Red Hat build of Apache Camel 4.4

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

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

Last Updated: 2024-06-29

Red Hat build of Apache Camel 4.4 将 Fuse 7 Applications 迁移到 Red Hat build of Apache Camel for Quarkus

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

法律通告

摘要

将 Fuse 7 Applications 迁移到 Red Hat build of Apache Camel for Quarkus 提供了有关从 Red Hat Fuse 7 迁移到 Red Hat build of Apache Camel 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。详情请查看 CTO Chris Wright 的信息。

第 1 章 将 FUSE 7 应用程序迁移到 APACHE CAMEL FOR QUARKUS 的红帽构建概述

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）

Red Hat build of Apache Camel for Quarkus

红帽构建的 Apache Camel for Quarkus 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 Red Hat build of Apache Camel for Quarkus 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 for Quarkus，并利用迁移流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。

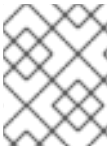
1.2. 架构更改

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



注意

您可以使用 Java DSL、XML IO DSL 或 YAML 在 Red Hat build of Apache Camel for Quarkus 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红帽构建的 Apache Camel for Quarkus 应用程序，并将必要的依赖项添加到红帽构建的 Apache Camel for Quarkus pom.xml 文件。

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，*请参阅使用红帽构建的 Apache Camel 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

{link

有关使用 XML IO DSL 扩展的更多信息，请参阅红帽构建的 Apache Camel for Quarkus 扩展中的 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。

流程

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。

第 3 章 迁移 APACHE CAMEL

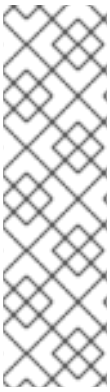
3.1. 关于 CAMEL 迁移指南

本指南详细介绍了迁移应用程序时必须考虑的 Apache Camel 组件中的更改。本指南提供有关以下更改的信息。

- 支持的 Java 版本
- 对 Apache Camel 组件的更改和已弃用的组件
- 对 API 和已弃用的 API 的更改
- EIP 更新
- 更新至追踪和健康检查

3.2. 迁移到 APACHE CAMEL 4

本节提供了可帮助您将 Apache Camel 应用程序从 3.20 或更高版本迁移到 4.0 的信息。



注意

有关单个版本的详情，请参考：

- Apache Camel 3.x 升级指南.
- Apache Camel 4.x 升级指南.

有关如何升级 Apache Camel Quarkus 的详情，请参考：

- Camel Quarkus 3.2.0 迁移指南.

3.2.1. Java 版本

Apache Camel 4 支持 Java 17。丢弃了对 Java 11 的支持。

3.2.2. 删除的组件

以下组件已被删除：

组件	其他组件
camel-any23	none
camel-atlasmap	none
camel-atmos	none
camel-caffeine-lrucache	camel-cache, camel-ignite, camel-infinispan

组件	其他组件
camel-cdi	camel-spring-boot, camel-quarkus
camel-corda	none
camel-directvm	camel-direct
camel-dozer	camel-mapstruct
camel-elasticsearch-rest	camel-elasticsearch
camel-gora	none
camel-hbase	none
camel-hyperledger-aries	none
camel-iota	none
camel-ipfs	none
camel-jbpm	none
camel-jclouds	none
camel-johnzon	camel-jackson, camel-fastjson, camel-gson
camel-microprofile-metrics	camel-micrometer, camel-opentelemetry
camel-milo	none
camel-opentracing	camel-micrometer, camel-opentelemetry
camel-rabbitmq	spring-rabbitmq-component
camel-rest-swagger	camel-openapi-rest
camel-restdsl-swagger-plugin	camel-restdsl-openapi-plugin
camel-resteasy	camel-cxf, camel-rest
camel-solr	none
camel-spark	none
camel-spring-integration	none

组件	其他组件
camel-swagger-java	camel-openapi-java
camel-websocket	camel-vertx-websocket
camel-websocket-jsr356	camel-vertx-websocket
camel-vertx-kafka	camel-kafka
camel-vm	camel-seda
camel-weka	none
camel-xstream	camel-jacksonxml
camel-zipkin	camel-micrometer, camel-opentelemetry

3.2.3. 日志记录

Camel 4 将日志 facade API **slf4j-api** 从 1.7 升级到 2.0。

3.2.4. JUnit 4

所有基于 JUnit 4.x 的 **camel-test** 模块已被删除。所有测试模块现在都使用 JUnit 5。

3.2.5. API 更改

以下 API 已被弃用并从版本 4 中删除：

- **org.apache.camel.ExchangePattern** 已删除了 **InOptionalOut**。
- 从 **CamelContext** 中移除 **getEndpointMap （）** 方法。
- 删除了 **@FallbackConverter**，因为您应该改为使用 **@Converter (fallback = true)**。
- removed **uri** 属性 **@EndpointInject**,**@Produce**, 和 **@Consume** 替代 值 （默认）。例如，**@Produce (uri = "kafka:cheese")** 应更改为 **@Produce ("kafka:cheese")**
- 删除了 **@UriEndpoint 上的标签**，因为您应该使用 **类别**。
- 删除了 **ProducerTemplate** 上的所有 **asyncCallback** 方法。改为使用 **asyncSend** 或 **asyncRequest**。
- Removed **org.apache.camel.spi.OnCamelContextStart**.改为使用 **org.apache.camel.spi.OnCamelContextStarting**。
- Removed **org.apache.camel.spi.OnCamelContextStop**.改为使用 **org.apache.camel.spi.OnCamelContextStopping**。
- 将 **org.apache.camel.ExtendedCamelContext** 与 **org.apache.camel.CamelContext** 分离。

- 使用 **getCamelContextExtension**替换 **org.apache.camel.CamelContext** 的 **adapt （）**
- 将 **org.apache.camel.ExtendedExchange** 与 **org.apache.camel.Exchange** 分离。
- 使用 **getExchangeExtension**替换 **org.apache.camel.ExtendedExchange** 的 **adapt （）**
- 交换失败处理状态已从定义为 **ExchangePropertyKey.FAILURE_HANDLED** 的属性移到 ExtendedExchange 的成员，可通过 'isFailureHandled （）'method 访问。
- 从 **org.apache.camel.util.concurrent.ThreadPoolRejectedPolicy** 中删除了 **Discard** 和 **DiscardOldest**。
- 删除了 **org.apache.camel.builder.SimpleBuilder**。在某些情况下，通常在 Camel 内部使用 Java DSL。
- 将 **org.apache.camel.support.IntrospectionSupport** 设置为 **camel-core-engine**，仅供内部使用。最终用户应使用 **org.apache.camel.spi.BeanInspection** 替代。
- 从 **org.apache.camel.catalog.CamelCatalog** 中删除了 **archetypeCatalogAsXml** 方法。
- **org.apache.camel.health.HealthCheck** 方法现在默认为 **false**，而不是 **true**。
- **向 org.apache.camel.StreamCache 添加了 位置 方法。**
- **从接口 org.apache.camel.main.Listener 配置 的方法已被删除**
- **org.apache.camel.support.EventNotifierSupport abstract 类现在实现了 CamelContextAware。**
- **CamelContext 上的 dumpRoutes 的类型已从 布尔值 改为 String，以允许指定 xml 或 yaml。**

注意

org.apache.camel.support.PluginHelper 提供对以前来自 CamelContext 的 Camel v3 中可用的各种扩展和上下文插件的简单访问。

3.2.6. EIP 更改

- **每个 EIPs 上删除了 &lt ;description> 的 lang 属性。**
- **InOnly 和 InOut EIPs 已被删除。反之，使用 SetExchangePattern 或 to 指定要使用的交换**

模式。

3.2.6.1. poll Enrich EIP

轮询的端点 URI 现在作为属性存储在 Exchange 上（带有键 CamelToEndpoint），与其他 EIPs 一样。在 URI 作为消息标头存储之前。

3.2.6.2. CircuitBreaker EIP

camel-resilience4j 中的以下选项被错误地定义为属性：

选项
bulkheadEnabled
bulkheadMaxConcurrentCalls
bulkheadMaxWaitDuration
timeoutEnabled
timeoutExecutorService
timeoutDuration
timeoutCancelRunningFuture

这些选项没有在 YAML DSL 中公开，在您需要迁移的 XML DSL 中：

```
<circuitBreaker>
    <resilience4jConfiguration>
        <timeoutEnabled>true</timeoutEnabled>
        <timeoutDuration>2000</timeoutDuration>
    </resilience4jConfiguration>
...
</circuitBreaker>
```

使用以下属性：

```
<circuitBreaker>
    <resilience4jConfiguration timeoutEnabled="true" timeoutDuration="2000"/>
...
```

```
</circuitBreaker>
```

3.2.7. XML DSL

在路由或节点上设置描述的 <description> 已从元素改为一个属性。

Example

```
<route id="myRoute" description="Something that this route do">
  <from uri="kafka:cheese"/>
  ...
</route>
```

3.2.8. 类型 Converter

String → java.io.File converter 已被删除。

3.2.9. Tracing

Tracer 和 Backlog Tracer 不再包含由 Rest DSL 或 route templates 或 Kamelets 创建的路由的内部追踪事件。您可以通过在 tracer 中设置 traceTemplates=true 来打开它。

Backlog Tracer 已被改进，并 *修复了* trace 消息标头（也流类型）。这意味着之前没有跟踪 InputStream 类型的标头，但现在包含。这可能意味着标头流在结尾处，并在后记录标头后可能会出现，因为标头值为空。

3.2.10. UseOriginalMessage / UseOriginalBody

当在 OnException、OnCompletion 或错误处理程序中启用 useOriginalMessage 或 useOriginalBody 时，原始消息正文会完全复制，如果可能转换为 StreamCache，以确保正文可以在访问时重新读取。在以前的版本中，原始正文没有转换为 StreamCache，这可能会导致正文无法读取或关闭流。

3.2.11. Camel Health

现在，健康检查只会开箱即用的就绪度检查。Camel 提供 CamelContextCheck 作为就绪度和存活度检查，因此开箱即用至少一个。默认只启用基于消费者的健康检查。

3.2.11.1. 制作者健康检查

camel.health.components-enabled 选项已重命名为 camel.health.producers-enabled。

有些组件（特别是 AWS）也为生成者提供健康检查；在 Camel 3.x 中，这些健康检查无法正常工作，并在源中禁用。要在 Camel 4 中继续此行为，基于生成者的健康检查被禁用。

请注意，camel-kafka 附带基于制作者的健康检查，在 Camel 3 中工作，因此 Camel 4 中的这个更改意味着这个健康检查被禁用。

您必须在全局范围内启用制作者健康检查，例如在 application.properties 中：

```
camel.health.producers-enabled = true
```

3.2.12. JMX

Camel 现在还包括 doCatch 的 MBeans，并在处理器 MBeans 的树中进行 最后。

ManagedChoiceMBean 已将 choiceStatistics 重命名为 extendedInformation。ManagedFailoverLoadBalancerMBean 将 exceptionStatistics 重命名为 extendedInformation。

CamelContextMBean 和 CamelRouteMBean 删除了方法 dumpRouteAsXml （布尔值 resolvePlaceholders，布尔值 resolveDelegateEndpoints）。

3.2.13. YAML DSL

向后兼容模式 Camel 3.14 或更早版本，允许作为 *路由* 子级 *步骤* 已被删除。

新语法为：

```
- route:
```

```
from:
  uri: "direct:info"
  steps:
  - log: "message"
```

3.2.14. backlog Tracing

选项 backlogTracing=true 现在会自动启用来在启动时启动 tracer。在以前的版本中，tracer 仅可用，之后必须手动启用。可以通过设置 backlogTracingStandby=true 来归档旧行为。

将以下类从 camel-management-api JAR 中的 org.apache.camel.api.management.mbean.BacklogTracerEventMessage 移到 org.apache.camel.spi.BacklogTracerEventMessage。

org.apache.camel.impl.debugger.DefaultBacklogTracerEventMessage 已重构为接口 org.apache.camel.spi.BacklogTracerEventMessage，其中包含有关 traced 消息的一些额外详情。例如，Camel 现在捕获包含输入和输出（如果为 InOut）消息的 *第一个和最后一个* trace。

3.2.15. XML 序列化

使用 ModelToXMLDumper 的默认 xml 序列化已被改进，现在使用 camel-xml-io 模块中的生成的 xml serializer，而不是从 camel-jaxb 的一个 JAXB。

3.2.16. OpenAPI Maven 插件

camel-restdsl-openapi-plugin Maven 插件现在使用 platform-http 作为生成的 Rest DSL 代码中的默认其余组件，因为它是一个更好的默认设置，它与 Quarkus 一起工作。

3.2.17. 组件更改

3.2.17.1. 类别

org.apache.camel.Category 的枚举数量已从 83 减少到 37，这意味着使用删除的值的自定义组件需要选择一个剩余的值。我们这样做是为了整合 Camel 社区中所有组件的类别。

3.2.17.2. camel-openapi-rest-dsl-generator

此 dsl-generator 已将底层模型类(apicurio-data-models)从 1.1.27 更新至 2.0.3。

3.2.17.3. camel-atom

camel-atom 组件已将 Apache Abdera 的第三方从 Apache Abdera 改为 RSSReader。这意味着源对象已从 org.apache.abdera.model.Feed 改为 com.apptasticsoftware.rssreader.Item。

3.2.17.4. camel-azure-cosmosdb

itemPartitionKey 已更新。现在，一个字符串 a not a PartitionKey。CAMEL-19222 中的更多详细信息。

3.2.17.5. camel-bean

当使用 method 选项引用特定方法时，并使用参数类型和值，例如 ："bean:myBean?method=foo(com.foo.MyOrder, true) "， 任何类类型现在都必须使用 .class 语法，即 com.foo.MyOrder.MyOrder。

Example

```
"bean:myBean?method=foo(com.foo.MyOrder.class, true)"
```

这也适用于 Java 类型，如 String、int。

```
"bean:myBean?method=bar(String.class, int.class)"
```

3.2.17.6. camel-box

从 Box Java SDK v2 升级到 v4，其有一些方法签名更改。获取文件缩略图的方法不再可用。

3.2.17.7. camel-caffeine

keyType 参数已被删除。缓存的密钥现在仅是 String 类型。CAMEL-18877 中的更多信息。

3.2.17.8. camel-fhir

底层 hapi-fhir 库已从 4.2.0 升级到 6.2.4。只有 Delete API 方法已更改，现在返回 ca.uhn.fhir.rest.api.MethodOutcome 而不是 org.hl7.fhir.instance.model.api.IBaseOperationOutcome。有关底层更改的详细列表（只在 Camel 中使用 hapi-fhir 客户端，请参阅 hapi-fhir 客户端）。

3.2.17.9. camel-google

基于 API 的组件 camel-google-drive,camel-google-calendar,camel-google-sheets 和 camel-google-mail 已从 Google Java SDK v1 升级到 v2，以及最新的 API 修订。camel-google-drive 和 camel-google-sheets 有一些 API 方法更改，但其他方法与之前相同。

3.2.17.10. camel-http

组件已升级至使用 Apache HttpComponents v5，这会影响底层客户端的配置方式。有 4 个不同的超时(connectionRequestTimeout,connectTimeout,soTimeout , soTimeout,)而不是最初 3 (connectionRequestTimeout,connectTimeout, 和 socketTimeout)，它们的默认值已更改。详情请参考文档。

请注意，socketTimeout 已从 HttpClient 的可能配置参数中删除，改为使用 responseTimeout。

最后，选项 soTimeout 以及 SocketConfig 中包含的任何参数，需要以 httpConnection. 前缀。（包括定义到 HttpClientBuilder 和 RequestConfig 的参数等）需要加上 httpClient 前缀。

3.2.17.11. camel-http-common

org.apache.camel.http.common.HttpBinding 中的 API 稍微更改为可重复利用。parseBody 方法现在使用 HttpServletRequest 作为输入参数。所有 HttpMessage 已更改为通用 消息类型。

3.2.17.12. camel-kubernetes

io.fabric8:kubernetes-client 库已被升级，一些已弃用的 API 用量已被删除。之前带有 replace 前缀的操作现在带有 update 前缀。

例如，replaceConfigMap 现在是 updateConfigMap，replacePod 现在为 updatePod 等。类 KubernetesOperations 中的对应常量也被重命名。REPLACE_CONFIGMAP_OPERATION 现在是 UPDATE_CONFIGMAP_OPERATION，REPLACE_POD_OPERATION 现在为 UPDATE_POD_OPERATION 等。

3.2.17.13. camel-web3j

camel-web3j 将 web3j JAR 从 3.x 升级到 5.0，它有许多 API 更改，因此一些以前的 API 调用不会被提供。

3.2.17.14. camel-main

以下常数已从 BaseMainSupport / Main to Main Constants 移动：

旧名称	新名称
Main.DEFAULT_PROPERTY_PLACEHOLDER_LOCATION	MainConstants.DEFAULT_PROPERTY_PLACEHOLDER_LOCATION
Main.INITIAL_PROPERTIES_LOCATION	MainConstants.INITIAL_PROPERTIES_LOCATION
Main.OVERRIDE_PROPERTIES_LOCATION	MainConstants.OVERRIDE_PROPERTIES_LOCATION
Main.PROPERTY_PLACEHOLDER_LOCATION	MainConstants.PROPERTY_PLACEHOLDER_LOCATION

3.2.17.15. camel-micrometer

指标已被重命名为遵循 Micrometer 命名约定。

旧名称	新名称
CamelExchangeEventNotifier	camel.exchange.event.notifier
CamelExchangesFailed	camel.exchanges.failed
CamelExchangesFailuresHandled	camel.exchanges.failures.handled
CamelExchangesInflight	camel.exchanges.external.redeliveries
CamelExchangesSucceeded	camel.exchanges.succeeded
CamelExchangesTotal	camel.exchanges.total
CamelMessageHistory	camel.message.history
CamelRoutePolicy	camel.route.policy

CamelRoutePolicyLongTask	camel.route.policy.long.task
CamelRoutesAdded	camel.routes.added
CamelRoutesRunning	camel.routes.running

3.2.17.16. camel-jbang

camel 依赖项 命令已被重命名为 camel 依赖项。

在 Camel JBang 中，init 和 run 目标的 -dir 参数已被重命名为需要 2 dashes --dir，与所有其他选项一样。

camel stop 命令现在默认停止所有正在运行的集成（删除了 --all 选项）。

Placeholders 替换 被修改为使用 #name 而不是 $name 语法。

3.2.17.17. camel-openapi-java

camel-openapi-java 组件已更改为使用 io.swagger.v3 库，而不是 io.apicurio.datamodels。因此，公共方法 org.apache.camel.openapi.RestOpenApiReader.read （）的返回类型是 io.swagger.v3.oas.models.OpenAPI，而不是 io.apicurio.datamodels.openapi.models.OasDocument。当解析 OpenAPI 2.0 (swagger)规格时，它由 swagger parser 自动升级到 OpenAPI 3.0.x。此版本还支持 OpenAPI 3.1.x 规格。

3.2.17.18. camel-optaplanner

camel-optaplanner 组件已改为使用 SolverManager。如果您在 Camel 3 中使用 SoverManager，则不需要在 Route 中使用布尔值 useSolverManager。弃用的 ProblemFactChange 已被 ProblemChange 替代。

新 URI 路径为：

```
from("optaplanner:myProblemName")
  .to("...")
```

您可以通过两种方式传递 Optaplanner SolverManager：

- 以 #parameter 的身份
- 作为标头

在 Quarkus 上运行 camel-optaplanner 时，请使用创建 SolverManager 的 Quarkus 方法。

您可以通过提供 XML 配置文件来迁移旧的 Camel Optaplanner 路由，它允许 Camel Optaplanner 处理为这些旧路由创建 SolverManager，具体如以下代码中所示：

提供 Optaplanner Routes XML 配置文件

```
from("optaplanner:myProblemName?configFile=PATH/TO/CONFIG.FILE.xml")
  .to("...")
```

注意

solver Daemon 解决方案应该迁移到使用 SolverManager。

3.2.17.19. camel-platform-http-vertx

如果路由或消费者被暂停，则 http status 503 现在会返回，而不是 404。

3.2.17.20. camel-salesforce

生成的 DTOs 上的 blob 字段的属性名称不再有 'Url' affixed。例如，ContentVersionUrl 属性现在是 ContentVersion。

3.2.17.21. camel-slack

默认延迟（在 slack 消费者上）从 0.5s 改为 10s，以避免被 Slack 的速率限制。

3.2.17.22. camel-micrometer-starter

uri 标签现在是静态的，而不是动态标签（默认情况下，因为 URI 使用动态值生成的标签太多）。这可以通过设置 camel.metrics.uriTagDynamic=true 来再次启用。

3.2.17.23. camel-platform-http-starter

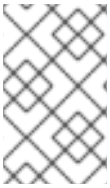
platform-http-starter 已从使用 camel-servlet 改为直接使用 HTTP 服务器。因此，所有 HTTP 端点不再以 servlet context-path 前缀（默认为 camel）。

例如：

HTTP 端点

```
from("platform-http:myservice")
  .to("...")
```

端点可以使用 http://localhost:8080/myservice 调用，因为没有使用 context-path。



注意

platform-http-starter 也可以用于 Rest DSL。

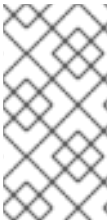
如果路由或消费者被暂停，则 http status 503 现在会返回，而不是 404。

3.2.17.24. camel-twitter

camel-twitter 组件已更新为使用 Twitter4j 版本 4.1.2，它已移动了几个类使用 的软件包。如果访问某些与二者相关的数据，如 Twit 状态，您需要将从 twitter4j.Status 中使用的软件包更新为 twitter4j.v1.Status。

3.3. 迁移到 APACHE CAMEL 3

本指南提供有关从 Red Hat Fuse 7 迁移到 Camel 3 的信息



注意

在组件（如模块化和 XML 架构更改）中，Fuse 7 和 Camel 3 之间有重要的区别。详情请查看每个组件部分。

红帽构建的 Apache Camel for Quarkus 支持 Camel 版本 4。本节提供了在将 Red Hat Fuse 7 应用程序迁移到带有 Camel 版本 3 的 Red Hat build of Apache Camel for Quarkus 时升级 Camel 的信息。

3.3.1. Java 版本

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

3.3.1.1. JDK 11 中删除了 JAXB

在 Java 11 中，JAXB 模块已从 JDK 中删除，因此您需要将它们添加为 Maven 依赖项（如果使用 JAXB 时，如使用 XML DSL 或 camel-jaxb 组件）：

```
<dependency>
  <groupId>javax.xml.bind</groupId>
  <artifactId>jaxb-api</artifactId>
  <version>2.3.1</version>
</dependency>

<dependency>
  <groupId>com.sun.xml.bind</groupId>
  <artifactId>jaxb-core</artifactId>
  <version>2.3.0.1</version>
</dependency>

<dependency>
  <groupId>com.sun.xml.bind</groupId>
  <artifactId>jaxb-impl</artifactId>
  <version>2.3.2</version>
</dependency>
```

注意

Java Platform Standard Edition 11 Development Kit (JDK 11)在 Camel 3.x 版本中已弃用，在发行版本 4.x 中不被支持。

3.3.2. 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。

3.3.3. 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*

3.3.4. 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。

3.3.5. 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。

3.3.5.1. 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：

camel-http , camel-hdfs , camel-mina , camel-mongodb , camel-netty , camel-netty-http , camel-quartz , camel-restlet , camel-rx , camel-jibx , camel-boon dataformat , camel-linkedin

Linkedin API 不再被支持。

camel-zookeeper

组件路由策略功能已删除。使用 ZooKeeperClusterService 或 camel-zookeeper-master。

camel-jetty

不再支持 制作者 （已被删除）。改为使用 camel-http 组件。

Twitter-streaming

删除，因为它依赖于已弃用的 Twitter Streaming API，且无法正常工作。

3.3.5.2. 重命名组件

在 Camel 3.x 中重命名以下组件。

camel-microprofile-metrics

重命名为 camel-micrometer

test

重命名为 dataset-test，并从 camel-core 移到 camel-dataset JAR。

http4

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

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

重命名为 mina，它对应于从 org.apache.camel.component.mina2 到 org.apache.camel.component.mina 的组件软件包。现在支持的方案是 mina。

mongodb3

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

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

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

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

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。

camel-jetty9

重命名为 camel-jetty。现在，支持的方案是 jetty。

3.3.6. Camel 组件的更改

3.3.6.1. 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。

3.3.6.2. 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。

3.3.6.3. 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特定的依赖项。

3.3.6.4. Camel CXF

camel-cxf JAR 已分为 SOAP 与 REST。从 camel-cxf 迁移时，我们建议您从以下列表中选择特定的 JAR。

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

例如，如果您在 SOAP 中使用 CXF，则在从 camel-cxf 进行迁移时选择 camel-cxf-soap 和 camel-cxf-transport。

3.3.6.4.1. Camel CXF 更改命名空间

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。

3.3.6.5. 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，则必须明确设置它。

3.3.6.6. 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。

3.3.6.7. 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
```

3.3.6.8. 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);
```

3.3.6.9. 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。

3.3.6.10. XML DSL 迁移

XML DSL 稍微改变。

自定义负载均衡器 EIP 已从 < custom> 改为 < 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>。

3.3.7. 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。

第 4 章 迁移 CAMEL QUARKUS 项目

4.1. 将项目更新至最新的 QUARKUS 版本

我们建议您使用 Maven 更新项目，并将项目升级到最新的 Quarkus 版本。

重要

对于使用 Hibernate ORM 或 Hibernate Reactive 的项目，请查看 Hibernate ORM 5 到 6 的迁移 快速参考。以下 update 命令只涵盖本指南的子集。

4.1.1. 先决条件

- 大约 30 分钟
- 正确安装了 JAVA_HOME 的 JDK
- Apache Maven 3.8.6
- 另外，如果要使用 Quarkus CLI
- 基于 Camel Quarkus 版本 2.13 或更高版本的项目。

4.1.2. 使用 Maven 更新

1. 配置扩展 registry 客户端，如 Quarkus Getting Started 指南中的 Configuring Quarkus extension registry client 部分所述。
2. 使用 Maven 更新：

   进入项目目录，将项目更新至最新的流：

   a. 确保 Quarkus Maven 插件版本与最新支持的 Red Hat build of Quarkus 版本一致。

b. 使用以下命令运行更新：

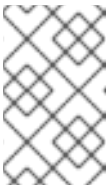
```
mvn io.quarkus.platform:quarkus-maven-plugin:3.8.4:update -N
```

对于多模块项目，始终首先尝试以下命令：

```
mvn io.quarkus.platform:quarkus-maven-plugin:3.8.4:update
```

如果这个命令失败，您可以尝试这个较长的命令：

```
find . -type f -name "pom.xml" -execdir sh -c 'mvn io.quarkus.platform:quarkus-maven-plugin:3.8.4:update -N' \;
```



注意

由于 OpenRewrite 存在问题，迁移日志中会预先设置警告。

选填

默认情况下，这个命令会更新到最新的当前版本。要更新到特定流而不是最新的当前版本，请在这个命令中添加 stream 选项，后跟版本；例如： -Dstream=3.2

注意

多模块项目的更新可能会显示很多错误，因为更新工具无法使用 <packaging> pom</packaging> 更新模块。

如果存在这些模块（通常包含版本），请手动更新它们。

3. 分析更新命令输出以了解潜在的指令，并根据需要执行建议的任务。

4. 使用 diff 工具来检查所有更改。

5. 查看 update 命令没有更新的项目的迁移指南。如果您的项目有这样的项目，请实施这些主

题中建议的额外步骤。

6. *在部署到生产之前，确保项目构建时没有错误，所有测试都通过应用功能。*

7. *在将更新的 Quarkus 应用程序部署到生产环境前，请确保以下内容：*

   - *项目构建时无错误。*
   - *所有测试都通过。*
   - *应用程序可以正常工作。*

第 5 章 其他资源

有关红帽构建的 Apache Camel for Quarkus 的更多信息，请参阅以下文档：

- Red Hat build of Apache Camel for Quarkus Extensions
- 红帽构建的 Apache Camel for Quarkus 入门
- 使用红帽构建的 Apache Camel for Quarkus 开发应用程序
- 将应用程序迁移到红帽构建的 Quarkus 版本 3.8