Red Hat Fuse 7.13

工具 Tutorials

如何在 CodeReady Studio 中使用 Fuse 工具的示例

Last Updated: 2024-07-01

Red Hat Fuse 7.13 工具 Tutorials

如何在 CodeReady Studio 中使用 Fuse 工具的示例

法律通告

摘要

本指南包含许多简单的教程，它们演示了如何使用红帽 Fuse 工具提供的工具来开发和测试应用程序。

目录

使开源包含更多 **4**

第 1 章 关于 FUSE 工具 TUTORIALS **5**
先决条件 5
FUSE 工具指南概述 5
关于示例应用程序 5
关于资源文件 6

第 2 章 设置您的环境 **7**
目标 7
开始前 7
创建 FUSE 集成项目 7
设置组件标签以显示 ID 值 13
下载项目的测试消息 14
查看测试信息 15
后续步骤 16

第 3 章 定义路由 **17**
目标 17
开始前 17
配置源端点 17
配置接收器端点 18
后续步骤 20

第 4 章 运行路由 **21**
目标 21
先决条件 21
运行路由 21
验证路由 22
后续步骤 23

第 5 章 添加基于内容的路由器 **24**
目标 24
先决条件 24
添加和配置基于内容的路由器 24
添加和配置日志记录 28
添加和配置消息标头 29
添加和配置分支以处理有效订购 32
验证 CBR 39
后续步骤 40

第 6 章 在路由上下文中添加另一个路由 **41**
目标 41
先决条件 41
配置现有路由的端点 41
添加第二个路由 42
配置选择分支以处理美国订购 43
配置其他有效分支来处理德国订购 49
验证第二个路由 54
后续步骤 58

第 7 章 调试路由上下文 **59**
目标 59

先决条件 59
设置断点 59
逐步浏览路由上下文 60
更改变量的值 65
缩小 CAMEL DEBUGGER 的重点 70
验证更改消息变量值的影响 72
后续步骤 73

第 8 章 通过路由追踪消息 74
目标 74
先决条件 74
设置 FUSE 集成视角 74
启动消息追踪 77
丢弃正在运行的 ZOOORDERAPP 项目的消息 81
配置消息视图 82
逐步浏览消息跟踪 84
后续步骤 87

第 9 章 使用 JUNIT 测试路由 88
概述 88
目标 88
先决条件 88
创建 SRC/TEST 文件夹 89
创建 JUNIT 测试案例 92
修改 BLUEPRINTXMLTEST 文件 97
修改 POM.XML 文件 101
运行 JUNIT 测试 102
进一步阅读 103
后续步骤 103

第 10 章 将项目发布到红帽 FUSE 104
目标 104
先决条件 104
定义 RED HAT FUSE SERVER 105
配置发布选项 110
连接到运行时服务器 115
卸载 ZOOORDERAPP 项目 117

使开源包含更多

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

第 1 章 关于 FUSE 工具 TUTORIALS

红帽 Fuse 工具指南为使用 Fuse 工具开发、运行、测试和部署 Apache Camel 应用程序提供实践介绍。

先决条件

开始前，您应该熟悉以下软件：

- Apache Camel
- Apache Maven

FUSE 工具指南概述

以下是教程的摘要以及您在每个教程中的内容：

- **第 2 章 *设置您的环境***
  创建 Fuse 集成项目并设置教程资源文件（消息和路由上下文文件示例）。当您创建项目时，它会自动创建路由上下文和一个初始路由。
- **第 3 章 *定义路由***
  定义简单路由的端点，该路由从文件夹检索消息并将其复制到另一个文件夹。
- **第 4 章 *运行路由***
  查看测试消息。运行路由，并通过查看测试消息从源文件夹复制到目标文件夹来验证它是否工作。
- **第 5 章 *添加基于内容的路由器***
  添加基于内容的路由器，以过滤消息并根据消息中的内容将其复制到不同的目标文件夹中。
- **第 6 章 *在路由上下文中添加另一个路由***
  添加另一个路由，以进一步过滤消息并根据消息中的内容将其复制到不同的目标文件夹中。
- **第 7 章 *调试路由上下文***
  使用 Camel debugger 设置断点，然后逐步调试路由，以检查路由和消息变量。
- **第 8 章 *通过路由追踪消息***
  将消息放到路由上，并通过所有路由节点跟踪它们。
- **第 9 章 *使用 JUnit 测试路由***
  为路由创建 JUnit 测试案例，然后测试路由。
- **第 10 章 *将项目发布到红帽 Fuse***
  逐步完成将 Apache Camel 项目发布到红帽 Fuse 的过程：定义本地服务器、配置发布选项、启动服务器、发布项目、连接到服务器，并验证项目已成功构建并发布。

有关 Fuse 工具功能的更多详细信息，请参阅 工具用户指南。

关于示例应用程序

您在 Fuse 工具指南中构建的示例应用程序模拟一个简单的订购应用程序，以便 zoos 订购 animals。提供了示例 XML 消息 - 每个 XML 消息包括客户信息(zoo 的名称、城市和国家/地区)和订单信息（请求的类型和数量，以及允许的最大异常数）。

使用 Fuse 工具，您可以创建一个获取传入示例消息的蓝图项目，根据其内容（评估与无效顺序）过滤它们，然后进一步按 zoo 的位置（计数）对有效顺序进行排序。在后面的教程中，您将使用示例应用调试路由上下文，通过路由跟踪消息，测试使用 JUnit 的路由，最后发布 Fuse 项目。

关于资源文件

每个教程都基于上一个教程。一个教程生成的代码是下一教程的起点，以便您可以按顺序完成教程。另外，在完成第一个教程后，您可以使用其中一个提供的上下文文件作为起点来按顺序执行任何其他教程。

教程依赖于 **Fuse-tooling-tutorials-jbds-10.3.zip** 文件中提供的资源文件，位于 此处。这个 zip 文件包含两个文件夹：

messages

此文件夹包含六个消息文件，名为 **message1.xml**,**message2.xml**, ... , **message6.xml**。在第一个教程 第 2 章 *设置您的环境* 中，您要创建保存这些消息文件的目录，同时查看其内容。所有教程都需要这些消息文件。

blueprintContexts

这个文件夹包含三个路由上下文文件：

- **Blueprint1.xml** - 这是通过完成 第 3 章 *定义路由* 教程生成的解决方案路由上下文。您可以使用它作为以下教程的起点：
  - 第 4 章 *运行路由*
  - 第 5 章 *添加基于内容的路由器*
- **Blueprint2.xml** - 这是 第 5 章 *添加基于内容的路由器* 教程的解决方案上下文文件。您可以使用 **blueprint2.xml** 作为 第 6 章 *在路由上下文中添加另一个路由* 教程的起点。
- **Blueprint3.xml** - 这是 第 6 章 *在路由上下文中添加另一个路由* 教程的解决方案上下文文件。您可以使用 **blueprint3.xml** 作为这些教程的起点：
  - 第 7 章 *调试路由上下文*
  - 第 8 章 *通过路由追踪消息*
  - 第 9 章 *使用 JUnit 测试路由*
  - 第 10 章 *将项目发布到红帽 Fuse*

第 2 章 设置您的环境

本教程介绍了创建 Fuse 集成项目的过程。该项目包含初始路由和默认的 CamelContext。路由是消息传输的处理器链。CamelContext 是一个单一路由规则基础，用于定义配置路由的上下文，并在端点（消息源和目标）之间消息交换过程中使用的策略。

您必须先完成本教程，然后才能遵循任何其他教程。

目标

在本教程中，您将完成以下任务：

- 创建 Fuse 集成项目
- 为您的项目下载测试消息(XML 文件)
- 查看测试信息

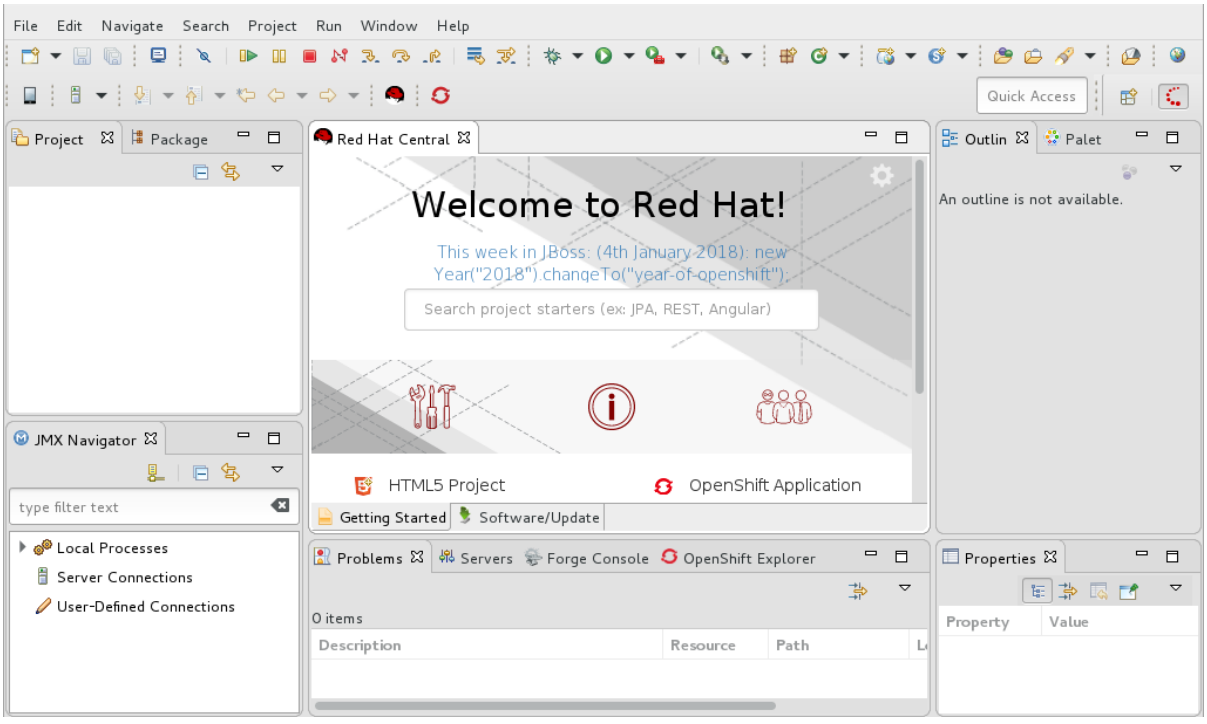
开始前

在设置 Fuse 集成项目前，您必须使用 Fuse 工具安装 Red Hat CodeReady Studio。有关如何安装 CodeReady Studio 的信息，请访问红帽客户门户 以获取您的平台安装指南。

在遵循 第 10 章 *将项目发布到红帽 Fuse* 指南中的步骤前，您必须安装 Java 8。

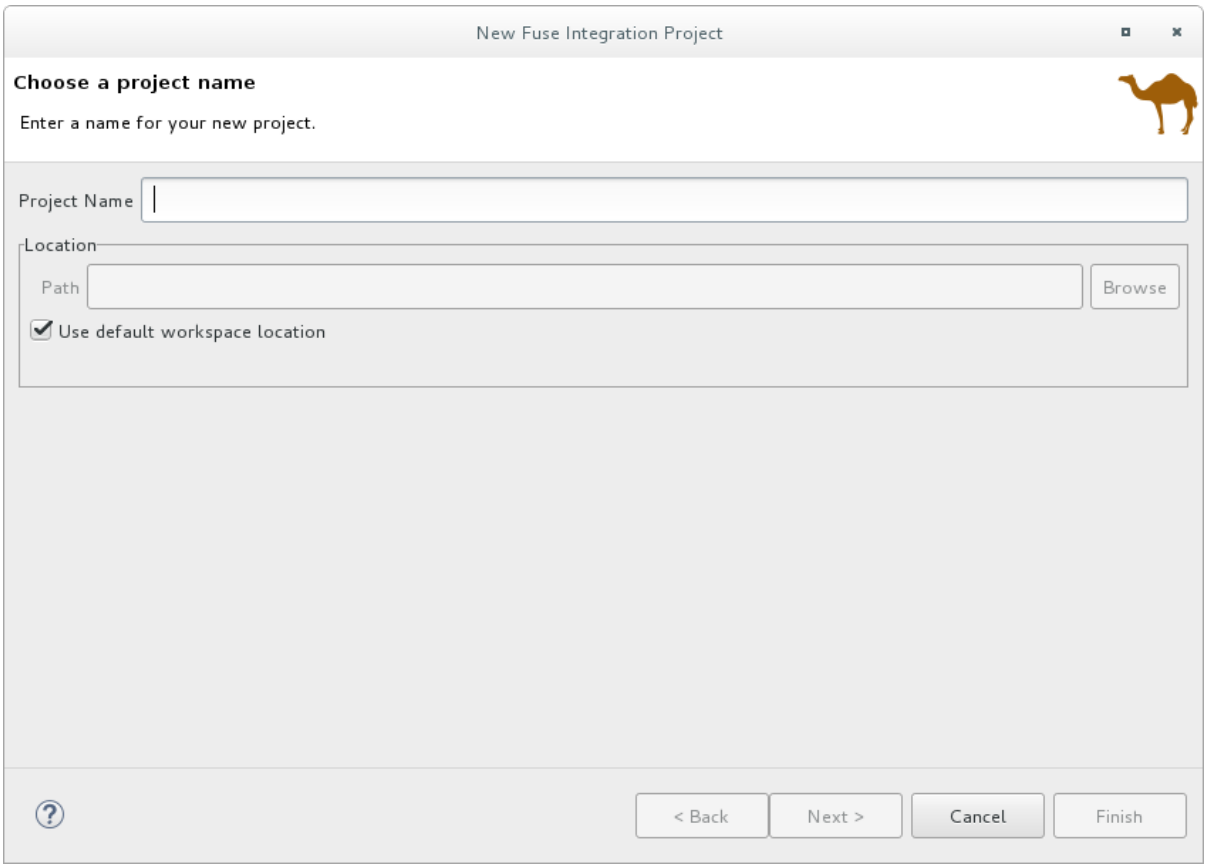
创建 FUSE 集成项目

1. 打开 Red Hat CodeReady Studio。
   首次启动 CodeReady Studio 时，它会在 **JBoss** 透视图中打开：



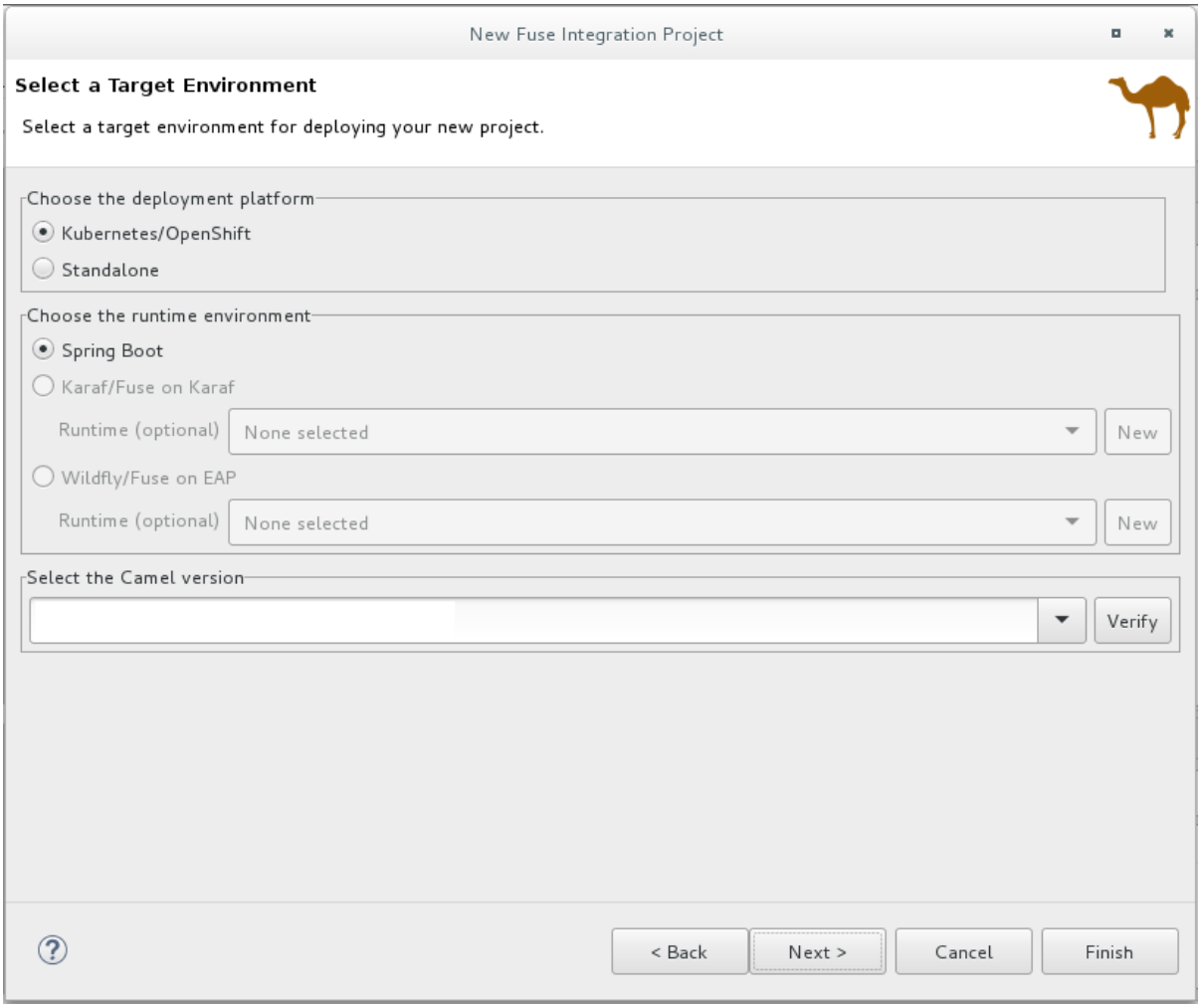
   否则，它会在之前 CodeReady Studio 会话中使用的视角中打开。

2. 在菜单中，选择 **File → New → Fuse Integration Project** 以打开 **New Fuse Integration Project** 向导：



3. 在 **Project Name** 字段中，输入 **ZooOrderApp**。
保留 **Use default 工作区位置** 选项被选择。

4. 点 **Next** 以打开 **Select a Target Runtime** 页面：



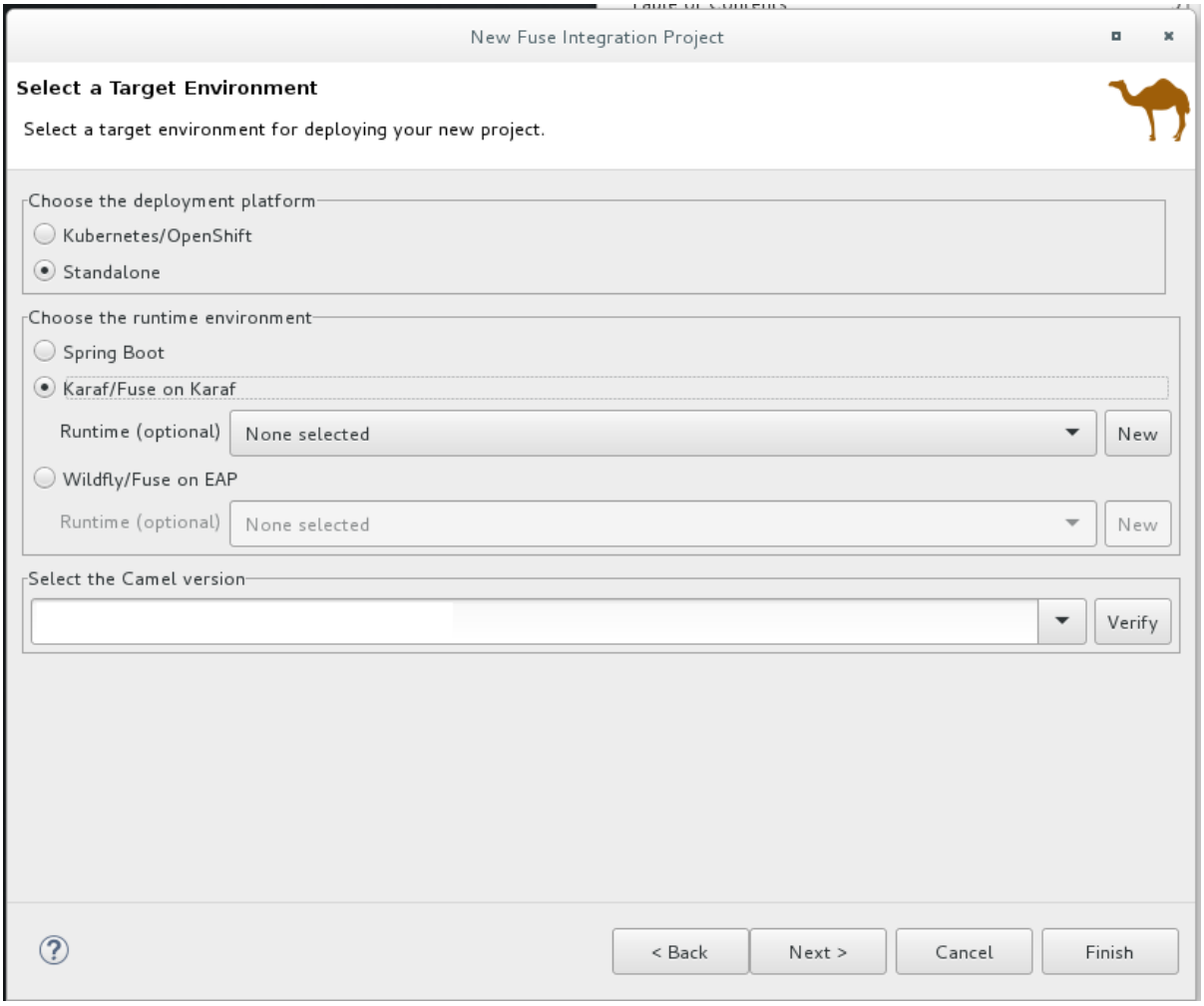
5. 为**部署平台选择独立**。

6. 选择 **Karaf/Fuse on Karaf** 并接受为运行时 **选择的 None**。

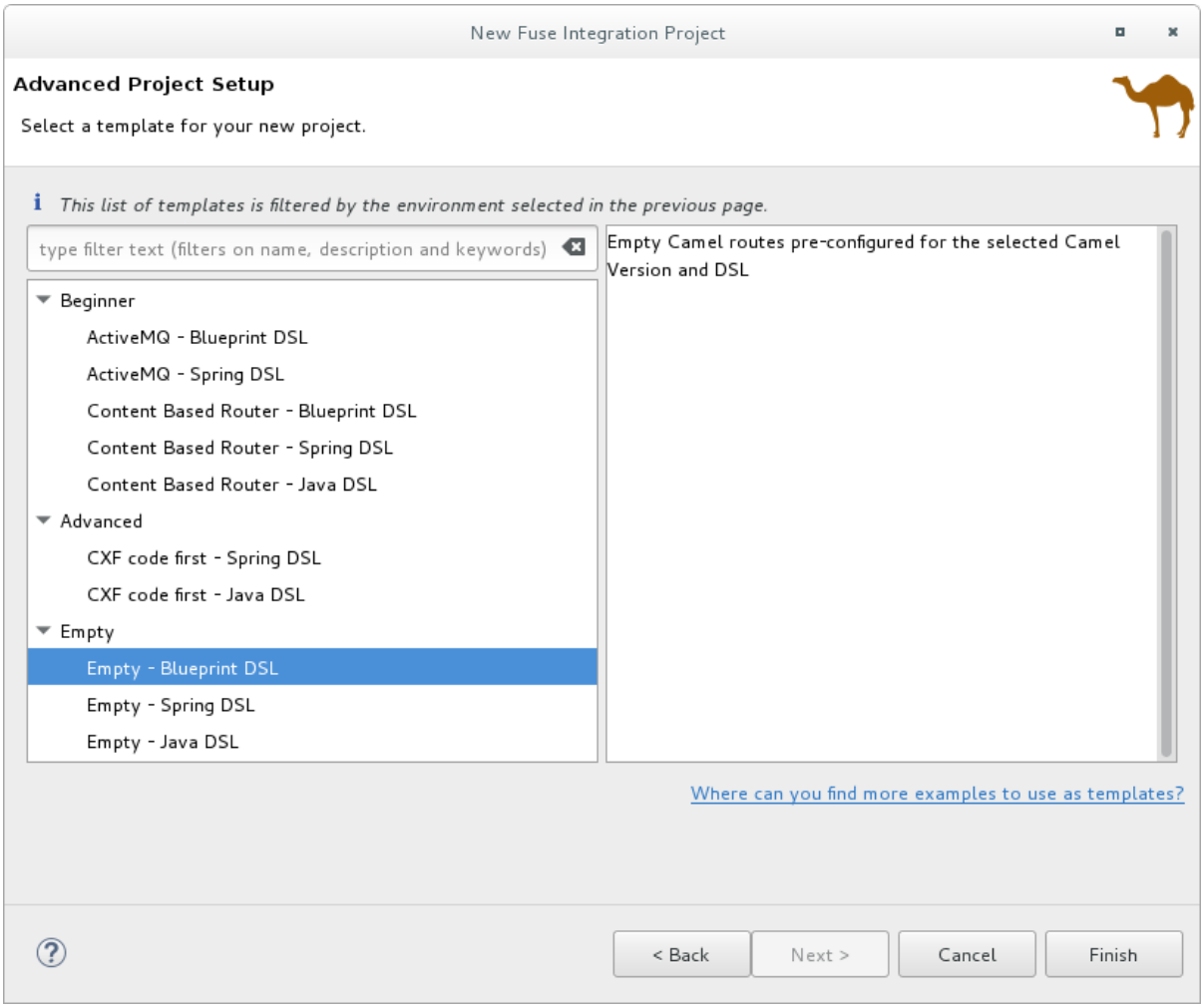
注意

您稍后会在 第 10 章 *将项目发布到红帽 Fuse* 指南中添加运行时。

7. 接受默认的 Apache **Camel 版本**。

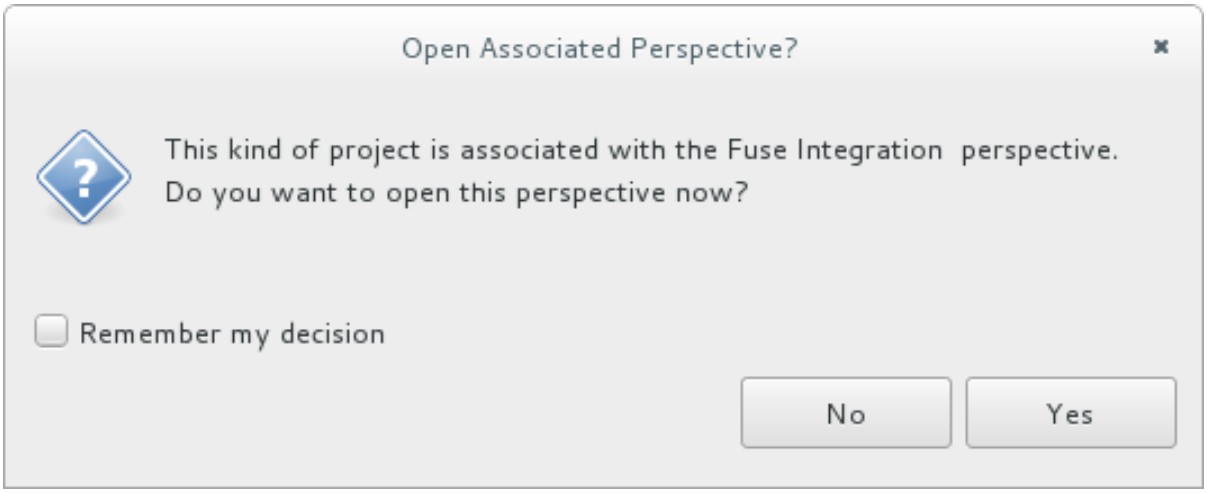


8. 点 **Next** 以打开 **Advanced Project Setup** 页面，然后选择 **Empty - Blueprint DSL** 模板：

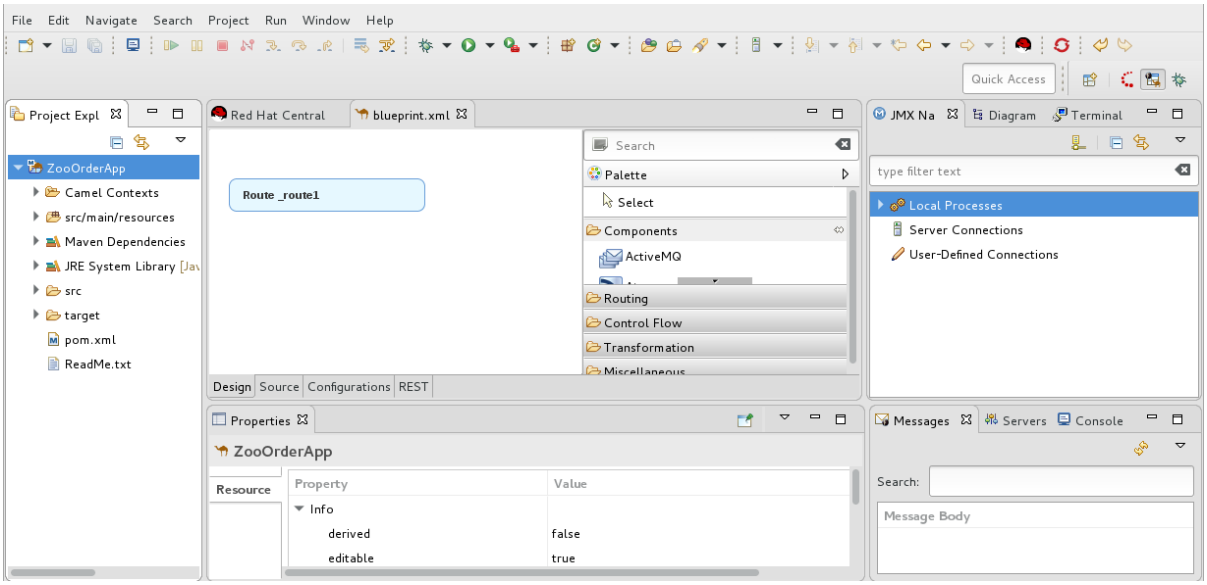


9. 点 **Finish**。
Fuse 工具开始从 Maven 存储库下载 - 所有需要构建项目的文件，然后将新项目添加到 **Project Explorer** 视图。

如果 CodeReady Studio 尚未显示 **Fuse Integration** 视角，它会询问您现在是否要切换到它：

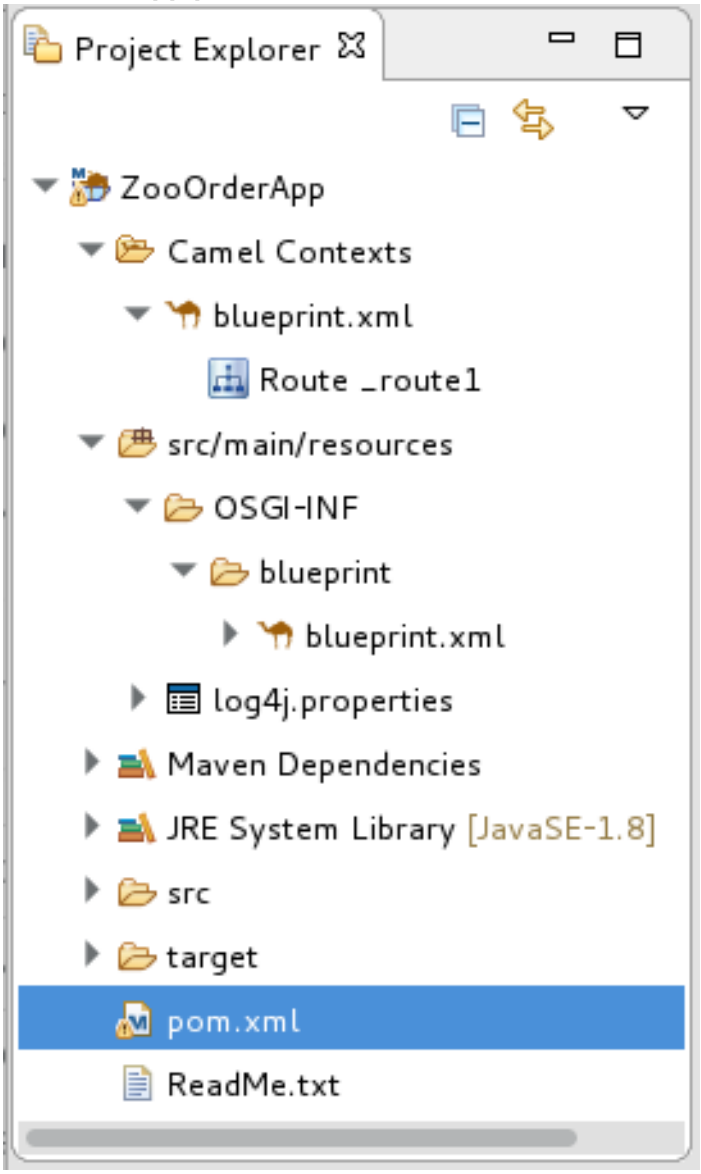


10. 单击 **Yes** 。
新的 **ZooOrderApp** 项目在 **Fuse Integration** 视角中打开：



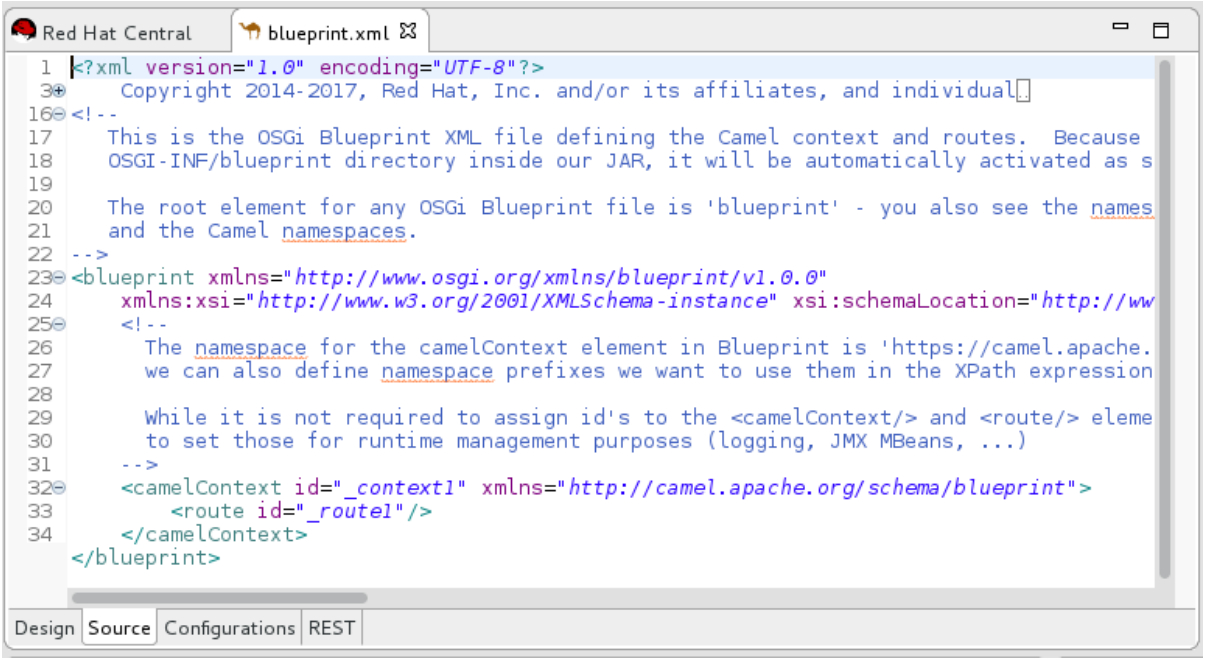
ZooOrderApp 项目包含创建和运行路由所需的所有文件，包括：

- **ZooOrderApp/pom.xml** HEKETI-wagonA Maven 项目文件。



- **ZooOrderApp/src/main/resources/OSGI-INF/blueprint/blueprint.xml** wagon-wagonA Blueprint XML 文件，其中包含 Camel 路由上下文和一个初始空路由。

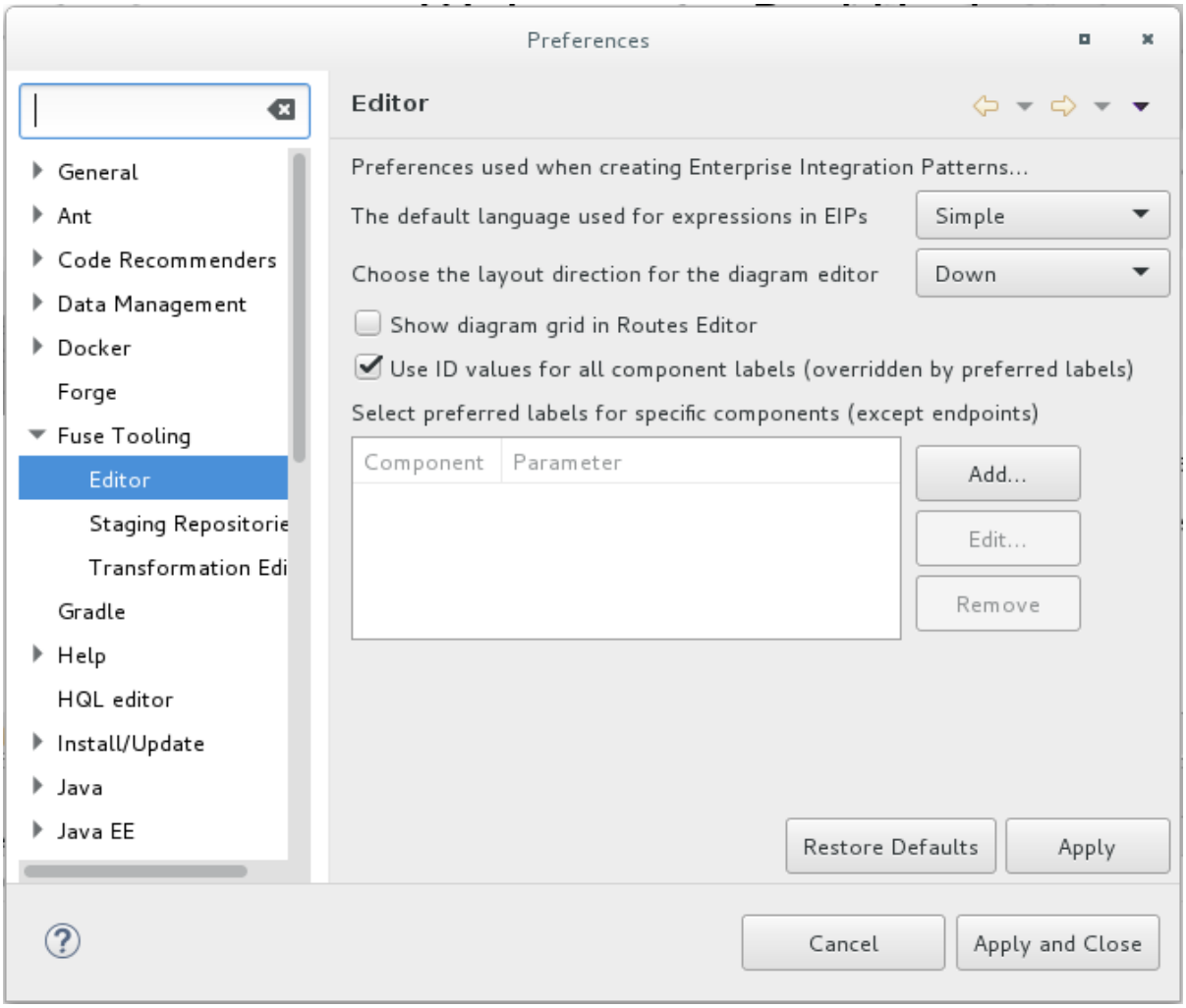
11. 要查看初始路由上下文，请在 Editor 视图中打开 **blueprint.xml** 文件，然后点 **Source** 选项卡。



设置组件标签以显示 ID 值

为确保在 Design canvas 上的模式和组件的标签与工具 Tutorials 中显示的标签相同：

1. 打开 Editor 首选项页面：
   - 在 Linux 和 Windows 机器上，选择 **Windows → Preferences → Fuse Tooling → Editor**。
   - 在 OS X 上，选择 **CodeReady Studio → Preferences → Fuse Tooling → Editor**。
2. 检查 **所有组件标签的 Use ID 值**。



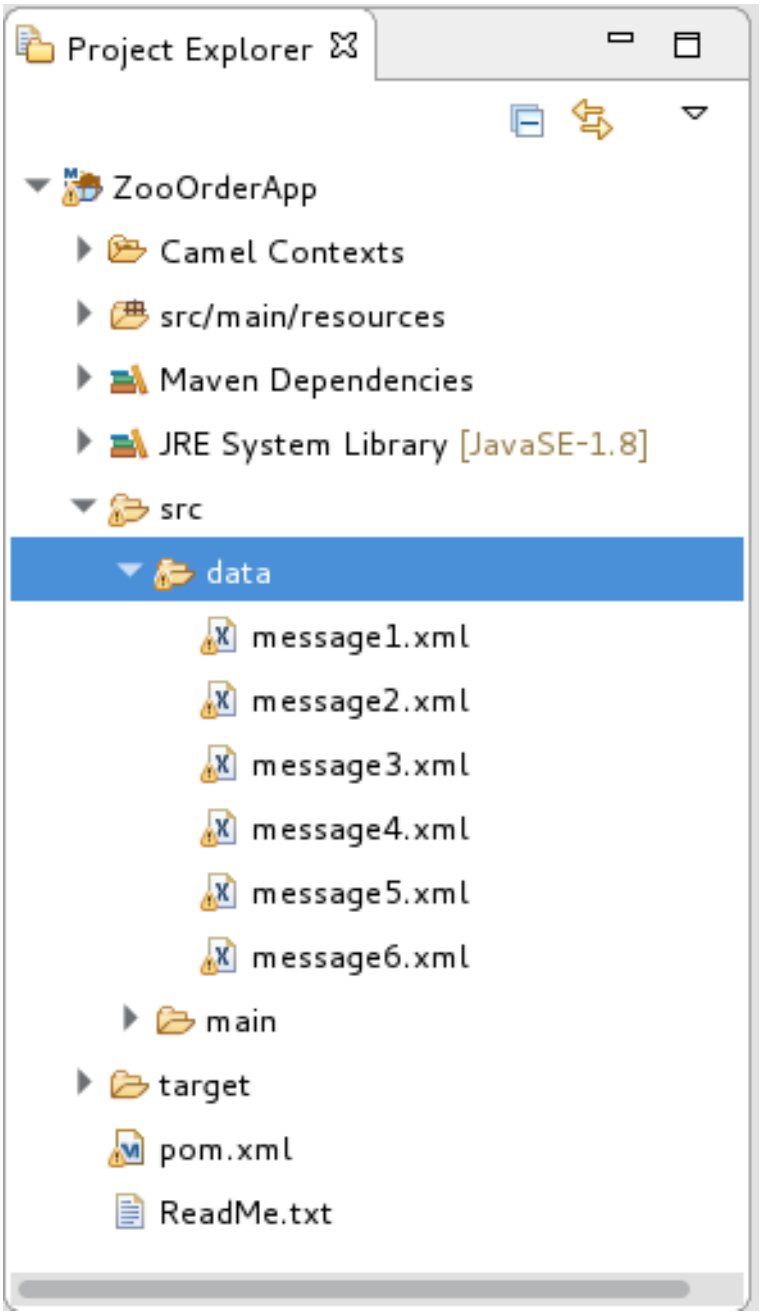
3. 单击**应用并关闭**。

下载项目的测试消息

提供了 XML 消息文件示例，以便您可以在使用工具 Tutorials 时测试 ZooOrderApp 项目。消息包含 zoo animals 的顺序信息。例如，为 Chicago zoo 的顺序是五个 wombats。

将提供的测试信息(XML 文件)下载到项目中：

1. 在 CodeReady Studio **Project Explorer** 视图中，创建一个文件夹来包含测试消息：
   a. 右键单击 **ZooOrderApp/src** 文件夹，然后选择 **New → Folder**。**New Folder** 向导将打开。
   b. 对于 **文件夹名称**，请键入 **data**。
   c. 点 **Finish**。
2. 点击此处 打开 Web 浏览器，进入提供的工具工具 Tutorial 资源 **Fuse-tooling-tutorials-jbds-10.3.zip** 文件的位置。
   将 **Fuse-tooling-tutorials-jbds-10.3.zip** 文件下载到 ZooOrderApp 项目工作区外部的便捷位置，然后将其解压缩。它包含两个文件夹，如 第 1 章 *关于 Fuse 工具 Tutorials* 所述。
3. 从 **messages** 文件夹，将六个 XML 文件复制到 **ZooOrderApp** 项目的 **src/data** 文件夹。



注意

您可以在 XML 文件中安全地忽略 。

查看测试信息

每个 XML 消息文件都包含来自 zoo （客户）数量的顺序。例如，'message1.xml' 文件包含来自 Brooklyn Zoo for 12 wombats 的顺序。

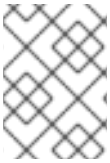
您可以在 **Editor** 视图中打开任何消息 XML 文件，以检查内容。

1. 在 **Project Explorer** 视图中，右键单击消息文件。
2. 在弹出菜单中选择 **Open**。
3. 点 **Source** 选项卡。
   XML 文件在 **Editor** 视图中打开。

例如，**message1.xml** 文件的内容显示来自 Bronx Zoo 为 12 个 wombats 的顺序：

```
<?xml version="1.0" encoding="UTF-8"?>

<order>
  <customer>
    <name>Bronx Zoo</name>
    <city>Bronx NY</city>
    <country>USA</country>
  </customer>
  <orderline>
    <animal>wombat</animal>
    <quantity>12</quantity>
  </orderline>
</order>
```



注意

您可以在新创建的 **message1.xml** 文件的第一行中安全地忽略  ，这建议您没有文档引用的 grammar 约束(DTD 或 XML Schema)。

下表提供了所有六个消息文件的内容概述：

表 2.1. 提供测试信息

msg#	<name>	<City>	<country>	<animal>	<quantity>
1	Bronx Zoo	Bronx NY	美国	Wombat	12
2	SAN Diego Zoo	SAN Diego CA	美国	giraffe	3
3	SEA Life Centre	Munich	德国	penguin	15
4	Berlin Zoo	Berlin	德国	emu	6
5	Philadelphia Zoo	Philapelphia PA	美国	giraffe	2
6	st Louis Zoo	st Loius MO	美国	penguin	10

后续步骤

现在，您已设置了 CodeReady Studio 项目，您可以继续使用 第 3 章 *定义路由* 指南来定义处理 XML 信息的路由。

第 3 章 定义路由

本教程介绍了在路由中添加和配置端点的步骤。端点定义源和接收器(sink)，用于通过路由传输的信息。对于 **ZooOrderApp** 项目，启动（源）端点是包含 XML 消息文件的文件夹。sink (finishing)端点是您在项目中指定的另一个文件夹。

目标

在本教程中，您将完成以下任务：

- 向路由添加源和接收器端点
- 配置端点
- 连接端点

开始前

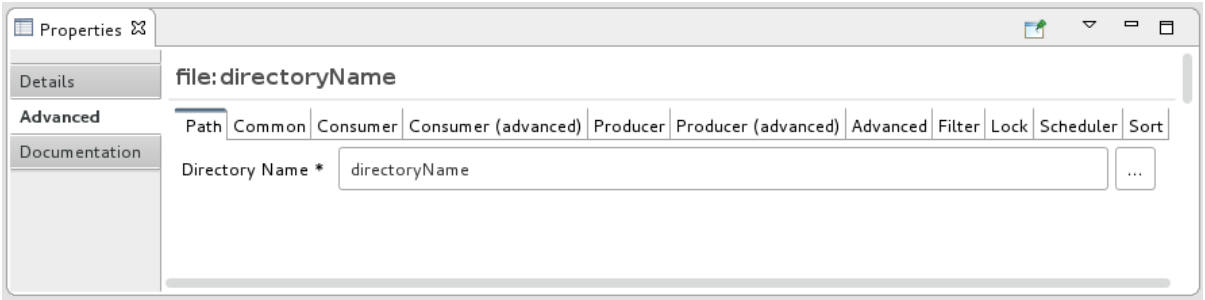
在开始本教程前：

1. 您必须设置工作区环境，如 第 2 章 *设置您的环境* 教程中所述。
2. 在 CodeReady Studio 中，在 **Editor** 视图中打开 **ZooOrderApp** 项目的 **/src/main/resources/OSGI-INF/blueprint/blueprint.xml** 文件。
3. 如果需要，请单击 **Editor** 视图底部的 **Design** 选项卡，以查看初始路由的图形显示，标记为 **Route_route1**。

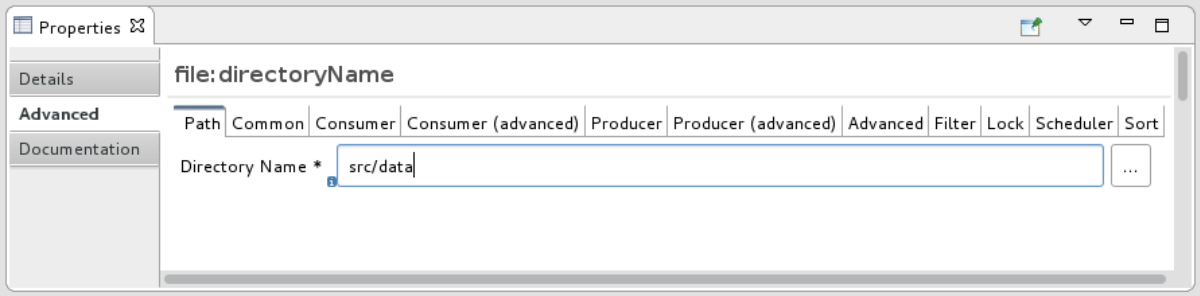
配置源端点

按照以下步骤将 **src/data** 文件夹配置为路由的源端点：

1. 将文件组件组件  拖到 canvas 中，并将它放到 **Route_route1** 容器节点中。
File 组件会更改为 Route_route1 容器节点内的 From _from1 节点。
2. **在 canvas 上，选择 From _from1 节点。**
Properties 视图位于 canvas 下，显示节点的属性字段进行编辑。
3. **要为消息文件指定源目录，在 Properties 视图中点 Advanced 选项卡：**



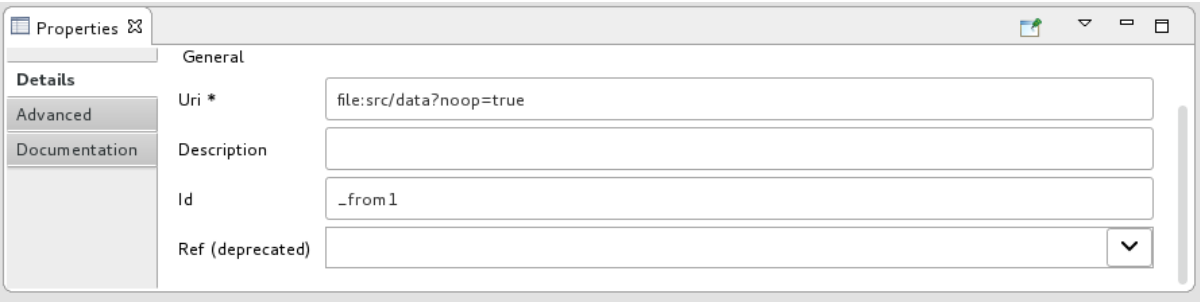
4. **在 Directory Name 字段中，输入 src/data ：**



路径 src/data 相对于项目的目录。

5. 在 Consumer 选项卡中，点 Noop 选项来启用 Noop 选项。
Noop 选项可防止消息 Serial.xml 文件从 src/data 文件夹中删除，并启用 idempotency 来确保每个 message#.xml 文件只消耗一次。

6. 选择 Details 选项卡，以打开文件节点的 Details 页面。
请注意，工具会自动使用您在 Advanced 选项卡上配置的 Directory Name 和 Noop 属性填充 Uri 字段。它还使用自动生成的 ID (_from1)填充 Id 字段：



注意

工具前缀使用下划线(_)自动生成 ID 值。您可以选择更改 ID 值。underscore 前缀不是一个要求。

将自动生成的 Id 保留为原样。

7. 选择 File → Save 保存路由。

配置接收器端点

添加并配置路由的 sink （目标）端点：

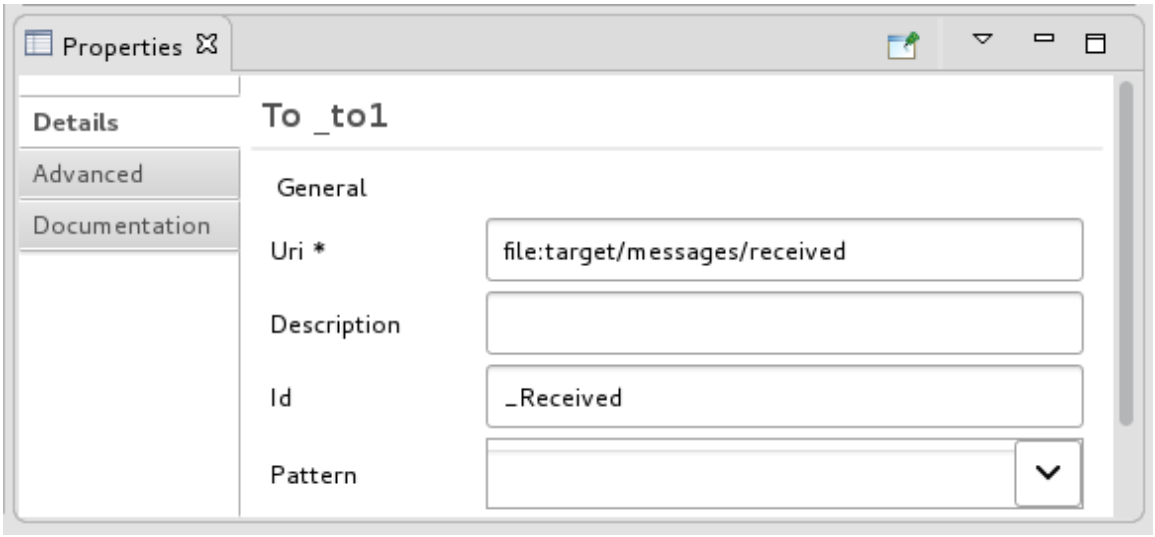
1. 将另一个文件 组件从 PrometheusRule 的Components drawer 拖放，并将它放到 Route_route1 容器节点上。
File 组件会更改为 Route_route1 容器节点内的 To_to1 节点。

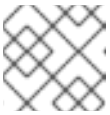
2. 在 canvas 上，选择 To_to1 节点。
Properties 视图位于 canvas 下，显示节点的属性字段进行编辑。

3. 在 Details 标签页中：

   a. 在 Uri 字段中，键入 file:target/messages/received。

   b. 在 Id 字段中，键入 _Received。

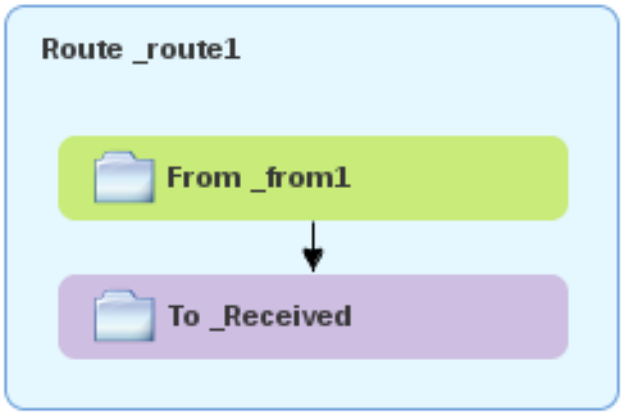




注意

该工具将在运行时创建 **target/messages/received** 文件夹。

4. **在 Route_route1 容器中，**选择 From _from1节点，并将其连接器箭头()拖到 To_Received 节点上，然后释放它：



注意

这两个文件节点已连接并一致，根据路由编辑器的布局首选项设置。选择是 Down（默认）和 Right。

要访问路由编辑器的布局首选项选项：

- 在 Linux 和 Windows 机器上，选择 Windows → Preferences → Fuse Tooling → Editor → 选择图编辑器的布局方向。
- 在 OS X 上，选择 CodeReady Studio → Preferences → Fuse Tooling → Editor → 选择图表编辑器的布局方向。

注意

如果在关闭项目前没有连接节点，工具会在重新打开时自动连接它们。

5. **保存 路由。**

6. **点击 canvas 底部的 Source 选项卡来显示路由的 XML：**

```
<?xml version="1.0" encoding="UTF-8"?>

<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.osgi.org/xmlns/blueprint/v1.0.0
    https://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd
    http://camel.apache.org/schema/blueprint
    http://camel.apache.org/schema/blueprint/camel-blueprint.xsd">

    <camelContext id="_context1" xmlns="http://camel.apache.org/schema/blueprint">
        <route id="_route1">
            <from id="_from1" uri="file:src/data?noop=true"/>
            <to id="_Received" uri="file:target/messages/received"/>
        </route>
    </camelContext>
</blueprint>
```

后续步骤

现在，您在路由中添加和配置了端点，您可以运行路由，如 第 4 章 *运行路由* 指南所述。

第 4 章 运行路由

本教程介绍了运行路由的过程，以验证路由是否已正确将信息从源端点传输到接收器端点。

目标

在本教程中，您将完成以下任务：

- 将路由作为本地 Camel 上下文运行（不带测试，因为您尚未设置测试）
- 通过路由发送消息
- 检查 sink 端点收到的消息，以确保路由正确处理测试信息

先决条件

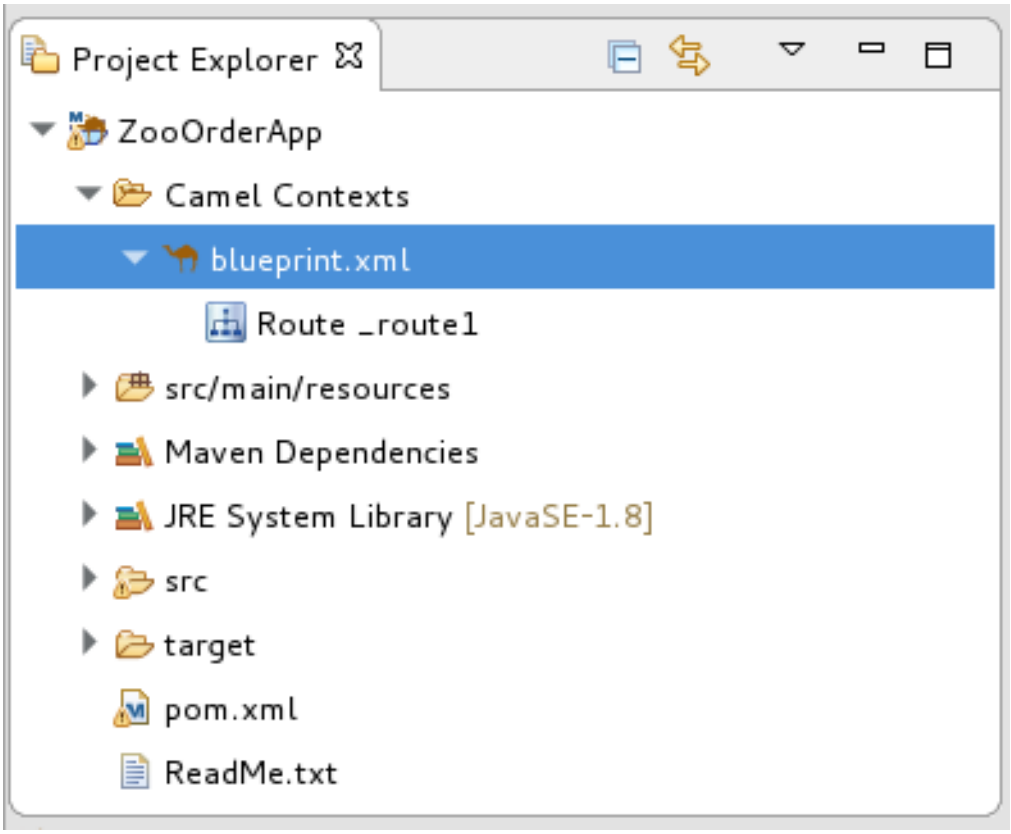
要启动本教程，您需要如下 ZooOrderApp 项目：

1. 完成 第 2 章 *设置您的环境* 指南。
2. 下面是其中之一：
   - 完成 第 3 章 *定义路由* 指南。
     or
   - 将项目的 blueprint.xml 文件替换为提供的 blueprintContexts/blueprint1.xml 文件，如 "关于资源文件"一节 所述。

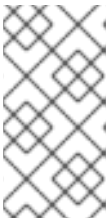
运行路由

运行路由：

1. 打开 ZooOrderApp 项目。
2. 在 Project Explorer 中，选择 ZooOrderApp/Camel Contexts/blueprint.xml ：



3. **右键单击 blueprint.xml，然后选择 Run As → Local Camel Context （无需测试）。**



注意

如果您选择 Local Camel Context，该工具会自动尝试针对提供的 JUnit 测试运行路由上下文。由于 JUnit 测试不存在，因此工具将恢复到在没有测试的情况下运行路由上下文。在 第 9 章 *使用 JUnit 测试路由* 教程中，您可以创建一个 JUnit 测试案例来测试 ZooOrderApp 项目。

Console 面板将打开，以显示反映项目执行进度的日志消息。开始时，Maven 下载更新本地 Maven 存储库所需的资源。Maven 下载过程可能需要几分钟时间。

4. **等待消息（与以下内容类似）在输出末尾出现：这些消息表示路由成功执行：**

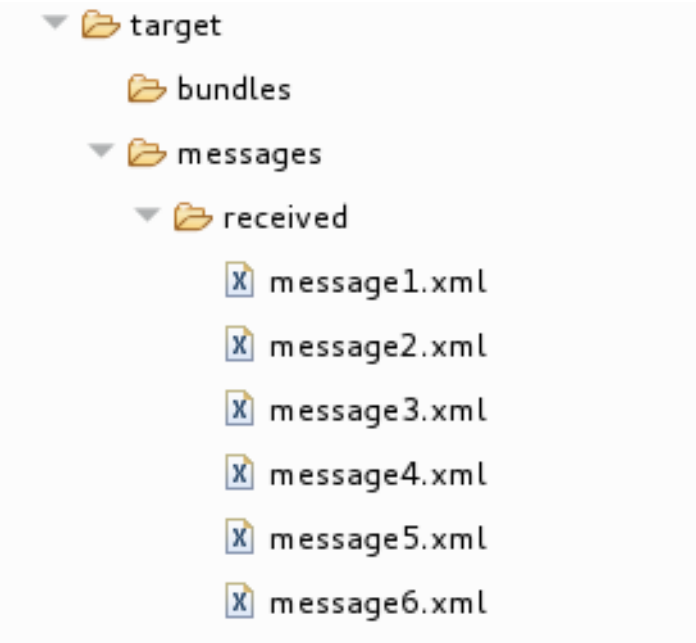
```
...
[Blueprint Event Dispatcher: 1] BlueprintCamelContext INFO  Route: _route1 started and
consuming from:Endpoint[file://src/data?noop=true]
[Blueprint Event Dispatcher: 1] BlueprintCamelContext INFO  Total 1 routes, of which 1 are
started.
[Blueprint Event Dispatcher: 1]BlueprintCamelContext INFO  Apache Camel 2.21.0.redhat-3
(CamelContext: ...) started in 0.163 seconds
[Blueprint Event Dispatcher: 1] BlueprintCamelContext INFO  Apache Camel 2.21.0.redhat-3
(CamelContext: ...) started in 0.918 seconds
```

5. **要关闭路由，请点击 Console 视图顶部的 。**

验证路由

要验证路由是否已正确执行，您可以检查消息 XML 文件是否从源文件夹(src/data)复制到目标文件夹(target/messages/received)。

1. 在 Project Explorer 中，选择 ZooOrderApp。
2. 右键单击，然后选择 Refresh。
3. 在 Project Explorer 中，找到 target/messages/ 文件夹并展开，以验证 target/messages/received 文件夹是否包含六个消息文件 message1.xml 到 message6.xml ：



4. 双击 message1.xml 在路由编辑器的 Design 选项卡中打开它，然后选择 Source 选项卡来查看 XML 代码：

```
<?xml version="1.0" encoding="UTF-8"?>

<order>
  <customer>
    <name>Bronx Zoo</name>
    <city>Bronx NY</city>
    <country>USA</country>
  </customer>
  <orderline>
    <animal>wombat</animal>
    <quantity>12</quantity>
  </orderline>
</order>
```

后续步骤

在 第 5 章 *添加基于内容的路由器* 教程中，您添加一个基于 Content-Based Router，它使用消息的内容来确定其目的地。

第 5 章 添加基于内容的路由器

本教程介绍了如何添加基于内容的路由器(CBR)并记录到路由。

CBR 根据内容将消息路由到目的地。在本教程中，根据每个消息 quantity 字段的值（按顺序数量）创建将消息路由到不同的文件夹(valid 或 invalid)的 CBR。每个顺序的最大 animals 值为 10。CBR 根据数量是否大于 10，将消息路由到不同的文件夹。例如，如果 zoo 订购五 zebras 并且只有三个 zebras 可用，则顺序将复制到无效的订购目标文件夹中。

目标

在本教程中，您将完成以下任务：

- 将基于内容的路由器添加到您的路由
- 配置基于内容的路由器：
  - 向基于内容的路由器的每个输出分支添加日志端点
  - 在每个日志端点后添加 Set Header EIP
  - 将 Otherwise 分支添加到基于内容的路由器

先决条件

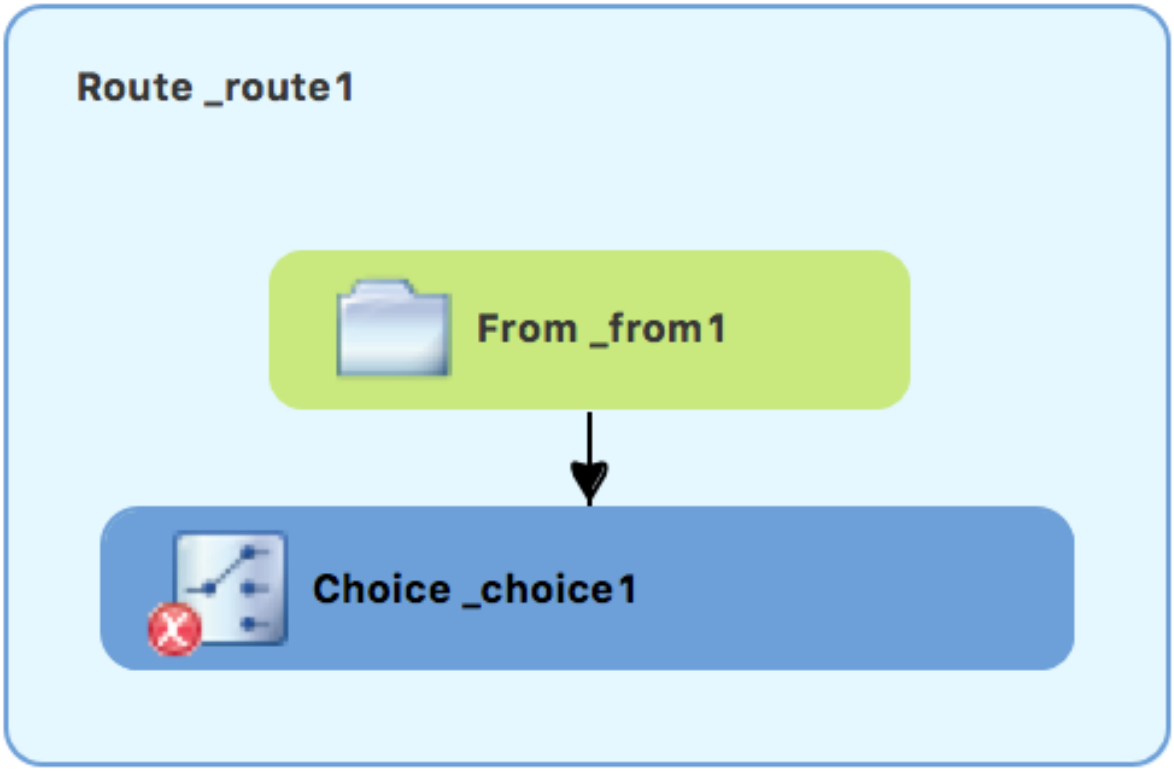
要启动本教程，您需要从以下之一生成的 ZooOrderApp 项目：

- 完成 第 4 章 *运行路由* 指南。
  or
- 完成 第 2 章 *设置您的环境* 指南，并使用提供的 **blueprintContexts/blueprint1.xml** 文件替换项目的 **blueprint.xml** 文件，如 “关于资源文件”一节 所述。

添加和配置基于内容的路由器

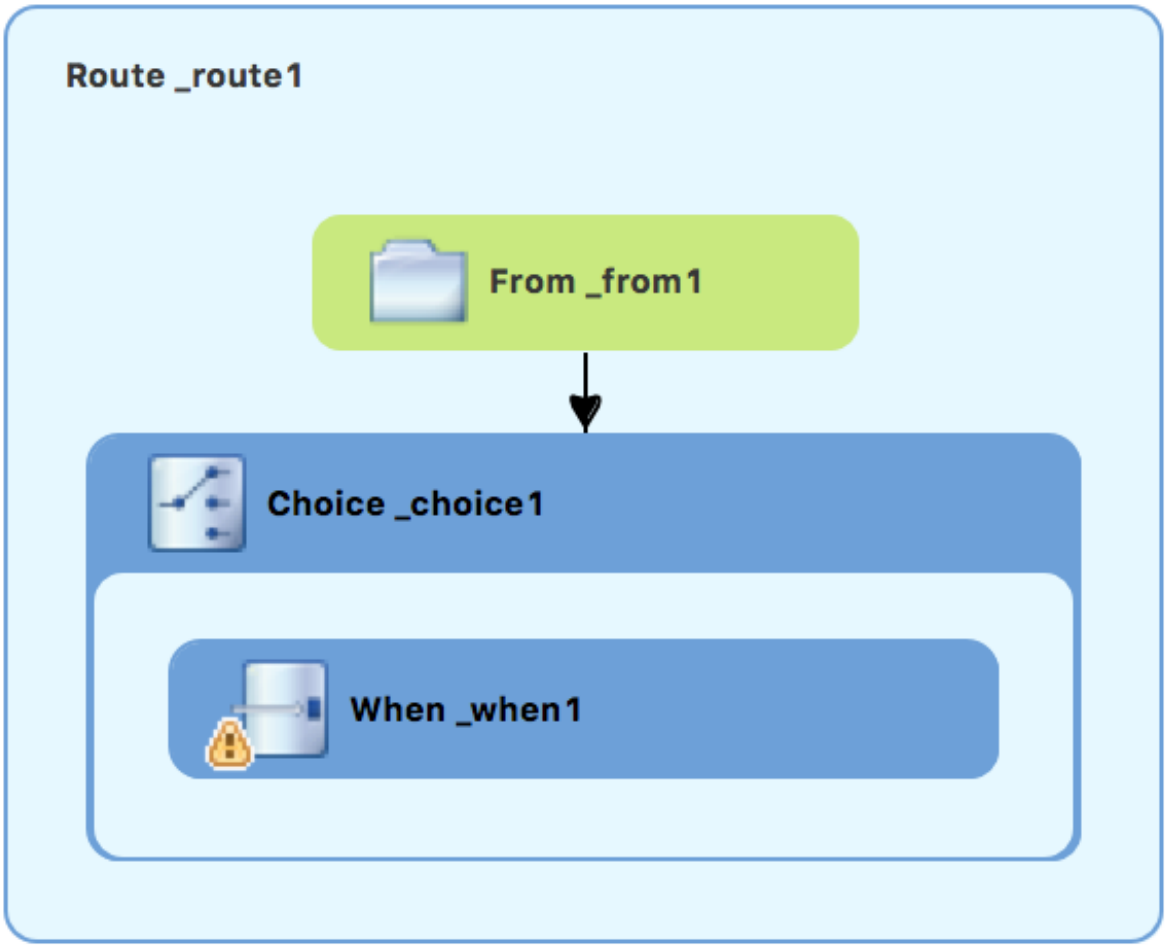
为您的路由添加和配置基于内容的路由器：

1. 在 Project Explorer 中，双击 **ZooOrderApp/src/main/resources/OSGI-INF/blueprint/blueprint.xml** 以在 Editor 视图中打开它。
2. 在 Design canvas 中，选择 **To_Received** 节点，然后选择回收站图标来删除它。
3. 在 Prod 中，打开 Routing drawer，单击 Choice ()模式，然后（在 Design canvas 中），单击 **From_from1** 节点。



Route_route1 容器扩展以容纳 Choice_choice1 节点。错误图标表示 Choice_choice1 节点需要您接下来添加的子节点。

4. 在 Routing drawer 中，点 When ( )模式，然后在 canvas 中点 Choice_choice1 节点。Choice_choice1 容器扩展以适应 When_when1 节点：



拒绝 When_when1 节点表示必须设置一个或多个所需的属性值。

注意

该工具可防止您将模式添加到 Route 容器中的无效点。

5. 在 canvas 上，选择 When_when1 节点，在 Properties 视图中打开其属性：

6. 点击 Expression 字段中的 按钮打开可用选项列表。

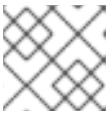
7. 选择 xpath （用于 XML 查询语言），因为测试消息使用 XML 编写。

注意

选择 Expression 语言后，Properties 视图会在 Expression 字段的缩进列表中直接显示其属性。此缩进列表中的 Id 属性设置表达式的 ID。Description 字段后面的 Id 属性设置 When 节点的 ID。

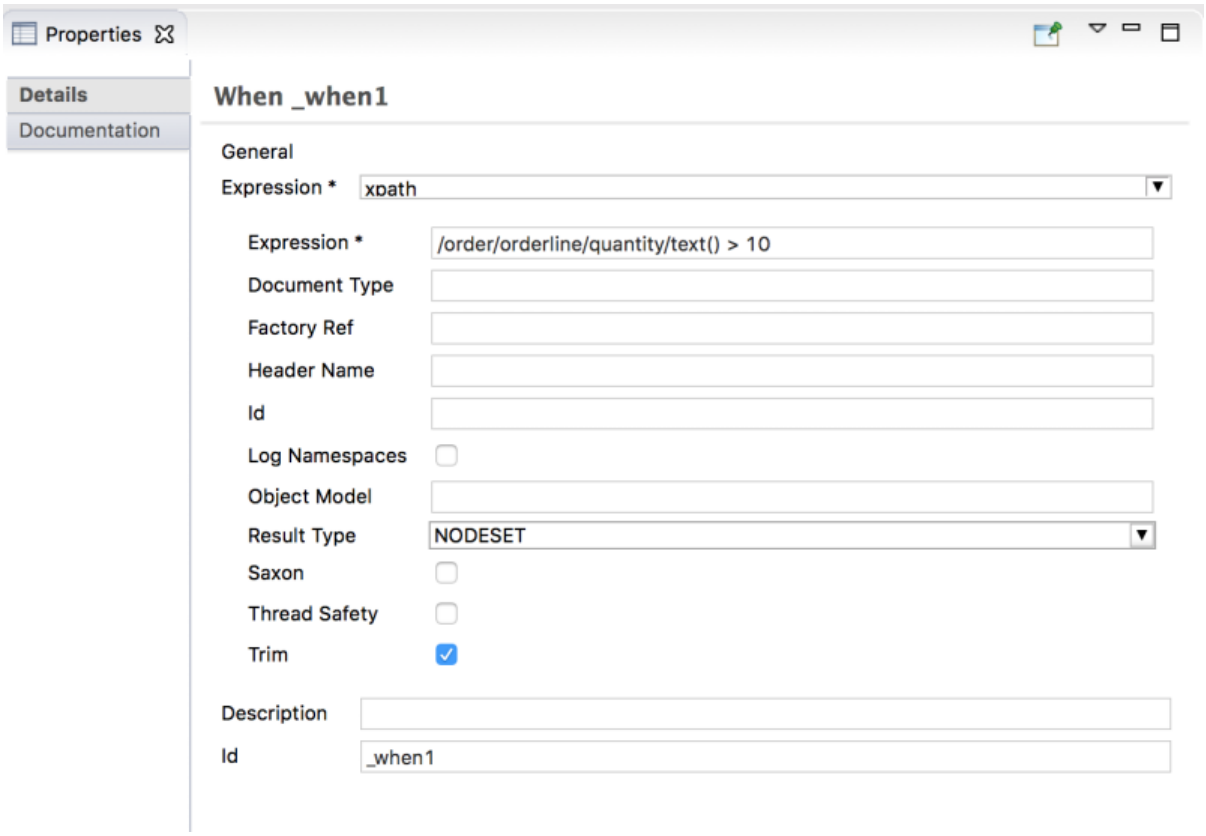
8. 在缩进 Expression 字段中，键入：/order/orderline/quantity/text （）> 10
此表达式指定，只有 quantity 字段的值大于路由中的 10 个传输此路径（到 invalidOrders 文件夹）的消息。

9. 保留每个剩余的属性。



注意

Trim 选项（默认启用）从消息中删除任何前导或尾随空格和换行符。



10. 保存 路由上下文文件。

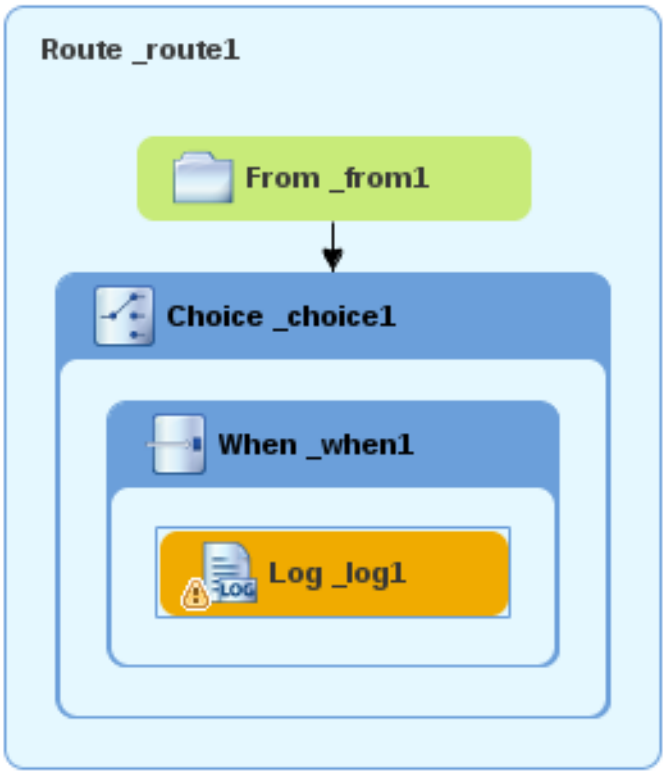
11. 点 Source 选项卡查看路由的 XML：

添加和配置日志记录

对于 ZooOrder 应用程序示例，您可以添加日志消息，以便在通过路由时跟踪 XML 消息。运行路由时，控制台 视图中会显示日志消息。

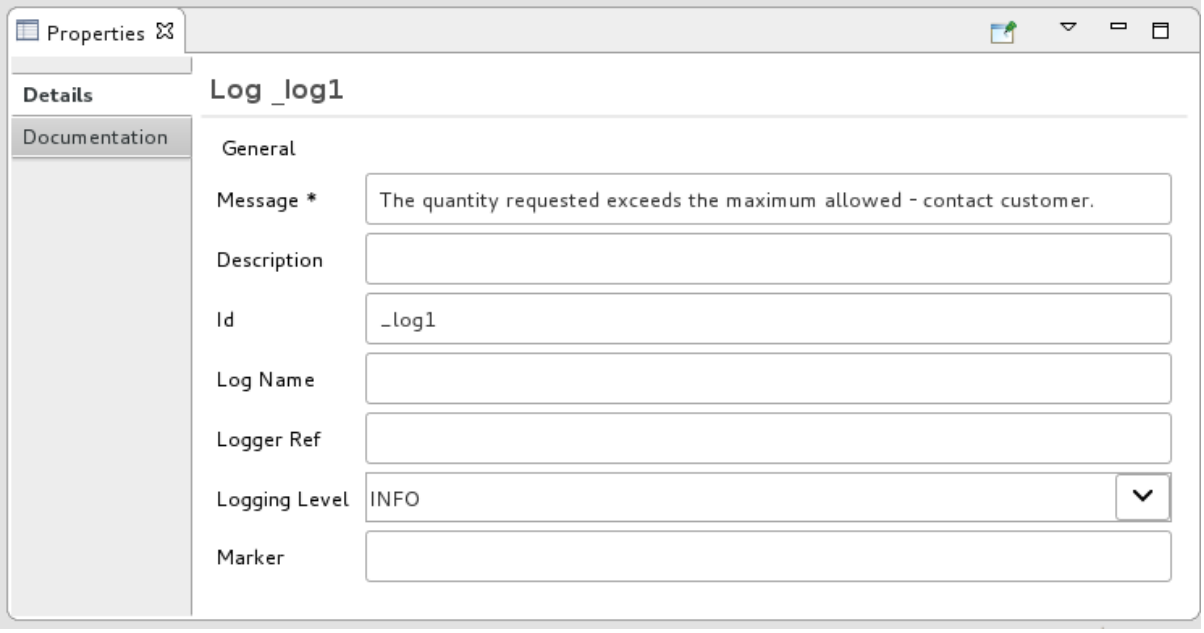
按照以下步骤将日志记录添加到 CBR 路由中：

1. **在 Design 选项卡中，打开 Components drawer，再单击 Log components (**  **)。**
2. **在 canvas 中，点 When_when1 节点。**
   When_when1 容器扩展以容纳 Log_log1 节点：



3. 在 canvas 上，选择 **Log_log1** 节点，以在 Properties 视图中打开其属性。

4. 在 Message 字段中，键入：请求的数量超过允许的最大 - 联系客户。



其余的属性保留原样。

+

注意

工具自动生成日志节点 id 值。在 Fuse Integration 视角的 Messages 视图中，工具会在 Trace NodeId 列中为消息实例插入日志节点 Id 字段的内容，在路由上启用了追踪（请参阅 第 8 章 *通过路由追踪消息* 指南）。在控制台中，每当路由运行时，它会将日志节点的 Message 字段的内容添加到日志数据中。

1. 保存 路由上下文文件。

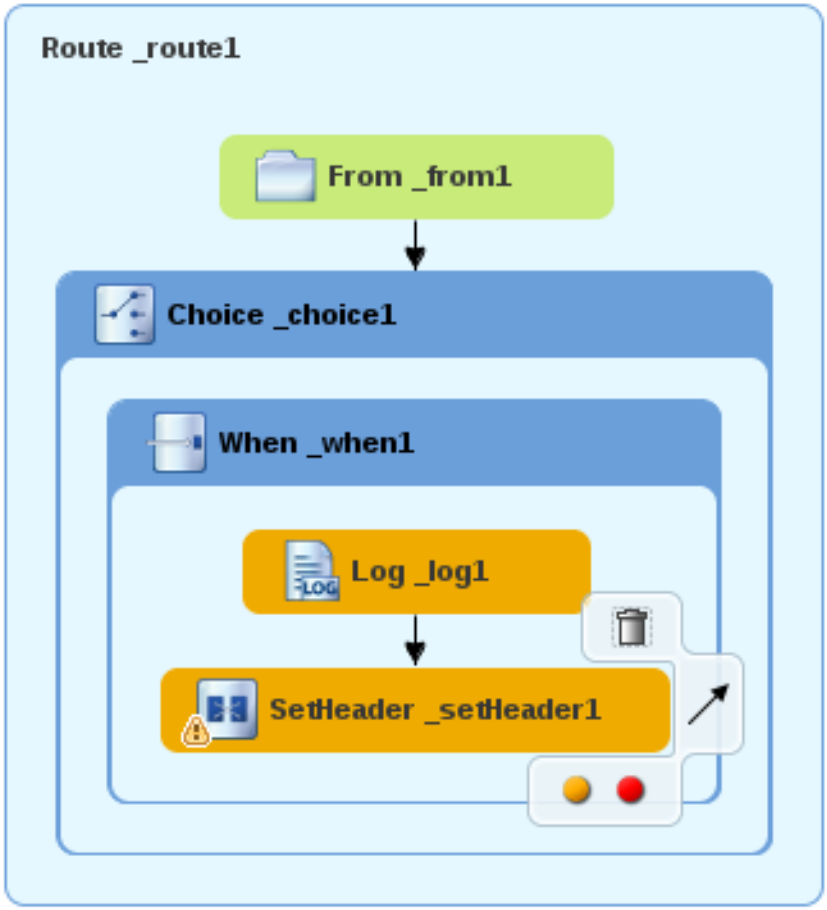
添加和配置消息标头

消息标头包含处理消息的信息。

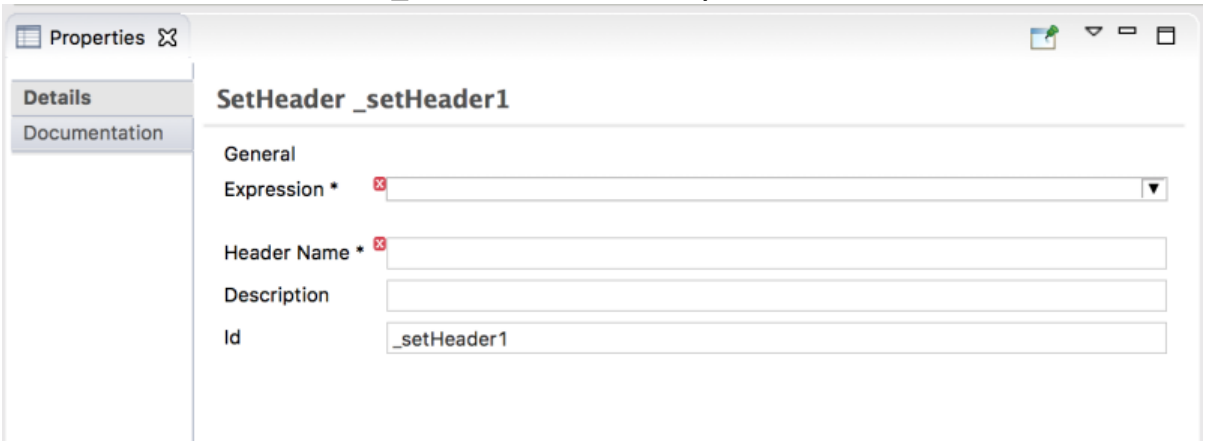
添加和配置消息标头：

1. 在 swig 中，打开 Transformation drawer，然后单击 Set Header ()模式。

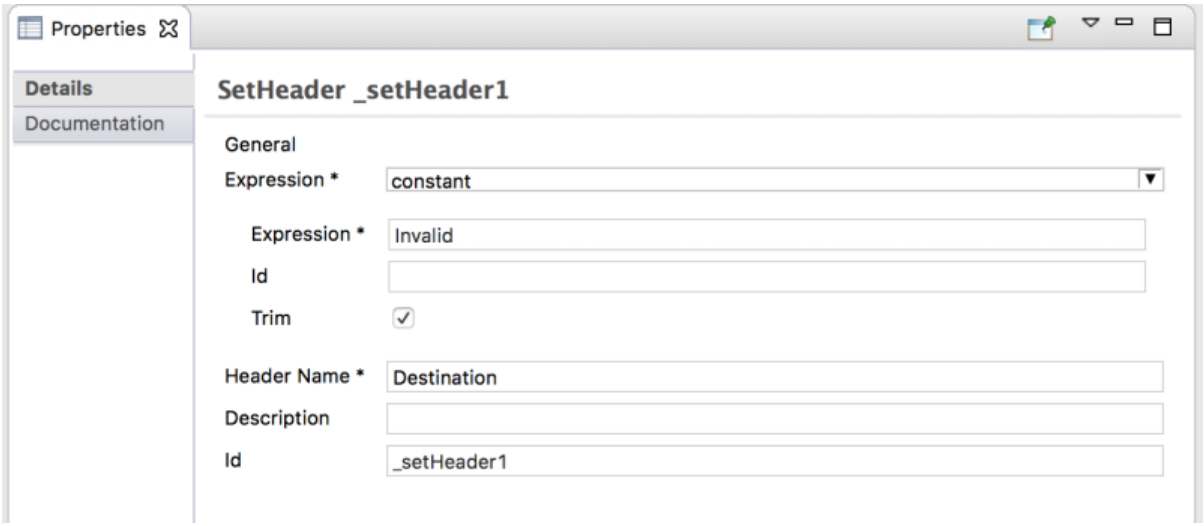
2. 在 canvas 中，单击 **Log_log1** 节点。
When_when1 容器扩展以容纳 SetHeader_setHeader1 节点：



3. 在 canvas 上，选择 **SetHeader_setHeader1** 节点在 Properties 视图中打开其属性：

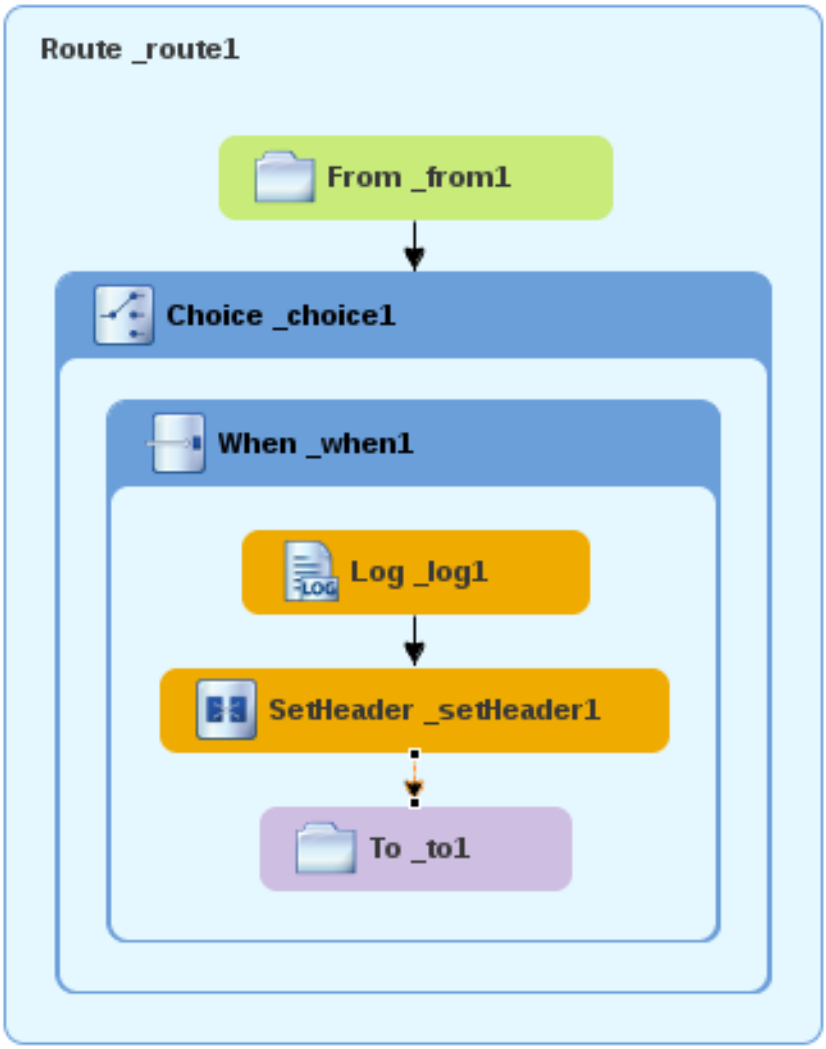


4. 点击 Expression 字段中的 ▼ 按钮打开可用语言列表，然后选择 常数。
5. 在缩进 Expression 字段中，键入 **Invalid**。
6. 在 Header Name 字段中，键入 **Destination**。
7. 其余的属性保留原样。

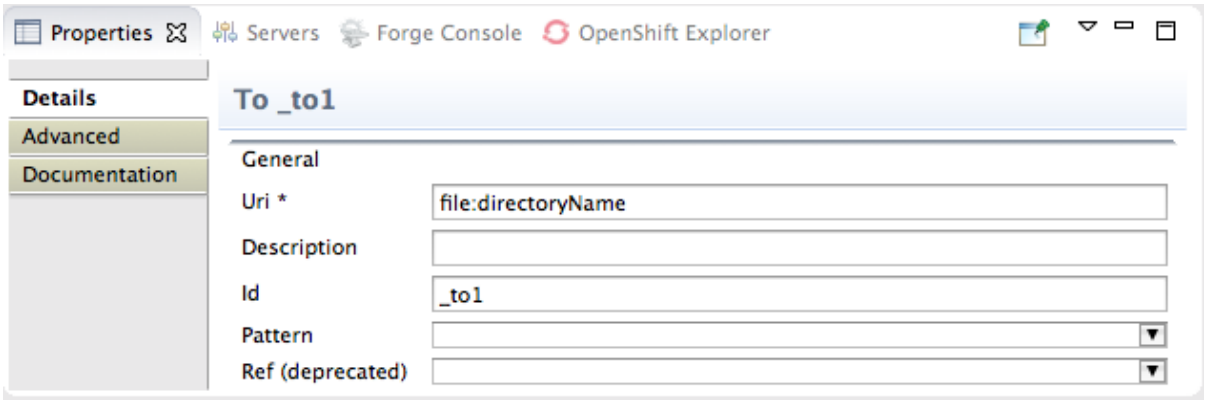


8. 在 swig 中，打开 Components drawer，然后单击 File ( )组件。

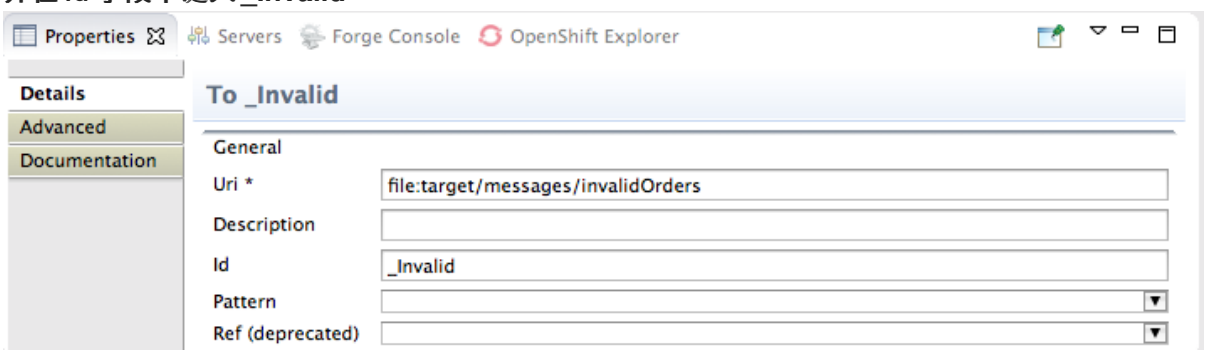
9. 在 canvas 中，点 SetHeader_setHeader1 节点。
When_when1 容器扩展以容纳 To_to1 节点。



10. 在 canvas 上，选择 To_to1 节点在 Properties 视图中打开其属性：

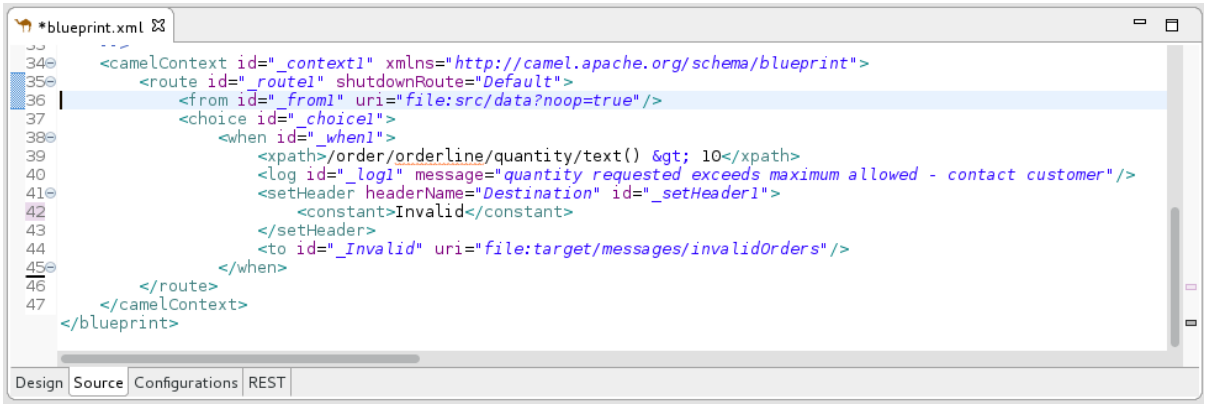


11. 在 Details 标签页中，将 *directoryName* 替换为 Uri 字段中的 target/messages/invalidOrders，并在 Id 字段中键入 _Invalid ：



12. 保存 路由上下文文件。

13. 点 Source 选项卡查看路由的 XML：



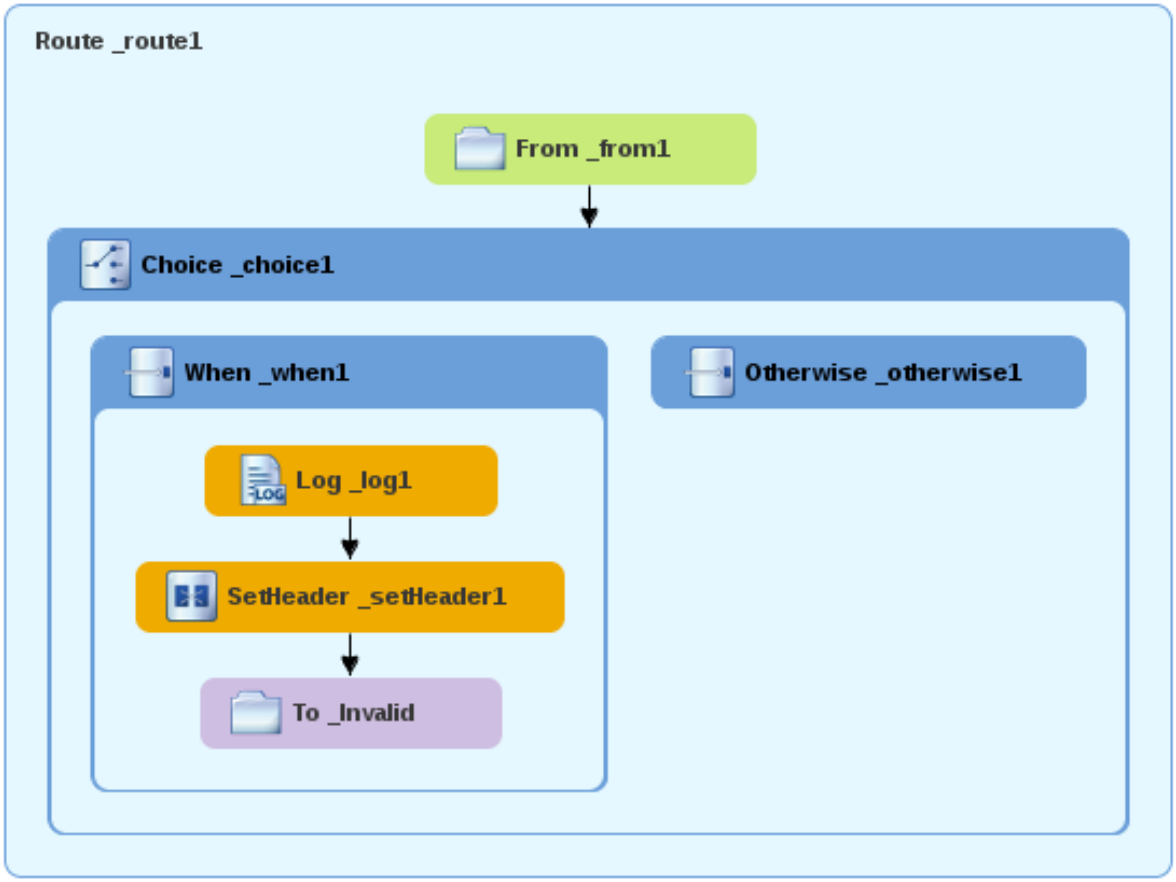
添加和配置分支以处理有效订购

目前，CBR 处理包含无效顺序的消息（数量值大于 10）。

要添加并配置路由分支来处理有效的订购（即，任何与 When_when1 节点设置的 XPath 表达式不匹配的任何 XML 消息）：

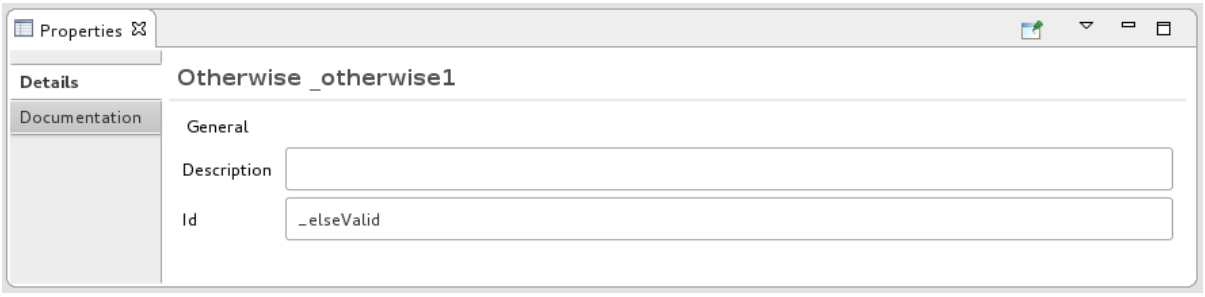
1. 在 swig 中，打开 Routing drawer，再单击 Otherwise ()模式。

2. 在 canvas 中，点 Choice_choice1 容器：



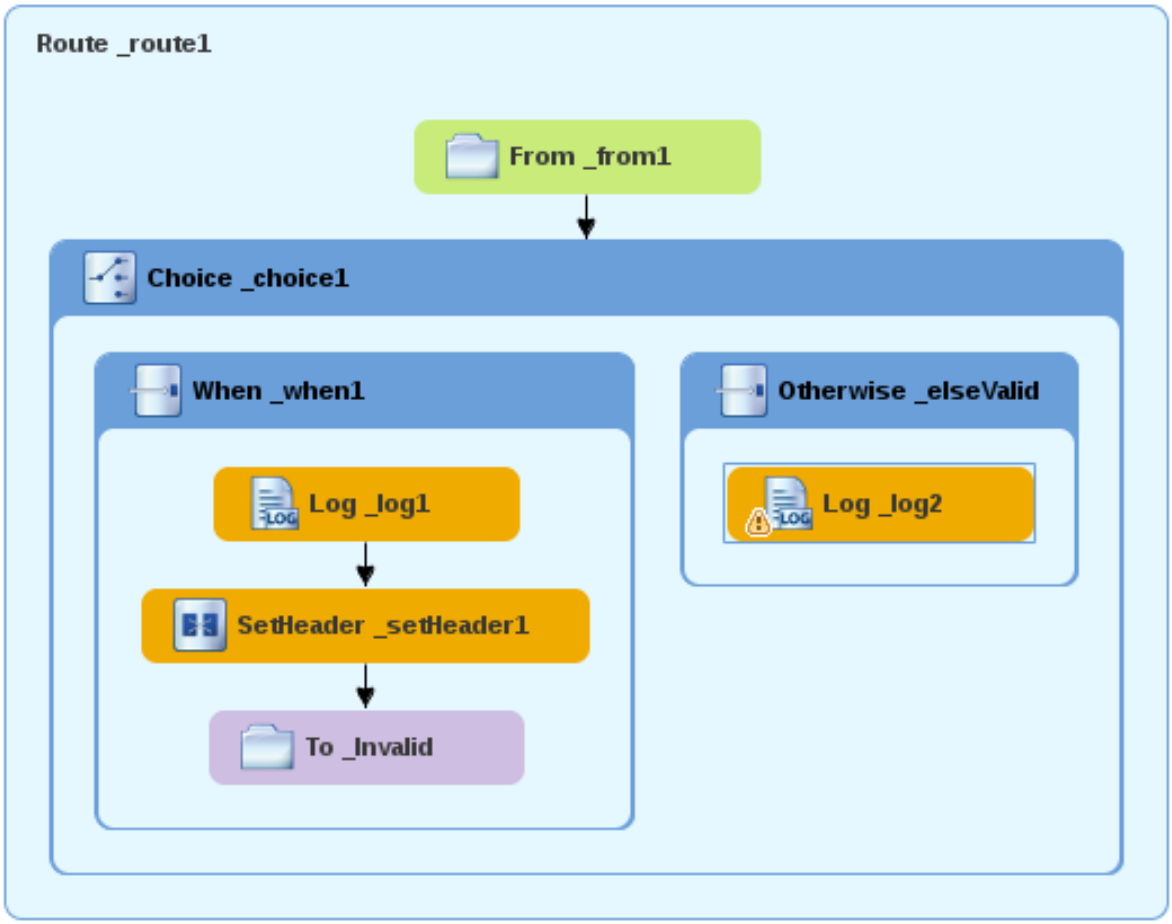
Choice_choice1 容器展开，以适应 Otherwise_otherwise1 节点。

3. **在 canvas 上，选择 Otherwise_otherwise1 节点，以在 Properties 视图中打开其属性。**
4. **在 Id 字段中，将 _otherwise1 改为 _elseValid ：**

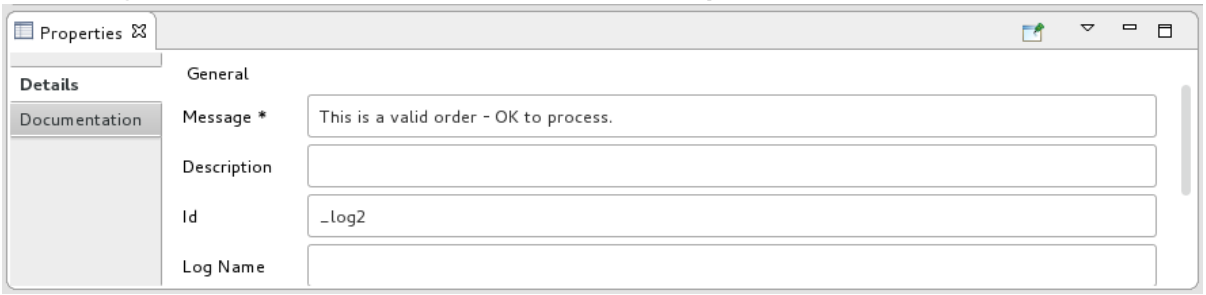


为**其他分支配置日志记录：**

1. **在 swig 中，打开 Components drawer，然后单击 Log (**  **)组件。**
2. **在 canvas 中，点 Otherwise_elseValid 节点：**
   Otherwise-elseValid 容器展开，以容纳 Log_log2 节点。



3. 在 canvas 上，选择 **Log_log2** 节点，以在 Properties 视图中打开其属性。
4. 在 Message 字段中，键入 **This is a valid order - OK to process.**

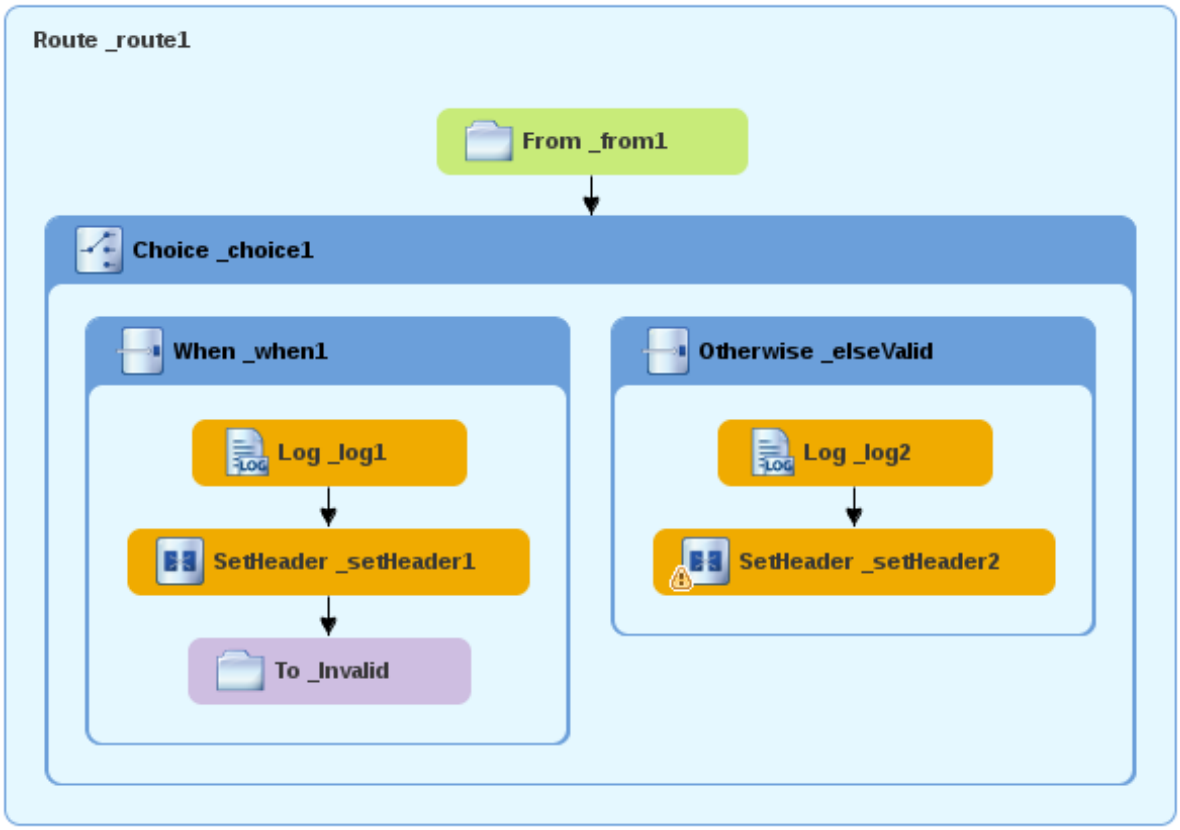


其余的属性保留原样。

5. 保存 路由。

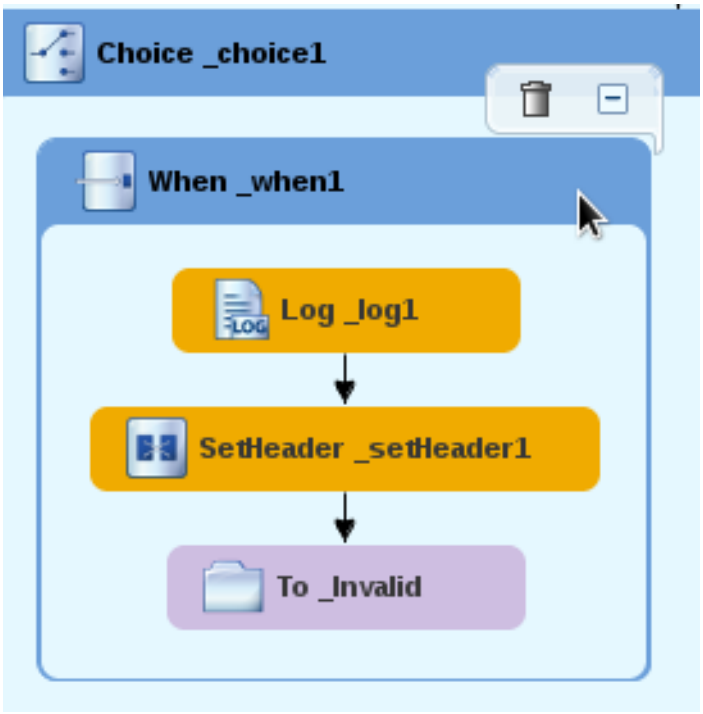
为其他分支配置消息标头：

1. 在 Prod 中，打开 Transformation drawer，然后单击 Set Header 模式。
2. 在 canvas 中，单击 **Log_log2** 节点。
Otherwise_elseValid 容器展开，以适应 SetHeader_setHeader2 节点。

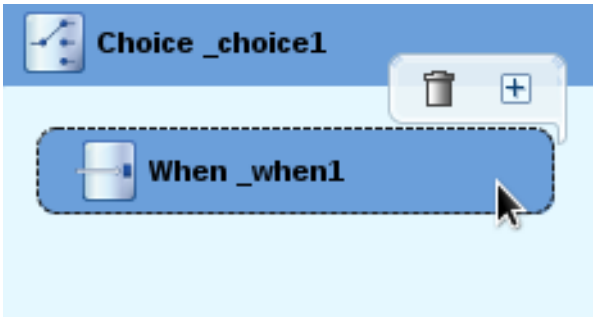

Route _route1
From _from1
Choice _choice1
When _when1
Log _log1
SetHeader _setHeader1
To _Invalid
Otherwise _elseValid
Log _log2
SetHeader _setHeader2

注意

当图表被拥塞时，您可以折叠容器来释放空间。要做到这一点，选择要折叠的容器，然后点击其 按钮：

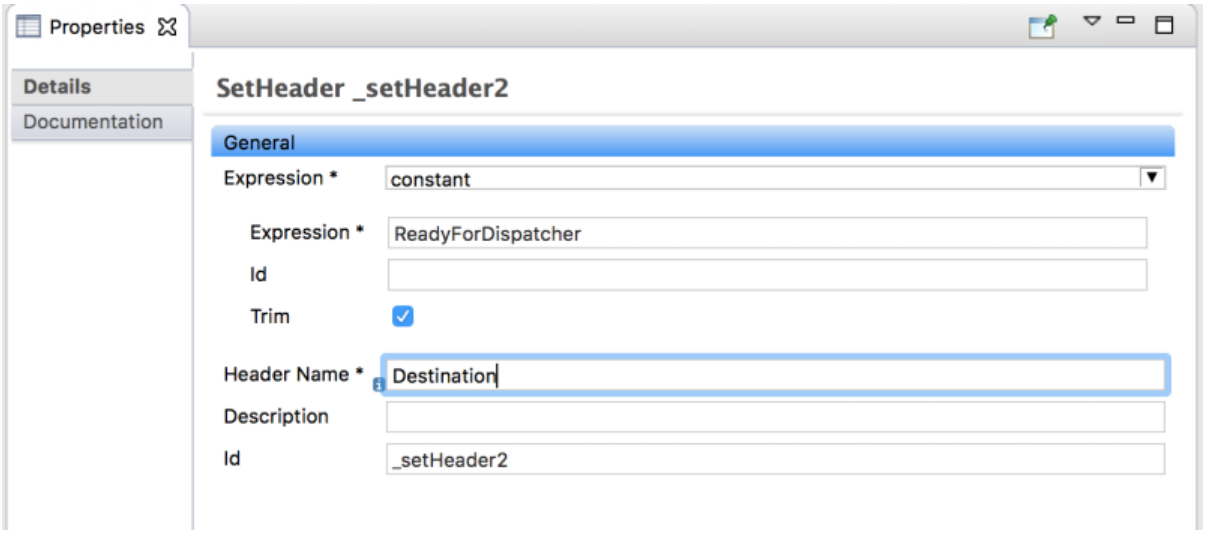


要重新打开容器，请选择它，然后点击其 按钮：



在 Design 选项卡中，联合和扩展容器不会影响路由上下文文件。它保持不变。

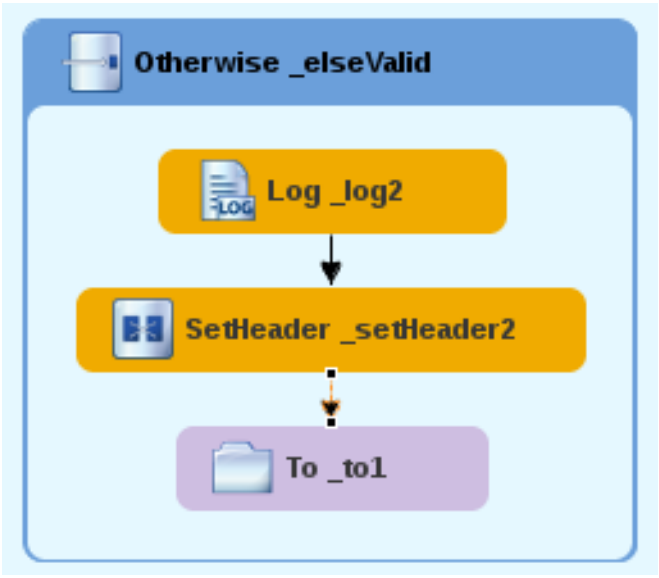
3. **在 canvas 上，选择 SetHeader_setHeader2 节点，以在 Properties 视图中打开其属性。**
4. **点击 Expression 字段中的 按钮打开可用语言列表，然后选择 常量。**
5. **在缩进 Expression 字段中，键入 ReadyForDispatcher。**
6. **在 Header Name 字段中，键入 Destination。**
7. **其余的属性保留原样。**



为有效信息指定目标文件夹：

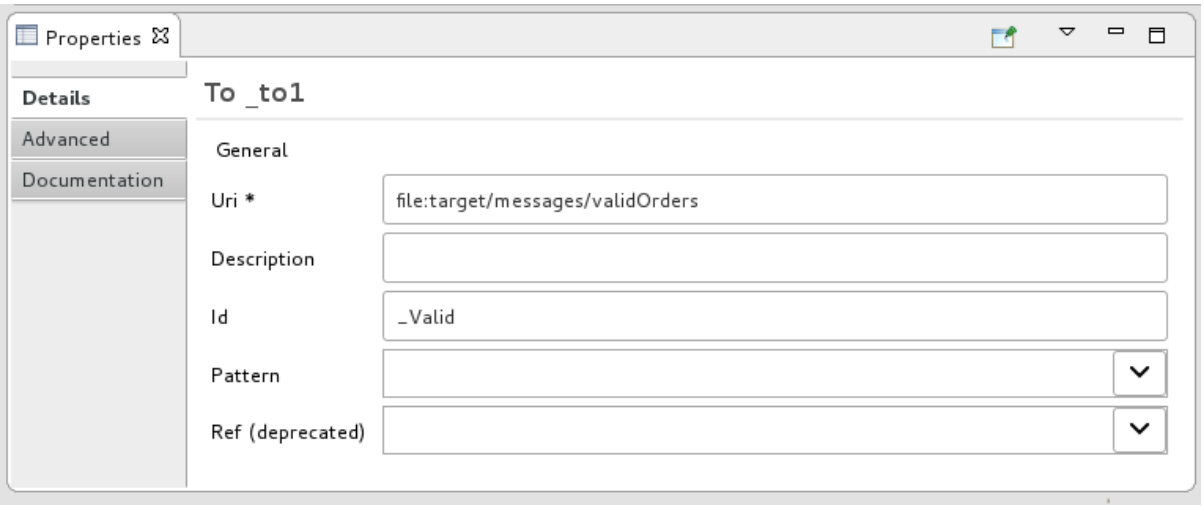
1. 在 swig 中，打开 Components drawer，然后选择 File ( )组件。

2. 在 canvas 中，点 SetHeader_setHeader2 节点。
Otherwise_elseValid 容器展开，以适应 To_to1 节点。

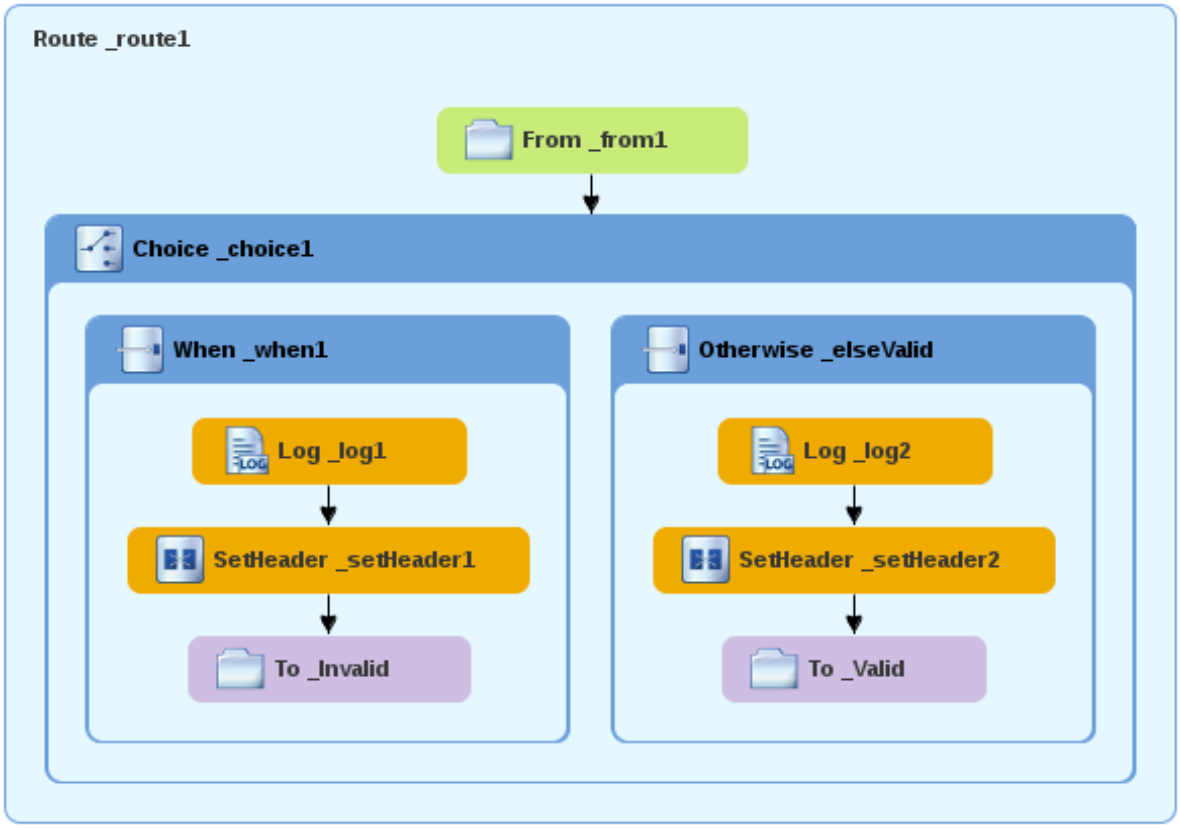


3. 在 canvas 上，选择 To_to1 节点，以在 Properties 视图中打开其属性。

4. 在 URI 字段中，将 *directoryName* 替换为 target/messages/validOrders，然后在 Id 字段中键入 _Valid。



5. **保存 路由上下文文件。**
   已完成的基于内容的路由器应类似如下：



6. **单击 canvas 左下方的 Source 选项卡，以显示路由的 XML。**

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.osgi.org/xmlns/blueprint/v1.0.0
   https://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd
   http://camel.apache.org/schema/blueprint
   http://camel.apache.org/schema/blueprint/camel-blueprint.xsd">

<camelContext id="_context1" xmlns="http://camel.apache.org/schema/blueprint">
      <route id="_route1">
         <from id="_from1" uri="file:src/data?noop=true"/>
```

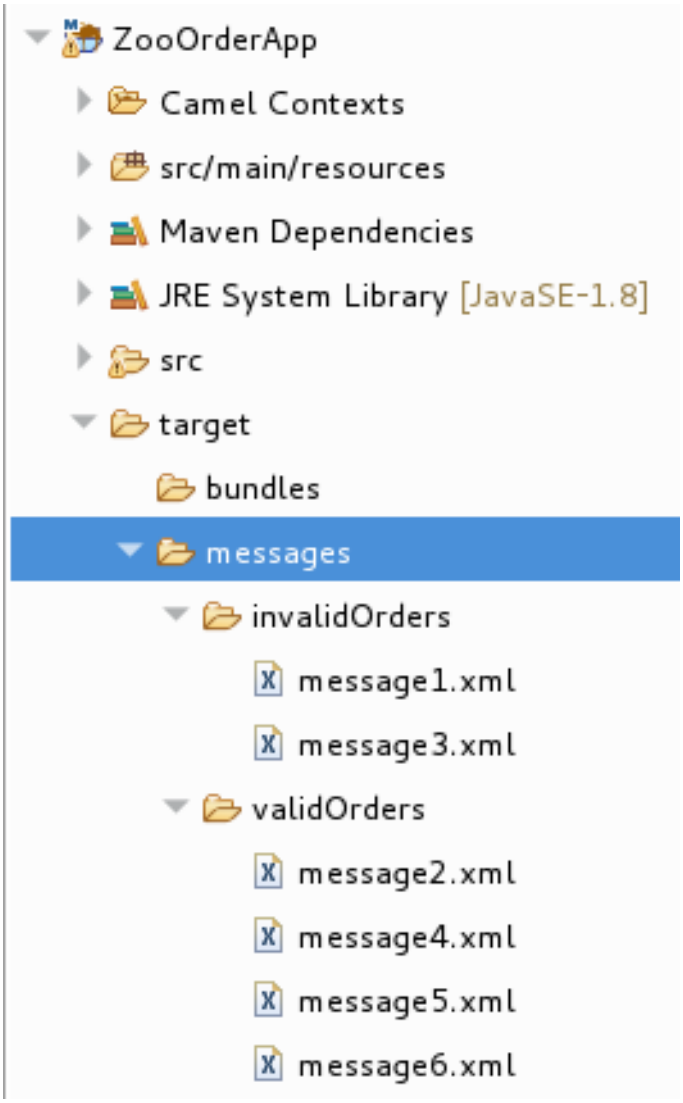
```
        <choice id="_choice1">
          <when id="_when1">
            <xpath>/order/orderline/quantity/text() &gt; 10</xpath>
            <log id="_log1" message="The quantity requested exceeds the maximum
allowed - contact customer."/>
            <setHeader headerName="Destination" id="_setHeader1">
              <constant>Invalid</constant>
            </setHeader>
            <to id="_Invalid" uri="file:target/messages/invalidOrders"/>
          </when>
          <otherwise id="_elseValid">
            <log id="_log2" message="This is a valid order - OK to process."/>
            <setHeader headerName="Destination" id="_setHeader2">
              <constant>ReadyForDispatcher</constant>
            </setHeader>
            <to id="_Valid" uri="file:target/messages/validOrders"/>
          </otherwise>
        </choice>
      </route>
    </camelContext>
</blueprint>
```

验证 CBR

您可以运行 "运行路由"一节 教程中描述的新路由，并查看 Console 视图来查看日志消息。

运行它后，要验证路由是否已正确执行，请检查 Project Explorer 中的目标目标文件夹：

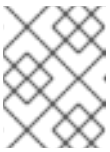
1. 选择 **ZooOrderApp**。
2. **右键单击该上下文菜单，然后选择 Refresh。**
3. **在项目根节点(ZooOrderApp)下，找到 target/messages/ 文件夹并将其展开。**



4. 检查 **target/messages/invalidOrders** 文件夹是否包含 **message1.xml** 和 **message3.xml**。在这些消息中，**quantity** 元素的值会超过 10。

5. 检查 **target/messages/validOrders** 文件夹是否包含包含有效顺序的四个消息文件：

   - **message2.xml**
   - **message4.xml**
   - **message5.xml**
   - **message6.xml**
     在这些消息中，**quantity** 元素的值小于或等于 10。



注意

要查看消息内容，请双击每个消息，以在路由编辑器的 XML 编辑器中打开它。

后续步骤

在下一教程 第 6 章 *在路由上下文中添加另一个路由* 中，您可以添加额外处理有效顺序消息的第二个路由。

第 6 章 在路由上下文中添加另一个路由

本教程介绍了如何在 ZooOrderApp 项目的 blueprint.xml 文件中添加第二个路由到 camel 上下文。第二个路由：

- 直接从第一个路由的分支终端获取消息（评估 顺序 ）。
- 根据客户的国家/地区对有效消息排序。
- 将每个消息发送到 ZooOrderApp/target/messages 文件夹中的对应 国家 文件夹。例如，来自 Chicago zoo 的顺序被复制到 USA 文件夹。

目标

在本教程中，您将完成以下任务：

- 重新配置现有路由以直接连接第二个路由
- 在您的 Camel 上下文中添加第二个路由
- 配置第二个路由，以直接从第一个路由的 otherwise 分支中获取消息
- 将基于内容的路由器添加到第二个路由
- 为第二个路由的内容基于内容的路由器的每个输出分支添加并配置消息标头、日志记录和目标目的地

先决条件

要启动本教程，您需要从以下之一生成的 ZooOrderApp 项目：

- 完成 第 5 章 *添加基于内容的路由器* 教程。
  or
- 完成 第 2 章 *设置您的环境* 教程，并将项目的 blueprint.xml 文件替换为提供的 blueprintContexts/blueprint2.xml 文件，如 "关于资源文件"一节 所述。

配置现有路由的端点

现有路由将所有有效顺序发送到 target/messages/validOrders 文件夹。

在本小节中，您将重新配置现有路由的 Otherwise _elseValid 分支的端点，以连接到第二个路由（您在下一部分中创建）。

配置现有路由以与第二个路由直接连接：

1. 在路由编辑器中打开 ZooOrderApp/src/main/resources/OSGI-INF/blueprint/blueprint.xml。
2. 在 canvas 上，选择 Route_route1 容器，以在 Properties 视图中打开其属性。
3. 向下滚动到 Shutdown Route 属性，然后选择 Default。
4. 在 canvas 上，选择终端文件节点 To_Valid 以在 Properties 视图中显示其属性。
5. 在 Uri 字段中，删除现有的文本，然后输入 direct:OrderFulfillment。

6. 在 Id 字段中，输入 _Fulfill。

注意

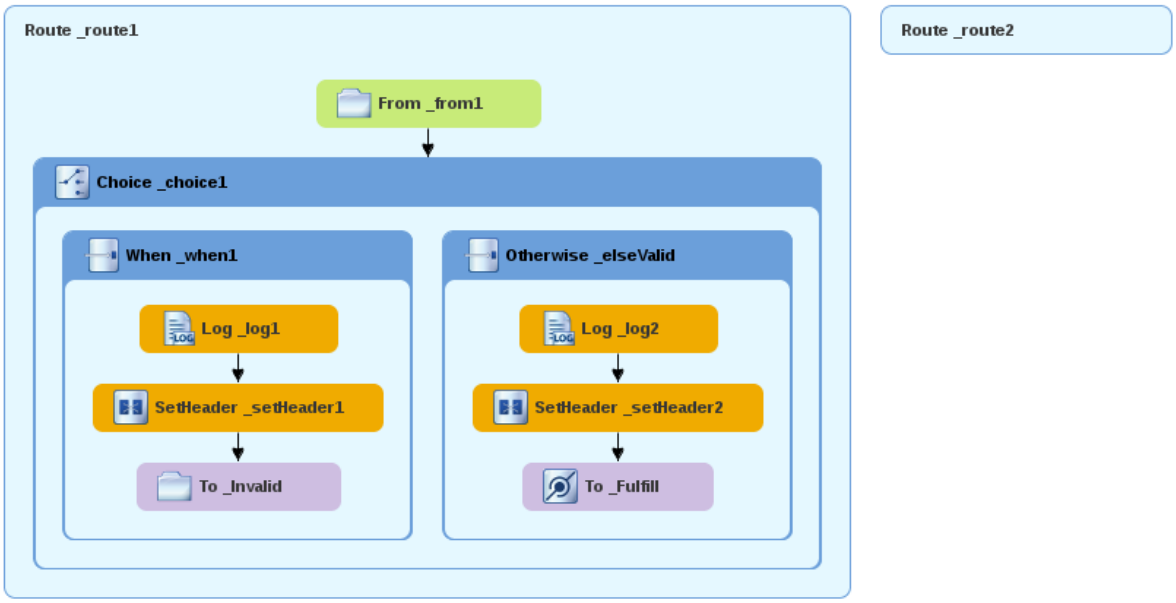
您可以替换为 Components → Direct 组件，而不是重新处理现有的 To_Valid 终端文件节点，而是使用与 repurposed To_Valid 节点相同的属性值替代。

要了解有关 直接 组件的更多信息，请参阅 Apache Camel 组件参考。

添加第二个路由

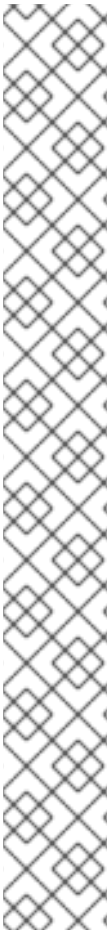
将另一个路由添加到路由上下文：

1. 在 swig 中，打开 Routing drawer，然后单击 Route ()模式。
2. 在 canvas 中，点 Route_route1 容器右侧的：



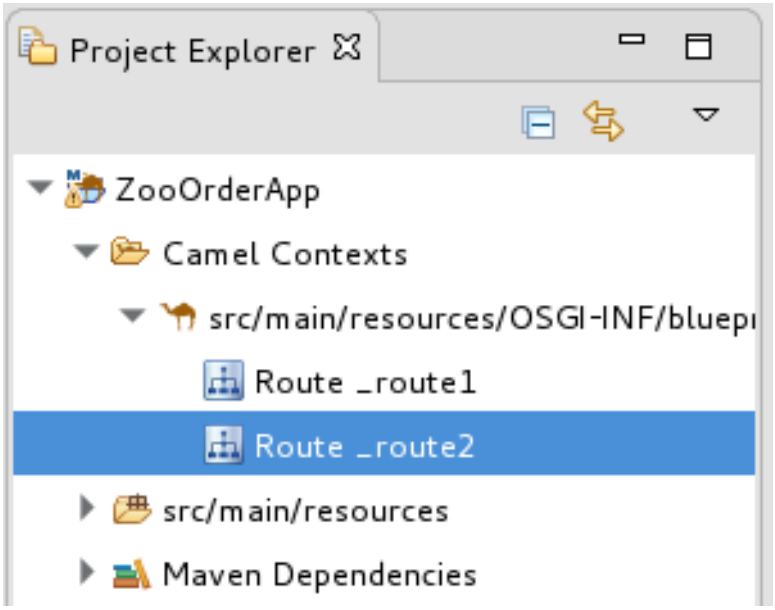
Route 模式成为 canvas 上的Route_route2 容器节点。

3. 单击 Route_route2 容器节点，以在 Properties 视图中显示其属性。将属性保留原样。
4. 保存该文件。



注意

随着路由上下文复杂性的增长，您可能需要在处理路由时将路由编辑器集中到单独的路由上。要做到这一点，在 Project Explorer 中，双击您希望路由编辑器在 canvas 上显示的路由，如 Route_route2 ：

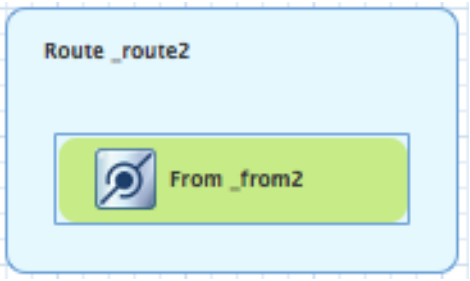


要在 canvas 上的路由上下文中显示所有路由，请双击 Camel Contexts 文件夹顶部的项目的 .xml 上下文文件条目(src/main/resources/OSGI-INF/...)。

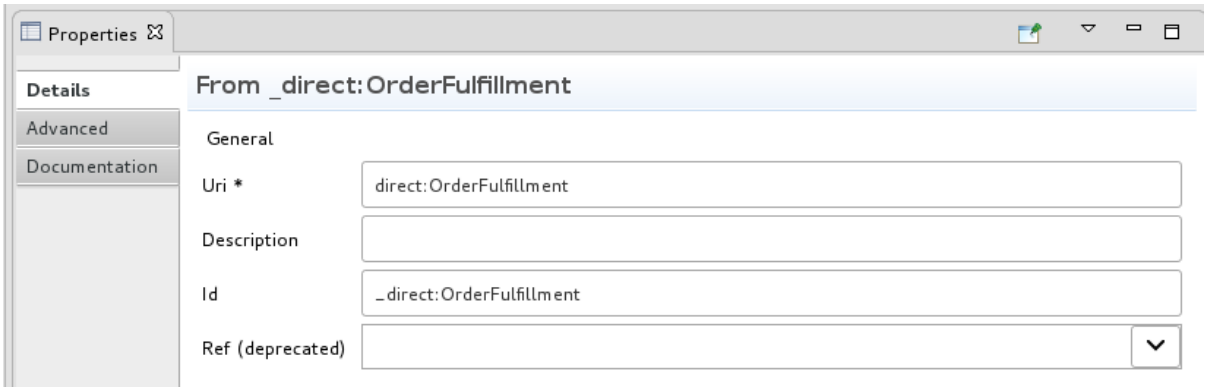
配置选择分支以处理美国订购

在本小节中，您将 Choice 分支添加到路由中，并将路由配置为发送美国订购到新的 target/messages/validOrders/USA 文件夹。您还可以设置消息标头和日志文件组件。

1. **在 swig 中，打开 Components drawer，然后选择 直接 组件(**  **)。**
2. **在 canvas 中，点 Route_route2 容器：**
   Route_route2 容器扩展以容纳 Direct 组件(From_from2 节点)：

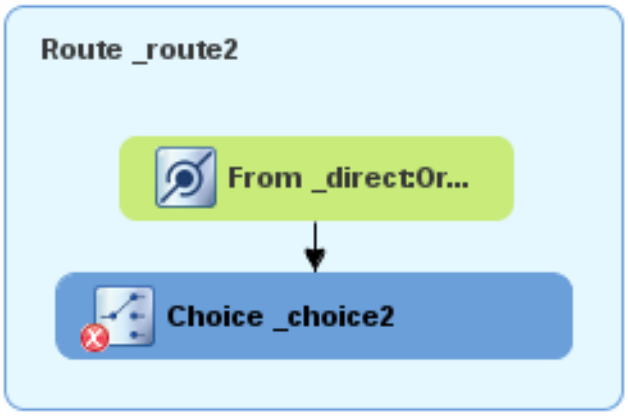


3. **在 canvas 上，单击 From_from2 节点，以在 Properties 视图中打开其属性。**
4. **在 Uri 字段中，将 name (following direct:)替换为 OrderFulfillment，然后在 Id 字段中输入 _direct:OrderFulfillment。**



5. 在 swig 中，打开 Routing drawer，然后选择 Choice ()模式。

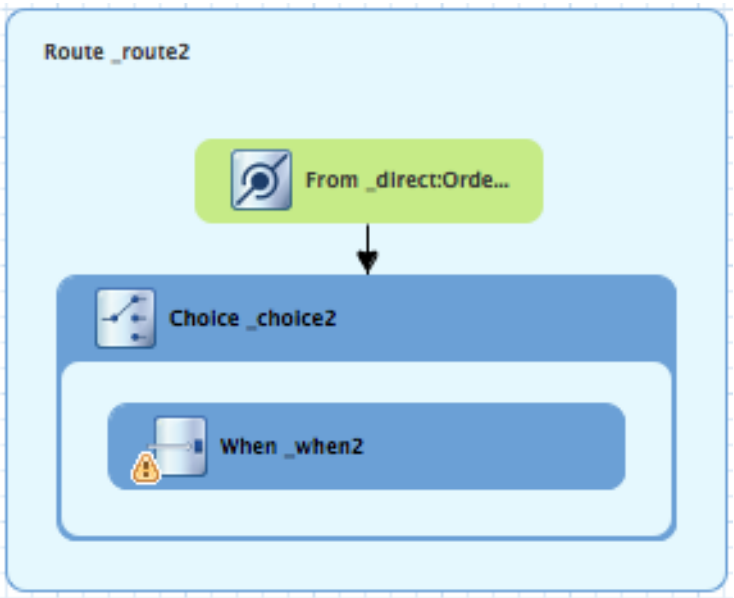
6. 在 canvas 中，点 From _direct:OrderFulfillment 节点。
Route_route2 容器扩展以适应 Choice_choice2 节点：



在 Properties 视图中，保留 Choice_choice2 节点的属性。

7. 在 swig 中，打开 Routing drawer，然后选择 When ()模式。

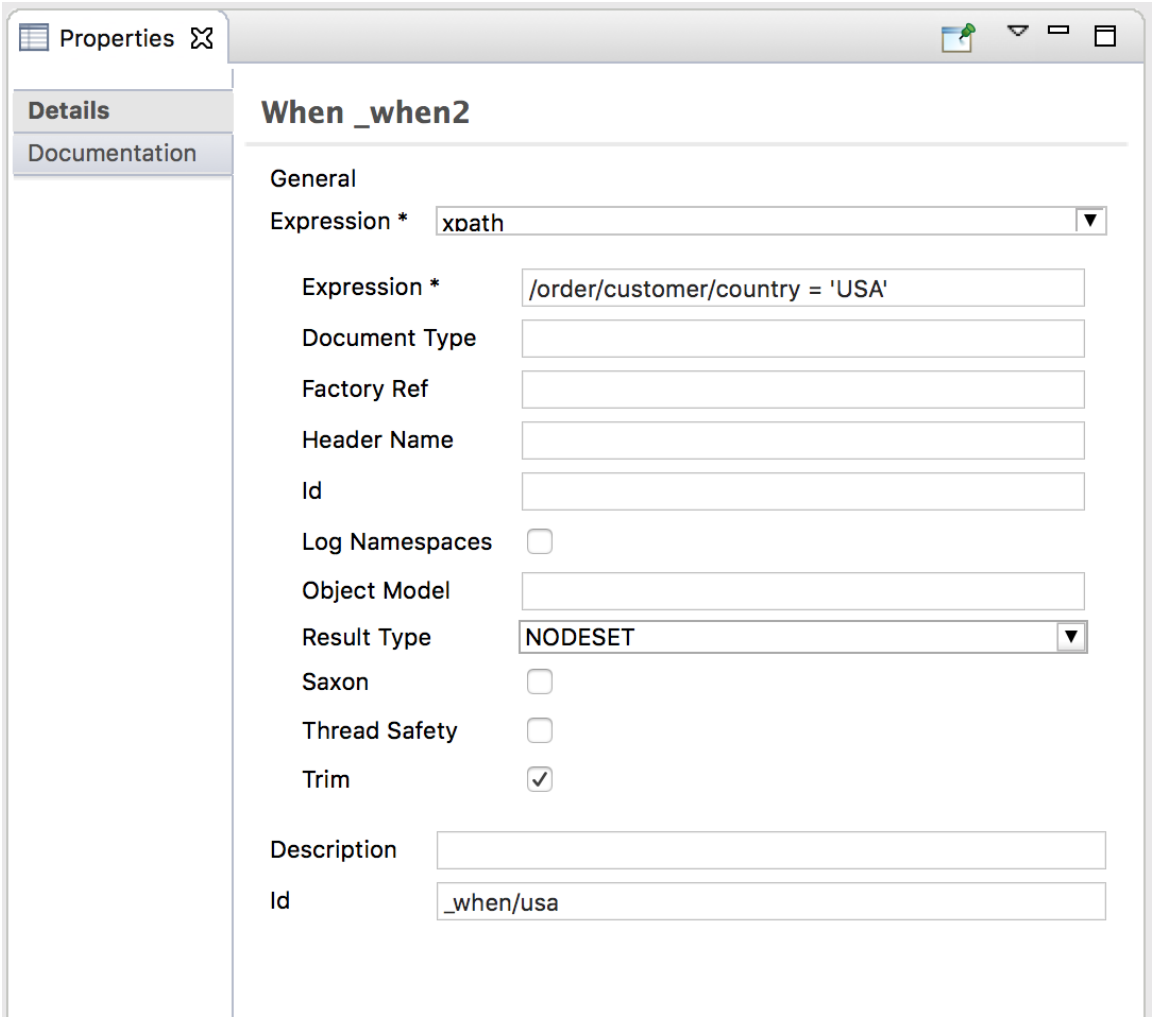
8. 在 canvas 中，单击 Choice_choice2 节点。
Choice_choice2 容器扩展了，以适应 When_when2 节点。



9. 在 canvas 上，选择 When_when2 节点，以在 Properties 视图中打开其属性。

10. 设置 When_when2 节点的属性，如下所示：

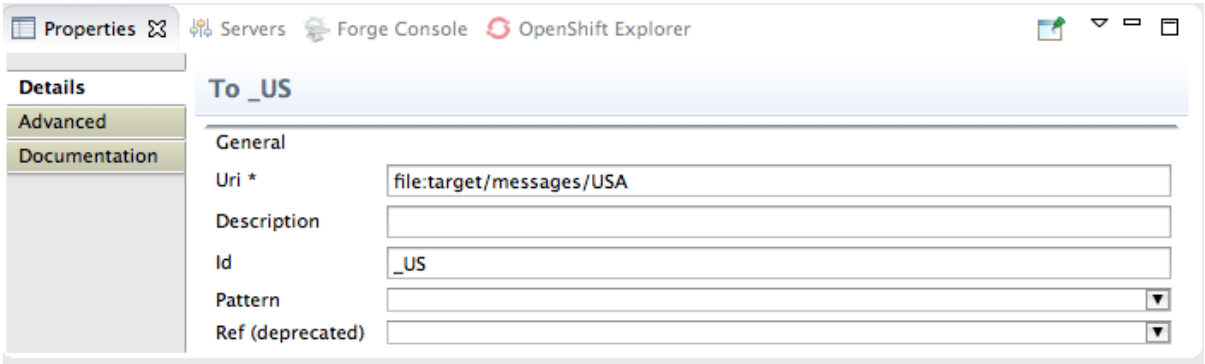
    - 从 Expression 下拉列表中选择 xpath。
    - 在缩进的 Expression 字段中，键入 /order/customer/country = 'USA'。
    - 保留 Trim 启用。
    - 在第二个 Id 字段中，键入 _when/usa



11. 在 swig 中，打开 Components drawer，然后选择 File 组件(1)。

12. 在 canvas 中，点 When_when/usa 容器。
    When_when/usa 容器扩展以容纳 To_to1 节点。

13. 在 Properties 视图中：

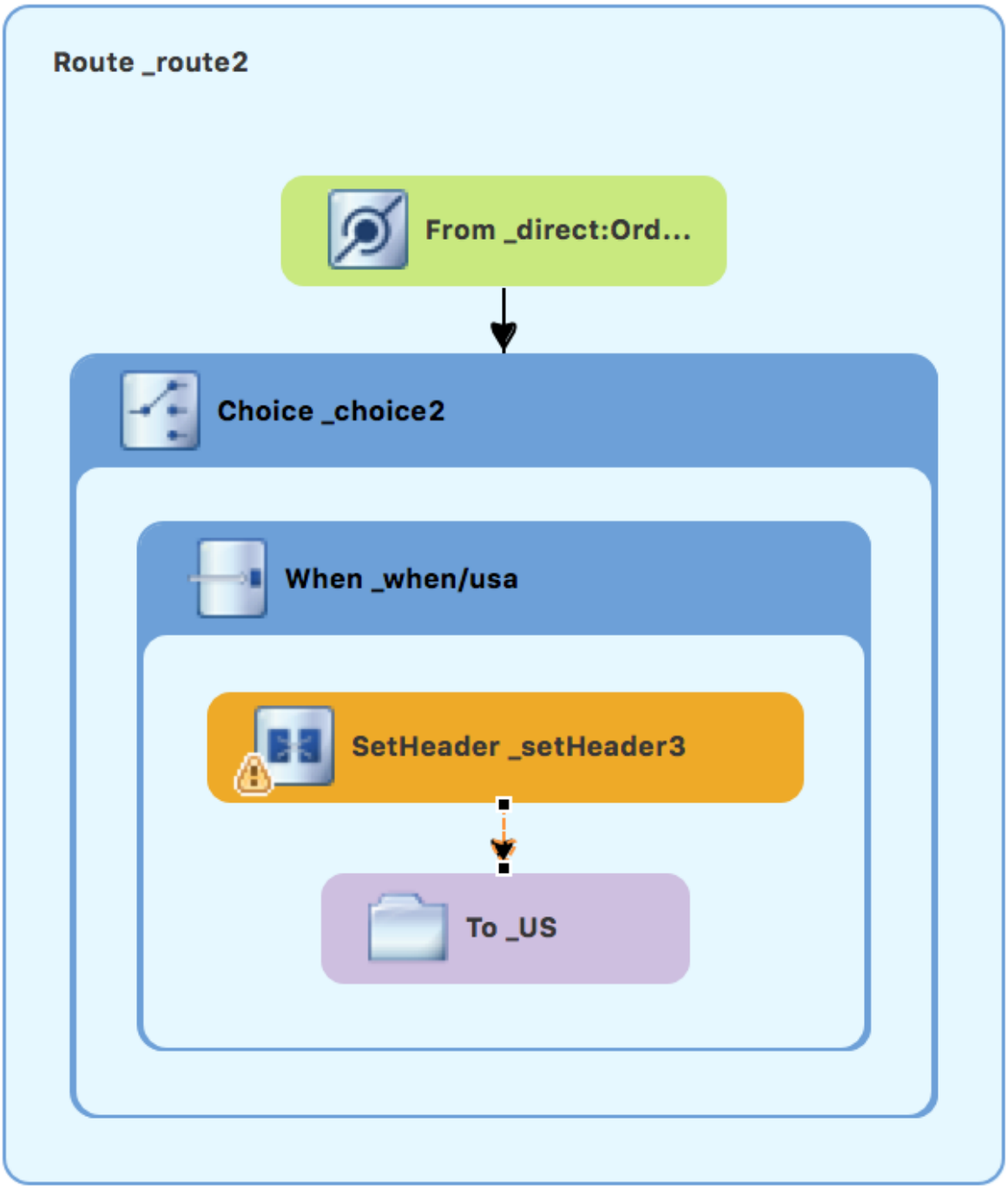


- 在 Uri 字段中，将 directoryName 替换为 target/messages/validOrders/USA。
- 在 Id 字段中，键入 _US。

14. 保存该文件。

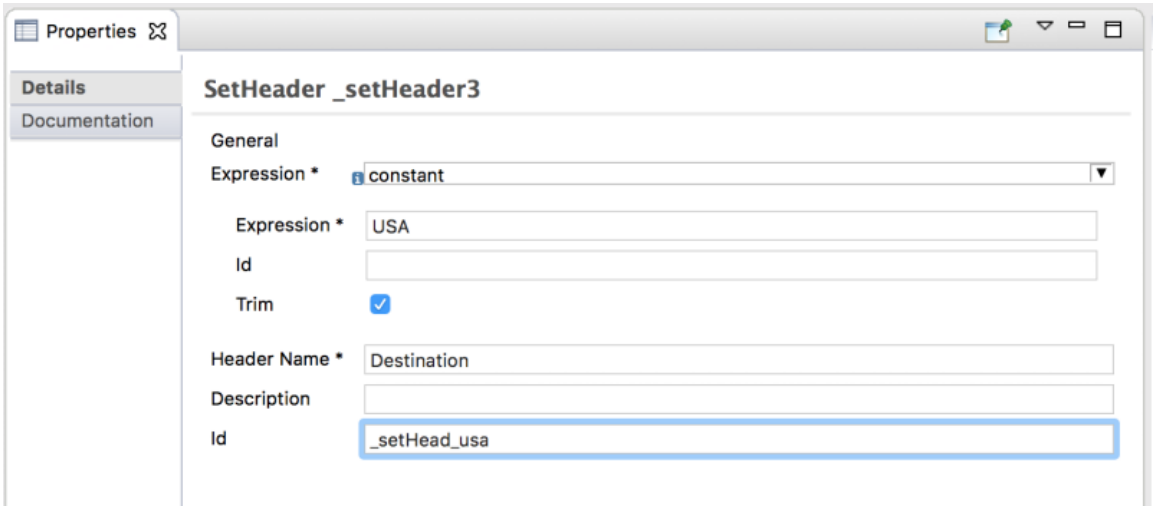
要设置消息标头并添加日志组件：

1. 在 Prod 中，打开 Transformation drawer，然后选择 Set Header 模式。
2. 在 canvas 中，点 When_when/usa 节点。
   When_when/usa 容器扩展以容纳 SetHeader_setHeader3 节点：

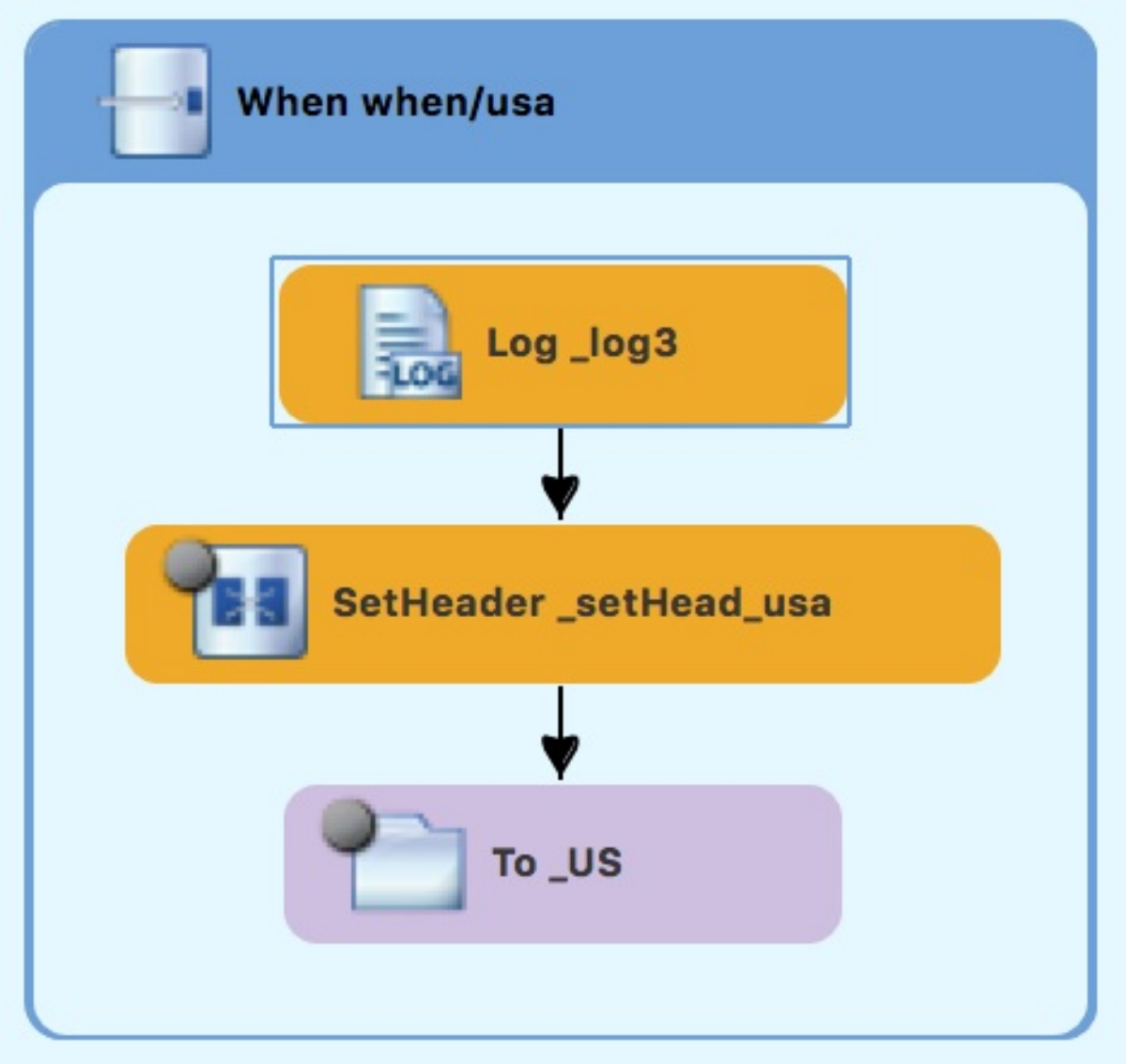


3. 在 canvas 上，选择 SetHeader_setHeader3 节点，以在 Properties 视图中打开其属性。
4. 设置节点的属性，如下所示：

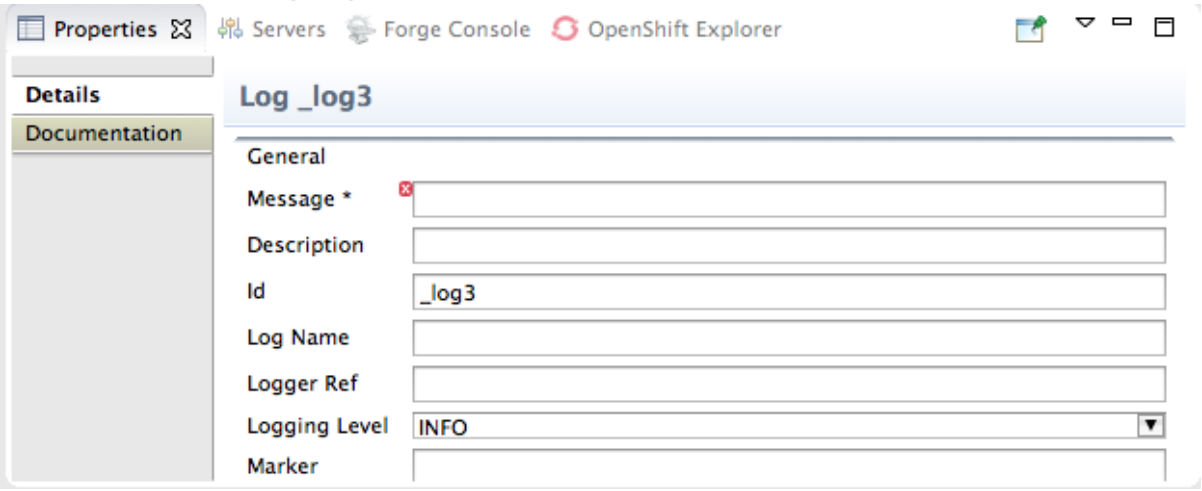
- 在 Expression 下拉菜单中选择 constant。
- 在缩进 Expression 字段中，键入： **USA**
- 保留 Trim 启用。
- 在 Header Name 字段中，键入： **Destination**
- 在第二个 Id 字段中，键入： **_setHead_usa**



5. 在 Prod 中，打开 Components drawer，然后选择 Log component ()。
6. 在 canvas 中，点 **SetHeader** 节点之上。
   When_when/usa 容器扩展以容纳 **Log_log3** 节点。

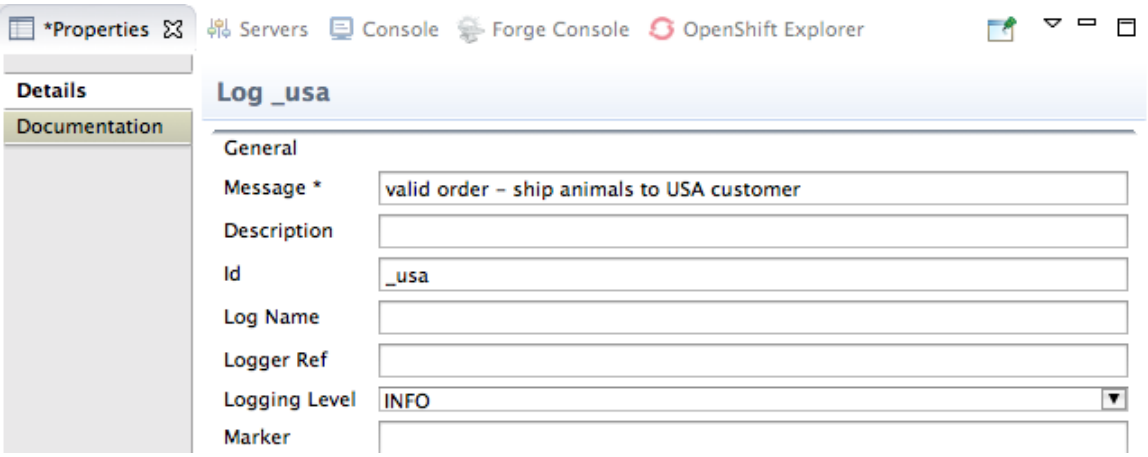


7. 在 canvas 上，选择 **Log_log3** 节点以在 Properties 视图中打开其属性：

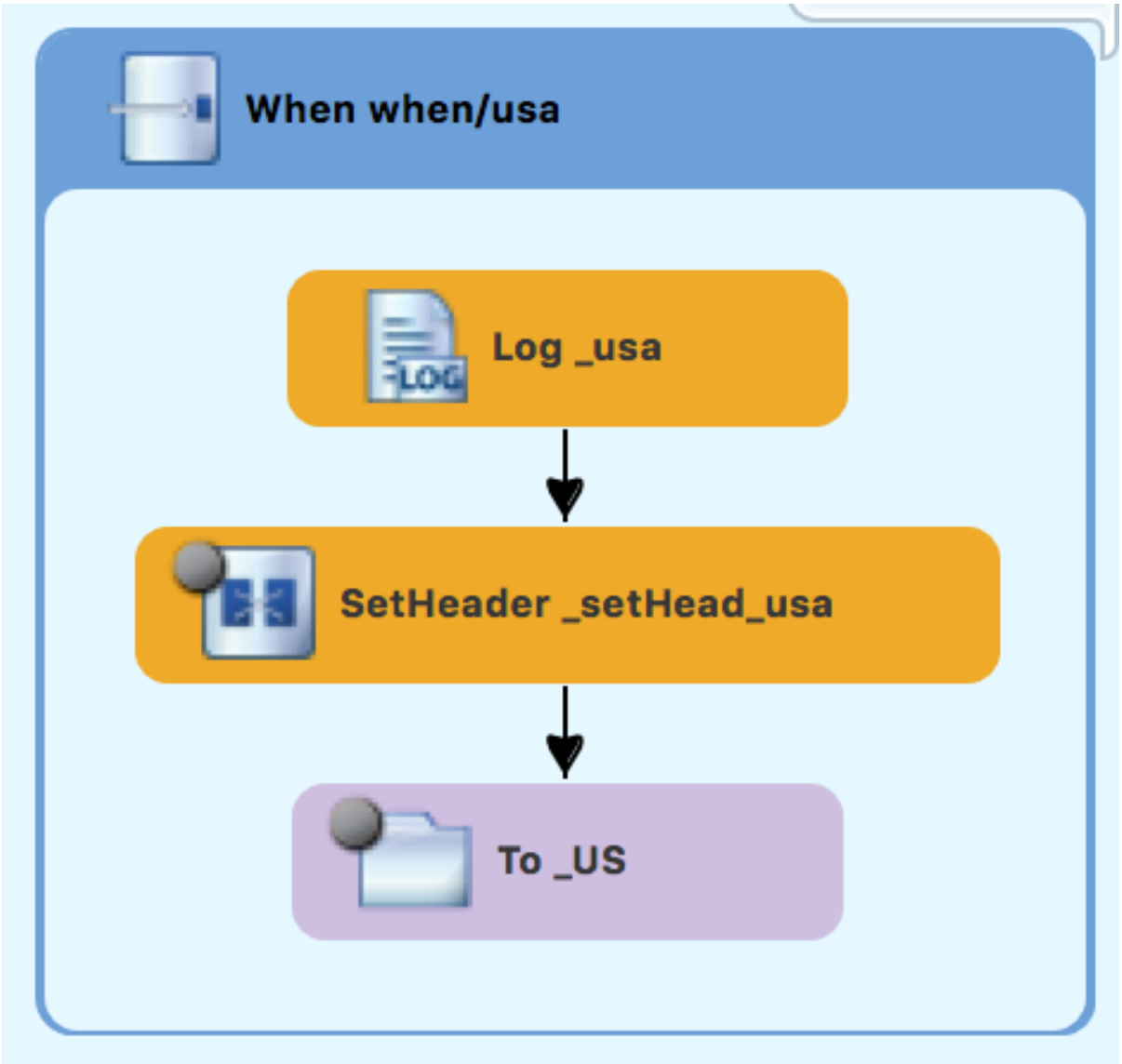


8. 在 Properties 视图中：

- 在 Message 字段中，键入 **Valid order - 向美国客户提供 imals。**
- 在 Id 字段中，键入 **_usa。**
- 将 日志记录级别 保留原样。



9. **保存该文件。**
   Route_route2 的 USA 分支应类似如下：

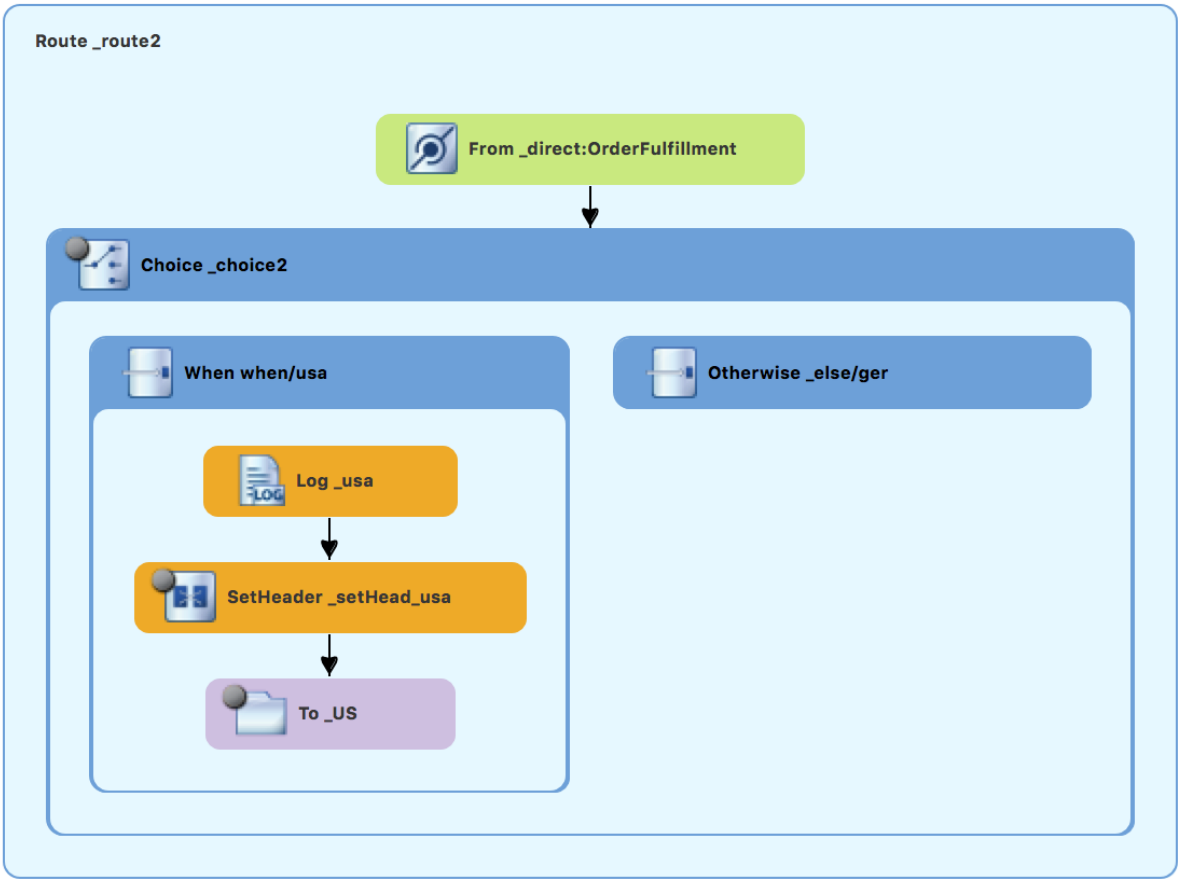


配置其他有效分支来处理德国订购

使用 canvas 上显示的 Route_route2 ：

1. **在 swig 中，打开 Routing drawer，然后选择 Otherwise pattern (**  **)。**

2. 在 canvas 中，单击 **Choice_choice2** 容器。
Choice_choice2 容器展开，以适应 **Otherwise_otherwise1** 节点。



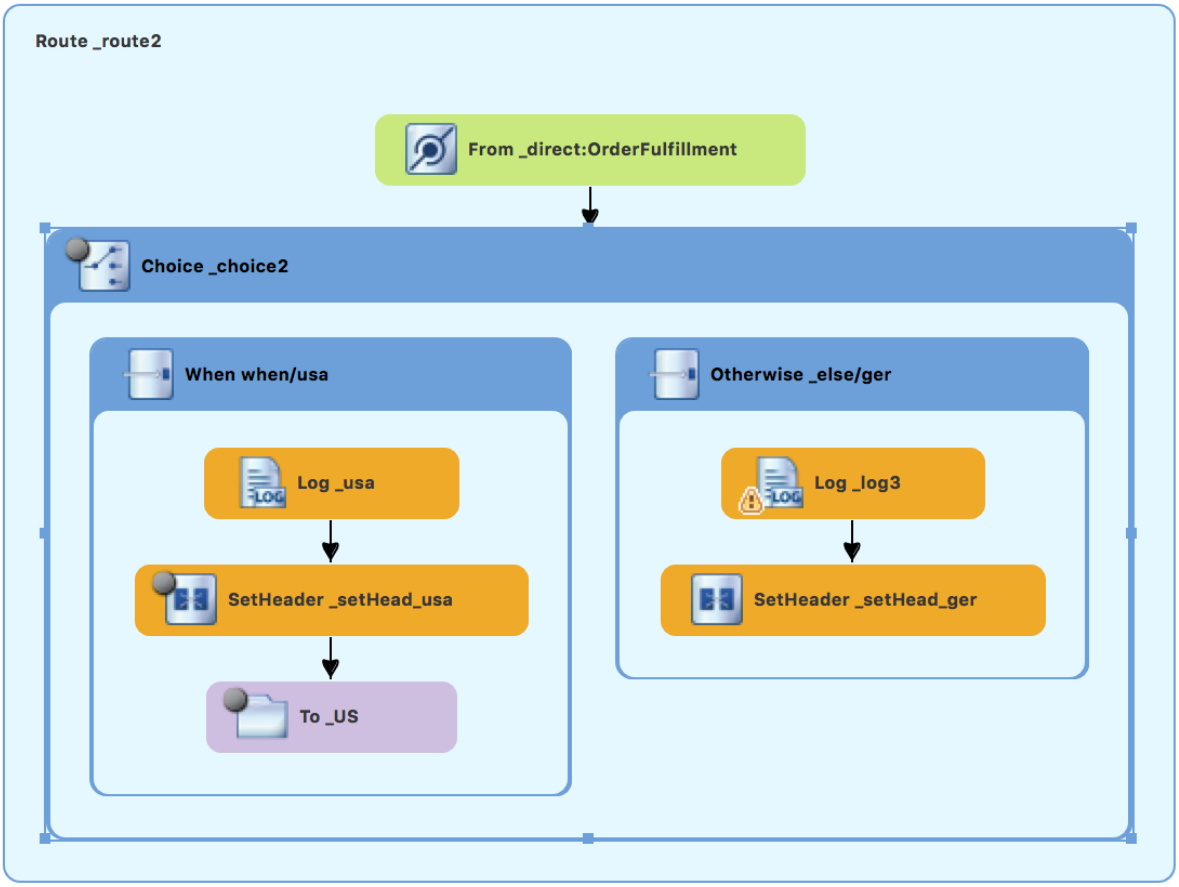
3. 选择 **Otherwise_otherwise1** 节点，以在 Properties 视图中打开其属性。

4. 在 Properties 视图中，在 Id 字段中输入 **_else/ger**。

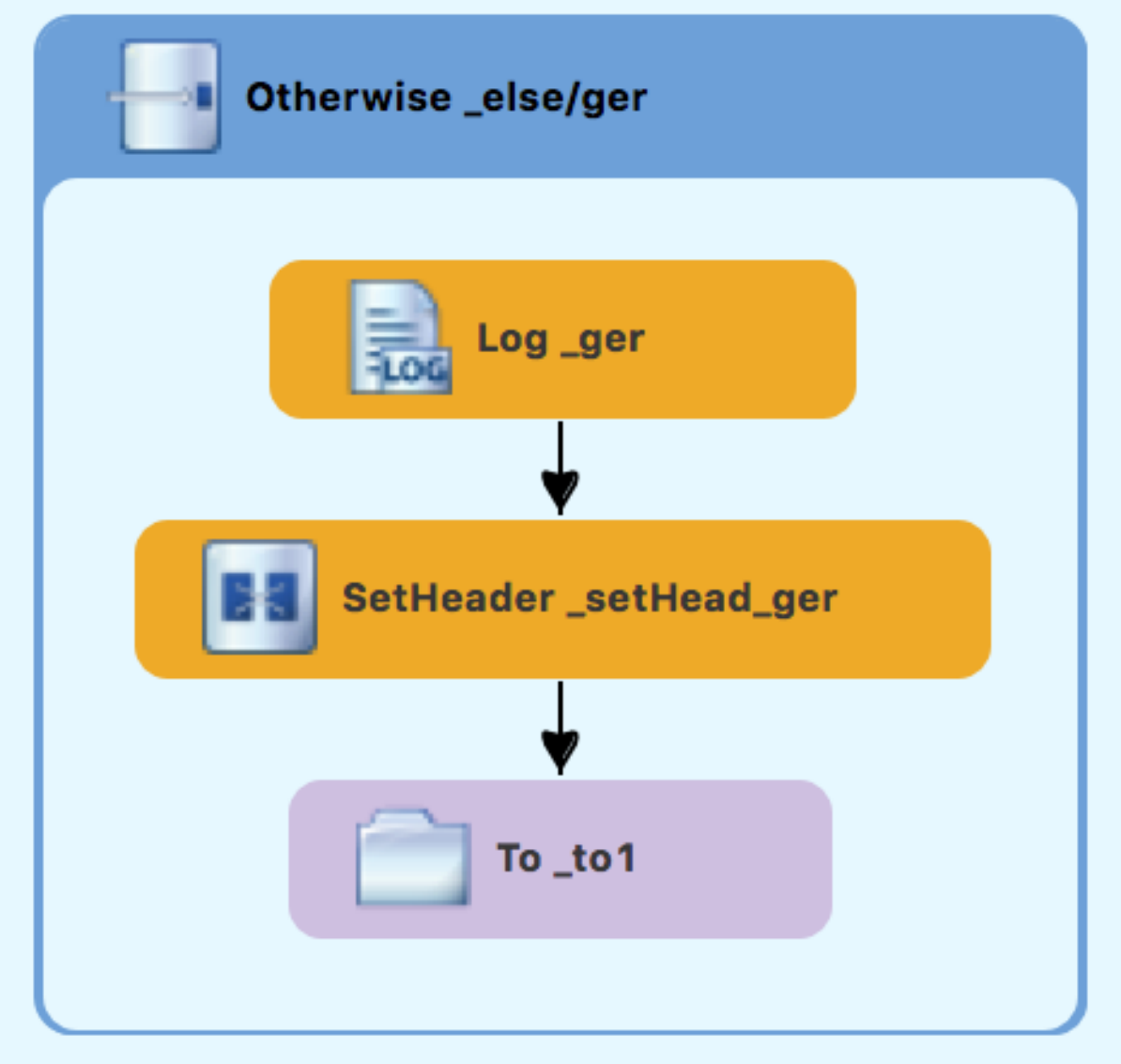
5. 在 Prod 中，打开 Transformation drawer，然后选择 Set Header pattern ()。

6. 在 canvas 中，单击 **Otherwise_else/ger** 节点。
Otherwise_else/ger 容器展开，以适应 **SetHeader_setHeader3** 节点。

7. 在 canvas 上，选择 **SetHeader_setHeader3** 节点，以在 Properties 视图中打开其属性。
8. 在 Properties 视图中：
   - 从 Expression 下拉列表中，选择 constant。
   - 在第二个 Expression 字段中，键入 **Germany**。
   - 将 Trim 保留为原样。
   - 在 Header Name 字段中，键入 **Destination**。
   - 在第二个 Id 字段中，键入 **_setHead_ger**。
9. 在 swig 中，打开 Components drawer，然后选择 **日志 模式(1)**。
10. 在 canvas 中，点 **SetHeader_setHead_ger** 节点下。
    Otherwise_else/ger 容器扩展，以适应 **Log_log3** 节点。如果需要，将连接器错误从 **Log_log3** 节点拖到 **SetHeader_setHead_ger** 节点：

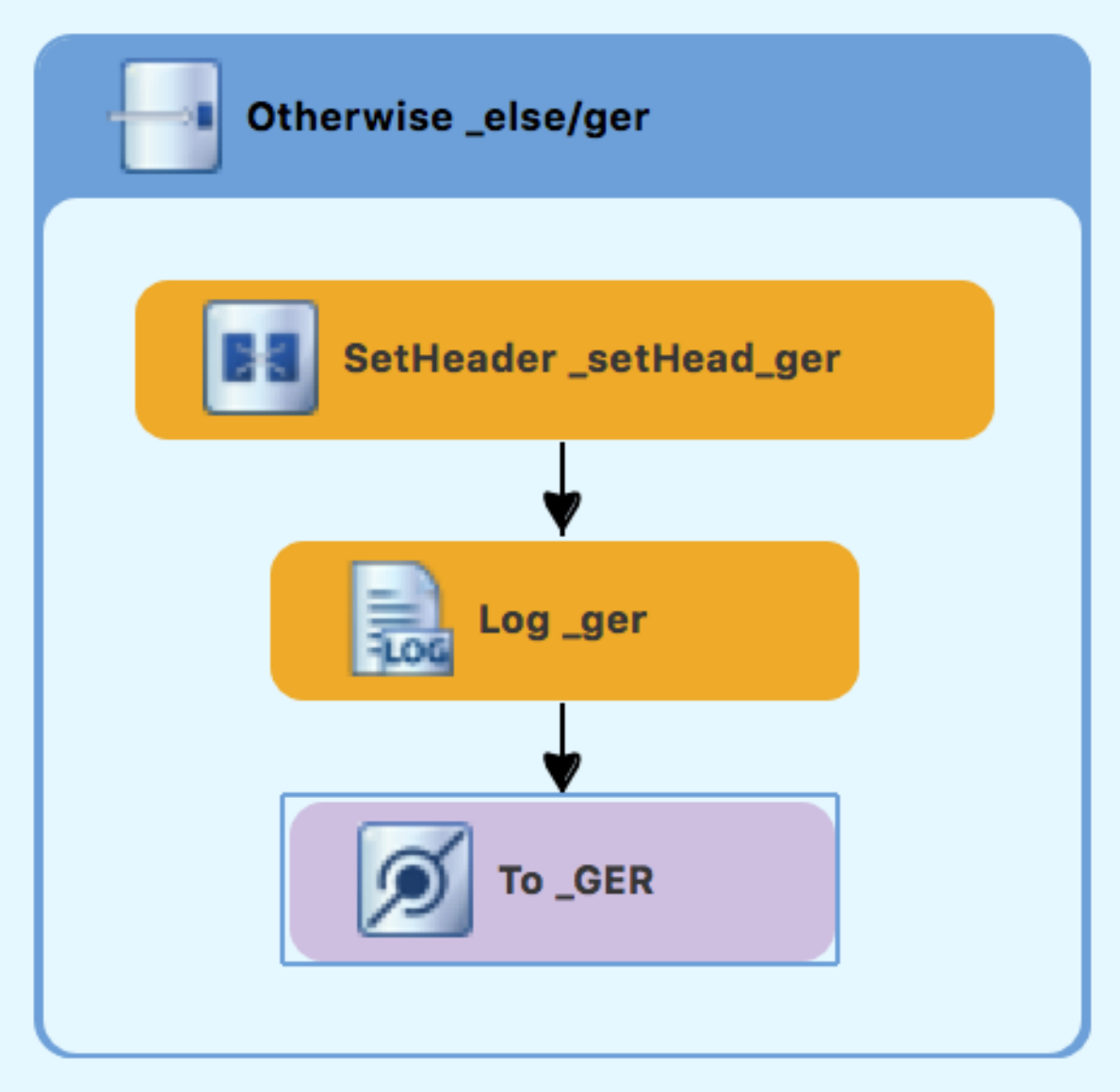


11. 在 canvas 上，选择 **Log_log3** 节点，以在 Properties 视图中打开其属性。
12. 在 Properties 视图中：
    - 在 Message 字段中，键入 **Valid order - 向德国客户发言。**
    - 在 Id 字段中，键入 **_ger。**
    - 将 **日志记录级别** 保留原样。
13. 在 Components drawer 中，选择一个 File 模式()，然后点击 **Log_ger** 节点下的文件模式。**Otherwise_else/ger** 容器扩展，以适应 **To_to1** 节点。如果需要，将连接器错误从 **SetHeader_setHead_ger** 节点拖到 **To_to1** 节点：



14. 在 canvas 上，选择 To_to1 节点，以在 Properties 视图中打开其属性。
15. 在 Properties 视图中：
    - 在 Uri 字段中，将 directoryName 替换为 target/messages/validOrders/Germany
    - 在 Id 字段中，键入 _GER。
16. 保存该文件。

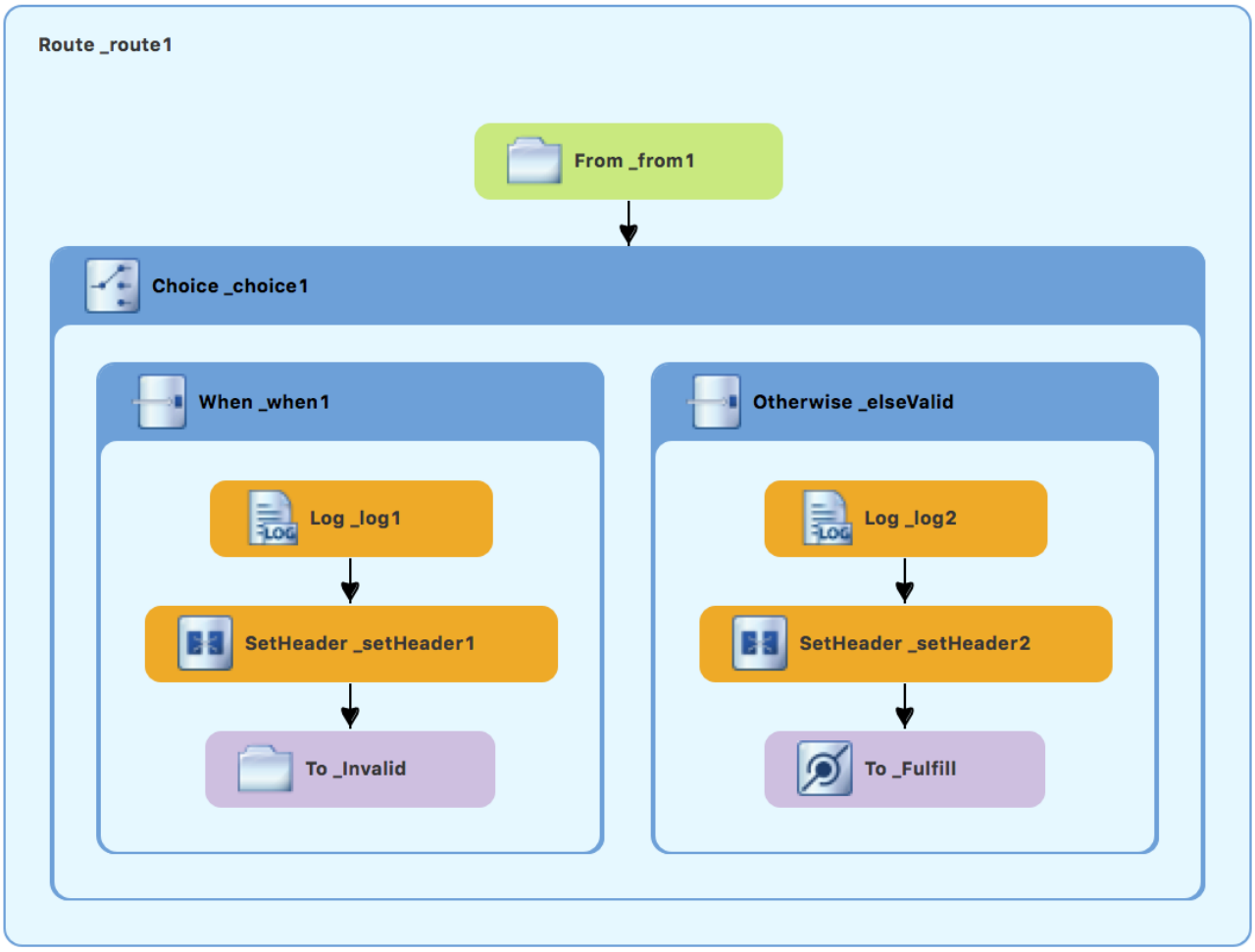
Route_route2 的德国分支应该类似如下：



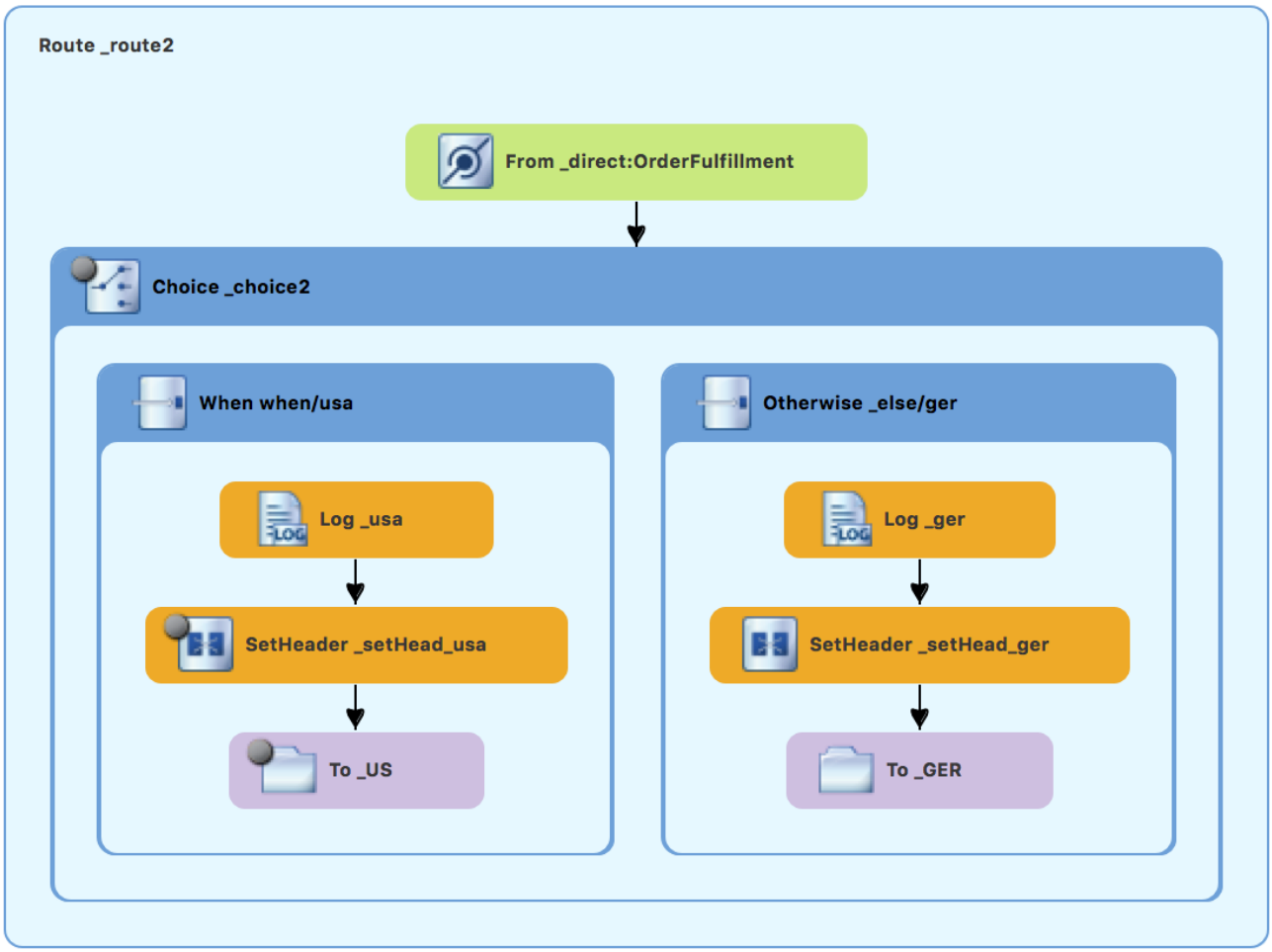
验证第二个路由

canvas 上的路由应类似如下：

完成的 route1


Route _route1
From _from1
Choice _choice1
When _when1
Log _log1
SetHeader _setHeader1
To _Invalid
Otherwise _elseValid
Log _log2
SetHeader _setHeader2
To _Fulfill

完成的 route2



在 canvas 底部的 Source 选项卡中，camelContext 元素的 XML 应该类似于例 6.1 “用于双路由内容的 XML” 所示：

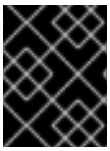
例 6.1. 用于双路由内容的 XML

```
<?xml version="1.0" encoding="UTF-8"?>

<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.osgi.org/xmlns/blueprint/v1.0.0
   https://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd
   http://camel.apache.org/schema/blueprint
   http://camel.apache.org/schema/blueprint/camel-blueprint.xsd">

   <camelContext id="_context1" xmlns="http://camel.apache.org/schema/blueprint">
     <route id="_route1" shutdownRoute="Default">
       <from id="_from1" uri="file:src/data?noop=true"/>
       <choice id="_choice1">
         <when id="_when1">
           <xpath>/order/orderline/quantity/text() &gt; 10</xpath>
           <log id="_log1" message="The quantity requested exceeds the maximum allowed -
contact customer."/>
           <setHeader headerName="Destination" id="_setHeader1">
             <constant>Invalid</constant>
           </setHeader>
           <to id="_Invalid" uri="file:target/messages/invalidOrders"/>
         </when>
```

```
            <otherwise id="_elseValid">
               <log id="_log2" message="This is a valid order - OK to process."/>
               <setHeader headerName="Destination" id="_setHeader2">
                  <constant>ReadyForDispatcher</constant>
               </setHeader>
               <to id="_Fulfill" uri="direct:OrderFulfillment"/>
            </otherwise>
         </choice>
      </route>
      <route id="_route2">
         <from id="_direct:OrderFulfillment" uri="direct:OrderFulfillment"/>
         <choice id="_choice2">
            <when id="when/usa">
               <xpath>/order/customer/country = 'USA'</xpath>
               <log id="_usa" message="Valid order - ship animals to USA customer"/>
               <setHeader headerName="Destination" id="_setHead_usa">
                  <constant>USA</constant>
               </setHeader>
               <to id="_US" uri="file:target/messages/validOrders/USA"/>
            </when>
            <otherwise id="_else/ger">
               <log id="_ger" message="Valid order - ship animals to Germany customer"/>
               <setHeader headerName="Destination" id="_setHead_ger">
                  <constant>Germany</constant>
               </setHeader>
               <to id="_GER" uri="file:target/messages/validOrders/Germany"/>
            </otherwise>
         </choice>
      </route>
   </camelContext>
</blueprint>
```



重要

如果工具将属性 shutdownRoute=" " 添加到第二个路由元素(<routeid="route2">)，请删除该属性。否则，ZooOrderApp 项目可能无法运行。

要确保您更新的项目按预期工作，请按照以下步骤执行：

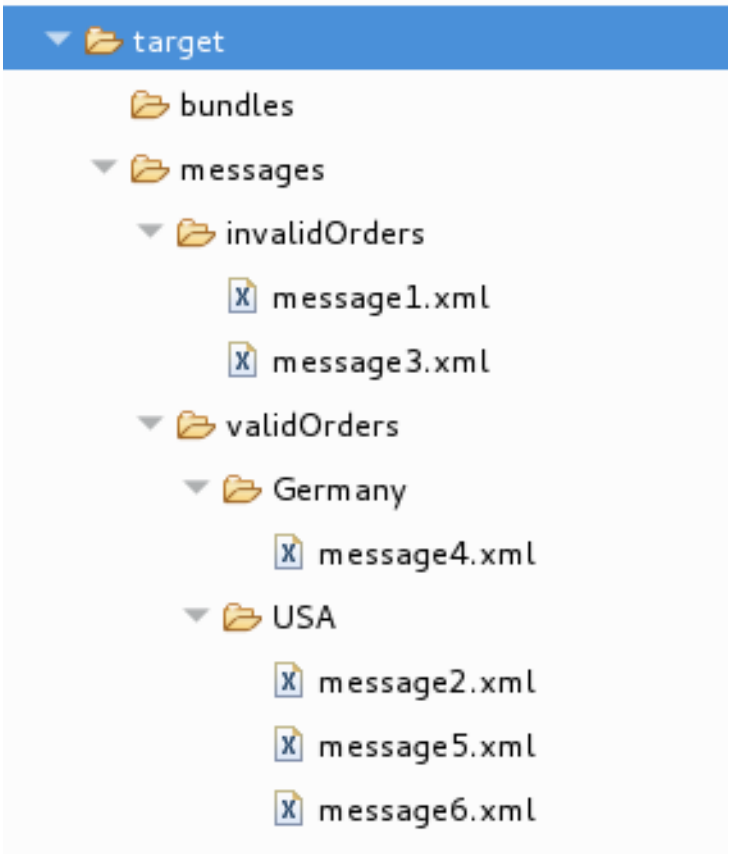
1. **以本地 Camel 上下文（不带测试）运行 ZooOrderApp/Camel Contexts/blueprint.xml。**
2. **检查控制台输出的末尾。您应该看到以下行：**

```
[ Blueprint Event Dispatcher: 1] BlueprintCamelContext          INFO  Route: _route1 started and consuming from: file://src/data?noop=true
[ Blueprint Event Dispatcher: 1] BlueprintCamelContext          INFO  Route: _route2 started and consuming from: direct://OrderFulfillment
[ Blueprint Event Dispatcher: 1] BlueprintCamelContext          INFO  Total 2 routes, of which 2 are started
[ Blueprint Event Dispatcher: 1] BlueprintCamelContext          INFO  Apache Camel 2.21.0.fuse-000112-redhat-3 (CamelContext: _context1) started in 0.318
[2.redhat.com:1099/jmxrmi/camel] DefaultManagementAgent         INFO  JMX Connector thread started and listening at: service:jmx:rmi:///jndi/rmi://ovpn-1
[1) thread #4 - file://src/data] _route1                        INFO  This is a valid order - OK to process.
[1) thread #4 - file://src/data] _route2                        INFO  Valid order - ship animals to USA customer
[1) thread #4 - file://src/data] _route1                        INFO  This is a valid order - OK to process.
[1) thread #4 - file://src/data] _route2                        INFO  Valid order - ship animals to Germany customer
[1) thread #4 - file://src/data] _route1                        INFO  This is a valid order - OK to process.
[1) thread #4 - file://src/data] _route2                        INFO  Valid order - ship animals to USA customer
[1) thread #4 - file://src/data] _route1                        INFO  The quantity requested exceeds the maximum allowed - contact customer.
[1) thread #4 - file://src/data] _route1                        INFO  This is a valid order - OK to process.
[1) thread #4 - file://src/data] _route2                        INFO  Valid order - ship animals to USA customer
[1) thread #4 - file://src/data] _route1                        INFO  The quantity requested exceeds the maximum allowed - contact customer.
```

3. **检查目标目标文件夹以验证路由是否已正确执行：**
   a. **在 Project Explorer 中，右键单击 ZooOrderApp，然后选择 Refresh。**

b. **展开 target/messages/ 文件夹。**
message*.xml 文件应该在您的目的地中分散，如下所示：

图 6.1. Project Explorer 中的目标消息目的地



后续步骤

在下一教程 第 7 章 *调试路由上下文* 中，您将了解如何使用 Fuse 工具调试器。

第 7 章 调试路由上下文

本教程介绍了如何使用 Camel 调试器查找本地运行路由上下文的逻辑错误。

目标

在本教程中，您将完成以下任务：

- 在两个路由中感兴趣的节点上设置断点
- 在 Debug 透视图中，逐步浏览路由并检查消息变量的值
- 再次逐步浏览路由，更改消息变量的值并观察其效果

先决条件

要启动本教程，您需要从以下之一生成的 ZooOrderApp 项目：

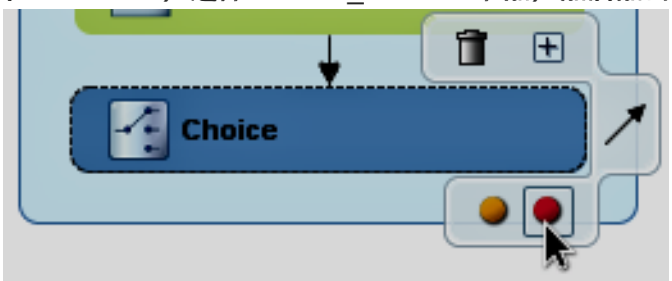
- 完成 第 6 章 *在路由上下文中添加另一个路由* 教程。
  or
- 完成 第 2 章 *设置您的环境* 教程，并将项目的 blueprint.xml 文件替换为提供的 blueprintContexts/blueprint3.xml 文件，如 "关于资源文件"一节 所述。

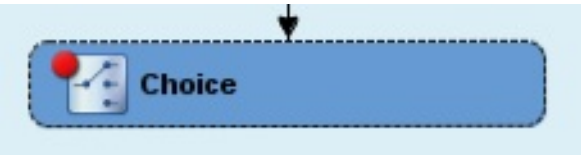
设置断点

在 Debugger 中，您可以设置条件和无条件断点。在本教程中，您只设置无条件断点。要了解如何设置条件断点（在调试会话过程中满足特定条件时触发），请参阅 工具用户指南。

设置无条件断点：

1. 如有必要，在路由编辑器中打开 ZooOrderApp/src/main/resources/OSGI-INF/blueprint/blueprint.xml。
2. 在 Project Explorer 中，展开 Camel Contexts → src/main/resources/OSGI-INF/blueprint/blueprint.xml 以公开这两个路由条目。
3. 双击 Route_route1 条目，在 Design 选项卡中将重点切换为 Route_route1。
4. 在 canvas 上，选择 Choice_choice1 节点，然后点击其  图标设置无条件断点：





注意

在路由编辑器中，您可以通过分别点击节点的 图标**或其** 图标**来禁用或删除特定的断点。您可以通过右击** canvas **并选择** Delete all breakpoints **来删除所有设置断点。**

5. **在以下 Route_Route1 节点上设置无条件断点：**
   - **Log_log1**
   - **SetHeader_setHeader1**
   - **To_Invalid**
   - **Log_log2**
   - **SetHeader_setHeader2**
   - **To_Fulfill**
6. **在** Project Explorer **中，双击 src/main/resources/OSGI-INF/blueprint 下的 Route_route2，以打开** canvas **上的 Route_route2。**
7. **在以下 Route_Route2 节点上设置无条件断点：**
   - **Choice_choice2**
   - **SetHeader_setHead_usa**
   - **Log_usa**
   - **OT_US**
   - **SetHeader_setHead_ger**
   - **Log_ger**
   - **OT_GER**

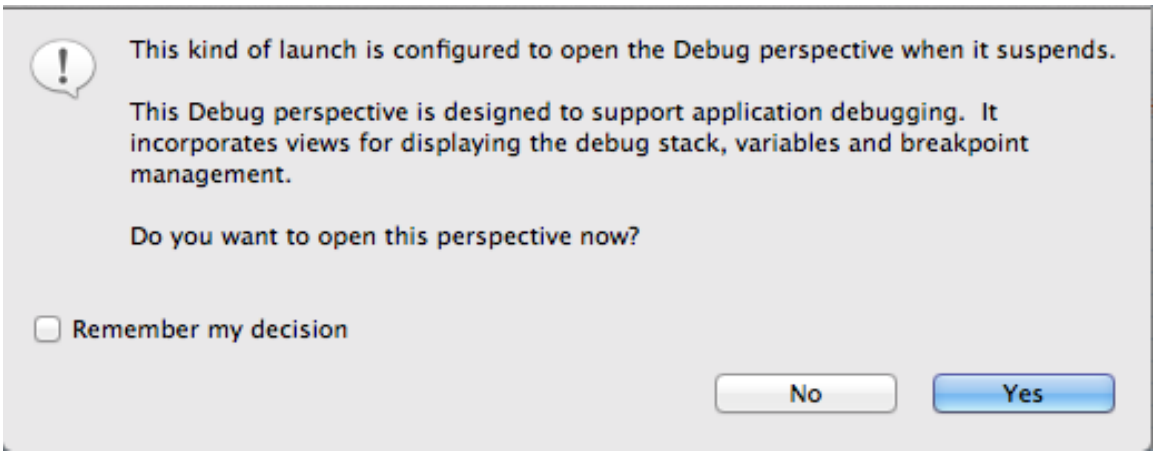
逐步浏览路由上下文

您可以通过两种方式逐步浏览路由上下文：

- Step over ( )- **无论断点是什么，都恢复到路由上下文中的下一个执行节点。**
- resume ( )- **跳到路由上下文中的下一个活跃断点。**

1. **在** Project Explorer **中，展开 ZooOrderApp 项目的 Camel Contexts 文件夹，以公开 blueprint.xml 文件。**

2. 右键单击 **blueprint.xml** 文件以打开其上下文菜单，然后点击 Debug As → Local Camel Context （不带测试）。
Camel debugger 会在它遇到的第一个断点挂起执行，并询问您是否要打开 Debug 视角：

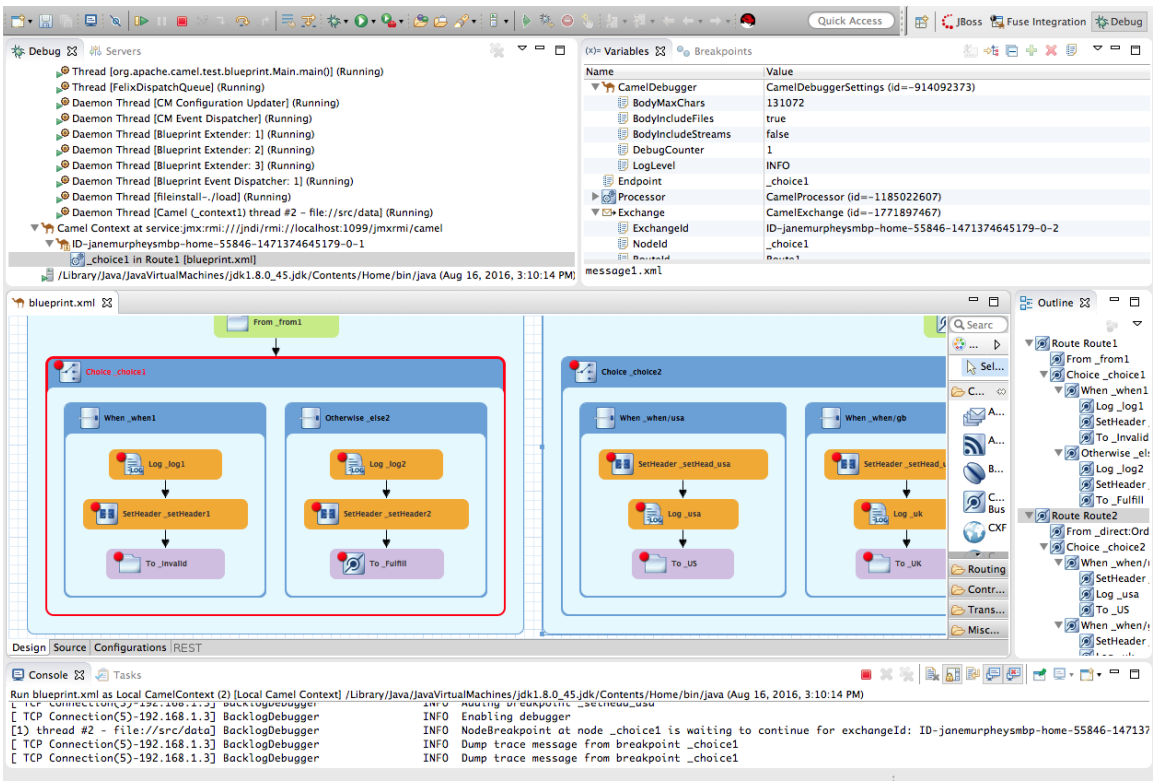


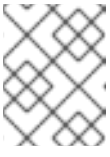
3. 单击 Yes。

注意

如果您单击 No，则确认窗格会出现多次。在第三个 refusal 后，它会消失，Camel debugger 会恢复执行。要与调试器交互，您需要通过点 Window → Open Perspective → >Debug 来打开 Debug 视角。

Debug perspective 将打开，其路由上下文在 _route1 [blueprint.xml] 中的 _choice1 中暂停，如 Debug 视图中所示：



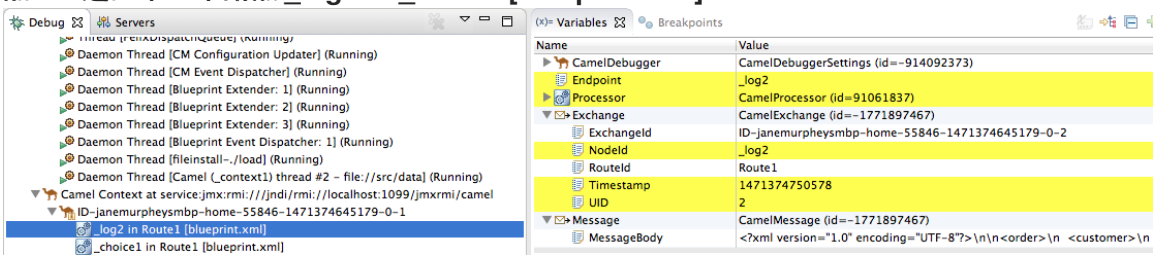


注意

断点在调试器自动恢复前最多需要五分钟，移至下一个断点或路由上下文的末尾（以下一步为准）。

4. 在 Variables 视图中，扩展节点以公开每个节点的可用变量和值。
当您逐步执行路由上下文时，其值自上次断点以黄色突出显示时所更改的变量。您可能需要扩展每个断点的节点，以显示已更改的变量。

5. 点 进入下一个断点 _log2 in _route1 [blueprint.xml]:



6. 展开 Variables 视图中的节点，以检查自 Route1 [blueprintxt.xml] 中 _choice1 的最后一个断点后更改的变量。

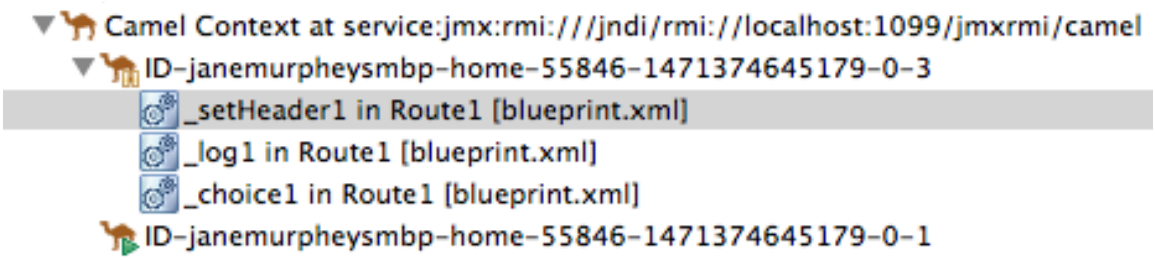
7. 点 进入下一个断点 _setHeader2 [blueprint.xml]。
检查自 Route1 [blueprint.xml] 中 _log2 的断点以来更改的变量（高亮）。

8. 在 Debug 视图中，单击 _route1 [blueprint.xml] 中的 _log2，以使用 _route1 [blueprint.xml] 中的 breakpoint _log2 中的变量值填充 Variables 视图。
在 Debug 视图中，您可以在同一消息流中的断点间切换，以快速比较和监控 Variables 视图中的变量值。

注意

消息流的长度可能会有所不同。对于传输 Route_route1 的 InvalidOrders 分支的消息，消息流是短的。对于传输 Route_route1 分支（继续到 Route_route2） 的消息，消息流会更长。

9. 继续逐步浏览路由上下文。当一个消息完成路由上下文并进入路由上下文时，新消息流会出现在 Debug 视图中，标记为一个新的面包屑图标 ID：



在这种情况下，ID-janemurpheysmbp-home-55846-1471374645179-0-3 标识第二个消息流，对应于输入路由上下文的 message2.xml。面包屑导航栏 ID 递增 2。

注意

交换和消息 ID 相同，在整个消息通过路由上下文期间保持不变。其 ID 由消息流的面包屑导航栏 ID 构建，以 1 递增。因此，如果是 message2.xml，其 ExchangeId 和 MessageId 是 ID-janemurpheysmbp-home-55846-1471374645179-0-4。

10.

当 message3.xml 在 _route_route1 [blueprint.xml] 中输入 breakpoint _choice1 时，检查 Processor 变量。显示的值是 message1.xml 和 message2.xml 的总指标，之前传输了路由上下文：

(x)= Variables | Breakpoints | Expressions

Name	Value
CamelDebugger	CamelDebuggerSettings (id=-914092373)
BodyMaxChars	131072
BodyIncludeFiles	true
BodyIncludeStreams	false
DebugCounter	13
LogLevel	INFO
Endpoint	_choice1
Processor	CamelProcessor (id=-1185022607)
ProcessorId	_choice1
RouteId	Route1
CamelId	_context1
CompletedExchanges	2
FailedExchanges	0
TotalExchanges	2
Redeliveries	0
ExternalRedeliveries	0
HandledFailures	0
LastProcessingTime	84876
MinProcessingTime	84876
AverageProcessingTime	122602
MaxProcessingTime	160329
TotalProcessingTime	245205
Exchange	CamelExchange (id=-1771897463)

计时指标以毫秒为单位。

11.

继续通过路由上下文逐步执行每个消息，检查每个处理步骤的变量和控制台输出。当 message6.xml 在 Route2 [blueprint.xml] 中输入 breakpoint To_GER 时，调试器开始关闭面包屑导航栏线程。

12.

在菜单栏中，点

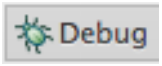
终止 Camel 调试器。控制台会终止，但您必须手动清除输出。

注意

在 Debug 视图中的 Camel Context 节点下选择了线程或端点时，您必须点

两次 - 来终止线程或端点，第二个用于终止 Camel 上下文，因此会话。

13. 在 Menu 栏中，右键单击



以打开上下文菜单，然后选择 Close 以关闭 Debug 视角。

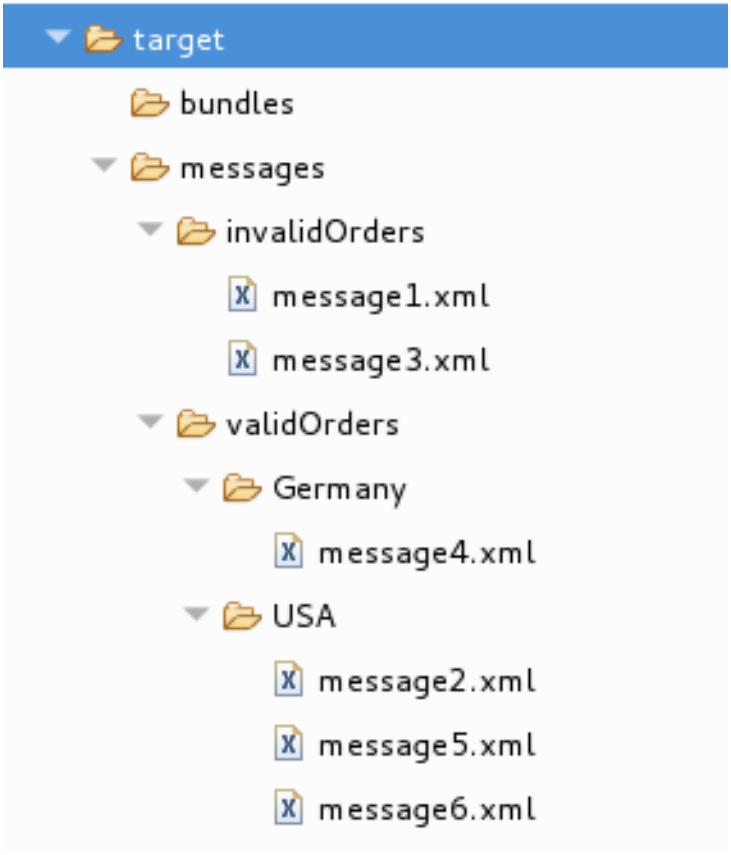
CodeReady Studio 会自动返回到您启动 Camel 调试器的视角。

14. 在 Project Explorer 中，右键单击项目，然后选择 Refresh 以刷新显示。

注意

如果您预先终止会话，在传输路由上下文的所有消息之前，您可能会在 ZooOrderApp/src/data 文件夹下看到，如下所示：message3.xml.camelLock。您需要将它移除，然后对项目再次运行 debugger。为此，请双击 .camelLock 消息打开其上下文菜单，然后选择 Delete。当被要求时，单击 OK 以确认删除。

15. 扩展 ZooOrderApp/target/messages/ 目录，以检查消息是否已发送到其预期的目的地：



将路由上下文保留原样，且所有断点都设置并启用。

更改变量的值

在本节中，您将变量添加到监视列表中，以便轻松检查其值在通过路由上下文时如何改变。您可以更改消息正文中的变量的值，然后观察更改如何通过路由上下文影响消息路由。

1. 要在 ZooOrderApp 项目上重新运行 Camel debugger，请右键单击 blueprint.xml 文件，然后单击 Debug As → Local Camel Context （没有测试）。

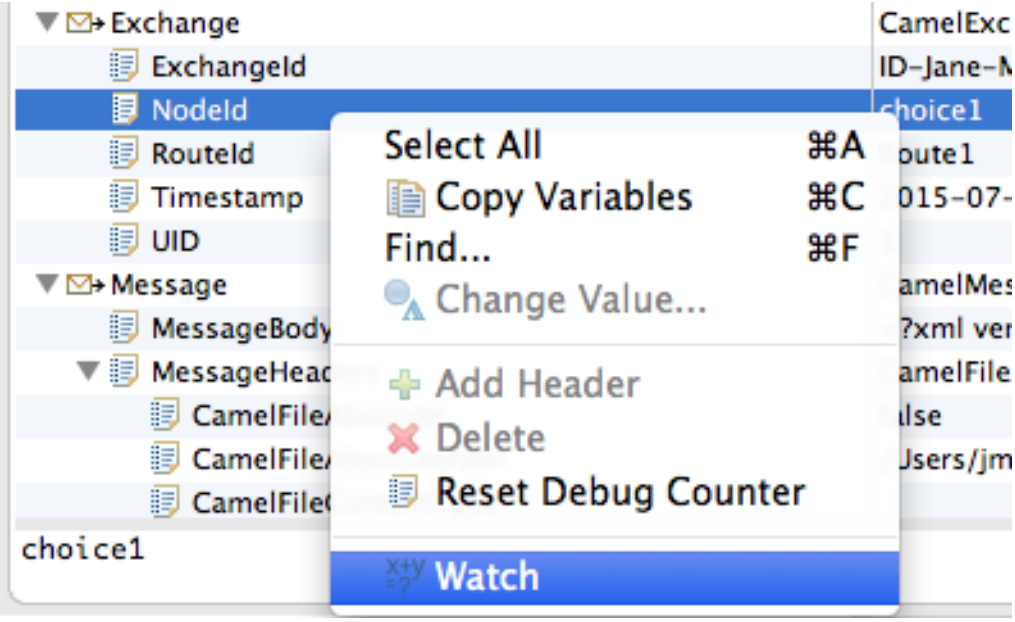
2. 在第一个断点上停止了 message1，在 _route1 [blueprint.xml] 中 停止 _choice1，将变量 NodeId 和 RouteId （在 Exchange category 中）和 MessageBody 和 CamelFileName （在 Message 类别中）添加到监视列表中。

   对于每个变量：

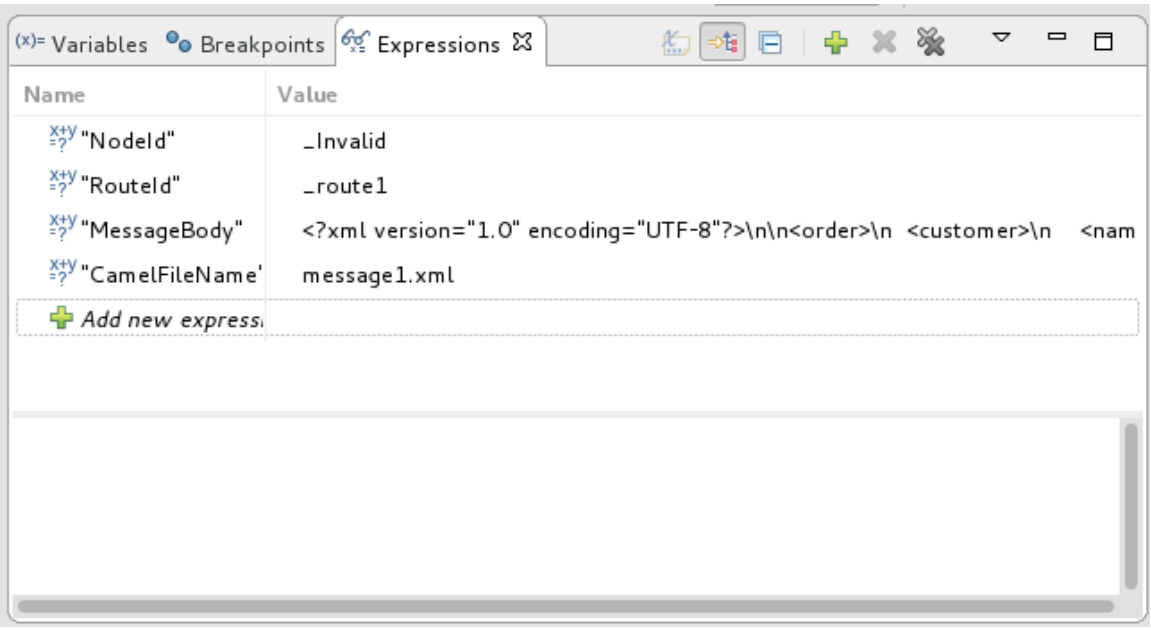
   a. 在 Variables 视图中，扩展适当的类别来公开目标变量：

b.

右键点击变量（本例中为 NodeId），以打开上下文菜单并选择 Watch ：



Expressions 选项卡将打开，列出您选择的要监视的变量：



注意

通过创建监视列表，您可以轻松检查感兴趣的多个变量的当前值。

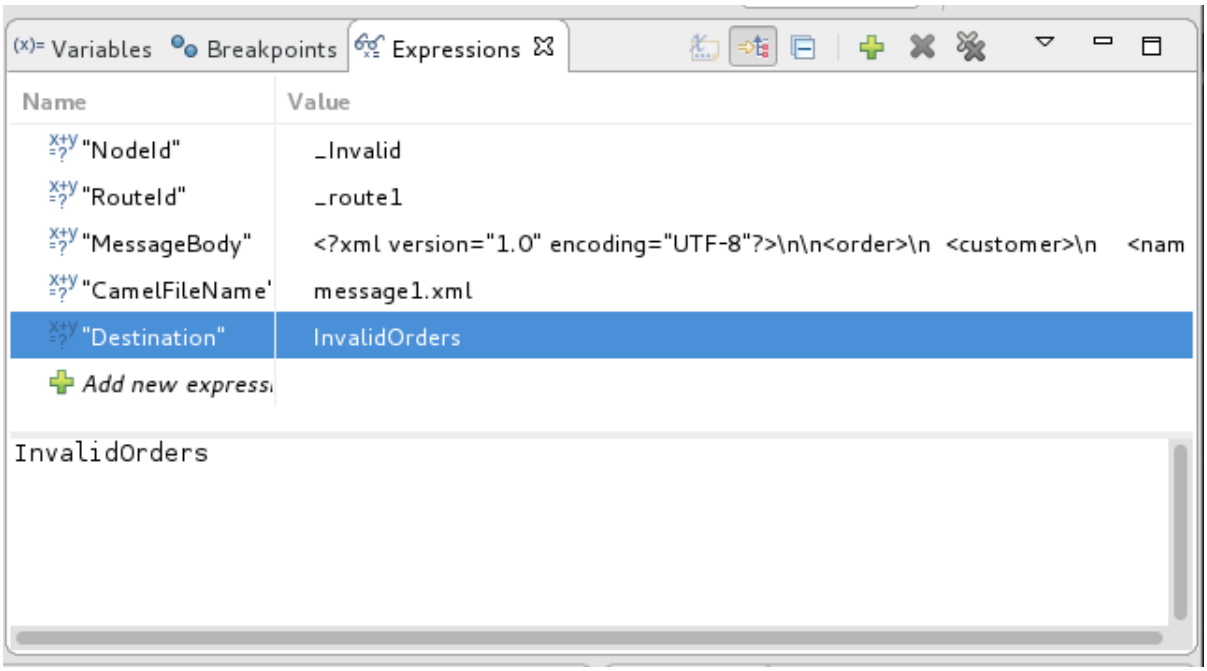
3.

通过路由上下文的第 message1 步，直到到达 _route1 [blueprint.xml] 中的第四个断点 _Fulfill。

4. 在 Variables 视图中，展开 Message 类别。

5. 将变量 Destination 添加到 watch 列表中。

   Expressions 视图现在应包含这些变量：



注意

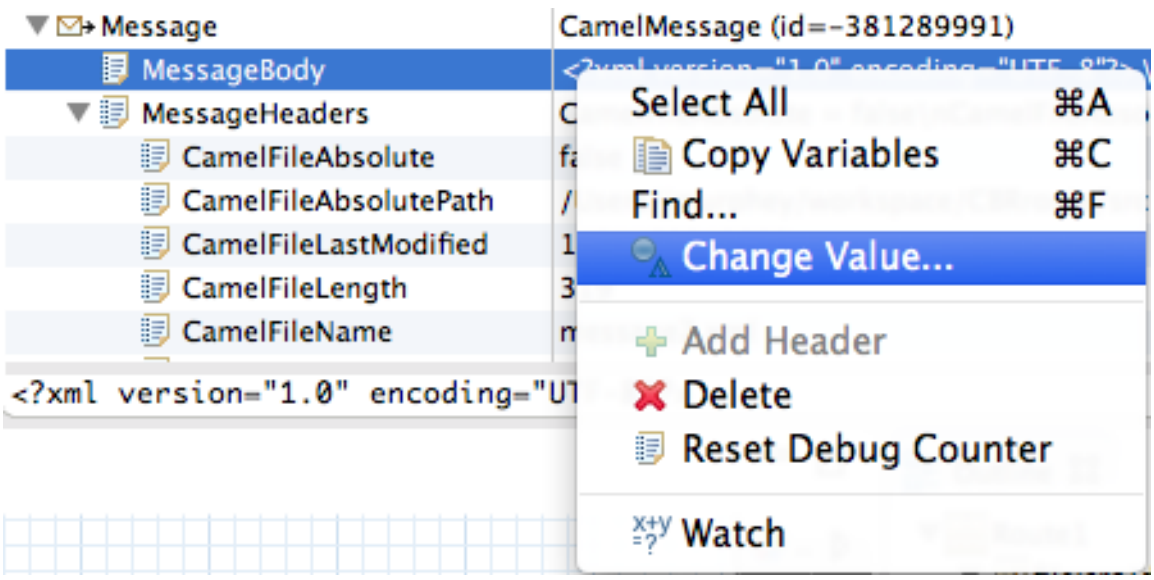
- 变量列表下的窗格显示所选变量的值。
- Expressions 视图会保留添加到列表中的所有变量，直到您明确删除它们。

6. 步骤 message1 通过路由上下文的其余部分，然后逐步 消息2 所有方法。

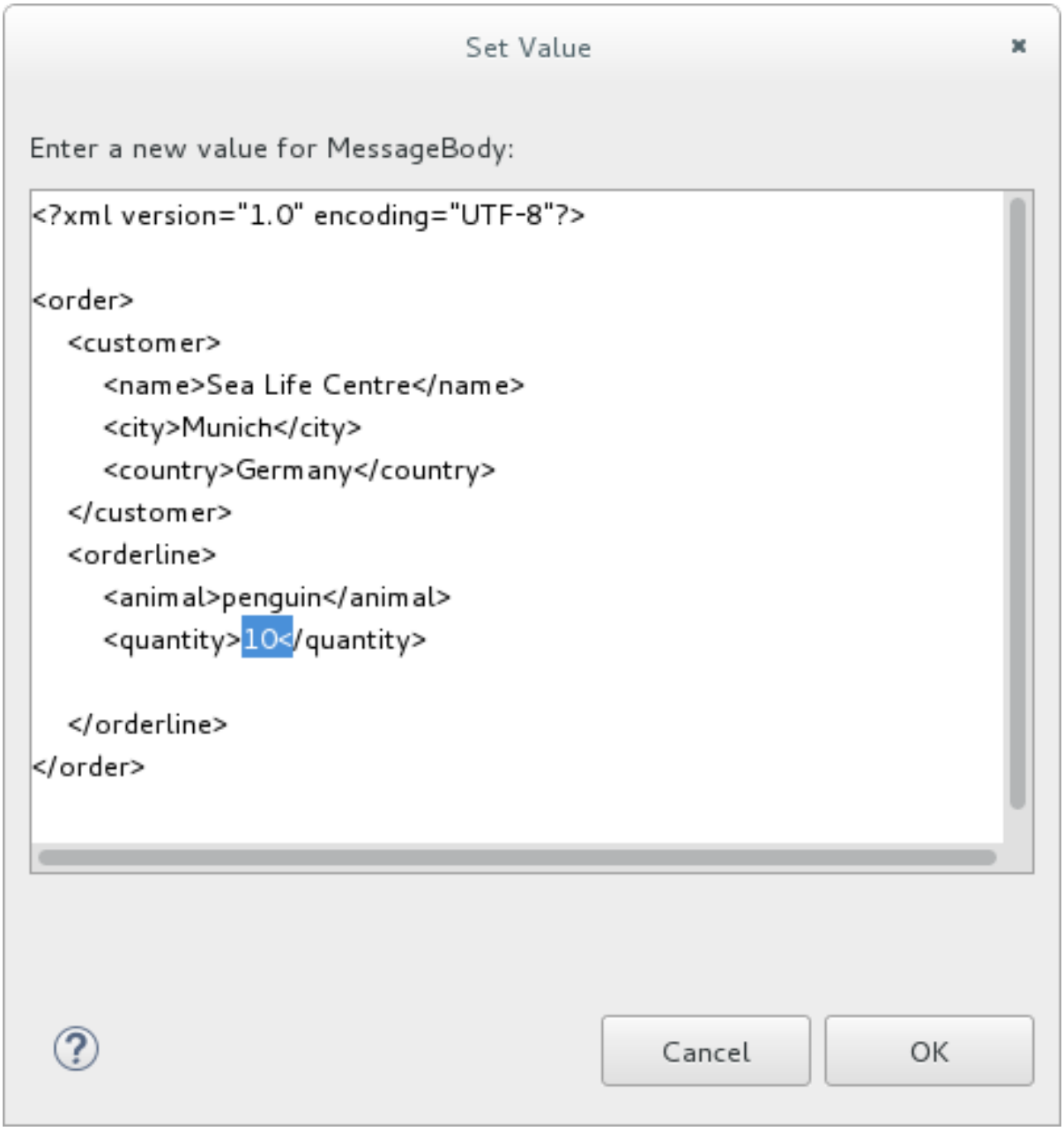
7. 在 _route1 [blueprint.xml] 中的 _choice1 中停止 message3。

8. 在 Variables 视图中，展开 Message 类别以公开 MessageBody 变量。

9. 右键点击 **MessageBody** 以打开其上下文菜单，然后选择 **Change Value** ：



10. 将 **数量** 的值从 **15** 改为 **10** （将其从无效顺序更改为有效顺序）：

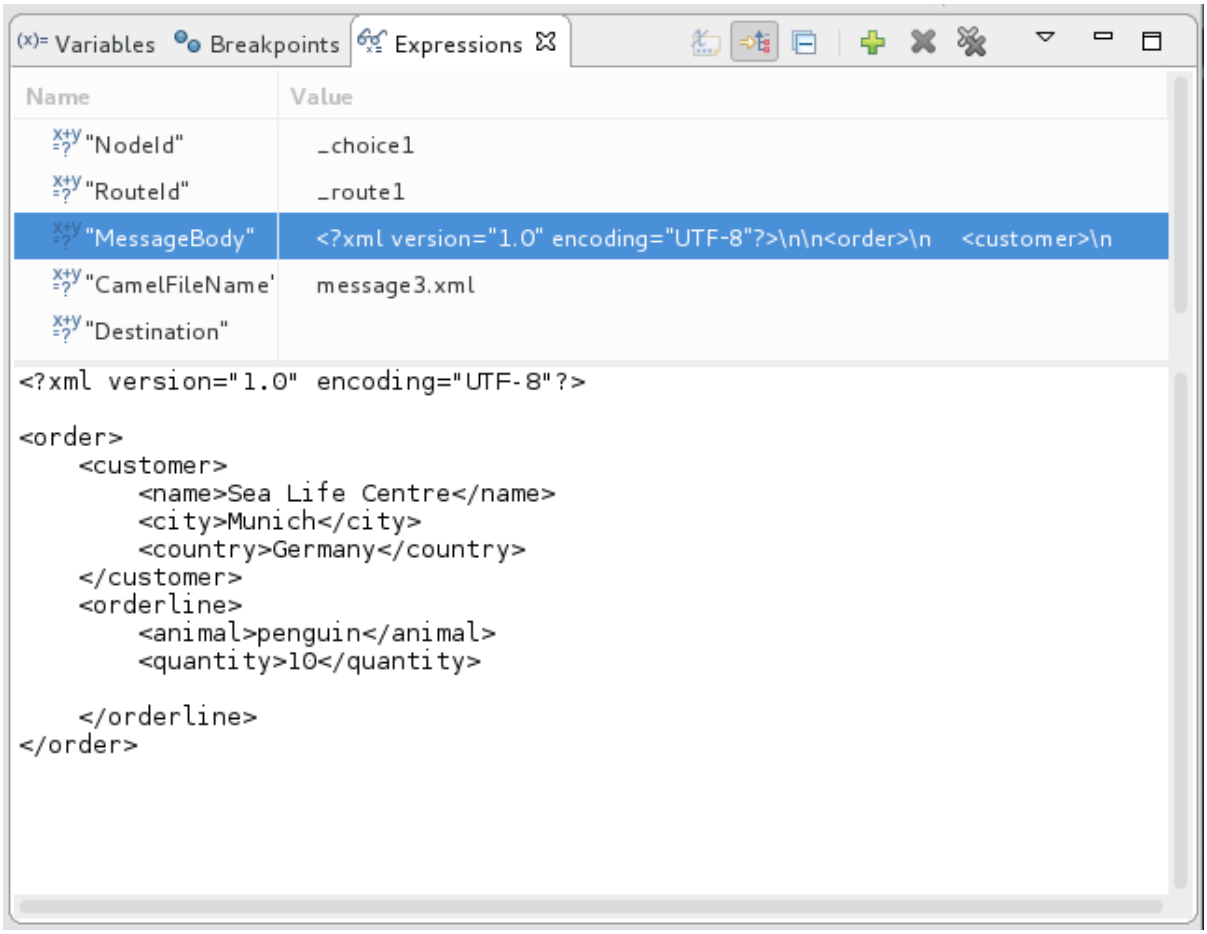


这只更改内存值（它不会编辑 **message3.xml** 文件）。

11. 点击 **确定**。

12. 切换到 **Expressions** 视图，然后选择 **MessageBody** 变量。

变量列表下的窗格显示 **message3** 的整个正文，从而可以轻松地检查订购项目的当前值：



13. **点**

进入下一个断点。

message3 现在遵循 To_Fulfill 和 Route_ route2 的分支，而不是遵循发送到To_Fulfill 和 Route_route2 的分支。

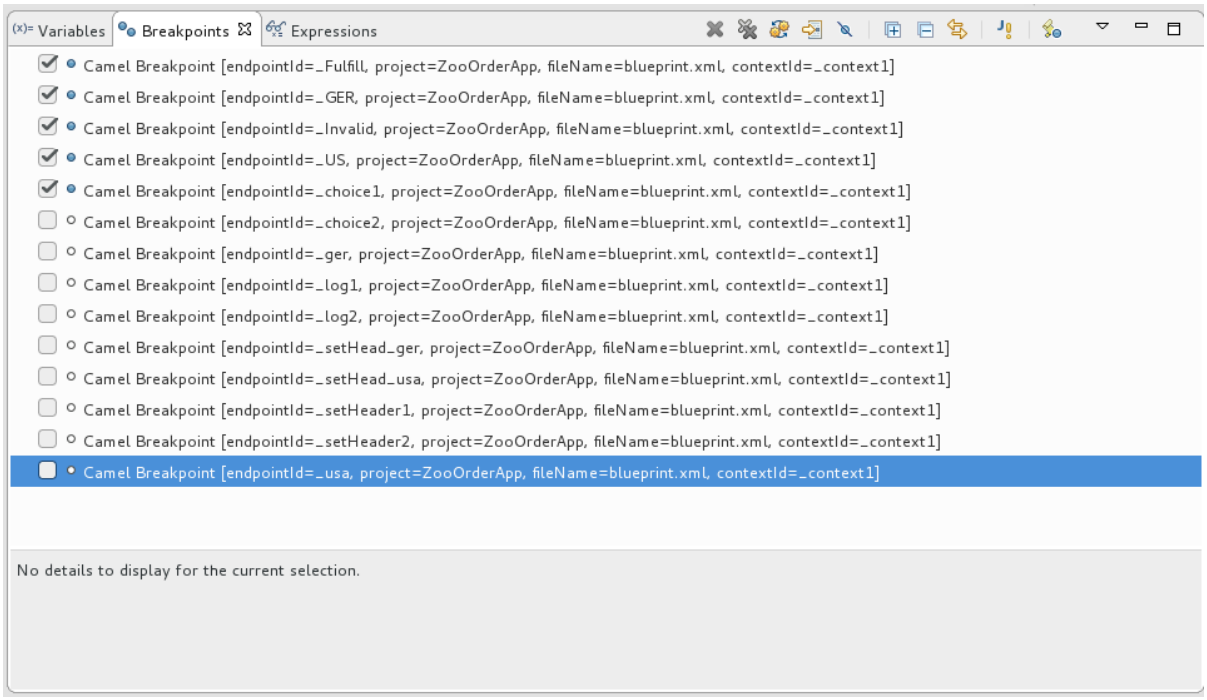
缩小 CAMEL DEBUGGER 的重点

您可以通过禁用和重新启用断点来临时缩小，然后重新扩展调试器的焦点：

1. **Step message4 through the routing context, check the Debug view, the Variables view, and the Console output in each step.**

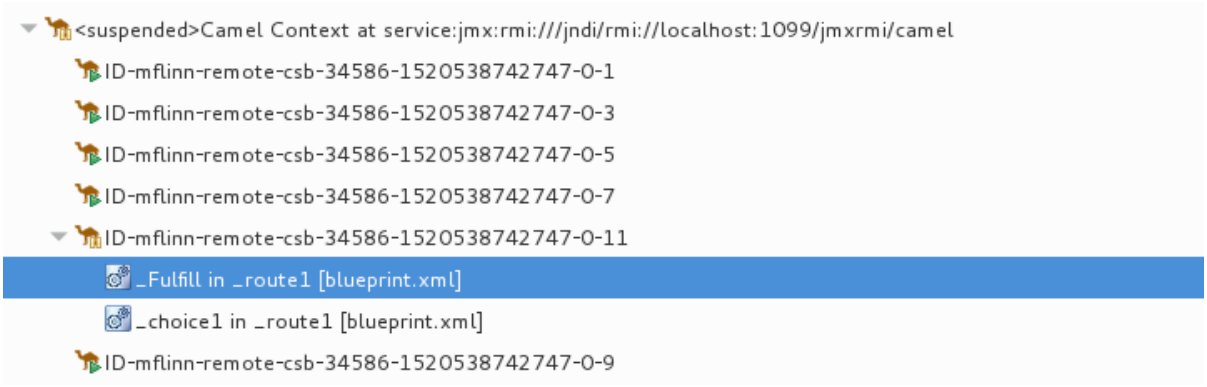
2. **在 _route1 [blueprint.xml] 中的 _choice1 中停止 message4。**

3. **切换到 Breakpoints 视图，然后清除 _choice1 下列出的断点旁边的每个复选框。清除断点的复选框会临时禁用它。**



4. **点**

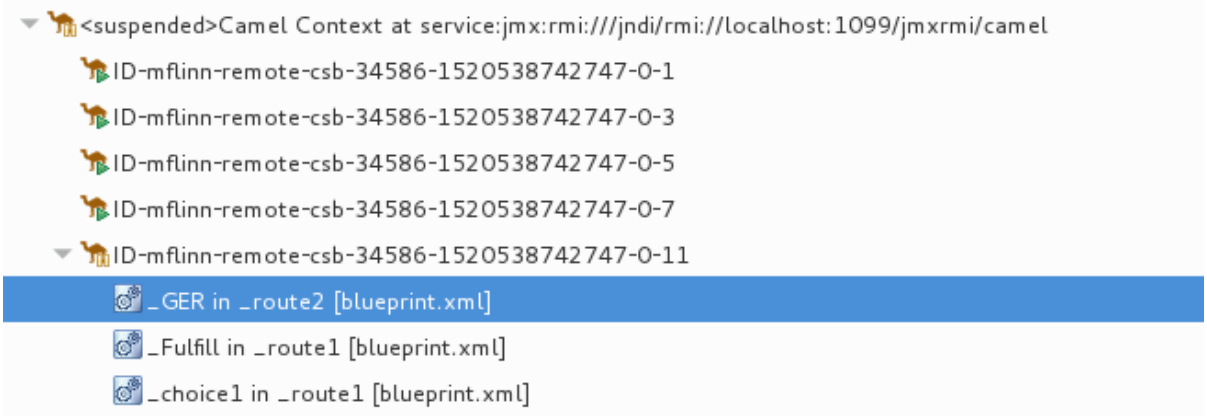
进入下一个断点：



调试器跳过禁用的断点并跳转到 _route1 [blueprint.xml] 中的 _FulFill。

5. **再次点**

来进入下一个断点：



debugger 在 _route2 [blueprint.xml] 中跳转到 _GER。

6. **点**

，通过路由上下文快速步进 message5 和 message6。

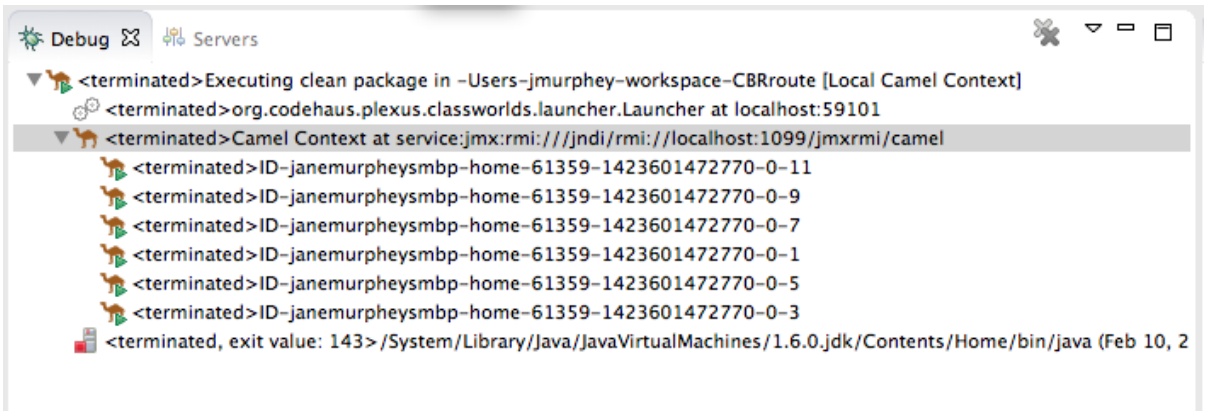
7. **切换到 Breakpoints 视图，再选中所有断点旁边的框以重新启用它们。**

验证更改消息变量值的影响

停止调试器并检查更改 'message1's quantity 变量值的结果：

1. **在工具栏中，点**

终止 Camel 调试器：



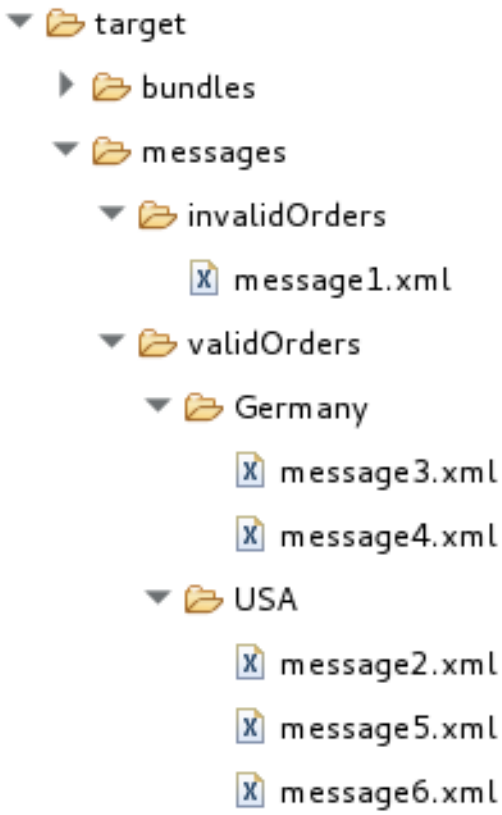
2. **点击控制台的**


按钮清除输出。

3. 关闭 Debug 透视图，再返回到您启动 Camel 调试器的视角。

4. 在 Project Explorer 中，刷新显示。

5. 扩展 ZooOrderApp/target/messages/ 目录，以检查消息是否按预期发送：



您应该会看到只有 message1 发送到 invalidOrders，并且 message3.xml 会出现在 validOrders/Germany 文件夹中。

后续步骤

在 第 8 章 *通过路由追踪消息* 教程中，您可以通过路由上下文跟踪信息，以确定您可以在什么位置优化并微调路由上下文的性能。

第 8 章 通过路由追踪消息

通过追踪，您可以截获消息，因为它从一个节点路由到另一个节点。您可以通过路由上下文跟踪消息，以查看您可以在什么位置优化并微调路由上下文的性能。本教程介绍了如何通过路由跟踪消息。

目标

在本教程中，您将完成以下任务：

- 在 Fuse Integration 视角中运行 ZooOrderApp
- 在 ZooOrderApp上启用追踪
- 将消息放到 ZooOrderApp 中，并通过所有路由节点跟踪它们

先决条件

要启动本教程，您需要从以下之一生成的 ZooOrderApp 项目：

- 完成 第 6 章 *在路由上下文中添加另一个路由* 教程。

  or

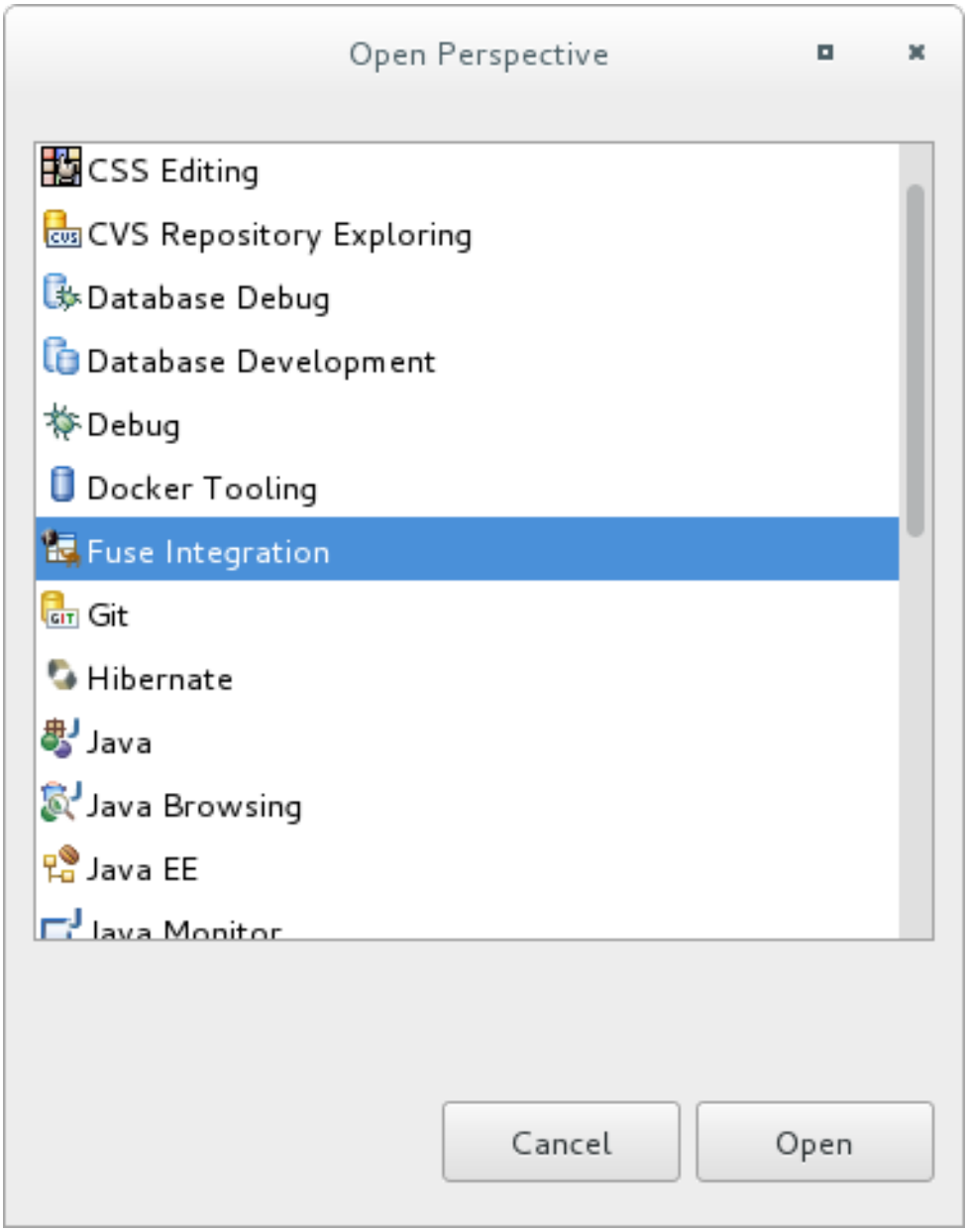
- 完成 第 2 章 *设置您的环境* 教程，并将项目的 blueprint.xml 文件替换为提供的 blueprintContexts/blueprint3.xml 文件，如 “关于资源文件”一节 所述。

设置 FUSE 集成视角

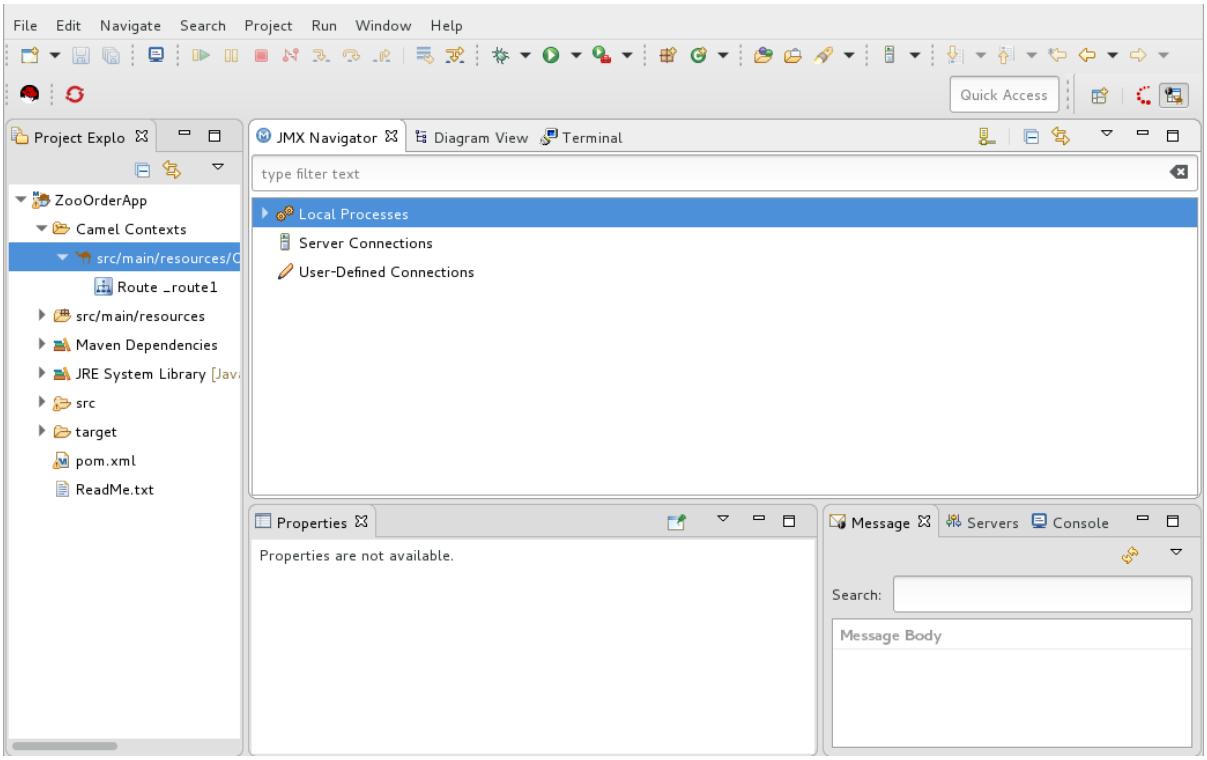
要设置工作区以方便消息追踪：

1. 点工具栏右侧的


按钮，然后从列表中选择 **Fuse Integration** ：

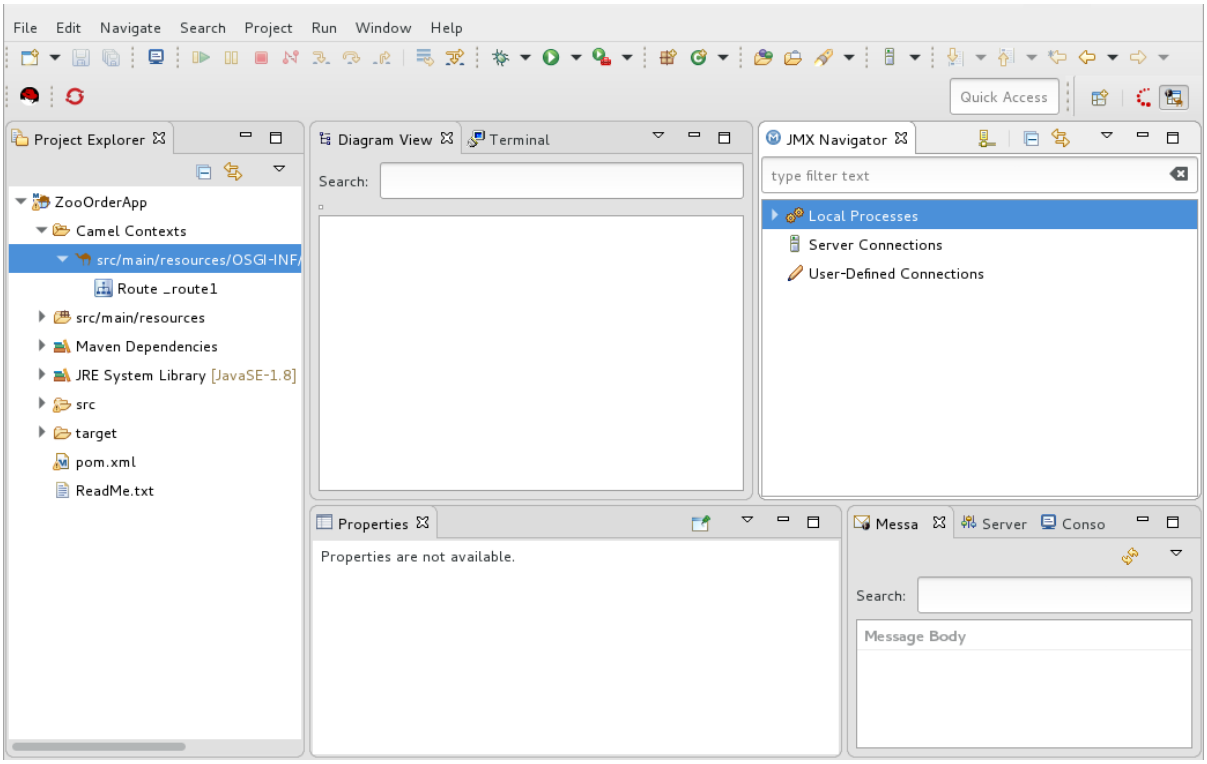


Fuse Integration 视角在默认布局中打开：



2.

将 JMX Navigator 选项卡拖到 Terminal 选项卡的最右侧，并将其保存在：

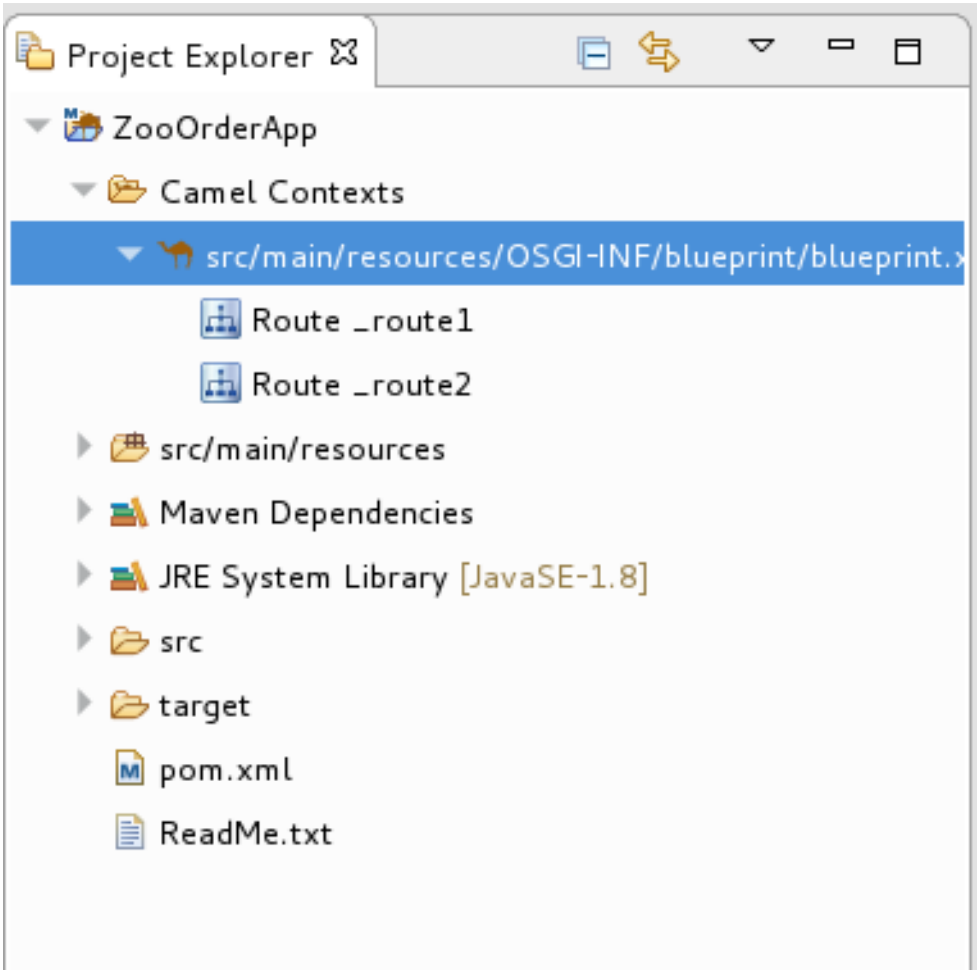


此安排为 Diagram View 提供了更多空间，以便以图形方式显示路由上下文的节点，这样您可以更轻松地跟踪消息在遍历路由上下文中的路径。

注意

为了轻松访问路由上下文 .xml 文件，特别是项目包含多个上下文时，工具会在 Project Explorer 的 Camel Contexts 文件夹下列出它们。

此外，路由上下文中的所有路由都直接显示为图标，在其上下文文件条目下直接显示。要在 canvas 上的路由上下文中显示单个路由，请在 Project Explorer 中双击其图标。要显示路由上下文中的所有路由，请双击上下文文件条目。



启动消息追踪

在 ZooOrderApp 项目中启动消息追踪：

1. 在 Project Explorer 中，展开 ZooOrderApp 项目，以公开 src/main/resources/OSGI-INF/blueprint/blueprint.xml。

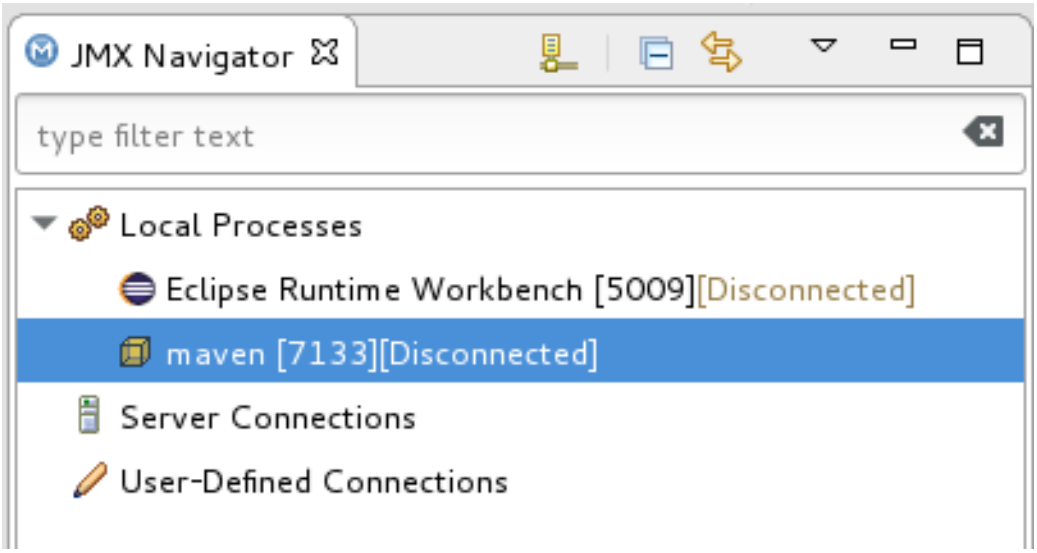
2. 右键单击 src/main/resources/OSGI-INF/blueprint/blueprint.xml 以打开上下文菜单。

3. 选择 **Run As** → **Local Camel Context （没有测试）**。

注意

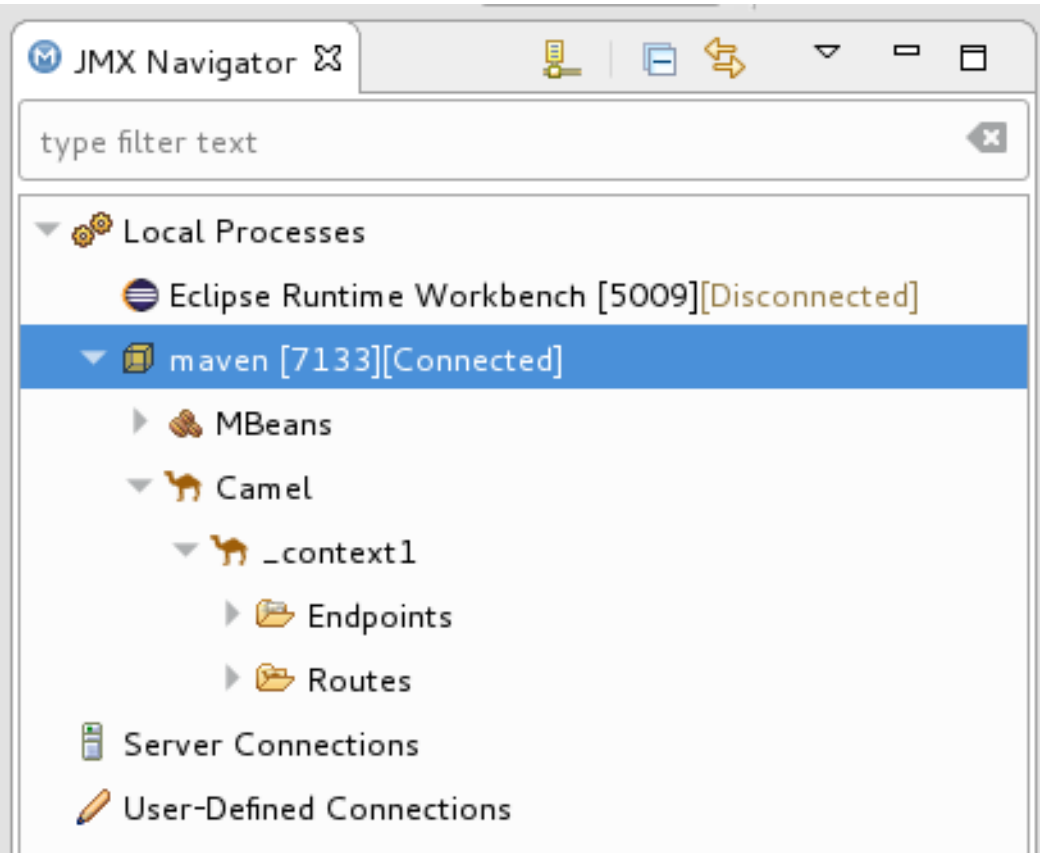
如果选择 **Local Camel Context**，则工具将恢复到无需测试的情况下运行，因为您尚未为 **ZooOrderApp** 项目创建了 **JUnit** 测试。稍后您将在 第 9 章 *使用 JUnit 测试路由* 中执行此操作。

4. **在 JMX 导航器 中，展开 Local process。**

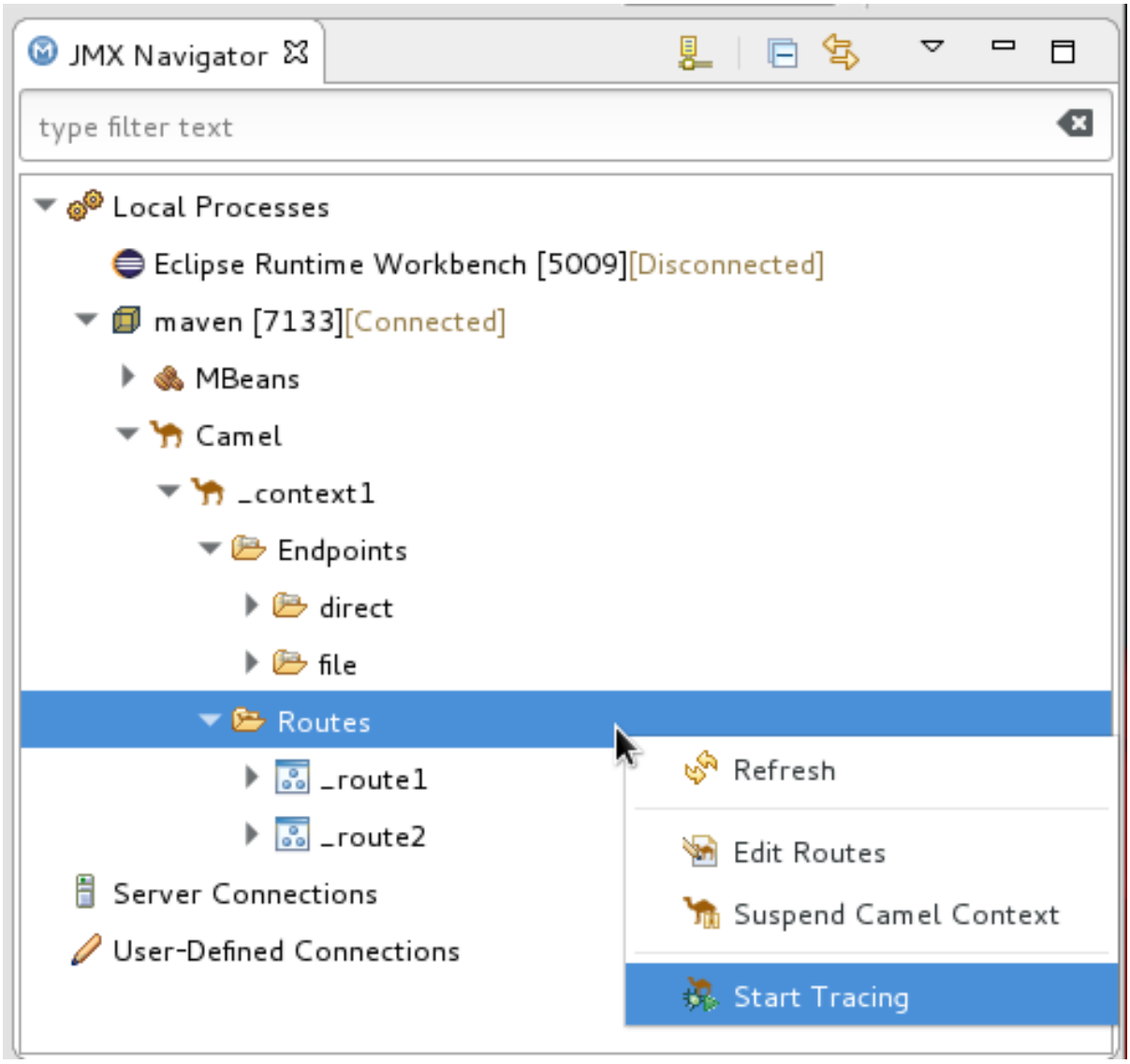


5. **右键单击 maven [ID] 节点，然后选择 Connect。**

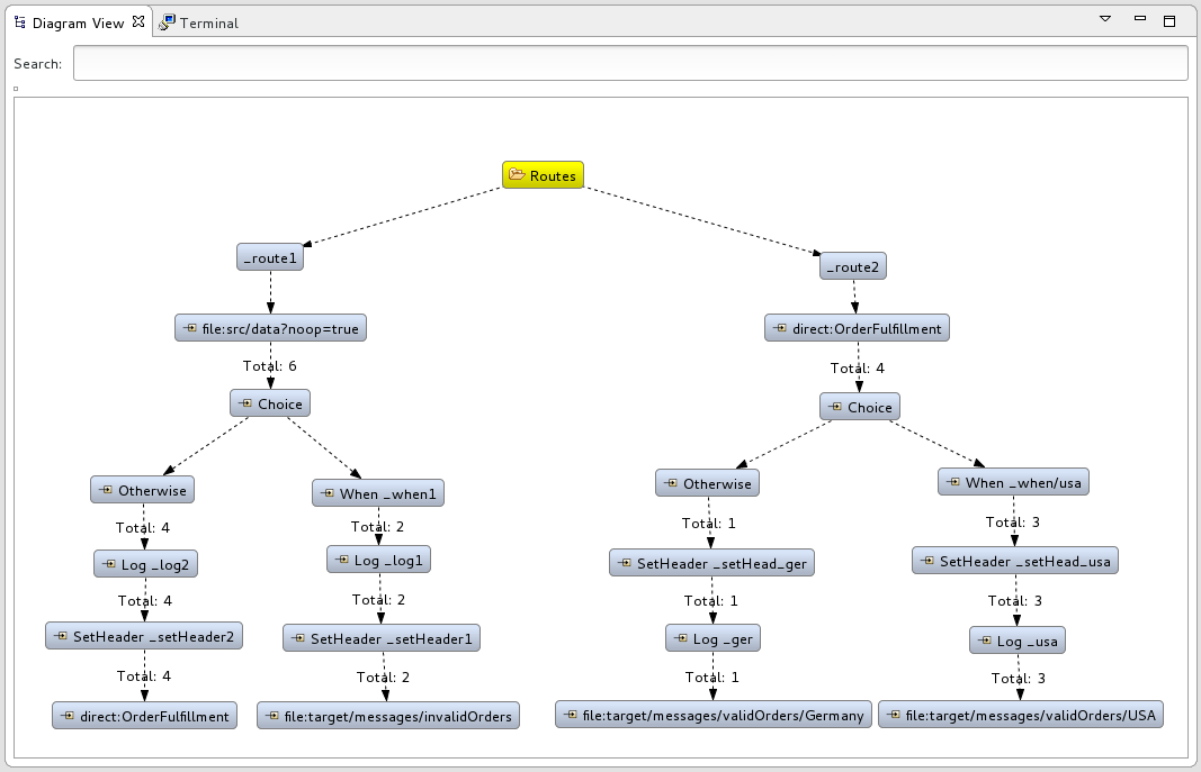
6. **扩展路由的元素：**



7. 右键点击 **Routes** 节点，然后选择 **Start Tracing** ：



该工具在 图表 View 中显示路由上下文的图形表示：

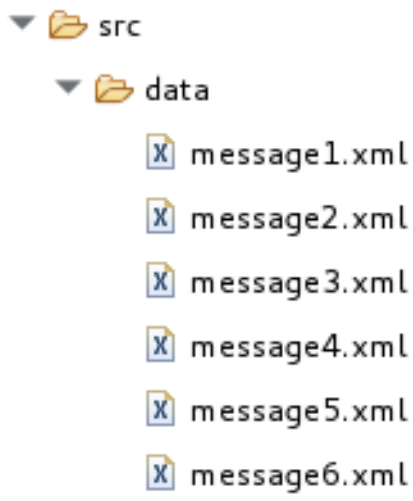


要查看所有消息流路径，您可能需要通过拖动节点以在 Diagram View 选项卡中重新安排节点。您可能还需要调整 Red Hat CodeReady Studio 中其他视图和标签页的大小，以允许 Diagram View 选项卡扩展。

丢弃正在运行的 ZOOORDERAPP 项目的消息

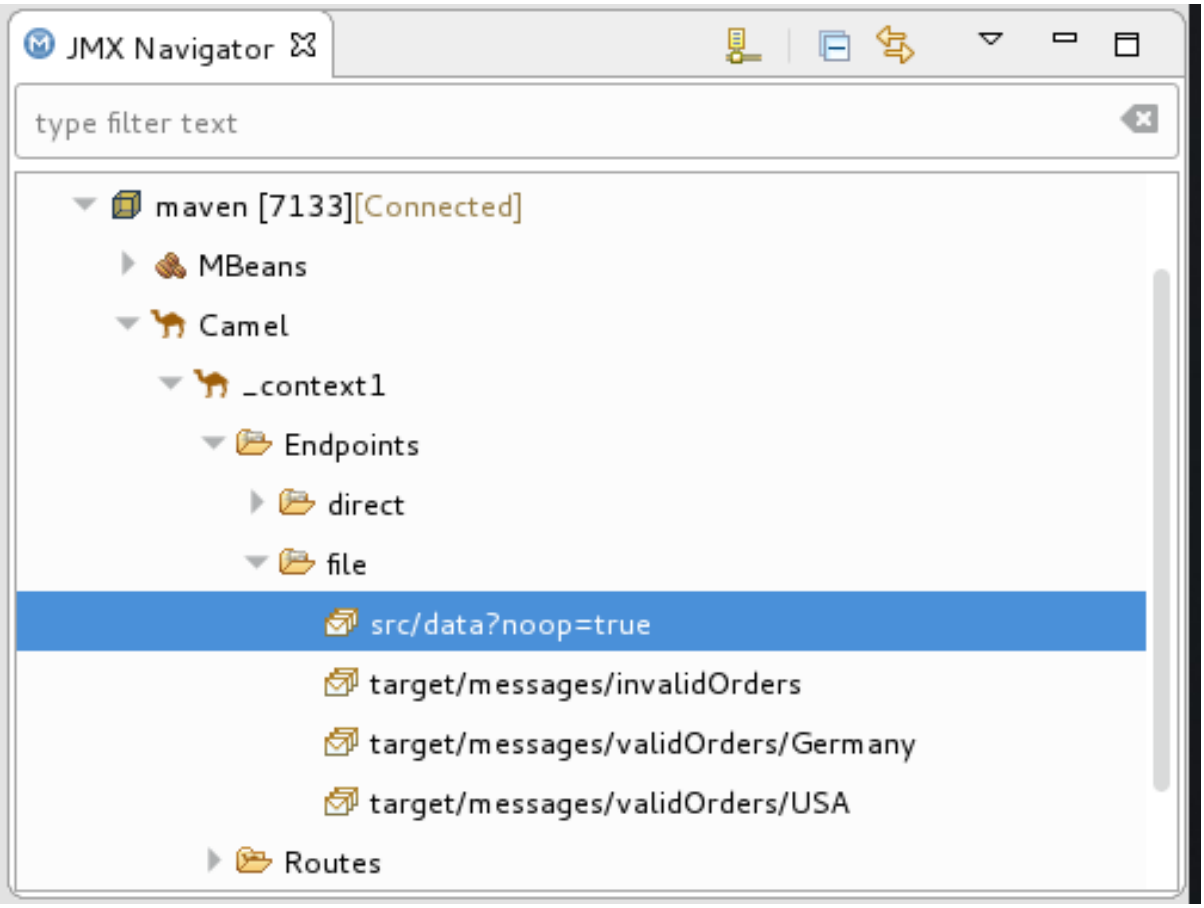
在运行的 ZooOrderApp 项目中丢弃消息：

1. **在 Project Explorer 中，展开 ZooOrderApp/src/data，以便您可以访问消息文件 (message1.xml 到 message6.xml)：**



2.

Drag message1.xml 并将它放到 JMX Navigator 中的 _context1>Endpoints>file>src/data?noop=true 节点上：



当消息遍历路由时，工具会跟踪并记录每个步骤的 passage。

配置消息视图

您必须刷新 Messages View，然后才能显示消息 trace。如果您希望它们在所有消息跟踪中保留，则需要在 Messages View 中配置列。

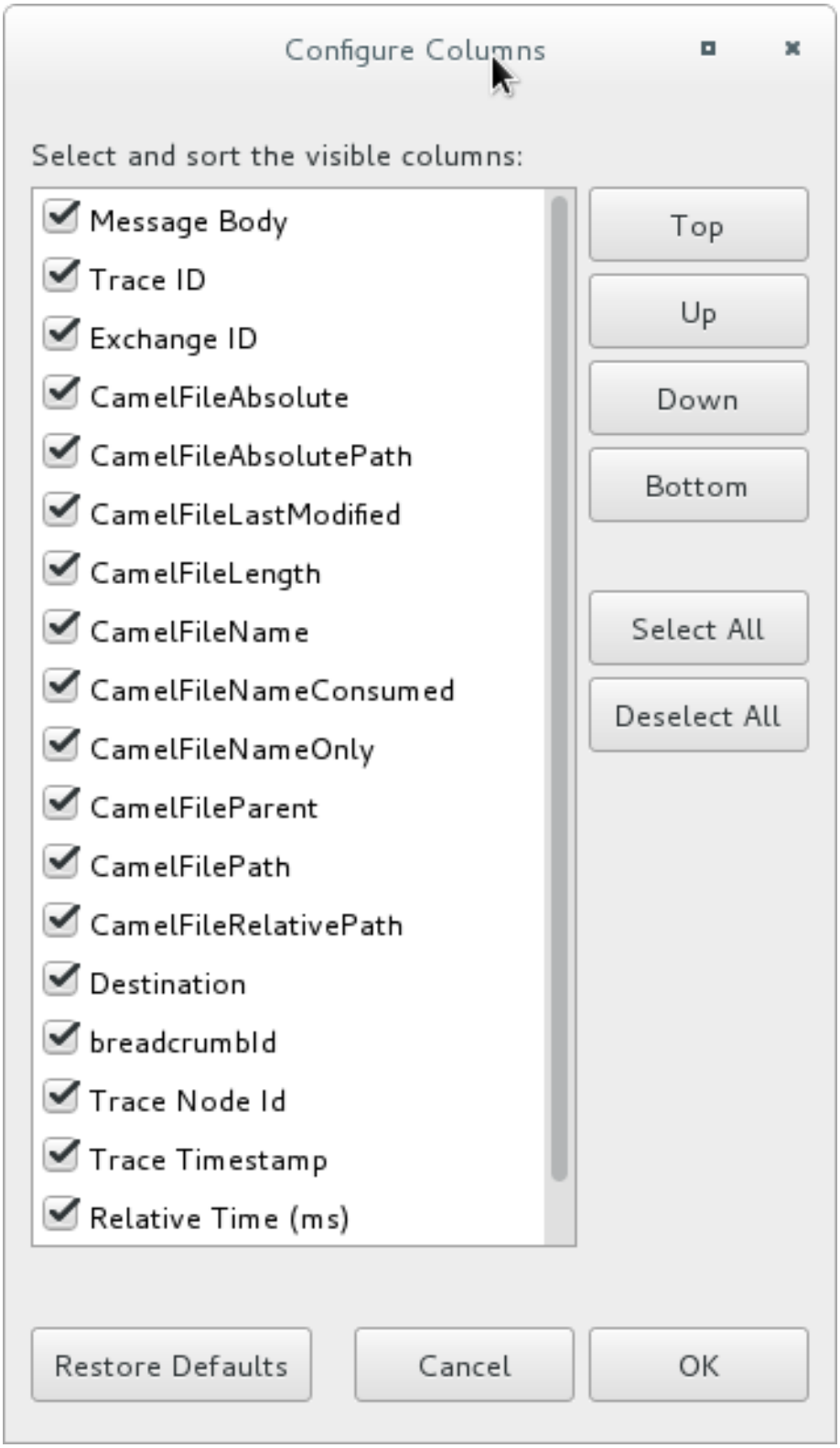
1. 打开 Messages View。

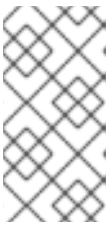
2. 点击面板菜单栏中的

(Refresh 按钮)，为视图填充 message1.xml 的消息 trace。

3. 点面板菜单栏中的

图标，然后选择 Configure Columns 以打开 Configure Columns 向导：





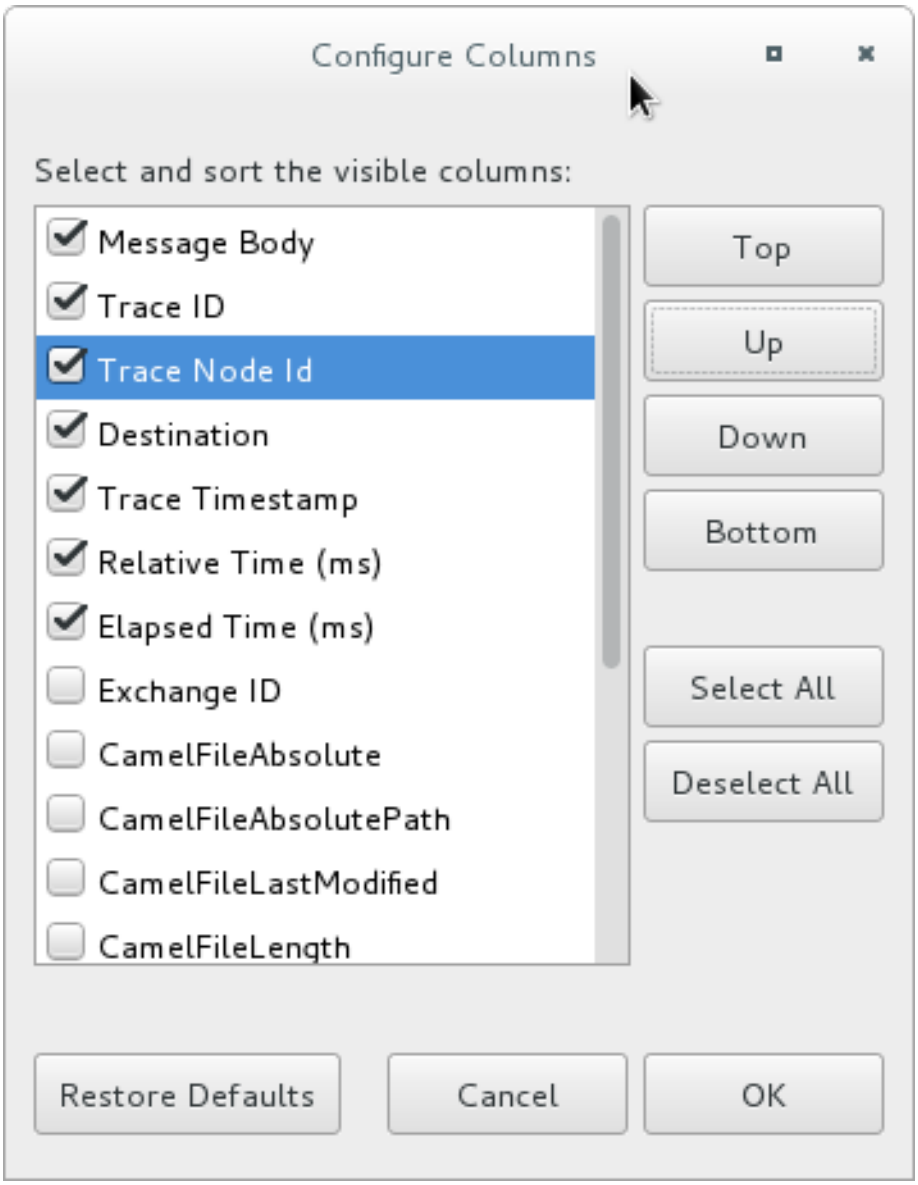
注意

请注意，在路由上下文中为消息设置的消息标头 Destination 会出现在列表中。

您可以选择或取消选择 Messages View，从 Messages View 中包含或排除项目。您可以重新安排在 Messages View 中显示的列顺序，通过突出显示各个项目、选定项目并在列表中移动

它们。

4. 在 Configure Columns 向导中，选择并按顺序排序列：



这些列及其顺序将在 Messages View 中保留，直到您再次更改它们。

注意

您可以在所有工具的表中控制列式布局。使用拖动方法临时重新排列表格格式。例如，拖动列的边框规则以扩展或合同其宽度。要隐藏列，请全额合同其边框。拖动列标题以在表中重新定位列。要保留您的安排，您必须使用 View → Configure Columns 方法。

逐步浏览消息跟踪

逐步浏览消息跟踪：

1. **拖动 message2.xml 并将其丢弃在 JMX Navigator 中的 _context1>Endpoints>file>src/data?noop=true 节点上。**

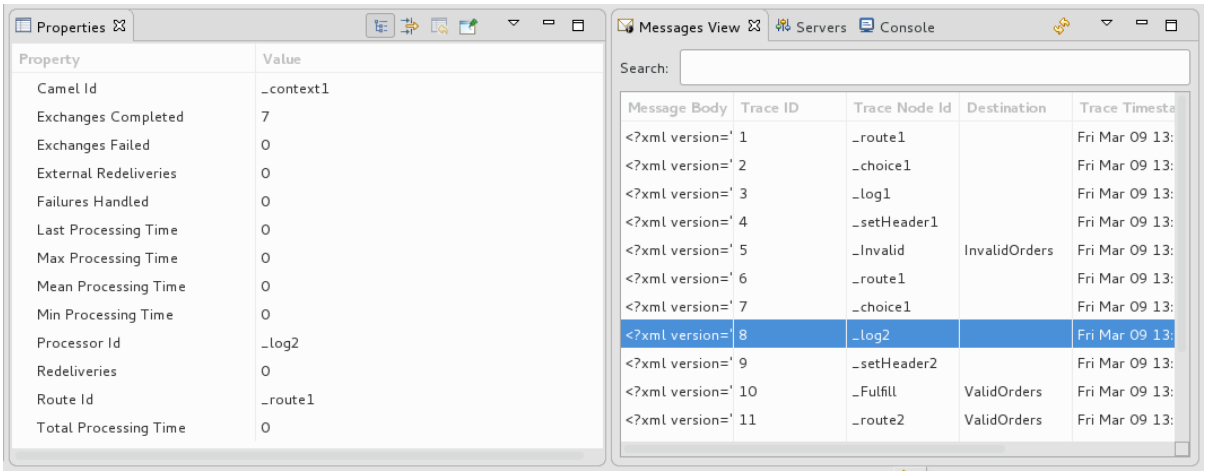
2. **从 Console 切换到 Messages View。**

3. **在 Messages View 中，点**

(Refresh 按钮)使用 message2.xml 消息跟踪填充视图。

每次在 JMX Navigator 中丢弃消息时，您需要刷新 Messages View，以使用消息 trace 填充它。

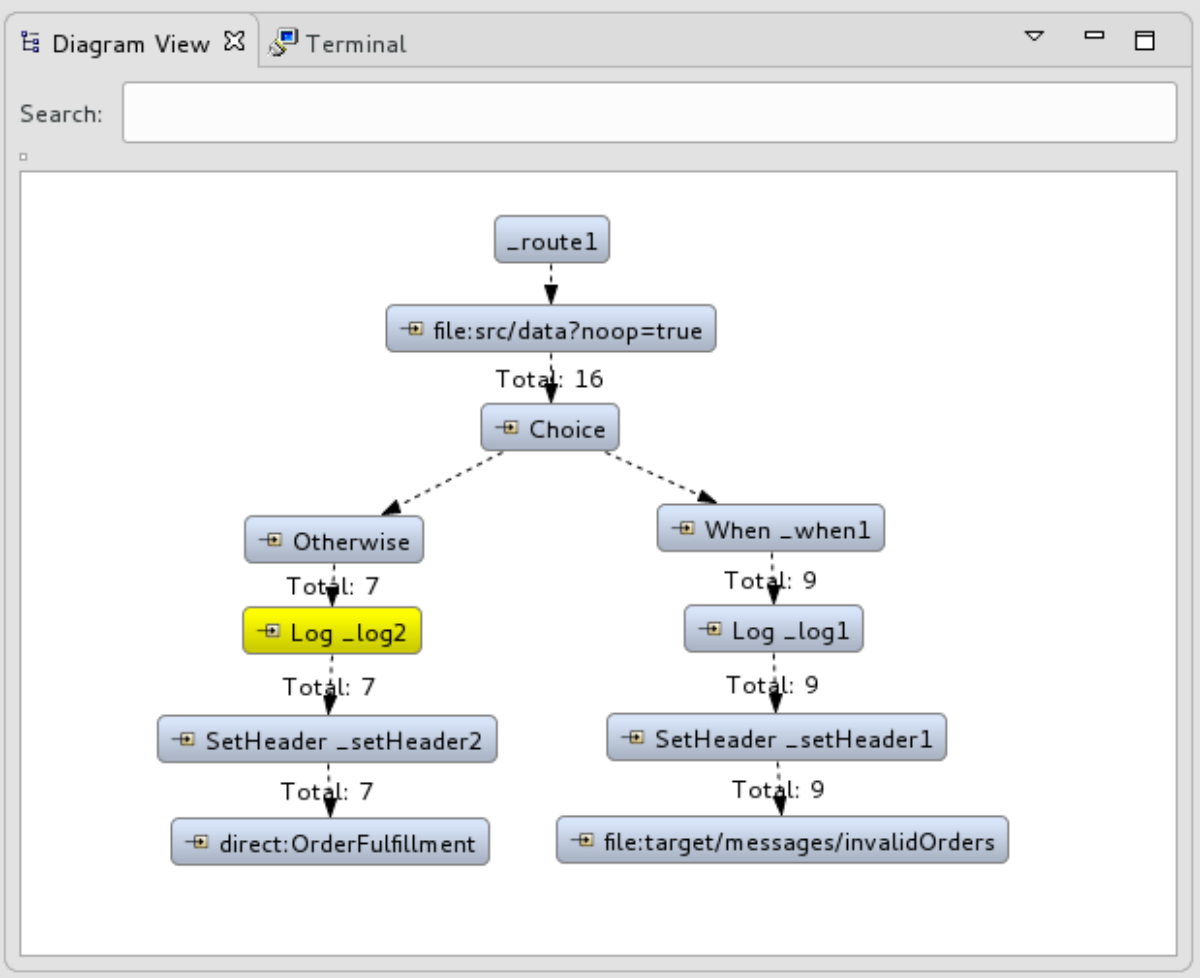
4. **点消息 trace 之一在 Properties 视图中查看它的详情：**



工具在 Properties 视图的前半部分显示消息追踪（包括消息标头）的详细信息，以及 Properties 视图底部的消息实例的内容。因此，如果应用程序在路由中的任何步骤中设置标头，您可以检查 Message Details 以查看它们是否按预期设置。

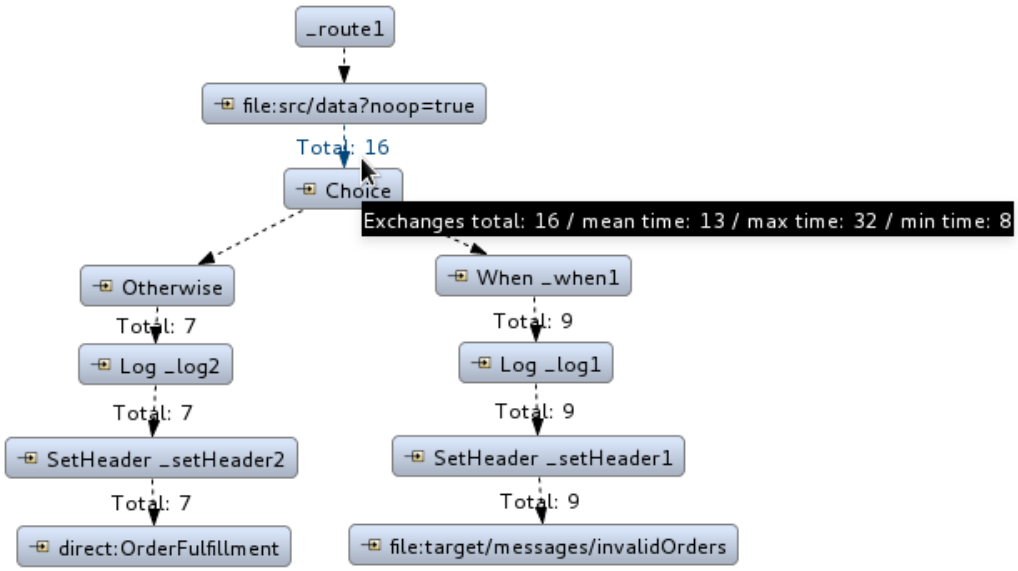
您可以通过突出显示每个消息实例来逐步浏览消息实例，以了解特定消息如何遍历路由，并在路由中的每个步骤中按预期处理。

5. **打开 图表 View，查看 路由中的关联步骤是否已突出显示：**



工具在 Diagram View 中提取路由，标记路径退出具有计时和性能指标（以毫秒为单位）的处理步骤。图中仅显示指标 总交换。

6. 将鼠标指针悬停在显示的指标上，以显示有关消息流的额外指标：



- 平均处理消息所需时间
- 处理消息所花费的最长时间
- 处理消息的最短时间

7. 另外，您还可以将 ZooOrderApp/src/data/ 中剩余的信息拖放到 _context1>Endpoints>file>src/data?noop=true 节点（只要启用了追踪）。

   在每个随后的 drop 中，点
   
   (Refresh 按钮)为 Messages View 填充新消息 trace。

8. 完成后：

   - 在 JMX Navigator 中，右键单击 _context1，然后选择 Stop Tracing Context。
   - 打开控制台 并点击面板右上角的
     
     按钮停止控制台。然后点击
     
     按钮来清除控制台输出。

后续步骤

在 第 9 章 *使用 JUnit 测试路由* 教程中，您要为项目创建一个 JUnit 测试案例，并以 Local Camel Context 运行您的项目。

第 9 章 使用 JUNIT 测试路由

本教程介绍了如何使用 New Camel Test Case 向导来为路由创建测试案例，然后测试路由。

概述

New Camel Test Case 向导会生成一个 boilerplate JUnit 测试案例。当您创建或修改路由（例如，向它添加更多处理器）时，您应该创建或修改生成的测试案例，以添加特定于您创建或更新的路由的预期和断言。这样可确保测试对路由有效。

目标

在本教程中，您将完成以下任务：

- 创建 /src/test/ 文件夹以存储 JUnit 测试案例
- 为 ZooOrderApp 项目生成 JUnit 测试案例
- 修改新生成的 JUnit 测试案例
- 修改 ZooOrderApp 项目的 pom.xml 文件
- 使用新的 JUnit 测试案例运行 ZooOrderApp
- 观察输出

先决条件

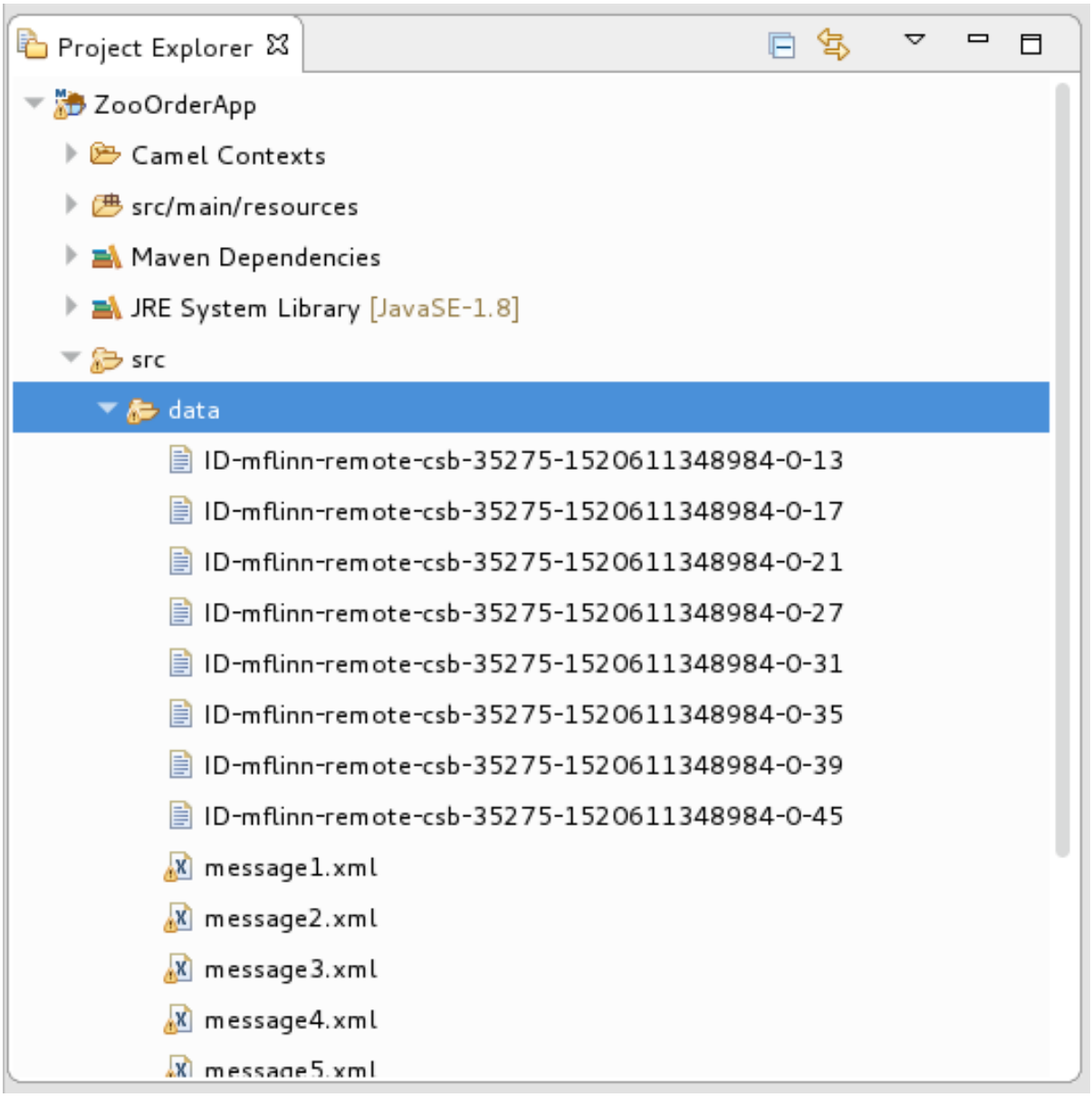
1. 要启动本教程，您需要从以下之一生成的 ZooOrderApp 项目：
   - 完成 第 8 章 *通过路由追踪消息* 教程。

or

- **完成 第 2 章 *设置您的环境* 教程，并将项目的 blueprint.xml 文件替换为提供的 blueprintContexts/blueprint3.xml 文件，如 “关于资源文件”一节 所述。**

2. **从 Project Explorer 中的 ZooOrderApp 项目的 /src/data/ 目录和 /target/messages/ 子目录中删除任何 trace 生成的消息。trace 生成的消息以 ID- 前缀开头。例如：图 9.1 “trace 生成的消息” 显示八个追踪生成的信息：**

图 9.1. trace 生成的消息



选择批处理中的所有 trace-generated 消息，右键单击并选择 Delete。

创建 SRC/TEST 文件夹

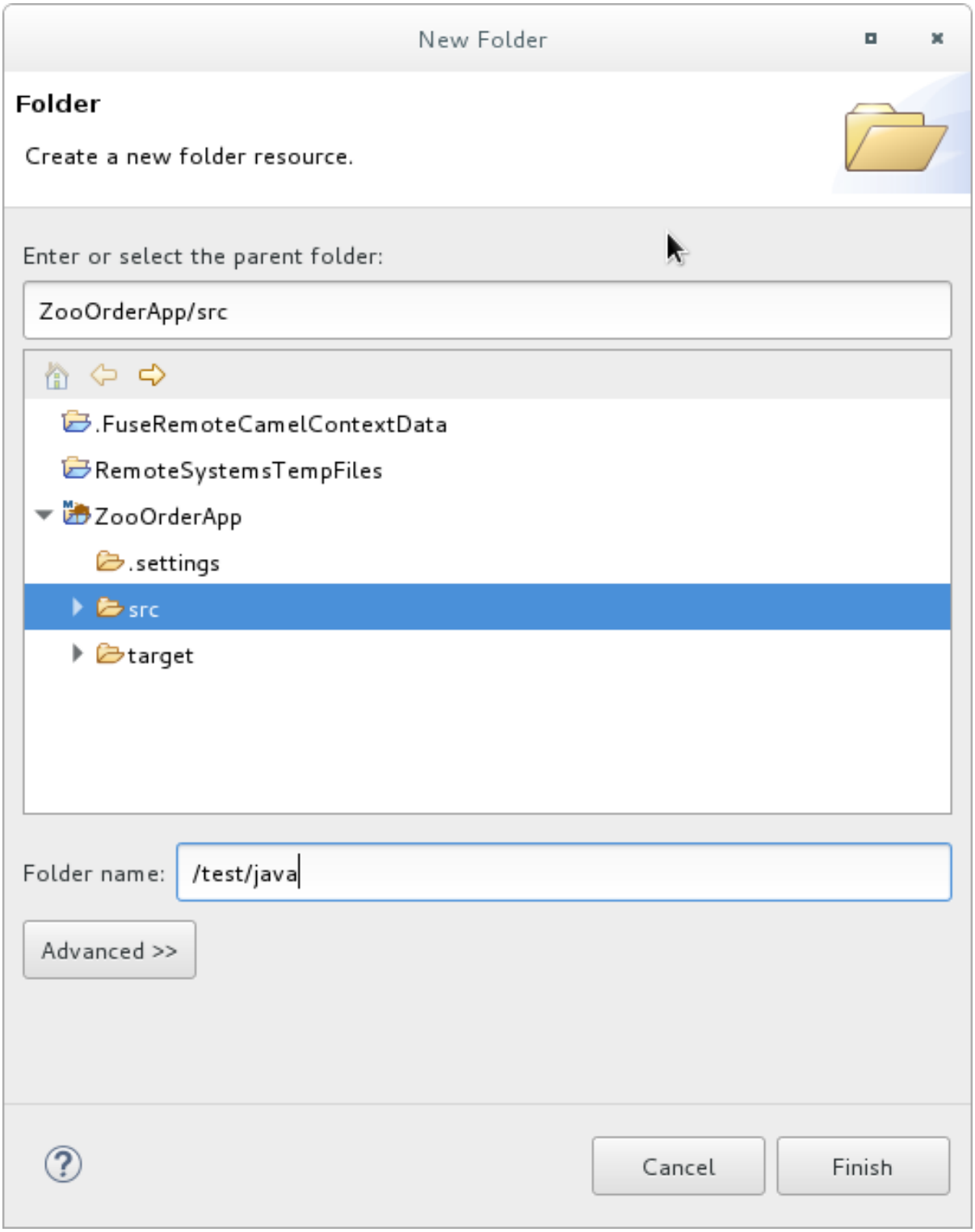
在为 ZooOrderApp 项目创建 JUnit 测试案例前，您必须为构建路径中包含的文件夹创建一个文件夹：

1. 在 Project Explorer 中，右键单击 ZooOrderApp 项目，然后选择 New → Folder。

2. 在 New Folder 对话框中，在项目树窗格中展开 ZooOrderApp 节点，再选择 src 文件夹。

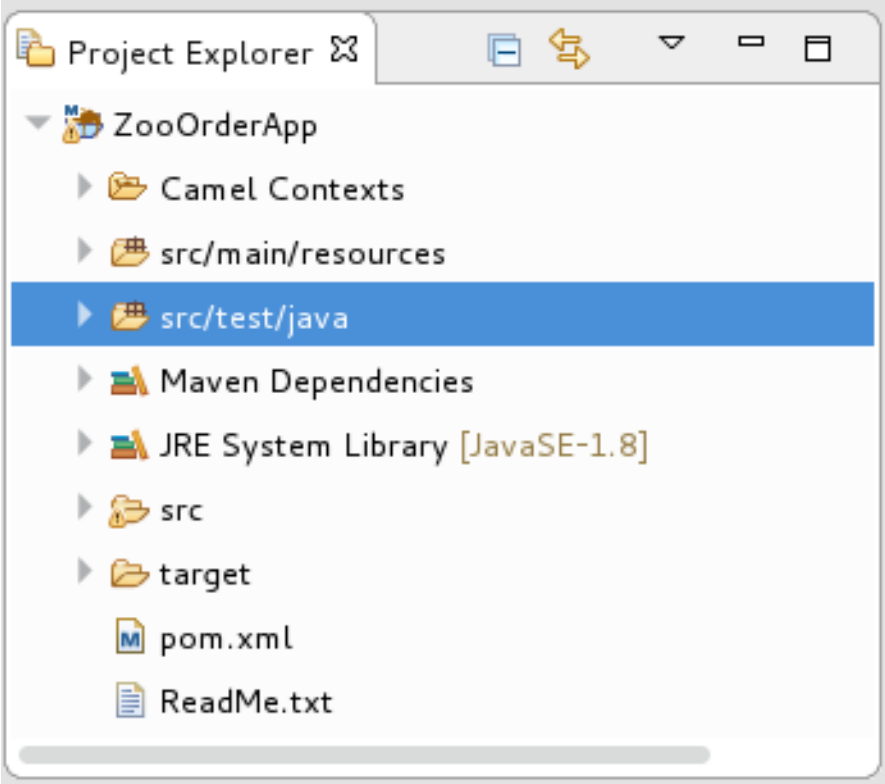
   确保 ZooOrderApp/src 出现在 Enter 或选择父文件夹 字段中。

3. 在 Folder name 中，输入 /test/java ：



4. **点 Finish。**

在 Project Explorer 中，新的 src/test/java 文件夹显示在 src/main/resources 文件夹下：

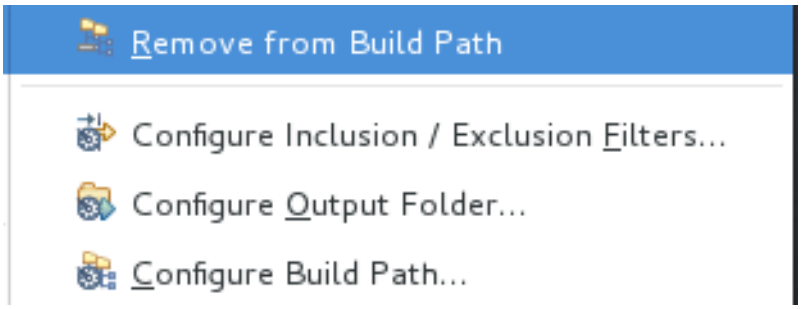


5. **验证新的 /src/test/java 文件夹是否包含在构建路径中。**

   a. **在 Project Explorer 中，右键单击 /src/test/java 文件夹，以打开上下文菜单。**

   b. **选择 Build Path 来查看菜单选项：**

   menu 选项 Remove from Build Path 验证 /src/test/java 文件夹目前是否包含在构建路径中：

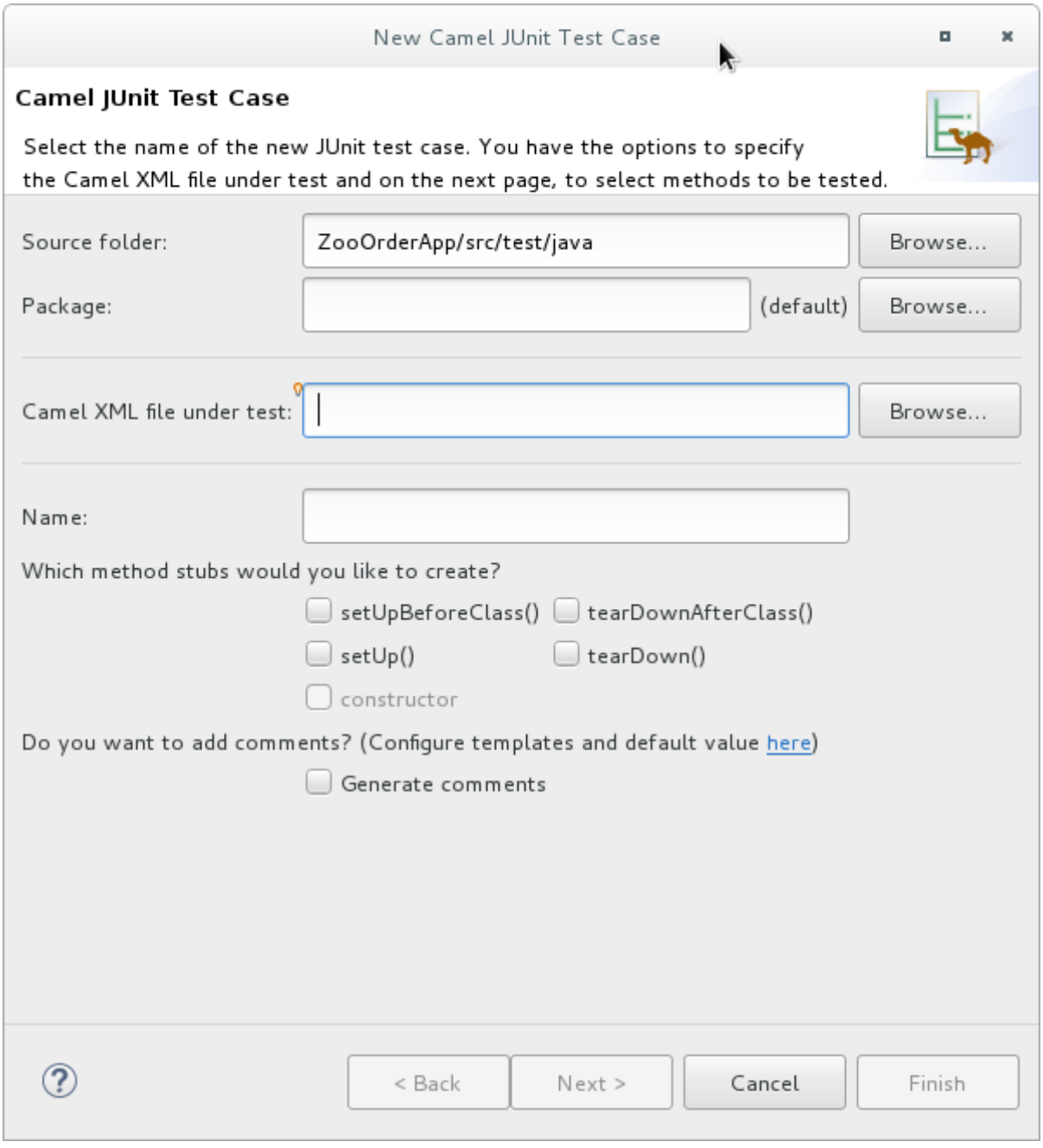


创建 JUNIT 测试案例

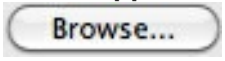
为 ZooOrderApp 项目创建 JUnit 测试案例：

1. **在 Project Explorer 中，选择 src/test/java。**

2. **右键，然后选择 New → Camel Test Case。**



3. **在 Camel JUnit Test Case 向导中，确保 Source folder 字段包含 ZooOrderApp/src/test/java。要找到正确的文件夹，请点击**



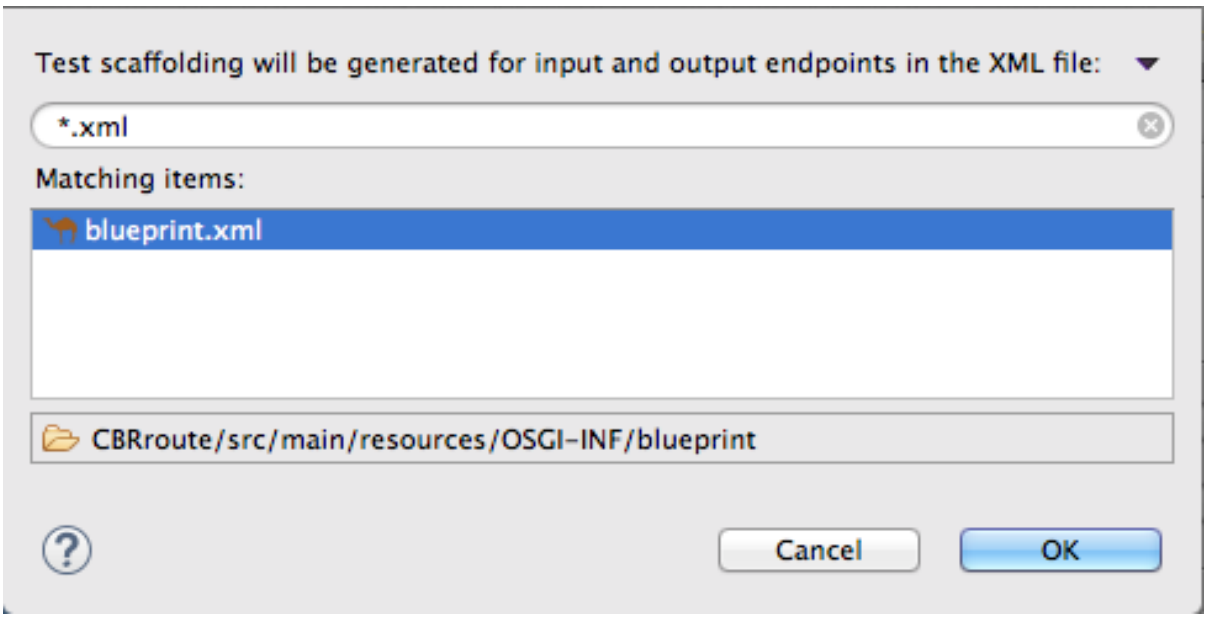
。

4. **在 Package 字段中，输入 tutorial.zooapp.route。此软件包将包括新的测试案例。**

5.

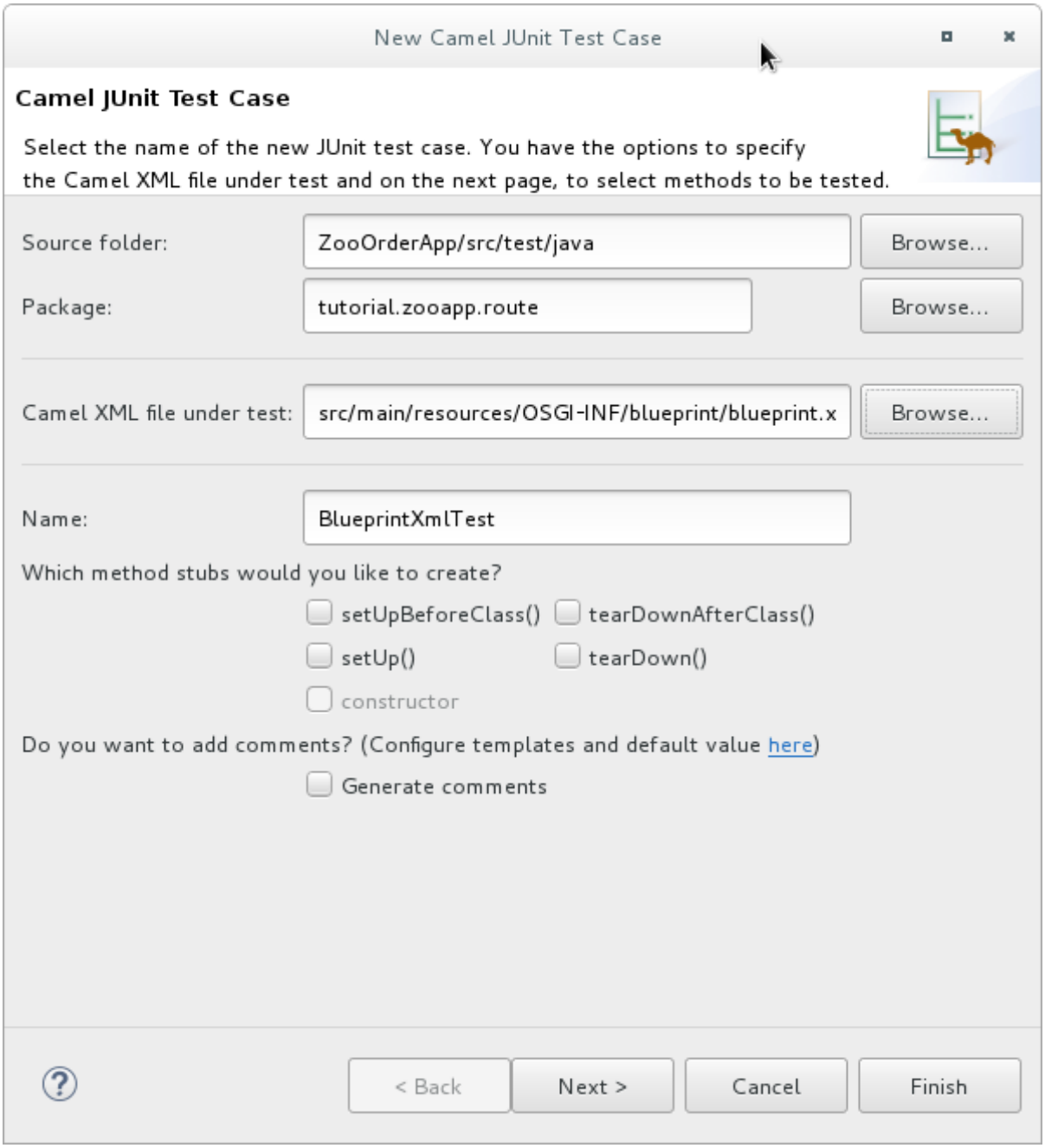
在 test 字段中的 Camel XML 文件中，点

打开配置为过滤 XML 文件的文件管理器，然后选择 ZooOrderApp 项目的 blueprint.xml 文件：



6.

点击 确定。Name 字段默认为 ***BlueprintXmlTest***。



7. **点 Next 以打开 Test Endpoints 页面。**

   默认情况下，会选择所有端点，并将包含在测试案例中。

8. **点 Finish。**

注意

若有提示，请将 JUnit 添加到构建路径中。

测试的工件添加到项目中，并出现在 **src/test/java** 下的 **Project Explorer** 中。实施测试案例的类在工具的 **Java** 编辑器中打开：

```
package tutorial.zooapp.route;

import org.apache.camel.EndpointInject;
import org.apache.camel.Produce;
import org.apache.camel.ProducerTemplate;
import org.apache.camel.builder.RouteBuilder;
import org.apache.camel.component.mock.MockEndpoint;
import org.apache.camel.test.blueprint.CamelBlueprintTestSupport;
import org.junit.Test;

public class BlueprintXmlTest extends CamelBlueprintTestSupport {

 // TODO Create test message bodies that work for the route(s) being tested
 // Expected message bodies
 protected Object[] expectedBodies = { "<something id='1'>expectedBody1</something>",
   "<something id='2'>expectedBody2</something>" };
 // Templates to send to input endpoints
 @Produce(uri = "file:src/data?noop=true")
 protected ProducerTemplate inputEndpoint;
 @Produce(uri = "direct:OrderFulfillment")
 protected ProducerTemplate input2Endpoint;
 // Mock endpoints used to consume messages from the output endpoints and then perform
assertions
 @EndpointInject(uri = "mock:output")
 protected MockEndpoint outputEndpoint;
 @EndpointInject(uri = "mock:output2")
 protected MockEndpoint output2Endpoint;
 @EndpointInject(uri = "mock:output3")
 protected MockEndpoint output3Endpoint;
 @EndpointInject(uri = "mock:output4")
 protected MockEndpoint output4Endpoint;

 @Test
 public void testCamelRoute() throws Exception {
  // Create routes from the output endpoints to our mock endpoints so we can assert expectations
  context.addRoutes(new RouteBuilder() {
   @Override
   public void configure() throws Exception {
    from("file:target/messages/invalidOrders").to(outputEndpoint);
    from("file:target/messages/validOrders/USA").to(output3Endpoint);
    from("file:target/messages/validOrders/Germany").to(output4Endpoint);
   }
  });

  // Define some expectations

  // TODO Ensure expectations make sense for the route(s) we're testing
  outputEndpoint.expectedBodiesReceivedInAnyOrder(expectedBodies);

  // Send some messages to input endpoints
  for (Object expectedBody : expectedBodies) {
```

```
    inputEndpoint.sendBody(expectedBody);
  }

  // Validate our expectations
  assertMockEndpointsSatisfied();
 }

 @Override
 protected String getBlueprintDescriptor() {
  return "OSGI-INF/blueprint/blueprint.xml";
 }

}
```

对于 ZooOrderApp 项目，这个生成的 JUnit 测试案例不足，它将无法成功运行。您需要修改它以及项目的 pom.xml，如 “修改 BlueprintXmlTest 文件”一节 和 “修改 pom.xml 文件”一节 所述。

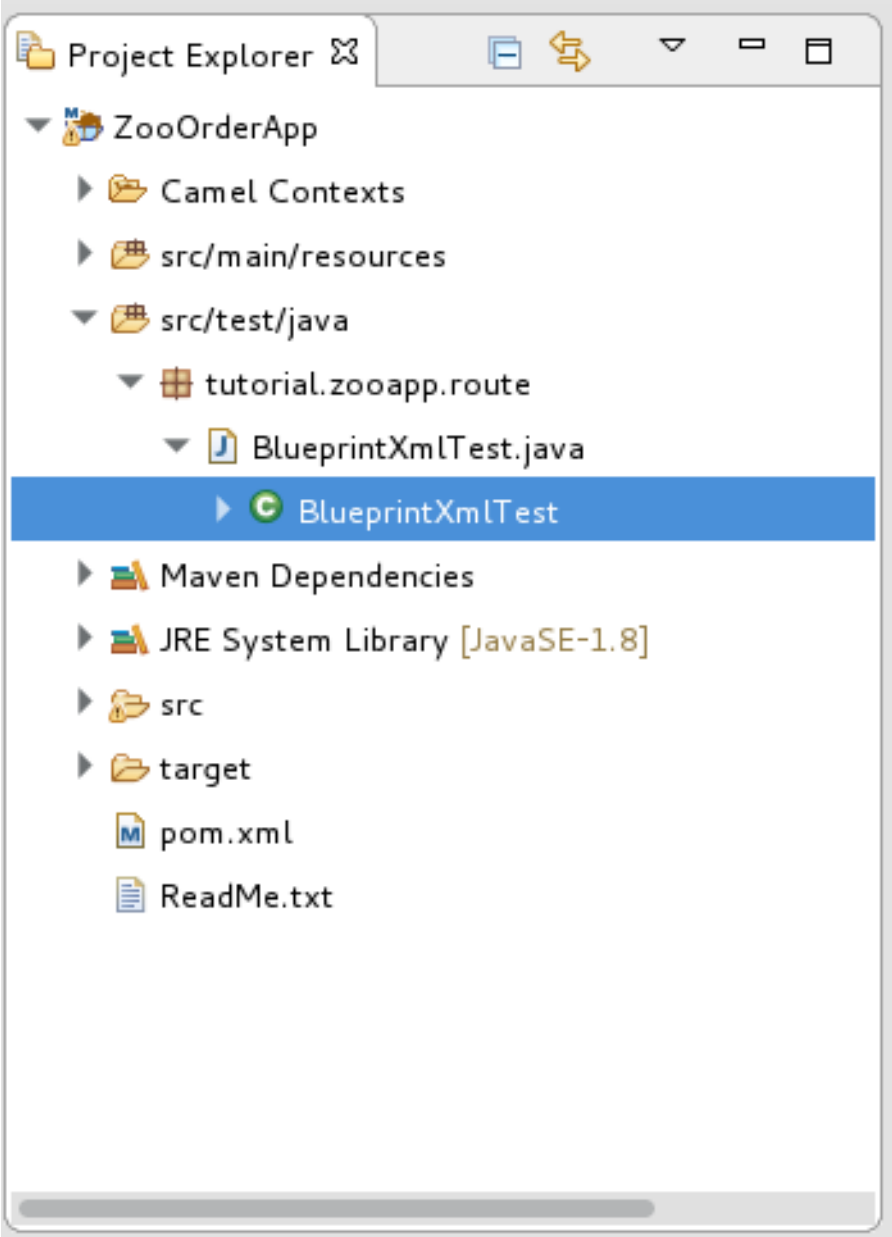
修改 BLUEPRINTXMLTEST 文件

您必须将 BlueprintXmlTest.java 文件修改为：

- 导入多个支持所需文件功能的类
- 创建用于保存各种源 .xml 文件的内容的变量
- 读取源 .xml 文件的内容
- 定义适当的预期

按照以下步骤修改 BlueprintXmlTest.java 文件：

1. 在 Project Explorer 中，展开 ZooOrderApp 项目，以公开 BlueprintXmlTest.java 文件：



2. **打开 BlueprintXmlTest.java 文件。**

3. **在 Java 编辑器中，单击 导入 org.apache.camel.EndpointInject; 旁边的展开按钮，以展开该列表。**

4. **添加以粗体文本显示的两行。根据下一部分的指示，添加第一行会导致当您更新 pom.xml 文件时解析的错误。**

```
package tutorial.zooapp.route;

import org.apache.camel.EndpointInject;
import org.apache.camel.Produce;
import org.apache.camel.ProducerTemplate;
import org.apache.camel.builder.RouteBuilder;
import org.apache.camel.component.mock.MockEndpoint;
```

```
import org.apache.camel.test.blueprint.CamelBlueprintTestSupport;
import org.apache.commons.io.FileUtils;
import org.junit.Test;
import java.io.File;
```

5. **向下滚动到直接在 // 预期消息正文后面的 行。**

6. **将这些行替换有 保护的 Object[] expectedBodies={ …… expectedBody2</something>"}; to these protected-busybox with those protected String body"; 行：**

```
protected String body1; protected String body2; protected String body3; protected String
body4; protected String body5; protected String body6;
```

7. **向下滚动到行 public void testCamelRoute （）会抛出 Exception {，并在它后面直接插入行 body Serial = FileUtils.readFileToString (new File ("src/data/message failing.xml"), "UTF-8"); 如下所示。这些行将指示错误，直到您按照下一节中的指示更新 pom.xml 文件。**

```
// Valid orders body2 = FileUtils.readFileToString(new File("src/data/message2.xml"), "UTF-
8"); body4 = FileUtils.readFileToString(new File("src/data/message4.xml"), "UTF-8"); body5 =
FileUtils.readFileToString(new File("src/data/message5.xml"), "UTF-8"); body6 =
FileUtils.readFileToString(new File("src/data/message6.xml"), "UTF-8"); // Invalid orders
body1 = FileUtils.readFileToString(new File("src/data/message1.xml"), "UTF-8"); body3 =
FileUtils.readFileToString(new File("src/data/message3.xml"), "UTF-8");
```

8. **向下滚动到在 // TODO Ensure 预期之后直接遵循的行，对我们正在测试的路由有意义。**

9. **将以 outputEndpoint.expectedBodiesReceivedInAnyOrder (expectedBodies) 开头的代码块替换为 …inputEndpoint.sendBody (expectedBody); } 替换为此处显示的行：**

```
// Invalid orders outputEndpoint.expectedBodiesReceived(body1, body3); // Valid orders for
USA output3Endpoint.expectedBodiesReceived(body2, body5, body6); // Valid order for
Germany output4Endpoint.expectedBodiesReceived(body4);
```

将剩余的代码保留原样。

10. **保存该文件。**

11. **检查您更新的 BlueprintXmlTest.java 文件是否具有所需的修改。它应该类似如下：**

```
package tutorial.zooapp.route;

import org.apache.camel.EndpointInject;
import org.apache.camel.Produce;
import org.apache.camel.ProducerTemplate;
import org.apache.camel.builder.RouteBuilder;
import org.apache.camel.component.mock.MockEndpoint;
import org.apache.camel.test.blueprint.CamelBlueprintTestSupport;
import org.apache.commons.io.FileUtils;
import org.junit.Test;
import java.io.file;

public class BlueprintXmlTest extends CamelBlueprintTestSupport {

 // TODO Create test message bodies that work for the route(s) being tested
 // Expected message bodies
 protected String body1;
 protected String body2;
 protected String body3;
 protected String body4;
 protected String body5;
 protected String body6;
 // Templates to send to input endpoints
 @Produce(uri = "file:src/data?noop=true")
 protected ProducerTemplate inputEndpoint;
 @Produce(uri = "direct:OrderFulfillment")
 protected ProducerTemplate input2Endpoint;
 // Mock endpoints used to consume messages from the output endpoints and then perform
assertions
 @EndpointInject(uri = "mock:output")
 protected MockEndpoint outputEndpoint;
 @EndpointInject(uri = "mock:output2")
 protected MockEndpoint output2Endpoint;
 @EndpointInject(uri = "mock:output3")
 protected MockEndpoint output3Endpoint;
 @EndpointInject(uri = "mock:output4")
 protected MockEndpoint output4Endpoint;

 @Test
 public void testCamelRoute() throws Exception {
  // Create routes from the output endpoints to our mock endpoints so we can assert
expectations
  context.addRoutes(new RouteBuilder() {
   @Override
   public void configure() throws Exception {
    // Valid orders
    body2 = FileUtils.readFileToString(new File("src/data/message2.xml"), "UTF-8");
    body4 = FileUtils.readFileToString(new File("src/data/message4.xml"), "UTF-8");
    body5 = FileUtils.readFileToString(new File("src/data/message5.xml"), "UTF-8");
    body6 = FileUtils.readFileToString(new File("src/data/message6.xml"), "UTF-8");

    // Invalid orders
    body1 = FileUtils.readFileToString(new File("src/data/message1.xml"), "UTF-8");
    body3 = FileUtils.readFileToString(new File("src/data/message3.xml"), "UTF-8");

    from("file:target/messages/invalidOrders").to(outputEndpoint);
```

```
    from("file:target/messages/validOrders/USA").to(output3Endpoint);
    from("file:target/messages/validOrders/Germany").to(output4Endpoint);
    from("direct:OrderFulfillment").to(output2Endpoint);
   }
  });

  // Define some expectations

  // TODO Ensure expectations make sense for the route(s) we're testing
  // Invalid orders
  outputEndpoint.expectedBodiesReceived(body1, body3);

  // Valid orders for USA
  output3Endpoint.expectedBodiesReceived(body2, body5, body6);

  // Valid order for Germany
  output4Endpoint.expectedBodiesReceived(body4);

  // Validate our expectations
  assertMockEndpointsSatisfied();
 }

 @Override
 protected String getBlueprintDescriptor() {
  return "OSGI-INF/blueprint/blueprint.xml";
 }

}
```

修改 POM.XML 文件

您需要将对 commons-io 项目的依赖添加到 ZooOrderApp 项目的 pom.xml 文件中：

1. **在 Project Explorer 中，选择位于目标文件夹下的 pom.xml，然后在工具的 XML 编辑器中打开它。**

2. **单击页面底部的 pom.xml 选项卡，以打开文件进行编辑。**

3. **将这些行添加到 < dependencies> 部分的末尾：**

```
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.5</version>
    <scope>test</scope>
</dependency>
```

4. 保存该文件。

运行 JUNIT 测试

运行测试：

1. 切换到 JBoss 透视图以释放更多工作区。

2. 在 Project Explorer 中，右键单击 ZooOrderApp 项目。

3. 选择 Run As → JUnit Test。

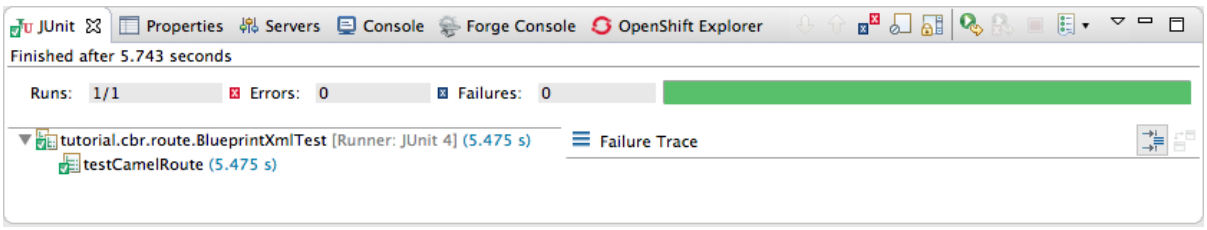
默认情况下，JUnit 视图在侧边栏中打开。（为了提供更好的视图，将其拖到底部，右面板显示 控制台、服务器以及 属性 选项卡。）

注意

有时，当 JUnit 首次在项目上运行时，测试会失败。再次运行测试会得到成功的结果。

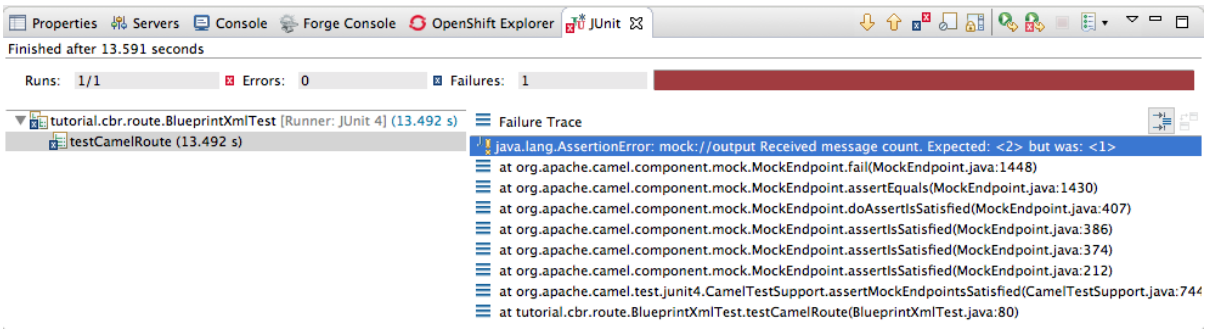
如果测试成功运行，您会看到类似如下的内容：

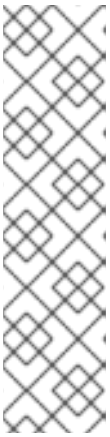
图 9.2. 成功 JUnit 运行



当测试失败时，您会看到类似如下的内容：

图 9.3. JUnit 运行失败





注意

如果您的执行环境没有设置为 Java SE 8，则 JUnit 将失败。JUnit 选项卡顶部的消息栏将显示一条错误消息，指出它无法找到正确的 SDK。

要解决这个问题，打开项目的上下文菜单，然后选择 Run As → Run Configuration → JRE。点 *Execution environment 字段旁边的 Environments] 按钮 找到并选择 Java SE 8 环境。

4. **检查输出并采取措施来解决任何测试失败。**

   要查看 JUnit 面板中显示的更多错误，请单击面板菜单栏中的
   
   以最大化视图。

在再次运行 JUnit 测试案例前，从 Project Explorer 中的 ZooOrderApp 项目的 /src/data 文件夹中删除所有 JUnit 生成的测试消息（请参阅 图 9.1 “trace 生成的消息”）。

进一步阅读

要了解有关 JUnit 测试的更多信息，请参阅 JUnit。

后续步骤

在 第 10 章 *将项目发布到红帽 Fuse* 教程中，您将了解如何将 Apache Camel 项目发布到红帽 Fuse 中。

第 10 章 将项目发布到红帽 FUSE

本教程介绍了将项目发布到红帽 Fuse 的过程。它假设您已在运行 Red Hat Fuse 工具的同一机器上安装了一个 Red Hat Fuse 实例。

目标

在本教程中，您将完成以下任务：

- 定义 Red Hat Fuse 服务器
- 配置发布选项
- 启动 Red Hat Fuse 服务器并发布 ZooOrderApp 项目
- 连接到 Red Hat Fuse 服务器
- 验证 ZooOrderApp 项目的捆绑包是否已成功构建并发布
- 卸载 ZooOrderApp 项目

先决条件

在开始本教程前：

- 访问 Red Hat Fuse 实例
- 在您的计算机上安装 Java 8
- ZooOrderApp 项目由以下之一生成：

- 完成 第 9 章 *使用 JUnit 测试路由* 教程。

  or

- 完成 第 2 章 *设置您的环境* 教程，并将项目的 blueprint.xml 文件替换为提供的 blueprintContexts/blueprint3.xml 文件，如 “关于资源文件”一节 所述。

定义 RED HAT FUSE SERVER

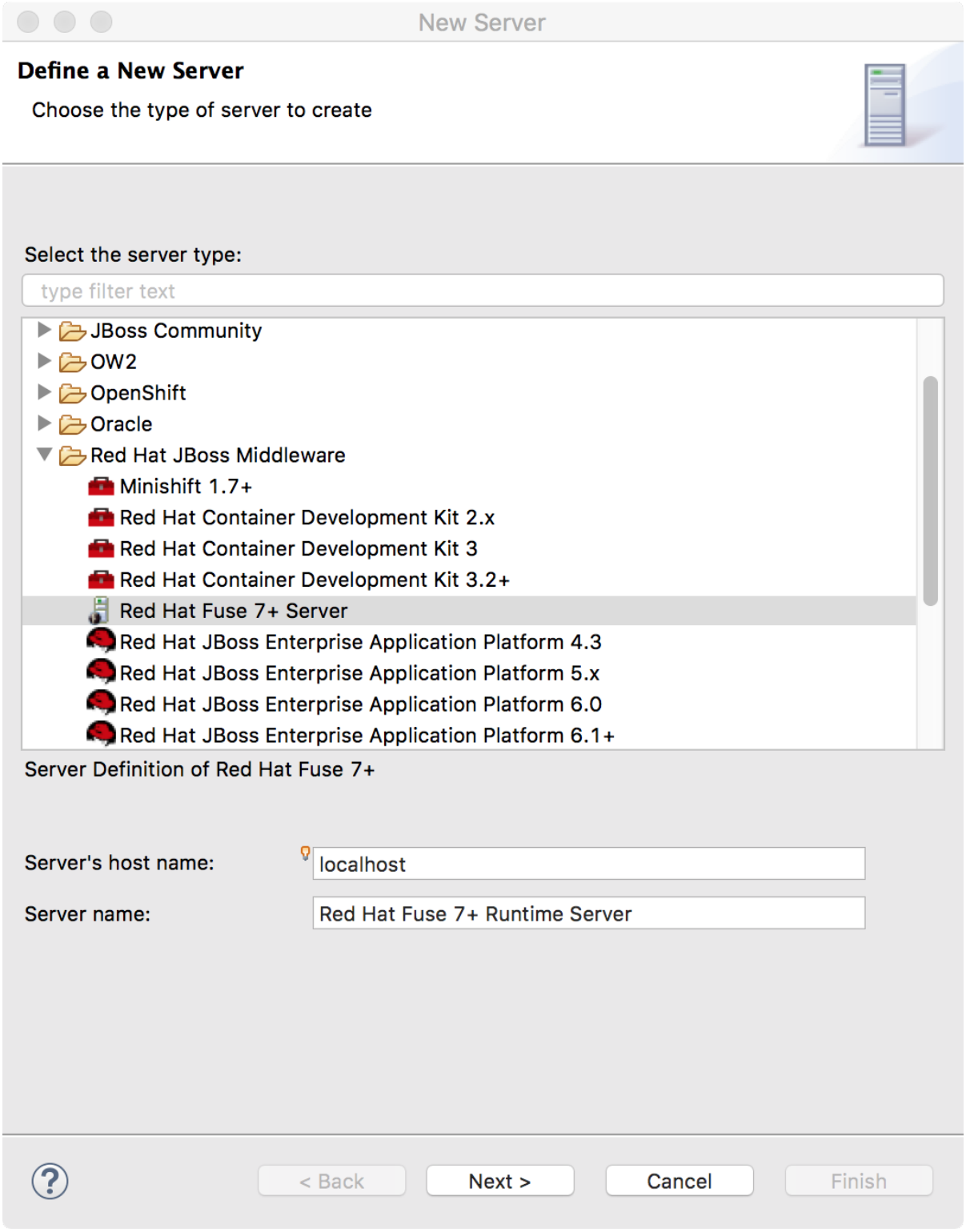
定义服务器：

1. 打开 Fuse Integration 透视图。
2. 单击右下角的 Servers 选项卡，以打开 Servers 视图。
3. 点 No servers 可用。点击此链接来创建新 server... 链接，以打开 Define a New Server 页面。

注意

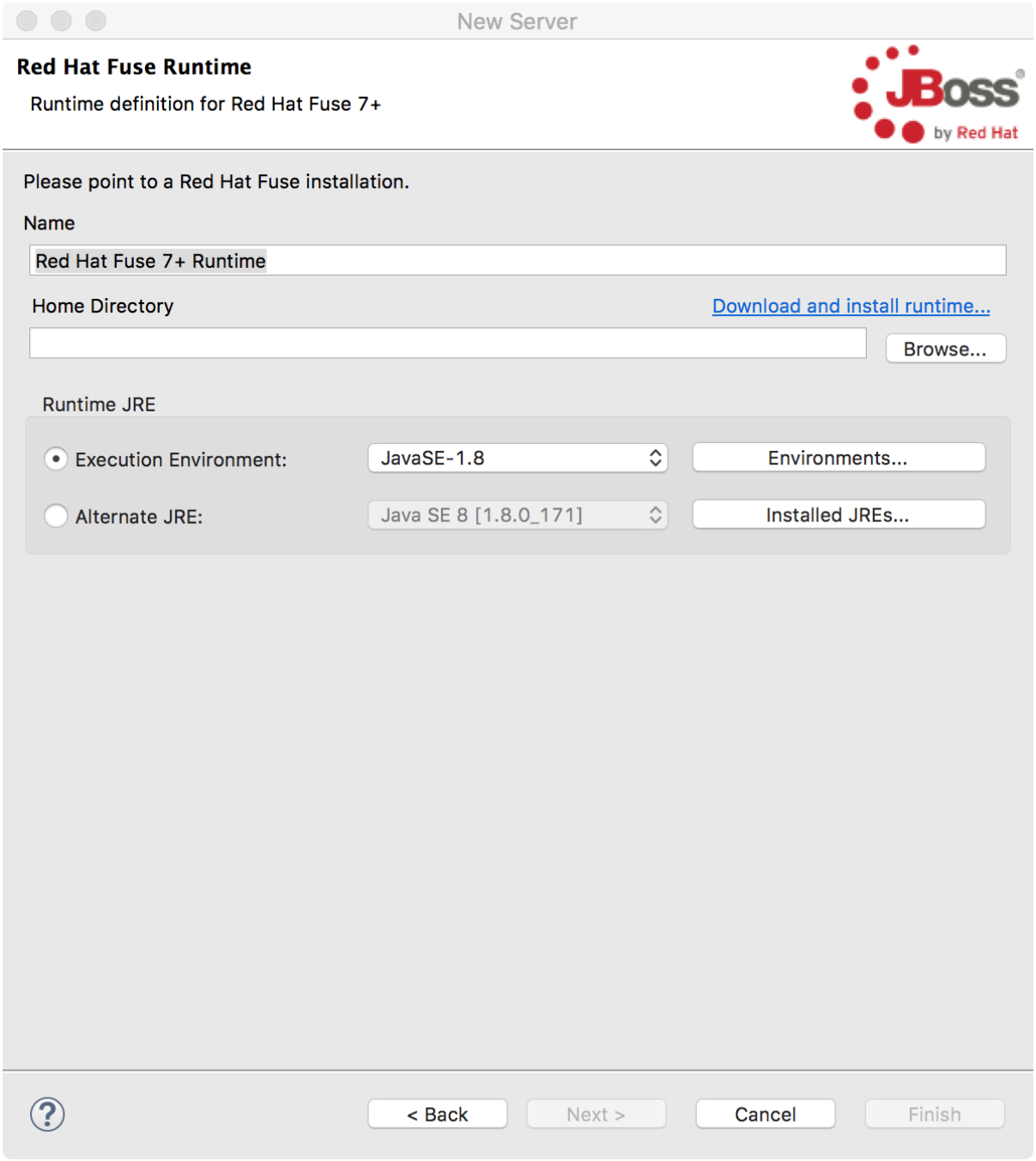
要在已经定义时定义新服务器，请在 Servers 视图中右键单击，然后选择 New → Server。

4. 扩展 Red Hat JBoss Middleware 节点以公开可用的服务器选项：



5. 选择 **Red Hat Fuse** 服务器。

6. **接受 服务器主机名(*localhost*)和 服务器名称 (Fuse *n.n* Runtime Server)的默认值，然后单击 Next 以打开 Runtime 页面：**



注意

如果您尚未安装 Fuse，现在可以使用 Download and install runtime 链接下载它。

如果您已经定义了服务器，工具会跳过此页面，并显示配置详情页面。

7. **接受 名称 的默认值。**

8. 单击 **Home Directory** 字段旁边的 **Browse**，以导航到安装并选择它。

9. 从 **Execution Environment** 旁边的下拉菜单中选择运行时 **JRE**。

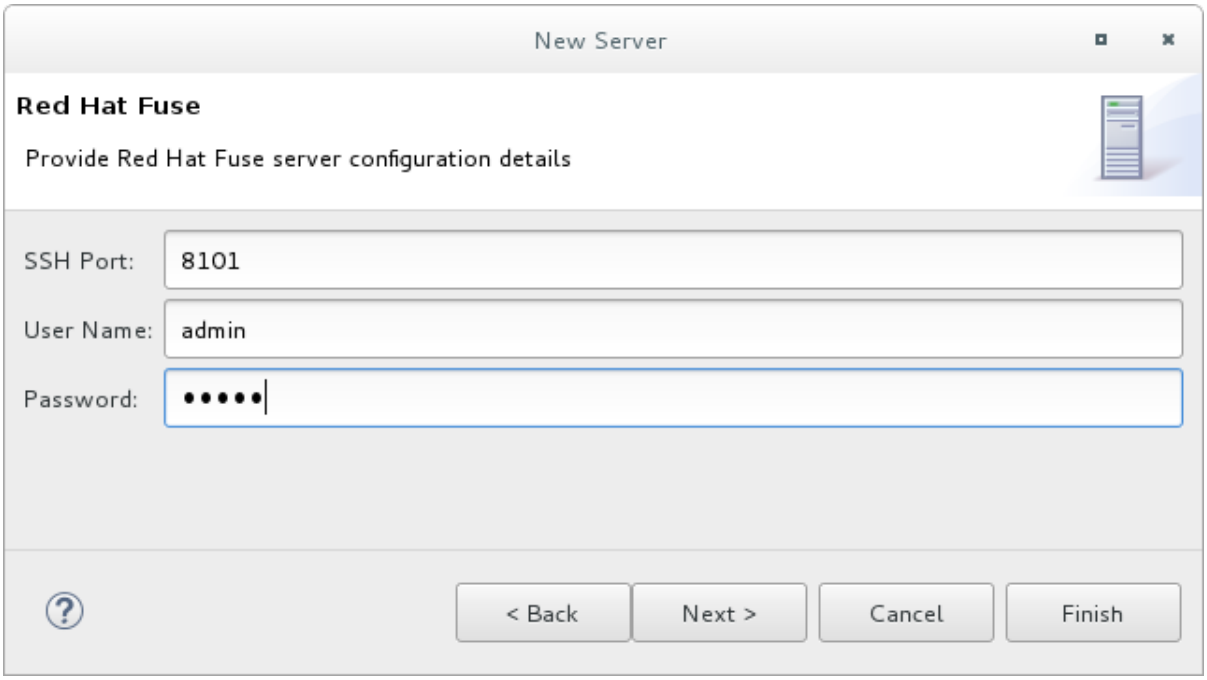
选择 **JavaSE-1.8** （推荐）。如有必要，单击 **Environments** 按钮，将其从列表中选中。

注意

Fuse 服务器需要 **Java 8** （推荐）。要为 **Execution Environment** 选择它，您必须已安装了它。

10. **Alternate JRE** 选项保留原样。

11. 点 **Next** 以保存 **Fuse** 服务器的运行时定义，并打开 **Fuse** 服务器配置详情页面 ：

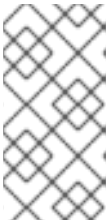


12. 接受 **SSH** 端口 的默认端口(*8101*)。

运行时使用 **SSH** 端口来连接服务器的 **Karaf shell**。如果此默认不正确，您可以通过在 **Red Hat Fuse** *installDir*/etc/org.apache.karaf.shell.cfg 文件中查找正确的端口号。

13. 在 用户名 中，输入用于登录服务器的名称。

这是存储在 Red Hat Fuse *installDir*'/etc/users.properties' 文件中的用户名。



注意

如果在 /etc/users.properties 文件中激活（未添加） 默认用户，工具将使用默认用户名和密码自动填充用户名和密码。

如果没有设置，您可以使用格式 user=password,role （如 joe=secret,Administrator）添加到该文件中，也可以使用 karaf jaas 命令集设置：

- jaas:realms swig-wagonto 列出域
- jaas:manage --index 1 swig- swigto 编辑第一个(server)域
- jaas:useradd <username> <password > HEKETI- swigto 添加用户和关联的密码
- jaas:roleadd <username> Administrator wagon-rhacmto 指定新用户的角色
- jaas:update HEKETI- swigto 使用新的用户信息更新域

如果已经为服务器选择了 jaas 域，您可以通过发出 JBossFuse:karaf@root>jaas:users 来发现用户名。

14. 在 Password 中，键入 用户名 所需的密码以登录到服务器。

这是在红帽 Fuse 的 *installDir*/etc/users.properties 文件中或 karaf jaas 命令中设定的密码。

15. 点 Finish。

Runtime Server [stopped, Synchronized] 会出现在 Servers 视图中。

16. 在 Servers 视图中，展开 Runtime Server：

JMX[Disconnected] 在 Runtime Server [stopped, Synchronized] 条目下显示为节点。

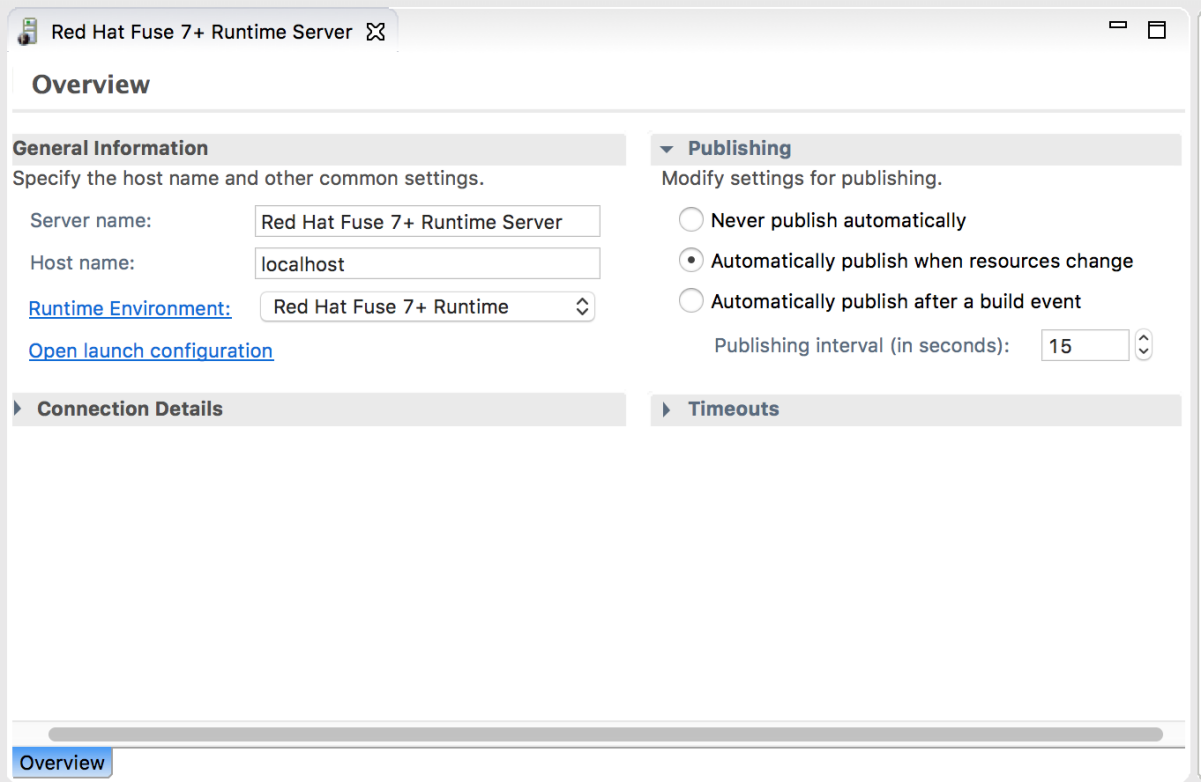
配置发布选项

使用发布选项，您可以配置 ZooOrderApp 项目如何以及何时发布到正在运行的服务器：

- 在保存对项目所做的更改后自动自动
- 更改并保存项目后自动配置的间隔
- 选择发布操作时手动

在本教程中，您可以在保存对 ZooOrderApp 项目的更改时配置立即发布。要做到这一点：

1. 在 Servers 视图中，双击 Runtime Server [stopped, Synchronized] 条目，以显示其概览。
2. 在服务器的 Overview 页面中，展开 Publishing 部分以公开选项。



确保启用 资源更改时，自动发布 选项。

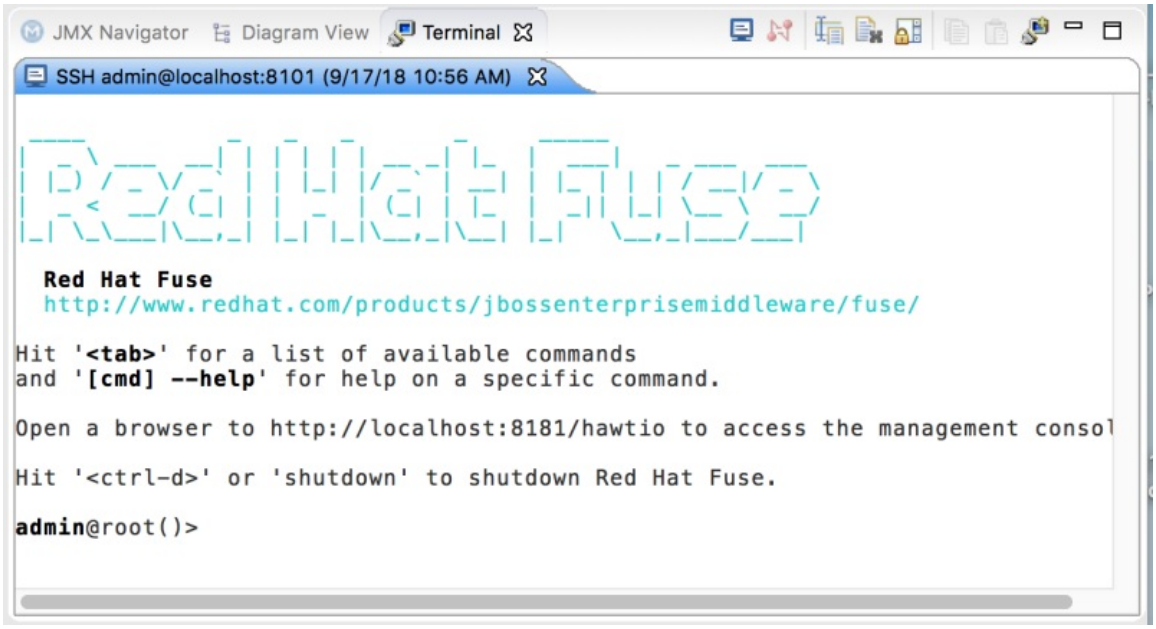
（可选）更改 发布间隔 值，以便在进行更改时加快或延迟发布项目。

3. **在 Servers 视图中，点**

。

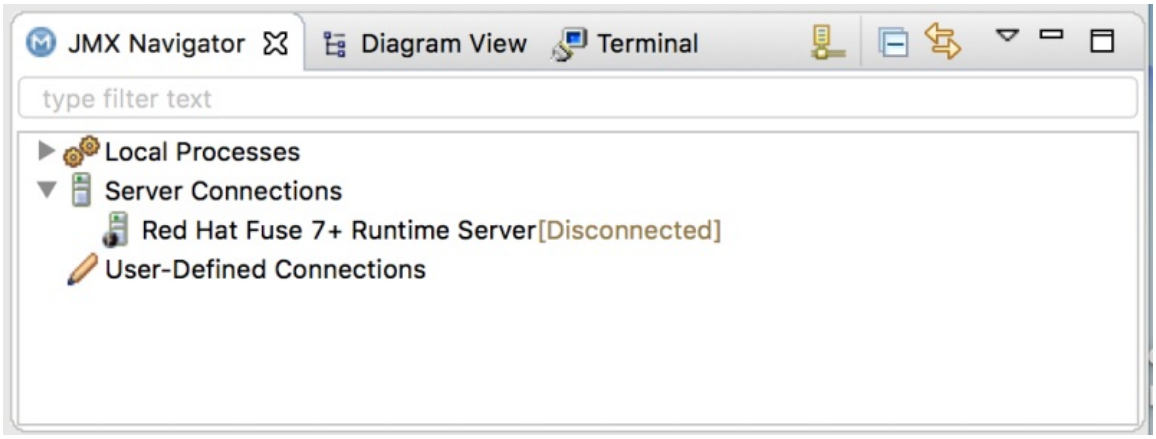
4. **等待几秒钟，使服务器启动。当它这样做时：**

- **Terminal 视图显示 splash 屏幕：**

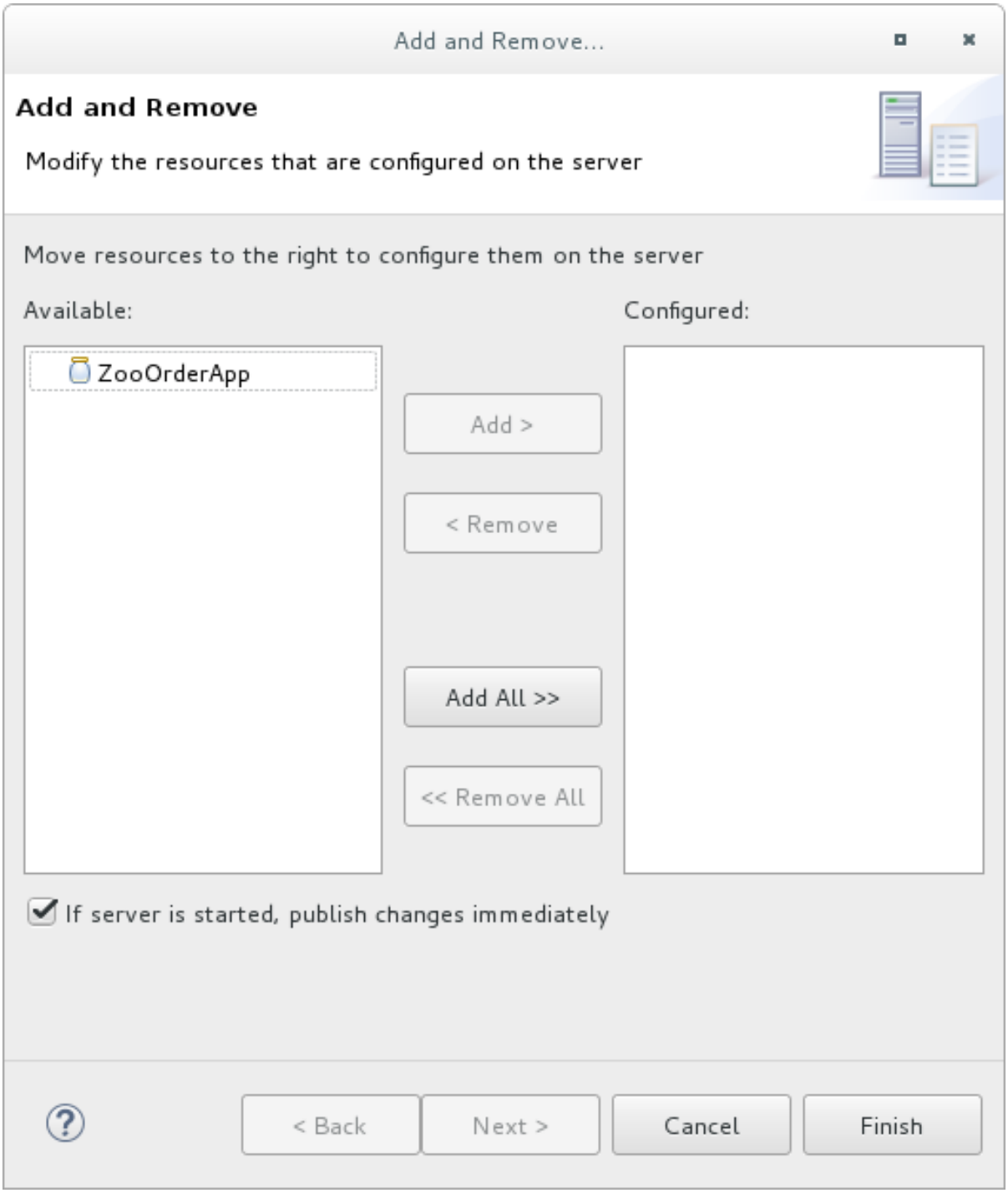


- **Servers 视图显示：**

- **JMX Navigator 显示 *n.n* Runtime Server[Disconnected ：**

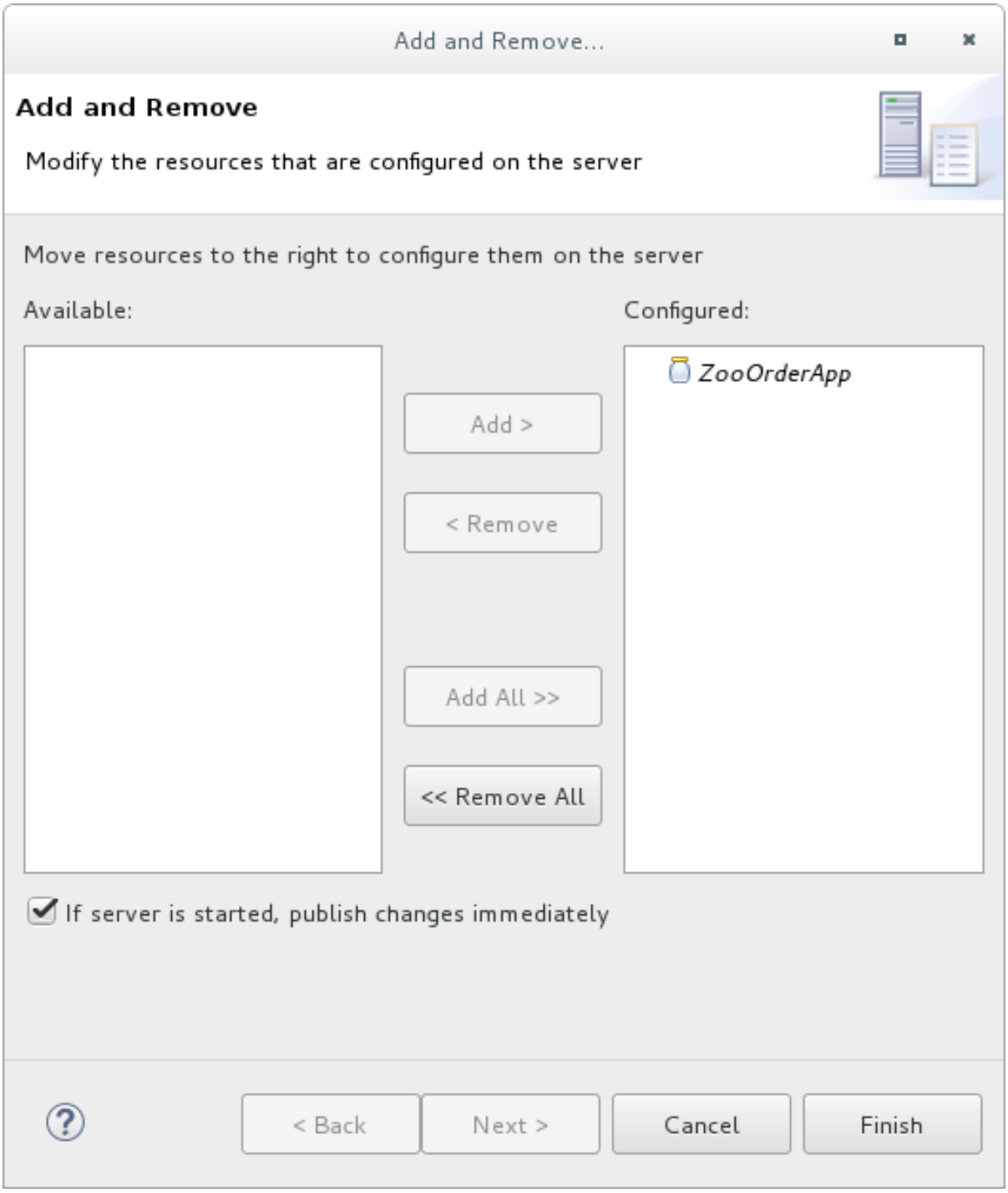


5. **在 Servers 视图中，右键单击 *n.n* Runtime Server [Started]，然后选择 Add and Remove 以打开 Add 和 Remove 页面：**



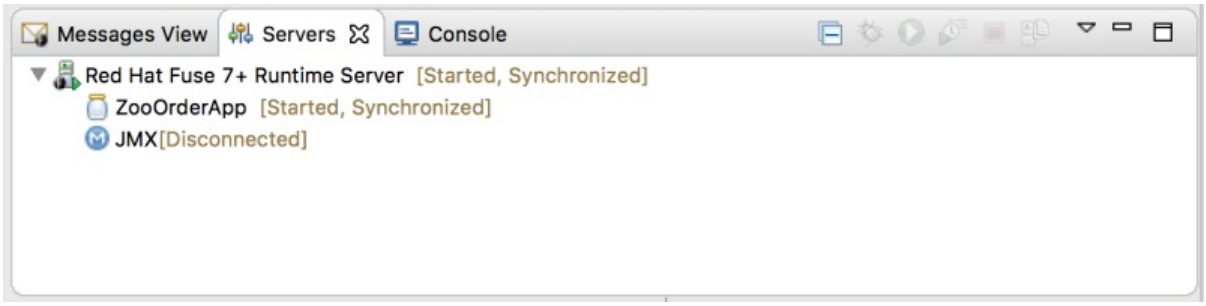
确保 如果服务器启动，则立即检查发布更改。

6. 选择 **ZooOrderApp 并点 Add 将其分配给 Fuse 服务器：**



7. **点 Finish。**

Servers 视图应该显示以下内容：



- **Runtime Server [Started, Synchronized]**

注意

对于服务器，同步 意味着服务器上发布的所有模块都与其本地对应的项相同。

- **ZooOrderApp [Started, Synchronized]**

注意

对于模块，sync 表示公布的模块与其本地对应的模块相同。由于启用了自动发布，因此对 ZooOrderApp 项目所做的更改将以秒为单位发布（根据 发布间隔的值）。

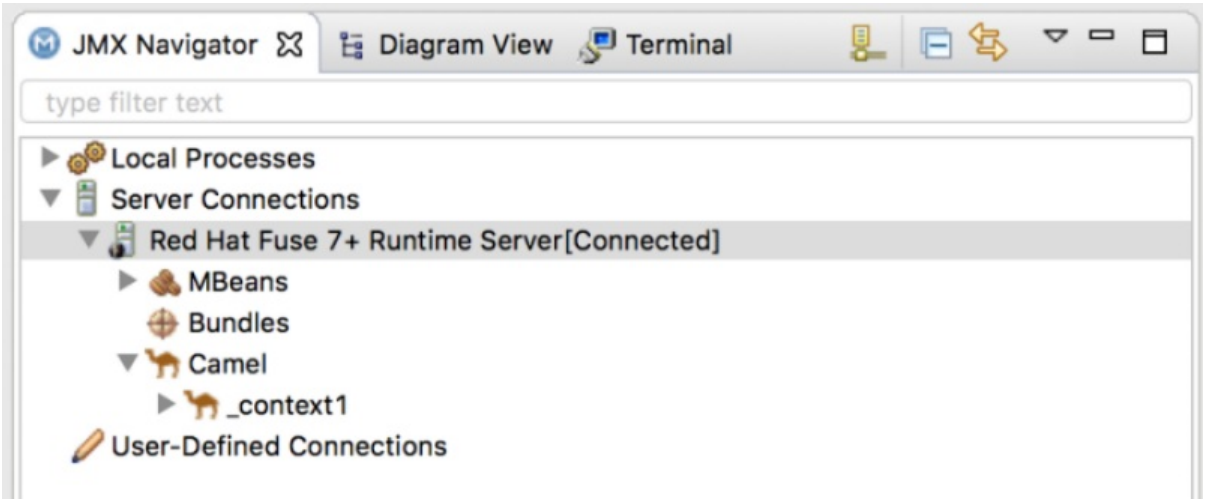
- **JMX[Disconnected]**

连接到运行时服务器

连接到运行时服务器后，您可以看到 ZooOrderApp 项目的已发布元素并与它们交互。

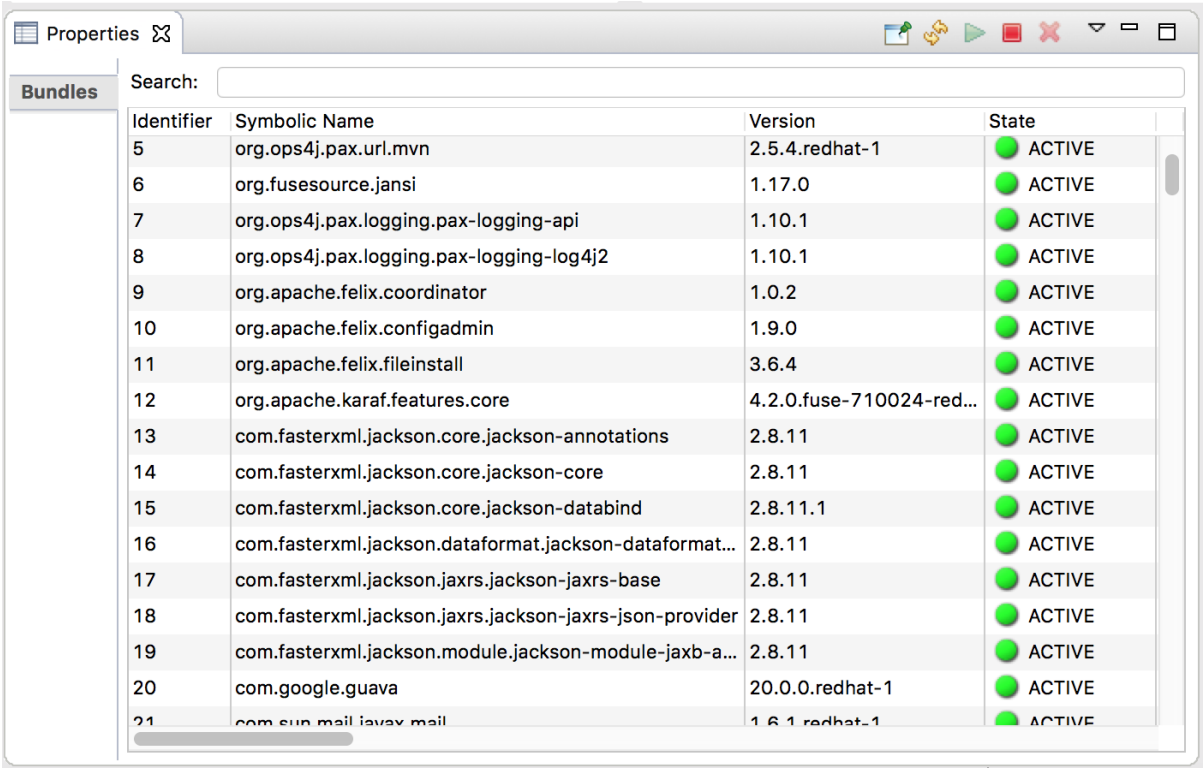
1. 在 Servers 视图中，双击 JMX[Disconnected] 以连接到运行时服务器。

2. 在 JMX Navigator 中，展开 Camel 文件夹以公开 ZooOrderApp 的元素。



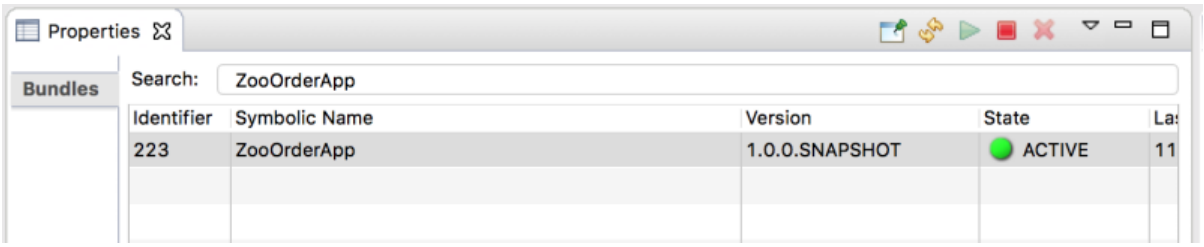
3.

点 Bundles 节点，使用运行时服务器上安装的捆绑包列表填充 Properties 视图：



4.

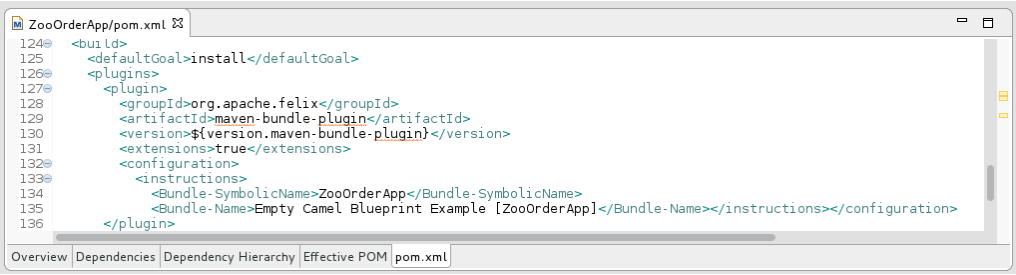
在 Search 字段中，键入 ZooOrderApp。显示对应的捆绑包：



注意

或者，您可以在 Terminal 视图中发出 osgi:list 命令，以查看在服务器运行时上安装的生成的捆绑包列表。该工具对 OSGi 捆绑包使用不同的命名方案，由 osgi:list 命令显示。在这种情况下，命令会返回 Camel Blueprint Quickstart，它出现在安装捆绑包列表的末尾。

在项目的 pom.xml 文件的 < build > 部分中，您可以找到捆绑包的符号链接名称及其捆绑包名称(OSGi)在 maven-bundle-plugin 条目中列出的：



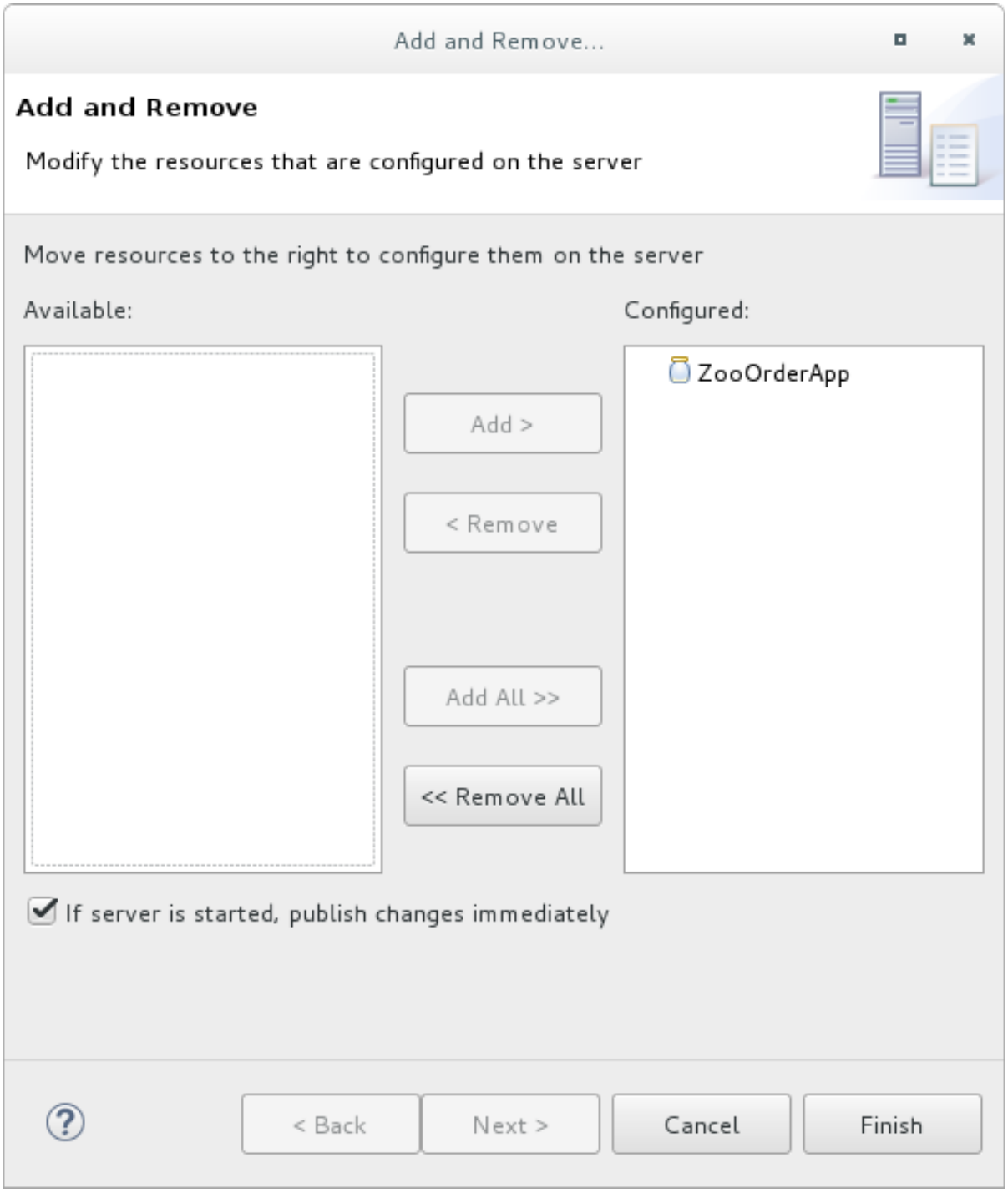
卸载 ZOOORDERAPP 项目

注意

您不需要断开 JMX 连接或停止服务器来卸载发布的资源。

从运行时服务器中删除 ZooOrderApp 资源：

1. **在 Servers 视图中，右键单击 *n.n* Runtime Server 以打开上下文菜单。**
2. **选择 "添加和删除" ：**

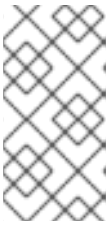


3. **在 Configured 列中，选择 ZooOrderApp，然后单击 Remove 将 ZooOrderApp 资源移到 Available 列中。**

4. **点 Finish。**

5. **在 Servers 视图中，右键单击 JMX[Connected]，然后单击 Refresh。**

   JMX[Connected] 下的 Camel 树会消失。



注意

在 **JMX Navigator** 中，**Server Connections** > ***n.n*** **Runtime Server[Connected]** 下的 **Camel** 树也会消失。

6. 当 **Properties** 视图中显示的 **Bundles** 页面时，向下滚动到列表的末尾，以验证 **ZooOrderApp** 的捆绑包不再被列出。