Red Hat build of Apache Camel Extensions for Quarkus 2.13

将 Fuse 7 Applications 迁移到 Camel Extensions for Quarkus

将 Fuse 7 Applications 迁移到红帽提供的 Camel Extensions for Quarkus

Last Updated: 2023-09-19

Red Hat build of Apache Camel Extensions for Quarkus 2.13 将 Fuse 7 Applications 迁移到 Camel Extensions for Quarkus

将 Fuse 7 Applications 迁移到红帽提供的 Camel Extensions for Quarkus

法律通告

摘要

将 Fuse 7 Applications 迁移到 Camel Extensions for Quarkus 提供了有关从 Red Hat Fuse 7 迁移到 Red Hat build for Camel Extensions for Quarkus 的信息。

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 将 FUSE 7 应用程序迁移到 CAMEL EXTENSIONS FOR QUARKUS 概述

fuse

红帽 Fuse 是基于 Apache Camel 和 Apache Karaf 等开源社区的敏捷集成解决方案。红帽 Fuse 是一个轻量级、灵活的集成平台，可实现快速的内部云集成。

您可以使用三个不同的运行时运行 Red Hat Fuse：

- 支持 OSGi 应用程序的 Karaf
- Spring Boot
- JBoss EAP （企业应用平台）

Camel Extensions for Quarkus

Camel Extensions for Quarkus (CEQ)将 Apache Camel 及其大量组件库的集成功能引入 Quarkus 运行时。Red Hat build of Camel Quarkus 为许多 Camel 组件提供 Quarkus 扩展。

Camel Quarkus 利用了 Camel 3 中带来的许多性能改进，从而降低内存占用量、对反映的依赖，以及更快的启动时间。

在 CEQ 应用程序中，您可以使用 Java DSL 定义 Camel 路由，以便您可以将 Fuse 应用程序中使用的 Camel 路由迁移到 CEQ。

Camel on EAP

遵循 OSGI 依赖项管理概念的 OSGI 依赖项管理概念和 EAP 遵循EE 规范的应用服务器受到容器化应用的采用的影响。

容器已逐渐成为打包应用的主要方法。因此，管理包括部署、扩展、集群和负载平衡的应用程序的责任已使用 Kubernetes 从应用服务器转移到容器编排。

虽然 EAP 在 Red Hat Openshift 上继续被支持，但 EAP 服务器上不再支持 Camel 3。因此，如果您在 EAP 服务器上运行 Fuse 7 应用程序，您应该考虑将您的应用程序迁移到红帽构建的 Apache Camel for Spring Boot 或 Red Hat build of Apache Camel Extensions for Quarkus，并利用迁移过程的好处来考虑重新设计，或部分重新设计的应用程序，从单调到微服务架构。

如果不使用 Openshift，在为 Spring Boot 和 Quarkus 部署应用程序时，RHEL 虚拟机仍然是有效的方法，而 Quarkus 也受益于其原生编译功能。评估支持在此类平台上管理微服务架构的工具非常重要。

红帽使用 Red Hat Ansible for Middleware 集合 通过 Ansible 提供此功能 。

1.1. 标准迁移路径

1.1.1. XML 路径

使用 Spring XML 或 Blueprint XML 编写的 Fuse 应用程序应迁移到基于 XML 的类别，并可针对迁移步骤没有区别的 Spring Boot 或 Quarkus 运行时。

1.1.2. Java 路径

使用 Java DSL 编写的 Fuse 应用程序应迁移到基于 Java 的类别，并可针对 Spring Boot 或 Quarkus 运行时，在迁移步骤中没有差别。

1.2. 架构更改

OpenShift 已经替换了 Fabric8 作为 Fuse 6 用户的运行时平台，也是您的 Fuse 应用程序迁移的建议目标。

迁移应用程序时，您应该考虑以下架构更改：

- 如果您的 Fuse 6 应用依赖于 Fabric8 服务发现，则在 OpenShift 上运行 Camel 3 时应使用 Kubernetes 服务发现。
- 如果您的 Fuse 6 应用程序依赖于 OSGi 捆绑包配置，则在 OpenShift 上运行 Camel 3 时应使用 Kubernetes ConfigMap 和 Secret。
- 如果您的应用程序使用基于文件的路由定义，请考虑在 OpenShift 上运行 Camel 3 时使用 AWS S3 技术。
- 如果您的应用程序使用标准文件系统，则生成的 Spring Boot 或 Quarkus 应用程序应该部署到标准 RHEL 虚拟机上，而不是 Openshift 平台。
- 对处理 SSL 要求的 Openshift 路由器的入站 HTTPS 连接委托。
- 将 Hystrix 功能委派给 Service Mesh。

第 2 章 将 CAMEL 路由从 FUSE 7 迁移到 CAMEL EXTENSIONS FOR QUARKUS (CEQ)

注意

您可以使用 Java DSL、XML IO DSL 或 YAML 在 CEQ 应用程序中定义 Camel 路由。

2.1. JAVA DSL 路由迁移示例

要将 Java DSL 路由定义从 Fuse 应用程序迁移到 CEQ，您可以将现有路由定义直接复制到 CEQ 应用，并将必要的依赖项添加到您的 CEQ pom.xml 文件中。

在本例中，我们将通过将 Java DSL 路由复制到 CEQ 应用中名为 **Routes.java** 的文件，将基于内容的路由定义从 Fuse 7 应用迁移到新的 CEQ 应用程序。

流程

1. 使用 **code.quarkus.redhat.com** 网站，选择本例所需的扩展：
   - camel-quarkus-file
   - camel-quarkus-xpath
2. 进入从上一步中提取生成的项目文件的目录：

```
$ cd <directory_name>
```

3. 在 **src/main/java/org/acme/** 子文件夹中，创建名为 **Routes.java** 的文件。
4. 将 Fuse 应用程序的路由定义添加到 **Routes.java** 中，如下例所示：

```
package org.acme;

import org.apache.camel.builder.RouteBuilder;

public class Routes extends RouteBuilder {
 // Add your Java DSL route definition here
    public void configure() {
     from("file:work/cbr/input")
            .log("Receiving order ${file:name}")
            .choice()
                .when().xpath("//order/customer/country[text() = 'UK']")
                    .log("Sending order ${file:name} to the UK")
                    .to("file:work/cbr/output/uk")
                .when().xpath("//order/customer/country[text() = 'US']")
                    .log("Sending order ${file:name} to the US")
                    .to("file:work/cbr/output/uk")
                .otherwise()
                    .log("Sending order ${file:name} to another country")
                    .to("file:work/cbr/output/others");
    }

}
```

5. 编译您的 CEQ 应用程序。

```
mvn clean compile quarkus:dev
```

注意

此命令编译项目，启动应用程序，并允许 Quarkus 工具监视工作区中的更改。项目中的任何修改都会自动在正在运行的应用程序中生效。

2.2. 蓝图 XML DSL 路由迁移

要将 Blueprint XML 路由定义从 Fuse 应用程序迁移到 CEQ，请使用 **camel-quarkus-xml-io-dsl** 扩展，并将 Fuse 应用程序路由定义直接复制到 CEQ 应用程序。然后，您需要将所需的依赖项添加到 CEQ **pom.xml** 文件中，并在 **application.properties** 文件中更新您的 CEQ 配置。

注意

CEQ 支持 Camel 3，而 Fuse 7 支持 Camel 2。有关将 Red Hat Fuse 7 应用程序迁移到 CEQ 时升级 Camel 的更多信息，请参阅 从 Camel 2 迁移到 Camel 3 。

有关在 Camel Quarkus 中使用 Bean 的更多信息，请参阅 *开发带有 Camel Extensions for Quarkus 指南中的 CDI 和 Camel Bean 组件* 部分。

2.2.1. xml-IO-DSL 限制

您可以使用 **camel-quarkus-xml-io-dsl** 扩展来帮助将 Blueprint XML 路由定义迁移到 CEQ。

camel-quarkus-xml-io-dsl 扩展只支持以下 < **camelContext&** gt; 子元素：

- routeTemplates
- templatedRoutes
- rests
- Routes
- routeConfigurations

注意

因为 Blueprint XML 支持 **camel-quarkus-xml-io-dsl** 扩展不支持的其他 bean 定义，您可能需要重写 Blueprint XML 路由定义中包含的其他 bean 定义。

您必须在单独的文件中定义每个元素(XML IO DSL)。例如，这是 Blueprint XML 路由定义的简化示例：

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
  <camelContext xmlns="http://camel.apache.org/schema/blueprint">
    <restConfiguration contextPath="/camel" />
    <rest path="/books">
      <get uri="/">
        <to ..../>
      </get>
```

```
        </rest>
        <route>
            <from ..../>
        </route>
    </camelContext>
</blueprint>
```

您可以使用以下文件中定义的 XML IO DSL 将此蓝图 XML 路由定义迁移到 CEQ：

src/main/resources/routes/camel-rests.xml

```
<rests xmlns="http://camel.apache.org/schema/spring">
    <rest path="/books">
    <get path="/">
        <to ..../>
    </get>
    </rest>
</rests>
```

src/main/resources/routes/camel-routes.xml

```
<routes xmlns="http://camel.apache.org/schema/spring">
    <route>
        <from ..../>
    </route>
</routes>
```

您必须使用 Java DSL 来定义不支持的其他元素，如 < **restConfiguration>**。例如，使用 **camel-rests.xml** 文件中定义的路由构建器，如下所示：

src/main/resources/routes/camel-rests.xml

```
import org.apache.camel.builder.RouteBuilder;
public class Routes extends RouteBuilder {
    public void configure() {
        restConfiguration()
            .contextPath("/camel");
    }
}
```

2.2.2. 蓝图 XML DSL 路由迁移示例

注意

有关使用 XML IO DSL 扩展的更多信息，请参阅 Camel Extensions for Quarkus 参考中的 XML IO DSL 文档。

在本例中，您要将基于内容的路由定义从 Fuse 应用程序迁移到新的 CEQ 应用程序，方法是将 Blueprint XML 路由定义复制到 CEQ 应用程序中名为 **camel-routes.xml** 的文件。

步骤

1. 使用 **code.quarkus.redhat.com** 网站，为本例选择以下扩展：

- camel-quarkus-xml-io-dsl
- camel-quarkus-file
- camel-quarkus-xpath

2. 选择 *Generate your application* 来确认您的选择并显示包含您生成的项目的存档的下载链接。
3. 选择 Download the ZIP 将带有生成的项目文件的存档保存到您的机器中。
4. 提取存档的内容。
5. 进入从上一步中提取生成的项目文件的目录：

```
$ cd <directory_name>
```

6. 在 **src/main/resources/routes/** 目录中创建一个名为 **camel-routes.xml** 的文件。
7. 将以下 **blueprint-example.xml** 示例中的 < **route** > 元素和子元素复制到 **camel-routes.xml** 文件中：

blueprint-example.xml

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
  <camelContext id="cbr-example-context"
xmlns="http://camel.apache.org/schema/blueprint">
    <route id="cbr-route">
      <from id="_from1" uri="file:work/cbr/input"/>
      <log id="_log1" message="Receiving order ${file:name}"/>
      <choice id="_choice1">
        <when id="_when1">
          <xpath id="_xpath1">/order/customer/country = 'UK'</xpath>
          <log id="_log2" message="Sending order ${file:name} to the UK"/>
          <to id="_to1" uri="file:work/cbr/output/uk"/>
        </when>
        <when id="_when2">
          <xpath id="_xpath2">/order/customer/country = 'US'</xpath>
          <log id="_log3" message="Sending order ${file:name} to the US"/>
          <to id="_to2" uri="file:work/cbr/output/us"/>
        </when>
        <otherwise id="_otherwise1">
          <log id="_log4" message="Sending order ${file:name} to another country"/>
          <to id="_to3" uri="file:work/cbr/output/others"/>
        </otherwise>
      </choice>
      <log id="_log5" message="Done processing ${file:name}"/>
    </route>
  </camelContext>
</blueprint>
```

camel-routes.xml

```
<route id="cbr-route">
  <from id="_from1" uri="file:work/cbr/input"/>
  <log id="_log1" message="Receiving order ${file:name}"/>
```

```
    <choice id="_choice1">
      <when id="_when1">
        <xpath id="_xpath1">/order/customer/country = 'UK'</xpath>
        <log id="_log2" message="Sending order ${file:name} to the UK"/>
        <to id="_to1" uri="file:work/cbr/output/uk"/>
      </when>
      <when id="_when2">
        <xpath id="_xpath2">/order/customer/country = 'US'</xpath>
        <log id="_log3" message="Sending order ${file:name} to the US"/>
        <to id="_to2" uri="file:work/cbr/output/us"/>
      </when>
      <otherwise id="_otherwise1">
        <log id="_log4" message="Sending order ${file:name} to another country"/>
        <to id="_to3" uri="file:work/cbr/output/others"/>
      </otherwise>
    </choice>
    <log id="_log5" message="Done processing ${file:name}"/>
  </route>
```

8. 修改 **application.properties**

```
# Camel
#
camel.context.name = camel-quarkus-xml-io-dsl-example
camel.main.routes-include-pattern = file:src/main/resources/routes/camel-routes.xml
```

9. 编译您的 CEQ 应用程序。

```
mvn clean compile quarkus:dev
```

注意

此命令编译项目，启动应用程序，并允许 Quarkus 工具监视工作区中的更改。项目中的任何修改都会自动在正在运行的应用程序中生效。

2.3. 其他资源

有关 Camel Extensions for Quarkus (CEQ)的更多信息，请参阅以下文档：

- Camel Extensions for Quarkus 参考
- Camel Extensions for Quarkus 入门
- 使用 Camel Extensions for Quarkus 开发应用程序

第 3 章 从 CAMEL 2 迁移到 CAMEL 3

Camel Extensions for Quarkus 支持 Camel 版本 3，而 Fuse 7 支持的 Camel 版本 2。本节提供了在将 Red Hat Fuse 7 应用程序迁移到 Camel Extensions for Quarkus 时升级 Camel 的信息。

3.1. JAVA 版本

Camel 3 支持 Java 17 和 Java 11，但不支持 Java 8。

3.2. CAMEL-CORE 的模块化

在 Camel 3.x 中，**camel-core** 已被分成多个 JAR，如下所示：

- camel-api
- camel-base
- camel-caffeine-lrucache
- camel-cloud
- camel-core
- camel-jaxp
- camel-main
- camel-management-api
- camel-management
- camel-support
- camel-util
- camel-util-json

Apache Camel 的 Maven 用户可以继续使用依赖项 **camel-core**，它对其所有模块有传输依赖项，但 **camel-main** 除外，因此不需要迁移。

3.3. 组件的模块化

在 Camel 3.x 中，一些 camel-core 组件被移到各个组件中。

- camel-attachments
- camel-bean
- camel-browse
- camel-controlbus
- camel-dataformat
- camel-dataset

- camel-direct
- camel-directvm
- camel-file
- camel-language
- camel-log
- camel-mock
- camel-ref
- camel-rest
- camel-saga
- camel-scheduler
- camel-seda
- camel-stub
- camel-timer
- camel-validator
- camel-vm
- camel-xpath
- camel-xslt
- camel-xslt-saxon
- camel-zip-deflater

3.4. 不支持每个应用程序有多个 CAMELCONTEXTS

对多个 CamelContexts 的支持已被删除，建议每个部署只有一个 CamelContext，并被支持。因此，各种 Camel 注释上的 **context** 属性（如 **@EndpointInject**、**@Produce**、**@Consume** 等）已被删除。

3.5. 弃用的 API 和组件

Camel 3 中删除了来自 Camel 2.x 的所有已弃用的 API 和组件。

3.5.1. 删除的组件

Camel 2.x 中的所有已弃用的组件都在 Camel 3.x 中删除，包括旧的 **camel-http**, **camel-hdfs**, **camel-mina**, **camel-mongodb**, **camel-netty**, **camel-netty-http**, **camel-quartz**, **camel-restlet** 和 **camel-rx** 组件。

- 删除了 **camel-jibx** 组件。
- 删除了 **camel-boon** 数据格式。

- 删除了 **camel-linkedin** 组件，因为 不再支持 Linkedin API 1.0。对新 2.0 API 的支持由 CAMEL-13813 跟踪。
- **camel-zookeeper** 删除了其路由策略功能，而是使用 **ZooKeeperClusterService** 或 **camel-zookeeper-master** 组件。
- **camel-jetty** 组件不再支持制作者（已被删除），改为使用 **camel-http** 组件。
- Twitter **-streaming** 组件已被删除，因为它依赖于已弃用的 Twitter Streaming API，不再可以正常工作。

3.5.2. 重命名组件

在 Camel 3.x 中重命名以下组件。

- camel **-microprofile-metrics** 已重命名为 **camel-micrometer**
- **测试** 组件已重命名为 **dataset-test**，并从 **camel-core** 移到 **camel-dataset** JAR 中。
- **http4** 组件已重命名为 **http**，它对应于从 **org.apache.camel.component.http4** 到 **org.apache.camel.component.http** 的组件软件包。现在，支持的方案只能是 **http** 和 **https**。
- **hdfs2** 组件已重命名为 **hdfs**，它对应于从 **org.apache.camel.component.hdfs2** 到 **org.apache.camel.component.hdfs** 的组件软件包。现在支持的方案是 **hdfs**。
- **mina2** 组件已重命名为 **mina**，它对应于来自从 **org.apache.camel.component.mina2** 到 **org.apache.camel.component.mina** 2 的软件包。现在支持的方案是 **mina**。
- **mongodb3** 组件已重命名为 **mongodb**，它对应于从 **org.apache.camel.component.mongodb3** 到 **org.apache.camel.component.mongodb** 的组件软件包。现在支持的方案是 **mongodb**。
- **netty4-http** 组件已重命名为 **netty-http**，它对应于从 **org.apache.camel.component.netty4.http** 到 **org.apache.camel.component.netty.http** 的组件软件包。现在支持的方案是 **netty-http**。
- **netty4** 组件已重命名为 **netty**，它对应于从 **org.apache.camel.component.netty4** 到 **org.apache.camel.component.netty** 的组件软件包。现在支持的方案是 **netty**。
- **quartz2** 组件已重命名为 **quartz**，它对应于从 **org.apache.camel.component.quartz2** 到 **org.apache.camel.component.quartz** 的组件软件包。现在支持的方案是 **quartz**。
- **rxjava2** 组件已重命名为 **rxjava**，它对应于从 **org.apache.camel.component.rxjava2** 到 **org.apache.camel.component.rxjava** 的组件软件包。
- 将 **camel-jetty9** 重命名为 **camel-jetty**。现在，支持的方案是 **jetty**。

3.6. CAMEL 组件的更改

3.6.1. Mock 组件

mock 组件已从 **camel-core** 移出。由于其 *assertion 子句构建器* 上的许多方法已被删除。

3.6.2. ActiveMQ

如果您使用 **activemq-camel** 组件，则应迁移到使用 **camel-activemq** 组件，其中组件名称已从 **org.apache.activemq.camel.component.ActiveMQComponent** 改为 **org.apache.camel.component.activemq.ActiveMQComponent**。

3.6.3. AWS

组件 **camel-aws** 被分成多个组件：

- camel-aws-cw
- camel-aws-ddb （包含 ddb 和 ddbstreams 组件）
- camel-aws-ec2
- camel-aws-iam
- camel-aws-kinesis （其中包含 kinesis 和 kinesis-firehose 组件）
- camel-aws-kms
- camel-aws-lambda
- camel-aws-mq
- camel-aws-s3
- camel-aws-sdb
- camel-aws-ses
- camel-aws-sns
- camel-aws-sqs
- camel-aws-swf

注意

建议为这些组件添加 specifc 依赖项。

3.6.4. Camel CXF

camel-cxf JAR 已分为 SOAP 与 REST 和 Spring JAR。当从 **came-cxf** 进行迁移时，建议从以下列表中选择特定的 JAR。

- **camel-cxf-soap**
- **camel-cxf-spring-soap**
- **camel-cxf-rest**
- **camel-cxf-spring-rest**
- **camel-cxf-transport**
- **camel-cxf-spring-transport**

例如，如果您使用 CXF 用于 SOAP 并使用 Spring XML，那么在从 **camel-cxf** 进行迁移时，请选择 **camel-cxf-spring-soap** 和 **camel-cxf-spring-transport**。

使用 Spring Boot 时，当您从 **camel-cxf-starter** 迁移到 SOAP 或 REST 时，从以下入门中选择：

- **camel-cxf-soap-starter**
- **camel-cxf-rest-starter**

camel-cxf XML XSD 模式也更改了命名空间。

表 3.1. 对命名空间的更改

旧命名空间	新命名空间
http://camel.apache.org/schema/cxf	http://camel.apache.org/schema/cxf/jaxws
http://camel.apache.org/schema/cxf/camel-cxf.xsd	http://camel.apache.org/schema/cxf/jaxws/camel-cxf.xsd
http://camel.apache.org/schema/cxf	http://camel.apache.org/schema/cxf/jaxrs
http://camel.apache.org/schema/cxf/camel-cxf.xsd	http://camel.apache.org/schema/cxf/jaxrs/camel-cxf.xsd

camel-cxf SOAP 组件被移到一个新的 **jaxws** 子软件包，即 **org.apache.camel.component.cxf** 现在是 **org.apache.camel.component.cxf.jaws**。例如，**CxfComponent** 类现在位于 **org.apache.camel.component.cxf.jaxws**。

3.6.5. FHIR

camel-fhir 组件已将其 hapi-fhir 依赖项升级到 4.1.0。默认 FHIR 版本已改为 R4。因此，如果需要 DSTU3，则必须明确设置。

3.6.6. Kafka

camel-kafka 组件删除了选项 **bridgeEndpoint** 和 **circularTopicDetection**，因为组件不再需要，因为组件在 Camel 2.x 上可以正常工作。换句话说，**camel-kafka** 将从 endpoint uri 发送消息到主题。要覆盖它，请使用带有新主题的 **KafkaConstants.OVERRIDE_TOPIC** 标头。请参阅 **camel-kafka** 组件文档以了解更多信息。

3.6.7. telegram

camel-telegram 组件已将授权令牌从 uri-path 移到查询参数，如 migrate

```
telegram:bots/myTokenHere
```

to

```
telegram:bots?authorizationToken=myTokenHere
```

3.6.8. JMX

如果您只使用 **camel-core** 作为依赖项运行 Camel 独立，并且希望开箱即用启用 JMX，则需要将 **camel-management** 添加为依赖项。

对于使用 **ManagedCamelContext**，您需要从 **CamelContext** 获取此扩展，如下所示：

```
ManagedCamelContext managed = camelContext.getExtension(ManagedCamelContext.class);
```

3.6.9. XSLT

XSLT 组件已从 camel-core 移到 **camel-xslt** 和 **camel-xslt-saxon**。组件被分开，因此 **camel-xslt** 用于使用 JDK XSTL 引擎(Xalan)，**camel-xslt-saxon** 是使用 Saxon 时的。这意味着，您应该在 Camel 端点 URI 中使用 **xslt** 和 **xslt-saxon** 作为组件名称。如果您使用 XSLT 聚合策略，则使用 **org.apache.camel.component.xslt.saxon.XsltSaxonAggregationStrategy** 进行 Saxon 支持。并使用 **org.apache.camel.component.xslt.saxon.XsltSaxonBuilder** 进行 Saxon 支持（如果使用 xslt 构建器）。另请注意，只有 **camel-xslt-saxon** 中也支持 **allowStax**，因为 JDK XSLT 不支持它。

3.6.10. XML DSL 迁移

XML DSL 稍微改变。

自定义负载均衡器 EIP 已从 < **custom> 改为 &lt**; **customLoadBalancer>**

在 **<secureXML>** tag 中，XMLSecurity 数据格式将属性 **keyOrTrustStoreParametersId** 重新命名为 **keyOrTrustStoreParametersRef**。

& **lt;zipFile&** gt; 数据格式已重命名为 < **zipfile>**。

3.7. 迁移 CAMEL MAVEN 插件

camel-maven-plugin 已分成两个 maven 插件：

camel-maven-plugin

camel-maven-plugin 具有 **run** 目标，旨在单独运行 Camel 应用程序。如需更多信息，请参阅 https://camel.apache.org/manual/camel-maven-plugin.html。

camel-report-maven-plugin

camel-report-maven-plugin 具有 **validate** 和 **route-coverage** 目标，用于生成 Camel 项目报告，如验证 Camel 端点 URI 和路由覆盖报告等。如需更多信息，请参阅 https://camel.apache.org/manual/camel-report-maven-plugin.html。