时间，此属性指定制作者在刷新元数据前等待的时间。	100	整数
camel.component.kafka.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 文档中称为 schema.registry.url。这个选项仅适用于 Confluent Platform （不适用于标准 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。	明文	字符串
camel.component.kafka.seek-to	设置 KafkaConsumer 将在启动时从开始或结束读取：start : read from end : read from end this is replace the previous attributes 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 组管理功能时检测失败的超时。	10000	整数

Name	描述	默认值	类型
camel.component.kafka.shutdown-timeout	超时时间（毫秒）以毫秒为单位等待消费者或生成者关闭并终止其 worker 线程。	30000	整数
camel.component.kafka.specific-avro-reader	这可让特定的 Avro reader 与 Confluent Platform schema registry 和 io.confluent.kafka.serializers.KafkaAvroDeserializer 搭配使用。这个选项仅适用于 Confluent Platform （不适用于标准 Apache Kafka）。	false	布尔值
camel.component.kafka.ssl-cipher-suites	密码套件列表。这是用于使用 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。如果配置了，则在其他 SSL 端点参数之前应用它。注意： Kafka 只支持从文件位置加载密钥存储，因此在 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 会被默认启用。		字符串
camel.component.kafka.ssl-endpoint-algorithm	端点识别算法，使用服务器证书验证服务器主机名。	https	字符串
camel.component.kafka.ssl-key-password	密钥存储文件中私钥的密码。对于客户端，这是可选的。		字符串
camel.component.kafka.ssl-keymanager-algorithm	SSL 连接的密钥管理器工厂使用的算法。默认值为为 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，可用于客户端的双向身份验证。		字符串
camel.component.kafka.ssl-keystore-password	密钥存储文件的存储密码。这对客户端是可选的，且仅在配置了 ssl.keystore.location 时才需要。		字符串

Name	描述	默认值	类型
camel.component.kafka.ssl-keystore-type	密钥存储文件的文件格式。对于客户端，这是可选的。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，对于大多数情况来说是理想的选择。最近的 JVM 中允许的值是 TLS、TLSv1.1 和 TLSv1.2。SSL、SSLv2 和 SSLv3 可能在较旧的 JVM 中被支持，但由于已知的安全漏洞，不建议使用它们的使用。		字符串
camel.component.kafka.ssl-provider	用于 SSL 连接的安全提供程序的名称。默认值为 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。默认值为为 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
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

Name	描述	默认值	类型
camel.component.kafka.value-deserializer	用于实现 Deserializer 接口的值反序列化类。	org.apache.kafka.common.serialization.StringDeserializer	字符串
camel.component.kafka.value-serializer	消息的序列化类。	org.apache.kafka.common.serialization.StringSerializer	字符串
camel.component.kafka.worker-pool	要在 kafka 服务器确认使用异步非阻塞处理从 KafkaProducer 发送的消息后，使用自定义 worker 池继续路由交换。如果使用这个选项，则必须处理线程池的生命周期，以便在不再需要时关闭池。选项是一个 java.util.concurrent.ExecutorService 类型。		ExecutorService
camel.component.kafka.worker-pool-core-size	kafka 服务器后用于继续路由交换的 worker 池的核心线程数量确认使用异步非阻塞处理从 KafkaProducer 发送的消息。	10	整数
camel.component.kafka.worker-pool-max-size	kafka 服务器后，用于继续路由交换的 worker 池的最大线程数量确认使用异步非阻塞处理从 KafkaProducer 发送的消息。	20	整数

第 35 章 KAMELET

支持生成者和消费者

Kamelet 组件支持使用 Endpoint 语义与 Camel Route 模板 引擎交互。

35.1. URI 格式

```
kamelet:templateId/routeId[?options]
```

35.2. 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

35.2.1. 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。

35.2.2. 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。

35.3. 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
位置 (common)	Kamelets 在文件系统中的位置。可以使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
block (producer)	如果向没有活跃消费者的 kamelet 端点发送消息，则我们可以告知制作者阻止并等待消费者变为 active。	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

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
routeTemplateLoaderListener (advanced)	**Autowired** To 插件当 Kamelet 组件从外部资源加载 Kamelets 时的自定义监听程序。		RouteTemplateLoaderListener

35.4. 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：

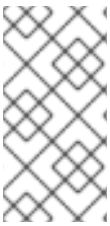
35.4.1. 路径参数(2 参数)

Name	描述	默认值	类型
templateId (common)	**必需** Route Template ID。		字符串
routeId (common)	路由 ID。默认值通知 ：如果没有提供，则 ID 将自动生成。		字符串

35.4.2. 查询参数(8 参数)

Name	描述	默认值	类型
位置 (common)	Kamelet 的位置，可以从文件系统、classpath 等指定为资源。位置无法使用通配符，且必须引用包含扩展名的文件，例如 file:/etc/foo-kamelet.xml。		字符串

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
exceptionHandler (consumer (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，则我们可以告知制作者阻止并等待消费者变为 active。	true	布尔值
failIfNoConsumers (producer)	当发送到没有活跃消费者的 kamelet 端点时，生成者是否应该失败。	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



注意

kamelet 端点是 lenient，这意味着端点接受传递给引擎的额外参数，并在路由材料时消耗。

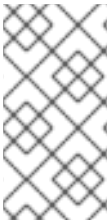
35.5. DISCOVERY（发现）

如果没有找到 Route 模板，kamelet 端点会尝试从文件系统加载相关的 kamelet 定义（默认为 classpath:/kamelets）。默认解析机制预期 kamelet 文件具有扩展名 .kamelet.yaml。

35.6. SAMPLES

kamelets 可以像标准 Camel 组件一样使用。例如，假设我们已创建了路由模板，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者处理器，我们需要识别路由的输入和输出端点，这通过使用 kamele:source 为输出端点标记输入端点和 kamelet:sink。

然后，可以实例化并调用模板，如下所示：

```
from("direct:setMyBody")
    .to("kamelet:setMyBody?bodyValue=myKamelet");
```

在 scenes 后，Kamelet Y 组件执行以下操作：

1. 它实例化了由给定 templateId 路径参数标识的 Route 模板（本例中为 setBody）
2. 它将象 直接 组件一样，并将当前路由连接到材料化。

如果您需要以编程方式完成此操作，它类似于：

```
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");
```

35.7. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	如果向没有活跃消费者的 kamelet 端点发送消息，则我们可以告知制作者阻止并等待消费者变为 active。	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
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

Name	描述	默认值	类型
camel.component.kamelet.location	Kamelets 在文件系统中的位置。可以使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	要插件自定义监听程序，以便在从外部资源加载 Kamelets 时提供自定义监听程序。选项是 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 36 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 中的任何支持的语言执行脚本。通过使用组件来执行语言脚本，它允许更动态的路由功能。例如，通过使用 Routing Slip 或 Dynamic Router EIPs，您可以将消息发送到脚本动态定义的语言端点。

此组件开箱即用 camel-core 的方框，因此不需要额外的 JAR。只有选择的语言需要包括额外的 Camel 组件，比如使用 Groovy 或 JavaScript 语言。

36.1. URI 格式

```
language://languageName[:script][?options]
```

您可以使用与 Camel 中 其他语言 支持的相同表示法为脚本引用外部资源。

```
language://languageName:resource:scheme:location][?options]
```

36.2. 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

36.2.1. 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。

36.2.2. 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。

36.3. 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
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

36.4. 端点选项

语言端点使用 URI 语法进行配置：

```
language:languageName:resourceUri
```

使用以下路径和查询参数：

36.4.1. 路径参数(2 参数)

Name	描述	默认值	类型
languageName (producer)	**必需** 设置要使用的语言名称。 Enum 值： • bean • 常数 • exchangeProperty • file • groovy • header • javascript • jsonPath • MVEL • ognl • ref • simple • spel • sql • terser • 令牌化 • 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

36.4.2. 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。可以启用此选项以完全访问当前 Exchange 和 CamelContext。这样做会产生潜在的安全风险，因为这会打开对 CamelContext API 完整功能的访问。	false	布尔值
binary (producer)	脚本是二进制内容还是文本内容。默认情况下，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以及重复使用 Notice 重新利用脚本可能会导致对一个 Camel org.apache.camel.Exchange 处理一个 Camel org.apache.camel.Exchange 的副作用。	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
convert (producer)	脚本的结果是否应用作消息正文。这个选项是默认的 true。	true	布尔值

36.5. 消息标头

以下消息标头可用于影响组件的行为

标头	描述
CamelLanguageScript	要在标头中提供的脚本。优先于在端点上配置的脚本。

36.6. 例子

例如，您可以使用 Simple 语言来消息转换消息。

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);
```

36.7. 从资源载入脚本

您可以为在 endpoint uri 或 Exchange.LANGUAGE_SCRIPT 标头中加载的脚本指定资源 uri。uri 必须以以下方案之一开始：file:、classpath: 或 http ：

默认情况下，脚本会加载一次并缓存。但是，您可以禁用 contentCache 选项，并在每次评估时载入脚本。例如，如果在磁盘上更改了文件 myscript.txt，则使用更新的脚本：

您可以使用 "resource:" 前缀来引用与 Camel 中 其他语言 类似的资源，如下所示。

36.8. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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

Name	描述	默认值	类型
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 37 章 LOG

仅支持生成者

Log 组件日志记录消息会交换底层日志记录机制。

Camel 使用 SLF4J，允许您通过 SLF4J 配置日志：

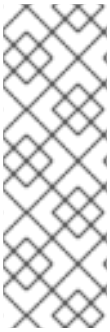
- Log4j
- Logback
- Java Util Logging

37.1. URI 格式

```
log:loggingCategory[?options]
```

其中 loggingCategory 是要使用的日志记录类别的名称。您可以以以下格式将查询选项附加到 URI 中，

```
?option=value&option=value&...
```



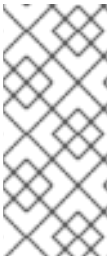
注意

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，则使用 loggingCategory 创建日志记录器实例。

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 还附带了 吞吐量 日志记录器，它会在指定 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 的详情。

37.2. 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

37.2.1. 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。

37.2.2. 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。

37.3. 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，则默认为 DefaultExchangeFormatter。		ExchangeFormatter

37.4. 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：

37.4.1. 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 **所需的** 名称。		字符串

37.4.2. 查询参数(27 参数)

Name	描述	默认值	类型
groupActiveOnly (producer)	如果为 true，如果没有接收新消息的时间间隔（如果为 false），则无论消息流量是什么，都会隐藏统计数据。	true	布尔值
groupDelay (producer)	设置统计数据的初始延迟（以 millis 为单位）。		Long
groupInterval (producer)	如果指定将按这个时间间隔对消息统计进行分组（以 millis 为单位）。		Long
groupSize (producer)	指定吞吐量日志记录的组大小的整数。		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
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，在日志中屏蔽敏感信息，如密码或密码短语。		布尔值
marker (producer)	要使用的可选标记名称。		字符串

Name	描述	默认值	类型
exchangeFormatter (advanced)	使用自定义交换格式：		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 (formatting)	打开所有选项的快速选项。（如果使用，则必须手动设置 maxChars）。	false	布尔值
showAllProperties (formatting)	显示所有交换属性（内部和外部）。	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)	如果交换具有 caught 异常，则会显示异常信息（无堆栈跟踪）。caught 异常作为属性存储在交换上（使用键 org.apache.camel.Exchange114EXCEPTION_CAUGHT），而一个 doCatch 可以捕获例外。	false	布尔值
showException (formatting)	如果交换具有例外，则显示异常消息（无 stacktrace）。	false	布尔值
showExchangeId (formatting)	显示唯一的交换 ID。	false	布尔值
showExchangePattern (formatting)	显示消息交换模式（或 MEP 用于短）。	true	布尔值
showFiles （格式）	如果已启用 Camel 将输出文件。	false	布尔值
showFuture （格式）	如果已启用的 Camel 将在将来的对象上等待它完成，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（仅限自定义）。使用 showAllProperties 显示内部和外部属性。	false	布尔值

Name	描述	默认值	类型
showStack Trace (formatting)	如果交换具有例外，显示堆栈追踪。仅在启用 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（如 java.io.InputStream）。如果您启用这个选项，那么您稍后可能无法访问消息正文，因为该日志记录器已读取该流。要修复这一点，您必须使用流缓存。	false	布尔值
skipBodyLineSeparator (formatting)	在记录消息正文时是否跳过行分隔符。这允许在一行中记录消息正文，将此选项设置为 false 将保留来自正文的任何行分隔符，然后记录正文。	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

37.5. 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>
```

37.6. 带有格式示例的常规日志记录器

在以下路由中，我们将进入的顺序记录在 INFO 级别，然后再处理顺序。

```
from("activemq:orders").
    to("log:com.mycompany.order?showAll=true&multiline=true").to("bean:processOrder");
```

37.7. 带有 GROUPSIZE 样本的 THROUGHPUT LOGGER

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");
```

37.8. 带有 GROUPINTERVAL 样本的吞吐量日志记录器

此路由将导致每 10 个消息记录一次，即使没有消息流量，也会显示初始 60s 延迟和统计数据。

```
from("activemq:orders").
    to("log:com.mycompany.order?
level=DEBUG&groupInterval=10000&groupDelay=60000&groupActiveOnly=false").to("bean:pr
ocessOrder");
```

以下内容会被记录：

```
"Received: 1000 new messages, with total 2000 so far. Last group took: 10000 millis which is: 100
messages per second. average: 100"
```

37.9. 屏蔽敏感信息，如密码

您可以通过将 logMask 标志设置为 true 来启用日志记录的安全掩码。请注意，这个选项也会影响日志 EIP。

在 CamelContext 级别的 Java DSL 中启用掩码：

```
camelContext.setLogMask(true);
```

在 XML 中：

```
<camelContext logMask="true">
```

您还可以在端点级别打开|关闭。要在端点级别的 Java DSL 中启用掩码，请在日志端点的 URI 中添加

logMask=true 选项：

```
from("direct:start").to("log:foo?logMask=true");
```

在 XML 中：

```
<route>
  <from uri="direct:foo"/>
  <to uri="log:foo?logMask=true"/>
</route>
```

org.apache.camel.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，掩码格式必须实施 org.apache.camel.spi.MaskingFormatter。

37.10. 日志输出的完整自定义

通过这个部分中所述的选项，您可以控制日志记录器的许多输出。但是，日志行始终遵循以下结构：

```
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可理解，这可能是因为您需要...

- *过滤打印的标头和属性，使其在 Insights 和详细程度之间平衡。*
- *将日志消息调整为您认为最易读的任何内容。*
- *通过日志减系统（如 Splunk）定制用于摘要的日志消息。*
- *以不同的方式打印特定的正文类型。*

每当需要绝对自定义时，您可以创建一个实施接口的类。在访问完整交换 的格式(Exchange) 方法中，

以便您可以选择并提取所需的准确信息，以自定义方式对其进行格式化并返回它。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您的自定义 ExchangeFormatter ：

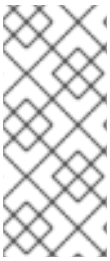
在 Registry 中显式实例化 LogComponent：

```
<bean name="log" class="org.apache.camel.component.log.LogComponent">
  <property name="exchangeFormatter" ref="myCustomFormatter" />
</bean>
```

37.10.1. 与配置相关的惯例

只需将 bean 替换为名称 logFormatter; 日志组件就足够智能，以便自动获取它。

```
<bean name="logFormatter" class="com.xyz.MyCustomExchangeFormatter" />
```



注意

ExchangeFormatter 应用到该 Camel 上下文中的所有日志端点。如果您需要不同的端点使用不同的 ExchangeFormatter，请根据需要实例化 LogComponent，并使用相关的 bean 名称作为端点前缀。

使用自定义日志格式时，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，因此如果您具有不同的参数，则不会共享，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"/>
```

37.11. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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，则默认为 DefaultExchangeFormatter。选项是 org.apache.camel.spi.ExchangeFormatter 类型。		ExchangeFormatter
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

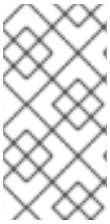
第 38 章 MAIL

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。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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



注意

POP3 或 IMAP
POP3 有一些限制，如果可能，则鼓励使用 IMAP。

注意

使用模拟邮件来测试
，您可以使用模拟框架进行单元测试，这可让您在不需要真实邮件服务器的情况下进行测试。但是，当您进入生产环境或其他需要向真实邮件服务器发送邮件的环境时，您应该不要包含模拟电子邮件。只是 classpath 上的 模拟-javamail.jar 存在，这意味着它将启动并避免发送邮件。

38.1. URI 格式

邮件端点可以具有以下 URI 格式之一（分别用于协议、SMTP、POP3 或 IMAP）：

```
smtp://[username@]host[:port][?options]
pop3://[username@]host[:port][?options]
imap://[username@]host[:port][?options]
```

邮件组件还支持这些协议的安全变体（通过 SSL 进行分层）。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]
```

38.2. 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

38.2.1. 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。

38.2.2. 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。

38.3. 组件选项

邮件组件支持 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)	消费者是否在轮询后关闭文件夹。将这个选项设置为 false，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)	处理邮件后，它可以复制到具有指定名称的邮件文件夹中。您可以使用键 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。这与设置 JVM 系统属性 mail.mime.encodefilename 类似。	false	布尔值
delete (consumer)	在消息被处理后删除消息。这可以通过在邮件中设置 DELETED 标志来完成。如果为 false，则设置 SEEN 标志。从 Camel 2.10 开始，您可以通过使用键 delete 设置标头来覆盖此配置选项，以确定是否应该删除邮件。	false	布尔值
disconnect (consumer)	消费者在轮询后是否应断开。如果启用此选项，它会强制 Camel 在每个轮询连接。	false	布尔值
handleFailedMessage (consumer)	如果邮件使用者无法检索给定的邮件消息，则此选项允许处理消费者的错误处理程序导致的异常。通过在消费者上启用网桥错误处理程序，然后 Camel 路由错误处理程序可以处理异常。默认行为是消费者抛出异常，并且批处理中的邮件无法由 Camel 路由。	false	布尔值
mimeDecodeHeaders (consumer)	这个选项启用对邮件标头的透明 MIME 解码和取消处理。	false	布尔值

Name	描述	默认值	类型
moveTo (consumer)	处理邮件后，它可以移到具有指定名称的邮件文件夹中。您可以使用键 moveTo 的标头来覆盖此配置值，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，会将 javax.mail.Message 标记为 peeked。这只适用于 IMAPMessage 消息类型。通过使用将邮件标记为 SEEN 的邮件，这允许我们在 Camel 中存在错误处理，请回滚邮件。	true	布尔值
skipFailedMessage (consumer)	如果邮件使用者无法检索给定的邮件邮件，此选项允许跳过邮件并继续检索下一个邮件。默认行为是消费者抛出异常，并且批处理中的邮件无法由 Camel 路由。	false	布尔值
unseen (consumer)	是否只通过不可预测的邮件限制。	true	布尔值
fetchSize (consumer (advanced))	设置轮询期间要消耗的最大消息数。如果载体文件夹包含很多消息，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，所有信息都会被消耗。将值设为 0 是一个特殊的地方，Camel 根本不消耗任何消息。	-1	int
folderName (consumer (advanced))	要轮询的文件夹。	INBOX	字符串
mapMailMessage (consumer (advanced))	指定 Camel 是否应该将收到的邮件映射到 Camel body/headers/attachments。如果设置为 true，邮件邮件的正文将映射到 Camel IN 消息的正文，邮件标头将映射到 IN 标头，并附加至 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 receivers （响应邮件的接收器）。使用逗号分隔多个电子邮件地址。		字符串
subject (producer)	正在发送的消息的主题。注： 在标头中设置主题优先于这个选项。		字符串
to (producer)	设置 To email address。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)	设置额外的 java 邮件属性，这将附加/覆盖基于所有其他选项设置的任何默认属性。如果您需要添加一些特殊选项，但希望保留其他特殊选项，这非常有用。		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，如果您以文本/html 格式发送电子邮件，并希望为非 HTML 电子邮件客户端提供替代邮件正文，请将此密钥的替代邮件正文设置为标头。	CamelMailAlternativeBody	字符串
attachmentsContentTransferEncodingResolver (advanced)	要使用自定义将ContentTransferEncodingResolver 解决用于附件的 content-type-encoding。		AttachmentsContentTransferEncodingResolver
authenticator (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
配置 （高级）	设置邮件配置。		MailConfiguration

Name	描述	默认值	类型
connectionTimeout (advanced)	连接超时（毫秒）。	30000	int
contentType (advanced)	邮件消息内容类型。将 text/html 用于 HTML 邮件。	text/plain	字符串
contentTypeResolver (advanced)	用于确定文件附加的 Content-Type 的解析器。		ContentTypeResolver
debugMode (advanced)	在底层邮件框架中启用调试模式。SUN 邮件框架默认将调试信息记录到 System.out。	false	布尔值
ignoreUnsupportedCharset (advanced)	选项使 Camel 在发送邮件时忽略本地 JVM 中不支持的 charset。如果 charset 不支持，则 charset=XXX（其中 XXX 代表不受支持的 charset）已从 content-type 中删除，它依赖于平台默认。	false	布尔值
ignoreUriScheme (advanced)	选项使 Camel 在发送邮件时忽略本地 JVM 中不支持的 charset。如果 charset 不支持，则 charset=XXX（其中 XXX 代表不受支持的 charset）已从 content-type 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
Session (advanced)	指定 camel 应用于所有邮件交互的邮件会话。在由某些其他资源（如 IaaS 容器）创建和管理邮件会话的情况下很有用。使用自定义邮件会话时，将使用邮件会话中的主机名和端口（如果在会话中配置）。		会话
useInlineAttachments (advanced)	是否使用内联或附加。	false	布尔值
HeaderFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 将标头过滤到或从 Camel 消息过滤。		HeaderFilterStrategy
密码 （安全）	登录的密码。另请参阅 setAuthenticator (MailAuthenticator)。		字符串
sslContextParameters (security)	使用 SSLContext 参数配置安全性：		SSLContextParameters
useGlobalSslContextParameters (security)	启用使用全局 SSL 上下文参数。	false	布尔值

Name	描述	默认值	类型
用户名 （安全）	用于登录的用户名。另请参阅 setAuthenticator (MailAuthenticator)。		字符串

38.4. 端点选项

邮件端点使用 URI 语法进行配置：

```
imap:host:port
```

使用以下路径和查询参数：

38.4.1. 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

38.4.2. 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)	消费者是否在轮询后关闭文件夹。将这个选项设置为 false，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)	处理邮件后，它可以复制到具有指定名称的邮件文件夹中。您可以使用键 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

Name	描述	默认值	类型
decodeFilename (consumer)	如果设置为 true，则使用 MimeUtility.decodeText 方法来解码文件名。这与设置 JVM 系统属性 mail.mime.encodefilename 类似。	false	布尔值
delete (consumer)	在消息被处理后删除消息。这可以通过在邮件中设置 DELETED 标志来完成。如果为 false，则设置 SEEN 标志。从 Camel 2.10 开始，您可以通过使用键 delete 设置标头来覆盖此配置选项，以确定是否应该删除邮件。	false	布尔值
disconnect (consumer)	消费者在轮询后是否应断开。如果启用此选项，它会强制 Camel 在每个轮询连接。	false	布尔值
handleFailedMessage (consumer)	如果邮件使用者无法检索给定的邮件消息，则此选项允许处理消费者的错误处理程序导致的异常。通过在消费者上启用网桥错误处理程序，然后 Camel 路由错误处理程序可以处理异常。默认行为是消费者抛出异常，并且批处理中的邮件无法由 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），以避免在服务器启动时下载数千个文件。将值设为 0 或负数以禁用这个选项。		int
mimeDecodeHeaders (consumer)	这个选项启用对邮件标头的透明 MIME 解码和取消处理。	false	布尔值
moveTo (consumer)	处理邮件后，它可以移到具有指定名称的邮件文件夹中。您可以使用键 moveTo 的标头来覆盖此配置值，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，会将 javax.mail.Message 标记为 peeked。这只适用于 IMAPMessage 消息类型。通过使用将邮件标记为 SEEN 的邮件，这允许我们在 Camel 中存在错误处理，请回滚邮件。	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，此选项允许跳过邮件并继续检索下一个邮件。默认行为是消费者抛出异常，并且批处理中的邮件无法由 Camel 路由。	false	布尔值
unseen (consumer)	是否只通过不可预测的邮件限制。	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
fetchSize (consumer (advanced))	设置轮询期间要消耗的最大消息数。如果载体文件夹包含很多消息，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，所有信息都会被消耗。将值设为 0 是一个特殊的地方，Camel 根本不消耗任何消息。	-1	int
folderName (consumer (advanced))	要轮询的文件夹。	INBOX	字符串
mailUidGenerator (consumer (advanced))	可插拔邮件生成器，允许使用自定义逻辑生成邮件邮件的 UUID。		MailUidGenerator
mapMailMessage (consumer (advanced))	指定 Camel 是否应该将收到的邮件映射到 Camel body/headers/attachments。如果设置为 true，邮件邮件的正文将映射到 Camel IN 消息的正文，邮件标头将映射到 IN 标头，并附加至 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))	指的是 mailBoxPostProcessAction，用于在正常处理结束时对交易执行后处理任务。		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

Name	描述	默认值	类型
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
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 receivers （响应邮件的接收器）。使用逗号分隔多个电子邮件地址。		字符串
subject (producer)	正在发送的消息的主题。注： 在标头中设置主题优先于这个选项。		字符串
to (producer)	设置 To email address。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)	设置额外的 java 邮件属性，这将附加/覆盖基于所有其他选项设置的任何默认属性。如果您需要添加一些特殊选项，但希望保留其他特殊选项，这非常有用。		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，如果您以文本/html 格式发送电子邮件，并希望为非 HTML 电子邮件客户端提供替代邮件正文，请将此密钥的替代邮件正文设置为标头。	CamelMailAlternativeBody	字符串
attachmentsContentTransferEncodingResolver (advanced)	要使用自定义将ContentTransferEncodingResolver 解决用于附件的 content-type-encoding。		AttachmentsContentTransferEncodingResolver
authenticator (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 （高级）	设置用于从 Camel 消息转换到邮件邮件的绑定。		MailBinding

Name	描述	默认值	类型
connectionTimeout (advanced)	连接超时（毫秒）。	30000	int
contentType (advanced)	邮件消息内容类型。将 text/html 用于 HTML 邮件。	text/plain	字符串
contentTypeResolver (advanced)	用于确定文件附加的 Content-Type 的解析器。		ContentTypeResolver
debugMode (advanced)	在底层邮件框架中启用调试模式。SUN 邮件框架默认将调试信息记录到 System.out。	false	布尔值
HeaderFilterStrategy (advanced)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 来过滤标头。		HeaderFilterStrategy
ignoreUnsupportedCharset (advanced)	选项使 Camel 在发送邮件时忽略本地 JVM 中不支持的 charset。如果 charset 不支持，则 charset=XXX（其中 XXX 代表不受支持的 charset）已从 content-type 中删除，它依赖于平台默认。	false	布尔值
ignoreUriScheme (advanced)	选项使 Camel 在发送邮件时忽略本地 JVM 中不支持的 charset。如果 charset 不支持，则 charset=XXX（其中 XXX 代表不受支持的 charset）已从 content-type 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
Session (advanced)	指定 camel 应用于所有邮件交互的邮件会话。在由某些其他资源（如 IaaS 容器）创建和管理邮件会话的情况下很有用。使用自定义邮件会话时，将使用邮件会话中的主机名和端口（如果在会话中配置）。		会话
useInlineAttachments (advanced)	是否使用内联或附加。	false	布尔值
idempotentRepository (filter)	可插拔存储库 org.apache.camel.spi.IdempotentRepository，它允许集群从同一站使用，并使存储库协调邮件是否对消费者处理有效。默认情况下不使用存储库。		IdempotentRepository
idempotentRepositoryRemoveOnCommit (filter)	使用幂等存储库时，当成功处理邮件并提交后，消息 ID 会从幂等存储库（默认）或保留在存储库中。默认情况下，它假定消息 id 是唯一的，且没有保留在存储库中的值，因为邮件消息将标记为 see/moved 或 removed，以防止再次消耗它。因此，将消息 id 存储在幂等存储库中具有较少的值。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)	引用 javax.mail.search.SearchTerm，它允许根据搜索标准（如主题、正文、来自等）过滤邮件。		SearchTerm
backoffErrorThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前发生的后续错误轮询（因为某些错误）的数量。		int
backoffIdleThreshold (scheduler)	在 backoffMultipler 应该 kick-in 之前应该发生的后续空闲轮询数量。		int
backoffMultiplier (scheduler)	如果一行中有很多后续空闲/errors，则让调度的轮询消费者避退。然后，倍数是在下一次实际尝试再次发生前跳过的轮询数量。当使用这个选项时，还必须配置 backoffIdleThreshold 和/或 backoffErrorThreshold。		int
delay (scheduler)	下一次轮询前的时间（毫秒）。	60000	long
greedy (scheduler)	如果启用了 greedy，如果上一个运行轮询 1 或更多消息，则 ScheduledPollConsumer 将立即运行。	false	布尔值
initialDelay (scheduler)	第一次轮询开始前的毫秒。	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
密码 （安全）	登录的密码。另请参阅 setAuthenticator (MailAuthenticator)。		字符串
sslContextParameters (security)	使用 SSLContext 参数配置安全性：		SSLContextParameters
用户名 （安全）	用于登录的用户名。另请参阅 setAuthenticator (MailAuthenticator)。		字符串
sortTerm (sort)	排序消息的顺序。只支持 IMAP。在使用 POP3 或 IMAP 服务器没有 SORT 功能时，模拟到某种程度。		SortTerm[]

38.4.3. 端点示例

通常，您可以使用登录凭证指定 URI，如下所示（示例为 SMTP）：

```
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
```

38.4.4. 组件别名名称

- IMAP
- IMAPs
- POP3s
- SMTP
- SMTPs

38.4.5. 默认端口

支持默认端口号。如果省略端口号，Camel 会根据协议决定要使用的端口号。

协议	默认端口号
SMTP	25
SMTPS	465
POP3	110
POP3S	995

协议	默认端口号
IMAP	143
IMAPS	993

38.5. 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，或者通过组件或端点配置提供配置的 SSLContextParameters。

38.5.1. 使用 JSSE 配置工具

邮件组件通过 Camel JSSE 配置实用程序支持 SSL/TLS 配置。这个工具可显著减少您需要编写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 的端点配置

```
...
<camel:sslContextParameters id="sslContextParameters">
  <camel:trustManagers>
    <camel:keyStore resource="/users/home/server/truststore.jks" password="keystorePassword"/>
```

```
  </camel:trustManagers>
</camel:sslContextParameters>...
...
<to uri="smtps://smtp.google.com?
username=user@gmail.com&password=password&sslContextParameters=#sslContextParameters"/
>...
```

38.5.2. 直接配置 JavaMail

Camel 使用 Jakarta JavaMail，它只信任由已知的证书颁发机构（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 信任存储文件中，请覆盖默认的 JVM 信任存储文件（请参阅 JavaMail 中的 SSLNOTES.txt ）。

38.6. 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 的主题可以使用 IN 消息的 header 属性进行配置。以下代码演示了这一点：

同样适用于其他 MimeMessage 标头，如接收者，因此您可以使用与 To 相同的标头属性：

使用 mailProducer 时，应可以从 Camel 消息标头中使用键 CamelMailMessageId 获取 MimeMessage 的消息 ID。

38.7. 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总是优先于端点 URI 中预先配置的接收者。其重点在于，如果您在消息标头中提供任何接收者，这是您获取的。端点 URI 中预先配置的接收者被视为回退。

在以下示例中，电子邮件消息发送到 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 设置都会被忽略，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配置设置。

```
Map<String, Object> headers = new HashMap<String, Object>();
headers.put("to", "davsclaus@apache.org");

template.sendBodyAndHeaders("smtp://admin@localhost?to=info@mycompany.com", "Hello
World", headers);
```

38.8. 用于更轻松地配置的多个接收者

可以使用逗号分隔的或以分号分开的列表来设置多个接收者。这同时适用于标头设置，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。

38.9. 设置发送者名称和电子邮件

您可以使用格式（名为 &lt ;email& gt; ）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");
```

38.10. JAVAMAIL API (EX SUN JAVAMAIL)

javamail API 在 hood 下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。特别是 POP3 具有比 IMAP 更有限的功能。

- *javamail POP3 API*

- *javamail IMAP API*
- *一般有关 MAIL 标记*

38.11. SAMPLES

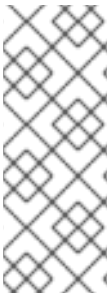
我们从一个简单路由开始，该路由将从 JMS 队列接收的消息作为电子邮件发送。电子邮件帐户是 mymailserver.com 上的 admin 帐户。

```
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。

```
from("imap://admin@mymailserver.com?password=secret&unseen=true&delay=60000")
    .to("seda://mails");
```

38.12. 使用附加示例发送邮件



注意

所有 Camel 组件不再支持 附件，它基于 Java 激活框架，通常仅由邮件 API 使用。由于许多其他 Camel 组件不支持附件，因此当附件通过路由传播时可能会丢失。因此，thumb 的规则是在向邮件端点发送消息前添加附件。

邮件组件支持附加。在以下示例中，我们发送一条邮件，其中包含带有徽标文件附件的纯文本消息。

38.13. SSL 示例

在本例中，我们希望轮询 Google 邮件中的邮件。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 Google 邮件帐户并更改您的设置以允许 IMAP 访问来实现。Google 对如何执行此操作有广泛的文档。

```
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 Google 邮件中的 Google 邮件，并将收到的消息记录到 新的mail 日志记录器类别。
运行启用 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=[...
```

38.14. 使用附加示例消耗邮件

在这个示例中，我们轮询一个载体，并将邮件中的所有附件存储为文件。首先，我们定义了一个路由来轮询语音。由于本示例基于 google 邮件，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());
```

我们不使用一个处理器来记录邮件，以便我们可以处理来自 java 代码的邮件：

```
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();
        }
    }
}
```

您可以看到处理附件的 API 是一个位冲突，但在有些情况下，您可以获取 javax.activation.DataHandler，以便您可以使用标准 API 处理附件。

38.15. 如何使用附件分割邮件

在这个示例中，我们消耗可能有多个附件的邮件。我们希望在每个单独附件使用 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，每个消息都只有一个附件。要做到这一点，我们需要向 Splitter 提供自定义表达式，其中我们提供了一个 List<Message>，其中包含带有单个附件的五个消息。

该代码开箱即用了 camel-mail 组件中的 Camel 2.10 中的框。代码位于类：org.apache.camel.component.mail.SplitAttachmentsExpression，您可以在此处 的源代码中找到。

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 中声明一个方法调用表达式，如下所示

```
<split>
  <method beanType="org.apache.camel.component.mail.SplitAttachmentsExpression"/>
  <to uri="mock:split"/>
</split>
```

您还可以将附件分割为 byte[]，以作为消息正文存储。这可以通过使用布尔值 true 创建表达式来实现

```
SplitAttachmentsExpression split = SplitAttachmentsExpression(true);
```

然后，将表达式与分割 EIP 搭配使用。

38.16. 使用自定义搜索TERM

您可以在 mail Endpoint 上配置 searchTerm，允许您过滤掉不需要的邮件。

例如，要将邮件过滤为在 Subject 或文本中包含 Camel，您可以执行以下操作：

```
<route>
  <from uri="imaps://mymailseerver?
```

```
username=foo&password=secret&searchTerm.subjectOrBody=Camel"/>
  <to uri="bean:myBean"/>
</route>
```

请注意，我们使用 "searchTerm.subjectOrBody" 作为参数键来指示我们要搜索邮件主题或正文，使其包含"Camel"。
类 org.apache.camel.component.mail.SimpleSearchTerm 有配置多个选项：

或者，为了让新的不可预测的电子邮件在 24 小时内返回。请注意"now-24h"语法。详情请查看下表。

```
<route>
  <from uri="imaps://mymailseerver?
username=foo&password=secret&searchTerm.fromSentDate=now-24h"/>
  <to uri="bean:myBean"/>
</route>
```

您可以在端点 uri 配置中有多个 searchTerm。然后，它们将使用 AND 运算符组合在一起，因此这两个条件都必须匹配。例如，要获得最后的十倍电子邮件返回 24 小时，该电子邮件在邮件主题中有 Camel：

```
<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 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，您可以使用 Camel 路由中的 192.168.1.0/24beanId 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，有一个构建程序类，可使用 org.apache.camel.component.mail.SearchTermBuilder 类来构建复合搜索 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();
```

38.17. 轮询优化

参数 maxMessagePerPoll 和 fetchSize 允许您限制每个轮询应处理的数字消息。在处理包含大量消息的文件夹时，这些参数有助于防止性能不佳。在以前的版本中，这些参数已被评估得太晚，因此大型竞争可能仍然会导致性能问题。使用 Camel 3.1 这些参数会在轮询早期评估，以避免这些问题。

38.18. 使用带有额外 JAVA 邮件发送器属性的标头

发送邮件时，您可以使用以 java .smtp 开头的密钥为来自 Exchange 的 JavaMailSender 提供动态 java 邮件属性。

您可以设置任何可在 Java 邮件文档中找到的 java.smtp 属性。

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 不使用 自定义 JavaMailSender 时被支持。

38.19. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 imap 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>
```

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	设置额外的 java 邮件属性，这将附加/覆盖基于所有其他选项设置的任何默认属性。如果您需要添加一些特殊选项，但希望保留其他特殊选项，这非常有用。选项是 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，如果您以文本/html 格式发送电子邮件，并希望为非 HTML 电子邮件客户端提供替代邮件正文，请将此密钥的替代邮件正文设置为标头。	CamelMailAlternativeBody	字符串
camel.component.mail.attachments-content-transfer-encoding-resolver	要使用自定义将ContentTransferEncodingResolver 解决用于附件的 content-type-encoding。选项是一个 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

Name	描述	默认值	类型
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
camel.component.mail.close-folder	消费者是否在轮询后关闭文件夹。将这个选项设置为 false，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（毫秒）。	30000	整数
camel.component.mail.content-type	邮件消息内容类型。将 text/html 用于 HTML 邮件。	text/plain	字符串
camel.component.mail.content-type-resolver	用于确定文件附加的 Content-Type 的解析器。选项是 org.apache.camel.component.mail.ContentTypeResolver 类型。		ContentTypeResolver
camel.component.mail.copy-to	处理邮件后，它可以复制到具有指定名称的邮件文件夹中。您可以使用键 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	在底层邮件框架中启用调试模式。SUN 邮件框架默认将调试信息记录到 System.out。	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。这与设置 JVM 系统属性 mail.mime.encodefilename 类似。	false	布尔值
camel.component.mail.delete	在消息被处理后删除消息。这可以通过在邮件中设置 DELETED 标志来完成。如果为 false，则设置 SEEN 标志。从 Camel 2.10 开始，您可以通过使用键 delete 设置标头来覆盖此配置选项，以确定是否应该删除邮件。	false	布尔值
camel.component.mail.disconnect	消费者在轮询后是否应断开。如果启用此选项，它会强制 Camel 在每个轮询连接。	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	设置轮询期间要消耗的最大消息数。如果载体文件夹包含很多消息，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，所有信息都会被消耗。将值设为 0 是一个特殊的地方，Camel 根本不消耗任何消息。	-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	如果邮件使用者无法检索给定的邮件消息，则此选项允许处理消费者的错误处理程序导致的异常。通过在消费者上启用网桥错误处理程序，然后 Camel 路由错误处理程序可以处理异常。默认行为是消费者抛出异常，并且批处理中的邮件无法由 Camel 路由。	false	布尔值
camel.component.mail.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 将标头过滤到或从 Camel 消息过滤。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		HeaderFilterStrategy
camel.component.mail.ignore-unsupported-charset	选项使 Camel 在发送邮件时忽略本地 JVM 中不支持的 charset。如果 charset 不支持，则 charset=XXX（其中 XXX 代表不受支持的 charset）已从 content-type 中删除，它依赖于平台默认。	false	布尔值
camel.component.mail.ignore-uri-scheme	选项使 Camel 在发送邮件时忽略本地 JVM 中不支持的 charset。如果 charset 不支持，则 charset=XXX（其中 XXX 代表不受支持的 charset）已从 content-type 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 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 是否应该将收到的邮件映射到 Camel body/headers/attachments。如果设置为 true，邮件邮件的正文将映射到 Camel IN 消息的正文，邮件标头将映射到 IN 标头，并附加至 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	这个选项启用对邮件标头的透明 MIME 解码和取消处理。	false	布尔值
camel.component.mail.move-to	处理邮件后，它可以移到具有指定名称的邮件文件夹中。您可以使用键 moveTo 的标头来覆盖此配置值，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	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，会将 javax.mail.Message 标记为 peeked。这只适用于 IMAPMessage 消息类型。通过使用将邮件标记为 SEEN 的邮件，这允许我们在 Camel 中存在错误处理，请回滚邮件。	true	布尔值
camel.component.mail.reply-to	Reply-To receivers （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 应用于所有邮件交互的邮件会话。在由某些其他资源（如 IaaS 容器）创建和管理邮件会话的情况下很有用。使用自定义邮件会话时，将使用邮件会话中的主机名和端口（如果在会话中配置）。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，此选项允许跳过邮件并继续检索下一个邮件。默认行为是消费者抛出异常，并且批处理中的邮件无法由 Camel 路由。	false	布尔值

Name	描述	默认值	类型
camel.component.mail.ssl-context-parameters	使用 SSLContext 参数配置安全性：选项是 org.apache.camel.support.jsse.SSLContextParameters 类型。		SSLContextParameters
camel.component.mail.subject	正在发送的消息的主题。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 email address。使用逗号分隔多个电子邮件地址。		字符串
camel.component.mail.unseen	是否只通过不可预测的邮件限制。	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)或 Base-64 编码(false) Default 为 false。	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 标头是否为消息正文(true)的一部分，还是设置为 Camel 标头(false)。默认值为 false。	false	布尔值
camel.dataformat.mime-multipart.include-headers	定义将哪些 Camel 标头作为 MIME 标头包含在 MIME 多部分的 regex。这只有在 headersInline 设为 true 时才起作用。默认为不包含标头。		字符串

Name	描述	默认值	类型
camel.dataformat.mime-multipart.multipart-sub-type	指定 MIME 多部分的子类型。默认为 hybrid。	mixed	字符串
camel.dataformat.mime-multipart.multipart-without-attachment	定义没有附件的消息是否也被放入 MIME 多部分（只有一个正文部分）。默认值为 false。	false	布尔值

第 39 章 MICROSOFT OAUTH 的邮件

从 Camel 3.18.4 开始。

邮件 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。

将以下依赖项添加到此组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-mail-microsoft-oauth</artifactId>
  <version>3.20.1.redhat-00050</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

导入 camel-mail-microsoft-oauth 将自动导入 camel-mail 组件。

39.1. MICROSOFT EXCHANGE ONLINE OAUTH2 邮件验证器 IMAP 示例

要使用 OAuth，必须在 Azure Active Directory 中注册一个应用程序。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据流访问交换通信。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 中声明并注册，一个 org.apache.camel.component.mail.MicrosoftExchangeOnlineOAuth2MailAuthenticator 的实例。

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 40 章 MAPSTRUCT

从 Camel 3.19 开始

仅支持生成者。

camel-mapstruct 组件用于使用 转换 POJO。

40.1. URI 格式

```
mapstruct:className[?options]
```

其中 className 是要转换为的 POJO 的完全限定类名称。

40.2. 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

40.2.1. 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。

40.2.2. 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。

40.3. 组件选项

MapStruct 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
mapperPackageName (producer)	**需要** 的软件包名称，Camel 应该发现 Mapstruct 映射类。可以使用逗号分隔多个软件包名称。		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
mapStructConverter (advanced)	**Autowired** 使用自定义 MapStructConverter，如适应特殊运行时。		MapStructMapperFinder

40.4. 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：

40.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
classname (producer)	**必需** 映射结构应转换为(target)的 POJO 的完全限定类名称。		字符串

40.4.2. 查询参数(2 参数)

Name	描述	默认值	类型
mandatory (producer)	是否存在映射结构转换器才能转换为 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

40.5. 设置 MAPSTRUCT

camel-mapstruct 组件必须使用一个或多个软件包名称进行配置，用于类路径扫描 MapStruct Mapper 类。这是必要的，因为 映射程序 类用于使用 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，以获取名称以 Mapper 结尾的类。然后会内省这些类来发现映射方法。这些映射方法随后会注册到 Camel registry 中。这意味着，您还可以使用类型转换器将 POJO 转换为 MapStruct，例如：

```
from("direct:foo")
  .convertBodyTo(MyFooDto.class);
```

其中 MyFooDto 是一个 POJO，其中 MapStruct 能够转换为/从中转换。

40.6. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用映射结构时，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>
```

组件支持 5 个选项，如下所列。

Name	描述	默认值	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
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

Name	描述	默认值	类型
camel.component.mapstruct.map-struct-converter	使用自定义 MapStructConverter，如适应特殊运行时。选项是一个 org.apache.camel.component.mapstruct.MapStructMapperFinder 类型。		MapStructMapperFinder
camel.component.mapstruct.mapper-package-name	Camel 应该发现映射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 41 章 MASTER

仅支持消费者

Camel-Master 端点提供了一种方式来确保集群中只有一个消费者从给定端点消耗；如果 JVM 结束，则会自动故障转移。

如果您需要从一些传统后端使用，这些后端不支持并发消耗，或者由于商业或稳定性的原因，您可以随时有一个连接，则这非常有用。

41.1. 使用 MASTER 端点

只需使用 master:someName: 前缀的任何 camel 端点，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，有 master 组件确保集群中任何给定时间只在一个节点中激活该路由。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，且只有此路由处于活动状态，因此只有此路由将使用来自 jms:foo 的消息。如果此路由停止或意外终止，则 master 组件将检测到这一点，并重新激活另一个节点，然后变为 active，然后变为 active，并启动来自 jms:foo 的消息。

注意

Apache ActiveMQ 5.x 的功能开箱即用，称为 Exclusive Consumers。

41.2. URI 格式

```
master:namespace:endpoint[?options]
```

其中 endpoint 是您希望在主/从模式下运行的任何 Camel 端点。

41.3. 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

41.3.1. 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。

41.3.2. 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。

41.4. 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
服务 (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

41.5. 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：

41.5.1. 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
delegateUri (consumer)	**必需** 在 master/slave 模式中使用的端点 uri。		字符串

41.5.2. 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

41.6. 示例

您可以保护集群的 Camel 应用程序，使其仅消耗来自一个活跃节点的文件。

```
// the file endpoint we want to consume from
String url = "file:target/inbox?delete=true";

// use the camel master component in the clustered group named myGroup
// to run a master/slave mode in the following Camel url
from("master:myGroup:" + url)
    .log(name + " - Received file: ${file:name}")
    .delay(delay)
    .log(name + " - Done file:     ${file:name}")
    .to("file:target/outbox");
```

主(master)组件利用 CamelClusterService，您可以使用

- ***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
```

41.7. 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*

41.8. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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
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

Name	描述	默认值	类型
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 org.apache.camel.cluster.CamelClusterService.Selector 类型。		CamelClusterService$Selector

第 42 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。

42.1. 先决条件

您必须具有有效的凭证才能授权访问存储桶/文件夹。如需更多信息，请参阅 Minio。

42.2. 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，请使用以下代码片段：

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

42.3. 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

42.3.1. 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。

42.3.2. 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。

42.4. 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

Name	描述	默认值	类型
customHttpClient (common)	设置自定义 HTTP 客户端以进行身份验证的访问。		OkHttpClient
endpoint (common)	端点可以是 URL、域名、IPv4 地址或 IPv6 地址。		字符串
minioClient (common)	对 registry 中的 Minio Client 对象的 **Autowired** 引用。		MinioClient
objectLock (common)	在新存储桶时设置。	false	布尔值
policy (common)	在方法中设置的此队列的策略。		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)	用于指示使用安全连接的标记来 minio 服务。	false	布尔值
serverSideEncryption (common)	服务器端加密。		ServerSideEncryption
serverSideEncryptionCustomerKey (common)	在复制/附加对象时源对象的服务器端加密。		ServerSideEncryptionCustomerKey
autoCloseBody (consumer)	如果此选项为 true 且 includeBody 为 true，则在交换完成时将调用 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
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
bypassGovernanceMode (consumer)	如果您要在删除特定对象时绕过GovernanceMode，则设置此标志。	false	布尔值

Name	描述	默认值	类型
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 MinioConstants114BUCKET_NAME 和 MinioConstantsöOBJECT_NAME 标头进行过滤，或者只有 MinioConstants114OBJECT_NAME 标头。	true	布尔值
delimiter (consumer)	ListObjectsRequest 中使用的分隔符仅用于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的内容。如果为 false，则标头将使用 Minio 对象元数据设置，但正文将是 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 用来设置的标记包括文件夹。	false	布尔值
includeUserMetadata (consumer)	ListObjectsRequest 中使用的标志，用于获取具有用户元数据的对象。	false	布尔值
includeVersions (consumer)	ListObjectsRequest 中使用的标志，用于获取带有版本控制的对象。	false	布尔值
length (consumer)	来自偏移的对象数据的字节数。		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

Name	描述	默认值	类型
modifiedSince (consumer)	为 get 对象的参数设置修改，因为参数为 get 对象设置。		ZonedDateTime
moveAfterRead (consumer)	在检索后，将对象从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
notMatchETag (consumer)	为 get 对象设置不匹配 ETag 参数。		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)	对象数据的开始字节位置。		long
prefix (consumer)	对象名称以前缀开头。		字符串
recursion (consumer)	递归列出，而不是目录结构模拟。	false	布尔值
startAfter (consumer)	在此对象名称后列出存储桶中的对象。		字符串
unModifiedSince (consumer)	为 get 对象的参数设置未修改。		ZonedDateTime
useVersion1 (consumer)	为 true 时，则使用 REST API 的版本 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
deleteAfterWrite (producer)	上传 Minio 文件后删除文件对象。	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

Name	描述	默认值	类型
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 访问密钥或 Minio 访问密钥。如果没有设置 camel，则会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 密钥。如果没有设置 camel，则会连接到服务以进行匿名访问。		字符串

42.5. 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：

42.5.1. 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

42.5.2. 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)	设置自定义 HTTP 客户端以进行身份验证的访问。		OkHttpClient
endpoint (common)	端点可以是 URL、域名、IPv4 地址或 IPv6 地址。		字符串
minioClient (common)	对 registry 中的 Minio Client 对象的 **Autowired** 引用。		MinioClient
objectLock (common)	在新存储桶时设置。	false	布尔值
policy (common)	在方法中设置的此队列的策略。		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)	用于指示使用安全连接的标记来 minio 服务。	false	布尔值
serverSideEncryption (common)	服务器端加密。		ServerSideEncryption
serverSideEncryptionCustomerKey (common)	在复制/附加对象时源对象的服务器端加密。		ServerSideEncryptionCustomerKey

Name	描述	默认值	类型
autoCloseBody (consumer)	如果此选项为 true 且 includeBody 为 true，则在交换完成时将调用 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)	如果您要在删除特定对象时绕过GovernanceMode，则设置此标志。	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 MinioConstants114BUCKET_NAME 和 MinioConstantsöOBJECT_NAME 标头进行过滤，或者只有 MinioConstants114OBJECT_NAME 标头。	true	布尔值
delimiter (consumer)	ListObjectsRequest 中使用的分隔符仅用于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的内容。如果为 false，则标头将使用 Minio 对象元数据设置，但正文将是 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 用来设置的标记包括文件夹。	false	布尔值
includeUserMetadata (consumer)	ListObjectsRequest 中使用的标志，用于获取具有用户元数据的对象。	false	布尔值
includeVersions (consumer)	ListObjectsRequest 中使用的标志，用于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)	来自偏移的对象数据的字节数。		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 对象的参数设置修改，因为参数为 get 对象设置。		ZonedDateTime
moveAfterRead (consumer)	在检索后，将对象从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
notMatchETag (consumer)	为 get 对象设置不匹配 ETag 参数。		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)	对象数据的开始字节位置。		long
prefix (consumer)	对象名称以前缀开头。		字符串
recursion (consumer)	递归列出，而不是目录结构模拟。	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存储桶中的对象。		字符串
unModifiedSince (consumer)	为 get 对象的参数设置未修改。		ZonedDateTime
useVersion1 (consumer)	为 true 时，则使用 REST API 的版本 1。	false	布尔值

Name	描述	默认值	类型
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)	上传 Minio 文件后删除文件对象。	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
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
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
accessKey (security)	Amazon AWS Secret 访问密钥或 Minio 访问密钥。如果没有设置 camel，则会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 密钥。如果没有设置 camel，则会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。

42.6. 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。

42.7. 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 ：此对象将存储或用于当前操作的存储桶名称。consumer ：包含此对象的存储桶的名称。		字符串
CamelMinioDestinationBucketName (producer) 常数：DESTINATION_BUCKET_NAME	用于当前操作的存储桶目的地名称。		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/恢复链中指定缓存行为。		字符串
CamelMinioContentDisposition (common) 常量：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，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) 常数：CONTENT_LENGTH	producer ：此对象的内容长度。consumer ：Content-Length HTTP 标头表示关联对象的大小（以字节为单位）。		Long

Name	描述	默认值	类型
CamelMinioContentMD5 (common) 常量：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
CamelMinioContentType (common) 常量：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 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 ：此对象将存储或用于当前操作的密钥。consumer ：存储此对象的密钥。		字符串
CamelMinioDestinationObjectName (producer) 常数：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 最后记录对关联对象的修改的日期和时间。		Date
CamelMinioStorageClass (producer) 常数：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）。只有当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

Name	描述	默认值	类型
CamelMinioCannedAcl (producer) 常数：CANNED_ACL	将应用到对象的 canned acl。有关允许的值，请参阅 com.amazonaws.services.s3.model.CannedAccessControlList。		字符串
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) 常量：SERVER_SIDE_ENCRYPTION	producer ：在使用 Minio 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。consumer ：在使用 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) 常数：EXPIRATION_TIME	过期时间。		字符串
CamelMinioReplicationStatus (common) 常数：REPLICATION_STATUS	复制状态。		字符串

Name	描述	默认值	类型
CamelMinioOffset (producer) 常量： OFFSET	偏移。		字符串
CamelMinioLength (producer) 常数： LENGTH	长度。		字符串

42.7.1. 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）*

42.7.2. 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 实例配置拥有更多控制，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");
```

42.7.3. 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。

42.8. 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 对 Minio 中不存在的存储桶的操作，则不会成功，并返回错误。

42.9. 在 REGISTRY 中自动检测 MINIO 客户端

组件能够检测 registry 中存在 Minio bean。如果它是唯一将用作客户端的实例，并且您不必将其定义为 uri 参数，如上例所示。这对端点的智能配置可能非常有用。

42.10. 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在同一原始存储桶中。

42.11. 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("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 ）。

42.12. 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 代理请求的示例，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。

42.13. 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-minio</artifactId>
    <version>${camel-version}</version>
</dependency>
```

其中 $3.18.3 必须替换为 Camel 的实际版本。

42.14. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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 访问密钥或 Minio 访问密钥。如果没有设置 camel，则会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 且 includeBody 为 true，则在交换完成时将调用 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
camel.component.minio.bypass-governance-mode	如果您要在删除特定对象时绕过GovernanceMode，则设置此标志。	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	设置自定义 HTTP 客户端以进行身份验证的访问。选项是一个 okhttp3.OkHttpClient 类型。		OkHttpClient

Name	描述	默认值	类型
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 MinioConstants114BUCKET_NAME 和 MinioConstantsöOBJECT_NAME 标头进行过滤，或者只有 MinioConstants114OBJECT_NAME 标头。	true	布尔值
camel.component.minio.delete-after-write	上传 Minio 文件后删除文件对象。	false	布尔值
camel.component.minio.delimiter	ListObjectsRequest 中使用的分隔符仅用于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
camel.component.minio.include-body	如果为 true，则交换正文将设置为文件的内容。如果为 false，则标头将使用 Minio 对象元数据设置，但正文将是 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 用来设置的标记包括文件夹。	false	布尔值
camel.component.minio.include-user-metadata	ListObjectsRequest 中使用的标志，用于获取具有用户元数据的对象。	false	布尔值
camel.component.minio.include-versions	ListObjectsRequest 中使用的标志，用于获取带有版本控制的对象。	false	布尔值

Name	描述	默认值	类型
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	来自偏移的对象数据的字节数。		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
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 对象的参数设置修改，因为参数为 get 对象设置。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.minio.not-match-e-tag	为 get 对象设置不匹配 ETag 参数。		字符串
camel.component.minio.object-lock	在新存储桶时设置。	false	布尔值

Name	描述	默认值	类型
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	对象数据的开始字节位置。		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	在方法中设置的此队列的策略。		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	递归列出，而不是目录结构模拟。	false	布尔值
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 密钥。如果没有设置 camel，则会连接到服务以进行匿名访问。		字符串
camel.component.minio.secure	用于指示使用安全连接的标记来 minio 服务。	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	在复制/附加对象时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存储桶中的对象。		字符串

Name	描述	默认值	类型
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 对象的参数设置未修改。选项是一个 java.time.ZonedDateTime 类型。		ZonedDateTime
camel.component.minio.use-version1	为 true 时，则使用 REST API 的版本 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 43 章 MLLP

支持生成者和消费者

MLLP 组件专门用于处理 MLLP 协议的负性，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 组件提供一个简单的配置 URI，自动 HL7 确认生成和自动确认干预。

MLLP 协议通常使用大量并发 TCP 连接 - 单个活跃的 TCP 连接是正常情况。因此，MLLP 组件使用基于标准 Java 套接字的简单线程连接模型。这会保持简单实施，并消除对 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 producer

MLLP 组件使用 byte[] payloads，并依赖于 Camel 类型转换将 byte[] 转换为其他类型。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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

43.1. 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

43.1.1. 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。

43.1.2. 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。

43.2. 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)	启用/禁用仅自动生成 MLLP Acknowledgement MLLP Consumers。	true	布尔值
charsetName (common)	设置要使用的默认 charset。		字符串
configuration (common)	设置在创建 MLLP 端点时要使用的默认配置。		MllpConfiguration
hl7Headers (common)	启用/禁用从 HL7 Message MLLP Consumers 中自动生成消息标头。	true	布尔值
requireEndOfData (common)	启用/禁用与 MLLP 标准的严格合规性。MLLP 标准指定 START_OF_BLOCKhl7 有效负载 END_OF_BLOCKEND_OF_DATA，但有些系统不会发送最终的 END_OF_DATA 字节。此设置控制是否需要最终的 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字符串。如果启用，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 MSH-18 的值将用于确定 th 个适当的字符集。如果没有设置 MSH-18，则使用默认的 ISO-8859-1 字符集。	true	布尔值
validatePayload (common)	启用/禁用 HL7 Payloads if enabled，从外部系统接收的 HL7 Payloads 将被验证（请参阅 Hl7Util.generateInvalidPayloadExceptionMessage 以了解验证的详情）。如果检测到无效的有效负载，则会抛出 MllpInvalidMessageException （消费者）或 MllpInvalidAcknowledgementException。	false	布尔值
acceptTimeout (consumer)	仅等待 TCP 连接 TCP 服务器时的超时（以毫秒为单位）。	60000	int
Back log (consumer)	传入连接的最大队列长度（要连接的请求）设置为 backlog 参数。如果当队列满时连接表示到达，则拒绝连接。	5	整数
bindRetryInterval (consumer)	仅 TCP 服务器 - 绑定尝试之间等待的毫秒数。	5000	int
bindTimeout (consumer)	仅 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 路由错误处理程序，这意味着消费者试图接收传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。如果禁用，使用者将使用 org.apache.camel.spi.ExceptionHandler 在 WARN 或 ERROR 级别记录它们并忽略来处理异常。	true	布尔值
lenientBind (consumer)	仅 TCP 服务器 - 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，并且已经建立最大连接，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)	仅建立 TCP 连接 TCP 客户端的超时（以毫秒为单位）。	30000	int
idleTimeoutStrategy (producer)	决定在闲置超时发生时要执行的操作。可能的值有：RESET ：将 SO_LINGER 设置为 0，并重置套接字 CLOSE ：关闭套接字安全默认值为 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)	设置用于字节的默认字符设置为/从字符串转换。	ISO-8859-1	字符串
logPhi (advanced)	是否记录 PHI。	true	布尔值
logPhiMaxBytes (advanced)	设置将在日志条目中记录的最大字节数 PHI。	5120	整数
readTimeout (advanced)	在收到 MLLP 帧的开头后，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)	重置客户端 TCP 连接前允许的大约空闲时间。null 值或值小于或等于零将禁用闲置超时。		整数

43.3. 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：

43.3.1. 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，即任何本地 IP 地址。		字符串
port (common)	TCP 连接所需的端口号。		int

43.3.2. 查询参数(26 参数)

Name	描述	默认值	类型
autoAck (common)	启用/禁用仅自动生成 MLLP Acknowledgement MLLP Consumers。	true	布尔值
charsetName (common)	设置要使用的默认 charset。		字符串
hl7Headers (common)	启用/禁用从 HL7 Message MLLP Consumers 中自动生成消息标头。	true	布尔值
requireEndOfData (common)	启用/禁用与 MLLP 标准的严格合规性。MLLP 标准指定 START_OF_BLOCKhl7 有效负载 END_OF_BLOCKEND_OF_DATA，但有些系统不会发送最终的 END_OF_DATA 字节。此设置控制是否需要最终的 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字符串。如果启用，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 MSH-18 的值将用于确定 th 个适当的字符集。如果没有设置 MSH-18，则使用默认的 ISO-8859-1 字符集。	true	布尔值
validatePayload (common)	启用/禁用 HL7 Payloads if enabled，从外部系统接收的 HL7 Payloads 将被验证（请参阅 Hl7Util.generateInvalidPayloadExceptionMessage 以了解验证的详情）。如果检测到无效的有效负载，则会抛出 MllpInvalidMessageException （消费者）或 MllpInvalidAcknowledgementException。	false	布尔值
acceptTimeout (consumer)	仅等待 TCP 连接 TCP 服务器时的超时（以毫秒为单位）。	60000	int

Name	描述	默认值	类型
Back log (consumer)	传入连接的最大队列长度（要连接的请求）设置为 backlog 参数。如果当队列满时连接表示到达，则拒绝连接。	5	整数
bindRetryInterval (consumer)	仅 TCP 服务器 - 绑定尝试之间等待的毫秒数。	5000	int
bindTimeout (consumer)	仅 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 路由错误处理程序，这意味着消费者试图接收传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。如果禁用，使用者将使用 org.apache.camel.spi.ExceptionHandler 在 WARN 或 ERROR 级别记录它们并忽略来处理异常。	true	布尔值
lenientBind (consumer)	仅 TCP 服务器 - 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，并且已经建立最大连接，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)	仅建立 TCP 连接 TCP 客户端的超时（以毫秒为单位）。	30000	int

Name	描述	默认值	类型
idleTimeoutStrategy (producer)	决定在闲置超时发生时要执行的操作。可能的值有：RESET ：将 SO_LINGER 设置为 0，并重置套接字 CLOSE ：关闭套接字安全默认值为 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)	在收到 MLLP 帧的开头后，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)	重置客户端 TCP 连接前允许的大约空闲时间。null 值或值小于或等于零将禁用闲置超时。		整数

43.4. 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 and AR)，或使用 CamelMllpAcknowledgement Exchange 属性来指定确认。此外，通过设置 CamelMllpAcknowledgementType Exchange 属性来控制将生成的确认类型。如果禁用了自动确认且交换模式是 InOnly，则 MLLP Consumer 可以读取消息，而不发送任何 HL7 Acknowledgement。

43.4.1. 消息标头

MLLP Consumer 在 Camel 消息中添加这些标头：

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。

43.4.2. 交换属性

确认 MLLP Consumer 生成的类型和 TCP 套接字的状态可以由 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	如果没有存在并且 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 CamelMllpAcknowledgement 或 CamelMllpAcknowledgementString 不存在，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 CamelMllpAcknowledgement 或 CamelMllpAcknowledgementString 不存在，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，则在发送数据前将关闭套接字
CamelMllpResetConnectionBeforeSend	布尔值	如果为 true，则在发送数据前重置套接字
CamelMllpCloseConnectionAfterSend	布尔值	如果为 true，则在发送数据后立即关闭套接字
CamelMllpResetConnectionAfterSend	布尔值	如果为 true，则在发送任何数据后立即重置套接字

43.5. MLLP PRODUCER

MLLP Producer 支持发送 MLLP-framed 消息并接收 HL7 Acknowledgements。MLLP Producer 中断 HL7 Acknowgments，并在收到负确认时引发异常。收到的确认是临时的，在进行负确认时会引发异常。使用 InOnly Exchange 模式配置时，MLLP Producer 可以忽略确认。

43.5.1. 消息标头

MLLP Producer 在 Camel 消息中添加这些标头：

键	描述
CamelMllpLocalAddress	套接字的本地 TCP 地址
CamelMllpRemoteAddress	套接字的远程 TCP 地址
CamelMllpAcknowledgement	HL7 Acknowledgment byte[] receive
CamelMllpAcknowledgementString	HL7 Acknowgment 收到，转换为字符串

43.5.2. 交换属性

TCP 套接字的状态可以由 Camel 交换上的这些属性控制：

键	类型	描述
CamelMllpCloseConnectionBeforeSend	布尔值	如果为 true，则在发送数据前将关闭套接字
CamelMllpResetConnectionBeforeSend	布尔值	如果为 true，则在发送数据前重置套接字
CamelMllpCloseConnectionAfterSend	布尔值	如果为 true，则在发送数据后立即关闭套接字
CamelMllpResetConnectionAfterSend	布尔值	如果为 true，则在发送任何数据后立即重置套接字

43.6. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	仅等待 TCP 连接 TCP 服务器时的超时（以毫秒为单位）。	60000	整数
camel.component.mllp.auto-ack	启用/禁用仅自动生成 MLLP Acknowledgement MLLP Consumers。	true	布尔值
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（要连接的请求）设置为 backlog 参数。如果当队列满时连接表示到达，则拒绝连接。	5	整数
camel.component.mllp.bind-retry-interval	仅 TCP 服务器 - 绑定尝试之间等待的毫秒数。	5000	整数
camel.component.mllp.bind-timeout	仅 TCP 服务器 - 重试绑定到服务器端口的毫秒数。	30000	整数
camel.component.mllp.bridge-error-handler	允许将消费者桥接到 Camel 路由错误处理程序，这意味着消费者试图接收传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。如果禁用，使用者将使用 org.apache.camel.spi.ExceptionHandler 在 WARN 或 ERROR 级别记录它们并忽略来处理异常。	true	布尔值
camel.component.mllp.charset-name	设置要使用的默认 charset。		字符串
camel.component.mllp.configuration	设置在创建 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	仅建立 TCP 连接 TCP 客户端的超时（以毫秒为单位）。	30000	整数
camel.component.mllp.default-charset	设置用于字节的默认字符设置为/从字符串转换。	ISO-8859-1	字符串

Name	描述	默认值	类型
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
camel.component.mllp.hl7-headers	启用/禁用从 HL7 Message MLLP Consumers 中自动生成消息标头。	true	布尔值
camel.component.mllp.idle-timeout	重置客户端 TCP 连接前允许的大约空闲时间。null 值或值小于或等于零将禁用闲置超时。		整数
camel.component.mllp.idle-timeout-strategy	决定在闲置超时发生时要执行的操作。可能的值有：RESET ：将 SO_LINGER 设置为 0，并重置套接字 CLOSE ：关闭套接字安全默认值为 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	仅 TCP 服务器 - 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 绑定前启动。	false	布尔值
camel.component.mllp.log-phi	是否记录 PHI。	true	布尔值
camel.component.mllp.log-phi-max-bytes	设置将在日志条目中记录的最大字节数 PHI。	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，并且已经建立最大连接，则新连接将立即重置。	5	整数
camel.component.mllp.read-timeout	在收到 MLLP 帧的开头后，使用的 SO_TIMEOUT 值（以毫秒为单位）。	5000	整数

Name	描述	默认值	类型
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
camel.component.mllp.require-end-of-data	启用/禁用与 MLLP 标准的严格合规性。MLLP 标准指定 START_OF_BLOCKhl7 有效负载 END_OF_BLOCKEND_OF_DATA，但有些系统不会发送最终的 END_OF_DATA 字节。此设置控制是否需要最终的 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字符串。如果启用，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 MSH-18 的值将用于确定 th 个适当的字符集。如果没有设置 MSH-18，则使用默认的 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，从外部系统接收的 HL7 Payloads 将被验证（请参阅 Hl7Util.generateInvalidPayloadExceptionMessage 以了解验证的详情）。如果检测到无效的有效负载，则会抛出 MllpInvalidMessageException （消费者）或 MllpInvalidAcknowledgementException。	false	布尔值

第 44 章 MOCK

仅支持生成者

对分布式和异步处理的测试非常困难。Mock、Test 和 Dataset 端点与 Camel 测试框架协同工作，从而通过使用 企业集成模式 和 Camel 的大量组件以及强大的 Bean 集成来简化您的单元和集成测试。

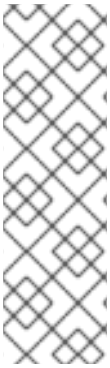
Mock 组件提供了一个强大的声明测试机制，类似于 jMock，它允许在测试开始前在任何 Mock 端点上创建声明性预期。然后，测试会运行，它通常会触发一个或多个端点的消息，最后会在测试案例中最终触发消息，以确保系统按预期工作。

这可让您测试各种问题，如下所示：

- 每个端点上收到正确的消息数量，
- 接收正确的有效负载（按正确的顺序），
- 消息按顺序到达端点，使用一些表达式来创建顺序测试功能，
- 消息到达某种类型的 Predicate，如特定的标头具有特定值，或者消息与某些 predicate 匹配，例如通过评估 XPath 或 XQuery 表达式。

注意

还有一个 Test 端点，它是一个 Mock 端点，但使用第二个端点来提供预期的消息正文列表，并自动设置 Mock 端点断言。换句话说，这是一个 Mock 端点，它会自动从文件或数据库中 的一些示例消息设置断言，例如：



注意

模拟端点无限期保留在内存中收到的交换。
请记住，Mock 旨在测试。当您向路由添加 Mock 端点时，发送到端点的每个交换都将在内存中存储（允许稍后验证），直到显式重置或 JVM 重新启动为止。如果您要发送高卷和/或大型信息，这可能会导致过量内存使用。如果您的目标是内联测试可部署的路由，请考虑在测试中使用 NotifyBuilder 或 AdviceWith，而不是直接添加 Mock 端点来路由。有两个新选项保留First，而 retainLast 可用于限制 Mock 端点保留在内存中的消息数量。

44.1. URI 格式

```
mock:someName[?options]
```

其中 someName 可以是唯一标识端点的任何字符串。

44.2. 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

44.2.1. 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。

44.2.2. 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。

44.3. 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)	在模拟收到传入的消息时打开日志记录。这将仅记录传入消息的 INFO 级别一次。如需更详细的日志记录，请将 org.apache.camel.component.mock.MockEndpoint 类的日志记录器设置为 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，则默认为 DefaultExchangeFormatter。		ExchangeFormatter

44.4. 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：

44.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
name (producer)	模拟端点 **所需的** 名称。		字符串

44.4.2. 查询参数(12 参数)

Name	描述	默认值	类型
assertPeriod (producer)	设置一个宽限期，模拟端点将重新分配，以确保初始断言仍然有效。例如，这用于精确有多个消息到达的声明。例如，如果 expectedMessageCount (int)设为 5，则在 5 个或更多消息到达时满足断言。为确保完全 5 个消息到达，您需要等待少量周期以确保没有进一步的消息到达。这是您可以使用此方法。默认情况下禁用这个周期。		long
expectedCount (producer)	指定此端点应接收的消息交换数量。beware：如果要期望 0 个消息，然后在测试启动时进行额外的操作，因为测试启动时需要设置一个断言周期，以便测试可以在一段时间内运行，以确保仍没有到达的消息；对于使用 setAssertPeriod (long)，您需要设置一个指定时间。另一种方法是使用 NotifyBuilder，并使用 notifier 知道 Camel 在对模拟调用 assertIsSatisfied （）方法之前，在路由某些消息前知道何时进行路由。这可让您不使用固定的断言周期来加快测试时间。如果您想成为完全 n 个消息到达这个模拟端点，请参阅 setAssertPeriod (long)方法以了解更多详细信息。	-1	int
failFast (producer)	设置是否 assertIsSatisfied （）是否应该在第一次检测到的失败时快速失败，同时可能会等待所有预期消息到达，然后再执行预期验证。默认为 true。设置为 false，以使用与 Camel 2.x 中一样的行为。	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
log (producer)	在模拟收到传入的消息时打开日志记录。这将仅记录传入消息的 INFO 级别一次。如需更详细的日志记录，请将 org.apache.camel.component.mock.MockEndpoint 类的日志记录器设置为 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 assertIsSatisfied （）的最短预期时间（以 millis 为单位）。		long
resultWaitTime (producer)	设置 assertIsSatisfied （）在达到前等待的最大时间（以 millis 为单位）。		long
retainFirst (producer)	指定只保留前 n 个接收的交换数。这在使用大量数据进行测试时，通过不存储每个交换端点接收的每个交换的副本来减少内存消耗。重要：在使用这个限制时，getReceivedCounter （）仍会返回实际收到的交换数量。例如，如果我们收到 5000 Exchanges，并且配置为只保留前 10 个交换，则 getReceivedCounter （）仍会返回 5000，但 getExchanges （）中只有前 10 个交换，并且 getReceivedExchanges （）方法。使用此方法时，不支持一些其他预期方法，例如 expectedBodiesReceived (Object...)在收到的第一个正文数量上设置预期。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方法。	-1	int

Name	描述	默认值	类型
retainLast (producer)	指定只保留最后 n 个接收的交换数。这在使用大量数据进行测试时，通过不存储每个交换端点接收的每个交换的副本来减少内存消耗。重要：在使用这个限制时，getReceivedCounter （）仍会返回实际收到的交换数量。例如，如果我们收到 5000 Exchanges，并且配置为只保留最后 20 个交换，则 getReceivedCounter （）仍会返回 5000，但 getExchanges （）中只有最后 20 个交换，并且 getReceivedExchanges （）方法。使用此方法时，不支持一些其他预期方法，例如 expectedBodiesReceived (Object...)在收到的第一个正文数量上设置预期。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方法。	-1	int
sleepForEmptyTest (producer)	当 expectedMessageCount (int)调用零时，允许指定 sleep 来等待此端点确实为空。		long
copyOnExchange (producer (advanced))	设置是否在这个模拟端点收到传入交换的深度副本。默认为 true。	true	布尔值

44.5. 简单示例

以下是正在使用的 Mock 端点的简单示例：首先，端点在上下文中解析。然后，我们设置预期，然后在测试运行后设定了我们预期的要求：

```
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达到预期。

当调用 assertIsSatisfied （） 时，Camel 默认会等待 10 秒。这可以通过设置 setResultWaitTime (millis) 方法来配置。

44.6. 使用 ASSERTPERIOD

满足断言后，Camel 将停止等待并继续 assertIsSatisfied 方法。这意味着，如果新消息到达模拟端点，只需要有位位，则 arrival 不会影响断言的结果。假设您要测试之后没有新消息到达一段时间，您可以通过设置 setAssertPeriod 方法来实现这一点，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();
```

44.7. 设置预期

您可以从 MockEndpoint 的 Javadoc 中看到用于设置预期的各种帮助程序方法。主要方法如下：

æ–¹æ³•	描述
expectedMessageCount(int)	在端点上定义预期的消息数。
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)	要添加期望消息按顺序收到的，请使用给定的 Expression 来比较消息。
expectsDescending(Expression)	要添加期望消息按顺序收到的，请使用给定的 Expression 来比较消息。
expectsNoDuplicates(Expression)	要添加预期没有接收重复消息的预期；使用 Expression 计算每个消息的唯一标识符。如果使用 JMS，或者消息中的一些唯一引用号，这可能类似于 **JMSMessageID**。

以下是另一个示例：

```
resultEndpoint.expectedBodiesReceived("firstMessageBody", "secondMessageBody",
"thirdMessageBody");
```

44.8. 向特定消息添加预期

另外，您可以使用 message (int messageIndex) 方法添加有关收到的特定消息的断言。

例如，要添加第一个消息的标头或正文的预期（使用零的索引，如 java.util.List），您可以使用以下代码：

```
resultEndpoint.message(0).header("foo").isEqualTo("bar");
```

camel-core 处理器测试 中使用的 Mock 端点有一些示例。

44.9. 模拟现有的端点

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。什么不同情况是，注入 Mock 端点并首先接收消息，然后将消息委派给目标端点。您可以将它视为拦截器和委托或端点监听程序。

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 功能模拟来自单元测试的给定路由中的所有端点，如下所示：

recommendationsWith 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 模拟的 endpoint>，如 mock:direct:foo。Camel 日志在 INFO 级别，即被模拟的端点：

```
INFO  Adviced endpoint [direct://foo] with mock endpoint [mock:direct:foo]
```

注意

模拟的端点没有参数

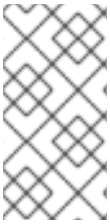
端点，这些端点会模拟它们的参数关闭。例如，端点 log:foo?showAll=true 将模拟到以下端点 模拟：log:foo。请注意，参数已被删除。

也可以仅使用模式模拟某些端点。例如，要模拟您执行 的所有日志 端点，如下所示：

recommendationsWith 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。请参见与 Camel 使用的同一匹配函数交互的更多详情。



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，这可能不合适。

44.10. 使用 CAMEL-TEST 组件模拟现有的端点

在使用 camel-test Test Kit 时，您可以轻松启用此行为，而不必使用 recommendations With 来指示 Camel 模拟端点。

同一路由可以按如下方式测试。请注意，我们从 isMockEndpoints 方法返回"Demo"，它会告知 Camel 模拟所有端点。

如果您只希望 mock 所有 log 端点，您可以返回 "log*"。

isMockEndpoints 使用 camel-test kit

```
public class IsMockEndpointsJUnit4Test extends CamelTestSupport {

    @Override
    public String isMockEndpoints() {
        // override this method and return the pattern for which endpoints to mock.
        // use * to indicate all
        return "*";
    }

    @Test
    public void testMockAllEndpoints() throws Exception {
        // notice we have automatic mocked all endpoints and the name of the endpoints is
"mock:uri"
        getMockEndpoint("mock:direct:start").expectedBodiesReceived("Hello World");
        getMockEndpoint("mock:direct:foo").expectedBodiesReceived("Hello World");
        getMockEndpoint("mock:log:foo").expectedBodiesReceived("Bye World");
        getMockEndpoint("mock:result").expectedBodiesReceived("Bye World");

        template.sendBody("direct:start", "Hello World");

        assertMockEndpointsSatisfied();

        // additional test to ensure correct endpoints in registry
        assertNotNull(context.hasEndpoint("direct:start"));
        assertNotNull(context.hasEndpoint("direct:foo"));
        assertNotNull(context.hasEndpoint("log:foo"));
        assertNotNull(context.hasEndpoint("mock:result"));
        // all the endpoints was mocked
        assertNotNull(context.hasEndpoint("mock:direct:start"));
        assertNotNull(context.hasEndpoint("mock:direct:foo"));
        assertNotNull(context.hasEndpoint("mock:log:foo"));
    }

    @Override
    protected RouteBuilder createRouteBuilder() throws Exception {
        return new RouteBuilder() {
            @Override
            public void configure() throws Exception {
                from("direct:start").to("direct:foo").to("log:foo").to("mock:result");

                from("direct:foo").transform(constant("Bye World"));
            }
```

```
        };
    }
}
```

44.11. 使用 XML DSL 模拟现有的端点

如果您不将 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。
解决方案是创建一个由单元测试使用的新 XML 文件，然后包含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，我们创建一个新的 XML 文件，该文件包括 camel-route.xml 文件，并使用类 org.apache.camel.impl.InterceptSendToMockEndpointStrategy 定义 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。

要只模拟所有 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>
```

44.12. 模拟端点并跳过发送到原始端点

有时，您希望轻松模拟并跳过发送到特定端点。因此，消息会被处理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"，以及 "direct:bar "。

recommendationsWith 模拟并跳过发送到端点

```
    @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 的同一示例

isMockEndpointsAndSkip using camel-test kit

```
public class IsMockEndpointsAndSkipJUnit4Test extends CamelTestSupport {

    @Override
    public String isMockEndpointsAndSkip() {
        // override this method and return the pattern for which endpoints to mock,
        // and skip sending to the original endpoint.
        return "direct:foo";
    }

    @Test
    public void testMockEndpointAndSkip() throws Exception {
        // notice we have automatic mocked the direct:foo endpoints and the name of the
endpoints is "mock:uri"
        getMockEndpoint("mock:result").expectedBodiesReceived("Hello World");
        getMockEndpoint("mock:direct:foo").expectedMessageCount(1);

        template.sendBody("direct:start", "Hello World");

        assertMockEndpointsSatisfied();

        // the message was not send to the direct:foo route and thus not sent to the seda
endpoint
        SedaEndpoint seda = context.getEndpoint("seda:foo", SedaEndpoint.class);
        assertEquals(0, seda.getCurrentQueueSize());
    }

    @Override
    protected RouteBuilder createRouteBuilder() throws Exception {
        return new RouteBuilder() {
            @Override
            public void configure() throws Exception {
                from("direct:start").to("direct:foo").to("mock:result");

                from("direct:foo").transform(constant("Bye World")).to("seda:foo");
            }
        };
    }
}
```

44.13. 限制要保留的消息数量

Mock 端点默认会保留它收到的每个交换的副本。因此，如果您使用很多消息进行测试，那么它将消耗内存。
我们引入了两个选项 保留First 和 retainLast，可用于仅保留 N of first 和/或最后一个交换。

例如，在下面的代码中，我们只想保留前 5 个副本，最后 5 个交换接收模拟。

```
MockEndpoint mock = getMockEndpoint("mock:data");
mock.setRetainFirst(5);
mock.setRetainLast(5);
mock.expectedMessageCount(2000);

mock.assertIsSatisfied();
```

使用此功能有一些限制。MockEndpoint 上的 getExchanges （） 和 getReceivedExchanges （） 方法只返回 Exchanges 的保留副本。因此，在上面的示例中，列表将包含 10 个交换；前五个和最后五个。
retainFirst 和 retainLast 选项还对您可以使用的预期方法有限制。例如，在消息正文、标头等上工作 的预期XXX 方法将仅对保留的消息进行操作。在上例中，它们只能测试保留 10 个消息的预期。

44.14. 使用 ARRIVAL 时间进行测试

Mock 端点将消息的 arrival 时间存储为交换上的属性

```
Date time = exchange.getProperty(Exchange.RECEIVED_TIMESTAMP, Date.class);
```

您可以使用这些信息知道消息何时到达模拟。但是，它还为了解之前和下一个消息之间的时间间隔提供基础。您可以使用此选项在 Mock 端点上设置使用 arrives DSL 的预期。

例如，在下一个操作前，第一个消息应该在 0-2 秒之间到达：

```
mock.message(0).arrives().noLaterThan(2).seconds().beforeNext();
```

您还可以将其定义为第二条消息（基于 0 索引）在上一个消息后没有超过 0-2 秒：

```
mock.message(1).arrives().noLaterThan(2).seconds().afterPrevious();
```

您还可以使用在 之间设置较低绑定。例如，假设它应该在 1-4 秒之间：

```
mock.message(1).arrives().between(1, 4).seconds().afterPrevious();
```

您还可以对所有信息设置预期，例如，指出它们间的差距应该最多 1 秒：

```
mock.allMessages().arrives().noLaterThan(1).seconds().beforeNext();
```

注意

上面的示例中，
使用 秒 作为时间单位，但 Camel 也提供 毫秒 和 分钟。

44.15. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用模拟时，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>
```

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
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，则默认为 DefaultExchangeFormatter。选项是 org.apache.camel.spi.ExchangeFormatter 类型。		ExchangeFormatter

Name	描述	默认值	类型
camel.component.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	在模拟收到传入的消息时打开日志记录。这将仅记录传入消息的 INFO 级别一次。如需更详细的日志记录，请将 org.apache.camel.component.mock.MockEndpoint 类的日志记录器设置为 DEBUG 级别。	false	布尔值

第 45 章 MONGODB

支持生成者和消费者

根据 Wikipedia："NoSQL 是一个移动提升了松散定义的非关系数据存储类，通过大量关系数据库和 ACID 保证进行破坏"。近年来，Croio 解决方案已持续增长，并且主要使用极端的站点和服务（如 Ice、LinkedIn、Faceter 等）被广泛使用它们来实现可伸缩性和创新。

基本上，Troio 解决方案与传统的 RDBMS （相关数据库管理系统）不同，它们不使用 SQL 作为其查询语言，一般不提供类似 ACID 的事务行为或关系数据。相反，它们围绕灵活的数据结构和模式的概念（这意味着，具有固定模式的数据库表的传统概念被丢弃），在专有硬件和 blazing-fast 处理方面具有非常可扩展性。

MongoDB 是一个非常流行的，Camel-mongodb 组件将 Camel 与 MongoDB 集成，允许您将 MongoDB 集合作为制作者（集合上的操作）和消费者（来自 MongoDB 集合中的文档）交互。

MongoDB 围绕文档的概念（而非办公室文档），而是在 JSON/BSON 和集合中定义的分层数据。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。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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

45.1. URI 格式

```
mongodb:connectionBean?
database=databaseName&collection=collectionName&operation=operationName[&moreOptions...]
```

45.2. 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

45.2.1. 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。

45.2.2. 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。

45.3. 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。从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

45.4. 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：

45.4.1. 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

45.4.2. 查询参数(27 参数)

Name	描述	默认值	类型
collection (common)	设置 MongoDB 集合的名称，以绑定到此端点。		字符串
collectionIndex (common)	设置集合索引(JSON FORMAT : \\{ field1 : order1, field2 : order2})。		字符串
createCollection (common)	如果集合不存在，在初始过程中创建集合。默认值为 true。	true	布尔值
数据库 （通用）	将 MongoDB 数据库的名称设置为 target。		字符串
hosts (common)	host:port 格式的 mongodb 服务器的主机地址。也可以使用多个地址，作为以逗号分隔的主机列表：host1:port1,host2:port2。如果指定了 hosts 参数，则忽略提供的 connectionBean。		字符串
mongoConnection (common)	设置用作连接到数据库的客户端的连接 bean。		MongoClient
operation (common)	设置此端点将根据 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 文档或 MongoIterable。DocumentList 或 MongoIterable 适用于 findAll 和 aggregate。文档适用于所有操作。 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 尾部光标将阻止，直到新数据到达为止。如果没有插入新数据，则光标将在一定时间后由 MongoDB 服务器自动释放和关闭。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，在下一次尝试前尝试失败。默认值为 1000ms。	1000	long
动态性(advanced)	设置此端点是否尝试从传入的 Exchange 属性中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。默认情况下禁用它以提高性能。启用它将需要最少的性能命中。	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 • 最接近的	PRIMARY	字符串
writeConcern (advanced)	使用从 MongoDB 请求的确认级别配置 connection bean，以便将操作写入独立 mongod、replicaset 或 cluster。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 • 最多	已确认	字符串

Name	描述	默认值	类型
writeResultAsHeader (advanced)	在写入操作中，它会确定是否将 WriteResult 返回为 OUT 消息的正文，我们将 IN 消息传送到 OUT，并将 WriteResult 作为标头附加。	false	布尔值
streamFilter (changeStream)	过滤更改流消费者的条件。		字符串
密码 （安全）	mongodb 连接的用户密码。		字符串
用户名 （安全）	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跟踪者托管多个跟踪程序。为了保持它们独立的，每个 tracker 应该都有自己的唯一的 persistentId。		字符串
persistentTailTracking (tail)	启用持久的尾部跟踪，这是在系统重启后跟踪最后一次消耗的消息的机制。下次系统启动时，端点将从最后停止记录的点恢复光标。	false	布尔值
tailTrackCollection (tail)	将保留尾部跟踪信息的集合。如果没有指定，则默认使用 MongoDbTailTrackingConfig114DEFAULT_COLLECTION。		字符串
tailTrackDb (tail)	指明尾部跟踪机制将保留哪些数据库。如果没有指定，则默认获取当前数据库。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)	将放置最后一个跟踪值的字段。如果没有指定，则默认使用 MongoDbTailTrackingConfig114DEFAULT_FIELD。		字符串
tailTrackIncreasingField (tail)	传入记录中的关联字段是增加的，将在每次生成时都用于定位尾部光标。光标将（重新）通过一个 type: tailTrackIncreasingField 大于 lastValue 的查询创建（可能从持久尾部跟踪中恢复）。可以是 Integer, Date, String 等类型。注意：当前不支持点表示法，因此字段应位于文档的最顶层。		字符串

45.5. 在 SPRING XML 中配置数据库

以下 Spring XML 创建了定义与 MongoDB 实例连接的 bean。

从 mongo java 驱动程序 3 开始， WriteConcern 和 readPreference 选项不会被动态修改。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>
```

45.6. 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>
```

45.7. MONGODB 操作 - PRODUCER 端点

45.7.1. 查询操作

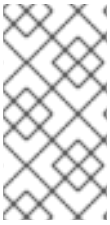
45.7.1.1. findById

此操作仅从集合中检索一个元素，其 _id 字段与 IN 消息正文的内容匹配。传入的对象可以是任何等同于 Bson 类型的对象。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 和 ObjectId 类型处理，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
此操作支持投射运算符。请参阅 指定字段过滤器（项目）。

45.7.1.2. 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，查询选择器应当为 Bson 类型，或转换为 Bson （例如，JSON 字符串或 HashMap）。如需更多信息，请参阅类型转换。

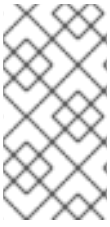
使用 MongoDB 驱动程序提供的 过滤器 创建查询选择器。

45.7.1.3. 没有查询选择器的示例（在集合中显示第一个文档）

```
from("direct:findOneByQuery")
   .to("mongodb:myDb?database=flights&collection=tickets&operation=findOneByQuery")
   .to("mock:resultFindOneByQuery");
```

45.7.1.4. 带有查询选择器的示例（在集合中调整第一个匹配文档）：

```
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。

45.7.1.5. findAll

findAll 操作返回与查询匹配的所有文档，或者根本不没有，在这种情况下，集合中包含的所有文档都会被返回。查询对象提取 CamelMongoDbCriteria 标头。如果 CamelMongoDbCriteria 标头是 null，查询对象将被提取消息正文，即它应该类型 Bson 或转换为 Bson。它可以是 JSON 字符串或 Hashmap。如需更多信息，请参阅类型转换。

45.7.1.5.1. 没有查询选择器的示例（重复集合中的所有文档）

```
from("direct:findAll")
   .to("mongodb:myDb?database=flights&collection=tickets&operation=findAll")
   .to("mock:resultFindAll");
```

45.7.1.5.2. 带有查询选择器的示例（重复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有效的检索：

标头键	快速常数	描述（从 MongoDB API 文档中提取）	预期类型
CamelMongoDbNumToSkip	**MongoDbConstants.NUM_TO_SKIP**	在光标开始时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。光标通常获取一系列结果对象，并将它们存储在本地。如果 batchSize 为正状态，它代表检索的每个对象批处理的大小。可以调整它以优化性能和限制数据传输。如果 batchSize 为负数，它将限制返回的数量对象，这适合最大批处理大小限制（通常为 4MB），而光标将被关闭。例如，如果 batchSize 是 -10，那么服务器将返回最多 10 个文档，并且尽可能在 4MB 中，然后关闭光标。请注意，这个功能与其中的 limit （）不同，该文档必须适合最大大小，而且无需发送请求来关闭光标服务器端。即使光标迭代后也可以更改批处理大小，在这种情况下，设置将在下一个批处理检索中应用。	int/Integer

标头键	快速常数	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。这是自 MongoDB Server 4.3.1 起的支持。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/Boolean

45.7.1.5.3. 带有选项 outputType=MongoIterable 和 batch 大小的示例

```
from("direct:findAll")
   .setHeader(MongoDbConstants.BATCH_SIZE).constant(10)
   .setHeader(MongoDbConstants.CRITERIA, constant(Filters.eq("name", "Raul Kripalani")))
   .to("mongodb:myDb?
database=flights&collection=tickets&operation=findAll&outputType=MongoIterable")
   .to("mock:resultFindAll");
```

findAll 操作也会返回以下 OUT 标头，以便在使用分页时迭代结果页面：

标头键	快速常数	描述（从 MongoDB API 文档中提取）	数据类型
CamelMongoDbResultTotalSize	**MongoDbConstants.RESULT_TOTAL_SIZE**	与查询匹配的对象数量。这不会考虑限制/skip。	int/Integer
CamelMongoDbResultPageSize	**MongoDbConstants.RESULT_PAGE_SIZE**	与查询匹配的对象数量。这不会考虑限制/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。

45.7.1.6. 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。请注意，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);
```

您可以提供一个查询 查询对象，它会被提取 CamelMongoDbCriteria 标头。如果 CamelMongoDbCriteria 标头为 null，查询对象将被提取消息正文，即它应该类型为 Bson 或转换为 Bson.，操作将返回与此条件匹配的文档数量。

```
Document query = ...
Long count = template.requestBodyAndHeader("direct:count", query,
MongoDbConstants.COLLECTION, "dynamicCollectionName");
```

45.7.1.7. 指定字段过滤器（项目）

默认情况下，查询操作将返回整个匹配对象（及其所有字段）。如果您的文档较大，且您只需要检索其字段的子集，您可以在所有查询操作中指定字段过滤器，只需设置相关的 Bson （或类型转换为 Bson，如 JSON 字符串、映射等）。

以下是一个示例，它使用 MongoDB 的 Projections 来简化 Bson 的创建。它检索 _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 _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);
```

45.7.1.8. 指定 sort 子句

根据特定字段排序，通常会要求从集合中获取 min/max 记录，它使用 MongoDB 的 Sorts 来简化 Bson 的创建。它检索 _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 标头可与 findOneByQuery 操作一起使用，以实现相同的结果。如果还指定了 FIELDS_PROJECTION 标头，则操作将返回可直接传递给另一个组件（例如，参数化 MyBatis SELECT 查询）的单个字段/值对。本例演示了从集合中获取最新文档，并根据 documentTimestamp 字段将结果减少到单个字段：

```
.from("direct:someTriggeringEvent")
.setHeader(MongoDbConstants.SORT_BY).constant(Sorts.descending("documentTimestamp"
))
.setHeader(MongoDbConstants.FIELDS_PROJECTION).constant(Projection.include("documen
tTimestamp"))
.setBody().constant("{}")
.to("mongodb:myDb?
database=local&collection=myDemoCollection&operation=findOneByQuery")
.to("direct:aMyBatisParameterizedSelect");
```

45.7.2. 创建/更新操作

45.7.2.1. insert

将新对象插入到 MongoDB 集合中，从 IN 消息正文获取。类型 conversion 试图将其转换为 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，端点将期望任何类型的对象列表、阵列或集合，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");
```

此操作将返回 WriteResult，具体取决于 WriteConcern 或 invokeGetLastError 选项的值，getLastError （） 将已被调用。如果您希望访问些操作的最终结果，可以通过在 WriteResult 上调用 getLastError() 或 getCachedLastError() 来获取 CommandResult。然后，您可以通过调用 CommandResult.ok （）、CommandResult.getErrorMessage （） 和/或 CommandResult.getException （） 来验证结果。

请注意，新对象的 _id 必须在集合中唯一。如果没有指定值，MongoDB 将为您自动生成值。但是，如果您确实指定它且不是唯一的，则插入操作将失败（对于 Camel 通知，则需要启用

invokeGetLastError 或设置等待写入结果的 WriteConcern。

这不是组件的一个限制，而是在 MongoDB 中用于更高的吞吐量。如果您使用自定义 _id，则需要确保应用程序级别是唯一的（也是良好的做法）。

插入的记录的 OID 存储在 CamelMongoOid 键(MongoDbConstants.OID constant)下的消息标头中。存储的值是 org.bson.types.ObjectId，用于单个插入或 java.util.List<org.bson.types.ObjectId>（如果已插入了多个记录）。

在 MongoDB Java Driver 3.x 中，insertOne 并插入Many 操作会返回 void。Camel 插入操作返回 Document 或 插入的文档列表。请注意，如果需要，每个文档都会由新的 OID 更新。

45.7.2.2. save

save 操作等同于一个 upsert (UPdate, inSERT)操作，其中将更新记录，如果它不存在，则会将其插入到一个原子操作中。MongoDB 将根据 _id 字段执行匹配。

如果进行更新，则对象会被完全替换，不允许使用 MongoDB 的 $modifiers。因此，如果要操作对象（如果已存在），有两个选项：

1. *执行查询以首先检索整个对象及其所有字段（不效率），在 Camel 中更改它，然后保存它。*
2. *使用带有 $modifiers 的 update 操作，该操作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，则操作将是一个插入，创建的新 _id 将放置在 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);
```

45.7.2.3. update

更新集合上的一个或多个记录。需要过滤查询和更新规则。

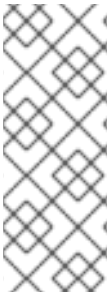
您可以使用 MongoDBConstants.CRITERIA 标头作为 Bson 定义过滤器，并将更新规则定义为 Body 中的 Bson。

注意

在增强
While 定义过滤器 后，使用 MongoDBConstants.CRITERIA 标头作为 Bson 在进行更新前查询 mongodb，您应该注意到，如果您使用带有聚合策略的增强模式，则需要在聚合过程中将其从生成的 camel 交换中删除，然后应用 mongodb 更新。如果您在聚合和/或重新定义 MongoDBConstants.CRITERIA 标头期间没有删除此标头，则在将 camel 交换发送到 mongodb producer 端点前，可能会以无效的 camel 交换有效负载结束。

第二种需要 List<Bson> 作为 IN 消息正文，其中包含完全 2 个元素：

- 元素 1 (index 0) IANA 过滤器查询 IANA 决定了哪些对象将受到影响，与典型的查询对象相同
- 元素 2 （索引 1）Demo 更新规则如何更新匹配对象。支持 MongoDB 中的所有 修饰符操作。



注意

多更新
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 匹配的记录，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)，带有更新的记录数量（从 WriteResult.getN （）中获得）。

支持以下 IN 消息标头：

标头键	快速常数	描述（从 MongoDB API 文档中提取）	预期类型
CamelMongoDbMultiUpdate	**MongoDbConstants.MULTIUPDATE**	如果更新应该应用到所有对象匹配。请参阅 http://www.mongodb.org/display/DOCS/Atomic+Operations	布尔值/Boolean
CamelMongoDbUpsert	**MongoDbConstants.UPSERT**	如果数据库应该创建元素（如果不存在）	布尔值/Boolean

例如，以下命令将通过将 " scientist " 字段的值设置为 "Darwin" 字段来更新其 filterField 字段等于 true 的所有记录：

```
// route: from("direct:update").to("mongodb:myDb?
database=science&collection=notableScientists&operation=update");
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);
```

45.7.3. 删除操作

45.7.3.1. remove

从集合中删除匹配的记录。IN 消息正文将充当移除过滤器查询，预期为 DBObject 类型或可转换的类型。

以下示例将删除其字段 'conditionField' 等于 true 的所有对象，在科学数据库中，notableScientists 集合中：

```
// 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)。

45.7.4. 批量写入操作

45.7.4.1. bulkWrite

批量执行写入操作，并控制执行顺序。需要一个 List<WriteModel<Document >> 作为 IN 消息正文，其中包含用于插入、更新和删除操作的命令。

以下示例将插入一个新的科学家 "Pierre Curie"，更新 id 为 "5" 的记录，将 "scientist" 项的值设置为 "Marie Curie" 并删除 id 为 "3" 的记录：

```
// route: from("direct:bulkWrite").to("mongodb:myDb?
database=science&collection=notableScientists&operation=bulkWrite");
List<WriteModel<Document>> bulkOperations = Arrays.asList(
        new InsertOneModel<>(new Document("scientist", "Pierre Curie")),
        new UpdateOneModel<>(new Document("_id", "5"),
                new Document("$set", new Document("scientist", "Marie Curie"))),
        new DeleteOneModel<>(new Document("_id", "3")));

BulkWriteResult result = template.requestBody("direct:bulkWrite", bulkOperations,
BulkWriteResult.class);
```

默认情况下，操作按顺序执行，并在第一个写入错误上中断，而不处理列表中的任何剩余的写入操作。要指示 MongoDB 继续处理列表中的剩余的写入操作，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	快速常数	描述（从 MongoDB API 文档中提取）	预期类型
CamelMongoDbBulkOrdered	**MongoDbConstants.BULK_ORDERED**	执行排序或未排序的操作执行。默认值为 true。	布尔值/Boolean

45.7.5. 其他操作

45.7.5.1. 聚合

使用正文中包含的给定管道执行聚合。聚合可能比较长且大量操作。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	快速常数	描述（从 MongoDB API 文档中提取）	预期类型
CamelMongoDbBatchSize	MongoDbConstants.BATCH_SIZE	设置每个批处理要返回的文档数。	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/Boolean

默认情况下，返回所有结果的列表。根据结果的大小，这可能会大量内存。更安全的替代方案是设置 outputType=MongoIterable。接下来的处理器将在消息正文中看到一个 iterable，允许它逐一执行结果。因此，设置批处理大小并返回可迭代功能可以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 ()) 足以逐一发送路由条目，但它仍会首先将所有条目加载到内存中。因此，需要调用 .streaming ()，以便批量将数据加载到内存中。

45.7.5.2. 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);
```

操作将返回与 shell 中显示的数据结构类似，格式为 OUT 消息正文中的 文档。

45.7.5.3. 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);
```

操作将返回与 shell 中显示的数据结构类似，格式为 OUT 消息正文中的 文档。

45.7.5.4. 命令

作为命令在数据库上运行正文。对于获取主机信息、复制或分片状态时，管理员操作非常有用。

集合参数不适用于此操作。

```
// route: from("command").to("mongodb:myDb?database=science&operation=command");
DBObject commandBody = new BasicDBObject("hostInfo", "1");
Object result = template.requestBody("direct:command", commandBody);
```

45.7.6. 动态操作

Exchange 可以通过设置由 MongoDbConstants.OPERATION_HEADER 常量定义的 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 operation 参数的值匹配。

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);
```

45.8. 消费者

有几种类型的消费者：

1. Tailable Cursor Consumer
2. 更改流消费者

45.8.1. Tailable Cursor Consumer

MongoDB 提供了即时使用集合中持续数据的机制，具体方法是保持光标打开，就如同 slirpnix 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到客户端，因此这种机制比调度的轮询效率要高得多，而不是让客户端以重新发出调度间隔来获取新数据。它还可减少其他冗余网络流量。

只有一个先决条件使用尾部光标：集合必须是"捕获集合"，这意味着它只存放 N 个对象，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 组件实现了一个尾部的光标消费者，供您在 Camel 路由中使用此功能。插入新对象后，MongoDB 会按照自然顺序将其推送到您的尾部光标消费者，这些消费者将它们转换为交换并触发您的路由逻辑。

45.9. 尾部光标消费者如何工作

要将光标转换为可尾部光标的光标，在第一次生成光标时，会将几个特殊标志发送到 MongoDB。创建后，光标将保持打开状态，并在调用 MongoCursor.next （） 方法时阻止，直到新数据到达新数据。但是，如果在确定的句点后没有出现新数据，MongoDB 服务器保留自己终止您的光标的权利。如果您希望继续使用新数据，您必须重新生成光标。为此，您必须记住您离开的位置，否则您将重新从顶部开始消耗。

Camel MongoDB 尾部光标消费者为您负责所有这些任务。您只需要为您的数据中的某些字段提供密钥，它在每次重新生成时都会作为位置的位置，例如：时间戳、顺序 ID 等。它可以是 MongoDB 支持的任何数据类型。找到日期、字符串和整数才能正常工作。在此组件上下文中，我们调用这个机制"需要跟踪"。

消费者将记住此字段的最后值，每当重新生成光标时，它将使用类似过滤器的过滤器运行查询，如 increasingField > lastValue，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 字段的密钥。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" 无法正常工作。如果您需要支持嵌套增加字段，请在 Camel JIRA 中创建一个问题单。

光标重新生成延迟： 要注意的是，如果初始操作中没有新数据，MongoDB 将立即终止光标。由于我们不希望在这种情况下对服务器造成负担，因此引入了 cursorRegenerationDelay 选项（默认值为 1000ms），您可以对其进行修改以满足您的需要。

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 collection，使用 "departureTime" 作为增加字段，默认的重新生成光标延迟为 1000ms。

45.10. 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，但您的最后一个值将会丢失，而您的尾部光标消费者将再次从顶部消耗，非常有可能将重复的记录发送到路由中。

要解决这种情况，您可以启用 持久的尾部跟踪 功能来跟踪 MongoDB 数据库中的特殊集合中消耗的值。当消费者再次进行初始化时，它将恢复最后的值，并在没有任何发生时继续。

最后读取值会保留在两个 occasions 上：每次重新生成光标和消费者关闭时，都会保留。我们以后可

能会考虑定期保留（每 5 秒清空一次），以便在需要时添加持久性。要请求此功能，请在 Camel JIRA 中打开一个 ticket。

45.11. 启用持久的尾部跟踪

要启用此功能，请在端点 URI 中至少设置以下选项：

- persistentTailTracking 选项为 true
- persistentId 选项指向这个消费者的唯一标识符，以便在许多消费者间重复使用相同的集合

另外，您可以将 tailTrackDb,tailTrackCollection 和 tailTrackField 选项设置为自定义存储运行时信息的位置。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，以下路由将使用"departureTime"作为增加字段的"flights.cancellations"大写集合，默认的重新生成光标延迟为 1000ms，持久性尾部跟踪打开，并在"flights.camelTracking" ID 下保留。将最后处理的值存储在"lastTrackingValue"字段下(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");
```

以下是与上述示例相同，但持久性尾部跟踪运行时信息将存储在"trackers.camelTrackers"集合中，位于"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");
```

45.11.1. 更改流消费者

通过更改流，应用程序可以访问实时数据更改，而不会降低 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 属性是标准的 MongoDB $match 聚合。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 值外部化成属性占位符，允许清理端点 URI 参数并更轻松地读取。

changeStreams 消费者类型也会返回以下 OUT 标头：

标头键	快速常数	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, 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。对于分片集合，还显示文档的完整分片键。如果 _id 字段已经是分片密钥的一部分，则不会重复它。	ObjectId

45.12. 类型转换

camel-mongodb 组件中包含的 MongoDbBasicConverters 类型转换器提供以下转换：

Name	from type	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**	**已实施 映射。**
fromStringToDocument	**字符串**	**文档**	使用 **com.mongodb.Document.parse （字符串 s）。**
fromStringToObjectId	**字符串**	**ObjectId**	通过新的 **ObjectId s 构建新的 ObjectId。**
fromFileToDocument	**File**	**文档**	在 hood 下使用 **fromInputStreamToDocument**
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>。

这个类型转换器是自动发现的，因此您不需要手动配置任何内容。

45.13. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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
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。从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 46 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，基于 Netty 项目版本 4。
Netty 是一个 NIO 客户端服务器框架，它允许快速轻松地开发 networkServerInitializerFactory 应用程序，如协议服务器和客户端。
Netty 大大简化和简化网络编程，如 TCP 和 UDP 套接字服务器。

此 camel 组件支持制作者和消费者端点。

Netty 组件有几个选项，允许对多个 TCP/UDP 通信参数（缓冲大小、inAlives、tcpNoDelay 等）进行精细控制，并促进 Camel 路由上的 In-Only 和 In-Out 通信。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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

46.1. URI 格式

netty 组件的 URI 方案如下

```
netty:tcp://0.0.0.0:99999[?options]
netty:udp://remotehost:99999/[?options]
```

此组件支持 TCP 和 UDP 的制作者和消费者端点。

46.2. 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

46.2.1. 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，组件可能有安全设置、用于身份验证的凭证、用于网络连接的 url 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，或者直接使用 Java 代码完成。

46.2.2. 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。

46.3. 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)	是否在使用后是否断开(close)与 Netty Channel 的连接。可用于消费者和生产者。	false	布尔值
keepalive (common)	设置以确保不活动而关闭套接字。	true	布尔值
reuseAddress (common)	设置以方便套接字多路。	true	布尔值
reuseChannel (common)	此选项允许生产者和消费者（客户端模式）在处理交换的生命周期内重复使用相同的 Netty Channel。如果您需要在 Camel 路由中多次调用服务器并希望使用相同的网络连接，这非常有用。使用此选项时，频道不会返回到连接池，直到交换完成为止；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交换属性上，其键为 NettyConstants，Y_CHANNEL 允许您在路由期间获取频道并使用它。	false	布尔值
sync (common)	设置为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)	设置以选择多播通过 UDP。	false	布尔值
clientMode (consumer)	如果 clientMode 为 true，netty consumer 会将地址连接为 TCP 客户端。	false	布尔值
reconnect (consumer)	只在消费者中的 clientMode 中使用，消费者将尝试断开连接时重新连接。	true	布尔值
reconnectInterval (consumer)	如果启用了 reconnect 和 clientMode，则使用。尝试重新连接的时间间隔（以毫秒为单位）。	10000	int

Name	描述	默认值	类型
Back log (consumer (advanced))	允许为 netty consumer (server)配置积压。请注意，回放是根据操作系统的最佳工作。将这个选项设置为 200、500 或 1000 等值，请告知 TCP 堆栈如果未配置此选项，则接受队列的时长，则 backlog 依赖于 OS 设置。		int
bossCount (consumer (advanced))	当 netty 在 nio 模式上工作时，它会使用来自 Netty 的默认 bossCount 参数，即 1。用户可以使用这个选项覆盖 Netty 中的默认 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，则此选项指定 NettyConsumer （如果它应该断开任何回复的回复），则此选项指定 NettyConsumer。	true	布尔值
executorService (consumer (advanced))	使用给定的 EventExecutorGroup。		EventExecutorGroup
maximumPoolSize (consumer (advanced))	为 netty consumer 排序的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 将使用 10 个线程，除非 2 x cpu_core 加上 1 的值更高，然后会覆盖和使用它。例如，如果存在 8 个内核，则消费者线程池将为 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对消息排序，并在某些消息被阻止时，则 nettys worker 线程(event loop)不受影响。		int
nettyServerBootstrapFactory (consumer (advanced))	使用自定义 NettyServerBootstrapFactory。		NettyServerBootstrapFactory
networkInterface (consumer (advanced))	使用 UDP 时，此选项可用于按名称指定网络接口，如 eth0 以加入多播组。		字符串

Name	描述	默认值	类型
noReplyLogLevel (consumer (advanced))	如果启用了同步，则指定 NettyConsumer 在日志记录没有回复时要使用的日志级别。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	WARN	LoggingLevel
serverClosedChannelExceptionCaughtLogLevel (consumer (advanced))	如果服务器(NettyConsumer)捕获了 java.nio.channels.ClosedChannelException，则它会使用此日志级别登录。这用于避免记录关闭的通道异常，因为客户端可能会立即断开连接，从而导致 Netty 服务器出现关闭异常。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	DEBUG	LoggingLevel
serverExceptionCaughtLogLevel (consumer (advanced))	如果服务器(NettyConsumer)捕获异常，则它会使用此日志级别记录其日志记录。 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
usingExecutorService (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)	等待套接字连接可用时的时间。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，允许将超时用于 Netty producer。默认情况下不使用超时。该值以秒为单位，因此 30000 为 30 秒。requestTimeout 使用 Netty 的 ReadTimeoutHandler 来触发超时。		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将 request/reply 与 netty producer 搭配使用时如何映射请求和回复消息。只有在您有一个将请求与回复一起映射（例如，请求和回复消息中存在关联 ID ）时，才应使用此项。如果要在 netty 中在同一频道（连接连接）上多路复用。执行此操作时，您必须有一个方法来关联请求和回复消息，以便您可以在继续路由前将正确的回复存储在动态 Camel Exchange 中。在构建自定义关联经理时，我们建议扩展 TimeoutCorrelationManagerSupport。这提供了对其他需要实现的超时和其他复杂性的支持。如需了解更多详细信息，请参阅 producerPoolEnabled 选项。		NettyCamelStateCorrelationManager
lazyChannelCreation (producer (advanced))	如果远程服务器在 Camel 生成者启动时没有启动并运行，则频道可能被创建以避免异常。	true	布尔值

Name	描述	默认值	类型
producerPoolEnabled (producer (advanced))	生成者池是否已启用。重要：如果您关闭此，那么也会将单个共享连接用于生成者，即使您正在执行请求/回复。这意味着，如果回复未排序，则响应交集存在潜在的问题。因此，您需要在请求和回复消息中有一个关联 ID，以便您可以将回复正确与负责继续处理 Camel 中的消息的 Camel 回调关联。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))	设置池中闲置实例数量的上限。	100	int
producerPoolMaxTotal (producer (advanced))	设置池可分配的对象数量的上限（检查到客户端，或闲置等待签出）在给定时间闲置。对没有限制使用负值。	-1	int
producerPoolMinEvictableIdle (producer (advanced))	在被闲置对象驱除前，设置对象可能闲置的最小时间（值为 millis）。	300000	long
producerPoolMinIdle (producer (advanced))	在驱除器线程（如果处于活动状态）前设置制作者池中允许的最小实例数量（如果处于活动状态）。		int
udpConnectionlessSending (producer (advanced))	这个选项支持较少的 udp 发送的连接，这是实际触发和忘记。如果没有侦听接收端口，则连接的 udp 会接收 PortUnreachableException。	false	布尔值
useByteBuf (producer (advanced))	如果 useByteBuf 为 true，netty producer 会在发送前将消息正文转换为 ByteBuf。	false	布尔值
hostnameVerification (security)	在 SSLEngine 上启用/禁用主机名验证。	false	布尔值
allowSerializedHeaders (advanced)	仅在 transferExchange 为 true 时用于 TCP。当设置为 true 时，标头和属性中的可序列化对象将添加到交换中。否则，Camel 将排除任何不可序列化的对象，并将它记录在 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)	使用显式 ChannelGroup。		ChannelGroup
nativeTransport (advanced)	是否使用原生传输而不是 NIO。原生传输利用主机操作系统，且仅在某些平台上被支持。您需要为您要使用的主机操作系统添加 netty JAR。详情请查看：	false	布尔值
options (advanced)	允许使用 option. 作为前缀来配置额外的 netty 选项。例如 option.child.keepAlive=false，来设置 netty 选项 child.keepAlive=false。有关可以使用的可能选项，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。详情请参阅 Jetty 文档和此邮件线程。		int
sendBufferSize (advanced)	在出站通信期间使用的 TCP/UDP 缓冲区大小。大小为字节。	65536	int
transferExchange (advanced)	仅用于 TCP。您可以在有线线上传输交换，而不只是正文。以下字段会被传输： 在 body, Out body, fault body, In headers, Out headers, fault headers, Exchange properties, exchange exception。这要求对象可以序列化。Camel 将排除任何不可序列化的对象，并将它记录在 WARN 级别。	false	布尔值
udpByteArrayCodec (advanced)	仅限 UDP。如果启用了使用字节数组 codec 而不是 Java 序列化协议。	false	布尔值
workerCount (advanced)	当 netty 在 nio 模式上工作时，它会使用来自 Netty 的默认 workerCount 参数（即 cpu_core_threads x 2）。用户可以使用这个选项覆盖 Netty 中的默认 workerCount。		int
workerGroup (advanced)	使用显式 EventLoopGroup 作为 boss 线程池。例如，与多个消费者或制作者共享线程池。默认情况下，每个消费者或生成者都有自己的 worker 池，具有 2 个 x cpu 计数内核线程。		EventLoopGroup

Name	描述	默认值	类型
allowDefaultCodec (codec)	如果两者都默认 codec，netty 组件会安装一个默认的 codec，编码器/解码器为 null，文本行为 false。将 allowDefaultCodec 设置为 false 可防止 netty 组件安装默认 codec 作为过滤器链中的第一个元素。	true	布尔值
autoAppendDelimiter (codec)	在使用文本 codec 发送时，是否自动附加缺少的最终分隔符。	true	布尔值
decoderMaxLineLength (codec)	文本代码的最大行长度。	1024	int
解码器 (codec)	要使用的解码器列表。您可以使用用逗号分开的值的字符串，并在 Registry 中查找值。只需记住使用 192.168.1.0/24 为值添加前缀，因此 Camel 知道它应该查找。		list
Delimiter (codec)	用于文本代码的分隔符。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。您可以使用用逗号分开的值的字符串，并在 Registry 中查找值。只需记住使用 192.168.1.0/24 为值添加前缀，因此 Camel 知道它应该查找。		list
编码 (codec)	用于文本代码的编码(charset 名称)。如果没有提供，Camel 将使用 JVM 默认 Charset。		字符串
文本行 (codec)	仅用于 TCP。如果没有指定 codec，您可以使用此标志来指示基于文本的 codec；如果没有指定，则为 false，则通过 TCP 假设对象序列化，但默认只允许对字符串进行序列化。	false	布尔值
enabledProtocols (security)	使用 SSL 时要启用哪个协议。	TLSv1, TLSv1.1, TLSv1.2	字符串
keyStoreFile (security)	用于加密的客户端证书密钥存储。		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）	使用 SSH 加密/解密有效负载的密码设置。		字符串
securityProvider (security)	用于有效负载加密的安全提供程序。如果没有设置，则默认为 SunX509。		字符串
SSL （安全）	设置以指定 SSL 加密是否应用到此端点。	false	布尔值
sslClientCertHeaders (security)	启用并在 SSL 模式中，Netty consumer 将增强 Camel Message，其标头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 SSLContext 参数配置安全性：		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

46.4. 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：

46.4.1. 路径参数(3 参数)

Name	描述	默认值	类型
protocol (common)	**必需** 使用 tcp 或 udp 的协议。 Enum 值： • tcp • udp		字符串
host (common)	**必需** 主机名。对于消费者，主机名为 localhost 或 0.0.0.0。对于制作者，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

46.4.2. 查询参数(71 参数)

Name	描述	默认值	类型
disconnect (common)	是否在使用后是否断开(close)与 Netty Channel 的连接。可用于消费者和生产者。	false	布尔值
keepalive (common)	设置以确保不活动而关闭套接字。	true	布尔值
reuseAddress (common)	设置以方便套接字多路。	true	布尔值
reuseChannel (common)	此选项允许生产者和消费者（客户端模式）在处理交换的生命周期内重复使用相同的 Netty Channel。如果您需要在 Camel 路由中多次调用服务器并希望使用相同的网络连接，这非常有用。使用此选项时，频道不会返回到连接池，直到交换完成为止；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交换属性上，其键为 NettyConstants，Y_CHANNEL 允许您在路由期间获取频道并使用它。	false	布尔值
sync (common)	设置为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)	设置以选择多播通过 UDP。	false	布尔值
clientMode (consumer)	如果 clientMode 为 true，netty consumer 会将地址连接为 TCP 客户端。	false	布尔值
reconnect (consumer)	只在消费者中的 clientMode 中使用，消费者将尝试断开连接时重新连接。	true	布尔值
reconnectInterval (consumer)	如果启用了 reconnect 和 clientMode，则使用。尝试重新连接的时间间隔（以毫秒为单位）。	10000	int
Back log (consumer (advanced))	允许为 netty consumer (server)配置积压。请注意，回放是根据操作系统的最佳工作。将这个选项设置为 200、500 或 1000 等值，请告知 TCP 堆栈如果未配置此选项，则接受队列的时长，则 backlog 依赖于 OS 设置。		int
bossCount (consumer (advanced))	当 netty 在 nio 模式上工作时，它会使用来自 Netty 的默认 bossCount 参数，即 1。用户可以使用这个选项覆盖 Netty 中的默认 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，则此选项指定 NettyConsumer （如果它应该断开任何回复的回复），则此选项指定 NettyConsumer。	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))	使用 UDP 时，此选项可用于按名称指定网络接口，如 eth0 以加入多播组。		字符串
noReplyLogLevel (consumer (advanced))	如果启用了同步，则指定 NettyConsumer 在日志记录没有回复时要使用的日志级别。 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)捕获了java.nio.channels.ClosedChannelException，则它会使用此日志级别登录。这用于避免记录关闭的通道异常，因为客户端可能会立即断开连接，从而导致 Netty 服务器出现关闭异常。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	DEBUG	LoggingLevel
serverExceptionCaughtLogLevel (consumer (advanced))	如果服务器(NettyConsumer)捕获异常，则它会使用此日志级别记录其日志记录。 Enum 值： • TRACE • DEBUG • INFO • WARN • ERROR • OFF	WARN	LoggingLevel
serverInitializerFactory (consumer (advanced))	使用自定义 ServerInitializerFactory。		ServerInitializerFactory
usingExecutorService (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)	等待套接字连接可用时的时间。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，允许将超时用于 Netty producer。默认情况下不使用超时。该值以秒为单位，因此 30000 为 30 秒。requestTimeout 使用 Netty 的 ReadTimeoutHandler 来触发超时。		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将 request/reply 与 netty producer 搭配使用时如何映射请求和回复消息。只有在您有一个将请求与回复一起映射（例如，请求和回复消息中存在关联 ID ）时，才应使用此项。如果要在 netty 中在同一频道（连接连接）上多路复用。执行此操作时，您必须有一个方法来关联请求和回复消息，以便您可以在继续路由前将正确的回复存储在动态 Camel Exchange 中。在构建自定义关联经理时，我们建议扩展 TimeoutCorrelationManagerSupport。这提供了对其他需要实现的超时和其他复杂性的支持。如需了解更多详细信息，请参阅 producerPoolEnabled 选项。		NettyCamelStateCorrelationManager
lazyChannelCreation (producer (advanced))	如果远程服务器在 Camel 生成者启动时没有启动并运行，则频道可能被创建以避免异常。	true	布尔值
producerPoolEnabled (producer (advanced))	生成者池是否已启用。重要：如果您关闭此，那么也会将单个共享连接用于生成者，即使您正在执行请求/回复。这意味着，如果回复未排序，则响应交集存在潜在的问题。因此，您需要在请求和回复消息中有一个关联 ID，以便您可以将回复正确与负责继续处理 Camel 中的消息的 Camel 回调关联。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))	设置池中闲置实例数量的上限。	100	int

Name	描述	默认值	类型
producerPoolMaxTotal (producer (advanced))	设置池可分配的对象数量的上限（检查到客户端，或闲置等待签出）在给定时间闲置。对没有限制使用负值。	-1	int
producerPoolMinEvictableIdle (producer (advanced))	在被闲置对象驱除前，设置对象可能闲置的最小时间（值为 millis）。	300000	long
producerPoolMinIdle (producer (advanced))	在驱除器线程（如果处于活动状态）前设置制作者池中允许的最小实例数量（如果处于活动状态）。		int
udpConnectionlessSending (producer (advanced))	这个选项支持较少的 udp 发送的连接，这是实际触发和忘记。如果没有侦听接收端口，则连接的 udp 会接收 PortUnreachableException。	false	布尔值
useByteBuf (producer (advanced))	如果 useByteBuf 为 true，netty producer 会在发送前将消息正文转换为 ByteBuf。	false	布尔值
hostnameVerification (security)	在 SSLEngine 上启用/禁用主机名验证。	false	布尔值
allowSerializedHeaders (advanced)	仅在 transferExchange 为 true 时用于 TCP。当设置为 true 时，标头和属性中的可序列化对象将添加到交换中。否则，Camel 将排除任何不可序列化的对象，并将它记录在 WARN 级别。	false	布尔值
channelGroup (advanced)	使用显式 ChannelGroup。		ChannelGroup
nativeTransport (advanced)	是否使用原生传输而不是 NIO。原生传输利用主机操作系统，且仅在某些平台上被支持。您需要为您要使用的主机操作系统添加 netty JAR。详情请查看：	false	布尔值
options (advanced)	允许使用 option. 作为前缀来配置额外的 netty 选项。例如 option.child.keepAlive=false，来设置 netty 选项 child.keepAlive=false。有关可以使用的可能选项，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

Name	描述	默认值	类型
receiveBufferSizePredictor (advanced)	配置缓冲区大小预测器。详情请参阅 Jetty 文档和此邮件线程。		int
sendBufferSize (advanced)	在出站通信期间使用的 TCP/UDP 缓冲区大小。大小为字节。	65536	int
Sync (advanced)	设置是否应严格使用同步处理。	false	布尔值
transferExchange (advanced)	仅用于 TCP。您可以在有线线上传输交换，而不只是正文。以下字段会被传输： 在 body, Out body, fault body, In headers, Out headers, fault headers, Exchange properties, exchange exception。这要求对象可以序列化。Camel 将排除任何不可序列化的对象，并将它记录在 WARN 级别。	false	布尔值
udpByteArrayCodec (advanced)	仅限 UDP。如果启用了使用字节数组 codec 而不是 Java 序列化协议。	false	布尔值
workerCount (advanced)	当 netty 在 nio 模式上工作时，它会使用来自 Netty 的默认 workerCount 参数（即 cpu_core_threads x 2）。用户可以使用这个选项覆盖 Netty 中的默认 workerCount。		int
workerGroup (advanced)	使用显式 EventLoopGroup 作为 boss 线程池。例如，与多个消费者或制作者共享线程池。默认情况下，每个消费者或生成者都有自己的 worker 池，具有 2 个 x cpu 计数内核线程。		EventLoopGroup
allowDefaultCodec (codec)	如果两者都默认 codec，netty 组件会安装一个默认的 codec，编码器/解码器为 null，文本行为 false。将 allowDefaultCodec 设置为 false 可防止 netty 组件安装默认 codec 作为过滤器链中的第一个元素。	true	布尔值
autoAppendDelimiter (codec)	在使用文本 codec 发送时，是否自动附加缺少的最终分隔符。	true	布尔值
decoderMaxLineLength (codec)	文本代码的最大行长度。	1024	int
解码器 (codec)	要使用的解码器列表。您可以使用用逗号分开的值的字符串，并在 Registry 中查找值。只需记住使用 192.168.1.0/24 为值添加前缀，因此 Camel 知道它应该查找。		list

Name	描述	默认值	类型
Delimiter (codec)	用于文本代码的分隔符。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。您可以使用用逗号分开的值的字符串，并在 Registry 中查找值。只需记住使用 192.168.1.0/24 为值添加前缀，因此 Camel 知道它应该查找。		list
编码 (codec)	用于文本代码的编码(charset 名称)。如果没有提供，Camel 将使用 JVM 默认 Charset。		字符串
文本行 (codec)	仅用于 TCP。如果没有指定 codec，您可以使用此标志来指示基于文本的 codec；如果没有指定，则为 false，则通过 TCP 假设对象序列化，但默认只允许对字符串进行序列化。	false	布尔值
enabledProtocols (security)	使用 SSL 时要启用哪个协议。	TLSv1, TLSv1.1, TLSv1.2	字符串
keyStoreFile (security)	用于加密的客户端证书密钥存储。		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）	使用 SSH 加密/解密有效负载的密码设置。		字符串
securityProvider (security)	用于有效负载加密的安全提供程序。如果没有设置，则默认为 SunX509。		字符串
SSL （安全）	设置以指定 SSL 加密是否应用到此端点。	false	布尔值
sslClientCertHeaders (security)	启用并在 SSL 模式中，Netty consumer 将增强 Camel Message，其标头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

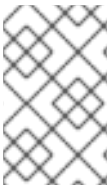
Name	描述	默认值	类型
sslContextParameters (security)	使用 SSLContext 参数配置安全性：		SSLContextParameters
sslHandler (security)	对可用于返回 SSL 处理程序的类的引用。		SslHandler
trustStoreFile (security)	用于加密的服务器端证书密钥存储。		File
trustStoreResource (security)	用于加密的服务器端证书密钥存储。默认情况下从 classpath 加载，但您可以使用 classpath:、file: 或 http: 前缀来加载来自不同系统的资源。		字符串

46.5. 基于 REGISTRY 的选项

codec Handlers 和 SSL Keystores 可以包含在 Registry 中，如 Spring XML 文件中。可以传递的值如下：

Name	描述
passphrase	使用 SSH 加密/解密有效负载的密码设置
keyStoreFormat	用于有效负载加密的密钥存储格式。如果没有设置，则默认为 "JKS"
securityProvider	用于有效负载加密的安全提供程序。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	用于加密的客户端证书密钥存储。默认从 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	自定义 **频道** 处理程序类，可用于执行特殊的出站有效负载。必须覆盖 io.netty.channel.ChannelInboundHandlerAdapter。

Name	描述
encoders	要使用的编码程序列表。您可以使用用逗号分开的值的字符串，并在 Registry 中查找值。只需记住使用 192.168.1.0/24 为值添加前缀，因此 Camel 知道它应该查找。
decoder	一个自定义 **频道处理程序** 类，可用于执行入站有效负载的特殊 marshalling。必须覆盖 io.netty.channel.ChannelOutboundHandlerAdapter。
解码器	要使用的解码器列表。您可以使用用逗号分开的值的字符串，并在 Registry 中查找值。只需记住使用 192.168.1.0/24 为值添加前缀，因此 Camel 知道它应该查找。



注意

请阅读以下有关使用不可共享编码器/解码器的信息。

46.5.1. 使用不可共享的编码器或解码器

如果您的编码器或解码器不可共享（例如，它们没有 @Shareable 类注释），则您的编码器/解码器必须实施 org.apache.camel.component.netty.ChannelHandlerFactory 接口，并以 newChannelHandler 方法返回新实例。这是为了确保可安全使用编码器/解码器。否则，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 工厂类，它有多种常用的方法。

46.6. 从 NETTY 端点向/发送消息

46.6.1. Netty Producer

在 Producer 模式中，组件提供使用 TCP 或 UDP 协议（具有可选 SSL 支持）将有效负载发送到套接字端点的功能。

producer 模式支持单向和基于请求的操作。

46.6.2. Netty Consumer

在 Consumer 模式中，组件提供以下功能：

- 使用 TCP 或 UDP 协议（可选 SSL 支持）侦听指定的套接字
- 使用文本/xml、二进制和基于序列化对象的有效负载接收套接字上的请求
- 在路由上作为消息交换发送它们。

消费者模式支持单向和基于请求的操作。

46.7. 例子

46.7.1. 使用 Request-Reply 和序列化对象有效负载的 UDP Netty 端点

请注意，默认不允许对象序列化，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);
        }
      }
    }
};
```

46.7.2. 使用单向通信基于 TCP 的 Netty 使用者端点

```
RouteBuilder builder = new RouteBuilder() {
  public void configure() {
      from("netty:tcp://0.0.0.0:5150")
```

```
        .to("mock:result");
  }
};
```

46.7.3. 使用 Request-Reply 通信的基于 SSL/TCP 的 Netty 使用者端点

使用 JSSE 配置工具

Netty 组件通过 Camel JSSE 配置实用程序支持 SSL/TLS 配置。这个工具可显著减少您需要编写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 的端点配置

```
...
  <camel:sslContextParameters
     id="sslContextParameters">
    <camel:keyManagers
       keyPassword="keyPassword">
      <camel:keyStore
         resource="/users/home/server/keystore.jks"
         password="keystorePassword"/>
    </camel:keyManagers>
  </camel:sslContextParameters>...
...
```

```
  <to uri="netty:tcp://0.0.0.0:5150?
sync=true&ssl=true&sslContextParameters=#sslContextParameters"/>
...
```

在 Jetty 组件中使用基本 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

如果您需要获取客户端证书的详细信息，您可以访问 javax.net.ssl.SSLSession。当 ssl=true 时，Netty 组件会将 SSLSession 存储为 Camel Message 上的标头，如下所示：

```
SSLSession session = exchange.getIn().getHeader(NettyConstants.NETTY_SSL_SESSION,
SSLSession.class);
// get the first certificate which is client certificate
javax.security.cert.X509Certificate cert = session.getPeerCertificateChain()[0];
Principal principal = cert.getSubjectDN();
```

请记住，设置 needClientAuth=true 以验证客户端，否则 SSLSession 无法访问客户端证书的信息，您可以得到 javax.net.ssl.SSLPeerUnverifiedException: peer not authenticated。如果客户端证书过

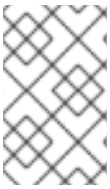
期或无效等，您可能还会获得此异常。

注意

选项 sslClientCertHeaders 可以设置为 true，然后使用具有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。

46.7.4. 使用多个代码

在某些情况下，可能需要将编码器和解码器链添加到 netty 管道。要将 multpile codecs 添加到 camel netty 端点中，应使用 'encoders' 和 'decoders' uri 参数。与 'encoder' 和 'decoder' 参数一样，它们用来提供应添加到管道中的 ChannelUpstreamHandlers 和 ChannelDownstreamHandlers 的引用 (list)。请注意，如果指定了 encodingrs，则编码器参数将被忽略，类似于解码器和解码器参数。



注意

更多有关使用不可共享编码器/解码器的信息。

codecs 列表需要添加到 Camel 的 registry 中，以便在创建端点时解析它们。

```
ChannelHandlerFactory lengthDecoder =
ChannelHandlerFactories.newLengthFieldBasedFrameDecoder(1048576, 0, 4, 0, 4);

StringDecoder stringDecoder = new StringDecoder();
registry.bind("length-decoder", lengthDecoder);
registry.bind("string-decoder", stringDecoder);

LengthFieldPrepender lengthEncoder = new LengthFieldPrepender(4);
StringEncoder stringEncoder = new StringEncoder();
registry.bind("length-encoder", lengthEncoder);
registry.bind("string-encoder", stringEncoder);

List<ChannelHandler> decoders = new ArrayList<ChannelHandler>();
decoders.add(lengthDecoder);
decoders.add(stringDecoder);

List<ChannelHandler> encoders = new ArrayList<ChannelHandler>();
encoders.add(lengthEncoder);
encoders.add(stringEncoder);

registry.bind("encoders", encoders);
registry.bind("decoders", decoders);
```

Spring 的原生集合支持可用于在应用程序上下文中指定 codec 列表

```
<util:list id="decoders" list-class="java.util.LinkedList">
        <bean class="org.apache.camel.component.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，在 netty 端点定义中使用 bean 名称作为逗号分隔的列表，也可以包含在 List 等中。

```
 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");
```

或者通过 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>
```

46.8. 完成后关闭频道

当作为服务器使用时，有时希望关闭频道，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，您也可以按如下方式指示 Camel 按每个消息：
要指示 Camel 关闭频道，您应该添加一个带有键 CamelNettyCloseChannelWhenComplete 的标头，设置为布尔值。
例如，以下示例将在将信息写入客户端后关闭频道：

```
from("netty:tcp://0.0.0.0:8080").process(new Processor() {
    public void process(Exchange exchange) throws Exception {
        String body = exchange.getIn().getBody(String.class);
        exchange.getOut().setBody("Bye " + body);
        // some condition which determines if we should close
        if (close) {

exchange.getOut().setHeader(NettyConstants.NETTY_CLOSE_CHANNEL_WHEN_COMPLETE,
true);
        }
    }
});
```

添加自定义频道管道工厂以完全控制创建的管道。

46.9. 自定义管道

自定义频道管道通过插入自定义处理程序、编码器和解码器，而无需以非常简单的方式在 Netty Endpoint URL 中为用户提供完全控制。

要添加自定义管道，必须通过上下文 registry (Registry 或 camel-spring ApplicationContextRegistry 等)创建自定义频道管道工厂。

自定义管道工厂必须构建如下

- 链接的频道管道工厂必须扩展抽象类 ClientPipelineFactory。
- Consumer linked 频道管道工厂必须扩展抽象类 ServerInitializerFactory。
- 类应该覆盖 initChannel （）方法，以便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、编码器或解码器到管道的管道。

以下示例演示了如何创建 ServerInitializerFactory 工厂

46.9.1. 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));
    }
}
```

然后可将自定义频道管道工厂添加到 registry 中，并以以下方式在 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)
```

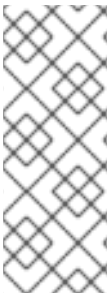
```
        .process(new Processor() {
          public void process(Exchange exchange) throws Exception {
            exchange.getOut().setBody(return_string);
          }
        }
    }
});
```

46.10. 重新使用 NETTY BOS 和 WORKER 线程池

Netty 有两种类型的线程池：boss 和 worker。默认情况下，每个 Netty consumer 和 producer 具有其专用线程池。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，有一个 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，您可以通过在 URI 中配置 workerPool 选项来引用此 worker 池，如下所示：

```
<route>
  <from uri="netty:tcp://0.0.0.0:5021?
textline=true&amp;sync=true&amp;workerPool=#sharedPool&amp;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>
```

因此，

46.11. 通过与请求/回复的单一连接进行多路的并发消息

当通过 netty producer 使用 Netty 进行请求/回复消息传递时，默认情况下，每个消息都通过非共享连接（池）发送。这样可确保回复自动映射到正确的请求线程，以便在 Camel 中进一步路由。在请求/回复消息之间的其他方面相关发生开箱即用，因为回复会回到发送请求的相同连接，并且此连接不会与其他连接共享。当响应返回时，连接会返回连接池，其中可以被其他人重复使用。

但是，如果您想要在单个共享连接上多路并发请求/响应，则需要通过设置 producerPoolEnabled=false 来关闭连接池。现在，如果回复未排序，这意味着有交集响应的潜在问题。因此，您需要在请求和回复消息中有一个关联 ID，以便您可以将回复正确与负责继续处理 Camel 中的消息的 Camel 回调关联。为此，您需要将 NettyCamelStateCorrelationManager 作为关联管理器实施，并通过 correlationManager=114myManager 选项进行配置。

注意

在构建自定义关联经理时，我们建议扩展 TimeoutCorrelationManagerSupport。这提供了对其他需要实现的超时和其他复杂性的支持。

您可以在 camel-example-netty-custom-correlation 目录下的 examples 目录中找到 Apache Camel 源代码的示例。

46.12. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	如果两者都默认 codec，netty 组件会安装一个默认的 codec，编码器/解码器为 null，文本行为 false。将 allowDefaultCodec 设置为 false 可防止 netty 组件安装默认 codec 作为过滤器链中的第一个元素。	true	布尔值
camel.component.netty.allow-serialized-headers	仅在 transferExchange 为 true 时用于 TCP。当设置为 true 时，标头和属性中的可序列化对象将添加到交换中。否则，Camel 将排除任何不可序列化的对象，并将它记录在 WARN 级别。	false	布尔值
camel.component.netty.auto-append-delimiter	在使用文本 codec 发送时，是否自动附加缺少的最终分隔符。	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，回放是根据操作系统的最佳工作。将这个选项设置为 200、500 或 1000 等值，请告知 TCP 堆栈如果未配置此选项，则接受队列的时长，则 backlog 依赖于 OS 设置。		整数
camel.component.netty.boss-count	当 netty 在 nio 模式上工作时，它会使用来自 Netty 的默认 bossCount 参数，即 1。用户可以使用这个选项覆盖 Netty 中的默认 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	设置以选择多播通过 UDP。	false	布尔值

Name	描述	默认值	类型
camel.component.netty.channel-group	使用显式 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
camel.component.netty.client-mode	如果 clientMode 为 true，netty consumer 会将地址连接为 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	等待套接字连接可用时的时间。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将 request/reply 与 netty producer 搭配使用时如何映射请求和回复消息。只有在您有一个将请求与回复一起映射（例如，请求和回复消息中存在关联 ID ）时，才应使用此项。如果要在 netty 中在同一频道（连接连接）上多路复用。执行此操作时，您必须有一个方法来关联请求和回复消息，以便您可以在继续路由前将正确的回复存储在动态 Camel Exchange 中。在构建自定义关联经理时，我们建议扩展 TimeoutCorrelationManagerSupport。这提供了对其他需要实现的超时和其他复杂性的支持。如需了解更多详细信息，请参阅 producerPoolEnabled 选项。选项是 org.apache.camel.component.netty.NettyCamelStateCorrelationManager 类型。		NettyCamelStateCorrelationManager
camel.component.netty.decoder-max-line-length	文本代码的最大行长度。	1024	整数
camel.component.netty.decoders	要使用的解码器列表。您可以使用用逗号分开的值的字符串，并在 Registry 中查找值。只需记住使用 192.168.1.0/24 为值添加前缀，因此 Camel 知道它应该查找。		字符串
camel.component.netty.delimiter	用于文本代码的分隔符。可能的值有 LINE 和 NULL。		TextLineDelimiter

Name	描述	默认值	类型
camel.component.netty.disconnect	是否在使用后是否断开(close)与 Netty Channel 的连接。可用于消费者和生产者。	false	布尔值
camel.component.netty.disconnect-on-no-reply	如果启用了同步，则此选项指定 NettyConsumer （如果它应该断开任何回复的回复），则此选项指定 NettyConsumer。	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
camel.component.netty.enabled-protocols	使用 SSL 时要启用哪个协议。	TLSv1, TLSv1.1, TLSv1.2	字符串
camel.component.netty.encoders	要使用的编码程序列表。您可以使用用逗号分开的值的字符串，并在 Registry 中查找值。只需记住使用 192.168.1.0/24 为值添加前缀，因此 Camel 知道它应该查找。		字符串
camel.component.netty.encoding	用于文本代码的编码(charset 名称)。如果没有提供，Camel 将使用 JVM 默认 Charset。		字符串
camel.component.netty.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	在 SSLEngine 上启用/禁用主机名验证。	false	布尔值
camel.component.netty.keep-alive	设置以确保不活动而关闭套接字。	true	布尔值
camel.component.netty.key-store-file	用于加密的客户端证书密钥存储。		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

Name	描述	默认值	类型
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，则频道可能被创建以避免异常。	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
camel.component.netty.maximum-pool-size	为 netty consumer 排序的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 将使用 10 个线程，除非 2 x cpu_core 加上 1 的值更高，然后会覆盖和使用它。例如，如果存在 8 个内核，则消费者线程池将为 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对消息排序，并在某些消息被阻止时，则 nettys worker 线程(event loop)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。详情请查看：	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	使用 UDP 时，此选项可用于按名称指定网络接口，如 eth0 以加入多播组。		字符串
camel.component.netty.no-reply-log-level	如果启用了同步，则指定 NettyConsumer 在日志记录没有回复时要使用的日志级别。		LoggingLevel
camel.component.netty.options	允许使用 option. 作为前缀来配置额外的 netty 选项。例如 option.child.keepAlive=false，来设置 netty 选项 child.keepAlive=false。有关可以使用的可能选项，请参阅 Netty 文档。		Map
camel.component.netty.passphrase	使用 SSH 加密/解密有效负载的密码设置。		字符串

Name	描述	默认值	类型
camel.component.netty.producer-pool-enabled	生成者池是否已启用。重要：如果您关闭此，那么也会将单个共享连接用于生成者，即使您正在执行请求/回复。这意味着，如果回复未排序，则响应交集存在潜在的问题。因此，您需要在请求和回复消息中有一个关联 ID，以便您可以将回复正确与负责继续处理 Camel 中的消息的 Camel 回调关联。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.producer-pool-max-idle	设置池中闲置实例数量的上限。	100	整数
camel.component.netty.producer-pool-max-total	设置池可分配的对象数量的上限（检查到客户端，或闲置等待签出）在给定时间闲置。对没有限制使用负值。	-1	整数
camel.component.netty.producer-pool-min-evictable-idle	在被闲置对象驱除前，设置对象可能闲置的最小时间（值为 millis）。	300000	Long
camel.component.netty.producer-pool-min-idle	在驱除器线程（如果处于活动状态）前设置制作者池中允许的最小实例数量（如果处于活动状态）。		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。详情请参阅 Jetty 文档和此邮件线程。		整数
camel.component.netty.reconnect	只在消费者中的 clientMode 中使用，消费者将尝试断开连接时重新连接。	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（以毫秒为单位）。	10000	整数
camel.component.netty.request-timeout	在调用远程服务器时，允许将超时用于 Netty producer。默认情况下不使用超时。该值以秒为单位，因此 30000 为 30 秒。requestTimeout 使用 Netty 的 ReadTimeoutHandler 来触发超时。		Long

Name	描述	默认值	类型
camel.component.netty.reuse-address	设置以方便套接字多路。	true	布尔值
camel.component.netty.reuse-channel	此选项允许生产者和消费者（客户端模式）在处理交换的生命周期内重复使用相同的 Netty Channel。如果您需要在 Camel 路由中多次调用服务器并希望使用相同的网络连接，这非常有用。使用此选项时，频道不会返回到连接池，直到交换完成为止；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交换属性上，其键为 NettyConstants，Y_CHANNEL 允许您在路由期间获取频道并使用它。	false	布尔值
camel.component.netty.security-provider	用于有效负载加密的安全提供程序。如果没有设置，则默认为 SunX509。		字符串
camel.component.netty.send-buffer-size	在出站通信期间使用的 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)捕获了 java.nio.channels.ClosedChannelException，则它会使用此日志级别登录。这用于避免记录关闭的通道异常，因为客户端可能会立即断开连接，从而导致 Netty 服务器出现关闭异常。		LoggingLevel
camel.component.netty.server-exception-caught-log-level	如果服务器(NettyConsumer)捕获异常，则它会使用此日志级别记录其日志记录。		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	启用并在 SSL 模式中，Netty consumer 将增强 Camel Message，其标头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 SSLContext 参数配置安全性：选项是 org.apache.camel.support.jsse.SSLContextParameters 类型。		SSLContextParameters

Name	描述	默认值	类型
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
camel.component.netty.sync	设置为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的 codec；如果没有指定，则为 false，则通过 TCP 假设对象序列化，但默认只允许对字符串进行序列化。	false	布尔值
camel.component.netty.transfer-exchange	仅用于 TCP。您可以在有线线上传输交换，而不只是正文。以下字段会被传输： 在 body, Out body, fault body, In headers, Out headers, fault headers, Exchange properties, exchange exception。这要求对象可以序列化。Camel 将排除任何不可序列化的对象，并将它记录在 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	仅限 UDP。如果启用了使用字节数组 codec 而不是 Java 序列化协议。	false	布尔值
camel.component.netty.udp-connectionless-sending	这个选项支持较少的 udp 发送的连接，这是实际触发和忘记。如果没有侦听接收端口，则连接的 udp 会接收 PortUnreachableException。	false	布尔值
camel.component.netty.use-byte-buf	如果 useByteBuf 为 true，netty producer 会在发送前将消息正文转换为 ByteBuf。	false	布尔值
camel.component.netty.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.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.worker-count	当 netty 在 nio 模式上工作时，它会使用来自 Netty 的默认 workerCount 参数（即 cpu_core_threads x 2）。用户可以使用这个选项覆盖 Netty 中的默认 workerCount。		整数
camel.component.netty.worker-group	使用显式 EventLoopGroup 作为 boss 线程池。例如，与多个消费者或制作者共享线程池。默认情况下，每个消费者或生成者都有自己的 worker 池，具有 2 个 x cpu 计数内核线程。选项是一个 io.netty.channel.EventLoopGroup 类型。		EventLoopGroup

第 47 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 mq 库之一，因此，如果您想要将其与 Java 项目集成 - Camel Paho connector 是一种好方法。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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

47.1. URI 格式

```
paho:topic[?options]
```

其中 topic 是主题的名称。

47.2. 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

47.2.1. 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。

47.2.2. 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。

47.3. 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)	如果连接丢失，设置客户端是否自动尝试重新连接到服务器。如果设置为 false，客户端不会在连接丢失时尝试自动连接到服务器。如果设置为 true，如果连接丢失，客户端会尝试重新连接服务器。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)	mq 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是否应该在重启和重新连接后记住状态。如果设置为 false，客户端和服务器在重启客户端后维护状态，则服务器和客户端的连接。维护状态 ：消息交付将可靠满足指定的 QOS，即使客户端、服务器或连接重启也是如此。服务器会将订阅视为 durable。如果设置为 true，则客户端和服务器在重启客户端后不维护状态，服务器或连接将不会被维护。这意味着，如果客户端、服务器或连接重启，则无法维护发送到指定的 QOS 的消息。服务器会将订阅视为不可durable。	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要等待与 MQTT 服务器的网络连接的最大时间间隔。默认超时为 30 秒。0 代表禁用超时处理意味着客户端将等待网络连接成功或失败。	30	int
filePersistenceDirectory (common)	文件持久性使用的基本目录。默认将使用 user 目录。		字符串
keepAliveInterval (common)	设置保留间隔。这个值（以秒为单位）定义发送或接收消息之间的最大时间间隔。它可让客户端检测服务器是否不再可用，而无需等待 TCP/IP 超时。客户端将确保每个情况下至少有一个消息在网络间传输。如果在时间段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，服务器将确认。0 代表禁用客户端中的 keepalive 处理。默认值为 60 秒。	60	int
maxInflight (common)	设置最大 inflight。请在高流量环境中增加这个值。默认值为 10。	10	int
maxReconnectDelay (common)	获取在重新连接之间等待的最大时间（以 millis 为单位）。	128000	int
mqttVersion (common)	设置 mq 版本。默认操作是与 3.1.1 版本连接，如果失败，则回退到 3.1。可以分别使用 mq_VERSION_3_1_1 或 MQTT_VERSION_3_1 选项，具体选择 3.1.1 或 3.1 版本。		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	Paho\":\"
QoS (common)	客户端服务质量(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。多个服务器可以用逗号分开。每个 serverURI 指定客户端可以连接的服务器的地址。两种类型的连接是 TCP 连接的 tcp://，对于 SSL/TLS 保护的 TCP 连接支持 ssl://。例如： tcp://localhost:1883 ssl://localhost:8883 如果未指定端口，对于 tcp:// URI，它将默认为 1883，对于 ssl:// URI，将默认为 1883。如果设置了 serverURIs，它会覆盖在 MQTT 客户端构造器上传递的 serverURI 参数。当尝试连接启动时，客户端将从列表中的第一个 serverURI 开始，并通过列表进行工作，直到与服务器建立连接。如果无法向任何服务器进行连接，则连接尝试会失败。指定客户端可以连接到的服务器列表具有多个用途：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，其中两个或多个 MQTT 服务器共享状态。MQTT 客户端可以连接到任何相等的服务器，并保证无论客户端连接到哪个服务器，消息都可以可靠交付且可暂停订阅。如果需要具有危险的订阅和/或可靠的消息发送，则 cleansession 标志必须设置为 false。可能需要指定一组不相等的服务器（如在高可用性选项中）。因为在服务器间没有状态共享可靠的消息交付，因此没有可用的订阅无效。如果使用 hunt 列表模式，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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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)。是否应保留消息。		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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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)。是否应保留消息。		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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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)。是否应保留消息。	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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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)。是否应保留消息。		字符串
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
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)	使用共享的 Paho 客户端。		MqttClient
customWebSocketHeaders (advanced)	为 WebSocket 连接设置自定义 WebSocket 标头。		Properties
executorServiceTimeout (advanced)	在强制终止前，设置 executor 服务在终止时应等待的时间（以秒为单位）。除非需要修改这个值，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
密码 （安全）	用于针对 MQTT 代理进行身份验证的密码。		字符串

Name	描述	默认值	类型
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 提供特定于应用程序的安全设置。		SocketFactory
sslClientProps (security)	设置连接的 SSL 属性。请注意，只有在有 Java 安全套接字扩展(JSSE)的实现时才有效这些属性。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 of: SSL, SSLv3, TLS, TLSv1, SSL_TLS. com.ibm.ssl.contextProvider Underlying JSSE 供应商。例如，IBMJSSE2 或 SunJSSE com.ibm.ssl.keyStore 包含您希望 KeyManager 使用的 KeyStore 对象的名称。例如 /mydir/etc/key.p12 com.ibm.ssl.keyStorePassword 您希望 KeyManager 使用的 KeyStore 对象的名称。密码可以采用纯文本，也可以使用静态方法模糊处理： com.ibm.micro.security.Password.obfuscate (char password)。这使用简单和不安全的 XOR 和 Base64 编码机制混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。 com.ibm.ssl.keyStoreType Type of key 存储，如 PKCS12、JKS 或 JCEKS. com.ibm.ssl.keyStoreProvider 存储提供程序，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.trustStore 包含您希望 TrustManager 使用的 KeyStore 对象的名称。 com.ibm.ssl.trustStorePassword 要使用的 TrustStore 对象的名称。密码可以采用纯文本，也可以使用静态方法模糊处理： com.ibm.micro.security.Password.obfuscate (char password)。这使用简单和不安全的 XOR 和 Base64 编码机制混淆密码。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 对象的类型。值与 keyStoreType. com.ibm.ssl.trustStoreProvider Trust 存储供应商相同，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.enabledCipherSuites A 列表。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 设置算法，该算法将用于实例化平台中提供的 KeyManagerFactory 对象，而不使用平台中提供的默认算法。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，它会在连接的握手后使用，您应该在验证主机名错误时自己执行操作。没有默认的 HostnameVerifier。		HostnameVerifier
userName (security)	用于针对 MQTT 代理进行身份验证的用户名。		字符串

47.4. 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：

47.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
topic (common)	主题 **所需的** 名称。		字符串

47.4.2. 查询参数(31 参数)

Name	描述	默认值	类型
automaticReconnect (common)	如果连接丢失，设置客户端是否自动尝试重新连接到服务器。如果设置为 false，客户端不会在连接丢失时尝试自动连接到服务器。如果设置为 true，如果连接丢失，客户端会尝试重新连接服务器。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)	mq 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是否应该在重启和重新连接后记住状态。如果设置为 false，客户端和服务器在重启客户端后维护状态，则服务器和客户端的连接。维护状态 ：消息交付将可靠满足指定的 QOS，即使客户端、服务器或连接重启也是如此。服务器会将订阅视为 durable。如果设置为 true，则客户端和服务器在重启客户端后不维护状态，服务器或连接将不会被维护。这意味着，如果客户端、服务器或连接重启，则无法维护发送到指定的 QOS 的消息。服务器会将订阅视为不可durable。	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要等待与 MQTT 服务器的网络连接的最大时间间隔。默认超时为 30 秒。0 代表禁用超时处理意味着客户端将等待网络连接成功或失败。	30	int
filePersistenceDirectory (common)	文件持久性使用的基本目录。默认将使用 user 目录。		字符串
keepAliveInterval (common)	设置保留间隔。这个值（以秒为单位）定义发送或接收消息之间的最大时间间隔。它可让客户端检测服务器是否不再可用，而无需等待 TCP/IP 超时。客户端将确保每个情况下至少有一个消息在网络间传输。如果在时间段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，服务器将确认。0 代表禁用客户端中的 keepalive 处理。默认值为 60 秒。	60	int
maxInflight (common)	设置最大 inflight。请在高流量环境中增加这个值。默认值为 10。	10	int
maxReconnectDelay (common)	获取在重新连接之间等待的最大时间（以 millis 为单位）。	128000	int
mqttVersion (common)	设置 mq 版本。默认操作是与 3.1.1 版本连接，如果失败，则回退到 3.1。可以分别使用 mq_VERSION_3_1_1 或 MQTT_VERSION_3_1 选项，具体选择 3.1.1 或 3.1 版本。		int
Persistence (common)	要使用的客户端持久性 - 内存或文件。 Enum 值： • FILE • MEMORY	MEMORY	Paho\":\"
QoS (common)	客户端服务质量(0-2)。	2	int

Name	描述	默认值	类型
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。多个服务器可以用逗号分开。每个 serverURI 指定客户端可以连接的服务器的地址。两种类型的连接是 TCP 连接的 tcp://，对于 SSL/TLS 保护的 TCP 连接支持 ssl://。例如： tcp://localhost:1883 ssl://localhost:8883 如果未指定端口，对于 tcp:// URI，它将默认为 1883，对于 ssl:// URI，将默认为 1883。如果设置了 serverURIs，它会覆盖在 MQTT 客户端构造器上传递的 serverURI 参数。当尝试连接启动时，客户端将从列表中的第一个 serverURI 开始，并通过列表进行工作，直到与服务器建立连接。如果无法向任何服务器进行连接，则连接尝试会失败。指定客户端可以连接到的服务器列表具有多个用途：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，其中两个或多个 MQTT 服务器共享状态。MQTT 客户端可以连接到任何相等的服务器，并保证无论客户端连接到哪个服务器，消息都可以可靠交付且可暂停订阅。如果需要具有危险的订阅和/或可靠的消息发送，则 cleansession 标志必须设置为 false。可能需要指定一组不相等的服务器（如在高可用性选项中）。因为在服务器间没有状态共享可靠的消息交付，因此没有可用的订阅无效。如果使用 hunt 列表模式，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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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)。是否应保留消息。		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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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)。是否应保留消息。		int
willRetained (common)	为连接设置 Last Will and Testament (LWT)。如果此客户端意外丢失了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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)。是否应保留消息。	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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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)。是否应保留消息。		字符串

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
exceptionHandler (consumer (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)	在强制终止前，设置 executor 服务在终止时应等待的时间（以秒为单位）。除非需要修改这个值，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
密码 （安全）	用于针对 MQTT 代理进行身份验证的密码。		字符串

Name	描述	默认值	类型
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 提供特定于应用程序的安全设置。		SocketFactory
sslClientProps (security)	设置连接的 SSL 属性。请注意，只有在有 Java 安全套接字扩展(JSSE)的实现时才有效这些属性。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 of: SSL, SSLv3, TLS, TLSv1, SSL_TLS. com.ibm.ssl.contextProvider Underlying JSSE 供应商。例如，IBMJSSE2 或 SunJSSE com.ibm.ssl.keyStore 包含您希望 KeyManager 使用的 KeyStore 对象的名称。例如 /mydir/etc/key.p12 com.ibm.ssl.keyStorePassword 您希望 KeyManager 使用的 KeyStore 对象的名称。密码可以采用纯文本，也可以使用静态方法模糊处理： com.ibm.micro.security.Password.obfuscate (char password)。这使用简单和不安全的 XOR 和 Base64 编码机制混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。 com.ibm.ssl.keyStoreType Type of key 存储，如 PKCS12、JKS 或 JCEKS. com.ibm.ssl.keyStoreProvider 存储提供程序，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.trustStore 包含您希望 TrustManager 使用的 KeyStore 对象的名称。 com.ibm.ssl.trustStorePassword 要使用的 TrustStore 对象的名称。密码可以采用纯文本，也可以使用静态方法模糊处理： com.ibm.micro.security.Password.obfuscate (char password)。这使用简单和不安全的 XOR 和 Base64 编码机制混淆密码。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 对象的类型。值与 keyStoreType. com.ibm.ssl.trustStoreProvider Trust 存储供应商相同，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.enabledCipherSuites A 列表。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 设置算法，该算法将用于实例化平台中提供的 KeyManagerFactory 对象，而不使用平台中提供的默认算法。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，它会在连接的握手后使用，您应该在验证主机名错误时自己执行操作。没有默认的 HostnameVerifier。		HostnameVerifier

Name	描述	默认值	类型
userName (security)	用于针对 MQTT 代理进行身份验证的用户名。		字符串

47.5. 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	要覆盖并发送到的主题名称，而不是在端点上指定的主题

47.6. 默认有效负载类型

默认情况下，Camel Paho 组件在从（或放入）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 构建 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 转换为字符串 （相反）：

```
// Receive payload
String payload = consumerTemplate.receiveBody("paho:topic", String.class);

// Send payload
String payload = "message";
producerTemplate.sendBody("paho:topic", payload);
```

47.7. SAMPLES

例如，以下片段从与 Camel 路由器相同的主机上安装的 mq 代理读取信息：

```
from("paho:some/queue")
    .to("mock:test");
```

虽然以下片段会向 MQTT 代理发送信息：

```
from("direct:test")
    .to("paho:some/target/queue");
```

例如，这是如何从远程 mq 代理读取信息：

```
from("paho:some/queue?brokerUrl=tcp://iot.eclipse.org:1883")
    .to("mock:test");
```

在这里，我们将覆盖默认主题，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");
```

47.8. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	如果连接丢失，设置客户端是否自动尝试重新连接到服务器。如果设置为 false，客户端不会在连接丢失时尝试自动连接到服务器。如果设置为 true，如果连接丢失，客户端会尝试重新连接服务器。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.autowired-enabled	是否启用自动关闭。这用于自动关闭选项（选项必须标记为 autowired），方法是在 registry 中查找查找是否有单个匹配类型实例，然后在组件上配置。这可以用于自动配置 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	mq 代理的 URL。	tcp://localhost:1883	字符串
camel.component.paho.clean-session	设置客户端和服务器是否应该在重启和重新连接后记住状态。如果设置为 false，客户端和服务器在重启客户端后维护状态，则服务器和客户端的连接。维护状态 ：消息交付将可靠满足指定的 QOS，即使客户端、服务器或连接重启也是如此。服务器会将订阅视为 durable。如果设置为 true，则客户端和服务器在重启客户端后不维护状态，服务器或连接将不会被维护。这意味着，如果客户端、服务器或连接重启，则无法维护发送到指定的 QOS 的消息。服务器会将订阅视为不可durable。	true	布尔值
camel.component.paho.client	使用共享的 Paho 客户端。选项是 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

Name	描述	默认值	类型
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要等待与 MQTT 服务器的网络连接的最大时间间隔。默认超时为 30 秒。0 代表禁用超时处理意味着客户端将等待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 java.util.Properties 类型。		Properties
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	在强制终止前，设置 executor 服务在终止时应等待的时间（以秒为单位）。除非需要修改这个值，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本目录。默认将使用 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	设置保留间隔。这个值（以秒为单位）定义发送或接收消息之间的最大时间间隔。它可让客户端检测服务器是否不再可用，而无需等待 TCP/IP 超时。客户端将确保每个情况下至少有一个消息在网络间传输。如果在时间段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，服务器将确认。0 代表禁用客户端中的 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	设置最大 inflight。请在高流量环境中增加这个值。默认值为 10。	10	整数

Name	描述	默认值	类型
camel.component.paho.max-reconnect-delay	获取在重新连接之间等待的最大时间（以 millis 为单位）。	128000	整数
camel.component.paho.mqtt-version	设置 mq 版本。默认操作是与 3.1.1 版本连接，如果失败，则回退到 3.1。可以分别使用 mq_VERSION_3_1_1 或 MQTT_VERSION_3_1 选项，具体选择 3.1.1 或 3.1 版本。		整数
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。		Paho\":\"
camel.component.paho.qos	客户端服务质量(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。多个服务器可以用逗号分开。每个 serverURI 指定客户端可以连接的服务器的地址。两种类型的连接是 TCP 连接的 tcp://，对于 SSL/TLS 保护的 TCP 连接支持 ssl://。例如： tcp://localhost:1883 ssl://localhost:8883 如果未指定端口，对于 tcp:// URI，它将默认为 1883，对于 ssl:// URI，将默认为 1883。如果设置了 serverURIs，它会覆盖在 MQTT 客户端构造器上传递的 serverURI 参数。当尝试连接启动时，客户端将从列表中的第一个 serverURI 开始，并通过列表进行工作，直到与服务器建立连接。如果无法向任何服务器进行连接，则连接尝试会失败。指定客户端可以连接到的服务器列表具有多个用途：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，其中两个或多个 MQTT 服务器共享状态。MQTT 客户端可以连接到任何相等的服务器，并保证无论客户端连接到哪个服务器，消息都可以可靠交付且可暂停订阅。如果需要具有危险的订阅和/或可靠的消息发送，则 cleansession 标志必须设置为 false。可能需要指定一组不相等的服务器（如在高可用性选项中）。因为在服务器间没有状态共享可靠的消息交付，因此没有可用的订阅无效。如果使用 hunt 列表模式，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 提供特定于应用程序的安全设置。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，只有在有 Java 安全套接字扩展(JSSE)的实现时才有效这些属性。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 of: SSL, SSLv3, TLS, TLSv1, SSL_TLS. com.ibm.ssl.contextProvider Underlying JSSE 供应商。例如，IBMJSSE2 或 SunJSSE com.ibm.ssl.keyStore 包含您希望 KeyManager 使用的 KeyStore 对象的名称。例如 /mydir/etc/key.p12 com.ibm.ssl.keyStorePassword 您希望 KeyManager 使用的 KeyStore 对象的名称。密码可以采用纯文本，也可以使用静态方法模糊处理： com.ibm.micro.security.Password.obfuscate (char password)。这使用简单和不安全的 XOR 和 Base64 编码机制混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。 com.ibm.ssl.keyStoreType Type of key 存储，如 PKCS12、JKS 或 JCEKS. com.ibm.ssl.keyStoreProvider 存储提供程序，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.trustStore 包含您希望 TrustManager 使用的 KeyStore 对象的名称。 com.ibm.ssl.trustStorePassword 要使用的 TrustStore 对象的名称。密码可以采用纯文本，也可以使用静态方法模糊处理： com.ibm.micro.security.Password.obfuscate (char password)。这使用简单和不安全的 XOR 和 Base64 编码机制混淆密码。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 对象的类型。值与 keyStoreType. com.ibm.ssl.trustStoreProvider Trust 存储供应商相同，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.enabledCipherSuites A 列表。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 设置算法，该算法将用于实例化平台中提供的 KeyManagerFactory 对象，而不使用平台中提供的默认算法。示例值： IbmX509 或 IBMJ9X509. com.ibm.ssl.trustManager 设置用于实例化 TrustManagerFactory 对象的算法，而不使用平台中提供的默认算法。示例值：PKIX 或 IBMJ9X509。选项是 java.util.Properties 类型。		Properties
camel.component.paho.ssl-hostname-verifier	为 SSL 连接设置 HostnameVerifier。请注意，它会在连接的握手后使用，您应该在验证主机名错误时自己执行操作。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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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)。是否应保留消息。		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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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)。是否应保留消息。		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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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)。是否应保留消息。	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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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)。是否应保留消息。		字符串

第 48 章 PAHO MQTT 5

支持生成者和消费者

paho mq5 组件使用带有 MQTT v5 的 Eclipse Paho 库为 mq 消息传递协议提供连接器。paho 是最流行的 mq 库之一，因此，如果您想要将其与 Java 项目集成 - Camel Paho connector 是一种好方法。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-paho-mqtt5</artifactId>
  <version>{CamelSBVersion}</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

48.1. URI 格式

```
paho-mqtt5:topic[?options]
```

其中 topic 是主题的名称。

48.2. 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

48.2.1. 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。

48.2.2. 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。

48.3. 组件选项

Paho mq 5 组件支持 32 个选项，如下所列。

Name	描述	默认值	类型
automaticReconnect (common)	如果连接丢失，设置客户端是否自动尝试重新连接到服务器。如果设置为 false，客户端不会在连接丢失时尝试自动连接到服务器。如果设置为 true，如果连接丢失，客户端会尝试重新连接服务器。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)	mq 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是否应该在重启和重新连接后记住状态。如果设置为 false，客户端和服务器在重启客户端后维护状态，则服务器和客户端的连接。维护状态 ：消息交付将可靠满足指定的 QOS，即使客户端、服务器或连接重启也是如此。服务器会将订阅视为 durable。如果设置为 true，则客户端和服务器在重启客户端后不维护状态，服务器或连接将不会被维护。这意味着，如果客户端、服务器或连接重启，则无法维护发送到指定的 QOS 的消息。服务器会将订阅视为不可durable。	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要等待与 MQTT 服务器的网络连接的最大时间间隔。默认超时为 30 秒。0 代表禁用超时处理意味着客户端将等待网络连接成功或失败。	30	int
filePersistenceDirectory (common)	文件持久性使用的基本目录。默认将使用 user 目录。		字符串
keepAliveInterval (common)	设置保留间隔。这个值（以秒为单位）定义发送或接收消息之间的最大时间间隔。它可让客户端检测服务器是否不再可用，而无需等待 TCP/IP 超时。客户端将确保每个情况下至少有一个消息在网络间传输。如果在时间段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，服务器将确认。0 代表禁用客户端中的 keepalive 处理。默认值为 60 秒。	60	int
maxReconnectDelay (common)	获取在重新连接之间等待的最大时间（以 millis 为单位）。	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(0-2)。	2	int
receiveMaximum (common)	设置接收的最大值。这个值代表了客户端同时处理的 QoS 1 和 QoS 2 出版物的限制。没有机制来限制服务器可能会尝试发送的 QoS 0 出版物数量。默认值为 65535。	65535	int

Name	描述	默认值	类型
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。多个服务器可以用逗号分开。每个 serverURI 指定客户端可以连接的服务器的地址。两种类型的连接是 TCP 连接的 tcp://，对于 SSL/TLS 保护的 TCP 连接支持 ssl://。例如： tcp://localhost:1883 ssl://localhost:8883 如果未指定端口，对于 tcp:// URI，它将默认为 1883，对于 ssl:// URI，将默认为 1883。如果设置了 serverURIs，它会覆盖在 MQTT 客户端构造器上传递的 serverURI 参数。当尝试连接启动时，客户端将从列表中的第一个 serverURI 开始，并通过列表进行工作，直到与服务器建立连接。如果无法向任何服务器进行连接，则连接尝试会失败。指定客户端可以连接到的服务器列表具有多个用途：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，其中两个或多个 MQTT 服务器共享状态。 MQTT 客户端可以连接到任何相等的服务器，并保证无论客户端连接到哪个服务器，消息都可以可靠交付且可暂停订阅。如果需要具有危险的订阅和/或可靠的消息发送，则 cleansession 标志必须设置为 false。可能需要指定一组不相等的服务器（如在高可用性选项中）。因为在服务器间没有状态共享可靠的消息交付，因此没有可用的订阅无效。如果使用 hunt 列表模式，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将保持会话的最长时间。如果客户端打算稍后连接到服务器，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 0，则会话将在网络连接关闭后立即结束。当客户端确定它不再可用于会话时，它应该断开 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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。为消息设置的 mq 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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。是否应保留消息。	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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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)	使用共享的 Paho 客户端。		MqttClient
customWebSocketHeaders (advanced)	为 WebSocket 连接设置自定义 WebSocket 标头。		Map
executorServiceTimeout (advanced)	在强制终止前，设置 executor 服务在终止时应等待的时间（以秒为单位）。除非需要修改这个值，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
密码 （安全）	用于针对 MQTT 代理进行身份验证的密码。		字符串

Name	描述	默认值	类型
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 提供特定于应用程序的安全设置。		SocketFactory
sslClientProps (security)	设置连接的 SSL 属性。请注意，只有在有 Java 安全套接字扩展(JSSE)的实现时才有效这些属性。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 of: SSL, SSLv3, TLS, TLSv1, SSL_TLS. com.ibm.ssl.contextProvider Underlying JSSE 供应商。例如，IBMJSSE2 或 SunJSSE com.ibm.ssl.keyStore 包含您希望 KeyManager 使用的 KeyStore 对象的名称。例如 /mydir/etc/key.p12 com.ibm.ssl.keyStorePassword 您希望 KeyManager 使用的 KeyStore 对象的名称。密码可以采用纯文本，也可以使用静态方法模糊处理： com.ibm.micro.security.Password.obfuscate (char password)。这使用简单和不安全的 XOR 和 Base64 编码机制混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。 com.ibm.ssl.keyStoreType Type of key 存储，如 PKCS12、JKS 或 JCEKS. com.ibm.ssl.keyStoreProvider 存储提供程序，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.trustStore 包含您希望 TrustManager 使用的 KeyStore 对象的名称。 com.ibm.ssl.trustStorePassword 要使用的 TrustStore 对象的名称。密码可以采用纯文本，也可以使用静态方法模糊处理： com.ibm.micro.security.Password.obfuscate (char password)。这使用简单和不安全的 XOR 和 Base64 编码机制混淆密码。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 对象的类型。值与 keyStoreType. com.ibm.ssl.trustStoreProvider Trust 存储供应商相同，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.enabledCipherSuites A 列表。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 设置算法，该算法将用于实例化平台中提供的 KeyManagerFactory 对象，而不使用平台中提供的默认算法。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，它会在连接的握手后使用，您应该在验证主机名错误时自己执行操作。没有默认的 HostnameVerifier。		HostnameVerifier

Name	描述	默认值	类型
userName (security)	用于针对 MQTT 代理进行身份验证的用户名。		字符串

48.4. 端点选项

Paho mq 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：

48.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
topic (common)	主题 **所需的** 名称。		字符串

48.4.2. 查询参数(32 参数)

Name	描述	默认值	类型
automaticReconnect (common)	如果连接丢失，设置客户端是否自动尝试重新连接到服务器。如果设置为 false，客户端不会在连接丢失时尝试自动连接到服务器。如果设置为 true，如果连接丢失，客户端会尝试重新连接服务器。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)	mq 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是否应该在重启和重新连接后记住状态。如果设置为 false，客户端和服务器在重启客户端后维护状态，则服务器和客户端的连接。维护状态 ：消息交付将可靠满足指定的 QOS，即使客户端、服务器或连接重启也是如此。服务器会将订阅视为 durable。如果设置为 true，则客户端和服务器在重启客户端后不维护状态，服务器或连接将不会被维护。这意味着，如果客户端、服务器或连接重启，则无法维护发送到指定的 QOS 的消息。服务器会将订阅视为不可durable。	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要等待与 MQTT 服务器的网络连接的最大时间间隔。默认超时为 30 秒。0 代表禁用超时处理意味着客户端将等待网络连接成功或失败。	30	int
filePersistenceDirectory (common)	文件持久性使用的基本目录。默认将使用 user 目录。		字符串
keepAliveInterval (common)	设置保留间隔。这个值（以秒为单位）定义发送或接收消息之间的最大时间间隔。它可让客户端检测服务器是否不再可用，而无需等待 TCP/IP 超时。客户端将确保每个情况下至少有一个消息在网络间传输。如果在时间段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，服务器将确认。0 代表禁用客户端中的 keepalive 处理。默认值为 60 秒。	60	int
maxReconnectDelay (common)	获取在重新连接之间等待的最大时间（以 millis 为单位）。	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(0-2)。	2	int
receiveMaximum (common)	设置接收的最大值。这个值代表了客户端同时处理的 QoS 1 和 QoS 2 出版物的限制。没有机制来限制服务器可能会尝试发送的 QoS 0 出版物数量。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。多个服务器可以用逗号分开。每个 serverURI 指定客户端可以连接的服务器的地址。两种类型的连接是 TCP 连接的 tcp://，对于 SSL/TLS 保护的 TCP 连接支持 ssl://。例如： tcp://localhost:1883 ssl://localhost:8883 如果未指定端口，对于 tcp:// URI，它将默认为 1883，对于 ssl:// URI，将默认为 1883。如果设置了 serverURIs，它会覆盖在 MQTT 客户端构造器上传递的 serverURI 参数。当尝试连接启动时，客户端将从列表中的第一个 serverURI 开始，并通过列表进行工作，直到与服务器建立连接。如果无法向任何服务器进行连接，则连接尝试会失败。指定客户端可以连接到的服务器列表具有多个用途：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，其中两个或多个 MQTT 服务器共享状态。MQTT 客户端可以连接到任何相等的服务器，并保证无论客户端连接到哪个服务器，消息都可以可靠交付且可暂停订阅。如果需要具有危险的订阅和/或可靠的消息发送，则 cleansession 标志必须设置为 false。可能需要指定一组不相等的服务器（如在高可用性选项中）。因为在服务器间没有状态共享可靠的消息交付，因此没有可用的订阅无效。如果使用 hunt 列表模式，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将保持会话的最长时间。如果客户端打算稍后连接到服务器，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 0，则会话将在网络连接关闭后立即结束。当客户端确定它不再可用于会话时，它应该断开 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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。为消息设置的 mq 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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。是否应保留消息。	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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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

Name	描述	默认值	类型
executorServiceTimeout (advanced)	在强制终止前，设置 executor 服务在终止时应等待的时间（以秒为单位）。除非需要修改这个值，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
密码 （安全）	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 提供特定于应用程序的安全设置。		SocketFactory

Name	描述	默认值	类型
sslClientProps (security)	设置连接的 SSL 属性。请注意，只有在有 Java 安全套接字扩展(JSSE)的实现时才有效这些属性。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 供应商。例如，IBMJSSE2 或 SunJSSE com.ibm.ssl.keyStore 包含您希望 KeyManager 使用的 KeyStore 对象的名称。例如 /mydir/etc/key.p12 com.ibm.ssl.keyStorePassword 您希望 KeyManager 使用的 KeyStore 对象的名称。密码可以采用纯文本，也可以使用静态方法模糊处理： com.ibm.micro.security.Password.obfuscate (char password)。这使用简单和不安全的 XOR 和 Base64 编码机制混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。 com.ibm.ssl.keyStoreType Type of key 存储，如 PKCS12、JKS 或 JCEKS. com.ibm.ssl.keyStoreProvider 存储提供程序，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.trustStore 包含您希望 TrustManager 使用的 KeyStore 对象的名称。 com.ibm.ssl.trustStorePassword 要使用的 TrustStore 对象的名称。密码可以采用纯文本，也可以使用静态方法模糊处理： com.ibm.micro.security.Password.obfuscate (char password)。这使用简单和不安全的 XOR 和 Base64 编码机制混淆密码。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 对象的类型。值与 keyStoreType. com.ibm.ssl.trustStoreProvider Trust 存储供应商相同，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.enabledCipherSuites A 列表。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 设置算法，该算法将用于实例化平台中提供的 KeyManagerFactory 对象，而不使用平台中提供的默认算法。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，它会在连接的握手后使用，您应该在验证主机名错误时自己执行操作。没有默认的 HostnameVerifier。		HostnameVerifier
userName (security)	用于针对 MQTT 代理进行身份验证的用户名。		字符串

48.5. 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	要覆盖并发送到的主题名称，而不是在端点上指定的主题

48.6. 默认有效负载类型

默认情况下，Camel Paho 组件在从（或放入）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 构建 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 转换为字符串 （相反）：

```
// Receive payload
String payload = consumerTemplate.receiveBody("paho:topic", String.class);

// Send payload
String payload = "message";
producerTemplate.sendBody("paho:topic", payload);
```

48.7. SAMPLES

例如，以下片段从与 Camel 路由器相同的主机上安装的 mq 代理读取信息：

```
from("paho:some/queue")
    .to("mock:test");
```

虽然以下片段会向 MQTT 代理发送信息：

```
from("direct:test")
    .to("paho:some/target/queue");
```

例如，这是如何从远程 mq 代理读取信息：

```
from("paho:some/queue?brokerUrl=tcp://iot.eclipse.org:1883")
    .to("mock:test");
```

在这里，我们将覆盖默认主题，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");
```

48.8. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	如果连接丢失，设置客户端是否自动尝试重新连接到服务器。如果设置为 false，客户端不会在连接丢失时尝试自动连接到服务器。如果设置为 true，如果连接丢失，客户端会尝试重新连接服务器。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-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
camel.component.paho-mqtt5.broker-url	mq 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是否应该在重启和重新连接后记住状态。如果设置为 false，客户端和服务器在重启客户端后维护状态，则服务器和客户端的连接。维护状态 ：消息交付将可靠满足指定的 QOS，即使客户端、服务器或连接重启也是如此。服务器会将订阅视为 durable。如果设置为 true，则客户端和服务器在重启客户端后不维护状态，服务器或连接将不会被维护。这意味着，如果客户端、服务器或连接重启，则无法维护发送到指定的 QOS 的消息。服务器会将订阅视为不可durable。	true	布尔值
camel.component.paho-mqtt5.client	使用共享的 Paho 客户端。选项是 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要等待与 MQTT 服务器的网络连接的最大时间间隔。默认超时为 30 秒。0 代表禁用超时处理意味着客户端将等待网络连接成功或失败。	30	整数

Name	描述	默认值	类型
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	在强制终止前，设置 executor 服务在终止时应等待的时间（以秒为单位）。除非需要修改这个值，否则不建议更改这个值。	1	整数
camel.component.paho-mqtt5.file-persistence-directory	文件持久性使用的基本目录。默认将使用 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	设置保留间隔。这个值（以秒为单位）定义发送或接收消息之间的最大时间间隔。它可让客户端检测服务器是否不再可用，而无需等待 TCP/IP 超时。客户端将确保每个情况下至少有一个消息在网络间传输。如果在时间段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，服务器将确认。0 代表禁用客户端中的 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	获取在重新连接之间等待的最大时间（以 millis 为单位）。	128000	整数

Name	描述	默认值	类型
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(0-2)。	2	整数
camel.component.paho-mqtt5.receive-maximum	设置接收的最大值。这个值代表了客户端同时处理的 QoS 1 和 QoS 2 出版物的限制。没有机制来限制服务器可能会尝试发送的 QoS 0 出版物数量。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。多个服务器可以用逗号分开。每个 serverURI 指定客户端可以连接的服务器的地址。两种类型的连接是 TCP 连接的 tcp://，对于 SSL/TLS 保护的 TCP 连接支持 ssl://。例如： tcp://localhost:1883 ssl://localhost:8883 如果未指定端口，对于 tcp:// URI，它将默认为 1883，对于 ssl:// URI，将默认为 1883。如果设置了 serverURIs，它会覆盖在 MQTT 客户端构造器上传递的 serverURI 参数。当尝试连接启动时，客户端将从列表中的第一个 serverURI 开始，并通过列表进行工作，直到与服务器建立连接。如果无法向任何服务器进行连接，则连接尝试会失败。指定客户端可以连接到的服务器列表具有多个用途：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，其中两个或多个 MQTT 服务器共享状态。MQTT 客户端可以连接到任何相等的服务器，并保证无论客户端连接到哪个服务器，消息都可以可靠交付且可暂停订阅。如果需要具有危险的订阅和/或可靠的消息发送，则 cleansession 标志必须设置为 false。可能需要指定一组不相等的服务器（如在高可用性选项中）。因为在服务器间没有状态共享可靠的消息交付，因此没有可用的订阅无效。如果使用 hunt 列表模式，必须将 cleansession 标志设置为 true。		字符串

Name	描述	默认值	类型
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将保持会话的最长时间。如果客户端打算稍后连接到服务器，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 0，则会话将在网络连接关闭后立即结束。当客户端确定它不再可用于会话时，它应该断开 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 提供特定于应用程序的安全设置。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，只有在有 Java 安全套接字扩展(JSSE)的实现时才有效这些属性。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 of: SSL, SSLv3, TLS, TLSv1, SSL_TLS. com.ibm.ssl.contextProvider Underlying JSSE 供应商。例如，IBMJSSE2 或 SunJSSE com.ibm.ssl.keyStore 包含您希望 KeyManager 使用的 KeyStore 对象的名称。例如 /mydir/etc/key.p12 com.ibm.ssl.keyStorePassword 您希望 KeyManager 使用的 KeyStore 对象的名称。密码可以采用纯文本，也可以使用静态方法模糊处理： com.ibm.micro.security.Password.obfuscate (char password)。这使用简单和不安全的 XOR 和 Base64 编码机制混淆密码。请注意，这只是一个简单的 scrambler 来混淆明文密码。 com.ibm.ssl.keyStoreType Type of key 存储，如 PKCS12、JKS 或 JCEKS. com.ibm.ssl.keyStoreProvider 存储提供程序，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.trustStore 包含您希望 TrustManager 使用的 KeyStore 对象的名称。 com.ibm.ssl.trustStorePassword 要使用的 TrustStore 对象的名称。密码可以采用纯文本，也可以使用静态方法模糊处理： com.ibm.micro.security.Password.obfuscate (char password)。这使用简单和不安全的 XOR 和 Base64 编码机制混淆密码。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 对象的类型。值与 keyStoreType. com.ibm.ssl.trustStoreProvider Trust 存储供应商相同，如 IBMJCE 或 IBMJCEFIPS. com.ibm.ssl.enabledCipherSuites A 列表。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 设置算法，该算法将用于实例化平台中提供的 KeyManagerFactory 对象，而不使用平台中提供的默认算法。示例值： IbmX509 或 IBMJ9X509. com.ibm.ssl.trustManager 设置用于实例化 TrustManagerFactory 对象的算法，而不使用平台中提供的默认算法。示例值：PKIX 或 IBMJ9X509。选项是 java.util.Properties 类型。		Properties
camel.component.paho-mqtt5.ssl-hostname-verifier	为 SSL 连接设置 HostnameVerifier。请注意，它会在连接的握手后使用，您应该在验证主机名错误时自己执行操作。没有默认的 HostnameVerifier。选项是 javax.net.ssl.HostnameVerifier 类型。		HostnameVerifier

Name	描述	默认值	类型
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
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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。为消息设置的 mq 属性。选项是 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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。是否应保留消息。	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了与服务器的连接，服务器将使用提供的详细信息向自己发布一条消息。要发布到的主题。		字符串

第 49 章 平台 HTTP

从 Camel 3.0 开始

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 或者其他运行时上运行 Camel 时。

将以下依赖项添加到此组件的 pom.xml 中：

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-platform-http</artifactId>
  <version>3.20.1.redhat-00050</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

49.1. 平台 HTTP 供应商

要使用平台 HTTP，需要在 classpath 上提供供应商（引擎）。这样做的目的是为不同运行时（如 Quarkus、PolernetX 或 Spring Boot）具有驱动程序。

目前，camel-platform-http-vertx 仅支持 Quarkus 和 VertX。此 JAR 必须在 classpath 上，否则无法使用平台 HTTP 组件，并且在启动时会抛出异常。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-platform-http-vertx</artifactId>
  <version>3.20.1.redhat-00050</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

49.2. 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

49.2.1. 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。

49.2.2. 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。

49.2.3. 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)	为服务请求的 HTTP 服务器引擎实施。		PlatformHttpEngine

49.2.4. 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：

49.2.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
path (consumer)	**需要此** 端点提供 HTTP 请求的路径，用于代理使用 'proxy'。		字符串

49.2.4.2. 查询参数(11 参数)

Name	描述	默认值	类型
consumed (consumer)	此端点接受的内容类型作为输入，如 application/xml 或 application/json. null 或 / 代表无限制。		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)	如果没有找到完全匹配，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)	如果启用并在响应正文的消费者端处理者失败的处理，则响应的堆栈跟踪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 将标头过滤到或从 Camel 消息过滤。		HeaderFilterStrategy
platformHttpEngine (advanced)	用于提供此端点请求的 HTTP 服务器引擎实施。		PlatformHttpEngine

49.3. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	为服务请求的 HTTP 服务器引擎实施。选项是 org.apache.camel.component.platform.http.spi.PlatformHttpEngine 类型。		PlatformHttpEngine

49.3.1. 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，有些标头填充了 HTTP 请求请求行中收到的绝对 URL。这些标头特定于更精简的平台。

目前，这个功能只支持 camel-platform-http-vertx 组件中的 Vert.x。

第 50 章 QUARTZ

仅支持消费者

Quartz 组件使用 Quartz Scheduler 2.x 提供已调度消息交付。每个端点代表不同的计时器（在 Quartz 术语中，一个 Trigger 和 JobDetail）。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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

50.1. URI 格式

```
quartz://timerName?options
quartz://groupName/timerName?options
quartz://groupName/timerName?cron=expression
quartz://timerName?cron=expression
```

组件使用 CronTrigger 或 SimpleTrigger。如果没有提供 cron 表达式，则组件将使用一个简单的触发器。如果没有提供 groupName，则 quartz 组件将使用 Camel 组名称。

50.2. 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

50.2.1. 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。

50.2.2. 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。

50.3. 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，它允许从 JMX 管理 Quartz 调度程序。这个选项是默认的 true。	true	布尔值

Name	描述	默认值	类型
prefixInstanceName (consumer)	是否使用 CamelContext 名称添加 Quartz Scheduler 实例名称。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。您可以将这个选项设置为 false，以在多个 CamelContext 之间重复使用 Quartz 调度程序实例。	true	布尔值
prefixJobNameWithEndpointId (consumer)	是否使用端点 ID 为 quartz 任务添加前缀。这个选项是默认的 false。	false	布尔值
properties (consumer)	配置 Quartz 调度程序的属性。		Map
propertiesFile (consumer)	要从 classpath 加载的属性的文件名。		字符串
propertiesRef (consumer)	对现有属性或 map 的引用，用于在 registry 中查找，用于配置 quartz。		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)	要使用自定义配置的 Quartz 调度程序，而不是创建新的调度程序。		scheduler
schedulerFactory (advanced)	使用用于创建调度程序的自定义调度程序Factory。		SchedulerFactory
autoStartScheduler (scheduler)	调度程序是否应自动启动。这个选项是默认的 true。	true	布尔值
interruptJobsOnShutdown (scheduler)	是否在关闭时中断作业，以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。如果启用此功能，则任何正在运行的作业都可能会因为中断而失败。当某个作业中断时，Camel 将标记交换以停止继续路由，并设置 java.util.concurrent.RejectedExecutionException，从而导致异常。因此请小心地使用它，因为它通常最好允许 Camel 作业安全完成和关闭。	false	布尔值
startDelayedSeconds (scheduler)	启动 quartz 调度程序前等待的秒数。		int

50.4. 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：

50.4.1. 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

50.4.2. 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，则在路由停止时触发器自动删除。否则，如果设为 false，它将保留在调度程序中。当设置为 false 时，它还意味着用户可以在 camel Uri 中重复使用预配置的触发器。只需确保名称匹配。请注意，您无法将 deleteJob 和 pauseJob 设置为 true。	true	布尔值
durableJob (consumer)	作业是否在孤立后保留（没有触发器指向它）。	false	布尔值

Name	描述	默认值	类型
pauseJob (consumer)	如果设置为 true，则在路由停止时自动暂停触发器。否则，如果设为 false，它将保留在调度程序中。当设置为 false 时，它还意味着用户可以在 camel Uri 中重复使用预配置的触发器。只需确保名称匹配。请注意，您无法将 deleteJob 和 pauseJob 设置为 true。	false	布尔值
recoverableJob (consumer)	如果遇到 'recovery' 或 'fail-over'，则指示调度程序是否应该重新执行作业。	false	布尔值
stateful (consumer)	使用 Quartz PersistJobDataAfterExecution 和 DisallowConcurrentExecution 而不是默认作业。	false	布尔值
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)	作业名称是否应该使用端点 ID 前缀。	false	布尔值
triggerParameters (advanced)	在触发器上配置附加选项：		Map
usingFixedCamelContextName (advanced)	如果为 true，JobDataMap 直接使用 CamelContext 名称来引用 CamelContext （如果为 false），JobDataMap 使用 CamelContext 管理名称，该名称可以在部署期间更改。	false	布尔值
autoStartScheduler (scheduler)	调度程序是否应自动启动。	true	布尔值

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早启动，以确保在作业启动前完全启动端点。负值转换了过去的触发开始时间。	500	long

50.4.3. 配置 quartz.properties 文件

默认情况下，Qrtz 将在 classpath 的 org/quartz 目录中查找 quartz.properties 文件。如果您使用 WAR 部署，这意味着将 quartz.properties 放到 WEB-INF/classes/org/quartz 中。

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>
```

50.5. 在 JMX 中启用 QUARTZ 调度程序

您需要配置 quartz 调度程序属性来启用 JMX。
这通常将选项 "org.quartz.scheduler.jmx.export" 设置为配置文件中的 真正 值。

这个选项默认为 true，除非明确禁用。

50.6. 启动 QUARTZ 调度程序

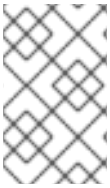
Quartz 组件提供了一个选项，使 Quartz 调度程序启动延迟，或者根本不自动启动。

这是一个示例：

```
<bean id="quartz" class="org.apache.camel.component.quartz.QuartzComponent">
    <property name="startDelayedSeconds" value="5"/>
</bean>
```

50.7. 集群

如果您在集群模式中使用 Quartz，如 JobStore 集群。然后，当节点停止/关闭时，Qrtz 组件不会暂停/删除触发器。https://access.redhat.com/documentation/zh-cn/red_hat_integration/2023.q2/html-single/camel_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，不会进行检查来确保用于端点的唯一作业名称/组。

50.8. 消息标头

Camel 将 Quartz Execution Context 中的 getters 添加为标头值。以下标头会被添加：calendar,fireTime,jobDetail,jobInstance,jobRuntTime,mergedJobDataMap,nextFireTime,previousFireTime,refireCount, result, scheduledFireTime,scheduler,trigger,triggerName,triggerGroup.

fireTime 标头包含触发交换时的 java.util.Date。

50.9. 使用 CRON TRIGGERS

quartz 支持 类似 Cron 的表达式，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，但为了保留有效的 URI 编码，我们允许使用 + 而不是空格。

例如，以下命令会在 weekdays 上每五分钟从 12pm (noon)到 6pm 一次触发一条消息：

```
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的 URI 语法的 URI 字符编码：

URI Character	Cron 字符
+	space

50.10. 指定时区

Quartz 调度程序允许您为每个触发器配置时区。例如，要使用国家/地区时区，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。

50.11. 配置不正确的说明

quartz 调度程序可以使用错误指令配置，以处理触发器的错误情况。您正在使用的 concrete 触发器类型将定义了一组额外的 MISFIRE_INSTRUCTION_XXX 常量，可以设置为此属性的值。

例如，将 simple 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 触发器有以下不正确的指令代表：

50.11.1. SimpleTrigger.MISFIRE_INSTRUCTION_FIRE_NOW = 1 (default)

指示调度程序在出错的情况下，MutyTrigger 希望由调度程序触发。

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 的触发器中使用，则相当于指令 MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_REMAINING_REPEAT_COUNT。

50.11.2. SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_EXISTING_REPEAT_COUNT = 2

指示调度程序在错误的情况下，SimpleTrigger 希望重新调度为"现在"（即使关联的 Calendar excludes 'now'）。这会产生 Trigger 终止时间，因此如果 'now' 在 Trigger 结束时不会再次触发。

使用这个指令可让触发器"forget"，启动时间并重复了它最初设置的重复计数（如果您因为一些原因希望告知原始值在一些时候存在）时才有问题。

50.11.3. SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NOW_WITH_REMAINING_REPEAT_COUNT = 3

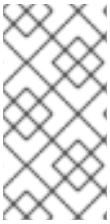
指示调度程序在错误的情况下，MutyTrigger 希望被重新调度为 'now'（即使关联的 Calendar excludes 'now'）,如果还没有丢失任何触发，则它也会被重新调度。这会产生 Trigger 终止时间，因此如果 'now' 在 Trigger 结束时不会再次触发。

使用这个指令可让触发器"forget"，并重复它最初被设置。相反，触发器的重复计数将更改为剩余的重复计数。这只是只有在出于某种原因想告知原始值在以后存在哪些原始值时才有问题。

如果所有未命中的重复时间，这个指令可能会导致 Trigger 在触发 'now' 后进入 'COMPLETE' 状态。

50.11.4. SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_REMAINING_COUNT = 4

指示调度程序在错误的情况下，MutyTrigger 希望在 'now' 后重新调度到下一个调度时间 - 如果还没有丢失任何关联的 Calendar，并将重复计数设置为它。



注意

如果所有触发时间都丢失，则此指令可能会导致 Trigger 直接进入 'COMPLETE' 状态。

50.11.5. SimpleTrigger.MISFIRE_INSTRUCTION_RESCHEDULE_NEXT_WITH_EXISTING_COUNT = 5

指示调度程序在错误的情况下，SimpleTrigger 希望在 'now' 后重新调度到下一个调度时间 - 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到达，这个指令可能会导致 Trigger 直接进入 'COMPLETE' 状态。

50.11.6. CronTrigger.MISFIRE_INSTRUCTION_FIRE_ONCE_NOW = 1 (default)

指示调度程序在异常情况时，C CronTrigger 现在希望由调度程序触发。

50.11.7. CronTrigger.MISFIRE_INSTRUCTION_DO_NOTHING = 2

指示调度程序在出错的情况下，CronTrigger 希望在当前时间后将其下次更新到调度中的下一次时间（不考虑任何关联的 Calendar，但现在不想触发它。

50.12. 使用 QUARTZSCHEDULEDPOLLCONSUMERSCHEDULER

Quartz 组件提供了一个轮询 Consumer 调度程序，它允许使用基于 cron 的调度来轮询消费者，如 File 和 FTP 用户。

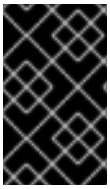
例如，要使用基于 cron 的表达式每第 2 秒轮询一次文件，可将 Camel 路由定义为：

```
from("file:inbox?scheduler=quartz&scheduler.cron=0/2+*+*+*+*+?")
  .to("bean:process");
```

请注意，我们定义了 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，则生成并使用 UUID。
triggerGroup	**QuartzScheduledPollConsumerScheduler**	**字符串**	指定触发器组。
timeZone	**默认值**	**TimeZone**	用于 CRON 触发器的时区。



重要

请记住，从端点 URI 配置这些选项必须以 调度程序 作为前缀。

例如，配置触发器 id 和组：

```
    from("file:inbox?scheduler=quartz&scheduler.cron=0/2+*+*+*+*+?
&scheduler.triggerId=myId&scheduler.triggerGroup=myGroup")
        .to("bean:process");
```

Spring 中还有一个 CRON 调度程序，因此您也可以使用以下内容：

```
    from("file:inbox?scheduler=spring&scheduler.cron=0/2+*+*+*+*+?")
      .to("bean:process");
```

50.13. CRON 组件支持

Quartz 组件可用作 Camel Cron 组件的实现。

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");
```

50.14. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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。这个选项是默认的 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

Name	描述	默认值	类型
camel.component.quartz.enable-jmx	是否启用 Quartz JMX，它允许从 JMX 管理 Quartz 调度程序。这个选项是默认的 true。	true	布尔值
camel.component.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，以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。如果启用此功能，则任何正在运行的作业都可能会因为中断而失败。当某个作业中断时，Camel 将标记交换以停止继续路由，并设置 java.util.concurrent.RejectedExecutionException，从而导致异常。因此请小心地使用它，因为它通常最好允许 Camel 作业安全完成和关闭。	false	布尔值
camel.component.quartz.prefix-instance-name	是否使用 CamelContext 名称添加 Quartz Scheduler 实例名称。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。您可以将这个选项设置为 false，以在多个 CamelContext 之间重复使用 Quartz 调度程序实例。	true	布尔值
camel.component.quartz.prefix-job-name-with-endpoint-id	是否使用端点 ID 为 quartz 任务添加前缀。这个选项是默认的 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	要从 classpath 加载的属性的文件名。		字符串
camel.component.quartz.properties-ref	对现有属性或 map 的引用，用于在 registry 中查找，用于配置 quartz。		字符串
camel.component.quartz.scheduler	要使用自定义配置的 Quartz 调度程序，而不是创建新的调度程序。选项是 org.quartz.Scheduler 类型。		scheduler
camel.component.quartz.scheduler-factory	使用用于创建调度程序的自定义调度程序Factory。选项是 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 51 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。

51.1. URI 格式

```
ref:someName[?options]
```

其中 someName 是 Registry 中的端点名称（通常是 Spring registry）。如果您使用 Spring registry，someName 是 Spring registry 中的端点的 bean ID。

51.2. 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

51.2.1. 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。

51.2.2. 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。

51.3. 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
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

51.4. 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：

51.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
name (common)	**要在** registry 中查找的端点名称。		字符串

51.4.2. 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
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

51.5. 运行时查找

当您需要动态发现 Registry 中的端点时，可以使用此组件，您可以在其中在运行时计算 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);
```

并且您可以在 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>
```

51.6. 示例

在以下示例中，我们在 URI 中使用 ref: 来引用带有 spring ID (endpoint2)的端点 ：

当然，您可以使用 ref 属性：

```
<to uri="ref:endpoint2"/>
```

这是编写它的更多常用方法。

51.7. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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 52 章 REST

支持生成者和消费者

REST 组件允许使用 Rest DSL 和插件到其他 Camel 组件作为 REST 传输来定义 REST 端点 (consumer)。

其余组件也可以用作客户端(producer)来调用 REST 服务。

52.1. URI 格式

```
rest://method:path[:uriTemplate]?[options]
```

52.2. 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

52.2.1. 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。

52.2.2. 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。

52.3. 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 将查找，如果有一个与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestConsumerFactory 在 registry 中注册。如果找到其中任一个，则会使用它。		字符串
apiDoc (producer)	要使用的 swagger 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
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 将查找，如果有一个与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestProducerFactory 在 registry 中注册。如果找到其中任一个，则会使用它。		字符串
host (producer)	要使用的 HTTP 服务的主机和端口（在 swagger 模式中覆盖主机）。		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
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 将查找，如果有一个与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestProducerFactory 在 registry 中注册。如果找到其中任一个，则会使用它。		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

52.4. 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：

52.4.1. 路径参数(3 参数)

Name	描述	默认值	类型

Name	描述	默认值	类型
method (common)	要使用 **的必要** HTTP 方法。 Enum 值： • get • post • put • delete • patch • head • trace • 连接 • 选项		字符串
path (common)	**必需** 基本路径。		字符串
uritemplate (common)	uri 模板。		字符串

52.4.2. 查询参数 (16 参数)

Name	描述	默认值	类型
consumed (common)	介质类型，例如： 'text/xml' 或 'application/json' this REST 服务接受。默认情况下，我们接受所有类型。		字符串
inType (common)	将传入的 POJO 绑定类型声明为 FQN 类名称。		字符串
outType (common)	将传出的 POJO 绑定类型声明为 FQN 类名称。		字符串
generate (common)	介质类型，如： 'text/xml' 或 'application/json' this REST 服务返回。		字符串
routeId (common)	此 REST 服务创建的路由名称。		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 将查找，如果有一个与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestConsumerFactory 在 registry 中注册。如果找到其中任一个，则会使用它。		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)	为制作者配置绑定模式。如果设置为 'off' 以外的任何内容，则制作者会尝试将传入消息的正文从 inType 转换为 json 或 xml，并将响应从 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 将查找，如果有一个与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestProducerFactory 在 registry 中注册。如果找到其中任一个，则会使用它。		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由 ampersand 分开的参数，如 foo=123&bar=456。		字符串

52.5. 支持的其余组件

以下组件支持其余消费者(Rest DSL)：

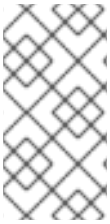
- ***camel-servlet***

以下组件支持剩余的制作者：

- camel-http

52.6. 路径和 URITEMPLATE 语法

path 和 uriTemplate 选项使用 REST 语法来定义，您可以使用参数定义 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配置这两个选项，这无关紧要。虽然同时配置路径和 uriTemplate 是 REST 的更常见做法。

以下是只使用路径的 Camel 路由

```
from("rest:get:hello")
  .transform().constant("Bye World");
```

以下路由使用参数，该参数映射到带有键"主题"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}");
```

52.7. REST 生成者示例

您可以使用其余组件调用 REST 服务，与其他 Camel 组件一样。

例如，要使用 hello/{me} 调用 REST 服务，您可以这样做

```
from("direct:start")
  .to("rest:get:hello/{me}");
```

然后，动态值 {me} 会映射到具有相同名称的 Camel 消息。要调用此 REST 服务，您可以发送空消息正文和一个标头，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 来选择将哪些 Camel 组件用作 HTTP 客户端，例如使用 http ：

```
restConfiguration().host("myserver:8080/foo").producerComponent("http");

from("direct:start")
  .to("rest:get:hello/{me}");
```

52.8. REST 生成者绑定

REST 生成者支持使用 JSon 或 XML （如 rest-dsl ）进行绑定。

例如，要将 jetty 与 json 绑定模式一起使用，您可以在其余配置中配置它：

```
restConfiguration().component("jetty").host("localhost").port(8080).bindingMode(RestBinding
Mode.json);
```

```
from("direct:start")
  .to("rest:post:user");
```

然后，当使用 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：

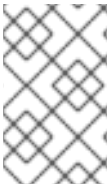
```
UserPojo user = new UserPojo();
user.setId(123);
user.setName("Donald Duck");

template.sendBody("direct:start", user);
```

在上例中，我们发送 POJO 实例 UserPojo 作为消息正文。由于我们在其余配置中打开了 JSon 绑定，因此在调用 REST 服务前，POJO 将从 POJO 变为 JSon。

但是，如果您还要为响应消息执行绑定（例如 REST 服务作为响应发送），则需要配置 outType 选项，以指定 POJO 的类名称，以便从 JSon 到 JSon 到 POJO。

例如，如果 REST 服务返回一个 JSon 有效负载，它绑定到 com.foo.MyResponsePojo，您可以配置它，如下所示：

```
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 POJO 绑定对于调用 REST 服务的响应消息，您必须配置 outType 选项。

52.9. 更多示例

请参阅 Rest DSL，其提供了更多示例，以及如何使用 Rest DSL 以互动方式定义它们。

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 作为可在 Apache Tomcat 上部署或类似的 Web 容器的传输。

52.10. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	要使用的 swagger 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
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
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

Name	描述	默认值	类型
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 将查找，如果有一个与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestConsumerFactory 在 registry 中注册。如果找到其中任一个，则会使用它。		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（在 swagger 模式中覆盖主机）。		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 将查找，如果有一个与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestProducerFactory 在 registry 中注册。如果找到其中任一个，则会使用它。		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 将查找，如果有一个与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestProducerFactory 在 registry 中注册。如果找到其中任一个，则会使用它。		字符串

第 53 章 SAGA

仅支持生成者

Saga 组件提供了一个网桥，用于使用 Saga EIP 在路由中执行自定义操作。

组件应该用于高级任务，如确定完成或组合了 Saga，并将 completionMode 设置为 MANUAL。

有关常见情况下使用 sagas 的帮助信息，请参阅 Saga EIP 文档。

53.1. URI 格式

```
saga:action
```

53.2. 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

53.2.1. 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。

53.2.2. 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。

53.3. 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
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

53.4. 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：

53.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
action (producer)	要执行 **的必要** 操作（完成或编译）。 Enum 值： • COMPLETE • COMPENSATE		SagaEndpointAction

53.4.2. 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

53.5. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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
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

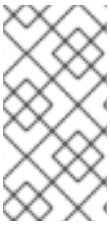
第 54 章 SALESFORCE

支持生成者和消费者

此组件支持生成者和消费者端点，以便使用 Java DTOs 与 Salesforce 进行通信。
存在一个生成这些 DTO 的companion maven 插件 Camel Salesforce 插件（请参阅以下内容）。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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



注意

希望贡献组件的开发人员会被指示查看 README.md 文件，了解如何开始和设置您的环境以运行集成测试。

54.1. 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

54.1.1. 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。

54.1.2. 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。

54.2. 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)	查询 APEX 方法的参数。		Map
apiVersion (common)	Salesforce API 版本。	53.0	字符串
backoffIncrement (common)	backoff 间隔递增流连接重启尝试尝试超过 CometD 自动连接。	1000	long
batchId (common)	批量 API 批处理 ID。		字符串

Name	描述	默认值	类型
contentType (common)	批量 API 内容类型，一个 XML, CSV, ZIP_XML, ZIP_CSV。 Enum 值： • XML • CSV • JSON • ZIP_XML • ZIP_CSV • ZIP_JSON		ContentType
defaultReplayId (common)	如果在 initialReplayIdMap 中没有找到任何值，则默认 replayId 设置。	-1	Long
fallBackReplayId (common)	ReplayId 在 Invalid Replay Id 响应后回退到。	-1	Long
格式 (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)	用于连接到 Salesforce 的自定义 Jetty Http 客户端。		SalesforceHttpClient
httpClientConnectionTimeout (common)	连接到 Salesforce 服务器时 HttpClient 使用的连接超时。	60000	long
httpClientIdleTimeout (common)	在等待 Salesforce 服务器响应时，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

Name	描述	默认值	类型
includeDetails (common)	在 Salesforce1 Analytics 报告中包含详细信息，默认为 false。		布尔值
initialReplayIdMap (common)	重播 ID 以从每个频道名称开始。		Map
InstanceID (common)	Salesforce1 分析报告执行实例 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。适用于某些 API，请参阅 Salesforce 文档。		整数
Locator (common)	Salesforce Bulk 2.0 API 提供的检测程序，用于获取查询任务的结果。		字符串
maxBackoff (common)	流连接重启尝试超过 CometD 自动连接的最大 backoff 间隔。	30000	long
maxRecords (common)	为 Bulk 2.0 Query 检索每个结果集合的最大记录数。请求仍受到大小限制的影响。如果您正在使用大量查询结果，您可能会在收到来自 Salesforce 的所有数据前遇到超时。要防止超时，请在 maxRecords 参数中指定客户端希望接收的最大记录数。这会将结果分成较小的集合，使用这个值作为最大大小。		整数
notFoundBehaviour (common)	设置从 Salesforce API 接收的 404 not found 状态的行为。应将正文设置为 NULL NotFoundBehaviourHQNULL，或在交换 NotFoundBehaviour EXCEPTION 上发出一个异常信号 - 默认。 Enum 值： • 例外 • NULL	例外	NotFoundBehaviour

Name	描述	默认值	类型
notifyForFields (common)	通知字段，选项为 ALL, REFERENCED, SELECT, WHERE。 Enum 值： • ALL • 引用 • 选择 • 其中		NotifyForFieldsEnum
notifyForOperationCreate (common)	通知创建操作，默认为 false (API 版本 = 29.0)。		布尔值
notifyForOperationDelete (common)	通知删除操作，默认为 false (API 版本 = 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 • 创建 • EXTENDED • 更新		NotifyForOperationsEnum
notifyForOperationUndelete (common)	通知未删除操作，默认为 false (API 版本 = 29.0)。		布尔值
notifyForOperationUpdate (common)	通知更新操作，默认为 false (API 版本 = 29.0)。		布尔值
ObjectMapper (common)	自定义 Jackson ObjectMapper，以便在序列化/解码 Salesforce 对象时使用。		ObjectMapper
软件包 （通用）	在什么软件包中，生成的 DTO 类。通常，类将使用 camel-salesforce-maven-plugin 生成。如果使用生成的 DTOs，在 parameters/header 值中使用短 SObject 名称的好处，则设置它。可以使用逗号分隔多个软件包。		字符串

Name	描述	默认值	类型
pkChunking (common)	使用 PK Chunking。仅用于原始 Bulk API。如果需要，批量 2.0 API 自动执行 PK 块。		布尔值
pkChunkingChunkSize (common)	用于 PK Chunking 的块大小。如果未指定，则 Salesforce 默认为 100,000。最大大小为 250,000。		整数
pkChunkingParent (common)	当您为共享对象上的查询启用 PK 块时，请指定父对象。块基于父对象的记录，而不是共享的对象的记录。例如，在查询 AccountShare 时，将 Account 指定为父对象。只要支持父对象，支持 PK 块来共享对象。		字符串
pkChunkingStartRow (common)	指定用作第一个块的下限 15 个字符或 18 个字符记录 ID。在重启批处理之间失败的作业时，请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时，查询 Locator 以供使用。在后续调用中使用这个值来检索其他记录。		字符串
rawPayload (common)	在请求和响应中使用原始有效负载字符串（根据格式 JSON 或 XML），而不是 DTOs，默认为 false。	false	布尔值
reportId (common)	Salesforce1 分析报告 Id.		字符串
reportMetadata (common)	Salesforce1 分析报告元数据进行过滤。		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

Name	描述	默认值	类型
sObjectIdValue (common)	SObject 外部 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)	在使用流 API 时是否更新现有的 Push 主题，默认为 false。	false	布尔值
config (common (advanced))	全局端点配置 - 用于设置所有端点通用的值。		SalesforceEndpointConfig
httpClientProperties (common (advanced))	用于设置可以在底层 HTTP 客户端上配置的任何属性。查看 SalesforceHttpClient 和 Jetty HttpClient 的属性，适用于所有可用选项。		Map
longPollingTransportProperties (common (advanced))	用于设置流 api 使用的 BayeuxClient (CometD)使用的 LongPollingTransport 上配置的任何属性。		Map
workerPoolMaxSize (common (advanced))	用于处理 HTTP 响应的线程池的最大大小。	20	int
workerPoolSize (common (advanced))	用于处理 HTTP 响应的线程池大小。	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，以指示在是否有成功时回滚所有记录。	false	布尔值
apexUrl (producer)	APEX 方法 URL。		字符串

Name	描述	默认值	类型
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。		字符串
rawMethod (producer)	用于 Raw 操作的 HTTP 方法。		字符串
rawPath (producer)	域名后面的端点 URL 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。不要 URL-encode 值，因为会自动完成此操作。		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
httpProxyExcludedAddresses (proxy)	不应该使用 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，请将 HTTP 代理配置为用作 SOCKS4 代理。	false	布尔值

Name	描述	默认值	类型
authenticationType (security)	要使用显式验证方法，USERNAME_PASSWORD、REFRESH_TOKEN 或 JWT 之一。Salesforce 组件可以自动确定要从属性集中使用的身份验证方法，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)	Salesforce 实例设置中配置的连接应用程序的 OAuth Consumer Secret。		字符串
httpProxyAuthUri (security)	对于 HTTP 代理服务器的身份验证，需要匹配代理服务器的 URI，以便 httpProxyUsername 和 httpProxyPassword 用于身份验证。		字符串
httpProxyPassword (security)	用于向 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 代理服务器的抢占基本/目标身份验证方法。		字符串
httpProxySecure (security)	如果设置为 false，在访问 HTTP 代理时禁用使用 TLS。	true	布尔值
httpProxyUseDigestAuth (security)	如果在对 HTTP 代理进行身份验证时使用 true Digest 身份验证，否则将使用基本授权方法。	false	布尔值
httpProxyUsername (security)	用于向 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，默认情况下从 Salesforce 身份验证成功接收到。		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，则使用登录 URL，在大多数情况下都适合。		字符串

Name	描述	默认值	类型
keystore (security)	在 OAuth JWT 流中使用的密钥存储参数。KeyStore 应该仅包含一个带有私钥和证书的条目。Salesforce 不会验证证书链，因此这容易是自签名证书。确保将证书上传到对应的连接的应用程序。		KeyStoreParameters
lazyLogin (security)	如果设置为 true 可防止组件在组件开始时向 Salesforce 进行身份验证。您通常会将其设置为 （默认）false 并早期进行身份验证，并立即了解任何身份验证问题。	false	布尔值
loginConfig (security)	所有嵌套 Bean 中的身份验证配置，所有的属性也可以直接在组件上设置。		SalesforceLoginConfig
loginUrl (security)	用于身份验证的 Salesforce 实例的 **所需** URL，默认为 https://login.salesforce.com。	https://login.salesforce.com	字符串
密码 （安全）	OAuth 流中使用的密码，以获取访问令牌的访问权限。使用密码 OAuth 流很容易入门，但通常要避免它的安全性，这比其它流安全。如果使用，请确保将安全令牌附加到密码的末尾。		字符串
refreshToken (security)	刷新令牌已在刷新令牌 OAuth 流中获取。一个需要设置 Web 应用并配置回调 URL 来接收刷新令牌，或使用 https://login.salesforce.com/services/oauth2/success 或 https://test.salesforce.com/services/oauth2/success 中的内置回调进行配置，然后在流末尾从 URL 撤销 refresh_token。请注意，在开发机构 Salesforce 中，允许在 localhost 上托管回调 Web 应用程序。		字符串
sslContextParameters (security)	要使用的 SSL 参数，请参阅 SSLContextParameters 类以了解所有可用选项。		SSLContextParameters
useGlobalSslContextParameters (security)	启用使用全局 SSL 上下文参数。	false	布尔值
userName (security)	OAuth 流中使用的用户名，以获取访问令牌的访问权限。使用密码 OAuth 流很容易入门，但通常要避免它的安全性，这比其它流安全。		字符串

54.3. 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：

54.3.1. 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 • query • queryMore • queryAll • search • apexCall • recent • createJob • getJob • closeJob • abortJob • createBatch		OperationName

Name	描述	默认值	类型
	• getBatch • getAllBatches • getRequest • getResults • createBatchQuery • getQueryResultIds • getQueryResult • getRecentReports • getReportDescription • executeSyncReport • executeAsyncReport • getReportInstances • getReportResults • limits • 批准 • 批准 • composite-tree • comp-batch • 复合 • compositeRetrieveSObjectCollections • compositeCreateSObjectCollections • compositeUpdateSObjectCollections • compositeUpsertSObjectCollections • compositeDeleteSObjectCollections • bulk2GetAllJobs • bulk2CreateJob • bulk2GetJob • bulk2CreateBatch • bulk2CloseJob • bulk2AbortJob • bulk2DeleteJob • bulk2GetSuccessfulResults • bulk2GetFailedResults		

Name	描述	默认值	类型
	• bulk2GetUnprocessedRecords • bulk2CreateQueryJob • bulk2GetQueryJob • bulk2GetAllQueryJobs • bulk2GetQueryJobResults • bulk2AbortQueryJob • bulk2DeleteQueryJob • raw		
topicName (consumer)	要使用的主题/频道的名称。		字符串

54.3.2. 查询参数(57 参数)

Name	描述	默认值	类型
apexMethod (common)	APEX 方法名称。		字符串
apexQueryParams (common)	查询 APEX 方法的参数。		Map
apiVersion (common)	Salesforce API 版本。	53.0	字符串
backoffIncrement (common)	backoff 间隔递增流连接重启尝试尝试超过 CometD 自动连接。	1000	long
batchId (common)	批量 API 批处理 ID。		字符串
contentType (common)	批量 API 内容类型，一个 XML, CSV, ZIP_XML, ZIP_CSV。 Enum 值： • XML • CSV • JSON • ZIP_XML • ZIP_CSV • ZIP_JSON		ContentType

Name	描述	默认值	类型
defaultReplayId (common)	如果在 initialReplayIdMap 中没有找到任何值，则默认 replayId 设置。	-1	Long
fallBackReplayId (common)	ReplayId 在 Invalid Replay Id 响应后回退到。	-1	Long
格式 (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)	用于连接到 Salesforce 的自定义 Jetty Http 客户端。		SalesforceHttpClient
includeDetails (common)	在 Salesforce1 Analytics 报告中包含详细信息，默认为 false。		布尔值
initialReplayIdMap (common)	重播 ID 以从每个频道名称开始。		Map
InstanceID (common)	Salesforce1 分析报告执行实例 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。适用于某些 API，请参阅 Salesforce 文档。		整数
Locator (common)	Salesforce Bulk 2.0 API 提供的检测程序，用于获取查询任务的结果。		字符串
maxBackoff (common)	流连接重启尝试超过 CometD 自动连接的最大 backoff 间隔。	30000	long
maxRecords (common)	为 Bulk 2.0 Query 检索每个结果集合的最大记录数。请求仍受到大小限制的影响。如果您正在使用大量查询结果，您可能会在收到来自 Salesforce 的所有数据前遇到超时。要防止超时，请在 maxRecords 参数中指定客户端希望接收的最大记录数。这会将结果分成较小的集合，使用这个值作为最大大小。		整数

Name	描述	默认值	类型
notFoundBehaviour (common)	设置从 Salesforce API 接收的 404 not found 状态的行为。应将正文设置为 NULL NotFoundBehaviourHQNULL，或在交换 NotFoundBehaviour EXCEPTION 上发出一个异常信号 - 默认。 Enum 值： • 例外 • NULL	例外	NotFoundBehaviour
notifyForFields (common)	通知字段，选项为 ALL, REFERENCED, SELECT, WHERE。 Enum 值： • ALL • 引用 • 选择 • 其中		NotifyForFieldsEnum
notifyForOperationCreate (common)	通知创建操作，默认为 false (API 版本 = 29.0)。		布尔值
notifyForOperationDelete (common)	通知删除操作，默认为 false (API 版本 = 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 • 创建 • EXTENDED • 更新		NotifyForOperationsEnum
notifyForOperationUndelete (common)	通知未删除操作，默认为 false (API 版本 = 29.0)。		布尔值

Name	描述	默认值	类型
notifyForOperationUpdate (common)	通知更新操作，默认为 false (API 版本 = 29.0)。		布尔值
ObjectMapper (common)	自定义 Jackson ObjectMapper，以便在序列化/解码 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 自动执行 PK 块。		布尔值
pkChunkingChunkSize (common)	用于 PK Chunking 的块大小。如果未指定，则 Salesforce 默认为 100,000。最大大小为 250,000。		整数
pkChunkingParent (common)	当您为共享对象上的查询启用 PK 块时，请指定父对象。块基于父对象的记录，而不是共享的对象的记录。例如，在查询 AccountShare 时，将 Account 指定为父对象。只要支持父对象，支持 PK 块来共享对象。		字符串
pkChunkingStartRow (common)	指定用作第一个块的下限 15 个字符或 18 个字符记录 ID。在重启批处理之间失败的作业时，请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时，查询 Locator 以供使用。在后续调用中使用这个值来检索其他记录。		字符串
rawPayload (common)	在请求和响应中使用原始有效负载字符串（根据格式 JSON 或 XML），而不是 DTOs，默认为 false。	false	布尔值
reportId (common)	Salesforce1 分析报告 Id.		字符串
reportMetadata (common)	Salesforce1 分析报告元数据进行过滤。		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

Name	描述	默认值	类型
sObjectId (common)	API 需要 SObject ID。		字符串
sObjectIdName (common)	SObject 外部 ID 字段名称。		字符串
sObjectIdValue (common)	SObject 外部 ID 字段值。		字符串
sObjectName (common)	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)	在使用流 API 时是否更新现有的 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
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，以指示在是否有成功时回滚所有记录。	false	布尔值

Name	描述	默认值	类型
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。		字符串
rawMethod (producer)	用于 Raw 操作的 HTTP 方法。		字符串
rawPath (producer)	域名后面的端点 URL 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。不要 URL-encode 值，因为会自动完成此操作。		字符串

54.4. 向 SALESFORCE 进行身份验证

组件支持三个 OAuth 身份验证流程：

- *OAuth 2.0 Username-Password Flow*
- *OAuth 2.0 刷新令牌流*
- *OAuth 2.0 JWT Bearer 令牌流*

对于每个流不同的属性集合，需要设置：

表 54.1. 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	连接的 App, Consumer Key	所有流
clientSecret	connected App, Consumer Secret	username-Password, Refresh Token
userName	Salesforce 用户名	用户名密码、JWT Bearer 令牌
password	Salesforce 用户密码	username-Password
refreshToken	来自 OAuth 流回调	刷新令牌
keystore	连接的应用程序，过期证书	JWT Bearer 令牌

组件自动决定您要配置的流，以删除模糊设置 authenticationType 属性。

注意

不建议在生产环境中使用 Username-Password Flow。

注意

JWT Bearer Token Flow 中使用的证书可以是自签名证书。保存证书和私钥的 KeyStore 必须仅包含单个证书私钥条目。

54.5. URI 格式

当用作消费者时，接收流事件时，URI 方案为：

```
salesforce:topic?options
```

当用作生成者时，调用 Salesforce REST API，URI 方案为：

```
salesforce:operationName?options
```

54.6. 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，在请求中以 Sforce 或 x-sfdc 开头的标头名称将出现在请求中，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：

```
// 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);
    }
}
```

此外，HTTP 响应状态代码和文本可作为标头 Exchange.HTTP_RESPONSE_CODE 和 Exchange.HTTP_RESPONSE_TEXT 提供。

54.7. 支持的 SALESFORCE API

组件支持以下 Salesforce API

制作者端点可以使用以下 API：大多数 API 每次处理一个记录，Query API 可以检索多个记录。

54.7.1. REST API

您可以将以下内容用于 operationName ：

- 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 - 使用外部 ID 更新或插入 SObject*
- *deleteSObjectWithId - 使用外部 ID 删除 SObject*
- *Query - 运行 Salesforce SOQL 查询*
- *queryMore - 使用从 'query' API 返回的结果链接检索更多结果（如果有大量结果）*
- *Search - 运行 Salesforce SOSL 查询*
- *限制 - 获取机构 API 用量限制*
- *recent - 获取最新的项目*

- ***Approval - 提交记录或记录（批量）以进行批准过程***
- ***Approvals - 获取所有批准过程的列表***
- ***复合 - 提交最多 25 个可能相关的 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。***
- ***comp-tree - 一次性创建最多 200 个带有父子关系（最多 5 级）的记录***
- ***comp-batch - 提交批处理中请求组成***
- ***CompRetrieveSObjectCollections - 获取同一对象类型的一个或多个记录。***
- ***CompCreateSObjectCollections - 添加最多 200 个记录，返回 SaveSObjectResult 对象的列表。***
- ***compUpdateSObjectCollections - 更新最多 200 个记录，返回 SaveSObjectResult 对象的列表。***
- ***CompUpsertSObjectCollections - 根据外部 ID 字段创建或更新最多 200 个记录。返回 UpsertSObjectResult 对象列表。***
- ***CompDeleteSObjectCollections - 删除最多 200 个记录，返回 SaveSObjectResult 对象的列表。***
- ***queryAll - 运行 SOQL 查询。它返回因为合并而删除的结果（将三个记录合并到其中一个记录中，删除其他记录，以及重新清除任何相关的记录）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 - 向 Salesforce 发送请求，并对端点、参数、正文等完全控制。*

例如，以下制作者端点使用 upsertSObject API，sObjectIdName 参数指定 'Name' 作为 external id 字段。请求消息正文应该是使用 maven 插件生成的 SObject DTO。如果现有记录被更新，或者具有新记录的 id 的 CreateSObjectResult 或创建新对象时的错误列表，响应消息将为空。

```
...to("salesforce:upsertSObject?sObjectIdName=Name")...
```

54.7.2. 批量 2.0 API

Bulk 2.0 API 与原始 Bulk API 相比具有简化的模型。使用它快速将大量数据加载到 Salesforce 中，或者从 Salesforce 中查询大量数据。数据必须以 CSV 格式提供。Bulk 2.0 的最低 API 版本是 v41.0。Bulk Queries 的最低 API 版本是 v47.0。下面提到的 DTO 类来自 org.apache.camel.component.salesforce.api.dto.bulkv2 软件包。支持以下操作：

- *bulk2CreateJob - 创建一个批量作业。在消息正文 中提供 一个作业实例。*
- *bulk2GetJob - 获取现有的作业。jobId 参数是必需的。*
- *bulk2CreateBatch - 向作业中添加 CSV 记录批处理。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 - 关闭一个作业。您必须关闭作业才能进行处理或中止/删除。jobId 参数是必需的。*
- *bulk2AbortJob - Abort 一个作业。jobId 参数是必需的。*
- *bulk2DeleteJob - 删除作业。jobId 参数是必需的。*
- *bulk2GetSuccessfulResults - 获取作业的成功结果。返回的消息正文将包含 CSV 数据的 InputStream。jobId 参数是必需的。*

- *bulk2GetFailedResults - 获取失败的作业结果。返回的消息正文将包含 CSV 数据的 InputStream。jobId 参数是必需的。*

- *bulk2GetUnprocessedRecords - 为作业获取未处理的记录。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，则还有额外的页面来获取页面，下一个RecordsUrl 属性包含要在后续调用上的 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 是一个批量查询作业。jobId 参数是必需的。*

- *bulk2DeleteQueryJob - 删除批量查询作业。jobId 参数是必需的。*

- *bulk2GetAllQueryJobs - 获取所有作业。响应正文是 QueryJobs 的实例。如果 done 属性为 false，则还有额外的页面来获取页面，下一个RecordsUrl 属性包含要在后续调用上的 queryLocator 参数中设置的值。*

54.7.3. REST Bulk (original) API

制作者端点可以使用以下 API：支持所有作业数据格式、i.e. xml、csv、zip/xml 和 zip/csv。路由必须对请求和响应进行 marshalled/unmarshalled。通常，请求是一些流源，如 CSV 文件，，响应也可以保存到要与请求关联的文件中。

您可以将以下内容用于 operationName ：

- *CreateJob - 创建 Salesforce Bulk 作业。必须在正文中提供 JobInfo 实例。pkChunking 选项支持 PK Chunking。详情请参阅 这里 的说明。*
- *getJob - 使用其 Salesforce Id 获取作业*
- *closeJob - 关闭一个作业*
- *abortJob - Aborts a Job*
- *createBatch - 提交 Bulk 作业中的批处理*
- *getBatch - 使用 Id 获取批处理*
- *getAllBatches - 获取 Bulk Job Id 的所有批处理*
- *getRequest - 获取 Batch 的请求数据(XML/CSV)*
- *getResults - 完成后获取 Batch 的结果*
- *createBatchQuery - 从 SOQL 查询创建批处理*
- *getQueryResultIds - 获取 Batch Query 的 Result Ids 列表*
- *getQueryResult - 获取 Result Id 的结果*
- *get recentlyReports - 通过向 Report List 资源发送 GET 请求，最多获取您最近查看的报告 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 - 与不更改过滤器同步运行报告并返回最新的概述数据。*
- *executeAsyncReport - 异步运行报告实例，并带有或没有过滤器并返回摘要数据。*
- *getReportInstances - 返回您请求异步运行的报告的实例列表。列表中的每个项目被视为单独的报告实例。*
- *getReportResults ：包含运行报告的结果。*

例如，以下制作者端点使用 createBatch API 创建作业批处理。消息中必须包含可转换为 InputStream （通常为 UTF-8 CSV 或 XML 内容）的正文，以及作业内容类型的标头字段 'jobId' 和 'contentType'，可以是 XML、CSV、ZIP_XML 或 ZIP_CSV。放置消息正文将在成功上包含 BatchInfo，或者在错误上抛出 SalesforceException。

```
...to("salesforce:createBatch")..
```

54.7.4. REST Streaming API

消费者端点可以使用以下语法流端点，以在创建/更新时接收 Salesforce 通知。

创建并订阅主题

```
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")...
```

54.7.5. 平台事件

要发出平台事件，请使用 createSObject 操作。并且设置消息正文可以是 JSON 字符串，也可以是带有键-值数据的 InputStream，该键是 sObjectName，需要设置为事件 API 名称，或者从

AbstractDTOBase 扩展的类以及事件的适当类名称。

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，使用带有前缀为 event/ （或 / event/ ）的平台事件的 API 名称的使用者端点，例如： Salesforce :events/Order_Event__e。从该端点消耗的处理器将分别接收 org.apache.camel.component.salesforce.api.dto.PlatformEvent 对象或 org.cometd.bayeux.Message，具体取决于 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);
```

54.7.6. 更改数据捕获事件

另一方面，可将 Salesforce 配置为发出通知以记录选择对象的更改。另一方面，Camel Salesforce 组件可能会响应此类通知，允许实例将 这些更改同步到外部系统。

通过订阅频道，可以在 Camel 路由的 from ("salesforce:XXX") 子句中指定感兴趣的通知，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")
```

收到的消息分别包含 body 中的 java.util.Map<String,Object > 或 org.cometd.bayeux.Message，具体取决于 rawPayload 为 false 或 true。CamelSalesforceChangeType 标头可值为 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 很适合更好地了解实施更改数据捕获集成的应用程序的子点。更改事件正文字段的动态性质、高级别复制步骤以及安全考虑可能值得关注。

54.8. 例子

54.8.1. 将文档上传到内容工作空间

使用 Processor 实例在 Java 中创建 ContentVersion ：

```
public class ContentProcessor implements Processor {
    public void process(Exchange exchange) throws Exception {
        Message message = exchange.getIn();

        ContentVersion cv = new ContentVersion();
        ContentWorkspace cw = getWorkspace(exchange);
        cv.setFirstPublishLocationId(cw.getId());
        cv.setTitle("test document");
        cv.setPathOnClient("test_doc.html");
        byte[] document = message.getBody(byte[].class);
        ObjectMapper mapper = new ObjectMapper();
        String enc = mapper.convertValue(document, String.class);
        cv.setVersionDataUrl(enc);
        message.setBody(cv);
    }

    protected ContentWorkspace getWorkSpace(Exchange exchange) {
        // Look up the content workspace somehow, maybe use enrich() to add it to a
```

```
        // header that can be extracted here
        ----
    }
}
```

将处理器的输出提供给 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");
```

54.9. 使用 SALESFORCE LIMITS API

使用 Salesforce ：限制 操作，您可以从 Salesforce 获取 API 限制，然后对收到的数据进行操作。salesforce:limits 操作的结果映射到 org.apache.camel.component.salesforce.api.dto.Limits 类，并可用于自定义处理器或表达式。

例如，请考虑您需要限制 Salesforce 的 API 使用量，以便为其他路由保留 10% 每日 API 请求。输出消息的正文包含 org.apache.camel.component.salesforce.api.dto.Limits 对象实例，可与 Content Based Router 和 Content Based Router and Spring Expression Language (SpEL) 结合使用，用于选择执行查询。

请注意，在 body.dailyApiRequests.remaining 中保存的整数值乘以 1.0 如何使表达式评估为与浮点算一样的表达式评估，而不包括浮动点，则最终最终产生集成块，从而导致有 0 （消耗一些 API 限制）或 1 (没有 API 限制)。

```
from("direct:querySalesforce")
    .to("salesforce:limits")
    .choice()
    .when(spel("#{1.0 * body.dailyApiRequests.remaining / body.dailyApiRequests.max < 0.1}"))
        .to("salesforce:query?...")
    .otherwise()
        .setBody(constant("Used up Salesforce API limits, leaving 10% for critical routes"))
    .endChoice()
```

54.10. 使用批准

所有属性都与带有 批准前缀的 Salesforce REST API 中完全相同。您可以通过设置 Endpoint 的 approval.PropertyName 来设置批准属性，这些属性将用作 template，即 body 或 header 中没有的任何属性都将从 Endpoint 配置中获取。或者，您可以通过将批准属性分配给 Registry 中的 bean 的引用来设置端点上 的批准 模板。

您还可以使用传入消息标头中使用相同的 approval.PropertyName 提供标头值。

最后，正文可以包含一个 AprovalRequest 或 ApprovalRequest 对象的 Iterable，以作为批处理处理。

要记住的重要事项是这三个机制中指定的值的优先级：

1. 正文中的值在任何其他前具有优先级
2. 消息标头中的值在模板值前具有优先权
3. 如果未指定标头或正文中的其他值，则会设置模板中的值

例如，要使用标头中的值为批准发送一个记录：

给定路由：

```
from("direct:example1")//
        .setHeader("approval.ContextId", simple("${body['contextId']}"))
        .setHeader("approval.NextApproverIds", simple("${body['nextApproverIds']}"))
        .to("salesforce:approval?"//
           + "approval.actionType=Submit"//
           + "&approval.comments=this is a test"//
           + "&approval.processDefinitionNameOrId=Test_Account_Process"//
           + "&approval.skipEntryCriteria=true");
```

您可以使用以下方法发送记录以进行批准：

```
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);
```

54.11. 使用 SALESFORCE 最新的项目 API

要获取最新的项目，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}");
```

54.12. 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，并在输入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，对于某些记录操作，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在树中标识各个对象的引用，用于数据库中的实例主密钥。

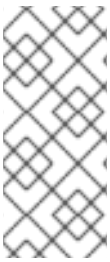
我们来看一个示例：

```
Account account = ...
Contact president = ...
Contact marketing = ...

Account anotherAccount = ...
Contact sales = ...
Asset someAsset = ...

// build the tree
SObjectTree request = new SObjectTree();
request.addObject(account).addChildren(president, marketing);
request.addObject(anotherAccount).addChild(sales).addChild(someAsset);

final SObjectTree response = template.requestBody("salesforce:composite-tree", tree,
SObjectTree.class);
final Map<Boolean, List<SObjectNode>> result = response.allNodes()
                                    .collect(Collectors.groupingBy(SObjectNode::hasErrors));

final List<SObjectNode> withErrors = result.get(true);
```

```
final List<SObjectNode> succeeded = result.get(false);

final String firstId = succeeded.get(0).getId();
```

54.13. 使用 SALESFORCE COMPOSITE API 提交批处理中的多个请求

Composite API batch 操作(comp-batch)允许您汇总批处理中的多个请求，然后从一个容器中提交它们，从而节省多个单独的请求的往返成本。然后，每个响应都会在保留顺序的响应列表中收到，因此第 N 条请求响应位于响应的 n 位置。



注意

结果可能与 API 到 API 的不同，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，字符串键和值或其他 java.util.Map 作为值。请求以 JSON 格式发出，并保存某些类型信息（例如，已知值是字符串的值以及数字的数值）。

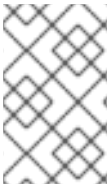
我们来看一个示例：

```
final String acountId = ...
final SObjectBatch batch = new SObjectBatch("38.0");

final Account updates = new Account();
updates.setName("NewName");
batch.addUpdate("Account", accountId, updates);

final Account newAccount = new Account();
newAccount.setName("Account created from Composite batch API");
batch.addCreate(newAccount);

batch.addGet("Account", accountId, "Name", "BillingPostalCode");

batch.addDelete("Account", accountId);

final SObjectBatchResponse response = template.requestBody("salesforce:composite-batch", 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
```

54.14. 使用 SALESFORCE COMPOSITE API 提交多个链请求

复合 操作允许最多提交 25 个请求，这些请求可以组合在一起，用于后续请求中生成的实例标识符。单个请求和响应与提供的 参考 相关联。



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 而异，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，字符串键和值或其他 java.util.Map 作为值。以 JSON 格式发出请求包含某些类型信息（例如，知道哪个值是字符串的值以及数字的数值）。

我们来看一个示例：

```
SObjectComposite composite = new SObjectComposite("38.0", true);

// first insert operation via an external id
final Account updateAccount = new TestAccount();
updateAccount.setName("Salesforce");
updateAccount.setBillingStreet("Landmark @ 1 Market Street");
updateAccount.setBillingCity("San Francisco");
updateAccount.setBillingState("California");
updateAccount.setIndustry(Account_IndustryEnum.TECHNOLOGY);
composite.addUpdate("Account", "001xx000003DlpcAAG", 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("001xx000003DlpcAAG");
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()
```

54.15. 使用 "RAW" 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 的形式创建正文，并使用 rawPayload=true 选项直接提交到 Salesforce 端点。

使用这个方法，您可以对 Salesforce 请求有完全的控制权。

POST 是向 Salesforce 发送原始复合请求的默认 HTTP 方法。使用 CompMethod 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。

54.16. 使用 RAW 操作

将 HTTP 请求发送到 Salesforce，并完全控制调用的所有方面。必须在路由中执行任何请求和响应正文的序列化或反序列化。Content-Type HTTP 标头将根据 format 选项自动设置，但可以使用 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**	域名后面的端点 URL 部分，如 '/services/data/v51.0/sobjects/Account/'		x
rawMethod	**字符串**	HTTP 方法		x
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，因为会自动完成此操作。		
rawHttpHeaders	**字符串**	以逗号分隔的消息标头列表，以作为 HTTP 标头包括		

54.16.1. 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作为 URL 参数包括：

```
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
```

54.16.2. SObject 示例

在本例中，我们将在 create 操作中传递 REST API。由于 原始 操作不执行任何序列化，因此我们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")
```

响应为：

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Result>
   <id>0034x00000RnV6zAAF</id>
   <success>true</success>
</Result>
```

54.17. 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。整个请求计数为单一调用，成为您的 API 限制。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(aka DTO)必须是 AbstractDescribedSObjectBase 的子类的实例。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。

54.17.1. 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	字符串或以逗号分隔的字符串列表	要返回的对象的一个或多个 ID 列表。所有 ID 必须属于相同的对象类型。		x
fields	字符串或以逗号分隔的字符串列表	响应中包含的字段列表。您指定的字段名称必须有效，并且每个字段必须具有 read-level 权限。		x
sObjectName	字符串	SObject 类型，如 帐户		x

参数	类型	描述	默认值	必填
sObjectClass	字符串	用于反序列化响应的 DTO 类的完全限定类名称。		如果 **sObjectName** 参数没有解析为由 **package** 选项指定的软件包中存在的类，则需要此项。

54.17.2. 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
allOrNone	布尔值	指定在创建任何对象失败时是否回滚整个请求(true)还是继续创建请求中的其他对象。	false	

54.17.3. 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	指明是否在更新任何对象失败时回滚整个请求(true)，或者继续更新请求中其他对象的单独更新。	false	

54.17.4. compositeUpsertSObjectCollections

根据外部 ID 字段创建或更新(upsert)最多 200 记录，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	到 upsert 的 SObjects 列表		x
allOrNone	布尔值	指明是否在任何对象的 upsert (true)或继续处理请求中其他对象的单独时回滚整个请求。	false	
sObjectName	字符串	SObject 类型，如 **帐户**		x
sObjectIdName	字符串	外部 ID 字段的名称		x

54.17.5. compositeDeleteSObjectCollections

删除最多 200 个记录，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
sObjectIds 或请求正文	字符串或以逗号分隔的字符串列表	要删除对象的最多 200 个 ID 列表。		x
allOrNone	布尔值	指定在删除任何对象失败时是否回滚整个请求(true)或继续删除请求中的其他对象。	false	

54.18. 向 SALESFORCE 发送 NULL 值

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");
```

54.19. 生成 SOQL 查询字符串

org.apache.camel.component.salesforce.api.utils.QueryHelper 包含生成 SOQL 查询的帮助方法。例如，要从 帐户 SObject 获取所有自定义字段，只需通过调用生成 SOQL SELECT：

```
String allCustomFieldsQuery = QueryHelper.queryToFetchFilteredFieldsOf(new Account(),
SObjectField::isCustom);
```

54.20. CAMEL SALESFORCE MAVEN 插件

此 Maven 插件为 Camel 生成 DTO。

出于安全考虑，建议不会在 pom.xml 中设置 clientId、clientSecret、userName 和 password 字段。该插件应该为其它属性配置，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 DTO 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。

54.21. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

组件支持 91 个选项，如下所列。

Name	描述	默认值	类型
camel.component.salesforce.all-or-none	复合 API 选项，以指示在是否有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

Name	描述	默认值	类型
camel.component.salesforce.apex-query-params	查询 APEX 方法的参数。		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	要使用显式验证方法，USERNAME_PASSWORD、REFRESH_TOKEN 或 JWT 之一。Salesforce 组件可以自动确定要从属性集中使用的身份验证方法，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	backoff 间隔递增流连接重启尝试尝试超过 CometD 自动连接。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
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	Salesforce 实例设置中配置的连接的应用程序的 OAuth Consumer Key。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	Salesforce 实例设置中配置的连接应用程序的 OAuth Consumer Secret。		字符串

Name	描述	默认值	类型
camel.component.salesforce.composite-method	复合（原始）方法。		字符串
camel.component.salesforce.config	全局端点配置 - 用于设置所有端点通用的值。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，一个 XML, CSV, ZIP_XML, ZIP_CSV。		ContentType
camel.component.salesforce.default-replay-id	如果在 initialReplayIdMap 中没有找到任何值，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
camel.component.salesforce.http-client	用于连接到 Salesforce 的自定义 Jetty Http 客户端。选项是一个 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	在等待 Salesforce 服务器响应时，HttpClient 使用的超时。	10000	Long

Name	描述	默认值	类型
camel.component.salesforce.http-client-properties	用于设置可以在底层 HTTP 客户端上配置的任何属性。查看 SalesforceHttpClient 和 Jetty HttpClient 的属性，适用于所有可用选项。		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	对于 HTTP 代理服务器的身份验证，需要匹配代理服务器的 URI，以便 httpProxyUsername 和 httpProxyPassword 用于身份验证。		字符串
camel.component.salesforce.http-proxy-excluded-addresses	不应该使用 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	应使用 HTTP 代理服务器的地址列表。		Set
camel.component.salesforce.http-proxy-password	用于向 HTTP 代理服务器进行身份验证的密码。		字符串
camel.component.salesforce.http-proxy-port	要使用的 HTTP 代理服务器的端口号。		整数
camel.component.salesforce.http-proxy-realm	代理服务器的域，用于针对 HTTP 代理服务器的抢占基本/目标身份验证方法。		字符串
camel.component.salesforce.http-proxy-secure	如果设置为 false，在访问 HTTP 代理时禁用使用 TLS。	true	布尔值
camel.component.salesforce.http-proxy-socks4	如果设置为 true，请将 HTTP 代理配置为用作 SOCKS4 代理。	false	布尔值

Name	描述	默认值	类型
camel.component.salesforce.http-proxy-use-digest-auth	如果在对 HTTP 代理进行身份验证时使用 true Digest 身份验证，否则将使用基本授权方法。	false	布尔值
camel.component.salesforce.http-proxy-username	用于向 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 报告中包含详细信息，默认为 false。		布尔值
camel.component.salesforce.initial-replay-id-map	重播 ID 以从每个频道名称开始。		Map
camel.component.salesforce.instance-id	Salesforce1 分析报告执行实例 ID。		字符串
camel.component.salesforce.instance-url	身份验证后使用的 Salesforce 实例的 URL，默认情况下从 Salesforce 身份验证成功接收到。		字符串
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，则使用登录 URL，在大多数情况下都适合。		字符串
camel.component.salesforce.keystore	在 OAuth JWT 流中使用的密钥存储参数。KeyStore 应该仅包含一个带有私钥和证书的条目。Salesforce 不会验证证书链，因此这容易是自签名证书。确保将证书上传到对应的连接的应用程序。选项是 org.apache.camel.support.jsse.KeyStoreParameters 类型。		KeyStoreParameters
camel.component.salesforce.lazy-login	如果设置为 true 可防止组件在组件开始时向 Salesforce 进行身份验证。您通常会将其设置为 （默认）false 并早期进行身份验证，并立即了解任何身份验证问题。	false	布尔值

Name	描述	默认值	类型
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。适用于某些 API，请参阅 Salesforce 文档。		整数
camel.component.salesforce.locator	Salesforce Bulk 2.0 API 提供的检测程序，用于获取查询任务的结果。		字符串
camel.component.salesforce.login-config	所有嵌套 Bean 中的身份验证配置，所有的属性也可以直接在组件上设置。选项是一个 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，默认为。		字符串
camel.component.salesforce.long-polling-transport-properties	用于设置流 api 使用的 BayeuxClient (CometD)使用的 LongPollingTransport 上配置的任何属性。		Map
camel.component.salesforce.max-backoff	流连接重启尝试超过 CometD 自动连接的最大 backoff 间隔。选项是一个长类型。	30000	Long
camel.component.salesforce.max-records	为 Bulk 2.0 Query 检索每个结果集合的最大记录数。请求仍受到大小限制的影响。如果您正在使用大量查询结果，您可能会在收到来自 Salesforce 的所有数据前遇到超时。要防止超时，请在 maxRecords 参数中指定客户端希望接收的最大记录数。这会将结果分成较小的集合，使用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 状态的行为。应将正文设置为 NULL NotFoundBehaviourHQNULL，或在交换 NotFoundBehaviour EXCEPTION 上发出一个异常信号 - 默认。		NotFoundBehaviour
camel.component.salesforce.notify-for-fields	通知字段，选项为 ALL, REFERENCED, SELECT, WHERE。		NotifyForFieldsEnum

Name	描述	默认值	类型
camel.component.salesforce.notify-for-operation-create	通知创建操作，默认为 false (API 版本 = 29.0)。		布尔值
camel.component.salesforce.notify-for-operation-delete	通知删除操作，默认为 false (API 版本 = 29.0)。		布尔值
camel.component.salesforce.notify-for-operation-undelete	通知未删除操作，默认为 false (API 版本 = 29.0)。		布尔值
camel.component.salesforce.notify-for-operation-update	通知更新操作，默认为 false (API 版本 = 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/解码 Salesforce 对象时使用。选项是一个 com.fasterxml.jackson.databind.ObjectMapper 类型。		ObjectMapper
camel.component.salesforce.packages	在什么软件包中，生成的 DTO 类。通常，类将使用 camel-salesforce-maven-plugin 生成。如果使用生成的 DTOs，在 parameters/header 值中使用短 SObject 名称的好处，则设置它。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。使用密码 OAuth 流很容易入门，但通常要避免它的安全性，这比其它流安全。如果使用，请确保将安全令牌附加到密码的末尾。		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 自动执行 PK 块。		布尔值

Name	描述	默认值	类型
camel.component.salesforce.pk-chunking-chunk-size	用于 PK Chunking 的块大小。如果未指定，则 Salesforce 默认为 100,000。最大大小为 250,000。		整数
camel.component.salesforce.pk-chunking-parent	当您为共享对象上的查询启用 PK 块时，请指定父对象。块基于父对象的记录，而不是共享的对象的记录。例如，在查询 AccountShare 时，将 Account 指定为父对象。只要支持父对象，支持 PK 块来共享对象。		字符串
camel.component.salesforce.pk-chunking-start-row	指定用作第一个块的下限 15 个字符或 18 个字符记录 ID。在重启批处理之间失败的作业时，请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时，查询 Locator 以供使用。在后续调用中使用这个值来检索其他记录。		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。		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	域名后面的端点 URL 部分。E.g., '/services/data/v52.0/sobjects/Account/'.		字符串
camel.component.salesforce.raw-payload	在请求和响应中使用原始有效负载字符串（根据格式 JSON 或 XML），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。不要 URL-encode 值，因为会自动完成此操作。		字符串
camel.component.salesforce.refresh-token	刷新令牌已在刷新令牌 OAuth 流中获取。一个需要设置 Web 应用并配置回调 URL 来接收刷新令牌，或使用内置回调进行配置，然后在流末尾从 URL 撤销 refresh_token。请注意，在开发机构 Salesforce 中，允许在 localhost 上托管回调 Web 应用程序。		字符串

Name	描述	默认值	类型
camel.component.salesforce.report-id	Salesforce1 分析报告 Id.		字符串
camel.component.salesforce.report-metadata	Salesforce1 分析报告元数据进行过滤。选项是一个 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
camel.component.salesforce.s-object-id-value	SObject 外部 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

Name	描述	默认值	类型
camel.component.salesforce.ssl-context-parameters	要使用的 SSL 参数，请参阅 SSLContextParameters 类以了解所有可用选项。选项是 org.apache.camel.support.jsse.SSLContextParameters 类型。		SSLContextParameters
camel.component.salesforce.update-topic	在使用流 API 时是否更新现有的 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。使用密码 OAuth 流很容易入门，但通常要避免它的安全性，这比其它流安全。		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 响应的线程池大小。	10	整数

第 55 章 SAP 组件

SAP 组件是由十个不同 SAP 组件组成的软件包。有远程功能调用(RFC)组件支持 sRFC、tRFC 和 qRFC 协议，并且有 IDoc 组件可使用 IDoc 格式的消息进行通信。组件使用 SAP Java Connector (SAP JCo)库来实现与 SAP 和 SAP IDoc 库双向通信，以中间文档(IDoc)格式传输文档。

55.1. 依赖项

将以下依赖项添加到此组件的 pom.xml 中：

```
<dependency>
  <groupId>org.fusesource</groupId>
  <artifactId>camel-sap-starter</artifactId>
  <version>3.20.1.redhat-00050</version>
</dependency>
```

55.1.1. SAP 组件的额外平台限制

由于 SAP 组件依赖于第三方 JCo 3 和 IDoc 3 库，所以只能在这些库支持的平台上安装它。

55.1.2. SAP JCo 和 SAP IDoc 库

使用 SAP 组件的先决条件是 SAP Java Connector (SAP JCo)库和 SAP IDoc 库被安装到 Java 运行时的 lib/ 目录中。您必须确保从 SAP Service Marketplace 为目标操作系统下载一组适当的 SAP 库。

库文件的名称因目标操作系统而异，如下所示。

表 55.1. 所需的 SAP 库

SAP 组件	Linux 和 UNIX	Windows
SAP JCo 3	**sapjco3.jar** **libsapjco3.so**	**sapjco3.jar** **sapjco3.dll**
SAP IDoc	**sapidoc3.jar**	**sapidoc3.jar**

55.2. URI 格式

SAP 组件提供了两种不同类型的端点：远程功能调用(RFC)端点和中间文档(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]
```

由 sap-endpointKind-destination 前缀的 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命名和配置。

以 sap-endpointKind-server 前缀的 URI 格式用于定义服务器端点（换句话说，Camel 使用者端点）和 serverName 是来自 SAP 实例的特定入站连接的名称。入站连接在组件级别命名和配置。

RFC 端点 URI 的其他组件如下：

rfcName

（必需） 在目的地端点 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 实例调用时端点处理的 RFC 名称。

queueName

指定此端点向发送 SAP 请求的队列。

IDoc 端点 URI 的其他组件如下：

idocType

（必需） 指定此端点生成的 IDoc 类型的基本 IDoc 类型。

idocTypeExtension

指定此端点生成的 IDoc Type 扩展（若有）。

systemRelease

指定这个端点生成的 IDoc 的相关 SAP Basis 发行版本（若有）。

applicationRelease

指定此端点生成的 IDoc 的关联应用程序发行版本（若有）。

queueName

指定此端点向发送 SAP 请求的队列。

55.2.1. 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

55.2.2. 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**，请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。

55.2.3. 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**，指定此端点会将异常传播到 SAP 中的调用者，而不是交换的异常处理程序。

55.2.4. RFC 和 IDoc 端点摘要

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 SAP 系统发出请求和响应时，应使用此端点。

注意

此组件使用的 sRFC 协议向 SAP 系统提供请求和响应，并 具有最佳工作量。如果在发送请求时出现通信错误，接收 SAP 系统中的远程功能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.在大多数情况下，当请求必须发送到接收 SAP 系统时，应使用此端点。为达到此目的，组件会生成一个事务 ID，它包括了通过路由交换中组件发送的每个请求。接收 SAP 系统在 发送请求 前记录附带的请求；如果 SAP 系统再次收到请求，则其 tid 不会提供请求。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内发送请求，了解它将只会被发送并执行一次。

注意

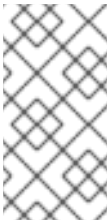
此组件使用的 tRFC 协议是异步的，不会返回响应。因此，此组件的端点不会返回响应消息。

注意

此组件不能保证通过端点对一系列请求的顺序，并且这些请求的交付和执行顺序因通信错误和重新发送请求而不同。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。此组件通过添加保证通过端点发送请求来扩展事务远程功能调用目的地 camel 组件的功能。如果一系列请求相互依赖，并且最多必须一次性 传送到接收 SAP 系统时，应使用此端点。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。排序保证是通过按 SAP 系统收到的顺序对 入站队列 进行序列化来实现的。入站队列由 SAP 中的 QIN 调度程序 处理。激活 入站队列时，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，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 系统的请求和响应时，应使用此组件及其端点。

sap-trfc-server

Camel SAP Transactional Remote Function Call Server Camel 组件.当发送 SAP 系统 最多将其请求发送到 Camel 路由时，应使用此端点。为达到此目的，发送 SAP 系统会生成一个事务 ID tid，它为每个请求发送给组件的端点。发送 SAP 系统首先会检查组件是否收到给定的 tid，然后再发送与 tid 关联的一系列请求。组件将检查它维护的 tid的列表，如果发送的 tid 不在该列表中，则记录

发送的 tid，然后响应发送的 SAP 系统，指示 tid 是否已被记录。如果之前未记录 tid，发送 SAP 系统才会发送一系列请求。这可让发送 SAP 系统可靠地向 camel 路由发送一系列请求。

sap-idoc-destination

Camel SAP IDoc Destination Camel 组件.当 Camel 路由向 SAP 系统发送中间文档(IDocs)列表时，应使用此端点。

sap-idoclist-destination

Camel SAP IDoc List Destination Camel 组件.当 Camel 路由向 SAP 系统发送中间文档(IDocs)列表列表时，应使用此端点。

sap-qidoc-destination

Camel SAP Queued IDoc Destination Camel 组件.当需要 Camel 路由按顺序向 SAP 系统发送中间文档(IDocs)列表时，应使用此组件及其端点。

sap-qidoclist-destination

Camel SAP Queued IDoc List Destination Camel 组件。当 camel 路由将 Intermediate 文档(IDocs)列表按顺序发送到 SAP 系统时，会使用这个组件及其端点。

sap-idoclist-server

Camel SAP IDoc List Server Camel 组件.当发送 SAP 系统需要向 Camel 路由发送中间文档列表时，应使用此端点。这个组件使用 tRFC 协议与 SAP 通信，如 sap-trfc-server-standalone 快速启动中所述。

55.2.5. SAP RFC 目标端点

RFC 目标端点支持到 SAP 的出站通信，使这些端点能够向 SAP 中的 ABAP 功能模块发出 RFC 调用。RFC 目标端点被配置为通过到 SAP 实例的特定连接对特定的 ABAP 函数发出 RFC 调用。RFC 目的地是用于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 IN-OUT 交换的输入消息中提取 RFC 请求，并在函数调用中分配该请求到 SAP。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，RFC 目标端点仅支持创建制作者。

55.2.6. SAP RFC 服务器端点

RFC 服务器端点支持 SAP 的入站通信，它允许 SAP 中的 ABAP 应用程序向服务器端点发出 RFC 调用。ABAP 应用程序与 RFC 服务器端点交互，就像它是远程功能模块一样。RFC 服务器端点被配置为通过 SAP 实例的特定连接接收对特定 RFC 函数的 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。交换的输出消息将返回为 RFC 调用的响应。由于 SAP RFC 服务器端点仅支持入站通信，RFC 服务器端点仅支持创建消费者。

55.2.7. SAP IDoc 和 IDoc 列表目的地端点

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，可与非 SAP 系统轻松交换。IDoc 目的地由一组连接参数指定，称为 目标数据。

IDoc list 目标端点与 IDoc 目标端点类似，但它处理的消息由 IDoc 文档列表组成。

55.2.8. 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 文档的列表。IDoc list 服务器由一组连接参数指定，称为 服务器数据。

55.2.9. 元数据软件仓库

元数据存储库用于存储以下类型的元数据：

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类型安全传输，然后再分配这些调用。存储库填充了存储库数据。Repository data 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，从功能模块传递，并且具有它描述的功能模块的唯一名称。

IDoc 类型描述

> IDoc 运行时使用此元数据来确保在发送到通信合作伙伴前正确格式化 IDoc 文档。基本 IDoc 类型由一个名称、允许的片段列表以及片段之间的分层关系描述组成。一些额外的限制可以在片段上实施：网段可以是强制或可选；可以为每个片段指定最小/最大范围（定义该片段的重复数量）。

因此，SAP 目的地和服务器端点需要访问存储库，以便发送和接收 RFC 调用并发送和接收 IDoc 文档。对于 RFC 调用，调用并由端点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，端点处理的所有 IDoc 类型和 IDoc 类型扩展的元数据必须位于存储库中。目的地和服务器端点使用的仓库位置在目标数据和相应连接的服务器数据中指定。

如果是 SAP 目标端点，它使用的存储库通常位于 SAP 系统中，默认为它连接的 SAP 系统。这个默认值需要在目标数据中明确配置。此外，目标端点进行的远程功能调用的元数据将已存在于存储库中，用于

它调用的任何现有功能模块。因此，目标端点发出的调用元数据不需要 SAP 组件中的配置。

另一方面，由服务器端点处理的功能调用的元数据通常不在 SAP 系统的软件仓库中，必须由位于 SAP 组件的存储库提供。SAP 组件维护命名元数据存储库的映射。存储库的名称与提供元数据的服务器名称对应。

55.3. CONFIGURATION

SAP 组件维护三个映射来存储目标数据、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，它会自动注入到 SAP 组件（在蓝图 XML 配置或 Spring XML 配置文件的上下文）。存储库数据存储 必须直接在相关的 SAP 组件上配置。

55.3.1. 配置概述

SAP 组件维护三个映射来存储目标数据、服务器数据和存储库数据。组件的属性 destinationDataStore 来存储目标名称的密钥，属性 serverDataStore，存储由服务器名称密钥的服务器数据，以及属性 repositoryDataStore，存储由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蓝图 XML 文件中配置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>
```

55.3.2. 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配置与 SAP 实例的唯一出站连接。每个条目的 key 是出站连接的名称，用于目的地端点 URI 的 destinationName 组件，如 URI 格式部分所述。

每个条目的值是目标数据配置对象 - 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，在按照上述蓝图 XML 文件配置目的地后，您可以使用以下 URI 在 quickstartDest 目的地上调用 BAPI_FLCUST_GETLIST 远程功能调用：

```
sap-srfc-destination:quickstartDest:BAPI_FLCUST_GETLIST
```

55.3.2.1. tRFC 和 qRFC 目的地的拦截器

前面的目的地配置示例显示 CurrentProcessorDefinitionInterceptStrategy 对象的实例化。此对象在 Camel 运行时安装拦截器，它允许 Camel SAP 组件在处理 RFC 事务时跟踪 Camel 路由中的位置。

重要

对于事务的 RFC 目标端点（如 sap-trfc-destination 和 sap-qrfc-destination），且必须在 Camel 运行时中安装，以便正确管理出站 RFC 通信。如果在运行时找不到策略，则 Destination RFC Transaction Handlers 会在 Camel 日志中发出警告。在这种情况下，需要重新置备并重启 Camel 运行时来正确地管理出站事务的 RFC 通信。

55.3.2.2. 登录和验证选项

下表列出了用于在 SAP 目标数据存储中配置目的地的 日志和身份验证选项 ：

Name	默认值	描述
client		SAP 客户端，在参数上强制的日志。

user		登录 user，在 参数上登录基于密码的身份验证。
aliasUser		可以使用登录用户别名，而不是登录用户。
userId		用于登录到 ABAP AS 的用户身份。JCo 运行时使用，如果目标配置使用 SSO/assertion ticket、证书、当前用户 , 或 SNC 环境进行身份验证。如果未设置用户和用户别名，则用户 ID 是必须的。此 ID 永远不会发送到 SAP 后端，它将在本地由 JCo 运行时使用。
passwd		登录 password，在 参数上登录基于密码的身份验证。
lang		如果未定义，在语言中登录，则使用默认用户语言。
mysapsso2		使用指定的 SAP Cookie Version 2 作为基于 SSO 的身份验证的票据上的日志。
x509cert		使用指定的 X509 证书进行基于证书的身份验证。
lcheck		向第一次调用 - 1（启用）发布身份验证。只用于特殊情况。
useSapGui		使用可见、隐藏或者不使用 SAP GUI
codePage		参数上的其他日志，以定义用于转换参数日志的代码页面。只用于特殊情况。
getsso2		在登录后对 SSO 票据进行排序，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**）。

55.3.2.3. 连接选项

下表列出了用于在 SAP 目标数据存储中配置目的地的连接选项：

Name	默认值	描述
saprouter		用于连接到 SAP 路由器后面的系统的 SAP 路由器字符串。SAP 路由器字符串包含 SAP 路由器的链及其端口号，并具有以下形式：**(/H/<host>[/S/<port>])+**。
sysnr		SAP ABAP 应用服务器的系统号，对于直接连接是必需的。
ashost		SAP ABAP 应用服务器，用于直接连接。
mshost		SAP 消息服务器，用于负载平衡连接的必要属性。
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，可在 **etc/services** 中查找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。
gwhost		允许指定协调网关，该网关应用于建立到应用服务器的连接。如果没有指定，则使用应用服务器上的网关。
gwserv		应在使用 gwhost 时设置。允许指定该网关中使用的端口。如果没有指定，则使用应用服务器上的网关端口。为了解析服务名称 sapgwXXX 中，通过操作系统的网络层执行 etc/services 的查找。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，用于负载平衡连接的必要属性。
group		SAP 应用服务器组，用于负载平衡连接的必要属性。

network	**LAN**	根据 JCo 和目标系统之间的网络质量来设置这个值，以优化性能。有效值为 **LAN** 或 **WAN**（仅适用于快速序列化）。如果将 **网络配置选项** 设置为 **WAN**，则会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**，则会使用非常快速的压缩算法，且数据分析仅在非常基本的级别执行。当您设置 **LAN** 选项时，压缩比率效率不低，但网络传输时间被视为不显著。默认设置为 **LAN**。
serializationFormat	**基于行**	有效值为 line **Based** 或 **columnBased**。对于快速序列化 **列**，必须设置。默认序列化设置为 **基于行**。

55.3.2.4. 连接池选项

下表列出了用于在 SAP 目标数据存储中配置目的地的 连接池 选项：

Name	默认值	描述
peakLimit	**0**	可以为目标同时创建的活跃出站连接的最大数量。值 **0** 允许无限数量的活跃连接。否则，如果值小于 **jpoolCapacity** 的值，它会自动增加到这个值。默认设置为 **poolCapacity** 的值，或者 **poolCapacity** 也未指定，则默认值为 **0 (** 无限)。
poolCapacity	**1**	目的地保持打开的最大空闲出站连接数。值 **0** 具有没有连接池（默认为 **1**）的影响。
expirationTime		在经过目标内部保存的空闲连接后，可以关闭的时间（毫秒）。
expirationPeriod		目标检查发布的连接过期的时间（毫秒）。
maxGetTime		如果应用程序已经分配了最大允许的连接数，等待连接的最大时间（毫秒）。

55.3.2.5. 安全网络连接选项

下表列出了用于在 SAP 目标数据存储中配置目的地 的安全网络选项 ：

Name	默认值	描述
sncMode		安全网络连接(SNC)模式、**0 (** 关闭)或 **1 (** on)。
sncPartnername		SNC 合作伙伴，例如：**p:CN=R3, O=XYZ-INC, C=EN**。
sncQop		SNC 安全级别： **1** 到 **9**。
sncMyname		本身 SNC 名称。覆盖环境设置。
sncLibrary		提供 SNC 服务的库路径。

55.3.2.6. 仓库选项

下表列出了用于在 SAP 目标数据存储中配置目的地 的存储库 选项：

Name	默认值	描述
repositoryDest		指定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。这可让您使用其他用户进行存储库查找。
repositoryPasswd		存储库用户的密码。必需，如果使用存储库用户。
repositorySnc		**（可选）** 如果将 SNC 用于此目的地，则可以为存储库连接关闭它（如果此属性设为 **0** ）。默认设置为 **jco.client.snc**_mode 的值。仅适用于特殊情况。

repositoryRoundtripOptimization		启用 **RFC_METADATA_GET** API，该 API 在单个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 系统中的 **RFC_METADATA_GET**。 如果没有设置属性，则目的地最初执行远程调用来检查 **RFC_METADATA_GET** 是否可用。如果可用，则目的地将使用它。 **注：** 如果存储库已经初始化（例如，由于某些其他目的地使用），此属性没有任何效果。通常，此属性与 ABAP 系统相关，在所有目的地上应具有相同的值，指向同一 ABAP 系统。有关后端先决条件，请参阅备注 1456826。

55.3.2.7. 跟踪配置选项

下表列出了用于在 SAP 目标数据存储中配置目的地的 trace 配置选项：

Name	默认值	描述
trace		启用/禁用 RFC 跟踪(**0** 或 **1**)。
cpicTrace		启用/禁用 CPIC trace **[0..3]**。

55.3.3. 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配置与 SAP 实例不同的入站连接。每个条目的 key 是出站连接的名称，用于服务器端点 URI 的 serverName 组件，如 URI 格式部分所述。

每个条目的值是 服务器数据 配置对象org.fusesource.camel.component.sap.model.rfc.impl.ServerDataImpl，它定义入站 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，本例如何配置目标连接 quickstartDest，服务器用来从远程 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，在按照上述蓝图 XML 文件配置目的地后，您可以使用以下 URI 处理来自调用客户端的 BAPI_FLCUST_GETLIST 远程功能调用：

> *sap-srfc-server:quickstartServer:BAPI_FLCUST_GETLIST*

55.3.3.1. 所需选项

服务器数据配置对象所需的选项如下：

Name	默认值	描述
gwhost		应该在其上注册服务器的网关主机。
gwserv		网关服务，这是可以在其上进行注册的端口。为了解析服务名称 **sapgwXXX**，**etc/services** 的查找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		注册的程序 ID。作为网关和 ABAP 系统中目的地上的标识符。
repositoryDestination		指定服务器可以使用的目的地名称，以便从托管在远程 SAP 服务器中的元数据存储库检索元数据。
connectionCount		应该在网关上注册的连接数。

55.3.3.2. 安全网络连接选项

服务器数据配置对象的安全连接选项如下：

Name	默认值	描述
sncMode		安全网络连接(SNC)模式、**0 (** 关闭)或 **1 (** on)。
sncQop		SNC 安全性级别，**1** 到 **9**。

sncMyname		服务器的 SNC 名称。覆盖默认 SNC 名称。通常，类似 **p:CN=JCoServer, O=ACompany, C=EN**.
sncLib		提供 SNC 服务的库路径。如果没有提供此属性，则使用 **jco.middleware.snc_lib** 属性的值。

55.3.3.3. 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，因此只能在 ABAP 系统的网关上注册服务器时，它才可以通过 SAProuter 访问。典型的路由器字符串为 **/H/firewall.hostname/H/**。
maxStartupDelay		失败时两次启动尝试之间的最长时间（以秒为单位）。在每次启动失败后，等待时间从最初 1 秒加倍，直到达到最大值，或者可以成功启动服务器。
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**。最多线程数量不能超过 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**。

55.3.4. 仓库配置

存储库的配置在 SAP 组件的 repositoryDataStore 属性中维护。此映射中的每个条目都配置不同的存储库。每个条目的 key 是存储库的名称，此密钥也对应于附加此存储库的服务器的名称。

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.fuesource.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，每个条目的键则是指定的 **function** 模块的名称。

仓库数据示例

以下代码演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>
```

55.3.4.1. 功能模板属性

功能模块的接口由四个参数列表组成，这些列表将数据传输回 **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 **RFC** 调用中传输的命名参数。支持以下参数列表和异常列表：

- **import** 参数列表包含发送到 **RFC** 调用中功能模块的参数值 ；
- **export** 参数列表包含 **RFC** 调用中 **function** 模块返回的参数值 ；

- *change 参数列表包含 RFC 调用中函数模块发送和返回的参数值 ；*
- *table 参数列表 包含 RFC 调用中函数模块发送和返回的内部表值。*
- *功能模块的接口也由 ABAP 异常列表组成， 该异常列表 可在 RFC 调用中调用模块时引发。*

功能模板描述了功能接口的每个参数列表中的参数的名称和类型，函数会抛出 ABAP 异常。功能模板对象维护五个元数据对象列表，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** 的列表。指定 RFC 调用中向 function 模块发送的参数。
changingParameterList	列表字段元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.ListFieldMeataDataImpl** 的列表。指定 RFC 调用中向函数模块和返回的参数。
exportParameterList	列表字段元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.ListFieldMeataDataImpl** 的列表。指定来自 function 模块的 RFC 调用返回的参数。
tableParameterList	列表字段元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.ListFieldMeataDataImpl** 的列表。指定 RFC 调用中向函数模块和返回的表参数。
exceptionList	ABAP 异常元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.AbapExceptionImpl** 的列表。指定 ABAP 异常可能会在 function 模块的 RFC 调用中引发。

功能模板示例

以下示例演示了如何配置功能模板：

```
<bean id="bookFlightFunctionTemplate"
    class="org.fusesource.camel.component.sap.model.rfc.impl.FunctionTemplateImpl">
    <property name="importParameterList">
```

```
        <list>
            ...
        </list>
    </property>
    <property name="changingParameterList">
        <list>
            ...
        </list>
    </property>
    <property name="exportParameterList">
        <list>
            ...
        </list>
    </property>
    <property name="tableParameterList">
        <list>
            ...
        </list>
    </property>
    <property name="exceptionList">
        <list>
            ...
        </list>
    </property>
</bean>
```

55.3.4.2. 列出字段元数据属性

列表字段元数据对象 ***org.fusesource.camel.component.sap.model.rfc.impl.ListFieldMeataDataImpl，指定参数列表中字段的名称和类型。对于元素外参数字段(CHAR、DATE、DECF34、TIME、BYTE、NUM、FLOAT、INTP、INT 1、INT2、DECF16、DECF34、STRING、XSTRING)，下表列出了可以在列表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	-	非统一代码布局的字段长度（以字节为单位）。这个值取决于参数类型。
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 需要此项。

optional	**false**	如果为 **true**，该字段是可选的，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 name,type,byteLength, 和 unicodeByteLength 属性。此外， BCD、FLOAT、DECF16 和 DECF34 字段要求在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， 下表列出了可以在列表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。
recordMetaData	-	结构或表的元数据。记录元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.RecordMetaDataImpl** 传递，以指定结构或表行中的字段。
optional	**false**	如果为 **true**，该字段是可选的，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了名为 CONNINFO 所需的 TABLE 参数，并带有 connectionInfo 记录元数据对象指定的行结构：

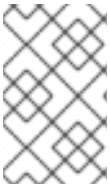
```
<bean    class="org.fusesource.camel.component.sap.model.rfc.impl.ListFieldMetaDataImpl">
    <property name="name" value="CONNINFO" />
    <property name="type" value="TABLE" />
    <property name="recordMetaData" ref="connectionInfo" />
</bean>
```

55.3.4.3. 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 STRUCTURE 或 TABLE 参数行的名称和内容。记录元数据对象维护记录字段元数据对象 org.fusesource.camel.component.sap.model.rfc.impl.FieldMetaDataImpl 的列表，用于指定嵌套结构或表行中的参数。

下表列出了可以在记录元数据对象上设置的配置属性：

Name	默认值	描述
name	-	记录的名称。
recordFieldMetaData	-	记录字段元数据对象列表 **org.fusesource.camel.component.sap.model.rfc.impl.FieldMetaDataImpl**。指定结构中包含的字段。



注意

需要记录元数据对象的所有属性。

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>
```

55.3.4.4. 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，但必须额外指定嵌套结构或表行中的单个字段位置的偏移。单个字段的非统一代码和 **Unicode** 偏移必须计算并从前一字段的非统一代码和 **Unicode** 字节长度总和来计算和指定。

注意

正确指定嵌套结构和表行中的字段偏移会导致底层 **JCo** 和 **ABAP** 运行时中的参数存储重叠并防止 **RFC** 调用中的正确传输值。

对于元素外参数字段(**CHAR**、**DATE**、**DECF34**、**TIME**、**BYTE**、**NUM**、**FLOAT**、**INTP**、**INT1**、**INT2**、**DECF16**、**DECF34**、**STRING**、**XSTRING**)，下表列出了可以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	-	非统一代码布局的字段长度（以字节为单位）。这个值取决于参数类型。
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）。这个偏移是保护结构中字段的字节位置。
unicodeByteOffset	-	Unicode 布局以字节为单位的字段偏移。这个偏移是保护结构中字段的字节位置。
十进制	**0**	字段值中的十进制数；参数类型 **BCD** 和 **FLOAT** 只需要。

对于类型为 TABLE 或 STRUCTURE 的复杂参数字段，下表列出了可以在记录字段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）。这个偏移是保护结构中字段的字节位置。
unicodeByteOffset	-	Unicode 布局以字节为单位的字段偏移。这个偏移是保护结构中字段的字节位置。
recordMetaData	-	结构或表的元数据。记录元数据对象 **org.fusesource.camel.component.sap.model.rfc.impl.RecordMetaDataImpl** 传递，以指定结构或表行中的字段。

Elementary 记录字段元数据示例

以下元数据配置在非Unicode 布局时指定名为 ARR DATE 的 DATE 字段参数，它位于 85 字节中，在 Unicode 布局的情况下，将 170 字节到 enclosing 结构中。

```
<bean    class="org.fusesource.camel.component.sap.model.rfc.impl.FieldMetaDataImpl">
   <property name="name" value="ARRDATE" />
   <property name="type" value="DATE" />
   <property name="byteLength" value="8" />
   <property name="unicodeByteLength" value="16" />
```

```
    <property name="byteOffset" value="85" />
    <property name="unicodeByteOffset" value="170" />
</bean>
```

复杂的记录字段元数据示例

以下元数据配置指定了名为 FLTINFO 的 STRUCTURE 字段参数，其结构由 flightInfo 记录元数据对象指定。参数位于非统一代码和 Unicode 布局的情况下的保护结构的开头。

```
<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>
```

55.4. 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	处理消息的最后目标端点的目标名称。
CamelSap.serverName	处理消息的最后服务器端点的服务器名称。
CamelSap.queueName	处理消息的最后排队端点的队列名称。

CamelSap.rfcName	处理消息的最后 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	IDoc 类型扩展（若有）用于处理消息的最后一个 IDoc 端点。
CamelSap.systemRelease	系统发行版本（若有）是最后一个 IDoc 端点来处理消息。
CamelSap.applicationRelease	应用程序发行版本（若有）是最后一个 IDoc 端点来处理消息。

55.5. 交换属性

SAP 组件添加以下交换属性：

属性	描述
CamelSap.destinationPropertiesMap	包含交换遇到的每个 SAP 目标属性的映射。该映射通过目的地名称的键，每个条目都是包含该目的地配置属性的 **java.util.Properties** 对象。
CamelSap.serverPropertiesMap	包含交换遇到的每个 SAP 服务器属性的映射。该映射由服务器名称密钥，每个条目都是包含该服务器配置属性的 **java.util.Properties** 对象。

55.6. RFC 的消息正文

55.6.1. 请求和响应对象

SAP 端点希望收到含有 SAP 请求对象的消息正文，并将返回一个含有 SAP 响应对象的消息正文。SAP 请求和响应是固定的映射数据结构，其中包含命名字段，每个字段都有预定义的数据类型。

请注意，SAP 请求和响应中的命名字段特定于 SAP 端点，每个端点在 SAP 请求中定义参数并响应它。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;
    ...
}
```

55.6.2. 结构对象

SAP 请求和响应对象都作为结构对象在 Java 中表示，它支持 org.fusesource.camel.component.sap.model.rfc.Structure 接口。此接口扩展了 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，结构接口提供了一个类型限制的方法来检索字段值。

结构对象使用 Eclipse Modeling Framework (EMF)在组件运行时实施，并支持框架的 EObject 接口。结构对象的实例已附加元数据，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。详情请参考 EMF 文档。

注意

尝试获取在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、STRUCTURE 和 TABLE 的值的字段。

注意

创建这些类型的实例并将其添加到结构中是必需的。在以保护结构访问时，根据需要创建这些字段值的实例。

55.6.3. 字段类型

驻留在 SAP 请求或响应的结构对象中的字段可以是 元素 或复杂 的字段。元素字段包含一个 scalar 值，而复杂字段将包含一个或多个元素或复杂类型的字段。

55.6.3.1. 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。下表总结了可能驻留在结构对象中的元素字段类型：

字段类型	对应的 Java 类型	字节长度	Unicode byte Length	number Decimals Digits	描述
CHAR	**java.lang.String**	1 到 65535	1 到 65535	-	ABAP 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': 打包的 BCD 号。BCD 号包含每个字节的两个数字。
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 bytes 数组
NUM	**java.lang.String**	1 到 65535	1 到 65535	-	ABAP Type 'N': Fixed sized number string
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 字节整数
INT2	**java.lang.Integer**	2	2	-	ABAP Type 'S': 2 字节整数
INT1	**java.lang.Integer**	1	1	-	ABAP Type 'B': 1 字节整数

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 字节 Decimal 浮动点号
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

55.6.3.2. 字符字段类型

字符字段包含一个固定大小字符字符串，可在底层 JCo 和 ABAP 运行时中使用非统一代码或 Unicode 字符编码。非统一字符字符串按字节编码一个字符。Unicode 字符字符串使用 UTF-16 编码来编码两个字节。字符字段值在 Java 中以 java.lang.String 对象表示，底层 JCo 运行时负责转换为其 ABAP 表示。

字符字段在其关联的 字节长度和 unicodeByteLength 属性中声明其字段长度，该属性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 字符字段是一个 8 个字符日期字段，其中年、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，格式为 HHMMSS，对应于 ABAP 类型 T。

55.6.3.3. 数字字段类型

number 字段包含一个数字。支持以下数字字段类型：

INT

INT number 字段是一个整数字段，作为 4 字节整数值存储在底层 JCo 和 ABAP 运行时，对应于 ABAP 类型 I。INT 字段值以 java.lang.Integer 对象的形式表示。

INT2

INT2 数字字段是作为 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。INT2 字段值以 java.lang.Integer 对象的形式表示。

INT1

INT1 字段是一个整数字段，作为 1 字节整数值存储在底层 JCo 和 ABAP 运行时值中，对应于 ABAP 类型 B。INT1 字段值以 java.lang.Integer 对象的形式表示。

浮点值

FLOAT 字段是一个二进制浮动点号字段，存储在底层 JCo 和 ABAP 运行时中的 8 字节双值，对应于 ABAP 类型 F。FLOAT 字段声明字段的值包含在其关联的十进制属性中。如果是 FLOAT 字段，这个十进制属性可在 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，存储在底层 JCo 和 ABAP 运行时中的 1 到 16 字节数字，对应于 ABAP 类型 P。压缩的数字会为每个字节存储两个十进制数字。BCD 字段在其关联的 字节长度和 unicodeByteLength 属性中声明其字段长度。如果是 BCD 字段，这些属性可以在 1 到 16 字节之间具有值，并且这两个属性都具有相同的值。BCD 字段声明字段的值包含在其关联的十进制属性中的十进制数字。如果是 BCD 字段，这个十进制属性可在 1 到 14 位之间具有值。BCD 字段值在 Java 中表示为 java.math.BigDecimal。

DECF16

DECF16 字段是一个十进制浮动点，存储为底层 JCo 和 ABAP 运行时的 8 字节 IEEE 754 十进制浮动点值，对应于 ABAP 类型 decfloat16。DECF16 字段的值有 16 十进制数字。DECF16 字段的值在 Java 中表示为 java.math.BigDecimal。

DECF34

DECF34 字段是一个十进制浮动点，存储为底层 JCo 和 ABAP 运行时的 16 字节 IEEE 754 十进制浮动点值，对应于 ABAP 类型 decfloat34。DECF34 字段的值具有 34 十进制数字。DECF34 字段

的值在 Java 中表示为 java.math.BigDecimal。

55.6.3.4. 十六进制字段类型

hex 字段包含原始二进制数据。支持以下十六进制字段类型：

BYTE

BYTE 字段是一个固定大小的字符串，作为字节数组存储在底层 JCo 和 ABAP 运行时，对应于 ABAP 类型 X。BYTE 字段在其关联 字节Length 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，并且这两个属性都具有相同的值。BYTE 字段的值在 Java 中以 byte[] 对象表示。

55.6.3.5. 字符串字段类型

字符串字段引用变量值。在运行时前，该字符串值的长度不会被修复。字符串值的存储在底层 JCo 和 ABAP 运行时动态创建。字符串字段本身的存储已被修复，仅包含字符串标头。

字符串

STRING 字段引用存储在底层 JCo 和 ABAP 运行时中的字符字符串作为 8 字节值。它对应于 ABAP 类型 G。STRING 字段的值在 Java 中以 java.lang.String 对象表示。

XSTRING

XSTRING 字段引用存储在底层 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 中以 byte[] 对象表示。

55.6.3.6. 复杂字段类型

复杂的字段可以是 structure 或 table 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 字节长度总数	-	ABAP Type 'u' 和 'v': Heterogeneous Structure

表	**org.fusesource.camel.component.sap.model.rfc.Table**	行结构的字节长度	行结构的 Unicode 字节长度	-	ABAP Type 'h': Table

55.6.3.7. struct 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。它对应于 ABAP 类型 u 或 v。STRUCTURE 字段的值表示在 Java 中，作为带有接口 org.fusesource.camel.component.sap.model.rfc.Structure 的结构对象。

55.6.3.8. 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。字段的值在 Java 中由一个带有接口 org.fusesource.camel.component.sap.model.rfc.Table 的表对象表示。

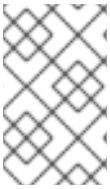
55.6.3.9. 表对象

表对象是一个 homogeneous 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接口中定义的标准方法访问和管理。此外，表接口提供了两种工厂方法，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)在组件运行时实施，并支持框架的 EObject 接口。表对象的实例已附加元数据，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。详情请参考 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。

55.7. IDOC 的消息正文

55.7.1. IDoc 消息类型

当使用 IDoc Camel SAP 端点之一时，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，消息正文为 Document 类型：

```
org.fusesource.camel.component.sap.model.idoc.Document
```

对于 sap-idoclist-destination 端点、sap-qidoclist-destination 端点或 sap-idoclist-server 端点，消息正文为 DocumentList 类型：

```
org.fusesource.camel.component.sap.model.idoc.DocumentList
```

55.7.2. IDoc 文档模型

对于 Camel SAP 组件，IDoc 文档使用 Eclipse Modeling Framework (EMF)进行建模，它提供了一个围绕底层 SAP IDoc API 的 wrapper 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。另外，文档 界面会公开以下方法：

```
// Java
package org.fusesource.camel.component.sap.model.idoc;
...
public interface Document extends EObject {
    // Access the field values from the IDoc control record
    String getArchiveKey();
```

```
    void setArchiveKey(String value);
    String getClient();
    void setClient(String value);
    ...

    // Access the IDoc document contents
    Segment getRootSegment();
}
```

以下方法通过 文档 界面公开：

访问控制记录的方法

大多数方法是访问或修改 IDoc 控制记录的字段值。这些方法的格式是 AttributeName，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 data 记录)的访问，将内容返回为 Segment 对象。每个 Segment 对象可以包含任意数量的子片段，网段可以嵌套为任意程度。

但请注意，网段层次结构的确切布局由文档的特定 IDoc 类型定义。在创建（或读取）网段层次结构时，您必须遵循 IDoc 类型定义的精确结构。

Segment 类型用于访问 IDoc 文档的数据记录，其中片段会根据文档 IDoc 类型定义的结构更宽松。另外，S eg ment 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) 方法对向片段添加新的（嵌套）子项特别有用。它返回一个类型为 SegmentList 的对象，它定义如下：

```
// Java
package org.fusesource.camel.component.sap.model.idoc;
...
public interface SegmentList<S extends Segment> extends EObject, EList<S> {
    S add();

    S add(int index);
}
```

因此，要创建 E1SCU_CRE 类型的数据记录，您可以使用类似如下的 Java 代码：

```
Segment rootSegment = document.getRootSegment();

Segment E1SCU_CRE_Segment = rootSegment.getChildren("E1SCU_CRE").add();
```

55.7.3. IDoc 与文档对象相关

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 对象的属性表示。

数据记录

数据记录由 Segment 对象表示，这些对象以嵌套片段层次结构组成。您可以通过 Document.getRootSegment 方法访问根片段。

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的 status 属性访问最新的状态值。

创建文档实例的示例

以下示例演示了如何使用 Java 中的 IDoc 模型 API 创建 IDoc 文档 FLCUSTOMER_CREATEFROMDATA01。

例 55.1. 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");
```

55.8. 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 55.2. 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**	创建日期 IDoc
creationTime	6	**CRETIM**	创建时间 IDoc
方向	1	**DIRECT**	方向
eDIMessage	14	**REFMES**	引用消息
eDIMessageGroup	14	**REFGRP**	引用消息组
eDIMessageType	6	**STDMES**	EDI 消息类型
eDIStandardFlag	1	**STD**	EDI 标准

属性	length	SAP 字段	描述
eDIStandardVersion	6	**STDVRS**	EDI 标准版本
eDITransmissionFile	14	**REFINT**	对更改文件的引用
iDocCompoundType	8	**DOCTYP**	IDoc 类型
iDocNumber	16	**DOCNUM**	IDoc number
iDocSAPRelease	4	**DOCREL**	SAP IDoc 发行版本
iDocType	30	**IDOCTP**	基本 IDoc 类型的名称
iDocTypeExtension	30	**CIMTYP**	扩展类型的名称
messageCode	3	**MESCOD**	逻辑消息代码
messageFunction	3	**MESFCT**	逻辑消息功能
messageType	30	**MESTYP**	逻辑消息类型
outputMode	1	**OUTMOD**	输出模式
recipientAddress	10	**RCVSAD**	接收器地址(SADR)
recipientLogicalAddress	70	**RCVLAD**	接收器的逻辑地址
recipientPartnerFunction	2	**RCVPFC**	接收器合作伙伴功能
recipientPartnerNumber	10	**RCVPRN**	合作伙伴的接收器号
recipientPartnerType	2	**RCVPRT**	合作伙伴的接收器类型
recipientPort	10	**RCVPOR**	接收器端口(SAP System、EDI 子系统)
senderAddress		**SNDSAD**	发件人地址(SADR)
senderLogicalAddress	70	**SNDLAD**	发送者的逻辑地址
senderPartnerFunction	2	**SNDPFC**	发件人合作伙伴功能

属性	length	SAP 字段	描述
senderPartnerNumber	10	**SNDPRN**	合作伙伴的发送方号
senderPartnerType	2	**SNDPRT**	合作伙伴的发送方类型
senderPort	10	**SNDPOR**	发件人端口(SAP System、EDI 子系统)
序列化	20	**SERIAL**	EDI/ALE: Serialization 字段
status	2	**状态**	IDoc 的状态
testFlag	1	**测试**	test 标记

55.8.1. 在 Java 中设置文档属性

在 Java 中设置控制记录属性时，将遵循 Java bean 属性的常见约定。也就是说，可以通过 getName 和 setName 方法访问 name 属性，以获取和设置属性值。例如，iDocType、iDocTypeExtension 和 messageType 属性可以在 Document 对象中设置：

```
// Java
document.setIDocType("FLCUSTOMER_CREATEFROMDATA01");
document.setIDocTypeExtension("");
document.setMessageType("FLCUSTOMER_CREATEFROMDATA");
```

55.8.2. 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>
```

55.9. 事务支持

55.9.1. BAPI 事务模型

SAP 组件支持 BAPI 事务模型与 SAP 的出站通信。如有必要，带有将 transacted 选项设置为 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，并在处理消息交换完成后结束有状态会话。如果消息交换的处理失败，同步对象将调用 BAPI 服务器方法 BAPI_TRANSACTION_ROLLBACK 并结束有状态会话。

55.9.2. 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 发送和处理保证。协议中发送的每个请求的 TID。使用 tRFC 协议发送应用程序必须在发送请求时识别具有唯一 TID 的请求的每个实例。应用程序可能会多次发送给定 TID 的请求，但协议确保请求在接收系统中发送并处理最多一次。在发送请求时，应用程序可以选择在遇到通信或系统错误时重新发送给定 TID 的请求，因此请不确定，无论该请求是否在接收系统中发送和处理。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 发送和处理其请求的保证。

55.9.3. 要使用哪些事务模型？

BAPI 事务是一种应用程序级别的事务，它对 SAP 数据库中 BAPI 方法或 RFC 功能采取的持久性数据更改实施 ACID 保证。RFC 事务是一种通信事务，在请求 BAPI 方法和/或 RFC 功能时，它会对交付保证(AT-MOST-ONCE、EXACTLY-ONCE-IN-ORDER)实施交付保证(AT-MOST-ONCE、EXACTLY-ONCE、EXACTLY-ONCE、EXACTLY-ONCE-IN-ORDER)。

55.9.4. 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，甚至多次发送消息。这意味着 Camel SAP 组件可能需要跟踪每个 交换 对象通过路由传递的事务 ID (TID)。现在，如果路由处理失败且必须重试，这种情况会非常复杂。RFC 事务语义要求，每个 RFC 目的地都必须使用第一次使用的相同 TID 调用（以及每个目的地的 TID 相互不同）。换句话说，Camel SAP 组件必须跟踪在路由中使用的 TID，并记住此信息，以便以正确顺序重新执行 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。此拦截器必须安装到 Camel 运行时中，以便 Camel SAP 组件跟踪路由中的 TID。

55.9.5. 事务 RFC 服务器端点

以下服务器端点支持 RFC 事务：

- sap-trfc-server

当 Camel 交换处理事务请求遇到处理错误时，Camel 通过其标准错误处理机制处理处理错误。如果 Camel 路由处理将交换配置为将错误传播到调用者，则启动交换的 SAP 服务器端点会记录失败，发送 SAP 系统会收到错误。然后，发送 SAP 系统可以通过发送具有相同 TID 的另一个事务请求来再次处理请求。

55.10. RFC 的 XML 序列化

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到 XML 文档。

55.10.1. XML 命名空间

存储库中的每个 RFC 为构成 Request 和 Response 对象的序列化形式的元素定义特定的 XML 命名空间。此命名空间 URL 的形式如下：

```
http://sap.fusesource.org/rfc/<Repository Name>/<RFC Name>
```

RFC 命名空间 URL 有一个通用的 http://sap.fusesource.org/rfc 前缀，后跟定义 RFC 元数据的存储库名称。URL 中的最终组件是 RFC 本身的名称。

55.10.2. 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 root 元素，并由请求的 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 响应对象将序列化到 XML 文档中，其中包含名为 Response 的 root 元素，并由响应的 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>
```

55.10.3. 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所属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 元素的类型名称将对应于定义结构的记录元数据对象的名称，如下例所示：

```
<xs:schema
    targetNamespace="http://sap.fusesource.org/rfc/nplServer/BOOK_FLIGHT">
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
   ...
   <xs:complexType name="FLTINFO_STRUCTURE”>
   ...
   </xs:complexType>
   ...
</xs:schema>
```

当将 JAXB bean 设置为 marshal 和 unmarshal 时，这种区别非常重要。

55.10.4. 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所属参数列表、结构或表行条目中的表字段名称。table 元素将包含一系列行元素，以存放表行条目的序列化值。

```
<BOOK_FLIGHT:CONNINFO
    xmlns:BOOK_FLIGHT="http://sap.fusesource.org/rfc/nplServer/BOOK_FLIGHT">
   <row ... > ... </row>
   ...
   <row ... > ... </row>
</BOOK_FLIGHT:CONNINFO>
```

请注意，RFC 命名空间中的表元素的类型名称对应于记录元数据对象的名称，该对象定义了 _TABLE

后缀后缀的表行结构。RFC 名称中的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>
```

当将 JAXB bean 设置为 marshal 和 unmarshal 时，这种区别非常重要。

55.10.5. Elementary 字段

参数列表或嵌套结构中的元素字段序列化为括起参数列表或结构的元素的属性。serialized 字段的属性名称与它所在的参数列表、结构或表行条目中的字段名称对应，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"/>
```

55.10.6. 日期和时间格式

date 和 Time 字段使用以下格式序列化为属性值：

```
yyyy-MM-dd'T'HH:mm:ss.SSSZ
```

日期字段将只按年、月份、天和时区组件集进行序列化：

```
DEPDATE="2014-03-19T00:00:00.000-0400"
```

时间字段将仅使用 hour, minute, second, millisecond 和 timezone 组件进行序列化：

```
DEPTIME="1970-01-01T16:00:00.000-0500"
```

55.11. IDOC 的 XML 序列化

IDoc 消息正文可以序列化为 XML 字符串格式，并提供内置类型转换器的帮助。

55.11.1. XML 命名空间

每个序列化 IDoc 都与 XML 命名空间关联，其通用格式如下：

```
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///
```

55.11.2. 内置类型转换器

Camel SAP 组件有一个内置类型转换器，能够将 Document 对象或 DocumentList 对象转换为 String 类型。

例如，要将 Document 对象序列化到 XML 字符串，您只需在 XML DSL 中的路由中添加以下行：

```
<convertBodyTo type="java.lang.String"/>
```

您还可以使用此方法将 XML 消息序列化到 文档 对象。例如，假设当前消息正文是一个序列化 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"/>
```

55.11.3. 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 元素可以是 idoc:Document （用于单个文档）或 idoc:DocumentList （用于文档列表）。它显示，单个已序列化为 idoc:Document 元素的 IDoc 文档。

例 55.2. 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>
```

55.12. 示例 1：从 SAP 读取数据

本例演示了一个路由，它从 SAP 读取 FlightCustomer Business 对象数据。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。

55.12.1. 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");
```

55.12.2. 用于路由的 XML DSL

和同一路由的 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>
```

55.12.3. createFlightCustomerGetListRequest bean

createFlightCustomerGetListRequest bean 负责使用后续 SAP 端点的 RFC 调用中使用的交换方法构建 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);
}
```

55.12.4. returnFlightCustomerInfo bean

returnFlightCustomerInfo bean 负责使用从之前的 **SAP** 端点接收的交换方法从 **SAP** 响应对象中提取数据。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);

}
```

55.13. 示例 2：向 SAP 写入数据

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。

55.13.1. 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");
```

55.13.2. 用于路由的 XML DSL

和同一路由的 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>
```

55.13.3. 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，端点会在调用 RFC 调用前确保启动 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 中永久保留路由的更改。

55.13.4. 填充请求参数

createFlightTripRequest 和 returnFlightTripResponse Bean 负责将请求参数填充至 SAP 请求中，并相应地从 SAP 响应中提取响应参数，如上例所示。

55.14. 示例 3：处理来自 SAP 的请求

本例演示了一个路由，它处理从 SAP 到 BOOK_FLIGHT RFC 的请求，该请求由路由实施。此外，它演示了组件的 XML 序列化支持，使用 JAXB 来 unmarshal 和 marshal SAP 请求对象，并响应自定义 Bean 的响应对象。

此路由代表批量代理 FlightCustomer，创建一个 FlightTrip 业务对象。路由首先将 SAP 服务器端点收到的 SAP 请求对象取消放入自定义 JAXB bean 中。然后，这个自定义 bean 在交换中多播到三个子路由，该路由收集旋转代理、flight 连接和传递创建动态往返所需的信息。最后的子路由在 SAP 中创建 flight trip 对象，如上例中所示。最后的子路由还会创建并返回自定义 JAXB bean，该 Bean 将置于 SAP 响应对象中，并由服务器端点返回。

55.14.1. 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);
```

55.14.2. 用于路由的 XML DSL

和同一路由的 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>
```

55.14.3. BookFlightRequest bean

以下列表演示了一个 **JAXB bean**，它从 **SAP BOOK_FLIGHT** 请求对象序列化形式取消**marshals**：

```
@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;

    ...
}
```

55.14.4. BookFlightResponse bean

*以下列表演示了一个 **JAXB bean，它遵循 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。

55.14.5. FlightInfo bean

*以下列表演示了一个 **JAXB bean，它采用复杂结构参数 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;

    ...
}
```

55.14.6. ConnectionInfoTable bean

以下列表演示了一个 JAXB bean，它对复杂表参数 CONNINFO 的序列化形式 marshals。请注意，JAXB bean 的 root 元素类型的名称对应于以 _TABLE 后缀后缀的行结构类型的名称，并且 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;

    ...
}
```

55.14.7. ConnectionInfo bean

*以下列表演示了一个 **JAXB bean**，它采用上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 56 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，计时器使用 JDK Timer。

您只能消耗来自此端点的事件。

56.1. URI 格式

```
scheduler:name[?options]
```

其中 name 是调度程序的名称，它在端点之间创建和共享。因此，如果您对所有调度程序端点使用相同的名称，则只使用一个调度程序线程池和线程，但您可以配置线程池以允许更多的并发线程。

注意

生成的交换的 IN 正文为 null。So exchange.getIn().getBody() returns null.

56.2. 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

56.2.1. 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。

56.2.2. 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。

56.3. 组件选项

调度程序组件支持 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
poolSize (scheduler)	调度线程池使用的线程池中内核线程数量。默认使用单个线程。	1	int

56.4. 端点选项

调度程序端点使用 URI 语法进行配置：

```
scheduler:name
```

使用以下路径和查询参数：

56.4.1. 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

56.4.2. 查询参数(21 参数)

Name	描述	默认值	类型
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
sendEmptyMessageWhenIdle (consumer)	如果轮询使用者没有轮询任何文件，您可以启用此选项来发送空消息（无正文）。	false	布尔值
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
Sync (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
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)	调度线程池使用的线程池中内核线程数量。默认使用单个线程。	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

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

56.5. 更多信息

此组件是一个调度程序 轮询 Consumer，您可以在其中找到关于以上选项的更多信息，以及 Polling Consumer 页面的示例。

56.6. 交换属性

触发计时器时，它会将以下信息作为属性添加到 Exchange 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。

56.7. 示例

要设置每 60 秒生成事件的路由：

```
from("scheduler://foo?delay=60000").to("bean:myBean?method=someMethodName");
```

以上路由将生成一个事件，然后在 Registry 中名为 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>
```

56.8. 强制调度程序在完成后立即触发

要在上一个任务完成后让调度程序触发，您可以设置 greedy=true 选项。但是，注意，调度程序将保持触发所有时间。因此请谨慎使用它。

56.9. 强制调度程序闲置

在有些情况下，您可能希望调度程序触发并增长。但有时您想要"tell the scheduler"没有要轮询的任务，因此调度程序可以使用 backoff 选项更改为闲置模式。为此，您需要在 交换.SCHEDULER_POLLED_MESSAGES 的交换上设置属性，使其布尔值为 false。这会导致使用者表示没有轮询消息。

消费者将默认返回 1 个消息轮询到调度程序，否则每次消费者完成处理交换时。

56.10. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

组件支持 4 个选项，如下所列。

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	调度线程池使用的线程池中内核线程数量。默认使用单个线程。	1	整数

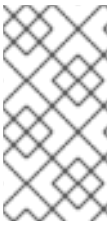
第 57 章 SEDA

支持生成者和消费者

SEDA 组件提供异步 SEDA 行为，以便在与制作者分开的线程中交换 BlockingQueue 和使用者。

请注意，队列仅在 单个 CamelContext 中可见。如果要跨 CamelContext 实例通信（例如，在 Web 应用程序间通信），请查看组件。

如果虚拟机在处理信息时终止，则此组件不会实现任何类型的持久性或恢复。如果您需要持久性、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。

57.1. URI 格式

```
seda:someName[?options]
```

其中 someName 可以是唯一标识当前 CamelContext 中的端点的任何字符串。

57.2. 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

57.2.1. 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。

57.2.2. 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。

57.3. 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

Name	描述	默认值	类型
defaultPollTimeout (consumer (advanced))	轮询时使用的超时时间（以毫秒为单位）。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让消费者在关闭时更快响应。	1000	int
defaultBlockWhenFull (producer)	是否向完整 SEDA 队列发送消息的线程是否被阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出异常，说明队列已满。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)	是否丢弃向完整 SEDA 队列发送消息的线程。默认情况下，会抛出异常，说明队列已满。通过启用此选项，调用线程将提供发送并继续，这意味着消息没有发送到 SEDA 队列。	false	布尔值
defaultOfferTimeout (producer)	是否向完整 SEDA 队列发送消息的线程是否被阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出异常，说明队列已满。通过启用这个选项，可将配置的超时添加到块问题单中。利用下线 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

57.4. 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：

57.4.1. 路径参数(1 参数)

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

57.4.2. 查询参数(18 参数)

Name	描述	默认值	类型
size (common)	SEDA 队列（例如，它可以保存的消息数）的最大容量。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)	并发线程处理交换数。	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 数量限制为最大 500。默认情况下，如果使用数字配置了端点，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))	指定是否允许多个使用者。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 队列发送一条消息，并为每个使用者收到消息的副本。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))	轮询时使用的超时时间（以毫秒为单位）。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让消费者在关闭时更快响应。	1000	int
purgeWhenStopping (consumer (advanced))	在停止消费者/路由时是否清除任务队列。这样可以更快地停止，因为队列上的任何待处理消息都会被丢弃。	false	布尔值
blockWhenFull (producer)	是否向完整 SEDA 队列发送消息的线程是否被阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出异常，说明队列已满。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)	当发送到没有活跃消费者的队列时，生成者是否应丢弃消息（不要向队列添加消息）。只有其中一个选项 discardIfNoConsumers 和 failIfNoConsumers 可以同时启用。	false	布尔值
discardWhenFull (producer)	是否丢弃向完整 SEDA 队列发送消息的线程。默认情况下，会抛出异常，说明队列已满。通过启用此选项，调用线程将提供发送并继续，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，生成者是否应该失败。只有其中一个选项 discardIfNoConsumers 和 failIfNoConsumers 可以同时启用。	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)	当队列已满时，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)	在 SEDA producer 停止等待异步任务完成前，超时（以毫秒为单位）。您可以使用 0 或负值禁用超时。	30000	long

Name	描述	默认值	类型
waitForTaskToComplete (producer)	指定调用者是否应该等待 async 任务完成的选项，然后再继续。支持以下三个选项： Always、Never 或 IfReplyExpected。前两个值是自解释。最后的值 IfReplyExpected 将仅在消息基于 Request Reply 时才会等待。默认选项为 IfReplyExpected。 Enum 值： • Never • IfReplyExpected • Always	IfReplyExpected	WaitForTaskToComplete
queue (advanced)	定义端点要使用的队列实例。		BlockingQueue

57.5. 选择 BLOCKINGQUEUE 实现

默认情况下，SEDA 组件始终会忽略 LinkedBlockingQueue，但您可以使用不同的实现，您可以引用您自己的 BlockingQueue 实现，在这种情况下没有使用 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>
```

57.6. 重新使用请求

SEDA 组件支持使用 Request Reply，调用者将等待 Async 路由完成。例如：

```
from("mina:tcp://0.0.0.0:9876?textline=true&sync=true").to("seda:input");

from("seda:input").to("bean:processInput").to("bean:createResponse");
```

在上面的路由中，我们在端口 9876 上有一个接受传入请求的 TCP 侦听程序。请求路由到 seda:input 队列。因为这是一个 Request Reply 消息，我们等待响应。当 seda:input 队列上的消费者完成时，它会将响应复制到原始消息响应。

57.7. 并发消费者

默认情况下，SEDA 端点使用单个使用者线程，但您可以将其配置为使用并发消费者线程。您可以使用它们，而不是线程池：

```
from("seda:stageName?concurrentConsumers=5").process(...)
```

与两者之间的区别一样，请注意 线程池 可在运行时动态增加/shrink，具体取决于负载，而并发消费者的数量始终被修复。

57.8. 线程池

请注意，通过执行以下操作将线程池添加到 SEDA 端点中：

```
from("seda:stageName").thread(5).process(...)
```

可以使用两个 BlockQueues: one 从 SEDA 端点获得，另一个来自线程池的工作队列，这可能不是您想要的。相反，您可能想要使用线程池配置直接端点，这样可同步和异步处理消息。例如：

```
from("direct:stageName").thread(5).process(...)
```

您还可以使用 concurrentConsumers 选项直接配置在 SEDA 端点上处理消息的线程数量。

57.9. 示例

在以下路由中，我们使用 SEDA 队列向这个 async 队列发送请求，以便能够在另一个线程中发送触发和用于get 消息以便进一步处理，并将这个线程中的恒定回复返回到原始调用者。

我们发送 Hello World 消息，并且希望回复正常。

```
@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 队列中消耗，以便进一步处理。由于这来自一个单元测试，它将发送到 模拟 端点，用户可以在单元测试中进行断言。

57.10. 使用多个CONSUMERS

在这个示例中，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");
            }
        };
    }
```

由于在 seda foo 端点上 指定了多个Consumers=true，因此我们可以拥有他们自己的消息副本作为一种 pub-sub 风格的消息传递。

因为 Bean 是单元测试的一部分，它们只是将消息发送到模拟端点。

57.11. 提取队列信息

如果需要，可以以这种方式获取队列大小等信息，而无需以这种方式使用 JMX：

```
SedaEndpoint seda = context.getEndpoint("seda:xxxx");
int size = seda.getExchanges().size();
```

57.12. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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
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	是否向完整 SEDA 队列发送消息的线程是否被阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出异常，说明队列已满。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	是否丢弃向完整 SEDA 队列发送消息的线程。默认情况下，会抛出异常，说明队列已满。通过启用此选项，调用线程将提供发送并继续，这意味着消息没有发送到 SEDA 队列。	false	布尔值
camel.component.seda.default-offer-timeout	是否向完整 SEDA 队列发送消息的线程是否被阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出异常，说明队列已满。通过启用这个选项，可将配置的超时添加到块问题单中。利用下线 java 队列的 .offer (timeout)方法。		Long
camel.component.seda.default-poll-timeout	轮询时使用的超时时间（以毫秒为单位）。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让消费者在关闭时更快响应。	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

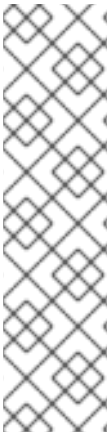
第 58 章 SERVLET

仅支持消费者

Servlet 组件提供基于 HTTP 的端点，用于消耗通过绑定到公布的 Servlet 的 HTTP 端点的 HTTP 请求。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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



注意

Stream

servlet 基于流，这意味着它收到的输入作为流提交给 Camel。这意味着您只能够读取一次流的内容。如果您遇到，当消息正文显示为空，或者您需要多次访问数据（例如：执行多播或重新发送错误处理）的情况下，您应该使用流缓存或将消息正文转换为 字符串，该字符串可以被多次读取。

58.1. URI 格式

```
servlet://relative_path[?options]
```

58.2. 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

58.2.1. 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。由于组件通常有预先配置的默认值，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。

58.2.2. 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。

58.3. 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
bridgeErrorHandler (consumer)	允许将消费者桥接到 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，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，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
muteException (consumer)	如果启用并在响应正文的消费者端处理者失败的处理，则响应的堆栈跟踪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))	是否作为 Camel Exchange 上的附件自动绑定多部分/格式数据。选项 attachmentMultipartBinding=true 和 disableStreamCache=false 无法一起工作。删除 disableStreamCache 以使用 AttachmentMultipartBinding。这默认关闭，因为这可能需要特定于 servlet 的配置才能使用 Servlets 启用此功能。	false	布尔值
fileNameExtWhitelist (consumer (advanced))	接受上传的文件的可接受的文件名扩展的白名单。可以使用逗号分隔多个扩展，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。默认情况下关闭。如果您启用此功能，则 Java 会将传入的数据从请求反序列化到 Java，这可能会存在潜在的安全风险。	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 控制 Camel 消息和 HttpClient 之间的映射。		HttpBinding
httpConfiguration (advanced)	使用共享 HttpConfiguration 作为基础配置。		HttpConfiguration
HeaderFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 将标头过滤到或从 Camel 消息过滤。		HeaderFilterStrategy

58.4. 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：

58.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
contextPath (consumer)	**必需** 要使用的上下文路径。		字符串

58.4.2. 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 的原始输入流是否缓存(Camel 会将流读取到内存/出口流到文件中，流缓存)缓存中。默认情况下，Camel 将缓存 Servlet 输入流，以支持多次读取它，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（如将其直接流传输到文件或其他持久性存储）时，您可以将此选项设置为 true。DefaultHttpBinding 将请求输入流复制到流缓存中，如果此选项为 false，则将其放在消息正文中，以支持多次读取流。如果您使用 Servlet 网桥/代理端点，请考虑启用此选项来提高性能，以防不需要多次读取消息有效负载。默认情况下，http producer 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 将标头过滤到或从 Camel 消息过滤。		HeaderFilterStrategy
httpBinding (common (advanced))	使用自定义 HttpBinding 控制 Camel 消息和 HttpClient 之间的映射。		HttpBinding
async (consumer)	将消费者配置为在 async 模式下工作。	false	布尔值

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获取传入消息或类似消息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
httpMethodRestrict (consumer)	仅在 HttpMethod 匹配时才允许使用，如 GET/POST/PUT 等。可以使用逗号分隔指定多个方法。		字符串
matchOnUriPrefix (consumer)	如果没有找到完全匹配，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)	如果启用并在响应正文的消费者端处理者失败的处理，则响应的堆栈跟踪不会包含异常的堆栈跟踪。	false	布尔值
responseBufferSize (consumer)	在 javax.servlet.ServletResponse 上使用自定义缓冲区大小。		整数
servletName (consumer)	要使用的 servlet 的名称。	Camel Servlet	字符串
transferException (consumer)	如果在消费者端启用和交换失败的处理，如果导致的例外在响应中作为 application/x-java-serialized-object 内容类型进行序列化化。在制作者一侧，异常会被反序列化并抛出，而不是 HttpOperationFailedException。需要对原因的异常进行序列化。默认情况下关闭。如果您启用此功能，则 Java 会将传入的数据从请求反序列化到 Java，这可能会存在潜在的安全风险。	false	布尔值
attachmentMultipartBinding (consumer (advanced))	是否作为 Camel Exchange 上的附件自动绑定多部分/格式数据。选项 attachmentMultipartBinding=true 和 disableStreamCache=false 无法一起工作。删除 disableStreamCache 以使用 AttachmentMultipartBinding。这默认关闭，因为这可能需要特定于 servlet 的配置才能使用 Servlets 启用此功能。	false	布尔值
eagerCheckContentAvailable (consumer (advanced))	如果 content-length 标头为 0 还是不存在，检查 HTTP 请求是否有内容。当 HTTP 客户端没有发送流化数据时，可以打开它。	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
fileNameExtWhitelist (consumer (advanced))	接受上传的文件的可接受的文件名扩展的白名单。可以使用逗号分隔多个扩展，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 IN exchange Form Encoded 正文将映射到 HTTP。把它设置为 false 将避免 HTTP 表单的正文映射。	true	布尔值
mapHttpMessageHeaders (consumer (advanced))	如果此选项为 true，则交换的 IN 交换标头将映射到 HTTP 标头。把它设置为 false 将避免 HTTP 标头映射。	true	布尔值
optionsEnabled (consumer (advanced))	指定是否为这个 Servlet 使用者启用 HTTP OPTIONS。默认情况下关闭 OPTIONS。	false	布尔值
traceEnabled (consumer (advanced))	指定是否为这个 Servlet 使用者启用 HTTP TRACE。默认情况下关闭 TRACE。	false	布尔值

58.5. 消息标头

Camel 将应用与 HTTP 组件相同的消息标头。

Camel 还会填充 所有 request.parameter 和 request.headers。例如，如果客户端请求具有 URL

http://myserver/myserver?orderid=123，则交换将包含名为 orderid 的标头，值为 123。

58.6. 使用方法

您只能从 Servlet 组件生成的端点使用。因此，它应仅用作 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。

58.7. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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。默认情况下关闭。如果您启用此功能，则 Java 会将传入的数据从请求反序列化到 Java，这可能会存在潜在的安全风险。	false	布尔值
camel.component.servlet.attachment-multipart-binding	是否作为 Camel Exchange 上的附件自动绑定多部分/格式数据。选项 attachmentMultipartBinding=true 和 disableStreamCache=false 无法一起工作。删除 disableStreamCache 以使用 AttachmentMultipartBinding。这默认关闭，因为这可能要求特定于 servlet 的配置在使用 Servlet 时启用此功能。	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

Name	描述	默认值	类型
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获取传入消息或类似消息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
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	接受上传的文件的可接受的文件名扩展的白名单。可以使用逗号分隔多个扩展，如 txt,xml。		字符串
camel.component.servlet.header-filter-strategy	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 将标头过滤到或从 Camel 消息过滤。选项是一个 org.apache.camel.spi.HeaderFilterStrategy 类型。		HeaderFilterStrategy
camel.component.servlet.http-binding	使用自定义 HttpBinding 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	使用共享 HttpConfiguration 作为基础配置。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	如果启用并在响应正文的消费者端处理者失败的处理，则响应的堆栈跟踪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

Name	描述	默认值	类型
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 59 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 传递消息正文中包含的消息。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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

59.1. URI 格式

发送消息到频道。

```
slack:#channel[?options]
```

向 slackuser 发送直接消息：

```
slack:@userID[?options]
```

59.2. 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

59.2.1. 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。

59.2.2. 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。

59.3. 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

59.4. 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：

59.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
channel (common)	**必需** 频道名称（已弃用）或 slackuser （首选 userName）直接向用户发送消息。		字符串

59.4.2. 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)	以自然顺序创建交换（最旧于最新的交换）。	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)	**弃用的** 是 bot 在向频道或用户发送消息时具有的用户名。		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

Name	描述	默认值	类型
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
timeUnit (scheduler)	initialDelay 和 delay 选项的时间单位。 Enum 值： • NANOSECONDS • MICROSECONDS • MILLISECONDS • SECONDS • MINUTES • HOURS • DAYS	MILLISECONDS	TimeUnit

Name	描述	默认值	类型
useFixedDelay (scheduler)	控制是否使用固定延迟或固定率。详情请参阅 JDK 中的 ScheduledExecutorService。	true	布尔值

59.5. 在 TTY XML 中配置

具有 XML 的 Slack 组件必须配置为 Spring 或 Blueprint bean，其中包含传入的 webhook url 或作为参数集成的应用令牌。

```
<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。

59.6. 示例

带有蓝图的 CamelContext 可以是：

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0" default-activation="lazy">

    <bean id="slack" class="org.apache.camel.component.slack.SlackComponent">
        <property name="webhookUrl"
value="https://hooks.slack.com/services/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"/>
    </bean>

    <camelContext xmlns="http://camel.apache.org/schema/blueprint">
        <route>
            <from uri="direct:test"/>
            <to uri="slack:#channel?iconEmoji=:camel:&amp;username=CamelTest"/>
        </route>
    </camelContext>

</blueprint>
```

59.7. 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 https://api.slack.com/block-kit 中了解更多有关它的信息。

```
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);
}
```

59.8. 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，您从常规频道获取最后的消息。消费者将跟踪最后一次消耗的消息的时间戳，在下次轮询时，它将从该时间戳检查。

您需要创建一个 Slack 应用程序并在工作区中使用它。

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 dialogType 选项设置它(PUBLIC_CHANNEL、PRIVATE_CHANNEL、MPIM、IM)

pureOrder 选项允许使用从最旧的消息到最新的消息。最初，您首先会获得最新的并消耗后（消息 3 IANA 消息 2 IANA 消息 1）

注意

您可以使用 dialogType 选项从不是公共的频道读取历史记录和消息 (PUBLIC_CHANNEL、PRIVATE_CHANNEL、MPIM、IM)

59.9. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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

Name	描述	默认值	类型
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 60 章 SPRING BATCH

从 Camel 2.10 开始

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。

Maven 用户必须在其 pom.xml 中为这个组件添加以下依赖项：

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

60.1. URI 格式

```
spring-batch:jobName[?options]
```

JOB NAME 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。

60.2. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

60.2.1. 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。

60.2.2. 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。

60.3. 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

60.4. 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：

60.4.1. 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

60.4.2. 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

60.5. 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。

60.6. 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();
```

60.7. 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。

60.7.1. 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>
```

60.7.2. 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>
```

60.7.3. 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>
```

60.7.4. 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>
```

60.8. SPRING BOOT AUTO-CONFIGURATION

当在 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>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.component.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

Name	描述	默认值	类型
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 61 章 SPRING JDBC

从 Camel 3.10 开始

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。

有关此组件的常规使用，请查看 JDBC 组件。

Maven 用户必须在其 pom.xml 中为这个组件添加以下依赖项：

```
<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>
```

61.1. 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

```
<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>
```

61.2. 配置选项

Camel 组件在两个独立级别上配置：

61.3. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

61.3.1. 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。

61.3.2. 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。

61.4. 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
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
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

61.5. 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：

61.5.1. 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

61.5.2. 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

Name	描述	默认值	类型
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
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
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

61.6. SPRING BOOT AUTO-CONFIGURATION

当在 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>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

组件支持下面列出的 4 个选项。

Name	描述	默认值	类型

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 62 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。

Maven 用户必须在其 pom.xml 中为这个组件添加以下依赖项：

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

62.1. 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。

62.2. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

62.2.1. 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。

62.2.2. 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。

62.3. 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

62.4. 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：

62.4.1. 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

62.4.2. 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

62.5. 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。

62.5.1. 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。

62.5.2. 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。

62.5.3. unbind

不需要其他条目，删除具有指定 dn 的节点。

62.5.4. 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。

62.5.5. 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

62.5.6. 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：

- public static final String 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
```

62.6. SPRING BOOT AUTO-CONFIGURATION

在 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>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

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 63 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。

Maven 用户必须在其 pom.xml 中为这个组件添加以下依赖项：

```
<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>
```

63.1. 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。

63.2. 配置选项

Camel 组件在两个级别上配置：

- 组件级别
- 端点级别

63.2.1. 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。

63.2.2. 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。

63.3. 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。必须在组件或端点上配置连接工厂。		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 用户与 JMS 代理的有效连接。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 producer 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： • direct • fanout • 标头 • topic	direct	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)	是否允许在没有正文的情况下发送消息。如果此选项为 false，且消息正文为 null，则会抛出 MessageConversionException。	false	布尔值

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
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

Name	描述	默认值	类型
HeaderFilterStrategy (filter)	使用自定义 org.apache.camel.spi.HeaderFilterStrategy 将标头过滤到或从 Camel 消息过滤。		HeaderFilterStrategy

63.4. 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：

63.4.1. 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

63.4.2. 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。必须在组件或端点上配置连接工厂。		ConnectionFactory

Name	描述	默认值	类型
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 从路由中消耗，并且您不希望 Camel 自动发送回复消息，则可以使用此选项，因为代码中的另一个组件处理回复消息。如果要在不同的消息代理之间将 Camel 用作代理，并且希望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
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 用户与 JMS 代理的有效连接。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 producer 也经过测试。	false	布尔值

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
autoStartup (consumer)	指定消费者容器是否应自动启动。	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： • direct • fanout • 标头 • topic	direct	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
exchangeType (consumer)	交换的类型。 Enum 值： • direct • fanout • 标头 • topic	direct	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

Name	描述	默认值	类型
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
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

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
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
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

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
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
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
Sync (advanced)	设置是否应严格使用同步处理。	false	布尔值

63.5. 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

63.6. 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>
```

63.7. 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")
```

63.8. 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	默认值
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	默认值
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。

63.9. 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。

63.10. 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 "));
```

63.11. 重复使用端点并发送到在运行时计算的不同目的地

如果您需要发送消息到大量不同的 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。这可显著减少内存和线程资源上创建的端点数量。

使用 toD 比使用标头指定动态目的地更容易

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）。

63.12. 使用 TOD

如果您需要发送消息到许多不同的交换，您必须重复使用端点，并使用 toD 指定带有简单语言的动态目的地。

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}");
```

63.13. SPRING BOOT AUTO-CONFIGURATION

在 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>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

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	是否允许在没有正文的情况下发送消息。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自动启动。	true	布尔值
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

Name	描述	默认值	类型
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。必须在组件或端点上配置连接工厂。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。	direct	字符串
camel.component.spring-rabbitmq.dead-letter-queue	死信队列的名称。		字符串
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

Name	描述	默认值	类型
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 将标头过滤到或从 Camel 消息过滤。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
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

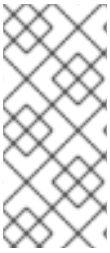
Name	描述	默认值	类型
camel.component.spring-rabbitmq.maximum-retry-attempts	如果 Camel 无法处理消息，则 Rabbitmq 使用者将重试相同的消息的次数。	5	整数
camel.component.spring-rabbitmq.message-converter	要使用自定义的 MessageConverter，以便您可以控制如何映射到 org.springframework.amqp.core.Message。选项是一个 org.springframework.amqp.support.converter.MessageConverter 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
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

Name	描述	默认值	类型
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
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 用户与 JMS 代理的有效连接。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 producer 也经过测试。	false	布尔值

第 64 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。



注意

先决条件

要使用此组件，您必须有一个 Redis 服务器正在运行。

64.1. URI 格式

```
spring-redis://host:port[?options]
```

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。

某些组件只有几个选项，其他组件可能会有许多选项。由于组件通常有预先配置的默认值，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。

64.3. 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

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

64.4. 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：

64.4.1. 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

64.4.2. 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	SET	命令

Name	描述	默认值	类型
	• DEL • TYPE • FLUSHDB • KEYS • RANDOMKEY • RENAME • RENAMENX • RENAMEX • DBSIZE • EXPIRE • EXPIREAT • TTL • 选择 • MOVE • FLUSHALL • GETSET • MGET • SETNX • SETEX • MSET • MSETNX • DECRBY • DECR • INCRBY • INCR • 附加 • SUBSTR • HSET • HGET • HSETNX • HMSET • HMGET • HINCRBY		

Name	描述	默认值	类型
	• HEXISTS • HDEL • HLEN • HKEYS • HVALS • HGETALL • RPUSH • LPUSH • LLEN • LRANGE • LTRIM • LINDEX • LSET • LREM • LPOP • RPOP • RPOPLPUSH • SADD • SMEMBERS • SREM • SPOP • SMOVE • SCARD • SISMEMBER • SINTER • SINTERSTORE • SUNION • SUNIONSTORE • SDIFF • SDIFFSTORE • SRANDMEMBER • ZADD • ZRANGE		

Name	描述	默认值	类型
	• ZREM • ZINCRBY • ZRANK • ZREVRANK • ZREVRANGE • ZCARD • ZSCORE • MULTI • 丢弃 • EXEC • WATCH • UNWATCH • SORT • BLPOP • BRPOP • AUTH • 订阅 • PUBLISH • 取消订阅 • PSUBSCRIBE • PUNSUBSCRIBE • ZCOUNT • ZRANGEBYSCORE • ZREVRANGEBYSCORE • ZREMRANGEBYRANK • ZREMRANGEBYSCORE • ZUNIONSTORE • ZINTERSTORE • 保存 • BGSAVE • BGREWRITEAOF • LASTSAVE • SHUTDOWN		

Name	描述	默认值	类型
	• INFO • MONITOR • SLAVEOF • CONFIG • STRLEN • 同步 • LPUSHX • PERSIST • RPUSHX • ECHO • LINSERT • DEBUG • BRPOPLPUSH • SETBIT • GETBIT • SETRANGE • GETRANGE • PEXPIRE • PEXPIREAT • GEOADD • GEODIST • 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

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

64.5. 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

Name	描述	默认值	类型
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
CamelRedis.Timeout (common) 常数 ： TIMEOUT	超时。		Long
CamelRedis.Offset (common) 常数 ： OFFSET	偏移。		Long

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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
CamelRedis.Longitude (common) 常数： LONGITUDE	拉特语.		å�Œ
CamelRedis.Radius (common) 常量： RADIUS	RADIUS。		å�Œ

64.6. 使用方法

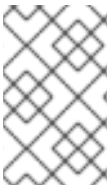
另外，请参阅可用的 单元测试。

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。

64.6.1. 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

哈希命令	描述	参数	结果
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
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>

哈希命令	描述	参数	结果
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>
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

列出命令	描述	参数	结果
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
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

列出命令	描述	参数	结果

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
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>

设置命令	描述	参数	结果
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
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

排序的集合命令	描述	参数	结果
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
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

排序的集合命令	描述	参数	结果
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

字符串命令	描述	参数	结果
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
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

字符串命令	描述	参数	结果
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
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>

关键命令	描述	参数	结果
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
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
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

其他命令	描述	参数	结果
丢弃	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

64.7. 依赖项

Maven 用户需要将以下依赖项添加到其 pom.xml 中：

pom.xml

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

64.8. SPRING BOOT AUTO-CONFIGURATION

当在 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>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.component.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

Name	描述	默认值	类型
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 65 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。

Maven 用户必须在其 pom.xml 中为这个组件添加以下依赖项：

```
<dependency>
    <groupId>org.apache.camel.springboot</groupId>
    <artifactId>camel-spring-ws-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>
```

65.1. URI 格式

此组件的 URI 方案如下

```
spring-ws:[mapping-type:]address[?options]
```

要公开 Web 服务 映射类型，需要设置为以下任意一种：

映射类型	描述
rootqname	提供根据消息中包含的 root 元素的合格名称来映射 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。

65.2. 配置选项

Camel 组件在两个级别上配置：

- *组件级别*
- *端点级别*

65.2.1. 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。

65.2.2. 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。

65.3. 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

65.4. 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：

65.4.1. 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

65.4.2. 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 SSLContext 参数配置安全性：		SSLContextParameters

65.5. 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
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

Name	描述	默认值	类型
CamelSpringWebserviceSoapHeader (producer) 常量 ：**SPRING_WS_SOAP_HEADER**	soap 标头源。		源
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

65.6. 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。

65.7. 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");
```

65.8. 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。

65.9. 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。

65.10. 如何使用风格表转换 SOAP 标头

标头转换过滤器(HeaderTransformationMessageFilter.java)可用于转换 soap 请求的 soap 标头。如果要使用标头转换过滤器，请参阅以下示例：

```
<bean id="headerTransformationFilter"
class="org.apache.camel.component.spring.ws.filter.impl.HeaderTransformationMessageFilter">
    <constructor-arg index="0" value="org/apache/camel/component/spring/ws/soap-header-transform.xslt"/>
</bean>
```

使用在 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/GetQuote&amp;messageFilter=#headerTransformationFilter"/>
</route>
```

65.11. 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。*

65.12. 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*/ }
```

65.13. 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>
```

65.14. 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。

65.15. 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)
```

65.15.1. 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.PayloadRootQNameEndpointMapping">
  <property name="mappings">
    <props>
      <prop key="{http://example.com/}GetFuture">FutureEndpointDispatcher</prop>
      <prop key="{http://example.com/}GetQuote">QuoteEndpointDispatcher</prop>
    </props>
  </property>
</bean>

<bean id="QuoteEndpointDispatcher"
class="org.apache.camel.component.spring.ws.bean.CamelEndpointDispatcher" />
<bean id="FutureEndpointDispatcher"
class="org.apache.camel.component.spring.ws.bean.CamelEndpointDispatcher" />
```

65.16. 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);
```

65.17. SPRING BOOT AUTO-CONFIGURATION

当在 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>
  <version>x.x.x</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

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 66 章 SQL

支持生成者和消费者

SQL 组件允许您使用 JDBC 查询来处理数据库。此组件和 JDBC 组件之间的区别在于，如果 SQL 查询是端点的属性，它将消息有效负载用作传递给查询的参数。

这个组件使用 spring-jdbc 后面的 springenes 进行实际 SQL 处理。

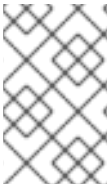
Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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

SQL 组件还支持：

- 适用于 Idempotent Consumer EIP 模式的 JDBC 存储库。请参见以下信息。
- 适用于聚合器 EIP 模式的 JDBC 存储库。请参见以下信息。

66.1. 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 :'114name_of_the_parameter' 样式使用命名参数，如下所示：

```
sql:select * from table where id=:#myId order by name[?options]
```

使用命名参数时，Camel 将以给定优先级查找名称：

1. 来自消息正文（如果它是一个 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，表示 SQL 查询的参数的标准 ? 符号被替换为 kiosk 符号，因为 ? 符号用于为端点指定选项。? 符号替换可以基于端点配置。

您可以将 SQL 查询外部化到 classpath 或文件系统中的文件，如下所示：

```
sql:classpath:sql/myquery.sql[?options]
```

myquery.sql 文件位于 classpath 中，只是纯文本

```
-- this is a comment
select *
from table
where
  id = :#${exchangeProperty.myId}
order by
  name
```

在文件中，您可以使用多行并根据需要格式化 SQL。还使用注释，如 - dash 行。

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

SQL 组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
数据源 (common)	**Autowired** 设置用于与数据库通信的 DataSource。		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。这个选项是默认的 true。	true	布尔值

66.4. 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：

66.4.1. 路径参数(1 参数)

Name	描述	默认值	类型

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: 作为前缀并指定文件的位置来外部化查询。		字符串

66.4.2. 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
数据源 (common)	**Autowired** 设置 DataSource，用于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而不是消息正文中。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，则该值将用作存储查询结果的标头名称，并保留原始消息正文。		字符串
outputType (common)	将消费者或生成者的输出设置为 SelectList 为 Map 列表，或者以以下方式选择One 作为单个 Java 对象：a）如果查询只有一个列，则返回 JDBC Column 对象。（如 SELECT COUNT （）FROM PROJECT 将返回一个 Long 对象。b）如果查询有多个列，则会返回此结果的映射。c 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。它假设您的类具有创建实例的默认构造器。d）如果查询导致多个行，它会抛出一个非唯一结果异常。StreamList 使用 Iterator 流查询的结果。这可以在流模式下与 Splitter EIP 一起使用，以流传输方式处理 ResultSet。 Enum 值： • SelectOne • SelectList • StreamList	Select List	SqlOutputType
分隔符 (common)	从消息正文中获取参数值时使用的分隔符（如果正文是 String 类型），以插入在 TTY 占位符上。请注意，如果您使用命名参数，则使用 Map 类型。默认值为逗号。	,	char

Name	描述	默认值	类型
breakBatchOnConsumeFail (consumer)	设置在Consume 失败时是否中断批处理。	false	布尔值
bridgeErrorHandler (consumer)	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
expectedUpdateCount (consumer)	设置在使用 onConsume 时验证的预期更新数。	-1	int
maxMessagesPerPoll (consumer)	设置要返回的最大消息数。		int
onConsume (consumer)	在处理每行后，如果成功处理交换，则可以执行此查询，例如将行标记为 handle。查询可以有参数。		字符串
onConsumeBatchComplete (consumer)	处理整个批处理后，可以对批量更新行执行此查询。查询不能有参数。		字符串
onConsumeFailed (consumer)	在处理每行后，如果交换失败，则可以执行此查询，例如将行标记为失败。查询可以有参数。		字符串
routeEmptyResultSet (consumer)	设置是否应允许将空结果设置发送到下一跳。默认为 false。因此，空结果会被过滤掉。	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，则消费者会破坏处理任何进一步的交换，从而导致回滚 eager。	false	布尔值
useIterator (consumer)	设置如何将结果集传送到路由。将发送表示为列表或单个对象。默认为 true。	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
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 在消费者处理行/批处理时执行查询。		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 消息作为持续处理的 OUT 消息。	false	布尔值
useMessageBodyForSql (producer)	是否使用消息正文作为 SQL，然后是参数的标头。如果启用了这个选项，则不会使用 uri 中的 SQL。请注意，消息正文中的查询参数由问号而不是 站 符号表示。	false	布尔值
alwaysPopulateStatement (advanced)	如果启用，则始终调用来自 org.apache.camel.component.sql.SqlPrepareStatementStrategy 的 populateStatementStrategy，如果没有准备预期的参数。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，这可避免读取无参数的 SQL 查询的消息正文/标题。	false	布尔值
parametersCount (advanced)	如果设置大于零，则 Camel 将使用此可替换的参数计数，而不是通过 JDBC 元数据 API 查询。如果 JDBC 供应商无法返回正确的参数数，则这非常有用，然后用户可能会覆盖。		int

Name	描述	默认值	类型
占位符 (advanced)	指定在 SQL 查询中将被替换的字符。请注意，它只是简单的 String.replaceAll （）操作，且不会涉及 SQL 解析（加引号的字符串也会改变）。	#	字符串
prepareStatementStrategy (advanced)	允许插件使用自定义 org.apache.camel.component.sql.SqlPrepareStatementStrategy 来控制查询的准备和准备声明。		SqlPrepareStatementStrategy
templateOptions (advanced)	使用映射中的键/值配置 Spring JdbcTemplate。		Map
usePlaceholder (advanced)	设置是否使用占位符，并将所有占位符字符替换为 SQL 查询中的符号。	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

66.5. 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（其中每个查询参数都由端点 URI 中一个 附件符号（或已配置的占位符）表示。如果消息正文不是数组或集合，则转换器会导致迭代一个对象，这是正文本身。

例如，如果消息正文是 java.util.List 实例，则列表中的第一个项将被取代在 SQL 查询中的第一个出现，则列表中的第二个项目将被取代，以此类推。

如果 batch 设为 true，则入站消息正文正文的解释稍微变化 - 而不是参数迭代器，组件需要一个包含参数迭代器的迭代器；外部迭代器的大小决定了批处理大小。

您可以使用选项 useMessageBodyForSql，允许将消息正文用作 SQL 语句，然后在带有键 SqlConstants.SQL_PARAMETERS 的标头中提供 SQL 参数。这样，SQL 组件可以更动态地工作，因为 SQL 查询来自消息正文。使用模板（如 Velocity、自由标记器）进行条件处理，例如，根据查询参数的存在，使用模板（如 Velocity、Freemarker）来包括或排除 where 子句。

66.6. 查询的结果

对于 选择 操作，结果是 List<Map<String, Object>& gt; 类型的实例，由 JdbcTemplate.queryForList （） 方法返回。对于 update 操作，返回 NULL 正文，因为更新操作仅设置为标头，永远不会设置为正文。

注意

有关更新操作的更多信息，请参阅 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，则结果将放置在标头中。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，用于在标头中查询序列或其它小值。将 outputHeader 和 outputType 组合在一起：

```
from("jms:order.inbox")
    .to("sql:select order_seq.nextval from dual?
outputHeader=OrderId&outputType=SelectOne")
    .to("jms:order.booking");
```

66.7. 使用 STREAMLIST

producer 支持 outputType=StreamList，它使用迭代器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();
```

66.8. 标头值

在执行更新操作时，SQL 组件会在以下消息标头中存储更新计数：

标头	描述
CamelSqlUpdateCount	为 **更新操作** 更新的行数，返回为 **Integer** 对象。使用 outputType=StreamList 时不提供此标头。
CamelSqlRowCount	为所选操作返回的行数，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 组件会在以下消息标头中使用生成的密钥和这些行编号存储行：

标头	描述
CamelSqlGeneratedKeysRowCount	包含生成的密钥的标头中的行数。
CamelSqlGeneratedKeyRows	包含生成的密钥的行（键的映射列表）。

66.9. 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。
To do that set the header CamelSqlRetrieveGeneratedKeys=true.然后，生成的密钥将使用上表中列出的键作为标头提供。

要指定应检索生成的列，请分别将标头 CamelSqlGeneratedColumns 设置为 String[] 或 int[]，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决定参数数量，可能还需要使用 parametersCount 选项。

66.10. DATASOURCE

您可以在 URI 中直接设置对 DataSource 的引用：

```
sql:select * from table where id=# order by name?dataSource=#myDS
```

66.11. 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，即 :114lic 和 :114min。
然后，Camel 将从消息正文或消息标头中查找这些参数。请注意，在上面的示例中，我们为命名参数设置了两个带有恒定值
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")
```

但是，如果消息正文是 java.util.Map，则指定的参数将从正文获取。

```
from("direct:projects")
  .to("sql:select * from projects where license = :#lic and id > :#min order by id")
```

66.12. 在制作者中使用表达式参数

在以下给定路由中，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")
```

66.12.1. 在消费者中使用表达式参数

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、date 或 something。

例如，以下示例中为 bean myIdGenerator 调用 nextId 方法：

```
from("sql:select * from projects where id = :#${bean:myIdGenerator.nextId}")
  .to("mock:result");
```

并且 bean 有以下方法：

```
public static class MyIdGenerator {

  private int id = 1;

  public int nextId() {
    return id++;
  }
```

请注意，没有带有消息正文和标头的现有 交换，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。

66.13. 使用带有动态值的 IN 查询

SQL producer 允许使用带有 IN 值动态计算的 IN 语句的 SQL 查询。例如，来自消息正文或标头等。

要使用 IN，您需要：

- 使用以下内容为参数名称 添加前缀：
- 在参数外添加 ( )

下面是一个更好的示例。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：

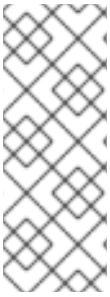
```
// use an array
template.requestBodyAndHeader("direct:query", "Hi there!", "names", new String[]{"Camel",
"AMQ"});

// 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");
```

查询也可以在端点中指定，而不是外部化（请注意，外部化使得维护 SQL 查询变得更加容易）

```
from("direct:query")
    .to("sql:select * from projects where project in (:#in:names) order by id")
    .to("log:query")
    .to("mock:query");
```

66.14. 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，我们需要创建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二进制字符串类型。它不映射到任何 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，因此我们不会在此处显示它。

66.14.1. 自定义 JDBC idempotency 存储库

您有几个选项来根据您的需要调整 org.apache.camel.processor.idempotent.jdbc.JdbcMessageIdRepository ：

参数	默认值	描述
createTableIfNotExists	true	定义 Camel 是否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，以指示表不存在。
createString	CREATE TABLE CAMEL_MESSAGEPROCESSED (processorName VARCHAR （ASP, 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的查询（结果不等于'0'）。它采用两个参数。第一个是处理器名称(**字符串**)，第二个是消息 ID (**字符串**)。
insertString	INSERT INTO CAMEL_MESSAGEPROCESSED (processorName, messageId, createdAt) VALUES (?, ?, ?)	用于在表中添加条目的声明。它采用三个参数。第一个是处理器名称(**字符串**)，第二个是消息 ID (**字符串**)，第三个是此条目添加到存储库中的时间戳 (**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。它采用两个参数。第一个是处理器名称(**字符串**)，第二个是消息 ID (**字符串**)。

*选项 **tableName** 可用于使用默认的 **SQL** 查询，但具有不同表名称。但是，如果要自定义 **SQL** 查询，您可以单独配置它们。*

66.14.2. Orp Lock aware Jdbc IdempotentRepository

***org.apache.camel.processor.idempotent.jdbc.JdbcMessageIdRepository** 的一个限制是它不处理 **JVM** 崩溃或非安全关闭的孤立锁定。如果您需要解决孤立锁处理，则这可能会导致未处理的文件/消息。这对 **camel-file**、**camel-ftp** 等实现。如果您需要解决孤立锁定处理，则使用 **org.apache.camel.processor.idempotent.jdbc.JdbcOrphanLockAwareIdempotentRepository**。此存储库跟踪由应用程序实例保存的锁定。对于保存的每个锁定，应用程序将向锁定存储库发送实时信号，从而使用当前的 **Timestamp** 更新 **createdAt** 列。当应用程序实例尝试获取锁定时，如果存在三个可能：*

- 锁定条目不存在，则使用 JdbcMessageIdRepository 的基本实现提供锁。
- lock already exists and the createdAt < System.currentTimeMillis() - lockMaxAgeMillis.在这种情况下，假设活跃的实例有锁定，并且没有为请求锁定的新实例提供锁定
- lock already exists and the createdAt > = System.currentTimeMillis() - lockMaxAgeMillis.在这种情况下，假设没有活动实例，没有锁定，并且为请求的实例提供锁定。原因是，如果原始实例有锁定（如果仍然在运行），它将使用 keepAlive 机制在 createdAt 上更新 Timestamp

此存储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，如 currentTimestamp - createdAt >= lockMaxAgeMillis，则锁定会被孤立。
lockKeepAliveIntervalMillis	保持实时更新的频率到创建At Timestamp 列。

66.14.3. Caching Jdbc IdempotentRepository

某些 SQL 实现不会因每个查询而快速进行。JdbcMessageIdRepository 实施在 SQL 事务中单独执行其幂等性检查。检查 100 个密钥可能需要几分钟。JdbcCachedMessageIdRepository 使用整个键列表在启动时预加载内存缓存。然后，在传递给原始实现前，首先检查此缓存。

与所有缓存实现一样，应该考虑与过时的数据和特定用途相关的注意事项。

66.15. 使用基于 JDBC 的聚合存储库

JdbcAggregationRepository 是一个 聚合Repository，可立即保留聚合的消息。这样可确保不会丢失消息，因为默认聚合器将仅在内存中使用 AggregationRepository。JdbcAggregationRepository 允许与 Camel 一起为聚合器提供持久支持。

只有成功处理 Exchange 时，它才会被标记为 complete，当 确认方法在 聚合 时被调用时会出现这种情况。这意味着，如果同一交换再次失败，它将会一直被重试，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 知道当 maximumRedeliveries 为 hit 时发送 Exchange 的位置。

您可以查看 camel-sql 单元测试中的一些示例，如 JdbcAggregateRecoverDeadLetterChannelTest.java

66.15.1. 数据库

要正常工作，每个聚合器都使用两个表：聚合并完成。按照惯例，完成的名称与聚合的名称相同，后缀为 "_COMPLETED "。名称必须使用 RepositoryName 属性在 Spring bean 中配置。在以下示例中，将使用聚合。

两个表的表结构定义相同：在这两种情况下，字符串值都用作键(id)，而 Blob 包含字节数组中的交换序列化。
但是，应该记住一个区别： id 字段没有与表相同的内容。
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 包含存储在相应 blob 字段中的交换的 id。

以下是用于创建表的 SQL 查询，只需将 "聚合" 替换为您的聚合器存储库名称。

```
CREATE TABLE aggregation (
 id varchar(255) NOT NULL,
 exchange blob NOT NULL,
 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)
);
```

66.16. 将正文和标头存储为文本

您可以配置 JdbcAggregationRepository 以存储消息正文，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，请使用以下 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，将存储库配置为启用此行为，如下所示：

```
<bean id="repo3"
  class="org.apache.camel.processor.aggregate.jdbc.JdbcAggregationRepository">
  <property name="repositoryName" value="aggregationRepo3"/>
  <property name="transactionManager" ref="txManager3"/>
  <property name="dataSource" ref="dataSource3"/>
  <!-- configure to store the message body and following headers as text in the repo -->
  <property name="storeBodyAsText" value="true"/>
  <property name="headersToStoreAsText">
    <list>
      <value>companyName</value>
      <value>accountName</value>
    </list>
  </property>
</bean>
```

66.16.1. codec (Serialization)

由于他们可以包含任何类型的有效负载，因此通过设计，交换不可序列化。它将转换为字节数组，以存储在数据库 BLOB 字段中。所有这些转换都由 JdbcCodec 类处理。代码的详情需要您注意的是：ClassLoadingAwareObjectInputStream。

ClassLoadingAwareObjectInputStream 已从 Apache ActiveMQ 项目中重复使用。它嵌套 ObjectInputStream，并将其与 ContextClassLoader 一起使用，而不是 当前的Thread。优点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。

66.16.2. 事务

需要 Spring PlatformTransactionManager 来编配事务。

66.16.2.1. 服务(Start/Stop)

start 方法验证数据库的连接以及所需的表是否存在。如果出现错误，它将在启动过程中失败。

66.16.3. 聚合器配置

根据目标环境，聚合器可能需要一些配置。如您知道，每个聚合器应具有自己的存储库（以及数据库中创建的对应表对）和数据源。如果默认 lobHandler 没有适应您的数据库系统，可以使用 lobHandler 属性注入它。

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>
```

66.16.4. 光率锁定

您可以在集群环境中打开 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相同的数据库。如果存在竞争条件，则 JDBC 驱动程序会抛出特定于供应商的异常，JdbcAggregationRepository 可以对其做出反应。要了解 JDBC 驱动程序导致的例外被认为是选择锁定错误，我们需要一个映射程序来执行此操作。因此，有一个 org.apache.camel.processor.aggregate.jdbc.JdbcOptimisticLockingExceptionMapper，您可以根据需要实施自定义逻辑。有一个默认的实现 org.apache.camel.processor.aggregate.jdbc.DefaultJdbcOptimisticLockingExceptionMapper，它可以正常工作：

完成以下检查：

- 如果原因的例外是 SQLException，则检查 SQLState （如果以 23 开始）。

- 如果原因的例外是 DataIntegrityViolationException
- 如果原因的例外类名称的名称中具有 "ConstraintViolation"。
- 如果配置了任何类名称，对 FQN 类名称的可选检查匹配。

您可以添加 FQN classnames，如果任何原因异常（或任何嵌套）等于任何 FQN 类名称，则它有一个 optimistick locking 错误。

下面是一个例子，其中从 JDBC 供应商定义 2 个额外的 FQN 类名称。

```
<bean id="repo"
class="org.apache.camel.processor.aggregate.jdbc.JdbcAggregationRepository">
  <property name="transactionManager" ref="transactionManager"/>
  <property name="repositoryName" value="aggregation"/>
  <property name="dataSource" ref="dataSource"/>
  <property name="jdbcOptimisticLockingExceptionMapper" ref="myExceptionMapper"/>
</bean>
<!-- use the default mapper with extraFQN class names from our JDBC driver -->
<bean id="myExceptionMapper"
class="org.apache.camel.processor.aggregate.jdbc.DefaultJdbcOptimisticLockingExceptionMapper">
  <property name="classNames">
    <util:set>
      <value>com.foo.sql.MyViolationExceptoion</value>
      <value>com.foo.sql.MyOtherViolationExceptoion</value>
    </util:set>
  </property>
</bean>
```

66.16.5. 传播行为

JdbcAggregationRepository 使用来自 Spring-TX 的两个不同 事务模板。一个是只读的，一个用于读写操作。

但是，当在自身使用 < transacted /> 的路由中使用 JdbcAggregationRepository 时，可能需要配置 JdbcAggregationRepository 中的事务模板使用的 传播行为。

以下是实现它的方法：

```
<bean id="repo"
class="org.apache.camel.processor.aggregate.jdbc.JdbcAggregationRepository">
  <property name="propagationBehaviorName" value="PROPAGATION_NESTED" />
</bean>
```

传播由 org.springframework.transaction.TransactionDefinition 接口常量来指定，因此 propagationBehaviorName 是方便的，允许使用常量名称。

66.16.6. PostgreSQL 问题单

有特殊数据库可能会导致 JdbcAggregationRepository 使用的最佳锁定问题。如果数据完整性违反异常（带有 SQLState 23505），PostgreSQL 会将连接标记为无效。这使得连接在嵌套的事务中有效不可用。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 .. 语句提供选择锁定行为。

这个声明是（带有默认聚合表定义）：

```
INSERT INTO aggregation (id, exchange) values (?, ?) ON CONFLICT DO NOTHING
```

详情请参考 PostgreSQL 文档。

当使用此条款时，java.sql.PreparedStatement.executeUpdate （） 调用会返回 0， 而不是抛出 SQLException with SQLState=23505。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。

66.17. CAMEL SQL STARTER

spring-boot 用户提供了一个启动程序模块。使用入门程序时，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。

66.18. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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

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，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，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。这个选项是默认的 true。	true	布尔值

第 67 章 STUB

支持生成者和消费者

Stub 组件提供了一种简单的方法，在开发或测试中存根出任何物理端点，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 的前面添加 stub: 到 stub out 端点。

Stub 组件内部 创建虚拟机 端点。Stub 和 VM 之间的主要区别在于，虚拟机将验证您提供的 URI 和参数，因此将 vm: 置于带有查询参数的典型 URI 前通常会失败。根根不会发生，因为它基本上会忽略所有查询参数，以便您快速退出路由中的一个或多个端点。

67.1. URI 格式

```
stub:someUri
```

其中 someUri 可以是具有任何查询参数的任何 URI。

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。

67.2.1.1. 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

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
defaultPollTimeout (consumer (advanced))	轮询时使用的超时时间（以毫秒为单位）。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让消费者在关闭时更快响应。	1000	int
defaultBlockWhenFull (producer)	是否向完整 SEDA 队列发送消息的线程是否被阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出异常，说明队列已满。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

Name	描述	默认值	类型
defaultDiscardWhenFull (producer)	是否丢弃向完整 SEDA 队列发送消息的线程。默认情况下，会抛出异常，说明队列已满。通过启用此选项，调用线程将提供发送并继续，这意味着消息没有发送到 SEDA 队列。	false	布尔值
defaultOfferTimeout (producer)	是否向完整 SEDA 队列发送消息的线程是否被阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出异常，说明队列已满。通过启用这个选项，可将配置的超时添加到块问题单中。利用下线 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

67.4. 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：

67.4.1. 路径参数(1 参数)

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

67.4.2. 查询参数(18 参数)

Name	描述	默认值	类型
size (common)	SEDA 队列（例如，它可以保存的消息数）的最大容量。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)	并发线程处理交换数。	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 数量限制为最大 500。默认情况下，如果使用数字配置了端点，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))	指定是否允许多个使用者。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 队列发送一条消息，并为每个使用者收到消息的副本。启用后，应在每个消费者端点上指定这个选项。	false	布尔值

Name	描述	默认值	类型
pollTimeout (consumer (advanced))	轮询时使用的超时时间（以毫秒为单位）。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让消费者在关闭时更快响应。	1000	int
purgeWhenStopping (consumer (advanced))	在停止消费者/路由时是否清除任务队列。这样可以更快地停止，因为队列上的任何待处理消息都会被丢弃。	false	布尔值
blockWhenFull (producer)	是否向完整 SEDA 队列发送消息的线程是否被阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出异常，说明队列已满。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)	当发送到没有活跃消费者的队列时，生成者是否应丢弃消息（不要向队列添加消息）。只有其中一个选项 discardIfNoConsumers 和 failIfNoConsumers 可以同时启用。	false	布尔值
discardWhenFull (producer)	是否丢弃向完整 SEDA 队列发送消息的线程。默认情况下，会抛出异常，说明队列已满。通过启用此选项，调用线程将提供发送并继续，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，生成者是否应该失败。只有其中一个选项 discardIfNoConsumers 和 failIfNoConsumers 可以同时启用。	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)	当队列已满时，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)	在 SEDA producer 停止等待异步任务完成前，超时（以毫秒为单位）。您可以使用 0 或负值禁用超时。	30000	long

Name	描述	默认值	类型
waitForTaskToComplete (producer)	指定调用者是否应该等待 async 任务完成的选项，然后再继续。支持以下三个选项： Always、Never 或 IfReplyExpected。前两个值是自解释。最后的值 IfReplyExpected 将仅在消息基于 Request Reply 时才会等待。默认选项为 IfReplyExpected。 Enum 值： • Never • IfReplyExpected • Always	IfReplyExpected	WaitForTaskToComplete
queue (advanced)	定义端点要使用的队列实例。		BlockingQueue

67.5. 例子

以下是一些 stubbing 端点 uri 示例

```
stub:smtp://somehost.foo.com?user=whatnot&something=else
stub:http://somehost.bar.com/something
```

67.6. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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

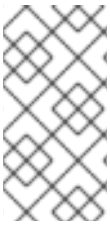
Name	描述	默认值	类型
camel.component.stub.bridge-error-handler	允许将消费者桥接到 Camel 路由错误处理程序，这意味着当消费者试图选择传入消息或类似信息时发生异常，现在将作为消息处理并由路由 Error Handler 处理。默认情况下，使用者将使用 org.apache.camel.spi.ExceptionHandler 来处理例外情况，该处理程序将被记录在 WARN 或 ERROR 级别，并忽略。	false	布尔值
camel.component.stub.concurrent-consumers	设置默认并发线程处理交换数。	1	整数
camel.component.stub.default-block-when-full	是否向完整 SEDA 队列发送消息的线程是否被阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出异常，说明队列已满。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	是否丢弃向完整 SEDA 队列发送消息的线程。默认情况下，会抛出异常，说明队列已满。通过启用此选项，调用线程将提供发送并继续，这意味着消息没有发送到 SEDA 队列。	false	布尔值
camel.component.stub.default-offer-timeout	是否向完整 SEDA 队列发送消息的线程是否被阻止，直到队列的容量不再耗尽为止。默认情况下，会抛出异常，说明队列已满。通过启用这个选项，可将配置的超时添加到块问题单中。利用下线 java 队列的 .offer (timeout)方法。		Long
camel.component.stub.default-poll-timeout	轮询时使用的超时时间（以毫秒为单位）。发生超时时，使用者可以检查是否允许继续运行。设置较低值可让消费者在关闭时更快响应。	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 68 章 电话报

支持生成者和消费者

平均报图组件提供对站报 Bot API 的访问。它允许基于 Camel 的应用程序通过充当 Bot 来发送和接收消息，参与与普通用户、私有和公共组或频道直接对话。

在使用此组件之前，必须创建 mustgram Bot，请按照 实验室gram Bot 开发人员 主页中的说明创建。创建新 Bot 时，B botFather 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。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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

68.1. URI 格式

```
telegram:type[?options]
```

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

Tailoringgram 组件支持 7 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
baseUri (advanced)	可用于设置替代的基本 URI，例如，您想要针对模拟图 API 测试组件。		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)	在端点中没有提供信息时，要使用的默认 报授权令牌。		字符串

68.4. 端点选项

Tailoringgram 端点使用 URI 语法进行配置：

```
telegram:type
```

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
type (common)	**必需** 端点类型。目前，只支持 'bots' 类型。 Enum 值： • bots		字符串

68.4.2. 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以接收的更新数量。	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）。为简短轮询放置 0，或为长时间轮询设置较大的数字。长时间轮询会产生较短的响应时间。	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
chatId (producer)	将接收生成的消息的 chat 的标识符。先从传入消息获取 chat id （例如，当电话报用户与 bot 开始对话时，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 chat id 可以动态为每个传出消息（使用正文或标头）进行动态设置。		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
baseUri (advanced)	可用于设置替代的基本 URI，例如，您想要针对模拟图 API 测试组件。		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
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
authorizationToken (security)	**必需** 使用 bot 的授权令牌（为 BotFather 掩码）。		字符串

68.4.3. 消息标头

Name	描述
CamelTelegramChatId	producer 端点使用此标头来解析将接收消息的 chat id。接收者 chat id 可以放在消息正文中（按优先级顺序）放在 **CamelTelegramChatId** 标头或端点配置(**chatId** 选项)。此标头也出现在所有传入消息中。
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大写或标题。
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**）。

68.5. 使用方法

Tailoringgram 组件支持 consumer 和 producer 端点。它还可在 reactive chat-bot 模式 （使用，然后生成消息）中使用。

68.6. 生成者示例

以下是如何通过 slirpgram Bot API 发送消息的基本示例。

in Java DSL

```
from("direct:start").to("telegram:bots?
authorizationToken=123456789:insertYourAuthorizationTokenHere");
```

或者在 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。

在不指定 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题中包含的信息来识别目标 chat。以下消息正文是允许生成者端点（即 OutgoingXXXMessage 类型的消息属于 org.apache.camel.component.telegram.model）

Java 类型	描述
OutgoingTextMessage	将文本消息发送到 chat
OutgoingPhotoMessage	将取消部署(JPG、PNG)发送到一个天
OutgoingAudioMessage	将 mp3 音频发送到 chat
OutgoingVideoMessage	向天发送 mp4 视频
OutgoingDocumentMessage	向天发送文件（任何介质类型）
OutgoingStickerMessage	将粘滞者发送到一个 chat (WEBP)
OutgoingAnswerInlineQuery	向内联查询发送回答
EditMessageTextMessage	编辑文本和游戏消息(editMessageText)
EditMessageCaptionMessage	编辑消息上限(editMessageCaption)

Java 类型	描述
EditMessageMediaMessage	编辑动画、音频、文档、语音或视频消息。(editMessageMedia)
EditMessageReplyMarkupMessage	要只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	要在 live_period 过期(stopMessageLiveLocation)前停止更新 bot 或 bot （用于内联 bot）发送的实时位置消息。
SendVenueMessage	发送有关 venue （发送）的信息
byte[]	发送支持的任何介质类型。它要求将 **CamelTelegramMediaType** 标头设置为适当的介质类型
字符串	向天发送文本消息，请执行以下操作：它会自动转换为 **OutgoingTextMessage**

68.7. 消费者示例

以下是如何接收电话报用户发送到配置的 Bot 的所有消息的基本示例。In Java DSL

```
from("telegram:bots?authorizationToken=123456789:insertYourAuthorizationTokenHere")
.bean(ProcessorBean.class)
```

或者在 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
    }

}
```

传入信息支持的类型有

Java 类型	描述
IncomingMessage	传入消息的完整对象表示
字符串	消息的内容，仅用于文本消息

68.8. 重新主动 CHAT-BOT 示例

reactive chat-bot 模式是使用 Camel 组件构建简单的 chat bot 的一种简单方式，它直接回复来自 gramgram 用户收到的 chat 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");
```

或者在 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 是一种简单的 bean，它实施通用字符串到字符串的方法。

```
public class ChatBotLogic {

    public String chatBotProcess(String message) {
```

```
        if( "do-not-reply".equals(message) ) {
            return null; // no response in the chat
        }

        return "echo from the bot: " + message; // echoes the message
    }

}
```

chatBotProcess 方法返回的每个非空字符串会自动路由到源自请求的 chat （因为 CamelTelegramChatId 标头用于路由消息）。

68.9. 获取 CHAT ID

如果您要在事件发生时将消息推送到特定的 guestfishgram chat，则需要检索对应的 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获取它。

首先，将 bot 添加到您要推送消息的 chat 中，然后运行如下路由：

```
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中，以及有关 chat (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其中。

```
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。

68.10. 自定义键盘

您可以自定义用户键盘，而不是要求他写一个选项。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 Markup。

```
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");
```

68.11. 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程序中添加 REST 实现。例如，Maven 用户可以将 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 添加到您要使用的电话报 URI 中。

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，将将配置为向它们发送消息。您需要确保您的服务器公开到互联网，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。

68.12. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用电话图时，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>
```

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	在端点中没有提供信息时，要使用的默认 报授权令牌。		字符串

Name	描述	默认值	类型
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，您想要针对模拟图 API 测试组件。		字符串
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。选项是一个 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。选项是一个 org.asynchttpclient.AsyncHttpClientConfig 类型。		AsyncHttpClientConfig
camel.component.telegram.enabled	是否启用电话报组件的自动配置。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 69 章 TIMER

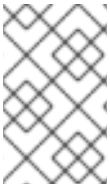
仅支持消费者

Timer 组件用于在计时器触发时生成消息交换，您只能消耗来自此端点的事件。

69.1. URI 格式

```
timer:name[?options]
```

其中 name 是 Timer 对象的名称，它在端点之间创建和共享。因此，如果您对所有计时器端点使用相同的名称，则只使用一个 Timer 对象和线程。



注意

生成的交换的 IN 正文为 null。So exchange.getIn().getBody() returns 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。

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

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

69.4. 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：

69.4.1. 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

69.4.2. 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
delay (consumer)	触发第一个事件前的延迟。	1000	long
fixedRate (consumer)	事件以大约常规的间隔进行，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、计时器计数等。此信息默认包含。	true	布尔值
period (consumer)	如果大于 0，请在每次期间生成定期事件。	1000	long
repeatCount (consumer)	指定触发的最大数量。因此，如果您将其设置为 1，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

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
守护进程 (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)	允许您指定用于使用 URI 语法设置 time 选项的自定义日期模式。		字符串
Sync (advanced)	设置是否应严格使用同步处理。	false	布尔值
时间 （高级）	应生成第一个事件的 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。		timer

69.5. 交换属性

触发计时器时，它会将以下信息作为属性添加到 Exchange 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。

69.6. 示例

要设置每 60 秒生成事件的路由：

```
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 中名为 myBean 的 bean 上调用 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>
```

69.7. 尽快触发

从 Camel 2.17 开始

您可能希望尽快在 Camel 路由中触发消息，您可以使用负延迟：

```
<route>
  <from uri="timer://foo?delay=-1"/>
  <to uri="bean:myBean?method=someMethodName"/>
</route>
```

这样，计时器会立即触发消息。

您还可以指定一个 repeatCount 参数以及负延迟，以在达到固定数字后停止触发消息。

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。

69.8. 仅触发一次

您可能希望仅在 Camel 路由中只触发一次消息，例如在启动路由时。要做到这一点，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>
```

69.9. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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
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 70 章 验证器

仅支持生成者

Validation 组件使用 JAXP Validation API 和任何受支持的 XML 模式语言（默认为 XML 架构）执行消息正文的 XML验证

请注意，组件还支持以下有用的模式语言：

- RelaxNG Compact 语法
- RelaxNG XML 语法

MSV 组件还支持 RelaxNG XML 语法。

70.1. URI 格式

```
validator:someLocalOrRemoteResource
```

其中 someLocalOrRemoteResource 是 classpath 上本地资源的一些 URL，或包含要验证的 XSD 文件系统上的远程资源或资源的完整 URL。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。

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

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

70.4. 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：

70.4.1. 路径参数(1 参数)

Name	描述	默认值	类型
resourceUri (producer)	对 classpath 上的本地资源 **所需的** URL，或在 Registry 中查找 bean 的引用，或者包含要验证的 XSD 文件系统上的远程资源或资源的完整 URL。		字符串

70.4.2. 查询参数(10 参数)

Name	描述	默认值	类型
failOnNullBody (producer)	如果没有正文，是否失败。	true	布尔值
failOnNullHeader (producer)	在针对标头验证时没有标头，是否会失败。	true	布尔值
headerName (producer)	针对标头而不是消息正文进行验证：		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
useSharedSchema (advanced)	Schema 实例是否应共享。引进了这个选项来解决 JDK 1.6.x 错误。Xerces 不应出现这个问题。	true	布尔值

70.5. 示例

*以下示例演示了如何 **配置来自端点 direct:start 的路由：start，然后进入两个端点之一，可以是 mock:valid 或 Mock:invalid，具体取决于 XML 是否与给定的模式匹配（在 classpath 上提供）。***

70.6. 高级：JMX 方法清除CACHEDSCHEMA

您可以强制在验证器端点中缓存的模式被清除，并使用 JMX 操作 清除 下一个进程调用来重新读取。您还可以使用此方法以编程方式清除缓存。此方法可用于 ValidatorEndpoint 类。

70.7. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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。选项是一个 org.apache.camel.component.validator.ValidatorResourceResolverFactory 类型。		ValidatorResourceResolverFactory

第 71 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：

- 电话报

Maven 用户可以将以下依赖项添加到它们的 pom.xml 中。

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

通常，支持 Webhook 的其他组件将带来此依赖项。

71.1. URI 格式

```
webhook:endpoint[?options]
```

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

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(base) path 元素。最好将其设置为随机字符串，因此未授权方无法猜测它。		字符串
webhookComponentName (consumer)	用于 REST 传输的 Camel Rest 组件，如 netty-http。		字符串
webhookExternalUrl (consumer)	Webhook 供应商可以看到的当前服务的 URL。		字符串
webhookPath (consumer)	公开 Webhook 端点的路径（如果存在的话）。		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
配置 （高级）	设置 webhook meta-component 的默认配置。		WebhookConfiguration

71.4. 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：

71.4.1. 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

71.4.2. 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(base) path 元素。最好将其设置为随机字符串，因此未授权方无法猜测它。		字符串
webhookComponentName (consumer)	用于 REST 传输的 Camel Rest 组件，如 netty-http。		字符串
webhookExternalUrl (consumer)	Webhook 供应商可以看到的当前服务的 URL。		字符串
webhookPath (consumer)	公开 Webhook 端点的路径（如果存在的话）。		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

71.5. 例子

webhook 组件示例包括在支持它的委派组件文档中。

71.6. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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(base) path 元素。最好将其设置为随机字符串，因此未授权方无法猜测它。		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

Name	描述	默认值	类型
camel.component.webhook.webhook-external-url	Webhook 供应商可以看到的当前服务的 URL。		字符串
camel.component.webhook.webhook-path	公开 Webhook 端点的路径（如果存在的话）。		字符串

第 72 章 XSLT

仅支持生成者

XSLT 组件允许您使用 XSLT 模板处理消息。当使用 Templating 来为请求生成响应时，这可能是理想的选择。

72.1. 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 72.1. 表 1。URI 示例

URI	描述
xslt:com/acme/mytransform.xsl	指的是 classpath 上的 com/acme/mytransform.xsl 文件
xslt:file:///foo/bar.xsl	引用文件 /foo/bar.xsl
xslt:http://acme.com/cheese/foo.xsl	引用远程 http 资源

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

XSLT 组件支持 7 个选项，如下所列。

Name	描述	默认值	类型
contentCache (producer)	加载时资源内容的缓存（样式表文件）。如果设置为 false Camel，则会在每个消息处理上重新加载样式表文件。这对开发非常有用。可使用 clearCachedStylesheet 操作强制通过 JMX 在运行时重新载入缓存的样式表。	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 转换工厂，请指定为 FQN 类名称。		字符串
transformerFactoryConfigurationStrategy (advanced)	在新创建的 TransformerFactory 实例上应用的配置策略。		TransformerFactoryConfigurationStrategy
uriResolver (advanced)	使用自定义 UriResolver。不应与选项 'uriResolverFactory' 一起使用。		URIResolver
uriResolverFactory (advanced)	使用自定义 UriResolver，它依赖于动态端点资源 URI。不应与选项 'uriResolver' 一起使用。		XsltUriResolverFactory

72.4. 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：

72.4.1. 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。您可以使用前缀：classpath, file, http, ref, 或 bean. classpath, 文件和 http 使用这些协议(classpath 为 default)。ref 将查找 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

72.4.2. 查询参数(13 参数)

Name	描述	默认值	类型
contentCache (producer)	加载时资源内容的缓存（样式表文件）。如果设置为 false Camel，则会在每个消息处理上重新加载样式表文件。这对开发非常有用。可使用 clearCachedStylesheet 操作强制通过 JMX 在运行时重新载入缓存的样式表。	true	布尔值
deleteOutputFile (producer)	如果您有 output=file，则此选项指定在交换完成处理时是否应该删除输出文件。例如，假设输出文件是一个临时文件，最好在使用后将其删除。	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
output (producer)	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中，因为文件直接流传输到 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，必须先创建指向文件名的任何路径，否则在运行时会抛出异常。 Enum 值： • 字符串 • bytes • DOM • file	字符串	XsltOutput

Name	描述	默认值	类型
transformerCacheSize (producer)	用于重复使用的 javax.xml.transform.Transformer 对象的数量，以避免调用 Template.newTransformer（）。	0	int
entityResolver (advanced)	使用带有 javax.xml.transform.sax.SAXSource 的自定义 org.xml.sax.EntityResolver。		EntityResolver
errorListener (advanced)	允许配置使用自定义 javax.xml.transform.ErrorListener。在执行此操作时，请注意默认错误监听程序，它会捕获任何错误或严重错误，并在交换上存储信息，因为属性没有被使用。因此，仅将这个选项用于特殊用例。		ErrorListener
resultsHandlerFactory (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 转换工厂。		TransformerFactory
transformerFactoryClass (advanced)	要使用自定义 XSLT 转换工厂，请指定为 FQN 类名称。		字符串
transformerFactoryConfigurationStrategy (advanced)	在新创建的 TransformerFactory 实例上应用的配置策略。		TransformerFactoryConfigurationStrategy
uriResolver (advanced)	使用自定义 javax.xml.transform.URIResolver。		URIResolver

72.5. 使用 XSLT 端点

以下格式是使用 XSLT 模板来公式 InOut 消息交换消息的响应（其中有一个 JMSReplyTo 标头）

```
from("activemq:My.Queue").
  to("xslt:com/acme/mytransform.xsl");
```

如果要使用 InOnly 并消耗信息并将其发送到另一个目的地，您可以使用以下路由：

```
from("activemq:My.Queue").
  to("xslt:com/acme/mytransform.xsl").
  to("activemq:Another.Queue");
```

72.6. 在 XSLT 中获取可使用的参数

默认情况下，所有标头都添加为 XSLT 中可用的参数。
要使参数可以使用，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 的顶层声明，以便它可用：

```
<xsl: ..... >

  <xsl:param name="myParam"/>

   <xsl:template ...>
```

72.7. SPRING XML 版本

要使用 Spring XML 中的上述示例，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>
```

72.8. 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 文件将相对于启动端点。

```
<xsl:include href="staff_template.xsl"/>
```

这意味着 Camel 将在 classpath 中找到文件，存为
org/apache/camel/component/xslt/staff_template.xsl

您可以使用 classpath: 或 file: 来指示 Camel 在 classpath 或文件系统中查找。如果省略了前缀，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 路径中直接引用。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"/>
```

72.9. 使用 XSL:INCLUDE 和默认前缀

Camel 将使用端点配置的前缀作为默认前缀。

您可以明确指定 file: 或 classpath: loading。如果需要，可以在 XSLT 脚本中混合两种加载类型。

72.10. 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。

72.11. 从 XSLT ERRORLISTENER 访问警告、错误和严重错误

任何警告/错误或严重错误都以带有密钥 Exchange.XSLT_ERROR、sExchange.XSLT_FATAL_ERROR 或 Exchange.XSLT_WARNING 的属性形式存储在当前交换中，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，如果员工有一个空的 dob 字段，则希望终止。以及使用 xsl:message 的自定义错误消息。

```
<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>
```

这个异常以带有密钥 Exchange .XSLT_WARNING 的警告形式存储在 Exchange 上。

72.12. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	加载时资源内容的缓存（样式表文件）。如果设置为 false Camel，则会在每个消息处理上重新加载样式表文件。这对开发非常有用。可使用 clearCachedStylesheet 操作强制通过 JMX 在运行时重新载入缓存的样式表。	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
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 转换工厂，请指定为 FQN 类名称。		字符串

Name	描述	默认值	类型
camel.component.xslt.transformer-factory-configuration-strategy	在新创建的 TransformerFactory 实例上应用的配置策略。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。不应与选项 'uriResolverFactory' 一起使用。选项是一个 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。不应与选项 'uriResolver' 一起使用。选项是一个 org.apache.camel.component.xslt.XsltUriResolverFactory 类型。		XsltUriResolverFactory

第 73 章 AVRO

此组件为 avro 提供数据格式，允许使用 Apache Avro 的二进制数据格式序列化和反序列化消息。由于 Camel 3.2 rpc 功能已移到单独的 camel-avro-rpc 组件中。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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

您可以使用 maven 和 ant 等从 schema 中轻松生成类。如需更多信息，请参阅 Apache Avro 文档。

73.1. AVRO DATAFORMAT 选项

Avro 数据格式支持 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。

73.2. AVRO 数据格式用法

使用 avro 数据格式像在路由中指定要 marshal 或 unmarshal 的类一样简单。

```
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 是一个 Avro Maven 插件生成的类。

或者在 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 数据格式来 umarshal。

73.3. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 avro 时，请确保使用以下 Maven 依赖项来支持自动配置：

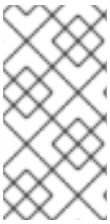
```
<dependency>
  <groupId>org.apache.camel.springboot</groupId>
  <artifactId>camel-avro-starter</artifactId>
</dependency>
```

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 74 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 unmarshal 有效负载 unmarshal Java 对象到 Avro 有效负载。



注意

如果您熟悉 Jackson，这个 Avro 数据格式的行为方式与其 JSON 对应部分相同，因此可用于注解用于 JSON 序列化/反序列化的类。

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

74.1. 配置 SCHEMARESOLVER

由于 Avro 序列化是基于架构的，因此这些数据格式要求您提供一个 SchemaResolver 对象，该对象可以查找每个交换的模式，这些交换将被所有 marshalled/unmarshalled。

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

74.2. AVRO JACKSON 选项

Avro Jackson 数据格式支持 18 个选项，如下所列。

Name	默认值	Java 类型	描述
objectMapper		字符串	在使用 Jackson 时，查找并使用给定 ID 的现有 ObjectMapper。
useDefaultObjectMapper		布尔值	是否查找并使用 registry 中的默认 Jackson ObjectMapper。
unmarshalType		字符串	取消总结时要使用的 java 类型的类名称。
jsonView		字符串	将 POJO 写入 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。这个选项是引用具有 JsonView 注解的类。

Name	默认值	Java 类型	描述
Include		字符串	如果要将 pojo 重命名为 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，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。
collectionType		字符串	指的是要在 registry 中查找的自定义集合类型。应很少使用这个选项，但允许使用与 java.util.Collection 不同的集合类型作为默认值。
useList		布尔值	要取消选择映射列表或 Pojo 列表。
moduleClassNames		字符串	使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为带有 FQN 类名称的 String。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。功能应该是与 com.fasterxml.jackson.databind.SerializationFeature、com.fasterxml.jackson.databind.DeserializationFeature 或 com.fasterxml.jackson.databind.MapperFeature 多功能中的 enum 匹配的名称。
disableFeatures		字符串	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的功能集合。功能应该是与 com.fasterxml.jackson.databind.SerializationFeature、com.fasterxml.jackson.databind.DeserializationFeature 或 com.fasterxml.jackson.databind.MapperFeature 多功能中的 enum 匹配的名称。
allowUnmarshallType		布尔值	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。这只在需要使用时才启用。
timezone		字符串	如果设置，Jackson 会在 marshalling/unmarshalling 时使用 Timezone。
autoDiscoverObjectMapper		布尔值	如果设置为 true，Jackson 会将对象映射器查找到 registry 中。

Name	默认值	Java 类型	描述
contentTypeHeader		布尔值	数据格式是否应使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 例如，marshalling 到 XML，对于数据格式为 JSON，application/json 用于数据格式。
schemaResolver		字符串	可选的 schema 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 registry。

74.3. 使用自定义 AVROMAPPER

如果需要更多控制映射配置，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。

74.4. 依赖项

要在 camel 路由中使用 Avro Jackson，您需要对实现此数据格式的 camel-jackson-avro 添加依赖项。

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

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

74.5. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 avro-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-avro-starter</artifactId>
</dependency>
```

–

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，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。	false	布尔值
camel.dataformat.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，Jackson 会将对象映射器查找到 registry 中。	false	布尔值
camel.dataformat.avro-jackson.auto-discover-schema-resolver	如果没有禁用，SchemaResolver 将查找 registry。	true	布尔值
camel.dataformat.avro-jackson.collection-type	指的是要在 registry 中查找的自定义集合类型。应很少使用这个选项，但允许使用与 java.util.Collection 不同的集合类型作为默认值。		字符串
camel.dataformat.avro-jackson.content-type-header	数据格式是否应使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 例如，marshalling 到 XML，对于数据格式为 JSON，application/json 用于数据格式。	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。功能应该是与 com.fasterxml.jackson.databind.SerializationFeature、com.fasterxml.jackson.databind.DeserializationFeature 或 com.fasterxml.jackson.databind.MapperFeature 多功能中的 enum 匹配的名称。		字符串

Name	描述	默认值	类型
camel.dataformat.avro-jackson.enable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上启用的功能集合。功能应该是与 com.fasterxml.jackson.databind.SerializationFeature、com.fasterxml.jackson.databind.DeserializationFeature 或 com.fasterxml.jackson.databind.MapperFeature 多功能中的 enum 匹配的名称。		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	如果要将 pojo 重命名为 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	将 POJO 写入 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。这个选项是引用具有 JsonView 注解的类。		字符串
camel.dataformat.avro-jackson.module-class-names	使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为带有 FQN 类名称的 String。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
camel.dataformat.avro-jackson.object-mapper	在使用 Jackson 时，查找并使用给定 ID 的现有 ObjectMapper。		字符串
camel.dataformat.avro-jackson.schema-resolver	可选的 schema 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，Jackson 会在 marshalling/unmarshalling 时使用 Timezone。		字符串

Name	描述	默认值	类型
camel.dataformat.avro-jackson.unmarshal-type	取消总结时要使用的 java 类型的类名称。		字符串
camel.dataformat.avro-jackson.use-default-object-mapper	是否查找并使用 registry 中的默认 Jackson ObjectMapper。	true	布尔值
camel.dataformat.avro-jackson.use-list	要取消选择映射列表或 Pojo 列表。	false	布尔值

第 75 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/绑定到使用注解定义的绑定映射的 Java Beans。使用 Bindy，您可以绑定来自源的数据，如：

- *CSV 记录，*
- *固定长度记录，*
- *FIX 消息，*
- *或几乎任何其他非结构化数据*

至一个或多个 Plain Old Java 对象(POJO)。bindy 根据 java 属性的类型转换数据。在某些情况下，POJO 可以与一对多的关系相关联。此外，对于如 Date, Double, Float, Integer, Short, Long 和 BigDecimal 等数据类型，您可以提供在属性格式化过程中要应用的模式。

对于 BigDecimal 号，您还可以定义精度和十进制或分组分隔符。

类型	格式类型	模式示例	Link
Date	**DateFormat**	**dd-MM-yyyy**	https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html
微小	**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）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绑定时，您可以将多个模型放在同一个软件包中。

75.1. 选项

Bindy dataformat 支持 5 个选项，如下所列。

Name	默认值	Java 类型	描述
type		**Enum**	**是否需要使用** Csv、fixed 或 KeyValue。 Enum 值： • csv • 已修复 • KeyValue
classType		**字符串**	要使用的模型类的名称。
locale		**字符串**	要配置要使用的默认区域设置，如我们用于单元状态。要使用 JVM 平台默认区域设置，请使用名称 default。
unwrapSingleInstance		**布尔值**	当 unmarshalling 应该取消嵌套并返回时，而不是嵌套到 java.util.List 中。
allowEmptyStream		**布尔值**	是否在 unmarshal 进程中允许空流。如果为 true，当提供没有记录的正文时，不会抛出任何异常。

75.2. 注解

创建的注解允许将不同模型概念映射到 POJO，如下所示：

- *记录类型(CSV、键值对（如 FIX 消息)、固定 length ...）*
- *链接（到另一个对象中的链接对象）*
- *DataField 及其属性(int、 type、 ...)*
- *KeyValuePairField （用于 key = value format，如 FIX 交易消息），*
- *部分（用于标识标头、正文和页脚部分）*
- *OneToMany,*
- *BindyConverter,*
- *FormatFactories*

本节将描述它们。

75.2.1. 1.CsvRecord

CsvRecord 注解用于识别模型的根类。它代表一个 record = "CSV 文件的一行"，并可链接到几个子模型类。

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

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标签(\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具有特殊含义的符号，如 '|' 符号，则必须屏蔽它，如 '|'。
allowEmptyStream	布尔值		false	allowEmptyStream 参数允许调整 CSV 文件的不密集型流。
autospanLine	布尔值		false	最后记录跨越其余行（可选）- 如果启用，则最后一列会自动到行尾，例如，如果其注释，则允许行包含所有字符，还包含分隔符。
CRLF	字符串		WINDOWS	用于在每个记录（可选）后添加cariage 返回的字符 - 允许定义要使用的 carriage 返回字符。如果您指定了之前列出的三个值，则输入的值(custom)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 文件应该以换行符或不结束（可选）结尾，则 endWithlineBreak 参数标志
generateHeaderColumns	布尔值		false	generateHeaderColumns 参数允许在 CSV 中添加，生成包含列名称的标头
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		"	是否使用给定引号字符（可选） marshal 列 - 允许在生成 CSV 时指定字段的引号字符。此注解与模型的根类关联，必须一次声明。
quoting	布尔值		false	指定在 marshaling 时是否必须加引号值（可选）
quotingEscaped	布尔值		false	指定在引用时是否必须转义值（可选）
removeQuotes	布尔值		true	如果 unmarshalling 应该尝试删除每个字段的引号，则 remove quote 参数标记
skipField	布尔值		false	skipField 参数将允许跳过 CSV 文件的字段。如果不需要一些字段，可以跳过它们。
skipFirstLine	布尔值		false	skipFirstline 参数将允许跳过 CSV 文件的第一行。这个行通常包含列定义

case 1 : separator = ','

用于隔离 CSV 记录中的字段的分隔符是 ：

> *10, J, Pauline, M, XD12345678, Fortis Dynamic 15/15, 2500, USD, 08-01-2009*

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

}
```

case 2 : separator = ';'

与上一个情况进行比较，这里的分隔符是 ; 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 = '|'

与上一个情况进行比较，这里的分隔符是 | 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 CSV 记录解析的字段包含 , 或 ; （也用作分隔符）时，应该找到另一个策略来告知 camel bindy 如何处理此问题单。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"）。

—	在这种情况下，作为单引号或双引号的行的前和最后一个字符将通过 bindy 删除。

```
"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来，并在从 CSV 到对象取消编译时自动删除这些引号。因此，不要在 分隔符中包含引号，但如下所示：

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

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

}
```

请注意，如果要从对象到 CSV 并使用引号，则需要使用 @CsvRecord 上的 quote 属性来指定要使用的字符，如下所示：

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

}
```

问题单 5：分隔符和跳过行

当客户端想要在文件的第一行中时，这个功能值得注意，数据字段的名称：

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

要告知绑定此第一行必须在解析过程中跳过，然后我们使用属性 ：

```
@CsvRecord(separator = ",", skipFirstLine = true)
```

```
public Class Order {

}
```

case 6 : generateHeaderColumns

要在生成的 CSV 的第一行中添加，在注解中必须将属性 generateHeaderColumns 设置为 true，如下所示：

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

}
```

因此，在 unmarshaling 过程中绑定将生成 CSV，如下所示：

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

问题单 7：cariage 返回

如果运行 camel-bindy 的平台不是 Windows，但 Macintosh 或 Unix，您可以更改 crlf 属性，如下所示。三个值可用：WINDOWS、UNIX 或 MAC

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

}
```

另外，如果您需要添加不同的行结尾字符，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尾行，后跟换行符：

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

}
```

case 8 : isOrdered

有时，从模型创建 CSV 记录期间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 来将这个属性与 DataField 注解的属性 位置 结合使用。

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

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

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

}
```

__	**pos** 用于解析文件流，而 **位置** 则用于生成 CSV。

75.2.2. 2.Link

link 注解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	标识类间关系的链接类型

从当前版本中，只允许一对一的关系。

例如：如果模型类客户端链接到 Order 类，请在 Order 类中使用注解链接，如下所示：

属性链接

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

  @DataField(pos = 1)
  private int orderNr;
```

```
    @Link
    private Client client;
}
```

对于类客户端：

类链接

```
@Link
public class Client {

}
```

75.2.3. 3.DataField

DataField 注释定义字段的属性。每个 datafield 都由记录中的位置、类型（字符串、int、date、...）以及可选的模式来标识。

注解名称	记录类型	级别
DataField	all	属性

参数名称	类型	必填	默认值	info
pos	int	✓		在输入记录中数据的位置，必须从 1 开始（必需）。请参阅 position 参数。
校准	字符串		R	将文本与右或左处对齐。使用 <tt>R</tt> 或 <tt>L</tt> 的值。
Clip	布尔值		false	如果在使用固定长度时超过允许的长度，则指示在字段中清除数据。
columnName	字符串			标题栏的名称（可选）。使用属性的名称作为默认值。仅在 **CsvRecord** 具有 **generateHeaderColumns = true**时才适用
decimalSeparator	字符串			与数字一起使用的十进制 9 月
defaultValue	字符串			如果没有设置值，则字段的默认值

参数名称	类型	必填	默认值	info
delimiter	字符串			如果字段具有变量长度，则使用可选分隔符
groupingSeparator	字符串			当我们想将/解析分成带有分组（例如 123,456.789 ）的数字时，将分号号分组到数字中
impliedDecimalSeparator	布尔值		false	指明在指定位置是否表示十进制点
length	int		0	如果记录设置为固定长度，则数据块的长度（字符数）
lengthPos	int		0	标识记录中为此字段定义预期固定长度的 data 字段
方法	字符串			调用在 DataField 上应用此类自定义的方法名称。这必须是 datafield 本身的方法，或者您必须提供类方法的静态完全限定名称，例如：查看 test 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 char to pad
pattern	字符串			Java 格式化器（示例为SimpleDateFormat）的模式将用于转换数据（可选）。如果使用模式，则建议在绑定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）必须与输入位置（可能）进行比较时，必须使用。请参阅 pos 参数。
精度	int		0	要创建的 \{@link java.math.BigDecimal} 编号的精度
required	布尔值		false	指明字段是否是必需的
舍入	字符串		CEILING	用于舍入/scale 定制值 : UP, DOWN, CEILING, FLOOR, HALF_UP, HALF_UP, HALF_DOWN,HALF_EVEN, UNNECESSARY e.g : Number = 123456.789, Precision = 2, Rounding = CEILING Result : 123456.79

参数名称	类型	必填	默认值	info
timezone	字符串			要使用的时区。
trim	布尔值		false	指明值是否应修剪

问题单 1：pos

此参数/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;

}
```

如本例中看到的那样，位置从 1 开始，但从类顺序中的 5 开始。类客户端中定义了从 2 到 4 的数字（请参阅 之后）。

位置继续在另一个模型类中

```
public class Client {

    @DataField(pos = 2)
    private String clientNr;

    @DataField(pos = 3)
    private String firstName;

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

问题单 2：模式

该模式可用于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;
}
```

问题单 3：精度

当您要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

location 属性将告知绑定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 进程与 unmarshaling 的 marshaling 的比较），则我们可以使用 位置 来指示这一点。

下面是一个示例：

位置在输出中有所不同

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

    // Positions of the fields start from 1 and not from 0

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

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

    @DataField(pos = 3, position = 9)
    private String firstName;

    @DataField(pos = 4, position = 8)
    private String lastName;

    @DataField(pos = 5, position = 7)
    private String instrumentCode;
```

```
    @DataField(pos = 6, position = 6)
    private String instrumentNumber;
}
```

注释 @DataField 的此属性必须与注解 @CsvRecord 的属性 isOrdered = true 结合使用。

问题单 5：必需

如果字段是必需的，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，则解析器将引发一个错误，其中包含以下信息：

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

case 6 : trim

如果字段有前导和/或尾随空格，则应在处理前删除它们，只需使用属性 trim set 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;
}
```

问题单 7 : defaultValue

如果没有定义字段，则使用 defaultValue 属性表示的值。

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;
}
```

问题单 8 : columnName

仅在 @CsvRecord 带有注解 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。

75.2.4. 4.FixedLengthRecord

FixedLengthRecord 注解用于识别模型的根类。它代表一个 record = "一个文件/消息行，其中包含数据固定长度（字符数）格式，并可链接到几个子模型类。这种格式是特定于位的，因为字段的数据可以与右侧或左处一致。

当数据大小没有完全填写字段长度时，我们可以添加"平板"字符。

注解名称	记录类型	级别
FixedLengthRecord	FIXED	类

参数名称	类型	必填	默认值	info
countGrapheme	布尔值		false	指明如何计算计费
CRLF	字符串		WINDOWS	用于在每个记录后（可选）添加cariage 返回的字符。可能的值：WINDOWS、UNIX、MAC 或 custom。这个选项仅在 marshalling 期间使用，unmarshalling 使用系统默认的 JDK 提供行分隔符，除非自定义 eol。
EOL	字符串			在未记录时考虑每个记录之后的字符（可选 - default: ""，它可帮助使用默认的 JDK 提供行分隔符），除非提供了任何其他值，否则此选项仅在 unmarshalling 时使用，其中 marshalling 使用系统默认提供的行分隔符，除非提供了任何其他值。
footer	类		void	表示此类型的记录后面可以跟随一个页记录（在文件的末尾）

参数名称	类型	必填	默认值	info
header	类		void	表示此类型的记录可以在文件开头的单个标头记录前面。
ignoreMissingChars	布尔值		false	指明是否忽略了太短行
ignoreTrailingChars	布尔值		false	表示在未记录/解析时，可以忽略超过最后一个映射文件的字符。此注解与模型的根类关联，必须一次声明。
length	int		0	记录的固定长度（字符数）。这意味着，记录始终为使用 \ {114paddingChar （）} 的长添加
name	字符串			描述记录的名称（可选）
paddingChar	char			平板的费用。
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）。

记录可能不是标头/页脚和主固定长度记录。

问题单 1：简单固定长度记录

这个简单示例演示了如何设计模型来解析/格式化固定消息

> *10A9PaulineMISINXD12345678BUYShare2500.45USD01-08-2009*

fixed-simple

```
@FixedLengthRecord(length=54, paddingChar=' ')
public static class Order {

    @DataField(pos = 1, length=2)
    private int orderNr;
```

```
    @DataField(pos = 3, length=2)
    private String clientNr;

    @DataField(pos = 5, length=7)
    private String firstName;

    @DataField(pos = 12, length=1, align="L")
    private String lastName;

    @DataField(pos = 13, length=4)
    private String instrumentCode;

    @DataField(pos = 17, length=10)
    private String instrumentNumber;

    @DataField(pos = 27, length=3)
    private String orderType;

    @DataField(pos = 30, length=5)
    private String instrumentType;

    @DataField(pos = 35, precision = 2, length=7)
    private BigDecimal amount;

    @DataField(pos = 42, length=3)
    private String currency;

    @DataField(pos = 45, length=10, pattern = "dd-MM-yyyy")
    private Date orderDate;
}
```

问题单 2：使用校准和 padding修复长度记录

这个更详细的示例演示了如何为字段定义对齐以及如何分配为 " here 的 padding 字符：

```
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;
}
```

问题单 3：字段 padding

有时，为记录定义的默认 padding不能应用于字段，因为我们有数字格式，其中我们想使用 '0' 而不是 '0' 而不是 '。在这种情况下，您可以在模型中使用 @DataField 上的属性 paddingChar 来设置此值。

```
10A9 PaulineM ISINXD12345678BUYShare000002500.45USD01-08-2009
```

fixed-padding-field

```
@FixedLengthRecord(length = 65, paddingChar = ' ')
public static class Order {

    @DataField(pos = 1, length = 2)
    private int orderNr;

    @DataField(pos = 3, length = 2)
    private String clientNr;

    @DataField(pos = 5, length = 9)
    private String firstName;

    @DataField(pos = 14, length = 5, align = "L")
```

```
    private String lastName;

    @DataField(pos = 19, length = 4)
    private String instrumentCode;

    @DataField(pos = 23, length = 10)
    private String instrumentNumber;

    @DataField(pos = 33, length = 3)
    private String orderType;

    @DataField(pos = 36, length = 5)
    private String instrumentType;

    @DataField(pos = 41, precision = 2, length = 12, paddingChar = '0')
    private BigDecimal amount;

    @DataField(pos = 53, length = 3)
    private String currency;

    @DataField(pos = 56, length = 10, pattern = "dd-MM-yyyy")
    private Date orderDate;
}
```

问题单 4：使用分隔符修复长度记录

固定长度记录有时在记录中以分隔的内容。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;
}
```

固定长度记录中的 pos 值可以选择使用正数、顺序值而不是精确列号来定义。

问题单 5：使用记录定义字段长度修复长度记录

有时，一个固定长度记录可能会包含一个字段，该字段定义同一记录中另一个字段的预期长度。在以下示例中，detect Number 字段值 的长度由记录中的 detect NumberLen 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;
}
```

问题单 6：使用标头和页脚修复长度记录

bindy 将发现配置为模型一部分的固定长度标头和页脚记录 - 如果注解类与主 @FixedLengthRecord 类或配置在一个配置的扫描软件包之一中存在。以下文本演示了两个固定长度记录，它们由标头记录和页脚记录阻止。

```
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;
}
```

问题单 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，它们包含比目标用例所需信息更多的信息。在这种情况下，可以跳过那些我们不需要的字段的声明和解析。为此，如果下一个声明字段的 pos 值超过最后一个解析字段的光标位置，则 Bindy 将跳过转发到记录中的下一个映射字段。对感兴趣的字段使用绝对 pos 位置（而不是 ordinal 值）会导致 Bindy 在两个字段之间跳过内容。

同样，可能不包括一些字段以外的任何内容是值得关注的。在这种情况下，您可以通过在 @FixedLengthRecord 声明上设置 ignoreTrailingChars 属性来告知 Bindy 来跳过除最后一个映射字段外的所有内容的解析。

```
@FixedLengthRecord(ignoreTrailingChars = true)
public static class Order {

    @DataField(pos = 1, length = 2)
    private int orderNr;

    @DataField(pos = 3, length = 2)
    private String clientNr;

    // any characters that appear beyond the last mapped field will be ignored

}
```

75.2.5. 5.消息

Message 注解用于识别模型的类，它们将包含键值对字段。这种类型的格式主要用于电信交换协议消息(FIX)。但是，此注解可用于通过键识别数据的任何其他格式。密钥对值通过分隔符相互分隔，可以是类似标签页的特殊字符(unicode 表示： \u0009)或标题起始(unicode 表示： \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，该类可以是 Order 类。这并不是必须的，但当您将 camel-bindy 与 camel-fix 结合使用时，这是基于快速Fix 项目的修复网关时非常有用。

注解名称	记录类型	级别
消息	键值对	类

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（必需）中分割值。可以是 '\u0001'、'\u0009'、'Evolution' 或 'anything'。
pairSeparator	字符串	✓		用于在令牌中分割键值对的对分隔符（必需）。可以是 '='、';' 或 'anything'。

参数名称	类型	必填	默认值	info
CRLF	字符串		WINDOWS	用于在每个记录后（可选）添加cariage 返回的字符。可能的值 = WINDOWS、UNIX、MAC 或 custom。如果您指定了之前列出的三个值，则输入的值(custom)将用作 CRLF 字符。
isOrdered	布尔值		false	指明消息必须在输出中排序。此注解与模型的消息类关联，必须一次声明。
name	字符串			描述消息的名称（可选）
type	字符串		FIX	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'

用于隔离 FIX 消息中的键值对字段的分隔符是 ASCII 01 字符或 unicode 格式 \u0001。必须再次转义此字符，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 ...
```

如何使用注解：

FIX - 消息

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

}
```

查看测试问题单

选项卡等 ASCII 字符 ... 无法在 WIKI 页面中显示。因此，有一个 camel-bindy 的测试问题单，以查看 FIX 消息看起来如何(https://github.com/apache/camel/blob/main/components/camel-bindy/src/test/data/fix/fix.txt)和 Order,Trailer, Header 类 (https://github.com/apache/camel/blob/main/components/camel-bindy/src/test/java/org/apache/camel/dataformat/bindy/model/fix/simple/Order.java)。

75.2.6. 6.KeyValuePairField

KeyValuePairField 注释定义键值对字段的属性。每个 KeyValuePairField 都由标签(= key)及其值关联、类型（字符串、int、date、...）标识，可选模式，如果需要该字段。

注解名称	记录类型	级别
KeyValuePairField	关键价值 - FIX	属性

参数名称	类型	必填	默认值	info
tag	int	✓		标签标识消息中的字段（必需）- 必须是唯一的
impliedDecimalSeparator	布尔值		false	Camel 2.11: 指示一个十进制点，代表在指定位置上代表的十进制点
name	字符串			字段的名称（可选）
pattern	字符串			格式器将用于转换数据的模式（可选）
position	int		0	生成的消息中的字段位置 - 当 FIX 消息中的键/标签的位置必须不同
精度	int		0	要创建的 BigDecimal 数的精度
required	布尔值		false	指明字段是否是必需的
timezone	字符串			要使用的时区。

问题单 1：标签

此参数表示消息中字段的键：

FIX 消息 - Tag

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

    @Link Header header;

    @Link Trailer trailer;

    @KeyValuePairField(tag = 1) // Client reference
    private String Account;

    @KeyValuePairField(tag = 11) // Order reference
    private String ClOrdId;

    @KeyValuePairField(tag = 22) // Fund ID type (Sedol, ISIN, ...)
    private String IDSource;

    @KeyValuePairField(tag = 48) // Fund code
    private String SecurityId;

    @KeyValuePairField(tag = 54) // Movement type ( 1 = Buy, 2 = sell)
    private String Side;

    @KeyValuePairField(tag = 58) // Free text
    private String Text;
}
```

问题单 2： 输出中的不同位置

如果我们放入 FIX 消息的 tag/keys 必须根据预定义的顺序进行排序，则使用注解 @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;
}
```

75.2.7. 7.部分

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

此注解只有一个属性/参数。

注解名称	记录类型	级别
部分	FIX	类

参数名称	类型	必填	默认值	info
number	int	✓		部分的数量

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 消息 - 第 - Footer

```
@Section(number = 3)
public class Trailer {

    @KeyValuePairField(tag = 10, position = 1)
    // CheckSum
    private int checkSum;

    public int getCheckSum() {
        return checkSum;
    }
```

75.2.8. 8.OneToMany

注解 @OneToMany 的目的是允许使用 List<?& gt; 字段定义 POJO 类或包含重复组的记录。

注意

一个ToMany 的限制会小心，到许多绑定者的限制不允许处理在多个层次结构级别上定义的重复副本。

在以下情况下，关系 oneToMany WORKS：

- *读取包含重复组的 FIX 消息(= 标签/密钥组)*
- *使用重复数据生成 CSV*

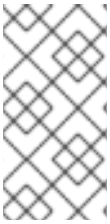
注解名称	记录类型	级别
OneToMany	all	属性

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：使用重复数据生成 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

重复数据涉及本书的标题及其发布日期，而第一个、姓氏和年龄是通用的，用于建模这一点的类。Author 类包含书签列表。

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;
}
```

问题单 2：读取包含标签/密钥组的 FIX 消息

以下是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 会被重复。

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;
}
```

75.2.9. 9.BindyConverter

注释 @BindyConverter 的目的是定义一个在字段级别使用的转换器。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();
    }
}
```

75.2.10. 10.FormatFactories

注释 @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);
            }
        };
    }
}
```

75.3. 支持的数据类型

DefaultFormatFactory 通过根据提供的 FormattingOptions 返回接口 FormatFactoryInterface 的实例来格式化以下数据类型：

- BigDecimal
- BigInteger
- 布尔值

- *BYTE*
- *字符*
- *Date*
- *å€Œ*
- *Enums*
- *æµ®ç‚¹å€¼*
- *整数*
- *LocalDate*
- *LocalDateTime*
- *LocalTime*
- *Long*
- *短*
- *字符串*

DefaultFormatFactory 可以通过在正在使用的 registry 中提供 FactoryRegistry 实例来覆盖（如 spring 或 JNDI）。

75.4. 使用 JAVA DSL

下一步实例化与此记录类型关联的 DataFormat 绑定 类，并提供类作为参数。

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

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

75.4.1. 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，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");
```

75.4.2. Unmarshaling

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

另外，您可以使用对数据格式的命名引用，然后在 Registry 中定义，例如 Spring XML 文件：

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

Camel 路由将提取 inbox 目录中的文件，将不marshall CSV 记录到模型对象的集合中，并将 collection
发送到 handleOrders 引用的路由。

返回的集合是 Map 对象的列表。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 每行可以对应多个对象。当您只期望每行返回一个对象时，这可能会混淆。

每个对象都可以使用其类名称来检索。

```
List<Map<String, Object>> unmarshaledModels = (List<Map<String, Object>>)
exchange.getIn().getBody();

int modelCount = 0;
for (Map<String, Object> model : unmarshaledModels) {
  for (String className : model.keySet()) {
    Object obj = model.get(className);
    LOG.info("Count : " + modelCount + ", " + obj.toString());
  }
 modelCount++;
}

LOG.info("Total CSV records received by the csv bean : " + modelCount);
```

假设您要从此映射中提取单个 Order 对象来处理路由，您可以按如下方式使用 Splitter 和 Processor 的组合：

```
from("file://inbox")
   .unmarshal(bindy)
   .split(body())
      .process(new Processor() {
         public void process(Exchange exchange) throws Exception {
            Message in = exchange.getIn();
            Map<String, Object> modelMap = (Map<String, Object>) in.getBody();
            in.setBody(modelMap.get(Order.class.getCanonicalName()));
         }
      })
      .to("direct:handleSingleOrder")
   .end();
```

请注意，Bindy 使用 CHARSET_NAME 属性或 CHARSET_NAME 标头（如 Exchange 接口中定义的 CHARSET_NAME 标头）进行为 unmarshalling 接收的输入流的字符集转换。在某些制作者（如 file-endpoint）中，您可以定义字符集。字符集转换可能已经由这个制作者完成。有时，您需要将此属性或标头从交换中删除，然后再将其发送到 unmarshal。如果您没有删除它，则可能会进行转换两次，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");
```

75.4.3. marshaling

要从模型对象集合中生成 CSV 记录，您可以创建以下路由：

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

75.5. 使用 SPRING XML

这实际上可轻松使用 Spring 作为您喜欢的 DSL 语言声明要用于 camel-bindy 的路由。以下示例显示了第一个从文件中提取记录的两个路由，取消选择内容并将其绑定到其模型。然后，结果会被发送到一个 pojo （不特殊）并将其放入队列中。

第二个路由将从队列中提取 pojos，并编译内容来生成包含 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现了序列化，否则队列管理器将引发错误。

75.6. 依赖项

要在 camel 路由中使用 Bindy，您需要添加实施此数据格式的 camel-bindy 的依赖关系。

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

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-bindy</artifactId>
  <version>{CamelSBVersion}</version>
</dependency>
```

75.7. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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，当提供没有记录的正文时，不会抛出任何异常。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.bindy-csv.class-type	要使用的模型类的名称。		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	要配置要使用的默认区域设置，如我们用于单元状态。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、fixed 或 KeyValue。		字符串
camel.dataformat.bindy-csv.unwrap-single-instance	当 unmarshalling 应该取消嵌套并返回时，而不是嵌套到 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，当提供没有记录的正文时，不会抛出任何异常。	false	布尔值
camel.dataformat.bindy-fixed.class-type	要使用的模型类的名称。		字符串
camel.dataformat.bindy-fixed.enabled	是否启用绑定修复的数据格式的自动配置。这默认是启用的。		布尔值
camel.dataformat.bindy-fixed.locale	要配置要使用的默认区域设置，如我们用于单元状态。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、fixed 或 KeyValue。		字符串
camel.dataformat.bindy-fixed.unwrap-single-instance	当 unmarshalling 应该取消嵌套并返回时，而不是嵌套到 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，当提供没有记录的正文时，不会抛出任何异常。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.bindy-kvp.class-type	要使用的模型类的名称。		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	要配置要使用的默认区域设置，如我们用于单元状态。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、fixed 或 KeyValue。		字符串
camel.dataformat.bindy-kvp.unwrap-single-instance	当 unmarshalling 应该取消嵌套并返回时，而不是嵌套到 java.util.List 中。	true	布尔值

第 76 章 HL7

HL7 组件用于使用 HAPI 库处理 HL7 MLLP 协议和 HL7 v2 消息。

这个组件支持以下内容：

- *Mina的 HL7 MLLP codec*
- *用于 Netty的 HL7 MLLP codec*
- *从/到 HAPI 和字符串输入 Converter*
- *使用 HAPI 库的 HL7 DataFormat*

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-hl7</artifactId>
  <version>{CamelSBVersion}</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

76.1. HL7 MLLP 协议

HL7 通常与 HL7 MLLP 协议一起使用，该协议是一个基于文本的基于 TCP 套接字的协议。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 有效负载的起始字节。

Name	默认值	描述
endByte1	**0x1c**	跨越 HL7 有效负载的第一个结束字节。
endByte2	**0x0d**	跨越 HL7 有效负载的第 2 个结束字节。
charset	JVM 默认	用于 codec 的编码(charset 名称)。如果没有提供，Camel 将使用 JVM 默认 Charset。
produceString	**true**	如果为 true，则 codec 会使用定义的 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 Data Format 可以决定 HL7 消息内容的实际 charset。
convertLFtoCR	**false**	将 **\n** 转换为 **\r** (**0x0d**, 13 十进制)，作为 HL7 stipulates**\r** 作为片段术语器。HAPI 库需要使用 **\r**。

76.1.1. 使用 Mina 公开 HL7 侦听程序

在 Spring XML 文件中，我们将一个 mina 端点配置为使用端口 8888 上的 TCP 侦听 HL7 请求：

```
<endpoint id="hl7MinaListener" uri="mina:tcp://localhost:8888?sync=true&amp;codec=#hl7codec"/>
```

sync=true 表示此监听程序是同步的，因此会将 HL7 响应返回给调用者。HL7 codec 使用 codec=114hl7codec 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>
```

端点 hl7MinaLlistener 可作为消费者在路由中使用，如这个 Java DSL 示例演示了：

```
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 looking 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
    }
```

76.1.2. 使用 Netty 公开 HL7 侦听器（可从 Camel 2.15 开始使用）

在 Spring XML 文件中，我们将一个 netty 端点配置为使用端口 8888 上的 TCP 侦听 HL7 请求：

```
<endpoint id="hl7NettyListener" uri="netty:tcp://localhost:8888?
sync=true&amp;encoders=#hl7encoder&amp;decoders=#hl7decoder"/>
```

sync=true 表示此监听程序是同步的，因此会将 HL7 响应返回给调用者。HL7 codec 是带有 encoders=114hl7encoder598andödecoders=114hl7decoder 的设置。请注意，hl7encoder 和 hl7decoder 只是 bean ID，因此可以以不同的方式命名。Bean 可以在 Spring XML 文件中设置：

```
<bean id="hl7decoder" class="org.apache.camel.component.hl7.HL7MLLPNettyDecoderFactory"/>
<bean id="hl7encoder" class="org.apache.camel.component.hl7.HL7MLLPNettyEncoderFactory"/>
```

然后，端点 hl7NettyListener 可作为消费者在路由中使用，如此 Java DSL 示例演示了：

```
from("hl7NettyListener")
  .bean("patientLookupService");
```

76.2. HL7 MODEL USING JAVA.LANG.STRING OR BYTE[]

HL7 MLLP codec 使用普通字符串作为其数据格式。Camel 使用其 Type Converter 将字符串转换为 HAPI HL7 模型对象，但如果您希望自行解析数据，则可以使用普通字符串对象。

您还可以通过将 produceString 属性设置为 false，让 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/从 HAPI HL7 模型对象转换。

76.3. 使用 HAPI 的 HL7V2 模型

HL7v2 模型使用 HAPI 库中的 Java 对象。使用这个库，您可以从大多数与 HL7v2 一起使用的 EDI 格式编码和解码。

以下示例是查找 ID 为 0101701234 的过期请求。

```
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，例如检索过期 ID：

```
Message msg = exchange.getIn().getBody(Message.class);
QRD qrd = (QRD)msg.get("QRD");
String patientId = qrd.getWhoSubjectFilter(0).getIDNumber().getValue(); // 0101701234
```

这在与 HL7 侦听器结合使用时非常强大，因为您不必使用 byte[]、字符串 或其他简单对象格式。您只能使用 HAPI HL7v2 模型对象。如果事先知道消息类型，则可以是更多的 type-safe ：

```
QRY_A19 msg = exchange.getIn().getBody(QRY_A19.class);
String patientId = msg.getQRD().getWhoSubjectFilter(0).getIDNumber().getValue();
```

76.4. HL7 DATAFORMAT

camel-hl7 JAR 附带了 HL7 数据格式，可用于 marshal 或 unmarshal HL7 模型对象。

HL7 数据格式支持 1 个选项，如下所列。

Name	默认值	Java 类型	描述
validate		布尔值	是否默认验证 HL7 消息是否为 true。

- marshal = from Message to byte stream （可以使用 HL7 MLLP codec）

- unmarshal = from byte stream to Message （您可以在从 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");
```

在上例中，HL7 从 HAPI Message 对象写入字节流，并放入 JMS 队列。
下一个示例是相反：

```
DataFormat hl7 = new HL7DataFormat();

from("jms:queue:hl7out")
  .unmarshal(hl7)
  .to("patientLookupService");
```

在这里，我们将字节流组合成一个 HAPI Message 对象，后者传递给我们的销售查找服务。

76.4.1. 分段分隔符

不要通过将 \n 转换为 \r 来自动修复片段分隔符。如果您
需要此转换，org.apache.camel.component.hl7.HL7114convertLFToCR 为这一目的提供了一个手动
表达式。

76.4.2. charset

marshal 和 unmarshal 评估字段 MSH-18 中提供的 charset。如果此字段为空，则假定假定对应的
Camel charset 属性/header 中包含的 charset。从 HL7DataFormat 类继承时，您还可以通过覆盖
observed CharsetName 方法来更改此默认行为。

Camel 中有一个简短的语法，用于常见使用的数据格式。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");
```

76.5. 消息标头

unmarshal 操作将 MSH 片段中的这些字段添加为 Camel 消息的标头：

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

键	MSH 字段	示例
CamelHL7Charset	MSH-18	UNICODE UTF-8

除 CamelHL7Context 外的所有标头都是 String 类型。如果缺少标头值，则其值为 null。

76.6. 依赖项

要在 Camel 路由中使用 HL7，您需要对上面列出的 camel-hl7 添加依赖项，该依赖项实施这些数据格式。

HAPI 库被分成 基本库和几个结构库，每个 HL7v2 消息版本对应一个：

- v2.1 结构库
- v2.2 结构库
- v2.3 结构库
- v2.3.1 结构库
- v2.4 结构库
- v2.5 结构库
- v2.5.1 结构库
- v2.6 结构库

默认情况下，camel-hl7 仅引用 HAPI 基础库。应用程序负责包括结构库本身。例如，如果应用程序可用于 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，也可以通过基本库下载所有结构库和所需的依赖项（捆绑包类路径）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>
```

76.7. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	是否默认验证 HL7 消息是否为 true。	true	布尔值

Name	描述	默认值	类型
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 77 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，这个 XML 数据格式的行为方式与其 JSON 对应部分相同，因此可用于注解用于 JSON 序列化/反序列化的类。

此扩展还模拟 JAXB 的"代码第一个"方法。

此数据格式依赖于 Woodstox （特别是用户打印等功能），这是一个快速高效的 XML 处理器。

```
from("activemq:My.Queue").
  unmarshal().jacksonxml().
  to("mqseries:Another.Queue");
```

77.1. JACKSONXML 选项

JacksonXML 数据格式支持 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	以格式方式启用用户友善打印输出。默认为 false。
unmarshalType		字符串	取消总结时要使用的 java 类型的类名称。
jsonView		字符串	将 POJO 写入 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。这个选项是引用具有 JsonView 注解的类。
Include		字符串	如果要将 pojo 重命名为 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，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。
collectionType		字符串	指的是要在 registry 中查找的自定义集合类型。应很少使用这个选项，但允许使用与 java.util.Collection 不同的集合类型作为默认值。

Name	默认值	Java 类型	描述
useList		布尔值	要取消选择映射列表或 Pojo 列表。
enableJaxbAnnotationModule		布尔值	使用 jackson 时是否启用 JAXB 注解模块。启用后，Jackson 可以使用 JAXB 注解。
moduleClassNames		字符串	使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为带有 FQN 类名称的 String。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。功能应该是与 com.fasterxml.jackson.databind.SerializationFeature、com.fasterxml.jackson.databind.DeserializationFeature 或 com.fasterxml.jackson.databind.MapperFeature 多功能中的 enum 匹配的名称。
disableFeatures		字符串	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的功能集合。功能应该是与 com.fasterxml.jackson.databind.SerializationFeature、com.fasterxml.jackson.databind.DeserializationFeature 或 com.fasterxml.jackson.databind.MapperFeature 多功能中的 enum 匹配的名称。
allowUnmarshallType		布尔值	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。这只在需要使用时才启用。
contentTypeHeader		布尔值	数据格式是否应使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 例如，marshalling 到 XML，对于数据格式为 JSON，application/json 用于数据格式。

77.1.1. 在 Spring DSL 中使用 Jackson XML

当在 Spring DSL 中使用 Data Format 时，您需要首先声明数据格式。这是在 DataFormats XML 标签中完成的。

```
    <dataFormats>
      <!-- here we define a Xml data format with the id jack and that it should use the
TestPojo as the class type when
           doing unmarshal. The 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>
```

77.1.2. 从 marshalling 中排除 POJO 字段

当将 POJO 写入 XML 时，您可能希望从 XML 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。首先创建一个或多个标记类。

使用带有 @JsonView 注释的标记类，使其包含/排除某些字段。该注解也适用于 getters。

最后，使用 Camel JacksonXMLDataFormat 将上述 POJO 写入 XML。

请注意，生成的 XML 中缺少 weight 字段：

```
<pojo age="30" weight="70"/>
```

77.2. 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);
```

在 Java DSL 中直接指定您的 JSON 视图，如下所示：

```
from("direct:inPojoAgeView").
  marshal().jacksonxml(TestPojoView.class, Views.Age.class);
```

在 XML DSL 中也是如此：

```
<from uri="direct:inPojoAgeView"/>
  <marshal>
    <jacksonxml unmarshalType="org.apache.camel.component.jacksonxml.TestPojoView"
jsonView="org.apache.camel.component.jacksonxml.Views$Age"/>
  </marshal>
```

77.3. 设置序列化包括选项

如果要将 pojo 重命名为 XML，并且 pojo 具有一些带有 null 值的字段。您希望跳过这些 null 值，然后在 pojo 上设置注解，

```
@JsonInclude(Include.NON_NULL)
public class MyPojo {
   ...
}
```

但这需要您在 pojo 源代码中包含该注解。您还可以配置 Camel JacksonXMLDataFormat 来设置 include 选项，如下所示：

```
JacksonXMLDataFormat format = new JacksonXMLDataFormat();
format.setInclude("NON_NULL");
```

或者，从 XML DSL 中将其配置为

```
    <dataFormats>
      <jacksonxml id="jacksonxml" include="NON_NULL"/>
    </dataFormats>
```

77.4. 使用动态类名称从 XML 取消归档到 POJO

如果您使用 jackson 取消编译 XML 来 POJO，您可以在消息中指定标头，以指示要取消设置哪个类名称。
如果消息中存在该标头，标头有键 CamelJacksonUnmarshalType，Jackson 将使用 POJO 类的 FQN 来取消编译 XML 有效负载。

```
 For JMS end users there is the JMSType header from the JMS spec that
indicates that also. To enable support for JMSType you would need to
turn that on, on the jackson data format as shown:
```

```
JacksonDataFormat format = new JacksonDataFormat();
format.setAllowJmsType(true);
```

或者，从 XML DSL 中将其配置为

```
<dataFormats>
  <jacksonxml id="jacksonxml" allowJmsType="true"/>
</dataFormats>
```

77.5. 从 XML 取消归档到 LIST<MAP> 或 LIST<POJO>

如果您使用 Jackson 将 XML 解压缩为映射/拓扑 jo 列表，您现在可以通过设置 useList="true" 或使用 org.apache.camel.component.jacksonxml.ListJacksonXMLDataFormat 来指定它。例如，对于 Java，您可以执行以下操作：

```
JacksonXMLDataFormat format = new ListJacksonXMLDataFormat();
// or
JacksonXMLDataFormat format = new JacksonXMLDataFormat();
format.useList();
// and you can specify the pojo class type also
format.setUnmarshalType(MyPojo.class);
```

如果您使用 XML DSL，您可以将 useList 属性配置为使用列表，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>
```

77.6. 使用自定义 JACKSON 模块

您可以使用 moduleClassNames 选项指定这些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，方法是使用默认构造器创建并用作原样。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"
```

77.7. 使用 JACKSON 启用或禁用功能

jackson 具有很多功能，您可以启用或禁用其 ObjectMapper 使用的功能。例如，要在 marshalling 时禁用未知属性失败，您可以使用 disableFeatures 进行配置：

```
<dataFormats>
    <jacksonxml id="jacksonxml" 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。功能的值必须是来自以下枚举类 Jackson 的枚举名称

- 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。

从 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);
```

77.8. 使用 JACKSON 将映射转换为 POJO

jackson ObjectMapper 可用于将映射转换为 POJO 对象。Jacks 组件附带 data converter，可用于将 java.util.Map 实例转换为非字符串、非专有和非数字对象。

```
Map<String, Object> invoiceData = new HashMap<String, Object>();
invoiceData.put("netValue", 500);
producerTemplate.sendBody("direct:mapToInvoice", invoiceData);
...
// Later in the processor
Invoice invoice = exchange.getIn().getBody(Invoice.class);
```

如果 Camel registry 中有一个对象映射程序实例，它将供转换器用来执行转换。否则将使用默认映射程序。

77.9. 格式化的 XML MARSHALLING (PRETTY-PRINTING)

使用用户 化打印选项 可以在 marshalling 时输出格式良好的 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 overlay jacksonxml （） DSL 方法，其支持用户 打印选项 与其他 unmarshalType、jsonView 等设置相结合。

77.10. 依赖项

要在 camel 路由中使用 Jackson XML，您需要对实现此数据格式的 camel-jacksonxml 添加依赖项。

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

最新版本的下载页面）。

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

77.11. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

组件支持 16 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.jacksonxml.allow-jms-type	用于 JMS 用户，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。	false	布尔值
camel.dataformat.jacksonxml.allow-unmarshall-type	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。这只在需要使用时才启用。	false	布尔值
camel.dataformat.jacksonxml.collection-type	指的是要在 registry 中查找的自定义集合类型。应很少使用这个选项，但允许使用与 java.util.Collection 不同的集合类型作为默认值。		字符串
camel.dataformat.jacksonxml.content-type-header	数据格式是否应使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 例如，marshalling 到 XML，对于数据格式为 JSON，application/json 用于数据格式。	true	布尔值

Name	描述	默认值	类型
camel.dataformat.jacksonxml.disable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的功能集合。功能应该是与 com.fasterxml.jackson.databind.SerializationFeature、com.fasterxml.jackson.databind.DeserializationFeature 或 com.fasterxml.jackson.databind.MapperFeature 多功能中的 enum 匹配的名称。		字符串
camel.dataformat.jacksonxml.enable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上启用的功能集合。功能应该是与 com.fasterxml.jackson.databind.SerializationFeature、com.fasterxml.jackson.databind.DeserializationFeature 或 com.fasterxml.jackson.databind.MapperFeature 多功能中的 enum 匹配的名称。		字符串
camel.dataformat.jacksonxml.enable-jaxb-annotation-module	使用 jackson 时是否启用 JAXB 注解模块。启用后，Jackson 可以使用 JAXB 注解。	false	布尔值
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	如果要将 pojo 重命名为 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	将 POJO 写入 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。这个选项是引用具有 JsonView 注解的类。		字符串
camel.dataformat.jacksonxml.module-class-names	使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为带有 FQN 类名称的 String。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	以格式方式启用用户友善打印输出。默认为 false。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.jacksonxml.unmarshal-type	取消总结时要使用的 java 类型的类名称。		字符串
camel.dataformat.jacksonxml.use-list	要取消选择映射列表或 Pojo 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 78 章 JAXB

JAXB 是一种数据格式，它使用 JAXB2 XML marshalling 标准，该标准包含在 Java 6 中，将 XML 有效负载 unmarshal Java 对象 unmarshal Java 对象到一个 XML 有效负载中。

78.1. 选项

JAXB 数据格式支持 19 个选项，如下所列。

Name	默认值	Java 类型	描述
contextPath		字符串	您的 JAXB 类所在的所需软件包名称。
contextPathIsClassName		布尔值	这可以设置为 true 来标记 contextPath 引用 classname 而不是软件包名称。
schema		字符串	针对现有模式进行验证：您可以使用前缀 classpath:, file: 或 http: 指定资源应该如何解析。您可以使用 ',' 字符分隔多个架构文件。
schemaSeverityLevel		Enum	设置在针对模式验证时要使用的模式严重性级别。此级别决定了触发 JAXB 停止继续解析的最低严重性错误。默认值为 0 (warning)表示任何错误（警告、错误或严重错误）将触发 JAXB 以停止。有三个级别： 0=warning, 1=error, 2=fatal 错误。 Enum 值： • 0 • 1 • 2
prettyPrint		布尔值	以格式方式启用用户友善打印输出。默认为 false。
ObjectFactory		布尔值	是否允许使用 ObjectFactory 类在 marshalling 期间创建 POJO 类。这只适用于尚未使用 JAXB 注解并提供 jaxb.index 描述符文件的 POJO 类。
ignoreHQBElement		布尔值	是否忽略 JAXBElement 元素 - 只需要在非常特殊用例中将 JAXBElement 元素设置为 false。
mustBeöBElement		布尔值	marhsalling 必须是带有 JAXB 注解的 java 对象。如果没有，则会失败。这个选项可以设置为 false 以放松，例如当数据已采用 XML 格式时。

Name	默认值	Java 类型	描述
filterNonXmlChars		布尔值	要忽略非 xml characheters，并使用空空间替换它们。
编码		字符串	要覆盖rule 并使用特定的编码。
片段		布尔值	打开 marshalling XML 片段树。默认情况下，B 在给定类上查找 XmlRootElement 注释，以便在整个 XML 树上运行。这很有用，但并不总是 - 有时生成的代码没有 XmlRootElement 注解，有时您只需要对树的一部分进行 unmarshall 部分。在这种情况下，您可以使用部分 unmarshalling。要启用此功能，您需要设置属性 partClass。Camel 会将此类传递到 JAXB 的 unmarshaler。
partClass		字符串	用于片段解析的类名称。请参阅 slice 选项的详情。
partNamespace		字符串	用于片段解析的 XML 命名空间。请参阅 slice 选项的详情。
namespacePrefixRef		字符串	当使用 JAXB 或 SOAP 进行 marshalling 时，JAXB 实施将自动分配命名空间前缀，如 ns2、ns3、ns4 等。要控制此映射，Camel 允许您引用包含所需映射的映射。
xmlStreamWriterWrapper		字符串	使用自定义 xml 流写入器。
schemaLocation		字符串	定义模式的位置。
noNamespaceSchemaLocation		字符串	定义无命名空间模式的位置。
jaxbProviderProperties		字符串	指的是一个自定义 java.util.Map，用于在包含 JAXB marshaller 的自定义 JAXB 提供程序属性的 registry 中查找。
contentTypeHeader		布尔值	数据格式是否应使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 例如，marshalling 到 XML，对于数据格式为 JSON，application/json 用于数据格式。

78.2. 使用 JAVA DSL

例如，以下命令使用名为 DataFormat 的 jaxb，它配置了多个 Java 软件包名称来初始化 JAXBContext。

```
DataFormat jaxb = new JaxbDataFormat("com.acme.model");

from("activemq:My.Queue").
```

```
  unmarshal(jaxb).
  to("mqseries:Another.Queue");
```

如果您希望对数据格式使用命名引用，然后在 Registry 中定义，如通过 Spring XML 文件。例如：

```
from("activemq:My.Queue").
  unmarshal("myJaxbDataType").
  to("mqseries:Another.Queue");
```

78.3. 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>
```

多个上下文路径

可以将这些数据格式与多个上下文路径一起使用。您可以使用：作为分隔符来指定上下文路径，例如 com.mycompany:com.mycompany2。请注意，这由 JAXB 实施处理，如果您使用不同于 RI 的供应商，可能会改变。

78.4. 部分 MARSHALLING/UNMARSHALLING

JAXB 2 支持 marshalling 和 unmarshalling XML 树片段。默认情况下，JAXB 会查找给定类上的

@XmlRootElement 注释，以便在整个 XML 树上运行。这非常有用，但并不总是 - 有时生成的代码没有 @XmlRootElement 注释，有时您只需要对树的一部分进行 unmarshall 部分。

在这种情况下，您可以使用部分 unmarshalling。要启用此功能，您需要设置属性 partClass。Camel 会将此类传递到 JAXB 的 unmarshaler。如果 JaxbConstants. theB_PART_CLASS 设置为标头之一，即使 DataFormat 上设置了 partClass 属性，DataFormat 上的属性将被传递，并使用标头中设置的。

对于 marshalling，您必须添加带有目标命名空间的 QName 的 partNamespace 属性。您可以找到的 Spring DSL 示例。

如果 JaxbConstants. theB_PART_NAMESPACE 设置为标头之一，即使 DataFormat 上设置了 partNamespace 属性，DataFormat 上的属性将被传递，并使用标头中设置的。在将 partNamespace 设置为 JaxbConstants. theB_PART_NAMESPACE 时，您需要指定其值 \{[namespaceUri]} [localPart]

```
...
.setHeader(JaxbConstants.JAXB_PART_NAMESPACE, simple("
{http://www.camel.apache.org/jaxb/example/address/1}address"));
...
```

78.5. 片段

JaxbDataFormat 具有新的属性片段，它可以设置 JAXB Marshaller 上的 Marshaller.osgiB_FRAGMENT 编码属性。如果您不希望 JAXB Marshaller 生成 XML 声明，您可以将此选项设置为 true。此属性的默认值为 false。

78.6. 忽略 NONXML CHARACTER

JaxbDataFormat 支持忽略 NonXML Character，您只需要将 filterNonXmlChars 属性设置为 true，JaxbDataFormat 会将 NonXML 字符替换为 '' when it is marshaling 或 unmarshaling 消息。您还可以通过设置 Exchange 属性 Exchange.FILTER_NON_XML_CHARS 来完成此操作。

	JDK 1.5	JDK 1.6+
使用过滤	stax API 和实现	否

	JDK 1.5	JDK 1.6+
不在使用中的过滤	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将流组合到 XML 的 XMLStreamWriter。使用这个配置，您可以添加自己的流写入器来完全删除、转义或替换非xml 字符。

```
JaxbDataFormat customWriterFormat = new JaxbDataFormat("org.apache.camel.foo.bar");
customWriterFormat.setXmlStreamWriterWrapper(new TestXmlStreamWriter());
```

以下示例演示了使用 Spring DSL 并启用 Camel 的 NonXML 过滤：

```
<bean id="testXmlStreamWriterWrapper" class="org.apache.camel.jaxb.TestXmlStreamWriter"/>
<jaxb filterNonXmlChars="true"  contextPath="org.apache.camel.foo.bar"
xmlStreamWriterWrapper="#testXmlStreamWriterWrapper" />
```

78.7. 使用 OBJECTFACTORY

如果您使用 XJC 从 schema 创建 java 类，您将获得 JAXB 上下文的 ObjectFactory。由于 ObjectFactory 使用 JAXBElement 来保存 schema 和元素实例值的引用，jaxbDataformat 默认会忽略 JAXBElement，并且您将获取元素实例值，而不是 JAXBElement 对象形成未合并的消息正文。

如果要获取 JAXBElement 对象形成 unmarshaled 消息正文，您需要将 JaxbDataFormat 对象的 ignore JAXBElement 属性设置为 false。

78.8. 设置编码

您可以将 编码 选项设置为在 marshalling 时使用。在 JAXB Marshaller 上，其 Marshaller. JAXB_ENCODING 编码属性。

您可以在声明 JAXB 数据格式时设置要使用的编码。您还可以在 Exchange 属性 Exchange.CHARSET_NAME 中提供编码。此属性将覆盖 JAXB 数据格式的编码集。

在这个 Spring DSL 中，我们定义了使用 iso-8859-1 作为编码。

78.9. 控制命名空间前缀映射

当使用 JAXB 或 SOAP 进行 marshalling 时，JAXB 实施将自动分配命名空间前缀，如 ns2、ns3、ns4 等。要控制此映射，Camel 允许您引用包含所需映射的映射。

请注意，这需要在 classpath 上具有 JAXB-RI 2.1 或更好（来自 SUN），因为映射功能取决于 JAXB 的实施，无论是支持。

例如，在 Spring XML 中，我们可使用映射来定义映射。在以下映射文件中，我们映射 SOAP 以作为前缀使用 soap。虽然我们的自定义命名空间 "http://www.mycompany.com/foo/2" 不使用任何前缀。

```
<util:map id="myMap">
  <entry key="http://www.w3.org/2003/05/soap-envelope" value="soap"/>
  <!-- we dont want any prefix for our namespace -->
  <entry key="http://www.mycompany.com/foo/2" value=""/>
</util:map>
```

要在 JAXB 或 SOAP 中使用此操作，您可以使用 namespacePrefixRef 属性来引用此映射，如下所示。然后，Camel 将在 Registry 中查找一个带有 id "myMap" 的 java.util.Map，这是前面定义的。

```
<marshal>
  <soapjaxb version="1.2" contextPath="com.mycompany.foo" namespacePrefixRef="myMap"/>
</marshal>
```

78.10. 模式验证

JAXB 数据格式支持通过 marshalling 和 unmarshalling from/to XML 来验证。您可以使用前缀 classpath:、file: 或 http: 指定资源应该如何解析。您可以使用 ',' 字符分隔多个架构文件。

使用 Java DSL，您可以使用以下方法进行配置：

```
JaxbDataFormat jaxbDataFormat = new JaxbDataFormat();
jaxbDataFormat.setContextPath(Person.class.getPackage().getName());
jaxbDataFormat.setSchema("classpath:person.xsd,classpath:address.xsd");
```

您可以使用 XML DSL 进行相同的操作：

```
<marshal>
    <jaxb id="jaxb" schema="classpath:person.xsd,classpath:address.xsd"/>
</marshal>
```

Camel 将创建并池实时上闲置 SchemaFactory 实例的下来，因为 JDK 附带的 SchemaFactory 不安全线程。
但是，如果您有一个安全线程的 SchemaFactory 实现，您可以将 JAXB 数据格式配置为使用此数据：

```
JaxbDataFormat jaxbDataFormat = new JaxbDataFormat();
jaxbDataFormat.setSchemaFactory(thradSafeSchemaFactory);
```

78.11. 模式位置

JAXB 数据格式支持在编译 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 进行相同的操作：

```
<marshal>
    <jaxb id="jaxb" schemaLocation="schema/person.xsd"/>
</marshal>
```

78.12. 已 XML 的 MARSHAL 数据

JAXB marshaller 要求消息正文兼容 JAXB，与 JAXBElement 类似，例如具有 JAXB 注解或扩展 JAXBElement 的 java 实例。在某些情况下，消息正文已位于 XML 中，而不是 String 类型。

您可以将其设置为 false 的新选项 mustBeturnBElement 来放松此检查，因此 JAXB marshaller 仅会尝试 marshal JAXBElements (javax.xml.bind. laspecIntrospector ApiisElement 返回 true)。在这些情况下，marshaller 回退将消息正文按原样处理。

78.13. 依赖项

要在 camel 路由中使用 JAXB，您需要对实现此数据格式的 camel-jaxb 添加依赖项。

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

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jaxb</artifactId>
  <version>{CamelSBVersion}</version>
</dependency>
```

78.14. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

组件支持 20 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.jaxb.content-type-header	数据格式是否应使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 例如，marshalling 到 XML，对于数据格式为 JSON，application/json 用于数据格式。	true	布尔值
camel.dataformat.jaxb.context-path	JAXB 类所在的软件包名称。		字符串
camel.dataformat.jaxb.context-path-is-class-name	这可以设置为 true 来标记 contextPath 引用 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	要覆盖rule 并使用特定的编码。		字符串

Name	描述	默认值	类型
camel.dataformat.jaxb.filter-non-xml-chars	要忽略非 xml characheters，并使用空空间替换它们。	false	布尔值
camel.dataformat.jaxb.fragment	打开 marshalling XML 片段树。默认情况下，B 在给定类上查找 XmlRootElement 注释，以便在整个 XML 树上运行。这很有用，但并不总是 - 有时生成的代码没有 XmlRootElement 注解，有时您只需要对树的一部分进行 unmarshall 部分。在这种情况下，您可以使用部分 unmarshalling。要启用此功能，您需要设置属性 partClass。Camel 会将此类传递到 JAXB 的 unmarshaler。	false	布尔值
camel.dataformat.jaxb.ignore-j-a-x-b-element	是否忽略 JAXBElement 元素 - 只需要在非常特殊用例中将 JAXBElement 元素设置为 false。	false	布尔值
camel.dataformat.jaxb.jaxb-provider-properties	指的是一个自定义 java.util.Map，用于在包含 JAXB marshaller 的自定义 JAXB 提供程序属性的 registry 中查找。		字符串
camel.dataformat.jaxb.must-be-j-a-x-b-element	marhsalling 必须是带有 JAXB 注解的 java 对象。如果没有，则会失败。这个选项可以设置为 false 以放松，例如当数据已采用 XML 格式时。	false	布尔值
camel.dataformat.jaxb.namespace-prefix-ref	当使用 JAXB 或 SOAP 进行 marshalling 时，JAXB 实施将自动分配命名空间前缀，如 ns2、ns3、ns4 等。要控制此映射，Camel 允许您引用包含所需映射的映射。		字符串
camel.dataformat.jaxb.no-namespace-schema-location	定义无命名空间模式的位置。		字符串
camel.dataformat.jaxb.object-factory	是否允许使用 ObjectFactory 类在 marshalling 期间创建 POJO 类。这只适用于尚未使用 JAXB 注解并提供 jaxb.index 描述符文件的 POJO 类。	false	布尔值
camel.dataformat.jaxb.part-class	用于片段解析的类名称。请参阅 slice 选项的详情。		字符串
camel.dataformat.jaxb.part-namespace	用于片段解析的 XML 命名空间。请参阅 slice 选项的详情。		字符串

Name	描述	默认值	类型
camel.dataformat.jaxb.pretty-print	以格式方式启用用户友善打印输出。默认为 false。	false	布尔值
camel.dataformat.jaxb.schema	针对现有模式进行验证：您可以使用前缀 classpath:, file: 或 http: 指定资源应该如何解析。您可以使用 ',' 字符分隔多个架构文件。		字符串
camel.dataformat.jaxb.schema-location	定义模式的位置。		字符串
camel.dataformat.jaxb.schema-severity-level	设置在针对模式验证时要使用的模式严重性级别。此级别决定了触发 JAXB 停止继续解析的最低严重性错误。默认值为 0 (warning)表示任何错误（警告、错误或严重错误）将触发 JAXB 以停止。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 流写入器。		字符串

第 79 章 JSON GSON

gson 是一个数据格式，它使用 Gson 库。

```
from("activemq:My.Queue").
  marshal().json(JsonLibrary.Gson).
  to("mqseries:Another.Queue");
```

79.1. GSON 选项

JSON Gson 数据格式支持 3 个选项，如下所列。

Name	默认值	Java 类型	描述
prettyPrint		布尔值	以格式方式启用用户友善打印输出。默认为 false。
unmarshalType		字符串	取消总结时要使用的 java 类型的类名称。
contentTypeHeader		布尔值	数据格式是否应使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 例如，marshalling 到 XML，对于数据格式为 JSON，application/json 用于数据格式。

79.2. 依赖项

要在 camel 路由中使用 Gson，您需要添加实施此数据格式的 camel-gson 的依赖关系。

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

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

79.3. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	数据格式是否应使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 例如，marshalling 到 XML，对于数据格式为 JSON，application/json 用于数据格式。	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	以格式方式启用用户友善打印输出。默认为 false。	false	布尔值
camel.dataformat.json-gson.unmarshal-type	取消总结时要使用的 java 类型的类名称。		字符串

第 80 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");
```

80.1. JACKS 选项

JSON Jackson 数据格式支持 20 个选项，如下所列。

Name	默认值	Java 类型	描述
objectMapper		字符串	在使用 Jackson 时，查找并使用给定 ID 的现有 ObjectMapper。
useDefaultObjectMapper		布尔值	是否查找并使用 registry 中的默认 Jackson ObjectMapper。
prettyPrint		布尔值	以格式方式启用用户友善打印输出。默认为 false。
unmarshalType		字符串	取消总结时要使用的 java 类型的类名称。
jsonView		字符串	将 POJO 写入 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。这个选项是引用具有 JsonView 注解的类。

Name	默认值	Java 类型	描述
Include		字符串	如果要将 pojo 重命名为 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，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。
collectionType		字符串	指的是要在 registry 中查找的自定义集合类型。应很少使用这个选项，但允许使用与 java.util.Collection 不同的集合类型作为默认值。
useList		布尔值	要取消选择映射列表或 Pojo 列表。
moduleClassNames		字符串	使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为带有 FQN 类名称的 String。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。

Name	默认值	Java 类型	描述
enableFeatures		**字符串**	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上启用的功能集合。功能应该是与 com.fasterxml.jackson.databind.SerializationFeature、com.fasterxml.jackson.databind.DeserializationFeature 或 com.fasterxml.jackson.databind.MapperFeature 多功能中的 enum 匹配的名称。
disableFeatures		**字符串**	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的功能集合。功能应该是与 com.fasterxml.jackson.databind.SerializationFeature、com.fasterxml.jackson.databind.DeserializationFeature 或 com.fasterxml.jackson.databind.MapperFeature 多功能中的 enum 匹配的名称。
allowUnmarshallType		**布尔值**	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。这只在需要使用时才启用。

Name	默认值	Java 类型	描述
timezone		字符串	如果设置，Jackson 会在 marshalling/unmarshalling 时使用 Timezone。这个选项对 Json DataFormat 等其他 Json DataFormat 的影响，如 gson、fastjson 和 xstream。
autoDiscoverObjectMapper		布尔值	如果设置为 true，Jackson 会将对象映射器查找到 registry 中。
contentTypeHeader		布尔值	数据格式是否应使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 例如，marshalling 到 XML，对于数据格式为 JSON，application/json 用于数据格式。
schemaResolver		字符串	可选的 schema 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 registry。

Name	默认值	Java 类型	描述
namingStrategy		字符串	如果设置，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。

80.2. 使用自定义对象映射程序

当需要更多控制映射配置时，您可以将 JacksonDataFormat 配置为使用自定义 ObjectMapper。

如果您在 registry 中设置 单个对象映射程序，则 Camel 将自动查找并使用此 对象映射程序。例如，如果您使用 Spring Boot，如果您启用了 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 中存在一个 bean 的 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。

80.3. 使用 JACKSON 自动进行类型转换

camel-jackson 模块允许将 Jackson 集成为 Type Converter。这的工作方式类似于与 Camel 类型转换器集成的 JAXB。

要使用这个 camel-jackson，这可以通过在 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，这意味着您可以使用 Jackson 的 JAXB 注解为 POJO 类添加注解。您还可以在 POJO 类中使用 Jackson 自己的注解。

80.4. 依赖项

要在 camel 路由中使用 Jackson，您需要对实现此数据格式的 camel-jackson 添加依赖项。

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

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

80.5. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。	false	布尔值
camel.dataformat.json-jackson.allow-unmarshall-type	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。这只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，Jackson 会将对象映射器查找到 registry 中。	false	布尔值
camel.dataformat.json-jackson.auto-discover-schema-resolver	如果没有禁用，SchemaResolver 将查找 registry。	true	布尔值
camel.dataformat.json-jackson.collection-type	指的是要在 registry 中查找的自定义集合类型。应很少使用这个选项，但允许使用与 java.util.Collection 不同的集合类型作为默认值。		字符串
camel.dataformat.json-jackson.content-type-header	数据格式是否应使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 例如，marshalling 到 XML，对于数据格式为 JSON，application/json 用于数据格式。	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。功能应该是与 com.fasterxml.jackson.databind.SerializationFeature、com.fasterxml.jackson.databind.DeserializationFeature 或 com.fasterxml.jackson.databind.MapperFeature 多功能中的 enum 匹配的名称。		字符串

Name	描述	默认值	类型
camel.dataformat.json-jackson.enable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上启用的功能集合。功能应该是与 com.fasterxml.jackson.databind.SerializationFeature、com.fasterxml.jackson.databind.DeserializationFeature 或 com.fasterxml.jackson.databind.MapperFeature 多功能中的 enum 匹配的名称。		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	如果要将 pojo 重命名为 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.json-jackson.json-view	将 POJO 写入 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。这个选项是引用具有 JsonView 注解的类。		字符串
camel.dataformat.json-jackson.module-class-names	使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为带有 FQN 类名称的 String。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，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	在使用 Jackson 时，查找并使用给定 ID 的现有 ObjectMapper。		字符串
camel.dataformat.json-jackson.pretty-print	以格式方式启用用户友善打印输出。默认为 false。	false	布尔值

Name	描述	默认值	类型
camel.dataformat.json-jackson.schema-resolver	可选的 schema 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，Jackson 会在 marshalling/unmarshalling 时使用 Timezone。这个选项对 Json DataFormat 等其他 Json DataFormat 的影响，如 gson、fastjson 和 xstream。		字符串
camel.dataformat.json-jackson.unmarshal-type	取消总结时要使用的 java 类型的类名称。		字符串
camel.dataformat.json-jackson.use-default-object-mapper	是否查找并使用 registry 中的默认 Jackson ObjectMapper。	true	布尔值
camel.dataformat.json-jackson.use-list	要取消选择映射列表或 Pojo 列表。	false	布尔值

第 81 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 有效负载到 Protobuf 有效负载。

注意

如果您熟悉 Jackson，则此 Protobuf 数据格式的行为与其 JSON 对应部分相同，因此可用于注解用于 JSON 序列化/反序列化的类。

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

81.1. 配置 SCHEMARESOLVER

由于 Protobuf 序列化是基于架构的，因此这些数据格式要求您提供一个 SchemaResolver 对象，该对象可以查找每个交换的模式，该交换将被完全使用/全部关闭。

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

81.2. PROTOBUF JACKSON 选项

Protobuf Jackson 数据格式支持 18 个选项，如下所列。

Name	默认值	Java 类型	描述
contentTypeHeader		布尔值	数据格式是否应使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 例如，marshalling 到 XML，对于数据格式为 JSON，application/json 用于数据格式。
objectMapper		字符串	在使用 Jackson 时，查找并使用给定 ID 的现有 ObjectMapper。
useDefaultObjectMapper		布尔值	是否查找并使用 registry 中的默认 Jackson ObjectMapper。

Name	默认值	Java 类型	描述
unmarshalType		字符串	取消总结时要使用的 java 类型的类名称。
jsonView		字符串	将 POJO 写入 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。这个选项是引用具有 JsonView 注解的类。
Include		字符串	如果要将 pojo 重命名为 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，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。
collectionType		字符串	指的是要在 registry 中查找的自定义集合类型。应很少使用这个选项，但允许使用与 java.util.Collection 不同的集合类型作为默认值。
useList		布尔值	要取消选择映射列表或 Pojo 列表。
moduleClassNames		字符串	使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为带有 FQN 类名称的 String。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。功能应该是与 com.fasterxml.jackson.databind.SerializationFeature、com.fasterxml.jackson.databind.DeserializationFeature 或 com.fasterxml.jackson.databind.MapperFeature 多功能中的 enum 匹配的名称。
disableFeatures		字符串	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上禁用的功能集合。功能应该是与 com.fasterxml.jackson.databind.SerializationFeature、com.fasterxml.jackson.databind.DeserializationFeature 或 com.fasterxml.jackson.databind.MapperFeature 多功能中的 enum 匹配的名称。
allowUnmarshallType		布尔值	如果启用，则允许 Jackson 在 unmarshalling 期间尝试使用 CamelJacksonUnmarshalType 标头。这只在需要使用时才启用。

Name	默认值	Java 类型	描述
timezone		字符串	如果设置，Jackson 会在 marshalling/unmarshalling 时使用 Timezone。
autoDiscoverObjectMapper		布尔值	如果设置为 true，Jackson 会将对象映射器查找到 registry 中。
schemaResolver		字符串	可选的 schema 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 registry。

81.3. 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。

81.4. 依赖项

要在 camel 路由中使用 Protobuf Jackson，您需要对实现此数据格式的 camel-jackson-protobuf 添加依赖项。

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

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

81.5. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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，允许 JMS spec 中的 JMSType 标头指定用于 unmarshal 的 FQN 类名称。	false	布尔值
camel.dataformat.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，Jackson 会将对象映射器查找到 registry 中。	false	布尔值
camel.dataformat.protobuf-jackson.auto-discover-schema-resolver	如果没有禁用，SchemaResolver 将查找 registry。	true	布尔值
camel.dataformat.protobuf-jackson.collection-type	指的是要在 registry 中查找的自定义集合类型。应很少使用这个选项，但允许使用与 java.util.Collection 不同的集合类型作为默认值。		字符串
camel.dataformat.protobuf-jackson.content-type-header	数据格式是否应使用数据格式的类型设置 Content-Type 标头。例如，用于数据格式的 application/xml 例如，marshalling 到 XML，对于数据格式为 JSON，application/json 用于数据格式。	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。功能应该是与 com.fasterxml.jackson.databind.SerializationFeature、com.fasterxml.jackson.databind.DeserializationFeature 或 com.fasterxml.jackson.databind.MapperFeature 多功能中的 enum 匹配的名称。		字符串

Name	描述	默认值	类型
camel.dataformat.protobuf-jackson.enable-features	在 Jackson com.fasterxml.jackson.databind.ObjectMapper 上启用的功能集合。功能应该是与 com.fasterxml.jackson.databind.SerializationFeature、com.fasterxml.jackson.databind.DeserializationFeature 或 com.fasterxml.jackson.databind.MapperFeature 多功能中的 enum 匹配的名称。		字符串
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	如果要将 pojo 重命名为 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	将 POJO 写入 JSON 时，您可能希望从 JSON 输出中排除某些字段。通过 Jackson，您可以使用 JSON 视图来实现这一目的。这个选项是引用具有 JsonView 注解的类。		字符串
camel.dataformat.protobuf-jackson.module-class-names	使用自定义 Jackson 模块 com.fasterxml.jackson.databind.Module 指定为带有 FQN 类名称的 String。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	在使用 Jackson 时，查找并使用给定 ID 的现有 ObjectMapper。		字符串
camel.dataformat.protobuf-jackson.schema-resolver	可选的 schema 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，Jackson 会在 marshalling/unmarshalling 时使用 Timezone。		字符串

Name	描述	默认值	类型
camel.dataformat.protobuf-jackson.unmarshal-type	取消总结时要使用的 java 类型的类名称。		字符串
camel.dataformat.protobuf-jackson.use-default-object-mapper	是否查找并使用 registry 中的默认 Jackson ObjectMapper。	true	布尔值
camel.dataformat.protobuf-jackson.use-list	要取消选择映射列表或 Pojo 列表。	false	布尔值

第 82 章 SOAP

SOAP 是一种数据格式，它使用 JAXB2 和 JAX-WS 注释来 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 marshalling 时如何控制命名空间前缀映射的详细信息，请参阅 JAXB。

82.1. SOAP 选项

SOAP 数据格式支持 6 个选项，如下所列。

Name	默认值	Java 类型	描述
contextPath		字符串	您的 JAXB 类所在的所需软件包名称。
编码		字符串	要覆盖rule 并使用特定的编码。
elementNameStrategyRef		字符串	引用用于从 registry 查找的元素策略。元素名称策略用于两个目的。第一个是查找给定对象的 xml 元素名称，在将对象组合到 SOAP 消息时执行 soap 操作。第二个是查找给定 soap 错误名称的例外类。以下三元素策略类名称开箱即用。QNameStrategy - 使用在实例化上配置的固定 qName。不支持异常查找 - 使用给定类型的 XMLType 注解中的名称和命名空间。如果没有设置命名空间，则使用 package-info。不支持别名的 ServiceInterfaceStrategy - 使用 webservice 界面中的信息来确定类型名称，并找到 SOAP 错误的所有类异常类位于软件包名称 org.apache.camel.dataformat.soap.name 中，如果您生成了带有 cxf-codegen 或类似的工具的 web 服务根代码，那么您可能希望使用 ServiceInterfaceStrategy。如果没有注解的服务接口，则应使用 QNameStrategy 或 TypeNameStrategy。
version		字符串	SOAP 版本应该是 1.1 或 1.2。默认为 1.1。
namespacePrefixRef		字符串	当使用 JAXB 或 SOAP 进行 marshalling 时，JAXB 实施将自动分配命名空间前缀，如 ns2、ns3、ns4 等。要控制此映射，Camel 允许您引用包含所需映射的映射。
schema		字符串	针对现有模式进行验证：您可以使用前缀 classpath:, file: 或 http: 指定资源应该如何解析。您可以使用 ',' 字符分隔多个架构文件。

82.2. ELEMENTNAMESTRATEGY

元素名称策略用于两个目的。第一个是查找给定对象的 xml 元素名称，在将对象组合到 SOAP 消息时执行 soap 操作。第二个是查找给定 soap 错误名称的例外类。

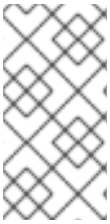
策略	使用方法
QNameStrategy	使用实例化上配置的固定 qName。不支持异常查找
TypeNameStrategy	使用给定类型的 @XMLType 注释中的 name 和 namespace。如果没有设置命名空间，则使用 package-info。不支持异常查找
ServiceInterfaceStrategy	使用 Web 服务接口的信息来确定类型名称，并查找 SOAP 错误的异常类

如果您生成了带有 cxf-codegen 或类似的工具的 web 服务存根代码，您可能想使用 ServiceInterfaceStrategy。如果没有注解的服务接口，则应使用 QNameStrategy 或 TypeNameStrategy。

82.3. 使用 JAVA DSL

以下示例使用名为 soap 的名为 DataFormat，它配置了软件包 com.example.customerservice 来初始化 JAXBContext。第二个参数是 ElementNameStrategy。路由可以 marshal 普通对象和例外。（请注意以下只是向队列发送 SOAP Envelope。Web 服务提供商实际上需要侦听队列以了解实际发生的 SOAP 调用，在这种情况下，它将是 SOAP 请求的一种方法是。如果您需要请求回复，您应该查看下一示例。

```
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
，因为来自 JAXB 数据格式的 SOAP 数据格式继承在此处都适用。

82.3.1. Using 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>
```

82.4. 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初始化，该定义会根据 JAX-WS 2.2 注解并满足 Document Bare 风格的要求。目标方法需要满足以下条件，遵循 JAX-WS 规格：1) 它最多有一个 in 或 in/out 非标头的参数，2) 如果它有一个非 void 的返回类型，它必须没有 in/out 或 out 非标题的参数，3) 如果它有一个返回类型 void，它需要最多有一个 in/out 或 out 非标头的参数。

ServiceInterfaceStrategy 应该使用布尔值参数初始化，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.class, true);
SoapJaxbDataFormat soapDataFormat = new
SoapJaxbDataFormat("com.example.customerservice.multipart", strat);
```

82.4.1. 拥有者对象映射

JAX-WS 指定对 In/Out 和 Out 参数使用 type-parameterized javax.xml.ws.Holder 对象。您可以直接使用参数类型的实例。camel-soap DataFormat marshals Holder 值根据 Holder 值类的 JAXB 映射而定。在未记录的响应中为 \'Holder 对象提供映射。

82.5. 例子

82.5.1. Web 服务客户端

以下路由支持 marshalling 请求并取消处理响应或错误。

```
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一个代理，并对上述路由发出 SOAP 调用。

```
import org.apache.camel.Endpoint;
import org.apache.camel.component.bean.ProxyHelper;
...

Endpoint startEndpoint = context.getEndpoint("direct:customerServiceClient");
ClassLoader classLoader = Thread.currentThread().getContextClassLoader();
// CustomerService below is the service endpoint interface, *not* the javax.xml.ws.Service
subclass
CustomerService proxy = ProxyHelper.createProxy(startEndpoint, classLoader,
CustomerService.class);
GetCustomersByNameResponse response = proxy.getCustomersByName(new
GetCustomersByName());
```

82.5.2. Web 服务服务器

使用以下路由设置 web 服务服务器，它侦听 jms 队列 customerServiceQueue，并使用类 CustomerServiceImpl 处理请求。课程的 customerServiceImpl 应该实施接口 CustomerService。而不必直接实例化服务器类，而是作为常规 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);
```

82.6. 依赖项

要在 camel 路由中使用 SOAP 数据格式，您需要将以下依赖项添加到 pom 中。

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-soap</artifactId>
  <version>{CamelSBVersion}</version>
</dependency>
```

82.7. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用 soapjaxb 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>
```

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

Name	描述	默认值	类型
camel.dataformat.soapjaxb.element-name-strategy-ref	引用用于从 registry 查找的元素策略。元素名称策略用于两个目的。第一个是查找给定对象的 xml 元素名称，在将对象组合到 SOAP 消息时执行 soap 操作。第二个是查找给定 soap 错误名称的例外类。以下三元素策略类名称开箱即用。QNameStrategy - 使用在实例化上配置的固定 qName。不支持异常查找 - 使用给定类型的 XMLType 注解中的名称和命名空间。如果没有设置命名空间，则使用 package-info。不支持别名的 ServiceInterfaceStrategy - 使用 webservice 界面中的信息来确定类型名称，并找到 SOAP 错误的所有类异常类位于软件包名称 org.apache.camel.dataformat.soap.name 中，如果您生成了带有 cxf-codegen 或类似的工具的 web 服务根代码，那么您可能希望使用 ServiceInterfaceStrategy。如果没有注解的服务接口，则应使用 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	要覆盖rule 并使用特定的编码。		字符串
camel.dataformat.soapjaxb.namespace-prefix-ref	当使用 JAXB 或 SOAP 进行 marshalling 时，JAXB 实施将自动分配命名空间前缀，如 ns2、ns3、ns4 等。要控制此映射，Camel 允许您引用包含所需映射的映射。		字符串
camel.dataformat.soapjaxb.schema	针对现有模式进行验证：您可以使用前缀 classpath:, file: 或 http: 指定资源应该如何解析。您可以使用 ',' 字符分隔多个架构文件。		字符串
camel.dataformat.soapjaxb.version	SOAP 版本应该是 1.1 或 1.2。默认为 1.1。	1.1	字符串

第 83 章 ZIP 文件

Zip File Data Format 是一个消息压缩和解压缩格式。消息可以组合到包含单个条目的 Zip 文件，并将包含单个条目的 Zip 文件取消合并（解压缩）到原始文件内容。只要使用了 Java 7 或更高版本，此数据格式支持 ZIP64。

83.1. ZIPFILE 选项

Zip 文件数据格式支持 4 个选项，如下所列。

Name	默认值	Java 类型	描述
usingIterator		布尔值	如果 zip 文件有多个条目，则此选项设置为 true，允许使用分割器 EIP 在流模式下使用迭代器分割数据。
allowEmptyDirectory		布尔值	如果 zip 文件有多个条目，则此选项设置为 true，即使目录为空，也可以获取迭代器。
preservePathElements		布尔值	如果文件名包含路径元素，请将此选项设置为 true，允许在 zip 文件中维护路径。
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，则默认为 1GB。如果解压缩的大小超过这个数量，则抛出 IOException。设置为 -1 以禁用设置最大解压缩的大小。

83.2. MARSHAL

在本例中，我们使用 Zip 文件压缩将常规文本/XML 有效负载归档到压缩的有效负载，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。此外，传出的 CamelFileName 消息标头会自动设置为传入 CamelFileName 消息标头的值，使用 ".zip" 后缀。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，输出将是名为 "test.txt.zip" 的 Zip 文件，其中包含一个名为 "test.txt" 的 Zip 条目：

```
from("file:input/directory?antInclude=*/.txt")
    .marshal().zipFile()
    .to("file:output/directory");
```

如果没有传入的 CamelFileName 消息标头（例如，如果文件组件不是消费者），则默认使用消息 ID，因为消息 ID 通常是唯一的 ID，如 ID-MACHINENAME-2443-1211712437-1-0.zip。如果要覆盖此行为，您可以在路由中显式设置 CamelFileName 标头的值：

```
from("direct:start")
    .setHeader(Exchange.FILE_NAME, constant("report.txt"))
    .marshal().zipFile()
    .to("file:output/directory");
```

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 Zip 条目。

83.3. UNMARSHAL

在本例中，我们将名为 MY_QUEUE 的 ActiveMQ 队列中的 Zip 文件有效负载解压缩到其原始格式，并将它转发到 UnZippedMessageProcessor。

```
from("activemq:queue:MY_QUEUE")
    .unmarshal().zipFile()
    .process(new UnZippedMessageProcessor());
```

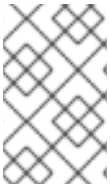
如果 zip 文件有多个条目，则 ZipFileDataFormat 的 usingIterator 选项为 true，您可以使用 splitter 来进一步工作。

```
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 作为直接分割器的表达式，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();
```

83.3.1. 聚合



注意

此聚合策略需要 eager 完成检查才能正常工作。

在这个示例中，我们将输入目录中找到的所有文本文件聚合到存储在输出目录中的单个 Zip 文件中。

```
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 创建，并带有 ".zip" 后缀。如果要覆盖此行为，您可以在路由中显式设置 CamelFileName 标头的值：

```
from("file:input/directory?antInclude=*/.txt")
   .aggregate(constant(true), new ZipAggregationStrategy())
      .completionFromBatchConsumer().eagerCheckCompletion()
      .setHeader(Exchange.FILE_NAME, constant("reports.zip"))
      .to("file:output/directory");
```

83.4. 依赖项

要在 camel 路由中使用 Zip 文件，您需要添加一个依赖项来实现这些数据格式的 camel-zipfile。

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

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

83.5. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

组件支持 5 个选项，如下所列。

Name	描述	默认值	类型
camel.dataformat.zipfile.allow-empty-directory	如果 zip 文件有多个条目，则此选项设置为 true，即使目录为空，也可以获取迭代器。	false	布尔值
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，则默认为 1GB。如果解压缩的大小超过这个数量，则抛出 IOException。设置为 -1 以禁用设置最大解压缩的大小。	1073741824	Long
camel.dataformat.zipfile.preserve-path-elements	如果文件名包含路径元素，请将此选项设置为 true，允许在 zip 文件中维护路径。	false	布尔值
camel.dataformat.zipfile.using-iterator	如果 zip 文件有多个条目，则此选项设置为 true，允许使用分割器 EIP 在流模式下使用迭代器分割数据。	false	布尔值

第 84 章 常数

Constant Expression Language 实际上是使用常量值或对象的方法。

注意

这是一个固定常量值（或对象），它只在启动路由时设置一次，如果您在路由过程中需要动态值，则不要使用这个值。

84.1. 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	设置常量类型的类名称。
trim		布尔值	是否修剪值以移除前导和结尾的空格和换行符。

84.2. 示例

setHeader EIP 可以使用如下常量表达式：

```
<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 端点的消息将具有带有键的标头， 其值为 value （字符串类型）。

以及使用 Java DSL 的同一示例：

```
from("seda:a")
  .setHeader("theHeader", constant("the value"))
  .to("mock:b");
```

84.2.1. 指定值类型

选项 resultType 可以用来指定值的类型，当将该值指定为 String 值时，该值在使用 XML 或 YAML DSL 时发生：

例如，要设置带有 int 类型的标头，您可以执行以下操作：

```
<route>
  <from uri="seda:a"/>
  <setHeader name="zipCode">
    <constant resultType="int">90210</constant>
  </setHeader>
  <to uri="mock:b"/>
</route>
```

84.3. 从外部资源加载常数

您可以外部化常量，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中的文件：

```
.setHeader("myHeader").constant("resource:classpath:constant.txt")
```

84.4. 依赖项

Constant 语言是 camel-core 的一部分。

84.5. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	设置与 Consul 搭配使用的 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 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服务调用实施。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	为 OkHttpClient 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服务调用实施。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	所需的服务的传输协议。	_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服务调用实施。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	要设置操作完成的最长时间。		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	在使用客户端查找时设置 API 版本。		字符串
camel.cloud.kubernetes.service-discovery.ca-cert-data	在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	设置在使用客户端查找时从文件载入的证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-data	在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	设置在使用客户端查找时从文件载入的客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-algo	设置客户端密钥存储算法，如在使用客户端查找时的 RSA。		字符串
camel.cloud.kubernetes.service-discovery.client-key-data	在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	设置在使用客户端查找时从文件载入的客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-passphrase	在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 查询的 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、环境。在使用客户端时，客户端会查询 kubernetes master 以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，服务名称通过 SRV 查询解析 ....svc... when 使用环境变量来查找该服务。默认使用默认环境。	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，设置 OAUTH 令牌以进行身份验证（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服务调用实施。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

Name	描述	默认值	类型
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服务调用实施。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 maximumSize 生效。然后该值可以等于 coreSize 或更高。	false	布尔值
camel.hystrix.circuit-breaker-enabled	是否使用 HystrixCircuitBreaker。如果为 false，则使用断路器逻辑以及允许的所有请求。这与 hardBreakerForceClosed （）的影响类似，但继续跟踪指标并了解它是否是打开/披露，此属性甚至不实例化断路器。	true	布尔值
camel.hystrix.circuit-breaker-error-threshold-percentage	一个错误百分比阈值（如 50），其中断路器将打开和拒绝请求。它将持续等待在断路器 BreakerSleepWindowInMilliseconds 中定义的持续时间；这与 HystrixCommandMetrics.getHealthCounts （）进行比较的错误百分比。	50	整数

Name	描述	默认值	类型
camel.hystrix.circuit-breaker-force-closed	如果为 true，HystrixCircuitBreaker114allowRequest （）始终返回 true 以允许请求，而不考虑 HystrixCommandMetrics.getHealthCounts （）的错误百分比。paraBreakerForceOpen （）属性具有优先权，因此如果设为 true，则它将不做任何操作。	false	布尔值
camel.hystrix.circuit-breaker-force-open	如果为 true，HystrixCircuitBreaker.allowRequest （）将始终返回 false，从而导致断路器打开（往返）并拒绝所有请求。This property takes precedence over circuitBreakerForceClosed();.	false	布尔值
camel.hystrix.circuit-breaker-request-volume-threshold	在 HystrixCircuitBreaker 之前必须存在 metricsRollingStatisticalWindowInMilliseconds （）中的最小请求数。如果低于这个数字，则无论错误百分比如何，时钟都不会往返。	20	整数
camel.hystrix.circuit-breaker-sleep-window-in-milliseconds	HystrixCircuitBreaker 往返后的时间（毫秒）打开它应该等待，然后再再次尝试请求。	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.ThreadPoolExecutor114setCorePoolSize (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 executionIsolationStrategy == SEMAPHORE 时才适用。	20	整数
camel.hystrix.execution-isolation-strategy	将执行哪些隔离策略 HystrixCommand.run （）。如果 THREAD，那么它将在单独的线程上执行，并发请求则受 thread-pool 中的线程数量限制。如果 SEMAPHORE，它将在调用线程和并发请求上执行，则由 semaphore 数限制。	THREAD	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（使用将来的站已取消。仅在 executionIsolationStrategy （）== THREAD 时才适用。	true	布尔值

Name	描述	默认值	类型
camel.hystrix.execution-timeout-enabled	是否为这个命令启用超时机制。	true	布尔值
camel.hystrix.execution-timeout-in-milliseconds	在一段时间内，命令将超时和停止执行的时间（毫秒）。如果 executionIsolationThreadInterruptOnTimeout == true，命令是线程隔离，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	失败时是否应尝试 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失败，且不会尝试检索回退。	10	整数
camel.hystrix.group-key	设置要使用的组密钥。默认值为 CamelHystrix。	Camel Hystrix	字符串
camel.hystrix.keep-alive-time	keep-alive 时间（以分钟为单位）传递给 ThreadPoolExecutor，或 theepAliveTime (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 - 不会影响实时更改队列大小。为此，请使用 queueSizeRejectionThreshold （）。	-1	整数
camel.hystrix.maximum-size	传递给 ThreadPoolExecutor thePoolSize (int)的最大线程池大小。这是可以在不拒绝 HystrixCommands 的情况下支持的最大并发量。请注意，只有在您设置了 allowMaximumSizeToDivergeFromCoreSize 时，此设置才会生效。	10	整数
camel.hystrix.metrics-health-snapshot-interval-in-milliseconds	在允许健康快照之间等待的时间（毫秒）来计算成功和错误百分比，并影响 HystrixCircuitBreaker.isOpen （）状态。在高容量上，错误百分比的持续计算可能会变得 CPU 密集型，从而控制计算的频率。	500	整数

Name	描述	默认值	类型
camel.hystrix.metrics-rolling-percentile-bucket-size	存储在滚动百分比的每个存储桶中的最大值数。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	10	整数
camel.hystrix.metrics-rolling-percentile-enabled	是否应该使用 HystrixRollingPercentile 在 HystrixCommandMetrics 中捕获百分比的指标。	true	布尔值
camel.hystrix.metrics-rolling-percentile-window-buckets	滚动百分比窗口的存储桶数。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	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。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	10000	整数
camel.hystrix.metrics-rolling-statistical-window-buckets	滚动统计信息窗口划分为的 bucket 数量。这会传递给 HystrixCommandMetrics 中的 HystrixRollingNumber。	10	整数
camel.hystrix.metrics-rolling-statistical-window-in-milliseconds	此属性以毫秒为单位设置统计滚动窗口的持续时间。这是为线程池保留的指标的时长。窗口划分为存储桶，按这些递增推出部署。	10000	整数
camel.hystrix.queue-size-rejection-threshold	队列大小拒绝阈值是一个人为最大大小，即使尚未达到 maxQueueSize，也会发生拒绝的最大值。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 相同的值。	Camel Hystrix	字符串

Name	描述	默认值	类型
camel.hystrix.thread-pool-rolling-number-statistical-window-buckets	滚动统计信息窗口划分为的 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
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

Name	描述	默认值	类型
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 waitDurationInOpenState 通过后，启用自动从 OPEN 转换到 HALF_OPEN 状态。	false	布尔值
camel.resilience4j.circuit-breaker-ref	引用现有的 io.github.resilience4j.circuitbreaker.CircuitBreaker 实例来查找并从 registry 中使用。使用时，不使用任何其他断路器选项。		字符串
camel.resilience4j.config-ref	引用现有的 io.github.resilience4j.circuitbreaker.CircuitBreakerConfig 实例来查找并从 registry 中使用。		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速率阈值。如果故障率等于或大于 CircuitBreaker 过渡到打开的阈值，并启动短路调用。阈值必须大于 0，且不能超过 100。默认值为 50 个百分比。		æµ®ç‚¹å€¼

Name	描述	默认值	类型
camel.resilience4j.minimum-number-of-calls	在 CircuitBreaker 可以计算错误率前，配置所需的最少调用（每个分片窗口周期）。例如，如果 minimumNumberOfCalls 为 10，则必须记录至少 10 个调用，然后才能计算失败率。如果记录了 9 个调用，CircuitBreaker 不会过渡到打开，即使所有 9 个调用都失败。默认 minimumNumberOfCalls 为 100。	100	整数
camel.resilience4j.permitted-number-of-calls-in-half-open-state	当 CircuitBreaker 处于一半时，配置允许的调用数量。大小必须大于 0。默认大小为 10。	10	整数
camel.resilience4j.sliding-window-size	配置 sliding 窗口的大小，用于记录 CircuitBreaker 关闭时调用的结果。slidingWindowSize 配置 sliding 窗口的大小。分片窗口可以是基于计数或基于时间的。如果 slidingWindowType 是 COUNT_BASED，则会记录并聚合最后一个 slidingWindowSize 调用。如果 slidingWindowType 是 TIME_BASED，则会记录并聚合最后一个 slidingWindowSize 秒的调用。slidingWindowSize 必须大于 0。minimumNumberOfCalls 必须大于 0。如果 slidingWindowType 是 COUNT_BASED，则 minimumNumberOfCalls 不能超过 slidingWindowSize。如果 slidingWindowType 是 TIME_BASED，您可以选择您需要的任何内容。默认 slidingWindowSize 为 100。	100	整数
camel.resilience4j.sliding-window-type	配置 sliding 窗口的类型，用于记录 CircuitBreaker 关闭时调用的结果。分片窗口可以是基于计数或基于时间的。如果 slidingWindowType 是 COUNT_BASED，则会记录并聚合最后一个 slidingWindowSize 调用。如果 slidingWindowType 是 TIME_BASED，则会记录并聚合最后一个 slidingWindowSize 秒的调用。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），调用被视为较慢，并提高较慢的调用百分比。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。当调用百分比相等或大于阈值时，CircuitBreaker 会过渡到打开并启动短路调用。阈值必须大于 0，且不能超过 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

Name	描述	默认值	类型
camel.resilience4j.wait-duration-in-open-state	配置等待持续时间（以秒为单位），指定 CircuitBreaker 在切换到一半打开前应保持打开的时长。默认值为 60 秒。	60	整数
camel.resilience4j.writable-stack-trace-enabled	启用可写堆栈跟踪。当设置为 false 时，Exception.getStack Trace 会返回零长度数组。当断路器处于打开状态时，这可用于减少日志垃圾邮件，因为例外的原因已经已知（断路器是短路的调用）。	true	布尔值
camel.rest.api-component	如果没有明确配置 API 组件，则作为 REST API （如 swagger）的 Camel 组件的名称（如 swagger），如果存在负责服务并生成 REST API 文档的 Camel 组件，或者 org.apache.camel.spi.RestApiProcessorFactory 在 registry 中注册，则 Camel 将查找。如果找到其中任一个，则会使用它。		字符串
camel.rest.api-context-path	设置 REST API 服务将使用的前导 API 上下文路径。这可用于使用 camel-servlet 等组件，其中部署的 Web 应用程序使用 context-path 部署。		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），可用于覆盖使用此配置的主机名生成的主机。		字符串
camel.rest.api-property	允许为 api 文档(swagger)配置多个附加属性。例如，将属性 api.title 设置为 mycolphone。		Map
camel.rest.api-vendor-extension	在 Rest API 中是否启用了厂商扩展。如果启用，则 Camel 将包含额外信息作为供应商扩展（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 标头是否受其消耗的/生成的设置的 Rest-DSL 配置支持。这可以打开它，以启用此检查。如果验证错误，则返回 HTTP 状态代码 415 或 406。默认值为 false。	false	布尔值

Name	描述	默认值	类型
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 将查找，如果有一个与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestConsumerFactory 在 registry 中注册。如果找到其中任一个，则会使用它。		字符串
camel.rest.component-property	允许为使用中的其他组件配置多个额外的属性。		Map
camel.rest.consumer-property	允许为使用中的其他使用者配置多个额外的属性。		Map
camel.rest.context-path	设置 REST 服务将使用的前导上下文路径。这可用于使用 camel-servlet 等组件，其中部署的 Web 应用程序使用 context-path 部署。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的属性。例如，将属性 prettyPrint 设置为 true，使 json 在用户模式中输出。属性可以加上前缀，以表示选项仅适用于 JSON 或 XML，对于 IN 或 OUT。前缀为： json.in. json.out. xml.in. xml.out。例如，值为 xml.out.mustBeHQBElement 的键仅适用于传出的 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

Name	描述	默认值	类型
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 85 章 CSIMPLE

CSimple 语言 被编译 简单 语言。

85.1. CSIMPLE 和 SIMPLE 之间的区别

简单的语言是动态表达式语言，运行时被解析为一组 Camel 表达式或 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，或通过 camel-csimple-joor 模块在 bootstrap 期间编译一次。

简单的语言通常是非常轻便且快速的，但对于某些用例，通过 OGNL 路径进行动态方法调用，然后简单的语言进行运行时内省和反映调用。这在性能方面有开销，这是创建 csimple 的原因之一。

csimple 语言需要类型safe，并且通过 OGNL 路径的方法调用需要在解析期间知道类型。这意味着，您需要在脚本中提供类类型的 csimple 语言表达式，而简单的内省则在运行时进行内省。

换句话说，简单语言使用 duck 类型（如果它看起来像一个 duck，并且像 duck），则 csimple 使用 Java 类型(typesafety)。如果存在类型错误，则 simple 将在运行时报告，并且 csimple 会出现 Java 编译错误。

85.1.1. 其他 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。在输入这些用例时，可使用以下功能 bodyAsIndex、headerAsIndex 和 exchangePropertyAsIndex 用于这些用例。

功能	类型	描述
bodyAsIndex(*type*, *index*)	类型	用于从现有 **集合**、**映射** 或数组（由索引查找）收集正文，然后将正文转换为由其 classname 确定的给定类型。转换的正文可以是 null。
mandatoryBodyAsIndex(*type*, *index*)	类型	用于从现有 **集合**、**映射** 或数组（由索引查找）收集正文，然后将正文转换为由其 classname 确定的给定类型。期望正文不是 null。

功能	类型	描述
headerAsIndex(*key*, *type*, *index*)	类型	用于从现有 **集合**、**映射** 或数组（由索引查找）收集标头值，然后将标头值转换为由其 classname 确定的给定类型。转换的标头可以是 null。
mandatoryHeaderAsIndex(*key*, *type*, *index*)	类型	用于从现有 **集合**、**映射** 或数组（由索引查找）收集标头值，然后将标头值转换为由其 classname 确定的给定类型。期望标头不是 null。
exchangePropertyAsIndex(*key*, *type*, *index*)	类型	用于从现有 **Collection**、**Map** 或数组（由索引查找）收集交换属性，然后将 exchange 属性转换为由其 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	用于从现有 **Collection**、**Map** 或数组（由索引查找）收集交换属性，然后将 exchange 属性转换为由其 classname 确定的给定类型。期望 exchange 属性不是 null。

例如，给定以下简单表达式：

```
Hello $\{body[0].name}
```

此脚本没有类型信息，简单语言将在运行时解决这个问题，方法是内省消息正文，如果基于集合查找第一个元素，则通过反映调用名为 getName 的方法。

在 csimple （编译）中，我们希望预编译这一点，因此最终用户必须使用 bodyAsIndex 函数提供类型信息：

```
Hello $\{bodyAsIndex(com.foo.MyUser, 0).name}
```

85.2. 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 **csimple 有两种方法***

- *在构建时使用 **camel-csimple-maven-plugin 生成源代码。***

- 使用 camel-csimple-joor，在 Camel 引导过程中执行运行时内存编译。

85.2.1. 使用 camel-csimple-maven-plugin

camel-csimple-maven-plugin Maven 插件用于从源代码发现所有 csimple 脚本，然后在 src/generated/java 文件夹中自动生成源代码，然后与所有其他源一起编译。

maven 插件将对 .java 和 .xml 文件(Java 和 XML DSL)进行源代码扫描。扫描程序限制用于检测某些代码模式，如果它们以不常/垃圾的方式使用，则可能会取消发现一些 csimple 脚本。

使用 camel-csimple-joor 的运行时编译没有这个限制。

优点是，所有 csimple 脚本将使用常规 Java 编译器编译，因此所有都作为应用程序 JAR 文件中的 .class 文件包含在方框中，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，以将 src/generated 包含在 Java 编译器的源文件夹列表中，以确保编译生成的源代码并包含在应用程序 JAR 文件中。

请参阅 Camel Examples 中的 camel-example-csimple 示例，它使用 maven 插件。

85.2.2. 使用 camel-csimple-joor

jOOR 库与 Java 编译器集成，并执行 Java 代码的运行时编译。

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。这个功能在 Ice、Camel Swarm 或任何类型的 Java Application Server 运行时不支持。

joor 不支持使用 fat jar 打包的 Spring Boot 的运行时编译 (https://github.com/jOOQ/jOOR/issues/69)，它可用于探索的类路径。

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。

85.3. 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。

85.4. 限制

目前，csimple 语言 不支持 ：

- *嵌套功能（功能内的功能）*
- *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
```

85.5. 自动导入

csimple 语言将从中自动导入：

```
import java.util.*;
import java.util.concurrent.*;
import java.util.stream.*;
import org.apache.camel.*;
import org.apache.camel.util.*;
```

85.6. 配置文件

您可以在 camel-csimple.properties 文件中配置 csimple 语言，该文件是从根 classpath 加载的。

例如，您可以通过添加以下内容在 camel-csimple.properties 文件中添加额外的导入：

```
import com.foo.MyUser;
import com.bar.*;
import static com.foo.MyHelper.*;
```

您还可以添加别名(key=value)，其中别名将被用作代码中的缩写。

```
echo()=${bodyAs(String)} ${bodyAs(String)}
```

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，从而生成一个脚本：

```
    .transform(csimple("Hello ${bodyAs(String)} ${bodyAs(String)}"))
```

85.7. 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。

85.8. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	设置与 Consul 搭配使用的 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 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服务调用实施。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	为 OkHttpClient 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服务调用实施。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	所需的服务的传输协议。	_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服务调用实施。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	要设置操作完成的最长时间。		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	在使用客户端查找时设置 API 版本。		字符串
camel.cloud.kubernetes.service-discovery.ca-cert-data	在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	设置在使用客户端查找时从文件载入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	在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	设置在使用客户端查找时从文件载入的客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-algo	设置客户端密钥存储算法，如在使用客户端查找时的 RSA。		字符串
camel.cloud.kubernetes.service-discovery.client-key-data	在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	设置在使用客户端查找时从文件载入的客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-passphrase	在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 查询的 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、环境。在使用客户端时，客户端会查询 kubernetes master 以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，服务名称通过 SRV 查询解析 ....svc... when 使用环境变量来查找该服务。默认使用默认环境。	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，设置 OAUTH 令牌以进行身份验证（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服务调用实施。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

Name	描述	默认值	类型
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服务调用实施。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 maximumSize 生效。然后该值可以等于 coreSize 或更高。	false	布尔值
camel.hystrix.circuit-breaker-enabled	是否使用 HystrixCircuitBreaker。如果为 false，则使用断路器逻辑以及允许的所有请求。这与 hardBreakerForceClosed （）的影响类似，但继续跟踪指标并了解它是否是打开/披露，此属性甚至不实例化断路器。	true	布尔值

Name	描述	默认值	类型
camel.hystrix.circuit-breaker-error-threshold-percentage	一个错误百分比阈值（如 50），其中断路器将打开和拒绝请求。它将持续等待在断路器 BreakerSleepWindowInMilliseconds 中定义的持续时间；这与 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，HystrixCircuitBreaker114allowRequest （）始终返回 true 以允许请求，而不考虑 HystrixCommandMetrics.getHealthCounts （）的错误百分比。paraBreakerForceOpen （）属性具有优先权，因此如果设为 true，则它将不做任何操作。	false	布尔值
camel.hystrix.circuit-breaker-force-open	如果为 true，HystrixCircuitBreaker.allowRequest （）将始终返回 false，从而导致断路器打开（往返）并拒绝所有请求。This property takes precedence over circuitBreakerForceClosed();.	false	布尔值
camel.hystrix.circuit-breaker-request-volume-threshold	在 HystrixCircuitBreaker 之前必须存在 metricsRollingStatisticalWindowInMilliseconds （）中的最小请求数。如果低于这个数字，则无论错误百分比如何，时钟都不会往返。	20	整数
camel.hystrix.circuit-breaker-sleep-window-in-milliseconds	HystrixCircuitBreaker 往返后的时间（毫秒）打开它应该等待，然后再再次尝试请求。	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.ThreadPoolExecutor114setCorePoolSize (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 executionIsolationStrategy == SEMAPHORE 时才适用。	20	整数

Name	描述	默认值	类型
camel.hystrix.execution-isolation-strategy	将执行哪些隔离策略 HystrixCommand.run （）。如果 THREAD，那么它将在单独的线程上执行，并发请求则受 thread-pool 中的线程数量限制。如果 SEMAPHORE，它将在调用线程和并发请求上执行，则由 semaphore 数限制。	THREAD	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（使用将来的站已取消。仅在 executionIsolationStrategy （）== THREAD 时才适用。	true	布尔值
camel.hystrix.execution-timeout-enabled	是否为这个命令启用超时机制。	true	布尔值
camel.hystrix.execution-timeout-in-milliseconds	在一段时间内，命令将超时和停止执行的时间（毫秒）。如果 executionIsolationThreadInterruptOnTimeout == true，命令是线程隔离，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	失败时是否应尝试 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失败，且不会尝试检索回退。	10	整数
camel.hystrix.group-key	设置要使用的组密钥。默认值为 CamelHystrix。	Camel Hystrix	字符串
camel.hystrix.keep-alive-time	keep-alive 时间（以分钟为单位）传递给 ThreadPoolExecutor，或 theepAliveTime (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 - 不会影响实时更改队列大小。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 thePoolSize (int)的最大线程池大小。这是可以在不拒绝 HystrixCommands 的情况下支持的最大并发量。请注意，只有在您设置了 allowMaximumSizeToDivergeFromCoreSize 时，此设置才会生效。	10	整数
camel.hystrix.metrics-health-snapshot-interval-in-milliseconds	在允许健康快照之间等待的时间（毫秒）来计算成功和错误百分比，并影响 HystrixCircuitBreaker.isOpen（）状态。在高容量上，错误百分比的持续计算可能会变得 CPU 密集型，从而控制计算的频率。	500	整数
camel.hystrix.metrics-rolling-percentile-bucket-size	存储在滚动百分比的每个存储桶中的最大值数。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	10	整数
camel.hystrix.metrics-rolling-percentile-enabled	是否应该使用 HystrixRollingPercentile 在 HystrixCommandMetrics 中捕获百分比的指标。	true	布尔值
camel.hystrix.metrics-rolling-percentile-window-buckets	滚动百分比窗口的存储桶数。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	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。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	10000	整数
camel.hystrix.metrics-rolling-statistical-window-buckets	滚动统计信息窗口划分为的 bucket 数量。这会传递给 HystrixCommandMetrics 中的 HystrixRollingNumber。	10	整数
camel.hystrix.metrics-rolling-statistical-window-in-milliseconds	此属性以毫秒为单位设置统计滚动窗口的持续时间。这是为线程池保留的指标的时长。窗口划分为存储桶，按这些递增推出部署。	10000	整数
camel.hystrix.queue-size-rejection-threshold	队列大小拒绝阈值是一个人为最大大小，即使尚未达到 maxQueueSize，也会发生拒绝的最大值。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 相同的值。	Camel Hystrix	字符串
camel.hystrix.thread-pool-rolling-number-statistical-window-buckets	滚动统计信息窗口划分为的 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 waitDurationInOpenState 通过后，启用自动从 OPEN 转换到 HALF_OPEN 状态。	false	布尔值
camel.resilience4j.circuit-breaker-ref	引用现有的 io.github.resilience4j.circuitbreaker.CircuitBreaker 实例来查找并从 registry 中使用。使用时，不使用任何其他断路器选项。		字符串
camel.resilience4j.config-ref	引用现有的 io.github.resilience4j.circuitbreaker.CircuitBreakerConfig 实例来查找并从 registry 中使用。		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速率阈值。如果故障率等于或大于 CircuitBreaker 过渡到打开的阈值，并启动短路调用。阈值必须大于 0，且不能超过 100。默认值为 50 个百分比。		æµ®ç‚¹å€¼
camel.resilience4j.minimum-number-of-calls	在 CircuitBreaker 可以计算错误率前，配置所需的最少调用（每个分片窗口周期）。例如，如果 minimumNumberOfCalls 为 10，则必须记录至少 10 个调用，然后才能计算失败率。如果记录了 9 个调用，CircuitBreaker 不会过渡到打开，即使所有 9 个调用都失败。默认 minimumNumberOfCalls 为 100。	100	整数
camel.resilience4j.permitted-number-of-calls-in-half-open-state	当 CircuitBreaker 处于一半时，配置允许的调用数量。大小必须大于 0。默认大小为 10。	10	整数
camel.resilience4j.sliding-window-size	配置 sliding 窗口的大小，用于记录 CircuitBreaker 关闭时调用的结果。slidingWindowSize 配置 sliding 窗口的大小。分片窗口可以是基于计数或基于时间的。如果 slidingWindowType 是 COUNT_BASED，则会记录并聚合最后一个 slidingWindowSize 调用。如果 slidingWindowType 是 TIME_BASED，则会记录并聚合最后一个 slidingWindowSize 秒的调用。slidingWindowSize 必须大于 0。minimumNumberOfCalls 必须大于 0。如果 slidingWindowType 是 COUNT_BASED，则 minimumNumberOfCalls 不能超过 slidingWindowSize。如果 slidingWindowType 是 TIME_BASED，您可以选择您需要的任何内容。默认 slidingWindowSize 为 100。	100	整数
camel.resilience4j.sliding-window-type	配置 sliding 窗口的类型，用于记录 CircuitBreaker 关闭时调用的结果。分片窗口可以是基于计数或基于时间的。如果 slidingWindowType 是 COUNT_BASED，则会记录并聚合最后一个 slidingWindowSize 调用。如果 slidingWindowType 是 TIME_BASED，则会记录并聚合最后一个 slidingWindowSize 秒的调用。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），调用被视为较慢，并提高较慢的调用百分比。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。当调用百分比相等或大于阈值时，CircuitBreaker 会过渡到打开并启动短路调用。阈值必须大于 0，且不能超过 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 在切换到一半打开前应保持打开的时长。默认值为 60 秒。	60	整数
camel.resilience4j.writable-stack-trace-enabled	启用可写堆栈跟踪。当设置为 false 时，Exception.getStack Trace 会返回零长度数组。当断路器处于打开状态时，这可用于减少日志垃圾邮件，因为例外的原因已经已知（断路器是短路的调用）。	true	布尔值
camel.rest.api-component	如果没有明确配置 API 组件，则作为 REST API （如 swagger）的 Camel 组件的名称（如 swagger），如果存在负责服务并生成 REST API 文档的 Camel 组件，或者 org.apache.camel.spi.RestApiProcessorFactory 在 registry 中注册，则 Camel 将查找。如果找到其中任一个，则会使用它。		字符串
camel.rest.api-context-path	设置 REST API 服务将使用的前导 API 上下文路径。这可用于使用 camel-servlet 等组件，其中部署的 Web 应用程序使用 context-path 部署。		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），可用于覆盖使用此配置的主机名生成的主机。		字符串
camel.rest.api-property	允许为 api 文档(swagger)配置多个附加属性。例如，将属性 api.title 设置为 mycolphone。		Map
camel.rest.api-vendor-extension	在 Rest API 中是否启用了厂商扩展。如果启用，则 Camel 将包含额外信息作为供应商扩展（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 标头是否受其消耗的/生成的设置的 Rest-DSL 配置支持。这可以打开它，以启用此检查。如果验证错误，则返回 HTTP 状态代码 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 将查找，如果有一个与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestConsumerFactory 在 registry 中注册。如果找到其中任一个，则会使用它。		字符串
camel.rest.component-property	允许为使用中的其他组件配置多个额外的属性。		Map
camel.rest.consumer-property	允许为使用中的其他使用者配置多个额外的属性。		Map
camel.rest.context-path	设置 REST 服务将使用的前导上下文路径。这可用于使用 camel-servlet 等组件，其中部署的 Web 应用程序使用 context-path 部署。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的属性。例如，将属性 prettyPrint 设置为 true，使 json 在用户模式中输出。属性可以加上前缀，以表示选项仅适用于 JSON 或 XML，对于 IN 或 OUT。前缀为： json.in. json.out. xml.in. xml.out。例如，值为 xml.out.mustBeHQBElement 的键仅适用于传出的 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 86 章 EXCHANGEPROPERTY

ExchangeProperty 表达式语言允许您提取命名交换属性的值。

86.1. 交换属性选项

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。

86.2. 示例

receiverList EIP 可以使用如下交换属性：

```
<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 中的同一示例：

```
from("direct:a").recipientList(exchangeProperty("myProperty"));
```

86.3. 依赖项

ExchangeProperty 语言是 camel-core 的一部分。

86.4. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	设置与 Consul 搭配使用的 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 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服务调用实施。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	为 OkHttpClient 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服务调用实施。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	所需的服务的传输协议。	_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服务调用实施。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	要设置操作完成的最长时间。		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	在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	在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	设置在使用客户端查找时从文件载入的证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-data	在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	设置在使用客户端查找时从文件载入的客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-algo	设置客户端密钥存储算法，如在使用客户端查找时的 RSA。		字符串
camel.cloud.kubernetes.service-discovery.client-key-data	在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	设置在使用客户端查找时从文件载入的客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-passphrase	在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 查询的 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、环境。在使用客户端时，客户端会查询 kubernetes master 以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，服务名称通过 SRV 查询解析 ....svc... when 使用环境变量来查找该服务。默认使用默认环境。	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，设置 OAUTH 令牌以进行身份验证（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服务调用实施。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服务调用实施。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 maximumSize 生效。然后该值可以等于 coreSize 或更高。	false	布尔值
camel.hystrix.circuit-breaker-enabled	是否使用 HystrixCircuitBreaker。如果为 false，则使用断路器逻辑以及允许的所有请求。这与 hardBreakerForceClosed （）的影响类似，但继续跟踪指标并了解它是否是打开/披露，此属性甚至不实例化断路器。	true	布尔值
camel.hystrix.circuit-breaker-error-threshold-percentage	一个错误百分比阈值（如 50），其中断路器将打开和拒绝请求。它将持续等待在断路器 BreakerSleepWindowInMilliseconds 中定义的持续时间；这与 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，HystrixCircuitBreaker114allowRequest （）始终返回 true 以允许请求，而不考虑 HystrixCommandMetrics.getHealthCounts （）的错误百分比。paraBreakerForceOpen （）属性具有优先权，因此如果设为 true，则它将不做任何操作。	false	布尔值
camel.hystrix.circuit-breaker-force-open	如果为 true，HystrixCircuitBreaker.allowRequest （）将始终返回 false，从而导致断路器打开（往返）并拒绝所有请求。This property takes precedence over circuitBreakerForceClosed();.	false	布尔值
camel.hystrix.circuit-breaker-request-volume-threshold	在 HystrixCircuitBreaker 之前必须存在 metricsRollingStatisticalWindowInMilliseconds （）中的最小请求数。如果低于这个数字，则无论错误百分比如何，时钟都不会往返。	20	整数
camel.hystrix.circuit-breaker-sleep-window-in-milliseconds	HystrixCircuitBreaker 往返后的时间（毫秒）打开它应该等待，然后再再次尝试请求。	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.ThreadPoolExecutor114setCorePoolSize (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 executionIsolationStrategy == SEMAPHORE 时才适用。	20	整数
camel.hystrix.execution-isolation-strategy	将执行哪些隔离策略 HystrixCommand.run （）。如果 THREAD，那么它将在单独的线程上执行，并发请求则受 thread-pool 中的线程数量限制。如果 SEMAPHORE，它将在调用线程和并发请求上执行，则由 semaphore 数限制。	THREAD	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（使用将来的站已取消。仅在 executionIsolationStrategy （）== THREAD 时才适用。	true	布尔值
camel.hystrix.execution-timeout-enabled	是否为这个命令启用超时机制。	true	布尔值
camel.hystrix.execution-timeout-in-milliseconds	在一段时间内，命令将超时和停止执行的时间（毫秒）。如果 executionIsolationThreadInterruptOnTimeout == true，命令是线程隔离，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	失败时是否应尝试 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失败，且不会尝试检索回退。	10	整数
camel.hystrix.group-key	设置要使用的组密钥。默认值为 CamelHystrix。	Camel Hystrix	字符串
camel.hystrix.keep-alive-time	keep-alive 时间（以分钟为单位）传递给 ThreadPoolExecutor，或 theepAliveTime (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 - 不会影响实时更改队列大小。为此，请使用 queueSizeRejectionThreshold （）。	-1	整数
camel.hystrix.maximum-size	传递给 ThreadPoolExecutor thePoolSize (int)的最大线程池大小。这是可以在不拒绝 HystrixCommands 的情况下支持的最大并发量。请注意，只有在您设置了 allowMaximumSizeToDivergeFromCoreSize 时，此设置才会生效。	10	整数
camel.hystrix.metrics-health-snapshot-interval-in-milliseconds	在允许健康快照之间等待的时间（毫秒）来计算成功和错误百分比，并影响 HystrixCircuitBreaker.isOpen （）状态。在高容量上，错误百分比的持续计算可能会变得 CPU 密集型，从而控制计算的频率。	500	整数
camel.hystrix.metrics-rolling-percentile-bucket-size	存储在滚动百分比的每个存储桶中的最大值数。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	10	整数
camel.hystrix.metrics-rolling-percentile-enabled	是否应该使用 HystrixRollingPercentile 在 HystrixCommandMetrics 中捕获百分比的指标。	true	布尔值
camel.hystrix.metrics-rolling-percentile-window-buckets	滚动百分比窗口的存储桶数。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	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。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	10000	整数
camel.hystrix.metrics-rolling-statistical-window-buckets	滚动统计信息窗口划分为的 bucket 数量。这会传递给 HystrixCommandMetrics 中的 HystrixRollingNumber。	10	整数
camel.hystrix.metrics-rolling-statistical-window-in-milliseconds	此属性以毫秒为单位设置统计滚动窗口的持续时间。这是为线程池保留的指标的时长。窗口划分为存储桶，按这些递增推出部署。	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一个人为最大大小，即使尚未达到 maxQueueSize，也会发生拒绝的最大值。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 相同的值。	Camel Hystrix	字符串
camel.hystrix.thread-pool-rolling-number-statistical-window-buckets	滚动统计信息窗口划分为的 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 waitDurationInOpenState 通过后，启用自动从 OPEN 转换到 HALF_OPEN 状态。	false	布尔值
camel.resilience4j.circuit-breaker-ref	引用现有的 io.github.resilience4j.circuitbreaker.CircuitBreaker 实例来查找并从 registry 中使用。使用时，不使用任何其他断路器选项。		字符串
camel.resilience4j.config-ref	引用现有的 io.github.resilience4j.circuitbreaker.CircuitBreakerConfig 实例来查找并从 registry 中使用。		字符串
camel.resilience4j.configurations	定义其他配置定义。		Map

Name	描述	默认值	类型
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速率阈值。如果故障率等于或大于 CircuitBreaker 过渡到打开的阈值，并启动短路调用。阈值必须大于 0，且不能超过 100。默认值为 50 个百分比。		æµ®ç‚¹â€¼
camel.resilience4j.minimum-number-of-calls	在 CircuitBreaker 可以计算错误率前，配置所需的最少调用（每个分片窗口周期）。例如，如果 minimumNumberOfCalls 为 10，则必须记录至少 10 个调用，然后才能计算失败率。如果记录了 9 个调用，CircuitBreaker 不会过渡到打开，即使所有 9 个调用都失败。默认 minimumNumberOfCalls 为 100。	100	整数
camel.resilience4j.permitted-number-of-calls-in-half-open-state	当 CircuitBreaker 处于一半时，配置允许的调用数量。大小必须大于 0。默认大小为 10。	10	整数
camel.resilience4j.sliding-window-size	配置 sliding 窗口的大小，用于记录 CircuitBreaker 关闭时调用的结果。slidingWindowSize 配置 sliding 窗口的大小。分片窗口可以是基于计数或基于时间的。如果 slidingWindowType 是 COUNT_BASED，则会记录并聚合最后一个 slidingWindowSize 调用。如果 slidingWindowType 是 TIME_BASED，则会记录并聚合最后一个 slidingWindowSize 秒的调用。slidingWindowSize 必须大于 0。minimumNumberOfCalls 必须大于 0。如果 slidingWindowType 是 COUNT_BASED，则 minimumNumberOfCalls 不能超过 slidingWindowSize。如果 slidingWindowType 是 TIME_BASED，您可以选择您需要的任何内容。默认 slidingWindowSize 为 100。	100	整数
camel.resilience4j.sliding-window-type	配置 sliding 窗口的类型，用于记录 CircuitBreaker 关闭时调用的结果。分片窗口可以是基于计数或基于时间的。如果 slidingWindowType 是 COUNT_BASED，则会记录并聚合最后一个 slidingWindowSize 调用。如果 slidingWindowType 是 TIME_BASED，则会记录并聚合最后一个 slidingWindowSize 秒的调用。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），调用被视为较慢，并提高较慢的调用百分比。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。当调用百分比相等或大于阈值时，CircuitBreaker 会过渡到打开并启动短路调用。阈值必须大于 0，且不能超过 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 在切换到一半打开前应保持打开的时长。默认值为 60 秒。	60	整数
camel.resilience4j.writable-stack-trace-enabled	启用可写堆栈跟踪。当设置为 false 时，Exception.getStack Trace 会返回零长度数组。当断路器处于打开状态时，这可用于减少日志垃圾邮件，因为例外的原因已经已知（断路器是短路的调用）。	true	布尔值
camel.rest.api-component	如果没有明确配置 API 组件，则作为 REST API （如 swagger）的 Camel 组件的名称（如 swagger），如果存在负责服务并生成 REST API 文档的 Camel 组件，或者 org.apache.camel.spi.RestApiProcessorFactory 在 registry 中注册，则 Camel 将查找。如果找到其中任一个，则会使用它。		字符串
camel.rest.api-context-path	设置 REST API 服务将使用的前导 API 上下文路径。这可用于使用 camel-servlet 等组件，其中部署的 Web 应用程序使用 context-path 部署。		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），可用于覆盖使用此配置的主机名生成的主机。		字符串
camel.rest.api-property	允许为 api 文档(swagger)配置多个附加属性。例如，将属性 api.title 设置为 mycolphone。		Map
camel.rest.api-vendor-extension	在 Rest API 中是否启用了厂商扩展。如果启用，则 Camel 将包含额外信息作为供应商扩展（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 标头是否受其消耗的/生成的设置的 Rest-DSL 配置支持。这可以打开它，以启用此检查。如果验证错误，则返回 HTTP 状态代码 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 将查找，如果有一个与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestConsumerFactory 在 registry 中注册。如果找到其中任一个，则会使用它。		字符串
camel.rest.component-property	允许为使用中的其他组件配置多个额外的属性。		Map
camel.rest.consumer-property	允许为使用中的其他使用者配置多个额外的属性。		Map
camel.rest.context-path	设置 REST 服务将使用的前导上下文路径。这可用于使用 camel-servlet 等组件，其中部署的 Web 应用程序使用 context-path 部署。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的属性。例如，将属性 prettyPrint 设置为 true，使 json 在用户模式中输出。属性可以加上前缀，以表示选项仅适用于 JSON 或 XML，对于 IN 或 OUT。前缀为： json.in. json.out. xml.in. xml.out。例如，值为 xml.out.mustBeHQBElement 的键仅适用于传出的 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 87 章 FILE

文件表达式语言是语言的扩展，添加了与文件相关的功能。这些功能与使用文件路径和名称的常见用例相关。目标是允许将表达式与

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，这意味着您可以在简单语言中直接使用所有文件语法。

87.1. 文件语言选项

文件语言支持 2 个选项，如下所列。

Name	默认值	Java 类型	描述
resultType		字符串	设置结果类型的类名称（输出中的类型）。
trim		布尔值	是否修剪值以移除前导和结尾的空格和换行符。

87.2. 语法

该语言是语言的 扩展，因此语法也适用。因此， 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 对象上的方法相同的表达式名称，例如 instance file: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 组件支持所有这些选项。

èi¨è¾¾å¼	类型	文件消费者	文件 Producer	FTP Consumer	FTP Producer	描述
file:name	字符串	是	否	是	否	请参考文件名（相对于起始目录，请参阅以下备注）

èi¨è¾¾å¼	类型	文件消费者	文件 Producer	FTP Consumer	FTP Producer	描述
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	文件扩展。如果文件扩展名有多个点，则此表达式剥离并仅返回最后一个部分。
file:name.noext	字符串	是	否	是	否	请参考没有扩展名的文件名（相对于起始目录，请参阅以下备注）
file:name.noext.single	字符串	是	否	是	否	请参考没有扩展名的文件名（相对于起始目录，请参阅以下备注）。如果文件扩展名有多个点，则此表达式仅剥离最后一个部分，并保留其他表达式。
file:onlyname	字符串	是	否	是	否	仅在没有前导路径的情况下引用文件名。
file:onlyname.noext	字符串	是	否	是	否	仅引用没有扩展名的文件名，且没有前导路径。
file:onlyname.noext.single	字符串	是	否	是	否	仅引用没有扩展名的文件名，且没有前导路径。如果文件扩展名有多个点，则此表达式仅剥离最后一个部分，并保留其他表达式。
file:ext	字符串	是	否	是	否	仅引用文件扩展
file:parent	字符串	是	否	是	否	指的是文件父项
file:path	字符串	是	否	是	否	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

èi¨è¾¾å¼	类型	文件消费者	文件 Producer	FTP Consumer	FTP Producer	描述
file:absolute.path	字符串	是	否	否	否	指的是绝对路径
file:length	Long	是	否	是	否	代表返回的文件长度为 Long 类型
file:size	Long	是	否	是	否	代表返回的文件长度为 Long 类型
file:modified	Date	是	否	是	否	引用最后修改的文件作为日期类型
date:_command:pattern_	字符串	是	是	是	是	对于使用 **java.text.SimpleDateFormat** 模式的日期格式。是语言的 **扩展**。额外的命令是： **文件** （仅限消费者）用于文件最后一次修改的时间戳。注意：也可以使用语言中的所有命令。

87.3. 文件令牌示例

87.3.1. 相对路径

我们为以下 相对 目录中的文件 hello.txt 有一个 java.io.File 处理： .\file language\test。我们将端点配置为使用此启动目录 .\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

èi¨è¾¾å¼	返回
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

87.3.2. 绝对路径

我们有以下 绝对 目录中的文件 hello.txt 的 java.io.File 处理： \workspace\camel\camel-core\target\file language\test。我们将端点配置为使用绝对启动目录 \workspace\camel\camel-core\target\file language。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
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

87.4. SAMPLES

您可以输入固定的文件名，如 myfile.txt ：

```
fileName="myfile.txt"
```

假设我们使用文件消费者读取文件，并希望移动读取文件以备份当前日期为子文件夹的文件夹。这可以通过使用类似如下的表达式来完成：

```
fileName="backup/${date:now:yyyyMMdd}/${file:name.noext}.bak"
```

也支持相对文件夹名称，因此假设备份文件夹应该是同级文件夹，然后您可以附加 ..，如下所示：

```
fileName="../backup/${date:now:yyyyMMdd}/${file:name.noext}.bak"
```

因为这是我们可从此语言访问所有好的语言的扩展，因此在这种情况下，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，我们也可以使用 bean 表达式来调用 POJO 类，该类会生成一些字符串输出（或转换为 String）：

```
fileName="uniquefile-${bean:myguidgenerator.generateid}.txt"
```

在课程中，这可合并到一个表达式中，您可以在一个组合表达式中使用 和 语言。这对这些通用文件路径模式非常强大。

87.5. 依赖项

文件语言是 camel-core 的一部分。

87.6. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	设置与 Consul 搭配使用的 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 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服务调用实施。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	为 OkHttpClient 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服务调用实施。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	所需的服务的传输协议。	_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服务调用实施。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	要设置操作完成的最长时间。		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	在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	在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	设置在使用客户端查找时从文件载入的证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-data	在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	设置在使用客户端查找时从文件载入的客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-algo	设置客户端密钥存储算法，如在使用客户端查找时的 RSA。		字符串
camel.cloud.kubernetes.service-discovery.client-key-data	在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	设置在使用客户端查找时从文件载入的客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-passphrase	在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 查询的 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、环境。在使用客户端时，客户端会查询 kubernetes master 以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，服务名称通过 SRV 查询解析 ....svc... when 使用环境变量来查找该服务。默认使用默认环境。	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，设置 OAUTH 令牌以进行身份验证（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服务调用实施。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服务调用实施。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 maximumSize 生效。然后该值可以等于 coreSize 或更高。	false	布尔值
camel.hystrix.circuit-breaker-enabled	是否使用 HystrixCircuitBreaker。如果为 false，则使用断路器逻辑以及允许的所有请求。这与 hardBreakerForceClosed （）的影响类似，但继续跟踪指标并了解它是否是打开/披露，此属性甚至不实例化断路器。	true	布尔值
camel.hystrix.circuit-breaker-error-threshold-percentage	一个错误百分比阈值（如 50），其中断路器将打开和拒绝请求。它将持续等待在断路器 BreakerSleepWindowInMilliseconds 中定义的持续时间；这与 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，HystrixCircuitBreaker114allowRequest （）始终返回 true 以允许请求，而不考虑 HystrixCommandMetrics.getHealthCounts （）的错误百分比。paraBreakerForceOpen （）属性具有优先权，因此如果设为 true，则它将不做任何操作。	false	布尔值
camel.hystrix.circuit-breaker-force-open	如果为 true，HystrixCircuitBreaker.allowRequest （）将始终返回 false，从而导致断路器打开（往返）并拒绝所有请求。This property takes precedence over circuitBreakerForceClosed();.	false	布尔值
camel.hystrix.circuit-breaker-request-volume-threshold	在 HystrixCircuitBreaker 之前必须存在 metricsRollingStatisticalWindowInMilliseconds （）中的最小请求数。如果低于这个数字，则无论错误百分比如何，时钟都不会往返。	20	整数
camel.hystrix.circuit-breaker-sleep-window-in-milliseconds	HystrixCircuitBreaker 往返后的时间（毫秒）打开它应该等待，然后再再次尝试请求。	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.ThreadPoolExecutor114setCorePoolSize (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 executionIsolationStrategy == SEMAPHORE 时才适用。	20	整数
camel.hystrix.execution-isolation-strategy	将执行哪些隔离策略 HystrixCommand.run （）。如果 THREAD，那么它将在单独的线程上执行，并发请求则受 thread-pool 中的线程数量限制。如果 SEMAPHORE，它将在调用线程和并发请求上执行，则由 semaphore 数限制。	THREAD	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（使用将来的站已取消。仅在 executionIsolationStrategy （）== THREAD 时才适用。	true	布尔值
camel.hystrix.execution-timeout-enabled	是否为这个命令启用超时机制。	true	布尔值
camel.hystrix.execution-timeout-in-milliseconds	在一段时间内，命令将超时和停止执行的时间（毫秒）。如果 executionIsolationThreadInterruptOnTimeout == true，命令是线程隔离，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	失败时是否应尝试 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失败，且不会尝试检索回退。	10	整数
camel.hystrix.group-key	设置要使用的组密钥。默认值为 CamelHystrix。	CamelHystrix	字符串
camel.hystrix.keep-alive-time	keep-alive 时间（以分钟为单位）传递给 ThreadPoolExecutor，或 theepAliveTime (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 - 不会影响实时更改队列大小。为此，请使用 queueSizeRejectionThreshold （）。	-1	整数
camel.hystrix.maximum-size	传递给 ThreadPoolExecutor thePoolSize (int)的最大线程池大小。这是可以在不拒绝 HystrixCommands 的情况下支持的最大并发量。请注意，只有在您设置了 allowMaximumSizeToDivergeFromCoreSize 时，此设置才会生效。	10	整数
camel.hystrix.metrics-health-snapshot-interval-in-milliseconds	在允许健康快照之间等待的时间（毫秒）来计算成功和错误百分比，并影响 HystrixCircuitBreaker.isOpen （）状态。在高容量上，错误百分比的持续计算可能会变得 CPU 密集型，从而控制计算的频率。	500	整数
camel.hystrix.metrics-rolling-percentile-bucket-size	存储在滚动百分比的每个存储桶中的最大值数。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	10	整数
camel.hystrix.metrics-rolling-percentile-enabled	是否应该使用 HystrixRollingPercentile 在 HystrixCommandMetrics 中捕获百分比的指标。	true	布尔值
camel.hystrix.metrics-rolling-percentile-window-buckets	滚动百分比窗口的存储桶数。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	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。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	10000	整数
camel.hystrix.metrics-rolling-statistical-window-buckets	滚动统计信息窗口划分为的 bucket 数量。这会传递给 HystrixCommandMetrics 中的 HystrixRollingNumber。	10	整数
camel.hystrix.metrics-rolling-statistical-window-in-milliseconds	此属性以毫秒为单位设置统计滚动窗口的持续时间。这是为线程池保留的指标的时长。窗口划分为存储桶，按这些递增推出部署。	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一个人为最大大小，即使尚未达到 maxQueueSize，也会发生拒绝的最大值。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 相同的值。	Camel Hystrix	字符串
camel.hystrix.thread-pool-rolling-number-statistical-window-buckets	滚动统计信息窗口划分为的 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 waitDurationInOpenState 通过后，启用自动从 OPEN 转换到 HALF_OPEN 状态。	false	布尔值
camel.resilience4j.circuit-breaker-ref	引用现有的 io.github.resilience4j.circuitbreaker.CircuitBreaker 实例来查找并从 registry 中使用。使用时，不使用任何其他断路器选项。		字符串
camel.resilience4j.config-ref	引用现有的 io.github.resilience4j.circuitbreaker.CircuitBreakerConfig 实例来查找并从 registry 中使用。		字符串
camel.resilience4j.configurations	定义其他配置定义。		Map

Name	描述	默认值	类型
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速率阈值。如果故障率等于或大于 CircuitBreaker 过渡到打开的阈值，并启动短路调用。阈值必须大于 0，且不能超过 100。默认值为 50 个百分比。		æµ®ç‚¹â€¼
camel.resilience4j.minimum-number-of-calls	在 CircuitBreaker 可以计算错误率前，配置所需的最少调用（每个分片窗口周期）。例如，如果 minimumNumberOfCalls 为 10，则必须记录至少 10 个调用，然后才能计算失败率。如果记录了 9 个调用，CircuitBreaker 不会过渡到打开，即使所有 9 个调用都失败。默认 minimumNumberOfCalls 为 100。	100	整数
camel.resilience4j.permitted-number-of-calls-in-half-open-state	当 CircuitBreaker 处于一半时，配置允许的调用数量。大小必须大于 0。默认大小为 10。	10	整数
camel.resilience4j.sliding-window-size	配置 sliding 窗口的大小，用于记录 CircuitBreaker 关闭时调用的结果。slidingWindowSize 配置 sliding 窗口的大小。分片窗口可以是基于计数或基于时间的。如果 slidingWindowType 是 COUNT_BASED，则会记录并聚合最后一个 slidingWindowSize 调用。如果 slidingWindowType 是 TIME_BASED，则会记录并聚合最后一个 slidingWindowSize 秒的调用。slidingWindowSize 必须大于 0。minimumNumberOfCalls 必须大于 0。如果 slidingWindowType 是 COUNT_BASED，则 minimumNumberOfCalls 不能超过 slidingWindowSize。如果 slidingWindowType 是 TIME_BASED，您可以选择您需要的任何内容。默认 slidingWindowSize 为 100。	100	整数
camel.resilience4j.sliding-window-type	配置 sliding 窗口的类型，用于记录 CircuitBreaker 关闭时调用的结果。分片窗口可以是基于计数或基于时间的。如果 slidingWindowType 是 COUNT_BASED，则会记录并聚合最后一个 slidingWindowSize 调用。如果 slidingWindowType 是 TIME_BASED，则会记录并聚合最后一个 slidingWindowSize 秒的调用。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），调用被视为较慢，并提高较慢的调用百分比。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。当调用百分比相等或大于阈值时，CircuitBreaker 会过渡到打开并启动短路调用。阈值必须大于 0，且不能超过 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 在切换到一半打开前应保持打开的时长。默认值为 60 秒。	60	整数
camel.resilience4j.writable-stack-trace-enabled	启用可写堆栈跟踪。当设置为 false 时，Exception.getStack Trace 会返回零长度数组。当断路器处于打开状态时，这可用于减少日志垃圾邮件，因为例外的原因已经已知（断路器是短路的调用）。	true	布尔值
camel.rest.api-component	如果没有明确配置 API 组件，则作为 REST API （如 swagger）的 Camel 组件的名称（如 swagger），如果存在负责服务并生成 REST API 文档的 Camel 组件，或者 org.apache.camel.spi.RestApiProcessorFactory 在 registry 中注册，则 Camel 将查找。如果找到其中任一个，则会使用它。		字符串
camel.rest.api-context-path	设置 REST API 服务将使用的前导 API 上下文路径。这可用于使用 camel-servlet 等组件，其中部署的 Web 应用程序使用 context-path 部署。		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），可用于覆盖使用此配置的主机名生成的主机。		字符串
camel.rest.api-property	允许为 api 文档(swagger)配置多个附加属性。例如，将属性 api.title 设置为 mycolphone。		Map
camel.rest.api-vendor-extension	在 Rest API 中是否启用了厂商扩展。如果启用，则 Camel 将包含额外信息作为供应商扩展（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 标头是否受其消耗的/生成的设置的 Rest-DSL 配置支持。这可以打开它，以启用此检查。如果验证错误，则返回 HTTP 状态代码 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 将查找，如果有一个与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestConsumerFactory 在 registry 中注册。如果找到其中任一个，则会使用它。		字符串
camel.rest.component-property	允许为使用中的其他组件配置多个额外的属性。		Map
camel.rest.consumer-property	允许为使用中的其他使用者配置多个额外的属性。		Map
camel.rest.context-path	设置 REST 服务将使用的前导上下文路径。这可用于使用 camel-servlet 等组件，其中部署的 Web 应用程序使用 context-path 部署。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的属性。例如，将属性 prettyPrint 设置为 true，使 json 在用户模式中输出。属性可以加上前缀，以表示选项仅适用于 JSON 或 XML，对于 IN 或 OUT。前缀为： json.in. json.out. xml.in. xml.out。例如，值为 xml.out.mustBeHQBElement 的键仅适用于传出的 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 88 章 标头

标头表达式语言允许您提取指定标头的值。

88.1. 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。

88.2. 用法示例

receiverList 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 中的同一示例：

```
from("direct:a").recipientList(header("myHeader"));
```

88.3. 依赖项

Header 语言是 camel-core 的一部分。

88.4. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	设置与 Consul 搭配使用的 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 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服务调用实施。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	为 OkHttpClient 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服务调用实施。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	所需的服务的传输协议。	_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服务调用实施。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	要设置操作完成的最长时间。		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	在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	在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	设置在使用客户端查找时从文件载入的证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-data	在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	设置在使用客户端查找时从文件载入的客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-algo	设置客户端密钥存储算法，如在使用客户端查找时的 RSA。		字符串
camel.cloud.kubernetes.service-discovery.client-key-data	在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	设置在使用客户端查找时从文件载入的客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-passphrase	在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 查询的 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、环境。在使用客户端时，客户端会查询 kubernetes master 以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，服务名称通过 SRV 查询解析 ....svc... when 使用环境变量来查找该服务。默认使用默认环境。	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，设置 OAUTH 令牌以进行身份验证（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服务调用实施。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服务调用实施。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 maximumSize 生效。然后该值可以等于 coreSize 或更高。	false	布尔值
camel.hystrix.circuit-breaker-enabled	是否使用 HystrixCircuitBreaker。如果为 false，则使用断路器逻辑以及允许的所有请求。这与 hardBreakerForceClosed （）的影响类似，但继续跟踪指标并了解它是否是打开/披露，此属性甚至不实例化断路器。	true	布尔值
camel.hystrix.circuit-breaker-error-threshold-percentage	一个错误百分比阈值（如 50），其中断路器将打开和拒绝请求。它将持续等待在断路器 BreakerSleepWindowInMilliseconds 中定义的持续时间；这与 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，HystrixCircuitBreaker114allowRequest （）始终返回 true 以允许请求，而不考虑 HystrixCommandMetrics.getHealthCounts （）的错误百分比。paraBreakerForceOpen （）属性具有优先权，因此如果设为 true，则它将不做任何操作。	false	布尔值
camel.hystrix.circuit-breaker-force-open	如果为 true，HystrixCircuitBreaker.allowRequest （）将始终返回 false，从而导致断路器打开（往返）并拒绝所有请求。This property takes precedence over circuitBreakerForceClosed();.	false	布尔值
camel.hystrix.circuit-breaker-request-volume-threshold	在 HystrixCircuitBreaker 之前必须存在 metricsRollingStatisticalWindowInMilliseconds （）中的最小请求数。如果低于这个数字，则无论错误百分比如何，时钟都不会往返。	20	整数
camel.hystrix.circuit-breaker-sleep-window-in-milliseconds	HystrixCircuitBreaker 往返后的时间（毫秒）打开它应该等待，然后再再次尝试请求。	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.ThreadPoolExecutor114setCorePoolSize (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 executionIsolationStrategy == SEMAPHORE 时才适用。	20	整数
camel.hystrix.execution-isolation-strategy	将执行哪些隔离策略 HystrixCommand.run （）。如果 THREAD，那么它将在单独的线程上执行，并发请求则受 thread-pool 中的线程数量限制。如果 SEMAPHORE，它将在调用线程和并发请求上执行，则由 semaphore 数限制。	THREAD	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（使用将来的站已取消。仅在 executionIsolationStrategy （）== THREAD 时才适用。	true	布尔值
camel.hystrix.execution-timeout-enabled	是否为这个命令启用超时机制。	true	布尔值
camel.hystrix.execution-timeout-in-milliseconds	在一段时间内，命令将超时和停止执行的时间（毫秒）。如果 executionIsolationThreadInterruptOnTimeout == true，命令是线程隔离，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	失败时是否应尝试 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失败，且不会尝试检索回退。	10	整数
camel.hystrix.group-key	设置要使用的组密钥。默认值为 CamelHystrix。	CamelHystrix	字符串
camel.hystrix.keep-alive-time	keep-alive 时间（以分钟为单位）传递给 ThreadPoolExecutor，或 theepAliveTime (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 - 不会影响实时更改队列大小。为此，请使用 queueSizeRejectionThreshold （）。	-1	整数
camel.hystrix.maximum-size	传递给 ThreadPoolExecutor thePoolSize (int)的最大线程池大小。这是可以在不拒绝 HystrixCommands 的情况下支持的最大并发量。请注意，只有在您设置了 allowMaximumSizeToDivergeFromCoreSize 时，此设置才会生效。	10	整数
camel.hystrix.metrics-health-snapshot-interval-in-milliseconds	在允许健康快照之间等待的时间（毫秒）来计算成功和错误百分比，并影响 HystrixCircuitBreaker.isOpen （）状态。在高容量上，错误百分比的持续计算可能会变得 CPU 密集型，从而控制计算的频率。	500	整数
camel.hystrix.metrics-rolling-percentile-bucket-size	存储在滚动百分比的每个存储桶中的最大值数。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	10	整数
camel.hystrix.metrics-rolling-percentile-enabled	是否应该使用 HystrixRollingPercentile 在 HystrixCommandMetrics 中捕获百分比的指标。	true	布尔值
camel.hystrix.metrics-rolling-percentile-window-buckets	滚动百分比窗口的存储桶数。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	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。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	10000	整数
camel.hystrix.metrics-rolling-statistical-window-buckets	滚动统计信息窗口划分为的 bucket 数量。这会传递给 HystrixCommandMetrics 中的 HystrixRollingNumber。	10	整数
camel.hystrix.metrics-rolling-statistical-window-in-milliseconds	此属性以毫秒为单位设置统计滚动窗口的持续时间。这是为线程池保留的指标的时长。窗口划分为存储桶，按这些递增推出部署。	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一个人为最大大小，即使尚未达到 maxQueueSize，也会发生拒绝的最大值。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 相同的值。	Camel Hystrix	字符串
camel.hystrix.thread-pool-rolling-number-statistical-window-buckets	滚动统计信息窗口划分为的 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 waitDurationInOpenState 通过后，启用自动从 OPEN 转换到 HALF_OPEN 状态。	false	布尔值
camel.resilience4j.circuit-breaker-ref	引用现有的 io.github.resilience4j.circuitbreaker.CircuitBreaker 实例来查找并从 registry 中使用。使用时，不使用任何其他断路器选项。		字符串
camel.resilience4j.config-ref	引用现有的 io.github.resilience4j.circuitbreaker.CircuitBreakerConfig 实例来查找并从 registry 中使用。		字符串
camel.resilience4j.configurations	定义其他配置定义。		Map

Name	描述	默认值	类型
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速率阈值。如果故障率等于或大于 CircuitBreaker 过渡到打开的阈值，并启动短路调用。阈值必须大于 0，且不能超过 100。默认值为 50 个百分比。		æµ®ç‚¹â€¼
camel.resilience4j.minimum-number-of-calls	在 CircuitBreaker 可以计算错误率前，配置所需的最少调用（每个分片窗口周期）。例如，如果 minimumNumberOfCalls 为 10，则必须记录至少 10 个调用，然后才能计算失败率。如果记录了 9 个调用，CircuitBreaker 不会过渡到打开，即使所有 9 个调用都失败。默认 minimumNumberOfCalls 为 100。	100	整数
camel.resilience4j.permitted-number-of-calls-in-half-open-state	当 CircuitBreaker 处于一半时，配置允许的调用数量。大小必须大于 0。默认大小为 10。	10	整数
camel.resilience4j.sliding-window-size	配置 sliding 窗口的大小，用于记录 CircuitBreaker 关闭时调用的结果。slidingWindowSize 配置 sliding 窗口的大小。分片窗口可以是基于计数或基于时间的。如果 slidingWindowType 是 COUNT_BASED，则会记录并聚合最后一个 slidingWindowSize 调用。如果 slidingWindowType 是 TIME_BASED，则会记录并聚合最后一个 slidingWindowSize 秒的调用。slidingWindowSize 必须大于 0。minimumNumberOfCalls 必须大于 0。如果 slidingWindowType 是 COUNT_BASED，则 minimumNumberOfCalls 不能超过 slidingWindowSize。如果 slidingWindowType 是 TIME_BASED，您可以选择您需要的任何内容。默认 slidingWindowSize 为 100。	100	整数
camel.resilience4j.sliding-window-type	配置 sliding 窗口的类型，用于记录 CircuitBreaker 关闭时调用的结果。分片窗口可以是基于计数或基于时间的。如果 slidingWindowType 是 COUNT_BASED，则会记录并聚合最后一个 slidingWindowSize 调用。如果 slidingWindowType 是 TIME_BASED，则会记录并聚合最后一个 slidingWindowSize 秒的调用。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），调用被视为较慢，并提高较慢的调用百分比。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。当调用百分比相等或大于阈值时，CircuitBreaker 会过渡到打开并启动短路调用。阈值必须大于 0，且不能超过 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 在切换到一半打开前应保持打开的时长。默认值为 60 秒。	60	整数
camel.resilience4j.writable-stack-trace-enabled	启用可写堆栈跟踪。当设置为 false 时，Exception.getStack Trace 会返回零长度数组。当断路器处于打开状态时，这可用于减少日志垃圾邮件，因为例外的原因已经已知（断路器是短路的调用）。	true	布尔值
camel.rest.api-component	如果没有明确配置 API 组件，则作为 REST API （如 swagger）的 Camel 组件的名称（如 swagger），如果存在负责服务并生成 REST API 文档的 Camel 组件，或者 org.apache.camel.spi.RestApiProcessorFactory 在 registry 中注册，则 Camel 将查找。如果找到其中任一个，则会使用它。		字符串
camel.rest.api-context-path	设置 REST API 服务将使用的前导 API 上下文路径。这可用于使用 camel-servlet 等组件，其中部署的 Web 应用程序使用 context-path 部署。		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），可用于覆盖使用此配置的主机名生成的主机。		字符串
camel.rest.api-property	允许为 api 文档(swagger)配置多个附加属性。例如，将属性 api.title 设置为 mycolphone。		Map
camel.rest.api-vendor-extension	在 Rest API 中是否启用了厂商扩展。如果启用，则 Camel 将包含额外信息作为供应商扩展（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 标头是否受其消耗的/生成的设置的 Rest-DSL 配置支持。这可以打开它，以启用此检查。如果验证错误，则返回 HTTP 状态代码 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 将查找，如果有一个与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestConsumerFactory 在 registry 中注册。如果找到其中任一个，则会使用它。		字符串
camel.rest.component-property	允许为使用中的其他组件配置多个额外的属性。		Map
camel.rest.consumer-property	允许为使用中的其他使用者配置多个额外的属性。		Map
camel.rest.context-path	设置 REST 服务将使用的前导上下文路径。这可用于使用 camel-servlet 等组件，其中部署的 Web 应用程序使用 context-path 部署。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的属性。例如，将属性 prettyPrint 设置为 true，使 json 在用户模式中输出。属性可以加上前缀，以表示选项仅适用于 JSON 或 XML，对于 IN 或 OUT。前缀为： json.in. json.out. xml.in. xml.out。例如，值为 xml.out.mustBeHQBElement 的键仅适用于传出的 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 89 章 JSONPATH

Camel 支持 JSONPath，以允许对 JSON 消息使用 Expression 或 Predicate。

89.1. 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**	是否允许使用简单 predicate 解析器来预解析 predicates。
writeAsString		**布尔值**	是否将每行/元素的输出写为 JSON 字符串值，而不是 Map/POJO 值。
headerName		**字符串**	作为输入的标头名称，而不是消息的正文。
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。

89.2. 例子

例如，您可以在 Predicate 中使用 JSONPath 和 Content Based Router EIP。

```
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>
```

89.3. JSONPATH SYNTAX

使用 JSONPath 语法需要一些时间来学习，即使用于基本的 predicates。例如，查找您必须做的所有开销：

```
$.store.book[?(@.price < 20)]
```

89.3.1. 简单 JSONPath 语法

但是，如果您只需将其写为：

```
store.book.price < 20
```

如果您只想查看带有价格键的节点，可以省略路径：

```
price < 20
```

要支持此功能，如果您已使用基本风格定义了 predicate，则 EasyPredicateParser，它会 kicks-in。这意味着 predicate 不得以 $ 符号开头，且只包含一个 operator。

简单的语法是：

```
left OP right
```

您可以在正确的运算符中使用 Camel 简单语言，例如：

```
store.book.price < ${header.limit}
```

有关更多语法示例，请参阅 JSONPath 项目页面。

89.4. 支持的消息正文类型

Camel JSonPath 支持以下消息正文：

类型	注释
File	从文件中读取
字符串	纯文本字符串
Map	消息正文为 **java.util.Map** 类型
list	消息正文为 **java.util.List** 类型
POJO	**可选**，如果 Jackson 位于 classpath 上，则 camel-jsonpath 可以使用 Jackson 读取消息正文作为 POJO，并转换为 **java.util.Map**，这由 JSonPath 支持。例如，您可以添加 **camel-jackson** 作为依赖项，使其包含 Jackson。
InputStream	如果没有上述类型匹配，则 Camel 将尝试将消息正文读取为 **java.io.InputStream**。

如果消息正文是不受支持的类型，则默认抛出异常，但您可以配置 JSonPath 来阻止异常（请参阅以下）

89.5. 禁止异常

默认情况下，如果 json 有效负载没有与配置的 jsonpath 表达式的有效路径，jsonpath 将抛出异常。在某些情况下，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 prevent Exceptions 选项设置为 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>
```

此选项也可用于 @JsonPath 注释。

89.6. 内联简单表达式

可以使用 简单 语法 ${xxx} 在 JSONPath 表达式中内联简单语言。

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>
```

89.7. JSONPATH 注入

您可以使用 Bean 集成 在 bean 上调用方法，并使用 JSONPath （通过 @JsonPath 注解）从消息中提取值并将其绑定到方法参数，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
    }
}
```

89.8. 编码检测

如果文档以 unicode (UTF-8, UTF-16LE, UTF-16BE, UTF-32LE, UTF-32BE)进行编码，则会自动检测到 JSON 文档的编码。如果编码是一个非unicode 编码，您可以确保以 String 格式输入文档到 JSONPath，或者在标头 CamelJsonPathJsonEncoding 中指定编码，该编码被定义为常数：JsonpathConstants.HEADER_JSON_ENCODING。

89.9. 将 JSON 数据拆分为 JSON 的子行

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，每个书都会记录，但消息正文是一个 Map 实例。有时，您可能想要将其输出为普通字符串 JSON 值，该值可使用 writeAsString 选项完成，如下所示：

```
from("direct:start")
    .split().jsonpathWriteAsString("$.store.book[*]")
    .to("log:book");
```

然后，每个书都会作为 String JSON 值记录。

89.10. 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 选项来将标头用作输入。

例如，要计算存储在名为 scalability 的标头中的 JSON 文档中的过期 数：

```
from("direct:start")
    .setHeader("numberOfBooks")
        .jsonpath("$..store.book.length()", false, int.class, "books")
    .to("mock:result");
```

在上面的 jsonpath 表达式中，我们将标头的 名称指定为 分号，并告知我们希望将结果转换为 int.class 的整数。

XML DSL 中的同一示例为：

```
<route>
  <from uri="direct:start"/>
  <setHeader name="numberOfBooks">
    <jsonpath headerName="books" resultType="int">$..store.book.length()</jsonpath>
  </setHeader>
  <to uri="mock:result"/>
</route>
```

89.11. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	是否允许使用简单 predicate 解析器来预解析 predicates。	true	布尔值
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

Name	描述	默认值	类型
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/元素的输出写为 JSON 字符串值，而不是 Map/POJO 值。	false	布尔值

第 90 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。

这是 XML DSL 中特定用途。

90.1. REF LANGUAGE OPTIONS

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。

90.2. 用法示例

XML DSL 中的 Splitter EIP 可以使用 < ref > 来利用自定义表达式，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。

以及使用 Java DSL 的同一示例：

```
from("seda:a").split().ref("myExpression").to("seda:b");
```

90.3. 依赖项

Ref 语言是 camel-core 的一部分。

90.4. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	设置与 Consul 搭配使用的 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 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服务调用实施。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	为 OkHttpClient 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服务调用实施。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	所需的服务的传输协议。	_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服务调用实施。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	要设置操作完成的最长时间。		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	在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	在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	设置在使用客户端查找时从文件载入的证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-data	在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	设置在使用客户端查找时从文件载入的客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-algo	设置客户端密钥存储算法，如在使用客户端查找时的 RSA。		字符串
camel.cloud.kubernetes.service-discovery.client-key-data	在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	设置在使用客户端查找时从文件载入的客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-passphrase	在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 查询的 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、环境。在使用客户端时，客户端会查询 kubernetes master 以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，服务名称通过 SRV 查询解析 ....svc... when 使用环境变量来查找该服务。默认使用默认环境。	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，设置 OAUTH 令牌以进行身份验证（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服务调用实施。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服务调用实施。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 maximumSize 生效。然后该值可以等于 coreSize 或更高。	false	布尔值
camel.hystrix.circuit-breaker-enabled	是否使用 HystrixCircuitBreaker。如果为 false，则使用断路器逻辑以及允许的所有请求。这与 hardBreakerForceClosed （）的影响类似，但继续跟踪指标并了解它是否是打开/披露，此属性甚至不实例化断路器。	true	布尔值
camel.hystrix.circuit-breaker-error-threshold-percentage	一个错误百分比阈值（如 50），其中断路器将打开和拒绝请求。它将持续等待在断路器 BreakerSleepWindowInMilliseconds 中定义的持续时间；这与 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，HystrixCircuitBreaker114allowRequest （）始终返回 true 以允许请求，而不考虑 HystrixCommandMetrics.getHealthCounts （）的错误百分比。paraBreakerForceOpen （）属性具有优先权，因此如果设为 true，则它将不做任何操作。	false	布尔值
camel.hystrix.circuit-breaker-force-open	如果为 true，HystrixCircuitBreaker.allowRequest （）将始终返回 false，从而导致断路器打开（往返）并拒绝所有请求。This property takes precedence over circuitBreakerForceClosed();.	false	布尔值
camel.hystrix.circuit-breaker-request-volume-threshold	在 HystrixCircuitBreaker 之前必须存在 metricsRollingStatisticalWindowInMilliseconds （）中的最小请求数。如果低于这个数字，则无论错误百分比如何，时钟都不会往返。	20	整数
camel.hystrix.circuit-breaker-sleep-window-in-milliseconds	HystrixCircuitBreaker 往返后的时间（毫秒）打开它应该等待，然后再再次尝试请求。	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.ThreadPoolExecutor114setCorePoolSize (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 executionIsolationStrategy == SEMAPHORE 时才适用。	20	整数
camel.hystrix.execution-isolation-strategy	将执行哪些隔离策略 HystrixCommand.run （）。如果 THREAD，那么它将在单独的线程上执行，并发请求则受 thread-pool 中的线程数量限制。如果 SEMAPHORE，它将在调用线程和并发请求上执行，则由 semaphore 数限制。	THREAD	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（使用将来的站已取消。仅在 executionIsolationStrategy （）== THREAD 时才适用。	true	布尔值
camel.hystrix.execution-timeout-enabled	是否为这个命令启用超时机制。	true	布尔值
camel.hystrix.execution-timeout-in-milliseconds	在一段时间内，命令将超时和停止执行的时间（毫秒）。如果 executionIsolationThreadInterruptOnTimeout == true，命令是线程隔离，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	失败时是否应尝试 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失败，且不会尝试检索回退。	10	整数
camel.hystrix.group-key	设置要使用的组密钥。默认值为 CamelHystrix。	Camel Hystrix	字符串
camel.hystrix.keep-alive-time	keep-alive 时间（以分钟为单位）传递给 ThreadPoolExecutor，或 theepAliveTime (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 - 不会影响实时更改队列大小。为此，请使用 queueSizeRejectionThreshold （）。	-1	整数
camel.hystrix.maximum-size	传递给 ThreadPoolExecutor thePoolSize (int)的最大线程池大小。这是可以在不拒绝 HystrixCommands 的情况下支持的最大并发量。请注意，只有在您设置了 allowMaximumSizeToDivergeFromCoreSize 时，此设置才会生效。	10	整数
camel.hystrix.metrics-health-snapshot-interval-in-milliseconds	在允许健康快照之间等待的时间（毫秒）来计算成功和错误百分比，并影响 HystrixCircuitBreaker.isOpen （）状态。在高容量上，错误百分比的持续计算可能会变得 CPU 密集型，从而控制计算的频率。	500	整数
camel.hystrix.metrics-rolling-percentile-bucket-size	存储在滚动百分比的每个存储桶中的最大值数。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	10	整数
camel.hystrix.metrics-rolling-percentile-enabled	是否应该使用 HystrixRollingPercentile 在 HystrixCommandMetrics 中捕获百分比的指标。	true	布尔值
camel.hystrix.metrics-rolling-percentile-window-buckets	滚动百分比窗口的存储桶数。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	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。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	10000	整数
camel.hystrix.metrics-rolling-statistical-window-buckets	滚动统计信息窗口划分为的 bucket 数量。这会传递给 HystrixCommandMetrics 中的 HystrixRollingNumber。	10	整数
camel.hystrix.metrics-rolling-statistical-window-in-milliseconds	此属性以毫秒为单位设置统计滚动窗口的持续时间。这是为线程池保留的指标的时长。窗口划分为存储桶，按这些递增推出部署。	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一个人为最大大小，即使尚未达到 maxQueueSize，也会发生拒绝的最大值。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 相同的值。	Camel Hystrix	字符串
camel.hystrix.thread-pool-rolling-number-statistical-window-buckets	滚动统计信息窗口划分为的 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 waitDurationInOpenState 通过后，启用自动从 OPEN 转换到 HALF_OPEN 状态。	false	布尔值
camel.resilience4j.circuit-breaker-ref	引用现有的 io.github.resilience4j.circuitbreaker.CircuitBreaker 实例来查找并从 registry 中使用。使用时，不使用任何其他断路器选项。		字符串
camel.resilience4j.config-ref	引用现有的 io.github.resilience4j.circuitbreaker.CircuitBreakerConfig 实例来查找并从 registry 中使用。		字符串
camel.resilience4j.configurations	定义其他配置定义。		Map

Name	描述	默认值	类型
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速率阈值。如果故障率等于或大于 CircuitBreaker 过渡到打开的阈值，并启动短路调用。阈值必须大于 0，且不能超过 100。默认值为 50 个百分比。		æµ®ç‚¹â€¼
camel.resilience4j.minimum-number-of-calls	在 CircuitBreaker 可以计算错误率前，配置所需的最少调用（每个分片窗口周期）。例如，如果 minimumNumberOfCalls 为 10，则必须记录至少 10 个调用，然后才能计算失败率。如果记录了 9 个调用，CircuitBreaker 不会过渡到打开，即使所有 9 个调用都失败。默认 minimumNumberOfCalls 为 100。	100	整数
camel.resilience4j.permitted-number-of-calls-in-half-open-state	当 CircuitBreaker 处于一半时，配置允许的调用数量。大小必须大于 0。默认大小为 10。	10	整数
camel.resilience4j.sliding-window-size	配置 sliding 窗口的大小，用于记录 CircuitBreaker 关闭时调用的结果。slidingWindowSize 配置 sliding 窗口的大小。分片窗口可以是基于计数或基于时间的。如果 slidingWindowType 是 COUNT_BASED，则会记录并聚合最后一个 slidingWindowSize 调用。如果 slidingWindowType 是 TIME_BASED，则会记录并聚合最后一个 slidingWindowSize 秒的调用。slidingWindowSize 必须大于 0。minimumNumberOfCalls 必须大于 0。如果 slidingWindowType 是 COUNT_BASED，则 minimumNumberOfCalls 不能超过 slidingWindowSize。如果 slidingWindowType 是 TIME_BASED，您可以选择您需要的任何内容。默认 slidingWindowSize 为 100。	100	整数
camel.resilience4j.sliding-window-type	配置 sliding 窗口的类型，用于记录 CircuitBreaker 关闭时调用的结果。分片窗口可以是基于计数或基于时间的。如果 slidingWindowType 是 COUNT_BASED，则会记录并聚合最后一个 slidingWindowSize 调用。如果 slidingWindowType 是 TIME_BASED，则会记录并聚合最后一个 slidingWindowSize 秒的调用。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），调用被视为较慢，并提高较慢的调用百分比。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。当调用百分比相等或大于阈值时，CircuitBreaker 会过渡到打开并启动短路调用。阈值必须大于 0，且不能超过 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 在切换到一半打开前应保持打开的时长。默认值为 60 秒。	60	整数
camel.resilience4j.writable-stack-trace-enabled	启用可写堆栈跟踪。当设置为 false 时，Exception.getStack Trace 会返回零长度数组。当断路器处于打开状态时，这可用于减少日志垃圾邮件，因为例外的原因已经已知（断路器是短路的调用）。	true	布尔值
camel.rest.api-component	如果没有明确配置 API 组件，则作为 REST API （如 swagger）的 Camel 组件的名称（如 swagger），如果存在负责服务并生成 REST API 文档的 Camel 组件，或者 org.apache.camel.spi.RestApiProcessorFactory 在 registry 中注册，则 Camel 将查找。如果找到其中任一个，则会使用它。		字符串
camel.rest.api-context-path	设置 REST API 服务将使用的前导 API 上下文路径。这可用于使用 camel-servlet 等组件，其中部署的 Web 应用程序使用 context-path 部署。		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），可用于覆盖使用此配置的主机名生成的主机。		字符串
camel.rest.api-property	允许为 api 文档(swagger)配置多个附加属性。例如，将属性 api.title 设置为 mycolphone。		Map
camel.rest.api-vendor-extension	在 Rest API 中是否启用了厂商扩展。如果启用，则 Camel 将包含额外信息作为供应商扩展（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 标头是否受其消耗的/生成的设置的 Rest-DSL 配置支持。这可以打开它，以启用此检查。如果验证错误，则返回 HTTP 状态代码 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 将查找，如果有一个与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestConsumerFactory 在 registry 中注册。如果找到其中任一个，则会使用它。		字符串
camel.rest.component-property	允许为使用中的其他组件配置多个额外的属性。		Map
camel.rest.consumer-property	允许为使用中的其他使用者配置多个额外的属性。		Map
camel.rest.context-path	设置 REST 服务将使用的前导上下文路径。这可用于使用 camel-servlet 等组件，其中部署的 Web 应用程序使用 context-path 部署。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的属性。例如，将属性 prettyPrint 设置为 true，使 json 在用户模式中输出。属性可以加上前缀，以表示选项仅适用于 JSON 或 XML，对于 IN 或 OUT。前缀为： json.in. json.out. xml.in. xml.out。例如，值为 xml.out.mustBeHQBElement 的键仅适用于传出的 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 91 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。

91.1. 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。

91.2. 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

变量	类型	描述
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**。

91.3. 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>
```

91.3.1. 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。

91.4. 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>
```

91.5. 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)
```

91.6. 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

91.7. 依赖项

要在 camel 路由中使用 XQuery，您需要添加实施 XQuery 语言的 camel-saxon 的依赖关系。

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

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-saxon</artifactId>
  <version>{CamelSBVersion}</version>
</dependency>
```

91.8. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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

Name	描述	默认值	类型
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
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

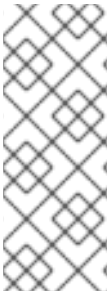
第 92 章 SIMPLE（简单）

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：

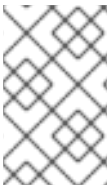
- Groovy
- MVEL
- OGNL



注意

如果简单语言使用 OGNL 表达式，如调用名为 myMethod 的方法在消息正文上调用名为 myMethod 的方法，则简单的语言需要将 camel-bean JAR 作为类路径依赖项：${body.myMethod（）}。在运行时，简单语言将为我们提供需要 camel-bean 组件的内置 OGNL 支持。

简单的语言将 ${body} 占位符用于复杂的表达式或功能。



注意

另请参阅编译的 CSimple 语言。

注意

替代语法
您还可以使用替代语法，其使用 $simple{ } 作为占位符。这可用于避免在将 Spring 属性占位符与 Camel 搭配使用时避免冲突。

92.1. 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。

92.2. 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	使用 Camel OGNL 表达式调用的 Exchange。
exchangeId	字符串	交换 ID
id	字符串	消息 ID
messageTimestamp	字符串	此消息来自的消息时间戳（自 epoc 起）。有些系统，如 JMS, Kafka, AWS 在 Camel 接收的事件/消息上有一个时间戳。如果存在时间戳，此方法会返回时间戳。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。只有在消费者可以从源事件中提取时间戳时，消息时间戳仅在某些 Camel 组件中可用。如果消息没有时间戳，则返回 0。
正文（body）	对象	正文
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 classname 决定的给定类型。转换的正文可以是 null。
bodyAs(*type*).**OGNL**	对象	将正文转换为由 classname 决定的给定类型，然后使用 Camel OGNL 表达式调用方法。转换的正文可以是 null。

变量	类型	描述
bodyOneLine	字符串	将正文转换为 String，并删除所有 line-breaks，以便字符串在一行内。
mandatoryBodyAs(*type*)	类型	将正文转换为由 classname 决定的给定类型，并且希望正文不是 null。
mandatoryBodyAs(*type*).**OGNL**	对象	将正文转换为由 classname 决定的给定类型，然后使用 Camel OGNL 表达式调用方法。
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	将 foo 标头视为映射，并使用 bar 作为键在映射上执行查询
header.foo.**OGNL**	对象	请参阅 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 classname 确定的给定类型
标头	Map	请参阅标头
exchangeProperty.foo	对象	请参阅交换中的 foo 属性
exchangeProperty[foo]	对象	请参阅交换中的 foo 属性
exchangeProperty.foo.**OGNL**	对象	请参阅交换上的 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对象，请参阅交换上的异常对象为 **null**。如果交换有任何，将回退和获取例外 (**Exchange.EXCEPTION_CAUGHT**)。
exception.**OGNL**	对象	请参阅使用 Camel OGNL 表达式对象调用的交换异常

变量	类型	描述
exception.message	字符串	如果没有在交换上设置异常，请参阅交换上的 exception.message。**如果交换有任何，将回退和获取例外 (Exchange.EXCEPTION_CAUGHT)。**
exception.stacktrace	字符串	如果没有对交换设置了异常，请参阅交换上的 exception.stracktrace。**如果交换有任何，将回退和获取例外 (Exchange.EXCEPTION_CAUGHT)。**
date:_command_	Date	评估日期对象。支持的命令现在为当前时间戳，在创建当前交换时 **exchangeCreated** 用于时间戳，**header.xxx** 使用带有键 xxx 的 Long/Date 对象标头。**exchangeProperty.xxx 使用 exchange 属性中的 Long/Date 对象，其键为 xxx。文件 最后一次修改的时间戳的文件（可供文件消费者使用）。命令接受偏移量，如 -24h 或 header.xxx+1h 或 now+1h30m-100。**
date:_command:pattern_	字符串	使用 **java.text.SimpleDateFormat** 模式进行日期格式。
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。
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。我们还支持组件所使用的 ?method=methodname 语法。Camel 默认将按指定名称查找 bean。但是，如果您需要引用 bean 类（如调用静态方法），您可以使用 type 前缀，如 **bean:type:fqnClassName**。
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	返回交换正在路由的当前路由的 id。
stepId	字符串	返回交换正在路由的当前步骤的 id。
threadName	字符串	返回当前线程的名称。可用于日志记录目的。
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 Registry 中查找给定 ID 的 bean：
type:name.field	对象	按其 FQN 名称引用类型或字段。要引用字段，您可以附加 .FIELD_NAME。例如，您可以将 Exchange 中的常量字段称为：**org.apache.camel.Exchange.FILE_NAME**
null	null	代表 **null**

变量	类型	描述
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* （包括）和最大（不包括）之间返回一个随机整数
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。这可以与 Splitter EIP 一起使用，将消息正文和组/批处理分割子消息到一组 N 子列表。这个方法的工作方式与 Groovy 中的协作方法类似。
skip(number)	iterator	skip 功能迭代消息正文并跳过第一个项目数量。这可以与 Splitter EIP 一起使用，以分割消息正文并跳过第 N 个项目的数量。
messageHistory	字符串	当前交换是如何路由的消息历史记录。这与路由 stack-trace 消息在未处理的异常时记录错误处理程序日志。
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，可以使用此选项。

92.3. OGNL 表达式支持

使用 OGNL 时，需要 camel-bean JAR 在 classpath 上。

Camel 的 OGNL 支持仅用于调用方法。您无法访问字段。Camel 支持访问 Java 阵列的长度字段。

Simple 和 Bean 语言现在支持 Camel OGNL 表示法，用来以类似方式的链调用 Bean。假设 Message IN 正文包含一个 POJO，它具有 getAddress （） 方法。

然后，您可以使用 Camel OGNL 表示法访问地址对象：

```
simple("${body.address}")
simple("${body.address.street}")
simple("${body.address.zip}")
```

Camel 了解 getters 的短名称，但您可以调用任何方法或使用实际名称，例如：

```
simple("${body.address}")
```

```
simple("${body.getAddress.getStreet}")
simple("${body.address.getZip}")
simple("${body.doSomething}")
```

如果正文没有地址，您还可以使用 null 安全运算符(?.)来避免 NPE

```
simple("${body?.address?.street}")
```

也可以在 Map 或 List 类型中索引，以便您可以：

```
simple("${body[foo].name}")
```

假设正文基于 映射，并使用 foo 作为键查找值，并在该值上调用 getName 方法。

如果键有空格，则必须 使用引号括起键，例如 'foo bar'：

```
simple("${body['foo bar'].name}")
```

您可以使用其密钥名称（带有或不带点）直接访问 Map 或 List 对象：

```
simple("${body[foo]}")
simple("${body[this.is.foo]}")
```

假设键 foo 没有值，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，例如从您可以做的地址获取行：

```
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，可用于从列表获取最后一个值。

```
simple("${body.address.lines[last]}")
```

要获得第二个值，您可以减去一个数字，因此我们可以使用 last-1 来指示这一点：

```
simple("${body.address.lines[last-1]}")
```

最后三部分就是：

```
simple("${body.address.lines[last-2]}")
```

您可以使用方法调用列表的大小方法

```
simple("${body.address.lines.size}")
```

Camel 还支持 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")
```

另外，您可以将此功能与 Operator 支持合并，如下所示：

```
simple("${body.address.zip} > 1000")
```

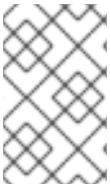
92.4. OPERATOR 支持

解析器仅限于只支持单个 Operator。

要启用它的值必须用 $\{ } 括起来。语法为：

```
${leftValue} OP rightValue
```

其中 rightValue 可以是字符串字面，用 ' '、null、一个常量值或其他表达式括起在 $\{ } 中。



注意

Operator 必须具有 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
=~	等于忽略问题单（在比较字符串值时忽略问题单）
>	大于
>=	大于或等于或等
<	小于
⇐	小于或等于不相等
!=	不相等
!=~	不等于忽略问题单（在比较字符串值时忽略问题单）
contains	在基于字符串的值中包含测试
!tains	在基于字符串的值中没有包括测试
~~	在基于字符串的值中忽略问题单敏感度，用于测试
!~~	在基于字符串的值中忽略问题单敏感度，用于测试
regex	对于定义为字符串值的给定正则表达式模式匹配
!regex	与定义为字符串值的给定正则表达式模式匹配
in	若要在一组值中匹配，每个元素必须以逗号分开。如果要包含空值，则必须使用双逗号（如 ',bronze,silver,gold' ）定义它，这是一组带有空值的四个值，然后是三 medals。

Operator	描述
!in	要匹配（如果没有在一组值中），则必须使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，这是一组带有空值的四个值，然后是三 medals。
is	如果左侧类型是值的实例，则匹配。
!is	如果左侧类型不是值的实例，则匹配。
range	对于匹配，如果左侧的值位于定义为 number: **from..to**. 的一系列值中。
!range	如果左侧的值不在定义为数字的一系列值中，则匹配： **from..to** .。
startsWith	为了测试左侧字符串是否以右手动字符串开头。
从开始	与 startWith operator 相同。
endsWith	用于测试左侧字符串是否以右手动字符串结尾。
结束	与 endWith 运算符相同。

以下 unary operator 可用于：

Operator	描述
++	将数字递增 1。左侧必须是函数，否则被解析为字面。
–	将数字减一。左侧必须是函数，否则被解析为字面。
\n	使用换行符：
\t	使用标签字符：
\r	使用 carriage 返回字符。
\}	使用 } 字符作为文本。这在使用简单语言构建 JSon 结构时可能需要这样做。

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 或 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")
```

92.4.1. 与不同类型的比较

当您将不同的类型（如 String 和 int ）进行比较时，您必须小心。Camel 将把左侧的类型用作第 1 个优先级。如果两个值都不能根据该类型进行比较，则回退到右侧类型。
这意味着您可以修改值来强制实施特定类型的。假设上面的 bar 值是一个 String。然后您可以模糊处理：

```
simple("100 < ${header.bar}")
```

然后，这样可确保 int 类型用作 1st 优先级。

如果 Camel 团队改进了二进制比较操作，以首选使用数字类型到字符串，这可能会在以后更改。通常是字符串类型，与数字进行比较时会导致问题。

```
// testing for null
simple("${header.baz} == null")

// testing for not null
simple("${header.baz} != null")
```

另外，一个更高级的示例，其中正确的值是另一个表达式

```
simple("${header.date} == ${date:now:yyyyMMdd}")

simple("${header.type} == ${bean:orderService?method=getOrderType}")
```

以及一个包含的示例，测试标题是否包含标题 Camel

```
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，如果标头等于列表中的任何值。每个元素必须以逗号分开，且没有空格。这也适用于数字等，因为 Camel 会将每个元素转换为左侧的类型。

```
simple("${header.type} in 'gold,silver'")
```

对于所有最后 3 个，我们也支持 negate 测试。

```
simple("${header.type} !in 'gold,silver'")
```

您可以测试类型是否为一个特定的实例，例如，一个字符串

```
simple("${header.type} is 'java.lang.String'")
```

我们为所有 java.lang 类型添加了一个简短的缩写，以便您可以将其编写为：

```
simple("${header.type} is 'String'")
```

也支持范围。范围间隔需要数字和结尾。例如，要测试值是否在 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'")
```

由于 XML DSL 没有作为 Java DSL 的所有功能，所以您必须根据所有不同的构建器方法进行测试，所以您必须使用其他语言对简单操作器进行测试。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>
```

92.4.2. 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'")
```

在课程中，还支持 ||。示例为：

```
simple("${header.title} contains 'Camel' || ${header.type'} == 'gold'")
```

92.5. 例子

在以下 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 Message Filter 模式上面的 predicate 测试，其中测试消息是否有 foo 标头（存在带有键 foo 的标头）。如果表达式评估为 true，则消息将路由到 mock:fooOrders 端点，否则将丢弃消息。

Java DSL 中的同一示例：

```
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，我们现在必须在表达式中使用 $\\{ } 占位符来允许 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 语言在 bean 上调用方法，以包含在返回的字符串中：

```
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 generator bean 上调用哪个方法，我们必须加上 .method 名称，例如在调用 generateId 方法的地方。

```
from("direct:order")
    .transform().simple("OrderId: ${bean:orderIdGenerator.generateId}")
    .to("mock:reply");
```

我们可以使用 ?method=methodname 选项熟悉 Bean 组件本身：

```
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>
```

有几种类型具有简短表示法，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，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，然后使用键 金级 查找并返回值。如果标头无法转换为 Map，则会抛出异常。如果名称 类型为 的标头不存在 null。

您可以嵌套功能，如下所示：

```
<setHeader name="myHeader">
  <simple>${properties:${header.someKey}}</simple>
</setHeader>
```

92.6. 设置结果类型

现在，您可以向 Simple 表达式提供一个结果类型，这意味着评估的结果将转换为所需的类型。这最可用于定义布尔值、整数等类型。

例如，要将标头设置为布尔值类型，您可以执行以下操作：

```
.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>
```

92.7. 使用 XML DSL 中的新行或标签页

现在，您可以在 XML DSL 中指定新行或标签页，因为您可以现在转义值

```
<transform>
  <simple>The following text\nis on a new line</simple>
</transform>
```

92.8. 领导和尾随空格处理

表达式的修剪属性可用于控制是否删除前导和尾随空格字符还是保留。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>
```

92.9. 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")
```

92.10. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	设置与 Consul 搭配使用的 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 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服务调用实施。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	为 OkHttpClient 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服务调用实施。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	所需的服务的传输协议。	_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服务调用实施。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	要设置操作完成的最长时间。		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	在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	在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	设置在使用客户端查找时从文件载入的证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-data	在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	设置在使用客户端查找时从文件载入的客户端证书数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-algo	设置客户端密钥存储算法，如在使用客户端查找时的 RSA。		字符串
camel.cloud.kubernetes.service-discovery.client-key-data	在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	设置在使用客户端查找时从文件载入的客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-passphrase	在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 查询的 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、环境。在使用客户端时，客户端会查询 kubernetes master 以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，服务名称通过 SRV 查询解析 ....svc... when 使用环境变量来查找该服务。默认使用默认环境。	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，设置 OAUTH 令牌以进行身份验证（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服务调用实施。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服务调用实施。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 maximumSize 生效。然后该值可以等于 coreSize 或更高。	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，则使用断路器逻辑以及允许的所有请求。这与 hardBreakerForceClosed （）的影响类似，但继续跟踪指标并了解它是否是打开/披露，此属性甚至不实例化断路器。	true	布尔值
camel.hystrix.circuit-breaker-error-threshold-percentage	一个错误百分比阈值（如 50），其中断路器将打开和拒绝请求。它将持续等待在断路器 BreakerSleepWindowInMilliseconds 中定义的持续时间；这与 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，HystrixCircuitBreaker114allowRequest （）始终返回 true 以允许请求，而不考虑 HystrixCommandMetrics.getHealthCounts （）的错误百分比。paraBreakerForceOpen （）属性具有优先权，因此如果设为 true，则它将不做任何操作。	false	布尔值
camel.hystrix.circuit-breaker-force-open	如果为 true，HystrixCircuitBreaker.allowRequest （）将始终返回 false，从而导致断路器打开（往返）并拒绝所有请求。This property takes precedence over circuitBreakerForceClosed();.	false	布尔值
camel.hystrix.circuit-breaker-request-volume-threshold	在 HystrixCircuitBreaker 之前必须存在 metricsRollingStatisticalWindowInMilliseconds （）中的最小请求数。如果低于这个数字，则无论错误百分比如何，时钟都不会往返。	20	整数
camel.hystrix.circuit-breaker-sleep-window-in-milliseconds	HystrixCircuitBreaker 往返后的时间（毫秒）打开它应该等待，然后再再次尝试请求。	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.ThreadPoolExecutor114setCorePoolSize (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 executionIsolationStrategy == SEMAPHORE 时才适用。	20	整数

Name	描述	默认值	类型
camel.hystrix.execution-isolation-strategy	将执行哪些隔离策略 HystrixCommand.run （）。如果 THREAD，那么它将在单独的线程上执行，并发请求则受 thread-pool 中的线程数量限制。如果 SEMAPHORE，它将在调用线程和并发请求上执行，则由 semaphore 数限制。	THREAD	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（使用将来的站已取消。仅在 executionIsolationStrategy （）== THREAD 时才适用。	true	布尔值
camel.hystrix.execution-timeout-enabled	是否为这个命令启用超时机制。	true	布尔值
camel.hystrix.execution-timeout-in-milliseconds	在一段时间内，命令将超时和停止执行的时间（毫秒）。如果 executionIsolationThreadInterruptOnTimeout == true，命令是线程隔离，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	失败时是否应尝试 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失败，且不会尝试检索回退。	10	整数
camel.hystrix.group-key	设置要使用的组密钥。默认值为 CamelHystrix。	Camel Hystrix	字符串
camel.hystrix.keep-alive-time	keep-alive 时间（以分钟为单位）传递给 ThreadPoolExecutor，或 theepAliveTime (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 - 不会影响实时更改队列大小。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 thePoolSize (int)的最大线程池大小。这是可以在不拒绝 HystrixCommands 的情况下支持的最大并发量。请注意，只有在您设置了 allowMaximumSizeToDivergeFromCoreSize 时，此设置才会生效。	10	整数
camel.hystrix.metrics-health-snapshot-interval-in-milliseconds	在允许健康快照之间等待的时间（毫秒）来计算成功和错误百分比，并影响 HystrixCircuitBreaker.isOpen（）状态。在高容量上，错误百分比的持续计算可能会变得 CPU 密集型，从而控制计算的频率。	500	整数
camel.hystrix.metrics-rolling-percentile-bucket-size	存储在滚动百分比的每个存储桶中的最大值数。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	10	整数
camel.hystrix.metrics-rolling-percentile-enabled	是否应该使用 HystrixRollingPercentile 在 HystrixCommandMetrics 中捕获百分比的指标。	true	布尔值
camel.hystrix.metrics-rolling-percentile-window-buckets	滚动百分比窗口的存储桶数。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	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。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	10000	整数
camel.hystrix.metrics-rolling-statistical-window-buckets	滚动统计信息窗口划分为的 bucket 数量。这会传递给 HystrixCommandMetrics 中的 HystrixRollingNumber。	10	整数
camel.hystrix.metrics-rolling-statistical-window-in-milliseconds	此属性以毫秒为单位设置统计滚动窗口的持续时间。这是为线程池保留的指标的时长。窗口划分为存储桶，按这些递增推出部署。	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一个人为最大大小，即使尚未达到 maxQueueSize，也会发生拒绝的最大值。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 相同的值。	Camel Hystrix	字符串
camel.hystrix.thread-pool-rolling-number-statistical-window-buckets	滚动统计信息窗口划分为的 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 waitDurationInOpenState 通过后，启用自动从 OPEN 转换到 HALF_OPEN 状态。	false	布尔值
camel.resilience4j.circuit-breaker-ref	引用现有的 io.github.resilience4j.circuitbreaker.CircuitBreaker 实例来查找并从 registry 中使用。使用时，不使用任何其他断路器选项。		字符串
camel.resilience4j.config-ref	引用现有的 io.github.resilience4j.circuitbreaker.CircuitBreakerConfig 实例来查找并从 registry 中使用。		字符串
camel.resilience4j.configurations	定义其他配置定义。		Map

Name	描述	默认值	类型
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速率阈值。如果故障率等于或大于 CircuitBreaker 过渡到打开的阈值，并启动短路调用。阈值必须大于 0，且不能超过 100。默认值为 50 个百分比。		æµ®ç‚¹â€¼
camel.resilience4j.minimum-number-of-calls	在 CircuitBreaker 可以计算错误率前，配置所需的最少调用（每个分片窗口周期）。例如，如果 minimumNumberOfCalls 为 10，则必须记录至少 10 个调用，然后才能计算失败率。如果记录了 9 个调用，CircuitBreaker 不会过渡到打开，即使所有 9 个调用都失败。默认 minimumNumberOfCalls 为 100。	100	整数
camel.resilience4j.permitted-number-of-calls-in-half-open-state	当 CircuitBreaker 处于一半时，配置允许的调用数量。大小必须大于 0。默认大小为 10。	10	整数
camel.resilience4j.sliding-window-size	配置 sliding 窗口的大小，用于记录 CircuitBreaker 关闭时调用的结果。slidingWindowSize 配置 sliding 窗口的大小。分片窗口可以是基于计数或基于时间的。如果 slidingWindowType 是 COUNT_BASED，则会记录并聚合最后一个 slidingWindowSize 调用。如果 slidingWindowType 是 TIME_BASED，则会记录并聚合最后一个 slidingWindowSize 秒的调用。slidingWindowSize 必须大于 0。minimumNumberOfCalls 必须大于 0。如果 slidingWindowType 是 COUNT_BASED，则 minimumNumberOfCalls 不能超过 slidingWindowSize。如果 slidingWindowType 是 TIME_BASED，您可以选择您需要的任何内容。默认 slidingWindowSize 为 100。	100	整数
camel.resilience4j.sliding-window-type	配置 sliding 窗口的类型，用于记录 CircuitBreaker 关闭时调用的结果。分片窗口可以是基于计数或基于时间的。如果 slidingWindowType 是 COUNT_BASED，则会记录并聚合最后一个 slidingWindowSize 调用。如果 slidingWindowType 是 TIME_BASED，则会记录并聚合最后一个 slidingWindowSize 秒的调用。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），调用被视为较慢，并提高较慢的调用百分比。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。当调用百分比相等或大于阈值时，CircuitBreaker 会过渡到打开并启动短路调用。阈值必须大于 0，且不能超过 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 在切换到一半打开前应保持打开的时长。默认值为 60 秒。	60	整数
camel.resilience4j.writable-stack-trace-enabled	启用可写堆栈跟踪。当设置为 false 时，Exception.getStack Trace 会返回零长度数组。当断路器处于打开状态时，这可用于减少日志垃圾邮件，因为例外的原因已经已知（断路器是短路的调用）。	true	布尔值
camel.rest.api-component	如果没有明确配置 API 组件，则作为 REST API （如 swagger）的 Camel 组件的名称（如 swagger），如果存在负责服务并生成 REST API 文档的 Camel 组件，或者 org.apache.camel.spi.RestApiProcessorFactory 在 registry 中注册，则 Camel 将查找。如果找到其中任一个，则会使用它。		字符串
camel.rest.api-context-path	设置 REST API 服务将使用的前导 API 上下文路径。这可用于使用 camel-servlet 等组件，其中部署的 Web 应用程序使用 context-path 部署。		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），可用于覆盖使用此配置的主机名生成的主机。		字符串
camel.rest.api-property	允许为 api 文档(swagger)配置多个附加属性。例如，将属性 api.title 设置为 mycolphone。		Map
camel.rest.api-vendor-extension	在 Rest API 中是否启用了厂商扩展。如果启用，则 Camel 将包含额外信息作为供应商扩展（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 标头是否受其消耗的/生成的设置的 Rest-DSL 配置支持。这可以打开它，以启用此检查。如果验证错误，则返回 HTTP 状态代码 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 将查找，如果有一个与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestConsumerFactory 在 registry 中注册。如果找到其中任一个，则会使用它。		字符串
camel.rest.component-property	允许为使用中的其他组件配置多个额外的属性。		Map
camel.rest.consumer-property	允许为使用中的其他使用者配置多个额外的属性。		Map
camel.rest.context-path	设置 REST 服务将使用的前导上下文路径。这可用于使用 camel-servlet 等组件，其中部署的 Web 应用程序使用 context-path 部署。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的属性。例如，将属性 prettyPrint 设置为 true，使 json 在用户模式中输出。属性可以加上前缀，以表示选项仅适用于 JSON 或 XML，对于 IN 或 OUT。前缀为： json.in. json.out. xml.in. xml.out。例如，值为 xml.out.mustBeHQBElement 的键仅适用于传出的 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 93 章 令牌化

令牌程序语言是 camel-core 中的内置语言，它最常与 Split EIP 一起使用，以使用基于令牌的策略分割消息。

令牌化语言旨在使用指定的分隔符模式对文本文档进行令牌化。它还可用于为 XML 文档提供一些有限功能。对于真实的 XML 感知令牌化，建议使用 XML 令牌化 语言，因为它提供了更快速、更有效的令牌化，专门用于 XML 文档。

93.1. 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 将令牌用作令牌化器，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 令牌对，则用作令牌的最终用户。您可以使用简单语言作为令牌来支持动态令牌。
inheritNamespaceTagName		字符串	要在使用 XML 时从 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	要令牌化的标头名称，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，则必须将这个选项设置为 true。
includeTokens		布尔值	在使用对时，是否在部分中包含令牌，其默认值为 false。
group		字符串	要将 N 个部分分组在一起，例如将大文件分成 1000 行的块。您可以使用简单语言作为组来支持动态组群大小。
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	要跳过最先的元素。
trim		布尔值	是否修剪值以移除前导和结尾的空格和换行符。

93.2. 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 将其分成部分，然后将每个请求转发到 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");
```

93.3. 另请参阅

有关更多示例，请参阅 Split EIP。

93.4. SPRING BOOT AUTO-CONFIGURATION

当在 Spring Boot 中使用令牌化时，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>
```

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	设置与 Consul 搭配使用的 ACL 令牌。		字符串

Name	描述	默认值	类型
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 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服务调用实施。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

Name	描述	默认值	类型
camel.cloud.consul.service-discovery.write-timeout-millis	为 OkHttpClient 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服务调用实施。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	所需的服务的传输协议。	_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服务调用实施。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.etcd.service-discovery.service-path	查找服务发现的路径。	/services/	字符串
camel.cloud.etcd.service-discovery.timeout	要设置操作完成的最长时间。		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	在使用客户端查找时设置 API 版本。		字符串
camel.cloud.kubernetes.service-discovery.ca-cert-data	在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	设置在使用客户端查找时从文件载入的证书颁发机构数据。		字符串
camel.cloud.kubernetes.service-discovery.client-cert-data	在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	设置在使用客户端查找时从文件载入的客户端证书数据。		字符串

Name	描述	默认值	类型
camel.cloud.kubernetes.service-discovery.client-key-algo	设置客户端密钥存储算法，如在使用客户端查找时的 RSA。		字符串
camel.cloud.kubernetes.service-discovery.client-key-data	在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	设置在使用客户端查找时从文件载入的客户端密钥存储数据。		字符串
camel.cloud.kubernetes.service-discovery.client-key-passphrase	在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 查询的 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、环境。在使用客户端时，客户端会查询 kubernetes master 以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，服务名称通过 SRV 查询解析 ....svc... when 使用环境变量来查找该服务。默认使用默认环境。	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

Name	描述	默认值	类型
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，设置 OAUTH 令牌以进行身份验证（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服务调用实施。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

Name	描述	默认值	类型
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服务调用实施。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 maximumSize 生效。然后该值可以等于 coreSize 或更高。	false	布尔值
camel.hystrix.circuit-breaker-enabled	是否使用 HystrixCircuitBreaker。如果为 false，则使用断路器逻辑以及允许的所有请求。这与 hardBreakerForceClosed （）的影响类似，但继续跟踪指标并了解它是否是打开/披露，此属性甚至不实例化断路器。	true	布尔值
camel.hystrix.circuit-breaker-error-threshold-percentage	一个错误百分比阈值（如 50），其中断路器将打开和拒绝请求。它将持续等待在断路器 BreakerSleepWindowInMilliseconds 中定义的持续时间；这与 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，HystrixCircuitBreaker114allowRequest （）始终返回 true 以允许请求，而不考虑 HystrixCommandMetrics.getHealthCounts （）的错误百分比。paraBreakerForceOpen （）属性具有优先权，因此如果设为 true，则它将不做任何操作。	false	布尔值

Name	描述	默认值	类型
camel.hystrix.circuit-breaker-force-open	如果为 true，HystrixCircuitBreaker.allowRequest（）将始终返回 false，从而导致断路器打开（往返）并拒绝所有请求。This property takes precedence over circuitBreakerForceClosed();.	false	布尔值
camel.hystrix.circuit-breaker-request-volume-threshold	在 HystrixCircuitBreaker 之前必须存在 metricsRollingStatisticalWindowInMilliseconds （）中的最小请求数。如果低于这个数字，则无论错误百分比如何，时钟都不会往返。	20	整数
camel.hystrix.circuit-breaker-sleep-window-in-milliseconds	HystrixCircuitBreaker 往返后的时间（毫秒）打开它应该等待，然后再再次尝试请求。	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.ThreadPoolExecutor114setCorePoolSize (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 executionIsolationStrategy == SEMAPHORE 时才适用。	20	整数
camel.hystrix.execution-isolation-strategy	将执行哪些隔离策略 HystrixCommand.run （）。如果 THREAD，那么它将在单独的线程上执行，并发请求则受 thread-pool 中的线程数量限制。如果 SEMAPHORE，它将在调用线程和并发请求上执行，则由 semaphore 数限制。	THREAD	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（使用将来的站已取消。仅在 executionIsolationStrategy （）== THREAD 时才适用。	true	布尔值
camel.hystrix.execution-timeout-enabled	是否为这个命令启用超时机制。	true	布尔值

Name	描述	默认值	类型
camel.hystrix.execution-timeout-in-milliseconds	在一段时间内，命令将超时和停止执行的时间（毫秒）。如果 executionIsolationThreadInterruptOnTimeout == true，命令是线程隔离，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	失败时是否应尝试 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失败，且不会尝试检索回退。	10	整数
camel.hystrix.group-key	设置要使用的组密钥。默认值为 CamelHystrix。	Camel Hystrix	字符串
camel.hystrix.keep-alive-time	keep-alive 时间（以分钟为单位）传递给 ThreadPoolExecutor，或 theepAliveTime (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 - 不会影响实时更改队列大小。为此，请使用 queueSizeRejectionThreshold （）。	-1	整数
camel.hystrix.maximum-size	传递给 ThreadPoolExecutor thePoolSize (int)的最大线程池大小。这是可以在不拒绝 HystrixCommands 的情况下支持的最大并发量。请注意，只有在您设置了 allowMaximumSizeToDivergeFromCoreSize 时，此设置才会生效。	10	整数
camel.hystrix.metrics-health-snapshot-interval-in-milliseconds	在允许健康快照之间等待的时间（毫秒）来计算成功和错误百分比，并影响 HystrixCircuitBreaker.isOpen （）状态。在高容量上，错误百分比的持续计算可能会变得 CPU 密集型，从而控制计算的频率。	500	整数
camel.hystrix.metrics-rolling-percentile-bucket-size	存储在滚动百分比的每个存储桶中的最大值数。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	10	整数

Name	描述	默认值	类型
camel.hystrix.metrics-rolling-percentile-enabled	是否应该使用 HystrixRollingPercentile 在 HystrixCommandMetrics 中捕获百分比的指标。	true	布尔值
camel.hystrix.metrics-rolling-percentile-window-buckets	滚动百分比窗口的存储桶数。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	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。这被传递到 HystrixCommandMetrics 中的 HystrixRollingPercentile 中。	10000	整数
camel.hystrix.metrics-rolling-statistical-window-buckets	滚动统计信息窗口划分为的 bucket 数量。这会传递给 HystrixCommandMetrics 中的 HystrixRollingNumber。	10	整数
camel.hystrix.metrics-rolling-statistical-window-in-milliseconds	此属性以毫秒为单位设置统计滚动窗口的持续时间。这是为线程池保留的指标的时长。窗口划分为存储桶，按这些递增推出部署。	10000	整数
camel.hystrix.queue-size-rejection-threshold	队列大小拒绝阈值是一个人为最大大小，即使尚未达到 maxQueueSize，也会发生拒绝的最大值。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 相同的值。	Camel Hystrix	字符串
camel.hystrix.thread-pool-rolling-number-statistical-window-buckets	滚动统计信息窗口划分为的 bucket 数量。这会传递到每个 HystrixThreadPoolMetrics 实例内的 HystrixRollingNumber。	10	整数

Name	描述	默认值	类型
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
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

Name	描述	默认值	类型
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 waitDurationInOpenState 通过后，启用自动从 OPEN 转换到 HALF_OPEN 状态。	false	布尔值
camel.resilience4j.circuit-breaker-ref	引用现有的 io.github.resilience4j.circuitbreaker.CircuitBreaker 实例来查找并从 registry 中使用。使用时，不使用任何其他断路器选项。		字符串
camel.resilience4j.config-ref	引用现有的 io.github.resilience4j.circuitbreaker.CircuitBreakerConfig 实例来查找并从 registry 中使用。		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速率阈值。如果故障率等于或大于 CircuitBreaker 过渡到打开的阈值，并启动短路调用。阈值必须大于 0，且不能超过 100。默认值为 50 个百分比。		æµ®ç‚¹å€¼

Name	描述	默认值	类型
camel.resilience4j.minimum-number-of-calls	在 CircuitBreaker 可以计算错误率前，配置所需的最少调用（每个分片窗口周期）。例如，如果 minimumNumberOfCalls 为 10，则必须记录至少 10 个调用，然后才能计算失败率。如果记录了 9 个调用，CircuitBreaker 不会过渡到打开，即使所有 9 个调用都失败。默认 minimumNumberOfCalls 为 100。	100	整数
camel.resilience4j.permitted-number-of-calls-in-half-open-state	当 CircuitBreaker 处于一半时，配置允许的调用数量。大小必须大于 0。默认大小为 10。	10	整数
camel.resilience4j.sliding-window-size	配置 sliding 窗口的大小，用于记录 CircuitBreaker 关闭时调用的结果。slidingWindowSize 配置 sliding 窗口的大小。分片窗口可以是基于计数或基于时间的。如果 slidingWindowType 是 COUNT_BASED，则会记录并聚合最后一个 slidingWindowSize 调用。如果 slidingWindowType 是 TIME_BASED，则会记录并聚合最后一个 slidingWindowSize 秒的调用。slidingWindowSize 必须大于 0。minimumNumberOfCalls 必须大于 0。如果 slidingWindowType 是 COUNT_BASED，则 minimumNumberOfCalls 不能超过 slidingWindowSize。如果 slidingWindowType 是 TIME_BASED，您可以选择您需要的任何内容。默认 slidingWindowSize 为 100。	100	整数
camel.resilience4j.sliding-window-type	配置 sliding 窗口的类型，用于记录 CircuitBreaker 关闭时调用的结果。分片窗口可以是基于计数或基于时间的。如果 slidingWindowType 是 COUNT_BASED，则会记录并聚合最后一个 slidingWindowSize 调用。如果 slidingWindowType 是 TIME_BASED，则会记录并聚合最后一个 slidingWindowSize 秒的调用。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），调用被视为较慢，并提高较慢的调用百分比。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。当调用百分比相等或大于阈值时，CircuitBreaker 会过渡到打开并启动短路调用。阈值必须大于 0，且不能超过 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

Name	描述	默认值	类型
camel.resilience4j.wait-duration-in-open-state	配置等待持续时间（以秒为单位），指定 CircuitBreaker 在切换到一半打开前应保持打开的时长。默认值为 60 秒。	60	整数
camel.resilience4j.writable-stack-trace-enabled	启用可写堆栈跟踪。当设置为 false 时，Exception.getStack Trace 会返回零长度数组。当断路器处于打开状态时，这可用于减少日志垃圾邮件，因为例外的原因已经已知（断路器是短路的调用）。	true	布尔值
camel.rest.api-component	如果没有明确配置 API 组件，则作为 REST API （如 swagger）的 Camel 组件的名称（如 swagger），如果存在负责服务并生成 REST API 文档的 Camel 组件，或者 org.apache.camel.spi.RestApiProcessorFactory 在 registry 中注册，则 Camel 将查找。如果找到其中任一个，则会使用它。		字符串
camel.rest.api-context-path	设置 REST API 服务将使用的前导 API 上下文路径。这可用于使用 camel-servlet 等组件，其中部署的 Web 应用程序使用 context-path 部署。		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），可用于覆盖使用此配置的主机名生成的主机。		字符串
camel.rest.api-property	允许为 api 文档(swagger)配置多个附加属性。例如，将属性 api.title 设置为 mycolphone。		Map
camel.rest.api-vendor-extension	在 Rest API 中是否启用了厂商扩展。如果启用，则 Camel 将包含额外信息作为供应商扩展（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 标头是否受其消耗的/生成的设置的 Rest-DSL 配置支持。这可以打开它，以启用此检查。如果验证错误，则返回 HTTP 状态代码 415 或 406。默认值为 false。	false	布尔值

Name	描述	默认值	类型
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 将查找，如果有一个与 Rest DSL 集成的 Camel 组件，或者 org.apache.camel.spi.RestConsumerFactory 在 registry 中注册。如果找到其中任一个，则会使用它。		字符串
camel.rest.component-property	允许为使用中的其他组件配置多个额外的属性。		Map
camel.rest.consumer-property	允许为使用中的其他使用者配置多个额外的属性。		Map
camel.rest.context-path	设置 REST 服务将使用的前导上下文路径。这可用于使用 camel-servlet 等组件，其中部署的 Web 应用程序使用 context-path 部署。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的属性。例如，将属性 prettyPrint 设置为 true，使 json 在用户模式中输出。属性可以加上前缀，以表示选项仅适用于 JSON 或 XML，对于 IN 或 OUT。前缀为： json.in. json.out. xml.in. xml.out。例如，值为 xml.out.mustBeHQBElement 的键仅适用于传出的 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

Name	描述	默认值	类型
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 94 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 感知令牌程序，可与 Split EIP 作为传统令牌来高效和有效地对 XML 文档进行令牌处理。

XML 令牌化不仅能够识别文档的 XML 命名空间和分层结构，还比传统令牌语言更高效地对 XML 文档进行令牌化。

其他依赖项

要使用此组件，需要额外依赖项，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>
```

94.1. XML 令牌化器选项

XML 令牌化语言支持 4 个选项，如下所列。

Name	默认值	Java 类型	描述
headerName		字符串	要令牌化的标头名称，而不使用消息正文。

Name	默认值	Java 类型	描述
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 - 将提取的令牌嵌套到其祖先上下文 u - 将提取的令牌解压缩到其子内容 t - 提取指定元素的文本内容。 Enum 值： • i • w • u • t
group		**整数**	将 N 个部分分组在一起。
trim		**布尔值**	是否修剪值以移除前导和结尾的空格和换行符。

94.2. 示例

请参阅 Split EIP，它带有使用 XML Tokenize 语言的示例。

94.3. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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

Name	描述	默认值	类型
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 - 将提取的令牌嵌套到其祖先上下文 u - 将提取的令牌解压缩到其子内容 t - 提取指定元素的文本内容。		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 95 章 XPATH

Camel 支持 XPath 以允许在 DSL 中使用 Expression 或 Predicate。

例如，您可以使用 XPath 在 Message Filter 中创建一个 predicate，或作为 Recipient List 的表达式。

95.1. XPATH 语言选项

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	文档类型的类名称为 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	对要在 registry 中查找的自定义 XPathFactory 的引用。
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可以帮助的命名空间。
headerName		字符串	作为输入的标头名称，而不是消息的正文。

Name	默认值	Java 类型	描述
threadSafety		布尔值	是否为 xpath 表达式返回的结果启用 thread-safety。这在使用 NODESET 作为结果类型时适用，返回的设置具有多个元素。在这种情况下，如果您同时处理 NODESET，比如在并行处理模式下从 Camel Splitter EIP 处理 NODESET，则可能会出现线程安全的问题。这个选项通过对节点进行确定的副本来防止并发问题。如果您在应用程序中使用 camel-saxon 或 Saxon，则建议打开此选项。saxon 有线程安全的问题，可通过打开此选项来防止这些问题。
preCompile		布尔值	是否在初始化阶段启用预编译 xpath 表达式。默认情况下启用预编译。这可用于关闭，例如，在启动阶段需要编译阶段的情况，例如，如果应用程序正在编译时间（例如，camel-quarkus），然后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。

95.2. 命名空间

您可以使用 Namespaces 帮助程序类轻松使用带有 XPath 表达式的命名空间。

95.3. 变量

XPath 中的变量在不同的命名空间中定义。默认命名空间是 http://camel.apache.org/schema/spring。

命名空间 URI	本地部分	类型	描述
http://camel.apache.org/xml/in/	in	消息	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
http://camel.apache.org/xml/variables/environment-variables	env	对象	OS 环境变量

命名空间 URI	本地部分	类型	描述
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*

95.3.1. 给定的命名空间

如果给出了命名空间，则 Camel 会精确指示要返回的内容。但是，当解析 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。

95.3.2. 没有给定的命名空间

如果没有给定命名空间，则 Camel 仅根据本地部分解析。Camel 将在以下步骤中尝试解析变量：

- *来自 variables，它使用 variable(name, value) fluent 构建程序设置。*
- *来自 message.in.header，如果有一个带有给定键的标头*
- *来自 exchange.properties，如果有一个带有给定键的属性*

95.4. FUNCTIONS

Camel 添加以下 XPath 函数，可用于访问交换：

功能	参数	类型	描述
in:body	none	对象	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** 搭配使用时。*

下面是一个示例，其中显示了一些正在使用的功能。

95.4.1. 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。

95.5. 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，当您将 XPath 用作 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将消息正文转换为 字符串，然后再安全地重新读取多次。

```
from("queue:foo").
  filter().xpath("//foo")).
  to("queue:bar")
```

```
from("queue:foo").
  choice().xpath("//foo")).to("queue:bar").
  otherwise().to("queue:others");
```

95.6. 设置结果类型

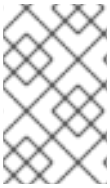
XPath 表达式将使用原生 XML 对象（如 org.w3c.dom.NodeList ）返回结果类型。但是，您可能希望结果类型成为 字符串。为此，您必须指示要使用的结果类型的 XPath。

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。

95.7. 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")
```

95.8. 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>
```

95.9. 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）。

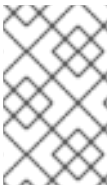
在下面的 XML 示例中，我们使用 Spring XML，其中在 root 标签 Bean 中声明命名空间，在 xmlns:foo="http://example.com/person" 一行中：

```
<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。

95.10. 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 @XPath 等各种语言从消息中提取值并将其绑定到方法参数。



注意

默认 @XPath 注释有 SOAP 和 XML 命名空间可用。

```
public class Foo {

    @Consume(uri = "activemq:my.queue")
    public void doSomething(@XPath("/person/@name") String name, String xml) {
        // process the inbound message here
    }
}
```

95.11. 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 交换。如果您要将其用作进行自定义 XPath 评估的帮助程序，请非常方便。

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。

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 字符串 结果评估是一个常见要求，并使这一简单：

```
String name = XPathBuilder.xpath("foo/bar").evaluate(context, "<foo><bar>cheese</bar>
</foo>");
```

95.12. 将 SAXON 与 XPATHBUILDER 搭配使用

您需要将 camel-saxon 添加为项目的依赖项。

现在，Saxon 与 XPathBuilder 搭配使用，它可以通过多种方式完成，如下所示

- 使用自定义 XPathFactory
- 使用 ObjectModel

95.12.1. 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。

这个单元测试显示了如何使用 Saxon 完成此操作：

如果 Camel 使用非默认 XPathFactory，则 Camel 将记录在 INFO 级别，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
```

95.12.2. 从 XML DSL 启用 Saxon

与 Java DSL 类似，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>
```

95.13. 命名空间审核以协助调试

用户经常遇到的许多与 XPath 相关的问题都链接到命名空间的使用。您可能在消息中存在的命名空间之间有一些错误对齐，并且您的 XPath 表达式了解或引用。无法找到 XML 元素和属性的 XPath predicates 或表达式，因为命名空间问题可能只是像 它们无法正常工作，当它们实际上都不存在 命名空间定义时。

XML 中的命名空间是完全必要的，同时我们希望通过对空闲命名空间实施一定程度或 voodoo 来简化它们的使用，但实际上，这个路径的任何操作都会与标准不兼容，并大大隐藏互操作性。

因此，我们最多可以通过为 XPath 表达式语言添加两个新功能并可从 predicates 和表达式访问，从而帮助您调试这些问题。

95.13.1. 日志记录 XPath 表达式/索引的命名空间上下文

每次在内部池中创建新的 XPath 表达式时，Camel 会在 org.apache.camel.language.xpath.XPathBuilder 日志记录器下记录表达式的命名空间上下文。由于 Camel 以分级方式表示命名空间上下文（父关系），所以整个树都会以递归方式输出，其格式如下：

> [me: {prefix -> namespace}, {prefix -> namespace}], [parent: [me: {prefix -> namespace}, {prefix -> namespace}], [parent: [me: {prefix -> namespace}]]]

这些选项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 logger）上启用 TRACE 日志记录
- 启用 logNamespaces 选项，如以下部分所示，这会在 INFO 级别中进行日志记录

95.13.2. 审计命名空间

Camel 可以在评估 XPath 表达式前发现并转储每个传入消息上存在的所有命名空间，从而为您提供帮助您分析并固定可能的命名空间问题所需的所有信息。

为达到此目的，它在内部使用另一个特殊定制的 XPath 表达式来提取消息中显示的所有命名空间映射，显示每个映射的前缀和完整命名空间 URI。

需要考虑的一些点：

- 隐式 XML 命名空间(xmlns:xml="http://www.w3.org/XML/1998/namespace")被禁止在输出中，因为它没有值
- 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。认为顶级"a"前缀，内部元素中可分配不同的命名空间，或者在内部范围内更改默认命名空间。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>
```

审计的结果将显示在 org.apache.camel.language.xpath.XPathBuilder 日志记录器下的 INFO 级别，并类似如下：

```
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]}
```

95.14. 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中的文件：

```
.setHeader("myHeader").xpath("resource:classpath:myxpath.txt", String.class)
```

95.15. 依赖项

要在 camel 路由中使用 XPath，您需要添加实施 XPath 语言的 camel-xpath 依赖项。

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

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-xpath</artifactId>
  <version>{CamelSBProjectVersion}</version>
</dependency>
```

95.16. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

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	文档类型的类名称为 org.w3c.dom.Document。		字符串
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	对要在 registry 中查找的自定义 XPathFactory 的引用。		字符串
camel.language.xpath.log-namespaces	是否记录在故障排除过程中可以帮助的命名空间。	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。默认情况下启用预编译。这可用于关闭，例如，在启动阶段需要编译阶段的情况，例如，如果应用程序正在编译时间（例如，camel-quarkus），然后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。这在使用 NODESET 作为结果类型时适用，返回的设置具有多个元素。在这种情况下，如果您同时处理 NODESET，比如在并行处理模式下从 Camel Splitter EIP 处理 NODESET，则可能会出现线程安全的问题。这个选项通过对节点进行确定的副本来防止并发问题。如果您在应用程序中使用 camel-saxon 或 Saxon，则建议打开此选项。saxon 有线程安全的问题，可通过打开此选项来防止这些问题。	false	布尔值

Name	描述	默认值	类型
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 96 章 KAMELET MAIN

从 Camel 3.11 开始

一个主要类，用于提升和运行带有 Kamelets （或普通 YAML 路由）的 Camel 独立，用于开发和测试目的。

96.1. 初始配置

KameletMain 预先配置有以下属性：

```
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。

96.2. 自动依赖项下载

Kamelet Main 也可以通过 http/https 从远程位置自动下载 Kamelet YAML 文件，并从 github 中自动下载 Kamelet YAML 文件。

Apache Camel Kamelet Catalog 中的官方 Kamelets 存储在 github 上，并可按原样开箱即用。

例如，Camel 路由可以在 YAML 中代码，它使用目录中的 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 97 章 OPENAPI JAVA

Rest DSL 可以与 camel-openapi-java 模块集成，该模块用于使用 OpenApi 来公开 REST 服务及其 API。

Maven 用户需要将以下依赖项添加到其 pom.xml 中。

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

camel-openapi-java 模块可以从 REST 组件使用（无需 servlet）

97.1. 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");
    }
}
```

97.2. 选项

可使用以下选项配置 OpenApi 模块：要使用 servlet 配置，您可以使用前面所示的 init-param。在 rest-dsl 中直接配置时，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	要设置主机名，请执行以下操作：如果没有配置 camel-openapi-java，则会根据 localhost 计算名称。
方案	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** ：要设置可用 REST 服务的基本路径。该路径相对（例如没有以 http/https 开始），camel-openapi-java 将在运行时计算绝对路径，该路径将是 **protocol://host:port/context-path/base.path**
api.path	字符串	设置 API 可用的路径（如 /api-docs）。该路径相对（例如，没有以 http/https 开头），而 camel-openapi-java 将在运行时计算绝对路径，该路径将是 **protocol://host:port/context-path/api.path**，使用相对路径更为容易。例如，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名称
api.contact.email	字符串	用于 API 相关计数器的电子邮件。

选项	类型	描述
api.contact.url	字符串	网站的 URL 以获取更多联系信息。
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。

97.3. 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 安全 Definition。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 - 使用提供的 url 进行隐式授权
- 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)来指定安全性要使用的其余操作。

```
.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")
```

此处的 get 操作使用 Api Key 安全性，而 put 操作则使用带有允许读和写片断范围的 OAuth 安全性。

97.4. JSON 或 YAML

camel-openapi-java 模块支持开箱即用的 JSon 和 Yaml。您可以使用 /openapi.json 或 /openapi.yaml 在请求 url 中指定。如果没有指定 none，则使用 HTTP Accept 标头来检测是否可以接受 json 或 yaml。如果接受两者或没有接受，则 json 将返回为默认格式。

97.5. 使用XFORWARDHEADERS 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，在 OpenAPI V3 中，它是 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 & X-Forwarded-Proto 决定。

这可以通过禁用 X-Forwarded 标头查找，并通过在 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);
```

97.6. 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，它演示了使用此 OpenApi 组件。

97.7. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

组件支持 1 个选项，如下所列。

Name	描述	默认值	类型
camel.openapi.enabled	启用 Camel Rest DSL 在 Spring Boot 中自动注册其 OpenAPI （如 swagger doc）， 允许 SpringDoc 等工具与 Camel 集成。	true	布尔值

第 98 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 消息的时间。

为发送到/来自 Camel 的传入和传出消息捕获事件(spans)。

98.1. CONFIGURATION

OpenTelemetry tracer 的配置属性有：

选项	默认值	描述
excludePatterns		设置排除模式，将禁用与模式匹配的 Camel 消息的追踪。内容是一个 Set<String>，其中键是模式。模式使用来自 Intercept 的规则。
编码	false	设置是否需要编码标头密钥（特定于connector）。该值是一个布尔值。短划线需要为 JMS 属性键对实例进行编码。

98.1.1. 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。

要明确配置 OpenTelemetry 支持，请实例化 OpenTelemetry Tracer 并初始化 camel 上下文。您可以选择指定 Tracer，或使用 Registry隐式发现它

```
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);
```

98.2. SPRING BOOT

添加 camel-opentelemetry-starter 依赖项，然后使用 @CamelOpenTelemetry 标注主类来打开 OpenTracing。

OpenTelemetry Tracer 从 camel 上下文的 Registry 中隐式获取，除非应用程序定义了 OpenTelemetry Tracer bean。

98.3. JAVA 代理

下载 最新版本的 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。软件包提供完全自动的开箱即用体验。

使用 JVM 的 -javaagent 标志启用检测代理。

```
java -javaagent:path/to/opentelemetry-javaagent.jar \
     -jar myapp.jar
```

默认情况下，OpenSSH Java 代理使用配置为将数据发送到 http://localhost:4317 的 OpenTelemetry 收集器 的 OTLP 导出器。

配置参数作为 Java 系统属性(-D 标志)或环境变量传递。如需完整的配置项目列表，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
```

98.4. SPRING BOOT AUTO-CONFIGURATION

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-opentelemetry-starter</artifactId>
  <version>3.20.1.redhat-00050</version>
  <!-- use the same version as your Camel core version -->
</dependency>
```

-

组件支持 2 个选项，如下所列。

Name	描述	默认值	类型
camel.opentelemetry.encoding	在标头(JMS 需要)激活或停用短划线编码，以进行消息传递。		布尔值
camel.opentelemetry.exclude-patterns	设置排除模式，将禁用与模式匹配的 Camel 消息的追踪。		Set

98.5. MDC LOGGING

当为活跃的 Camel 上下文启用 MDC Logging 时，会为每个路由添加 Trace ID 和生成 ID，其中键分别是 trace_id 和 span_id。

第 99 章 SPRING SECURITY

自 Camel 2.3 起

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。

99.1. 创建授权策略

对路由的访问由 SpringSecurityAuthorizationPolicy 对象的实例控制。策略对象包含运行一组端点和对 Spring Security AuthenticationManager 和 AccessDecisionManager 对象所需的 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）
authenticationManager	**authenticationManager**	上下文中 Spring Security **AuthenticationManager** 对象的名称
accessDecisionManager	**accessDecisionManager**	上下文中 Spring Security **AccessDecisionManager** 对象的名称

Name	默认值	描述
authenticationAdapter	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 （）**。

99.2. 控制对 CAMEL 路由的访问

需要 Spring Security AuthenticationManager 和 AccessDecisionManager 来使用此组件。以下是如何使用 Spring Security 命名空间在 Spring XML 中配置这些对象的示例：

```
<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

99.3. 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，从来自 Jetty 组件的 HTTP 请求标头获取凭证。无论如何收集凭据，都需要将它们放置在 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。

99.4. 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>
```

99.5. SPRING BOOT AUTO-CONFIGURATION

当在 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>
```

组件没有 Spring Boot auto 配置选项。

第 100 章 YAML DSL

从 Camel 3.9 开始

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。

100.1. 定义路由

路由是定义的一组元素，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"
```

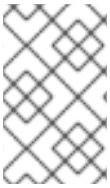
其中，

1

支持路由入口点，默认为 来自 和 rest。

2

处理步骤



注意

每个步骤代表一个 YAML 映射，它只有一个条目，其中字段名称是 EIP 名称。

作为常规规则，每个步骤提供相关定义声明的所有参数，但有一些次要区别/授权：

- 输出 Aware Steps

当交换与 过滤器 表达式或 分割表达式匹配时，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"
```

- 表达式感知步骤

有些 EIP （如 filter 和 split ）支持通过 expression 字段定义 表达式 ：

explicit Expression 字段

```
filter:
  expression:
    simple: "${in.header.continue} == true"
```

要使 DSL 更详细，您可以省略 expression 字段。

隐式表达式字段

```
filter:
  simple: "${in.header.continue} == true"
```

通常，可以内联定义表达式，如上例中所示，但是如果您需要提供更多信息，您可以"取消注册"表达式定义并配置表达式定义并配置表达式定义。

完整的表达式定义

```
filter:
  tokenize:
    token: "<"
    end-token: ">"
```

- 数据格式 Aware 步骤

EIP marshal 和 unmarshal 支持数据格式的定义：

```
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放置，如 json: {}

100.2. 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"
```

100.3. 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 的完全限定域名

(3) 要设置的 bean 的属性

bean 的属性可以使用映射或属性风格来定义，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。

100.4. 在语言上配置选项

有些 语言 有您可能需要使用的额外配置。

例如，可将 JSONPath 配置为忽略 JSON 解析错误。这在使用 Content Based Router 且希望将消息路由到不同的端点时。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的 JSonPath 表达式会失败，其他时间不会失败。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，以下消息失败了 JSonPath 表达式 person.middlename，因为 JSON 有效负载没有中间名称字段。为了缓解这一点，我们限制了例外。

```
{
  "person": {
    "firstname": "John",
    "lastname": "Doe"
  }
}
```

100.5. 外部示例

您可以使用 Camel 示例中的 main-yaml 找到一组示例，该示例演示了如何使用 YAML 创建 Camel 路由。您还可以引用使用 YAML 定义每个 Kamelets 的 Camel Kamelets。