Red Hat Fuse 7.11

设计 API

为 OpenShift 上的 Fuse 应用程序设计 REST API

Last Updated: 2023-07-26

Red Hat Fuse 7.11 设计 API

为 OpenShift 上的 Fuse 应用程序设计 REST API

法律通告

摘要

指南使用基于 Web 的 REST API Designer

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看我们的 CTO Chris Wright 信息。

第 1 章 API DESIGNER 概述

OpenShift 上的红帽 Fuse 提供 API Designer（基于 Web 的 API 编辑器），可用于设计符合 OpenAPI 规格的 REST API（版本 3 或 2），它是一个厂商中立并可移植的 API 服务打开描述格式。API Designer 是 Apicurio Studio 开源项目的"light"版本(https://www.apicur.io/)。这意味着您的 API Designer 会话无状态，您必须将 API 定义保存为每个会话末尾的 JSON 文件。

您还可以根据 REST API 定义，使用 API Designer 生成初步 Fuse 项目。在 Fuse 开发环境中，您可以完成项目的 Camel 路由并构建项目。最后，您可以在 OpenShift 的 Fuse 上部署生成的 REST 服务。

以下是如何使用 API Designer 在 OpenShift 应用程序解决方案中的 Fuse 中整合 REST API 的概述：

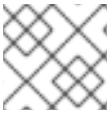
1. 将 API Designer 作为服务添加到 OpenShift 项目。
2. 在 API Designer 中：
   - 使用 API Designer 创建 API 定义。将 REST API 定义保存为本地文件系统的 JSON 文件。您可以在编辑会话中的任意点保存 API 定义，即使 API 定义未完成。
   - 将 API 定义上传到 API Designer。
   - 根据当前的 REST API 定义生成 Fuse Camel 项目。API Designer 提供可下载的 zip 文件，其中包含完整的 Maven 项目。
3. 在您的 Fuse 开发环境中，完成由生成的 Fuse 项目提供的框架实施。
4. 将 Fuse 应用程序构建并部署到 OpenShift。
5. （可选）将 Fuse 应用程序与红帽 3scale API 管理集成，使用 3scale 服务发现功能来查找和配置 Fuse 应用程序。

第 2 章 将 API DESIGNER 作为服务添加到 OPENSHIFT 集群

2.1. 将 API DESIGNER 作为服务添加到 OPENSHIFT 4 项目

对于 OpenShift 4.x，您需要验证 OpenShift 管理员是否已在项目中安装了 API Designer 操作器，如 OpenShift 的 Fuse 指南中所述。

另外，OpenShift 管理员可能还会将 API Designer 作为服务添加到项目中。如果没有，则必须执行该任务。



注意

API Designer **Apicurito** 的前一个名称仍可在 API Designer 操作员的界面中看到。

前提条件

您的 OpenShift 管理员已在 OpenShift 项目中安装了 API Designer 操作器。

流程

1. 在 Web 浏览器中，打开 OpenShift 控制台并使用您的凭证（例如，用户名 **developer** 和密码 **developer**）登录。
2. 选择包含 API Designer 操作器的项目。
3. 选择 **Topology**，验证您看到标记为 **fuse-apicurito** 的图标。
   如果 **apicurito-service-ui** 和 **apicurito-service-generator** 有图标，那么您的 OpenShift 管理员已将 API Designer 作为服务添加到项目中，您可以跳过其余步骤。

   如果没有可用于 Service **-service-ui** 和 **apicurito-service-generator** 的图标，请继续下一步。
4. 在左侧导航窗格中，单击 **Add+**。
5. 在 **Developer Catalog** 部分下，点 **Operator Backed**。
6. 在搜索字段中，键入 **Apicurito** 以过滤目录项。
7. 点 **由红帽卡提供的 Apicurito**。
8. 点 **Create**。
   此时会打开一个默认表单，其中包含 API Designer 实例的最小起始模板。接受默认值，或者（可选）编辑它们。
9. 单击 **Create** 以触发新 API Designer 实例的启动、服务和新 API Designer 实例的其他组件。
10. 要验证 API Designer 服务是否已添加到项目中，请选择 **Topology**，然后确认您看到 **apicurito-service-ui** 和 **apicurito-service-generator** 的图标。
11. 要打开 API 设计器，请单击 **apicurito-service-ui** 图标上的 URL 链接。

2.2. 将 API DESIGNER 作为服务添加到 OPENSHIFT 3.11 项目中

您可以通过从命令行部署 API Designer 模板，将 API Designer 作为服务添加到 OpenShift 3.11 项目。

先决条件

- 按照 OpenShift 系统管理员推荐的准则，获取将允许您访问 API Designer 的主机名。
- 通过在命令窗口中运行以下命令，验证 OpenShift 镜像和模板（包括 **apidesigner-ui** 和 **fuse-apidesigner-generator** ）的 Fuse 是否已安装到 OpenShift 集群：

```
oc get is -n openshift
```

如果没有预安装镜像和模板，或者所提供的版本已过时，安装（或更新）OpenShift 镜像和模板上的 Fuse，如 OpenShift 指南中的 Fuse 所述。

流程

从命令行添加 API Designer 服务：

1. 在命令窗口中登录到 OpenShift 服务器：

```
oc login -u developer -p developer
```

2. 创建新项目命名空间。例如，以下命令会创建一个名为 **myproject** 的新项目：

```
oc new-project myproject
```

3. 运行以下命令，基于 API Designer 模板创建新应用（其中 **myproject** 是项目的名称）：

```
oc new-app -n myproject -f https://raw.githubusercontent.com/jboss-fuse/application-templates/application-templates-2.2.0.fuse-sb2-7_11_0-00022-redhat-00001/fuse-apicurito.yml -p ROUTE_HOSTNAME=myhost
```

注： 另外，您可以通过在 **oc new-app** 命令中添加 additional **-p** 选项来指定其他模板参数。例如，如果您在 OpenShift 镜像和默认 **openshift** 命名空间以外的命名空间中安装了 Fuse，您可以设置 **IMAGE_STREAM_NAMESPACE** 以指定安装 Fuse 镜像流的命名空间：

```
oc new-app -n myproject -f https://raw.githubusercontent.com/jboss-fuse/application-templates/application-templates-2.2.0.fuse-sb2-7_11_0-00022-redhat-00001/fuse-apicurito.yml -p ROUTE_HOSTNAME=myhost -p IMAGE_STREAM_NAMESPACE=othernamespace
```

4. 运行以下命令，获取 API Designer 部署的状态和 URL：

```
oc status
```

如果没有部署 API Designer，请运行以下命令来验证您安装了 **apicurito-ui** 和 **fuse-apicurito-generator** 镜像的正确版本：

```
oc get is -n openshift | grep "apicurito"
```

5. 要从浏览器中访问 API Designer，请使用提供的 URL（例如 https://apicurito.192.168.64.12.nip.io）。

第 3 章 使用 API DESIGNER 设计和开发 API 定义

您可以使用 API Designer 设计和开发符合 OpenAPI 3（或 2）规范的 REST API 定义。

先决条件

- 您创建了 OpenShift 项目。
- 您已将 API Designer 服务添加到 OpenShift 项目。

3.1. 在 API DESIGNER 中创建 REST API 定义

以下步骤描述了如何创建 REST API 定义。

注意

- 您可以从 OpenShift 上的 Fuse Online 或 Fuse 访问 API Designer 用户界面。
- 对于 OpenShift 上的 Fuse，API Designer 是无状态的，这意味着它不会在 OpenShift 会话之间保存您的工作。您需要在会话之间将 API 保存到本地文件系统。

关于示例

Task Management API 示例模拟一个简单的 API，销售顾问可能会用来跟踪在与客户联系交互时需要执行的任务。"to-do"任务示例可以是"为新联系人创建帐户"或"放置现有联系顺序"。要实现任务管理 API 示例，您需要创建两个路径 - 一个用于任务，一个用于特定任务。然后，您可以定义创建任务的操作，检索所有任务或特定任务，更新任务，以及删除任务。

先决条件

- 您知道您要创建的 API 的端点。对于任务管理 API 示例，有两个端点：**/todo** 和 **/todo/{id}**。
- 对于 OpenShift 上的 Fuse，您可以创建一个 OpenShift 项目，并将 API Designer 服务添加到 OpenShift 项目中。

流程

1. 如果您使用 Fuse Online，请跳至第 2 步。
   如果要在 OpenShift 中使用 Fuse：

   a. 登录您的 OpenShift Web 控制台，然后打开包含 API Designer 的项目。

   b. 对于 OpenShift 4.x，选择 **Topology**，然后单击 **apicurito-service-ui** 图标上的 URL 链接。
   对于 OpenShift 3.11，从应用程序列表中点击 API Designer 的 URL，例如 **https://apidesigner-myproject.192.168.64.43.nip.io**

   为 API Designer 打开一个新的浏览器窗口或标签页。

注意

因为 API Designer 是 Apicurio Studio 开源项目 的"light"版本，所以在 API Designer 界面中显示 "Apicurio"。

2. 单击 **New API**。此时会打开一个新的 API 页面。
默认情况下，API Designer 使用 OpenAPI 3 规格。如果要使用 OpenAPI 2 规格，点 **New API** 按钮旁的箭头，然后选择 **OpenAPI 2**。

注意

如果基于 OpenAPI 2 规格打开 API，您可以使用 API Designer 的 **Convert to OpenAPI 3** 选项转换 API 来符合 OpenAPI 3 规格。

3. 更改 API 名称：

   a. 将光标悬停在名称上，然后点显示的编辑图标()。

   b. 编辑名称。例如，键入 **Task API**。

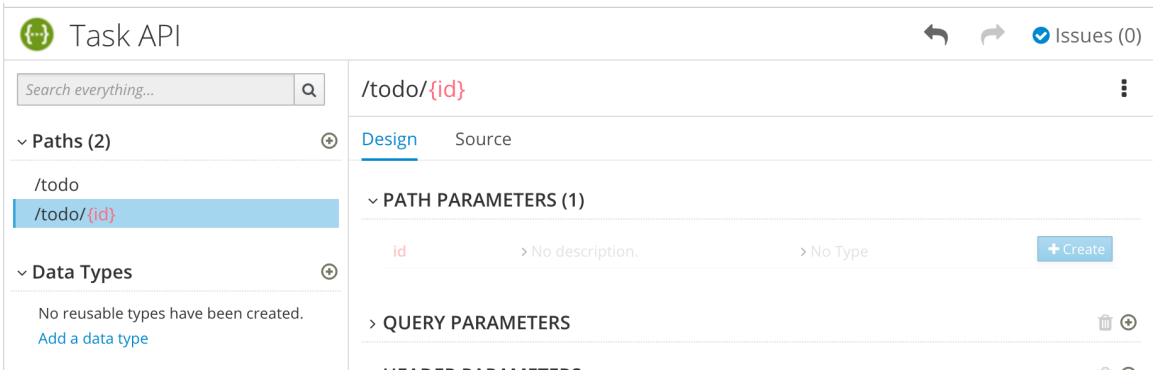
   c. 单击复选标记图标以确认名称更改。

4. （可选）：

   - 提供版本号和描述。
   - 添加联系信息（名称、电子邮件地址和 URL）。
   - 选择一个许可证。
   - 定义标签。
   - 定义一个或多个服务器。
   - 配置安全方案。
   - 指定安全要求。

5. 定义指向 API 各个端点的相对路径。字段名称必须以斜杠(/)开头。
对于 Task Management API 示例，创建两个路径：

   - 任务的路径： **/todo**
   - 按 ID 为特定任务的路径： **/todo/{id}**



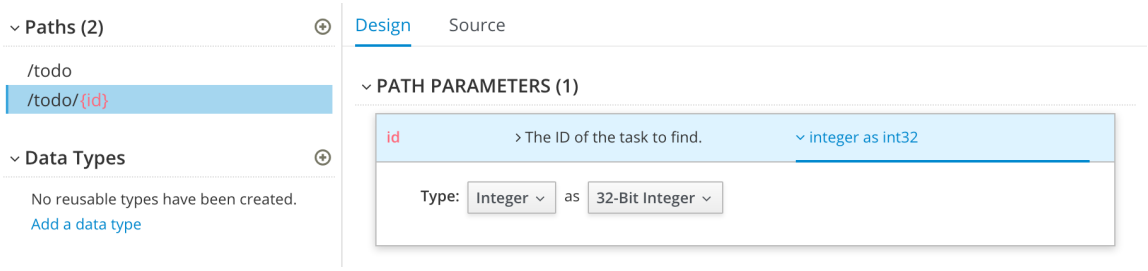
6. 指定任何路径参数的类型。
对于示例 **id** 参数：

   a. 在 **Paths** 列表中，单击 **/todo/{id}**。
   id 参数会出现在 **PATH PARAMETERS** 部分。

b. 点 Create。

c. 作为描述，键入：要查找的任务 ID。

d. 对于类型，选择 32 位整数。



7. 在 Data Types 部分中，为 API 定义可重复使用的类型。

a. 点 Add a data type。

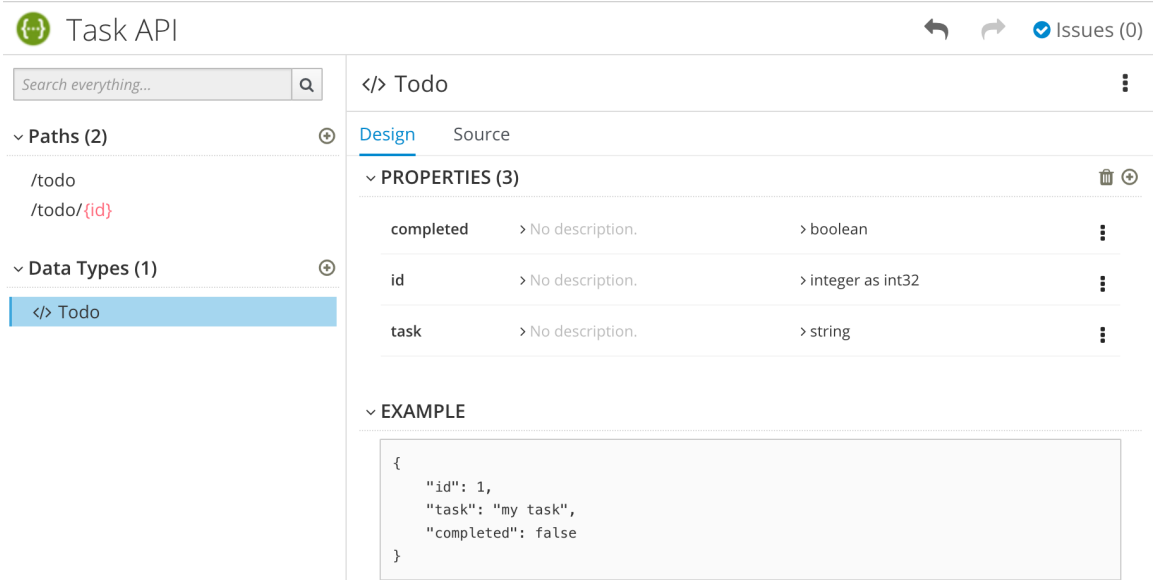
b. 在 Add Data Type 对话框中输入名称。对于 Task Management API 示例，键入Todo。

c. 另外，您还可以提供从哪个 API Designer 创建数据类型 schema 的示例。然后，您可以编辑生成的 schema。
对于 Task Management API 示例，从以下 JSON 示例开始：

```
{
    "id": 1,
    "task": "my task",
    "completed": false
}
```

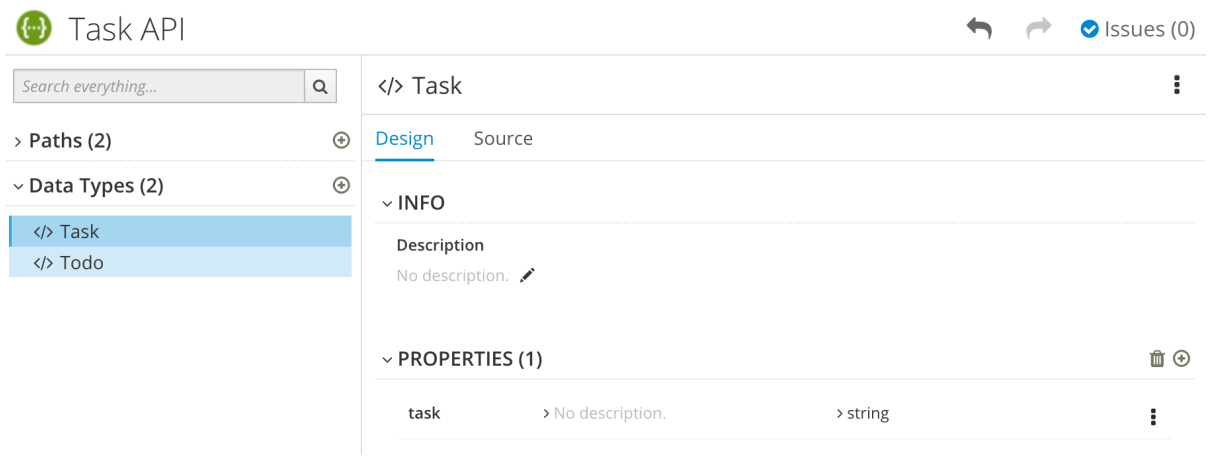
d. 另外，您可以选择使用数据类型创建 REST 资源。

e. 点击 Save。如果您提供了示例，API Designer 从示例生成 schema：



8. 另外，您可以添加编辑模式属性并添加新属性。

9. 对于 Task Management API 示例，使用类型为 string 的属性创建一个名为 Task 的另一个数据类型。



10. **对于每个路径，定义操作（GET、PUT、POST、DE、OPTIONS、HEAD 或 PATCH）。**
对于 Task Management API 示例，按照下表所示定义操作：

表 3.1. 任务管理 API 操作

路径	操作	描述	申请正文	响应
/todo	**POST**	创建新任务。	media Type: **application/json** Data Type: **Task**	• Status Code: **201** Description: **Task created** response Body: Media Type: **application/json** Data Type: **Todo**
/todo	**GET**	获取所有任务。	*Not applicable*	• Status Code： **200** Description：**任务列表**

路径	操作	描述	申请正文	响应
/todo/{id}	**GET**	按 ID 获取任务。	*Not applicable*	• Status Code: **200** Description: **Task found for ID** Response Body: Media Type: **application/json** Data type: **Task** • Status Code: **404** Description: **No Task with provided identifier.**
/todo/{id}	**PUT**	按 ID 更新任务。	request Body type: **Task**	• Status Code: **200** Description: **Completed** • Status Code： **400** Description: **Task not updated**
/todo/{id}	**删除**	按 ID 删除任务。	*Not applicable*	• Status Code: **200** Description: **Task delete** • Status Code： **400** Description: **Task not delete**

11. **解决所有问题，如 解决 API Designer 中的验证问题中所述。**

12. **对于 OpenShift 上的 Fuse，请点击 Save As 来保存 API 规格，然后选择 JSON 或 YAML 格式。JSON 或 YAML 文件下载到您的本地下载文件夹。默认文件名是 openapi-spec 具有适当文件扩展名。**

其他资源

- 有关 OpenAPI 规格的详情，请转至 ：https://github.com/OAI/OpenAPI-Specification

3.2. 解决 API DESIGNER 中的验证问题

当您创建和编辑 API 时，API Designer 会识别出问题，您必须使用感叹号(!)图标以及 API Designer 标题栏中的问题列表。

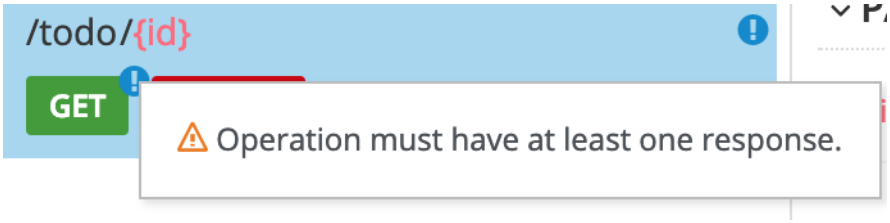
先决条件

- 在 API Designer 中打开 API。

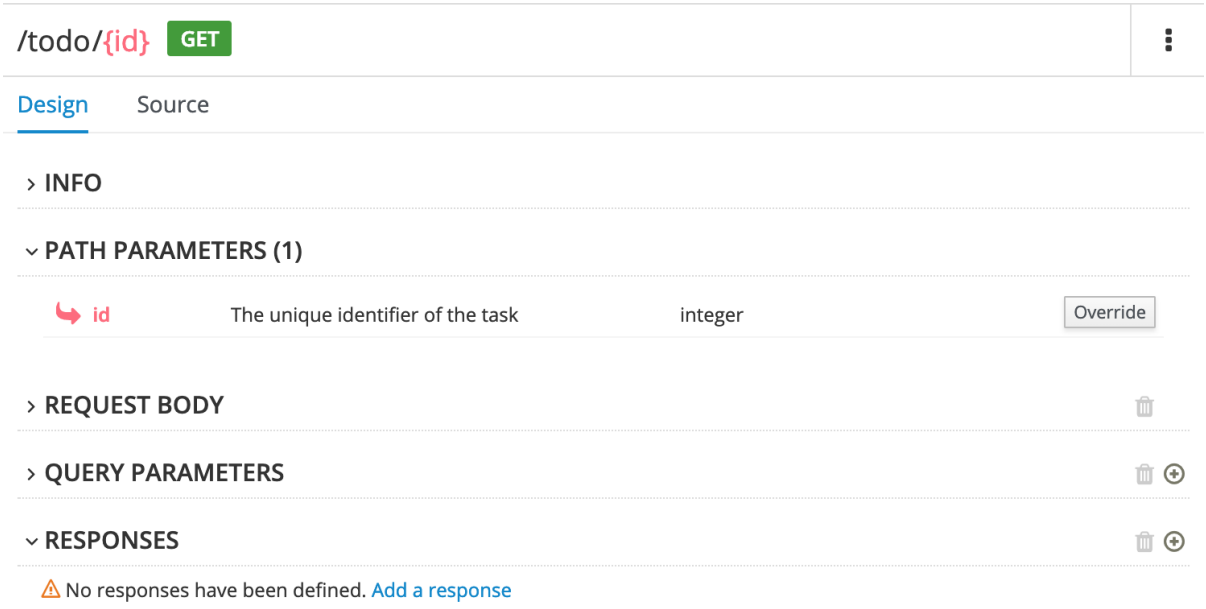
流程

1. 找到一个以感叹号(!)图标表示的问题。例如：

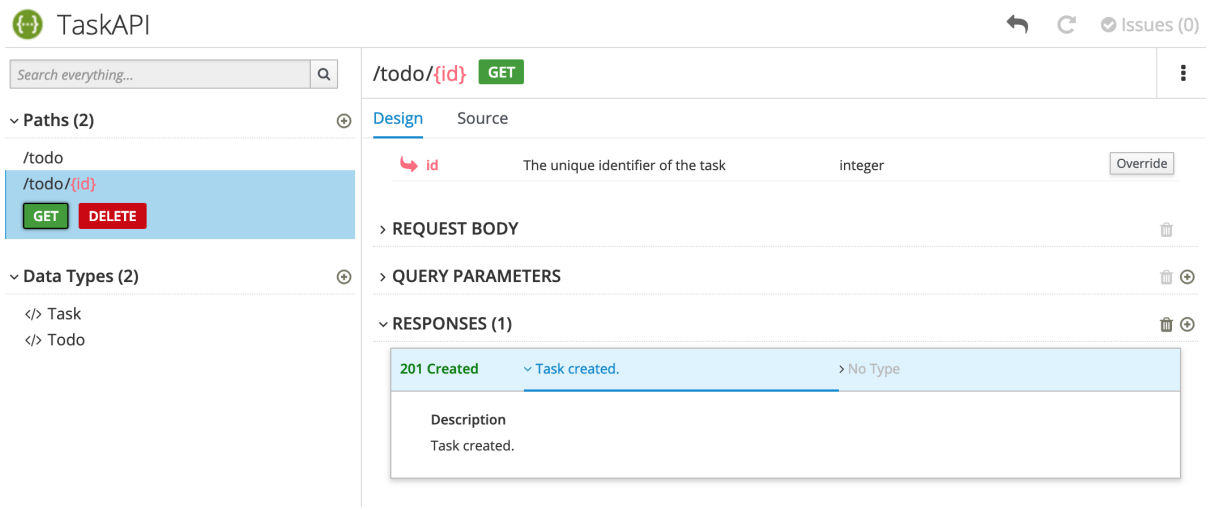
2. 点感叹图标查看问题的描述。例如：



3. 根据问题描述提供的信息，导航到问题的位置并加以修复。
例如，要修复"Operation 必须至少有一个响应"问题，请单击 GET 操作来打开它，然后单击 Add a response。



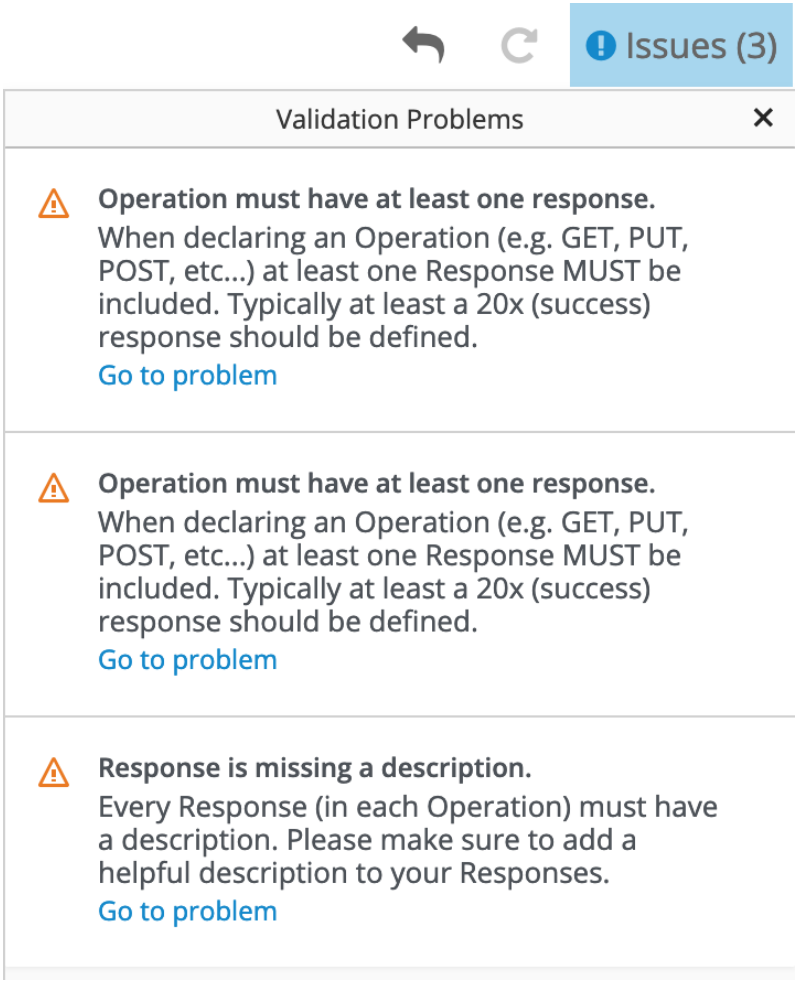
为响应输入描述后，这个问题会解决，感叹号图标会消失：



4. **有关所有问题的摘要：**

   a. **单击右上角的 issues 链接。**

   b. **点 Go 进入 特定问题进入问题的位置，以便您可以解决它。**



第 4 章 基于 REST API 实施、构建和部署 FUSE 应用程序

您可以使用 Red Hat Fuse API Designer 根据 REST API 定义生成 Camel Fuse 项目。在 Fuse 开发环境中，您可以完成 Camel 路由和 Rest DSL API。最后，您可以构建项目，并将生成的应用程序部署到 OpenShift 中的 Fuse 中。

先决条件

- 您有一个现有的 API 定义，它符合 OpenAPI 3（或 2）规范。例如，使用 API Designer 创建的 **openapi-spec.json** 文件。
- API Designer 已安装并在本地 OpenShift 集群上运行。
- 您有一个现有的 OpenShift 项目，其中 API Designer 添加为服务。
- 已安装 Maven 和红帽 Fuse。

以下主题描述了如何根据 REST API 实施、构建和部署 Fuse 应用程序：

- 第 4.1 节 “上传 API 定义到 API Designer”
- 第 4.2 节 “从 API Designer 生成 Fuse Camel 项目”
- 第 4.3 节 “完成 API Designer 生成的 Camel 项目”
- 第 4.4 节 “构建和部署 REST 服务”

4.1. 上传 API 定义到 API DESIGNER

您可以将现有的 API 定义上传到 API Designer。

先决条件

- 您有一个现有的 API 定义，它符合 OpenAPI 3（或 2）规范。例如，使用 API Designer 创建的 **openapi.json** 文件。
- API Designer 已安装并在本地 OpenShift 集群上运行。
- 您有一个现有的 OpenShift 项目，其中 API Designer 添加为应用。

流程

1. 在 OpenShift Web 控制台中，打开包含 API Designer 的项目。
2. 打开 API Designer 控制台。在项目的应用程序列表中，单击 apidesigner 下的 URL。例如：
   https://apidesigner-myproject.192.168.64.38.nip.io
   API Designer 控制台在一个单独的 Web 浏览器选项卡或窗口中打开。
3. 单击 Open API。
   此时会打开文件管理器窗口。
4. 在文件管理器窗口中：
   a. 导航到包含现有 OpenAPI 定义文件的文件夹，如 **openapi.json**。
   b. 选择 OpenAPI 定义文件，然后点 Open。

OpenAPI 定义在 API Designer 控制台中打开。

4.2. 从 API DESIGNER 生成 FUSE CAMEL 项目

您可以使用 API Designer 根据 API 定义生成 Fuse Camel 项目。

先决条件

- API Designer 已安装并在本地 OpenShift 集群上运行。
- 您有一个现有的 OpenShift 项目，其中 API Designer 添加为应用。
- 您已在 API Designer 控制台中创建或打开 API 定义文件。

流程

在 API Designer 控制台中：

1. 点 Generate。
2. 从下拉列表中选择 Fuse Camel Project。

API Designer 生成 **camel-project.zip** 文件，并将其下载到您的本地默认下载文件夹。

zip 文件包含 Fuse Camel 项目，它使用 Camel 的 Rest DSL 提供默认框架实施 API 定义，并包含所有资源操作。该项目还包括您用来生成项目的原始 OpenAPI 定义文件。

4.3. 完成 API DESIGNER 生成的 CAMEL 项目

API Designer 生成 Fuse 项目，它使用 Camel 的 Rest DSL 和覆盖所有资源操作提供默认的 API 定义实施。在 Fuse 开发环境中，您将完成该项目。

先决条件

- 您有 API Designer 生成的 **camel-project.zip** 文件。
- （可选）您已使用 Fuse 工具安装 Red Hat Developer Studio。

流程

1. 将 API Designer-generated **camel-project.zip** 文件解压缩到临时文件夹。
2. 打开 Red Hat Developer Studio。
3. 在 Developer Studio 中，选择 File → Import。
4. 在 Import 对话框中，选择 Maven → Existing Maven Projects。
5. 在编辑器中打开项目的 **camel-context.xml** 文件。
6. 单击 REST 选项卡，以编辑 Rest DSL 组件。
   有关定义 REST 服务的信息，请参阅 Apache Camel 开发指南中的 "定义 REST 服务"部分。

   有关使用 Swagger 支持扩展 JAX-RS 端点的详情，请参考 Apache CXF 开发指南。

有关使用 Fuse 工具 REST 编辑器的详情，请参考工具 指南中的 "查看和编辑 Rest DSL 组件"部分。

7. 在 Design 选项卡中，编辑 Camel 路由。
有关编辑 Camel 路由的信息，请参阅 工具用户指南中的"编辑路由上下文"一节。

4.4. 构建和部署 REST 服务

完成 Fuse 项目后，您可以在 OpenShift 中构建和部署项目。

先决条件

- 您有一个定义 REST 服务的完整 error-free Fuse 项目。
- 已安装 Java 8 JDK（或更新版本）和 Maven 3.3.x（或更新版本）。

流程

如果您有一个单节点 OpenShift 集群，如 Minishift 或 Red Hat Container Development Kit，则 已安装并运行，您可以在其中部署项目。

将此项目部署到正在运行的单节点 OpenShift 集群：

1. 登录到您的 OpenShift 集群：

```
$ oc login -u developer -p developer
```

2. 为项目创建一个新的 OpenShift 项目。例如，以下命令会创建一个名为 test-deploy 的新项目。

```
$ oc new-project test-deploy
```

3. 将目录改为包含 Fuse Camel 项目的文件夹（如 myworkspace/camel-project）：

```
$ cd myworkspace/camel-project
```

4. 将项目构建并部署到 OpenShift 集群：

```
$ mvn clean fabric8:deploy -Popenshift
```

5. 在您的浏览器中，打开 OpenShift 控制台并导航到项目（如 test-deploy）。等待直到您看到 camel-project 应用的 Pod 已启动。

6. 在项目的 Overview 页面上，找到 camel-project 应用程序的 URL。URL 使用以下格式：
http://camel-project-MY_PROJECT_NAME.OPENSHIFT_IP_ADDR.nip.io。

7. 点 URL 来访问该服务。

第 5 章 为 3SCALE 发现准备 API 服务

Red Hat 3scale API Management 是红帽的产品，可让您监管对公共互联网上的 API 服务的访问。3scale 的功能包括实施服务级别协议(SLA)的功能，管理 API 版本，提供安全性和身份验证服务等。Fuse 支持 3scale 的 *服务发现功能*，它可以轻松地从 3scale 管理门户 UI 发现 Fuse 服务。通过使用服务发现，您可以扫描在同一 OpenShift 集群中运行的 Fuse 应用程序，并将关联的 API 定义自动导入到 3scale。

先决条件

- 在 OpenShift 中部署并运行了提供 API 服务的 Fuse 应用。
- Fuse 应用标注了必要的注释，使它可以被 3scale 发现。

注意

由 API Designer 生成的 Fuse 项目预先配置为自动提供必要的注解。

对于不是由 API Designer 生成的 Fuse 项目，您必须配置项目，如 API Designer 生成的 Fuse 项目添加注解 中所述。

- 3scale API 管理系统被部署到与要发现的 API 服务 *相同的* OpenShift 集群上。

有关在 3scale 中发现 API 服务的步骤详情，请参阅 Red Hat 3scale API Management Portal 指南中的服务发现部分。

其他资源

- Red Hat 3scale API Management 产品页
- Red Hat 3scale API Management 文档

5.1. 为 API DESIGNER 生成的 FUSE 项目添加注解

为了使 3scale 能够发现 API 服务，提供 API 服务的 Fuse 应用程序必须包含 Kubernetes Service Annotations，使其可被发现。这些注解由 Service Discovery Enricher 提供，它作为 OpenShift Maven 插件的一部分。

对于 Apache Camel Rest DSL 项目，OpenShift Maven 插件默认运行服务发现增强器。

由 API Designer 生成的 Fuse 项目预先配置为自动提供所需的注解。

流程

对于不是由 API Designer 生成的 Fuse Rest DSL 项目，请按如下所示配置项目：

1. 编辑 Fuse 项目的 pom.xml 文件，使其包含 openshift-maven-plugin 依赖项，如下例所示：

```
<plugin>
    <groupId>org.jboss.redhat-fuse</groupId>
    <artifactId>openshift-maven-plugin</artifactId>
    <version>${fuse.version}</version>
    <executions>
      <execution>
        <goals>
```

```
            <goal>resource</goal>
            <goal>build</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
```

如果满足某些项目级别条件（例如，项目必须是 Camel Rest DSL 项目），OpenShift Maven 插件将运行服务发现功能。您不需要将 Service Discovery Enricher 指定为 pom.xml 文件中的依赖项，除非您要自定义 ener 的行为（如 自定义 API 服务注解值 所述）。

2. 在 Fuse Rest DSL 项目的 camel-context.xml 文件中，在 restConfiguration 元素中指定以下属性：

   - 方案 ：托管服务的 URL 的方案部分。您可以指定 "http" 或 "https"。
   - contextPath ：API 服务托管的 URL 的路径部分。
   - apiContextPath ：API 服务描述文档托管的位置的路径。如果文档是自托管，或者完整 URL 是外部的，您可以指定相对路径。
     以下来自 camel-context.xml 文件示例摘录显示了 restConfiguration 元素中的注解属性值：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
       http://camel.apache.org/schema/spring
http://camel.apache.org/schema/spring/camel-spring.xsd">

    <camelContext xmlns="http://camel.apache.org/schema/spring">
        <restConfiguration component="servlet" scheme="https"
            contextPath="myapi" apiContextPath="myapi/openapi.json"/>
...
```

增强器使用这些 restConfiguration 元素值提供的信息，为 discovery.3scale.net/scheme、discovery.3scale.net/path 以及 discovery.3scale.net/description-path 注解创建 discovery.3scale.net/description-path 注解，因此，该项目由 3scale 部署 OpenShift 服务被 3scale 发现，如红帽 3scale API 管理门户 中的服务发现部分所述。

enricher 添加以下标签和注解，以使服务可以被 3scale 发现：

- discovery.3scale.net 标签：默认情况下，enricher 将此值设置为 "true"。3scale 执行选择器定义以查找需要发现的所有服务时，它会使用此标签。
- 以下注解：
  - discovery.3scale.net/discovery-version:（可选）3scale 发现过程的版本。enricher 默认将此值设置为 "v1"。
  - discovery.3scale.net/scheme: 托管该服务的 URL 的方案部分。增强器会使用默认的 "http"，除非您在 restConfiguration 元素的 scheme 属性中覆盖它。另一个可能的值是 "https"。

- **discovery.3scale.net/path** ：托管该服务的 URL 的路径部分。如果路径为 "/"，则省略此注解。enricher 从 **restConfiguration** 元素的 **path** 属性中获取这个值。
- **discovery.3scale.net/port** ：服务的端口。增强器从 Kubernetes 服务定义中获取这个值，其中包含其公开的服务端口号。如果 Kubernetes 服务定义公开多个服务，则增强器会使用列出的第一个端口。
- **discovery.3scale.net/description-path**:（可选）OpenAPI 服务描述文档的路径。enricher 从 **restConfiguration** 元素的 **contextPath** 属性中获取这个值。

您可以自定义 Service Discovery Enricher 的行为，如 自定义 API 服务注解值 中所述。

5.2. 自定义 API 服务注解值

默认情况下，Maven Fabric8 插件运行 Fabric8 服务发现功能增强器。增强器向 Fuse Rest DSL 项目的 API 服务添加了注解，以便 API 服务可以被 3scale 发现，如 Red Hat 3scale API 管理 *管理门户指南中的使用服务发现* 所述。

enricher 为一些注解使用默认值，并从项目的 **camel-context.xml** 文件中获取其他注解的值。

您可以通过在 Fuse 项目 **pom.xml** 文件中定义值或 **service.yml** 文件中定义的值来覆盖 **camel-context.xml** 文件中定义的值。（如果您在两个文件中都定义了值，enricher 将使用 **service.yml** 文件中的值。） 请参阅 Fabric8 Service Discovery Enricher 元素以了解您可以为 Fabric8 Service Discovery Enricher 指定元素的描述。

流程

在 Fuse 项目 **pom.xml** 文件中指定注解值：

1. 在您选择的编辑器中打开 Fuse 项目的 **pom.xml** 文件。
2. 找到 **openshift-maven-plugin** 依赖项，如下例所示：

```
<plugin>
      <groupId>org.jboss.redhat-fuse</groupId>
      <artifactId>openshift-maven-plugin</artifactId>
      <version>${fuse.version}</version>
      <executions>
        <execution>
          <goals>
            <goal>resource</goal>
            <goal>build</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
```

3. 将 Fabric8 Service Discovery Enricher 作为依赖项添加到 openshift-maven 插件，如下例所示。

```
<plugin>
  <groupId>org.jboss.redhat-fuse</groupId>
  <artifactId>openshift-maven-plugin</artifactId>
  <version>${fuse.version}</version>
  <executions>
    <execution>
```

```
      <goals>
        <goal>resource</goal>
        <goal>build</goal>
      </goals>
    </execution>
  </executions>
  <dependencies>
    <dependency>
      <groupId>io.acme</groupId>
      <artifactId>myenricher</artifactId>
      <version>1.0</version>
      <configuration>
        <enricher>
          <config>
            <f8-service-discovery>
              <scheme>https</scheme>
              <path>/api</path>
              <descriptionPath>/api/openapi.json</descriptionPath>
            </f8-service-discovery>
          </config>
        </enricher>
      </configuration>
      </dependency>
    </dependencies>
</plugin>
```

4. **保存您的更改。**

另外，您可以使用 src/main/fabric8/service.yml 片段覆盖注解值，如下例所示：

```
kind: Service
name:
metadata:
  labels:
    discovery.3scale.net/discoverable : "true"
  annotations:
    discovery.3scale.net/discovery-version : "v1"
    discovery.3scale.net/scheme : "https"
    discovery.3scale.net/path : "/api"
    discovery.3scale.net/port : "443"
    discovery.3scale.net/description-path : "/api/openapi.json"
spec:
  type: LoadBalancer
```

5.3. FABRIC8 SERVICE DISCOVERY 增强了元素

下表描述了您为 Fabric8 Service Discovery Enricher 指定的元素（如果您想要覆盖默认值）和 camel-context.xml 文件中定义的值。

您可以在 Fuse Rest DSL 项目的 pom.xml 文件中定义这些值，也可以在 src/main/fabric8/service.yml 文件中定义。（如果您在两个文件中都定义了值，enricher 将使用 service.yml 文件中的值。）如需示例，请参阅自定义 API 服务注解值。

表 5.1. Fabric8 Service Discovery 增强了元素

元素	描述	默认
springDir	包含 **camel-context.xml** 文件的 spring 配置目录的路径。	**/src/main/resources/spring** 路径用于识别 Camel Rest DSL 项目。
scheme	托管该服务的 URL 的方案部分。您可以指定 "http" 或 "https"。	http
path	托管 API 服务的 URL 的路径部分。	
port	托管 API 服务的 URL 的端口部分。	80
descriptionPath	托管 API 服务描述文档的位置的路径。如果文档是自托管，或者完整 URL 是外部的，您可以指定相对路径。	
discoveryVersion	3scale 发现实现的版本。	v1
发现	将 **discovery.3scale.net** 标签设置为 **true** 或 **false** 的元素。 如果设置为 **true**，则 3scale 将尝试发现此服务。 如果设置为 **false**，则 3scale 将无法尝试发现此服务。 您可以将这个元素用作交换机，通过将它设置为 "false" 来临时关闭 3scale 发现集成。	如果您没有指定值，则增强器会尝试自动探测到该服务是否可以被发现。如果增强器确定服务无法发现，3scale 将无法尝试发现这个服务。