Red Hat Fuse 7.10

API の設計

OpenShift で Fuse アプリケーション用の REST API を設計する

Last Updated: 2023-07-29

Red Hat Fuse 7.10 API の設計

OpenShift で Fuse アプリケーション用の REST API を設計する

法律上の通知

概要

Web ベース REST API Designer の使用ガイド

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、CTO である Chris Wright のメッセージ をご覧ください。

第1章 API DESIGNER の概要

Red Hat Fuse on OpenShift は、Web ベースの API エディターである API Designer を提供します。API Designer を使用して API サービスの特定ベンダーに依存しない移植可能なオープン記述形式である OpenAPI 仕様 (バージョン 3 または 2) に準拠する REST API を設計することができます。API Designer は、軽量バージョンの Apicurio Studio オープンソースプロジェクト (https://www.apicur.io/) です。これは、API Designer セッションはステートレスであり、各セッションの終わりに API 定義を JSON ファイルとして保存する必要があることを意味しています。

また、API Designer を使用して REST API 定義に基づいて事前の Fuse プロジェクトを生成することができます。その後、Fuse 開発環境でプロジェクトの Camel ルートを完了し、プロジェクトをビルドすることができます。最後に、生成される REST サービスを Fuse on OpenShift にデプロイすることができます。

以下は、API Designer を使用して REST API を Fuse on OpenShift のアプリケーションソリューションに組み込む方法についての概要です。

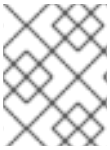
1. API Designer を OpenShift プロジェクトにサービスとして追加します。
2. API Designer で、以下を実行します。
   - API Designer で API 定義を作成します。REST API 定義を JSON ファイルとしてローカルファイルシステムに保存します。API 定義が完了していなくても、編集セッション中にいつでも API 定義を保存することができます。
   - API 定義を API Designer にアップロードします。
   - 現在の REST API 定義に基づいて Fuse Camel プロジェクトを生成します。API Designer は、完全な Maven プロジェクトが含まれるダウンロード可能な zip ファイルを提供します。
3. Fuse 開発環境で、生成される Fuse プロジェクトによって提供されるスケルトン実装を完了します。
4. Fuse アプリケーションをビルドし、OpenShift にデプロイします。
5. (任意の手順) Fuse アプリケーションを見つけ、これを設定するために 3scale サービス検出機能を使用して Fuse アプリケーションを Red Hat 3scale API Management に統合します。

第2章 API DESIGNER の OPENSHIFT クラスターへのサービスとしての追加

2.1. API DESIGNER をサービスとして OPENSHIFT 4 プロジェクトに追加

OpenShift 4.x では、Fuse on OpenShift ガイド の説明どおりに、OpenShift 管理者がプロジェクトに API Designer Operator をインストールしたことを確認する必要があります。

OpenShift 管理者が任意で API Designer をサービスとしてプロジェクトに追加した可能性もあります。そうでない場合は、それを行う必要があります。



注記

API Designer の旧名である **Apicurito** は今でも API Designer Operator のインターフェイスに表示されます。

前提条件

OpenShift 管理者が API Designer Operator を OpenShift プロジェクトにインストールしている。

手順

1. Web ブラウザーで OpenShift コンソールを開き、認証情報 (例: ユーザー名 **developer** およびパスワード **developer**) を使用してログインします。
2. API Designer Operator が含まれるプロジェクトを選択します。
3. **Topology** を選択し、**fuse-apicurito** というラベルが付けられたアイコンが表示されることを確認します。
   apicurito-service-ui および **apicurito-service-generator** のアイコンがある場合は、OpenShift 管理者は API Designer をサービスとしてプロジェクトに追加済みなので、残りのステップを省略できます。

   apicurito-service-ui および **apicurito-service-generator** のアイコンがない場合は、次の手順に進みます。
4. 左側のナビゲーションペインで **Add+** をクリックします。
5. **Developer Catalog** セクションで、**Operator Backed** をクリックします。
6. 検索フィールドに **Apicurito** と入力し、カタログソースをフィルターします。
7. **Apicurito provided by Red Hat**カードをクリックします。
8. **Create** をクリックします。
   API Designer インスタンスの最小限の開始テンプレートがあるデフォルトのフォームが開かれます。デフォルト値を使用するか、任意でデフォルト値を編集します。
9. **Create** をクリックし、新しい API Designer インスタンスの Pod、サービス、およびその他のコンポーネントの起動をトリガーします。
10. API Designer サービスがプロジェクトに追加されたことを確認するには、**Topology** を選択し、**apicurito-service-ui** および **apicurito-service-generator** のアイコンが表示されることを確認します。

11. API デザイナーを開くには、**apicurito-service-ui** アイコンの URL リンクをクリックします。

2.2. API DESIGNER をサービスとして OPENSHIFT 3.11 プロジェクトに追加

コマンドラインから API Designer テンプレートをデプロイすることで、API Designer をサービスとして OpenShift 3.11 プロジェクトに追加できます。

前提条件

- OpenShift システム管理者が推奨するガイドラインに従って、API Designer へのアクセスが可能なホスト名を取得します。
- コマンドウィンドウで以下のコマンドを実行して、Fuse on OpenShift イメージとテンプレート (**apidesigner-ui** および **fuse-apidesigner-generator** を含む) が OpenShift クラスターにインストールされていることを確認します。

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

イメージおよびテンプレートが事前にインストールされていない場合や、提供されたバージョンが古い場合は、Fuse on OpenShift Guide に説明されているように Fuse on OpenShift イメージおよびテンプレートをインストール (または更新) します。

手順

コマンドラインから API Designer サービスを追加するには、以下を実行します。

1. コマンドウィンドウで、OpenShift サーバーにログインします。

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

2. 新規プロジェクト namespace を作成します。たとえば、以下のコマンドは **myproject** という名前の新規プロジェクトを作成します。

```
oc new-project myproject
```

3. 次のコマンド (**myproject**はプロジェクトの名前) を実行して、APIDesigner テンプレートに基づいて新しいアプリケーションを作成します。

```
oc new-app -n myproject -f https://raw.githubusercontent.com/jboss-fuse/application-templates/application-templates-2.1.0.fuse-sb2-7_10_0-00015-redhat-00001/fuse-apicurito.yml -p ROUTE_HOSTNAME=myhost
```

注: オプションで、**oc new-app** コマンドに追加の **-p** オプションを追加することにより、他のテンプレートパラメーターを指定できます。たとえば、Fuse on OpenShift イメージおよびテンプレートをデフォルトの **openshift** namespace 以外の namespace にインストールしている場合、**IMAGE_STREAM_NAMESPACE** を設定して、Fuse イメージストリームがインストールされている namespace を指定できます。

```
oc new-app -n myproject -f https://raw.githubusercontent.com/jboss-fuse/application-templates/application-templates-2.1.0.fuse-sb2-7_10_0-00015-redhat-00001/fuse-apicurito.yml -p ROUTE_HOSTNAME=myhost -p IMAGE_STREAM_NAMESPACE=othernamespace
```

4. 以下のコマンドを実行して、API Designer デプロイメントのステータスおよび URL を取得します。

```
oc status
```

API Designer がデプロイされていない場合は、次のコマンドを実行して、正しいバージョンの **apicurito-ui** および **fuse-apicurito-generator** イメージがインストールされていることを確認します。

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

5. ブラウザーから API Designer にアクセスするには、提供される URL (例: https://apicurito.192.168.64.12.nip.io) を使用します。

第3章 API DESIGNER を使用した API 定義の設計および開発

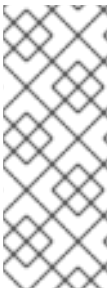
API Designer を使用して、OpenAPI 3 (または 2) 仕様に準拠する REST API 定義を設計し、開発できます。

前提条件

- OpenShift プロジェクトが作成済みである必要があります。
- API Designer サービスが OpenShift プロジェクトに追加済みである必要があります。

3.1. API DESIGNER での REST API 定義の作成

以下の手順では、REST API 定義を作成する方法を説明します。



注記

- Fuse Online または Fuse on OpenShift から API Designer ユーザーインターフェイスにアクセスできます。
- Fuse on OpenShift のみの場合、API Designer はステートレスです。つまり、作業が OpenShift セッション間で保存されません。セッションごとに API をローカルファイルシステムに保存する必要があります。

サンプルについて

Task Management API のサンプルは、営業コンサルタントがお客様側の担当者と対話する際に必要なタスクを追跡するために使用する可能性のある単純な API をシミュレートします。to-do タスクの例には create an account for a new contact (新規の担当者のアカウントの作成) や place an order for an existing contact (既存の担当者の注文処理) が含まれる可能性があります。Task Management API のサンプルを実装するには、複数のタスク用のパスと、特定のタスク用のパスの 2 つのパスを作成します。その後、タスクを作成し、すべてのタスクまたは特定のタスクを取得し、タスクを更新してタスクを削除する操作を定義します。

前提条件

- 作成する必要のある API のエンドポイントを把握している。Task Management API のサンプルの場合、**/todo** および **/todo/{id}** の 2 つのエンドポイントがあります。
- Fuse on OpenShift のみの場合、OpenShift プロジェクトを作成しており、API Designer サービスを OpenShift プロジェクトに追加している。

手順

1. Fuse Online を使用している場合は、ステップ 2 に進みます。
   Fuse on OpenShift を使用している場合:
   a. OpenShift Web コンソールにログインし、API Designer が含まれるプロジェクトを開きます。
   b. OpenShift 4.x の場合は、**Topology** を選択し、**apicurito-service-ui** アイコンの URL リンクをクリックします。
   OpenShift 3.11 の場合は、アプリケーションの一覧で API Designer の URL をクリックします (例: **https://apidesigner-myproject.192.168.64.43.nip.io**)。

API Designer 用の新規ブラウザーウィンドウまたはタブが開きます。

注記

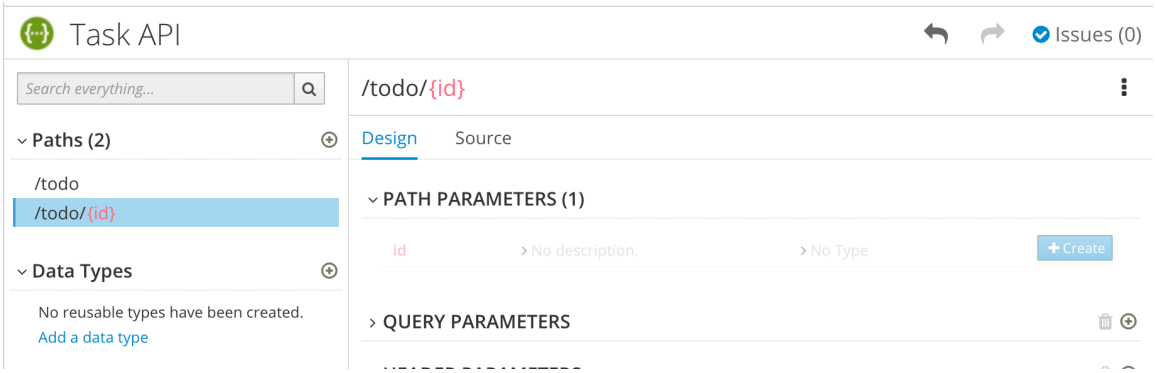
API Designer は軽量バージョンの Apicurio Studio オープンソースプロジェクト であるため、Apicurio が API Designer インターフェイスに表示されます。

2. **New API** をクリックします。新規の API ページが開きます。
デフォルトでは、API Designer は OpenAPI 3 仕様を使用します。OpenAPI 2 仕様を使用する場合は、**New API** ボタンの横にある矢印をクリックし、**OpenAPI 2** を選択します。

注記

OpenAPI 2 仕様に基づいて API を開く場合、API Designer の **Convert to OpenAPI 3** オプションを使用して、API が OpenAPI 3 仕様に準拠するよう変換することができます。

3. API 名を変更するには、以下を実行します。
   a. 名前にカーソルを合わせ、表示される編集アイコン () をクリックします。
   b. 名前を編集します。たとえば、**Task API** を入力します。
   c. チェックマークアイコンをクリックして、名前変更を確認します。
4. 任意で以下を行います。
   - バージョン番号と説明を提供します。
   - 連絡先情報 (名前、メールアドレス、および URL) を追加します。
   - ライセンスを選択します。
   - タグを定義します。
   - サーバーを 1 つ以上定義します。
   - セキュリティースキームを設定します。
   - セキュリティー要件を指定します。
5. API のそれぞれのエンドポイントへの相対パスを定義します。フィールド名はスラッシュ (/) で開始する必要があります。
Task Management API のサンプルでは、2 つのパスを作成します。
   - 複数のタスクのパス: **/todo**
   - ID 別の特定タスクのパス: **/todo/{id}**



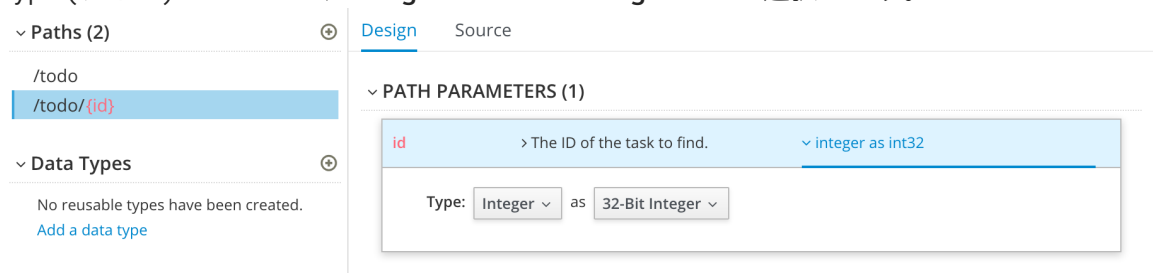
6. 任意のパスパラメーターのタイプを指定します。
id パラメーターのサンプルは、以下のようになります。

   a. **Paths** リストで、**/todo/{id}** をクリックします。
   id パラメーターが **PATH PARAMETERS** セクションに表示されます。

   b. **Create** をクリックします。

   c. description (説明) には、**The ID of the task to find.**と入力します。

   d. type (タイプ) については、**integer** を **32-Bit integer** として選択します。



7. **Data Types** セクションで、API の再利用可能なタイプを定義します。

   a. **Add a data type**をクリックします。

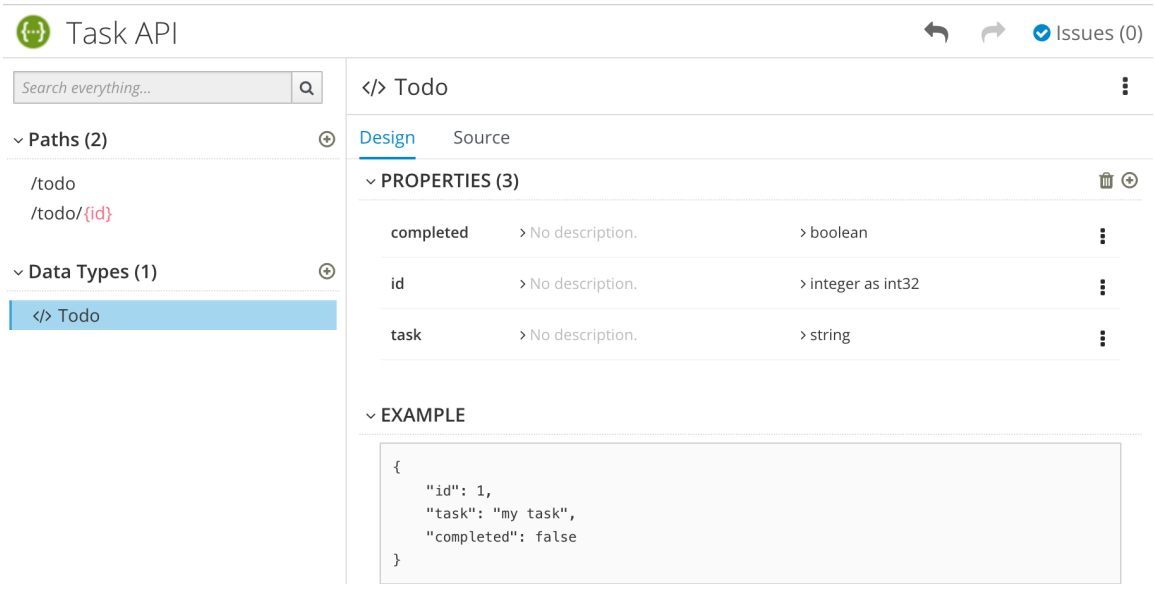
   b. **Add Data Type** ダイアログで名前を入力します。Task Management API のサンプルでは、**Todo** と入力します。

   c. 任意で、API Designer がデータタイプのスキーマの作成に使用するサンプルを指定できます。生成されるスキーマを編集できます。
   Task Management API のサンプルでは、以下の JSON 例で開始します。

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

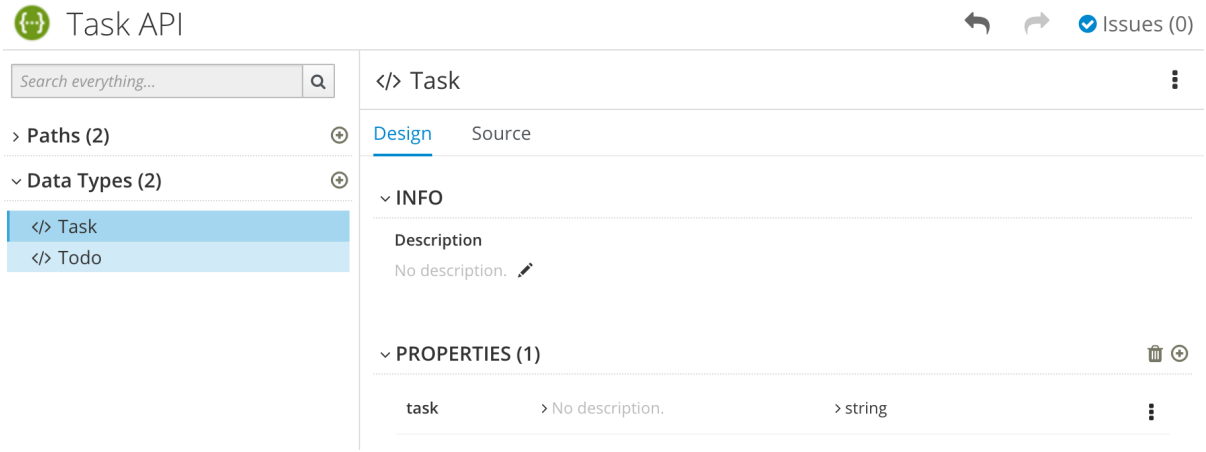
   d. オプションで、該当するデータタイプを指定して REST リソースの作成を選択できます。

   e. **Save** をクリックします。サンプルを指定している場合、API Designer はそのサンプルからスキーマを生成します。



8. オプションで、スキーマのプロパティーを編集し、新しいプロパティーを追加できます。

9. Task Management API のサンプルの場合、タイプ **string** の **task** という名前の 1 つのプロパティーを使用して、 **Task** という名前の別のデータタイプを作成します。



10. それぞれのパスについて、操作 (GET、PUT、POST、DELETE、 OPTIONS、HEAD、または PATCH) を定義します。
Task Management API のサンプルの場合、以下の表で説明されているように操作を定義します。

表3.1 Task Management API 操作

パス	操作	説明	リクエストボディー	応答

パス	操作	説明	リクエストボディー	応答
/todo	POST	新しいタスクを作成します。	メディアタイプ: **application/json** データタイプ: Task	• ステータスコード: 201 説明: Task created 応答ボディー: メディアタイプ: **application/json** データタイプ: Todo
/todo	GET	すべてのタスクを取得します。	**該当なし**	• ステータスコード: 200 説明: **タスクのリスト**
/todo/{id}	GET	ID 別にタスクを取得します。	**該当なし**	• ステータスコード: 200 説明: Task found for ID 応答ボディー: メディアタイプ: **application/json** データタイプ: Task • ステータスコード: 404 説明: No task with provided identifier found.
/todo/{id}	PUT	ID 別にタスクを更新します。	リクエストボディーのタイプ: Task	• ステータスコード: 200 説明: Completed • ステータスコード: 400 説明: Task not updated

パス	操作	説明	リクエストボディー	応答
/todo/{id}	**DELETE**	ID 別にタスクを削除します。	**該当なし**	• ステータスコード: 200 説明: **Task deleted** • ステータスコード: 400 説明: **Task not deleted**

11. API Designer での検証問題の解決 の説明どおりに問題を解決します。

12. Fuse on OpenShift のみの場合には、**Save As** をクリックして API 仕様を保存し、**JSON** または **YAML** 形式を選択します。
JSON または YAML ファイルがローカルダウンロードフォルダーにダウンロードされます。デフォルトのファイル名は、適切なファイル拡張子を含む **openapi-spec** です。

関連情報

- OpenAPI 仕様の詳細は、https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md を参照してください。

3.2. API DESIGNER での検証問題の解決

API の作成および編集時に、API Designer は感嘆符 (!) アイコンと、API Designer タイトルバーの問題のリストを使って解決する必要のある問題を特定します。

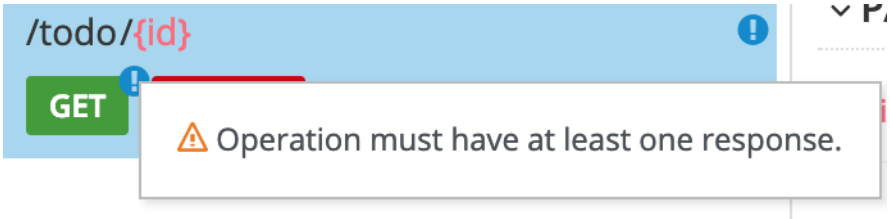
前提条件

- API Designer で API を開いている必要があります。

手順

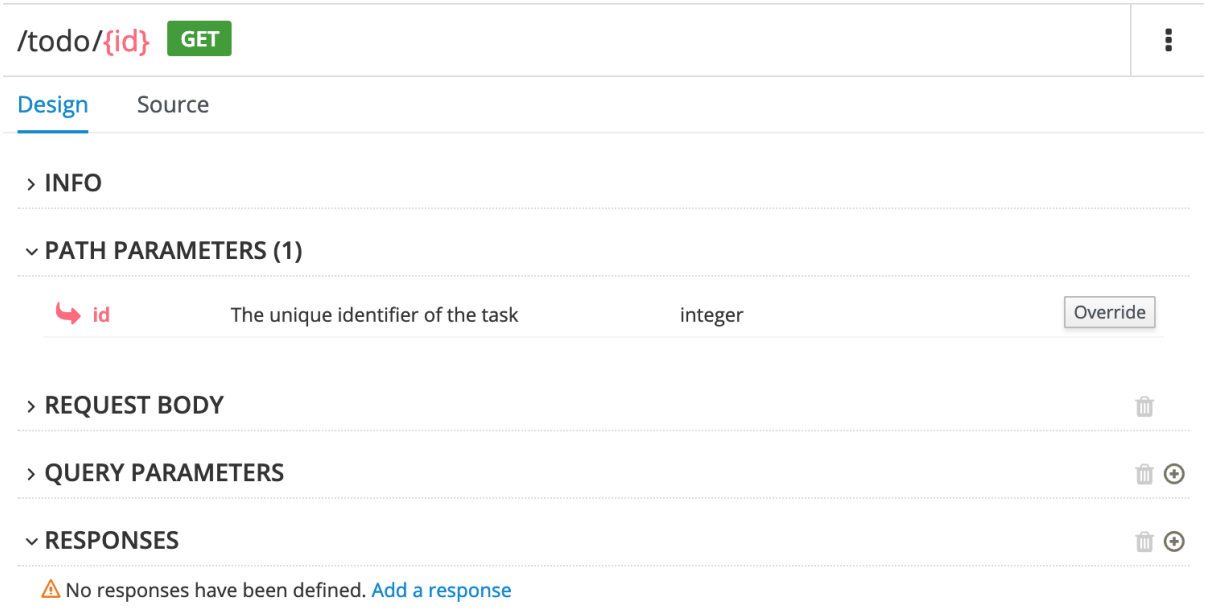
1. 感嘆符 (!) アイコンで示唆される問題を検索します。以下に例を示します。

2. 感嘆符アイコンをクリックして、問題の説明を表示します。以下に例を示します。

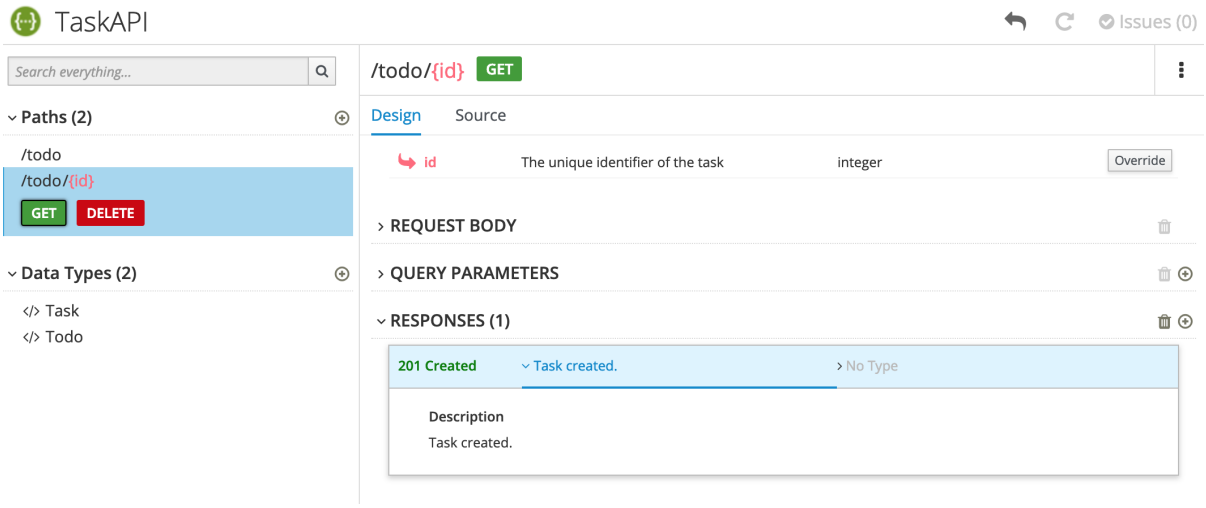


3. 問題の説明で提供される情報に基づいて、問題のある場所に移動し、修正します。

たとえば、Operation must have at least one response(オペレーションには 1 つ以上の応答が必要) の問題を修正するには、**GET** オペレーションをクリックして開き、**Add a response** をクリックします。



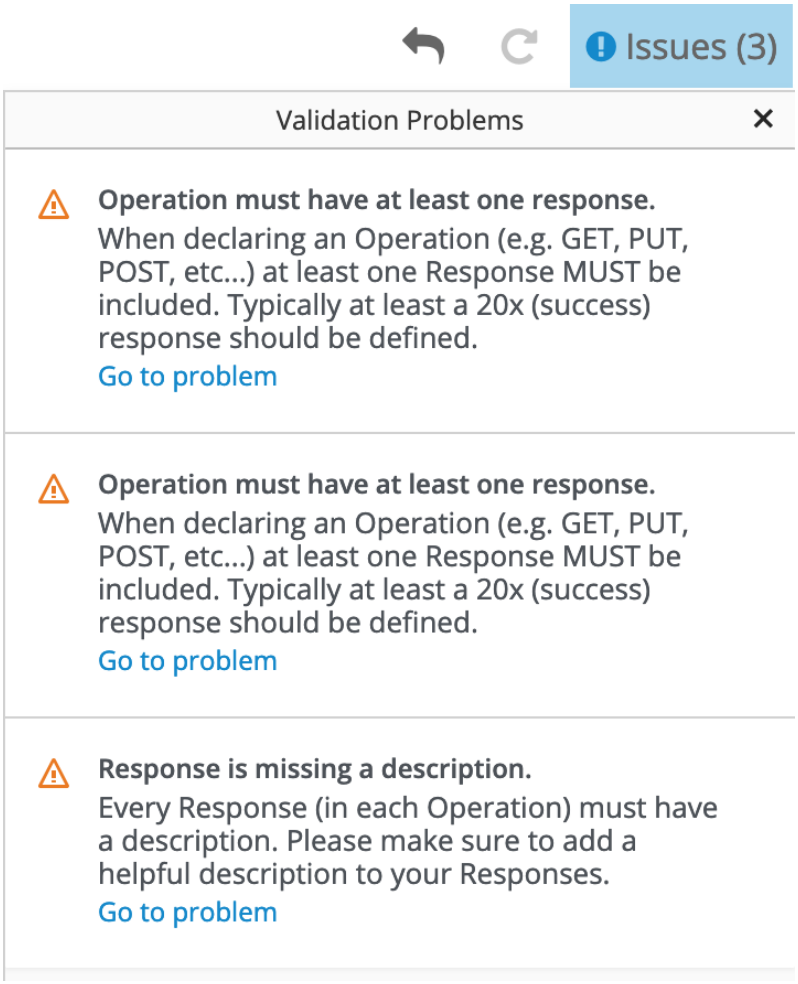
応答の説明の入力後、問題は解決され、感嘆符アイコンは表示されなくなります。



4. すべての問題の概要については、以下のようになります。

   a. 右上にある **Issues** リンクをクリックします。

   b. 特定の問題の **Go to a problem**をクリックし、その問題を解決するために問題のある場所に移動します。


Issues (3)
Validation Problems
Operation must have at least one response.
When declaring an Operation (e.g. GET, PUT, POST, etc...) at least one Response MUST be included. Typically at least a 20x (success) response should be defined.
Go to problem
Operation must have at least one response.
When declaring an Operation (e.g. GET, PUT, POST, etc...) at least one Response MUST be included. Typically at least a 20x (success) response should be defined.
Go to problem
Response is missing a description.
Every Response (in each Operation) must have a description. Please make sure to add a helpful description to your Responses.
Go to problem

第4章 REST API をベースとした FUSE アプリケーションの実装、ビルドおよびデプロイ

Red Hat Fuse API Designer を使用して、REST API 定義に基づいて Camel Fuse プロジェクトを生成できます。Fuse 開発環境では、Camel ルートおよび Rest DSL API を完了できます。最後に、プロジェクトをビルドし、作成されたアプリケーションを Fuse on OpenShift にデプロイすることができます。

前提条件

- OpenAPI 3 (または 2) 仕様に準拠する既存の API 定義が必要です。例: API Designer で作成した **openapi-spec.json** ファイル。
- API Designer がローカル OpenShift クラスターにインストールされ、実行されています。
- API Designer がサービスとして追加された既存の OpenShift プロジェクトがあります。
- Maven と Red Hat Fuse がインストールされています。

以下のトピックでは、REST API をベースとして Fuse アプリケーションの実装、ビルドおよびデプロイを実行する方法について説明します。

- 「API 定義の API Designer へのアップロード」
- 「API Designer からの Fuse Camel プロジェクトの生成」
- 「API Designer で生成される Camel プロジェクトの完了」
- 「REST サービスのビルドとデプロイ」

4.1. API 定義の API DESIGNER へのアップロード

既存の API 定義を API Designer にアップロードできます。

前提条件

- OpenAPI 3 (または 2) 仕様に準拠する既存の API 定義が必要です。例: API Designer で作成した **openapi.json** ファイル。
- API Designer がローカル OpenShift クラスターにインストールされ、実行されています。
- API Designer がアプリケーションとして追加された既存の OpenShift プロジェクトがあります。

手順

1. OpenShift Web コンソールで、API Designer を含むプロジェクトを開きます。
2. API Designer コンソールを開きます。プロジェクトのアプリケーション一覧で、apidesigner の下にある URL をクリックします。例: **https://apidesigner-myproject.192.168.64.38.nip.io**
   API Designer コンソールは、別の Web ブラウザータブまたはウインドウで開きます。
3. **Open API** をクリックします。
   ファイルマネージャーウィンドウが開きます。
4. ファイルマネージャーウィンドウで、以下を実行します。

a. 既存の OpenAPI 定義ファイル (例: **openapi.json**) が含まれるフォルダーに移動します。

b. OpenAPI 定義ファイルを選択し、**Open** をクリックします。
OpenAPI 定義が API Designer コンソールで開きます。

4.2. API DESIGNER からの FUSE CAMEL プロジェクトの生成

API Designer を使用し、API 定義に基づいて Fuse Camel プロジェクトを生成できます。

前提条件

- API Designer がローカル OpenShift クラスターにインストールされ、実行されています。
- API Designer がアプリケーションとして追加された既存の OpenShift プロジェクトがあります。
- API Designer コンソールで API 定義ファイルを作成し、開いています。

手順

API Designer コンソールで、以下を実行します。

1. **Generate** をクリックします。
2. ドロップダウンリストから **Fuse Camel Project** を選択します。

API Designer は **camel-project.zip** ファイルを生成し、ローカルのデフォルトダウンロードフォルダーにダウンロードします。

zip ファイルには、Camel の Rest DSL を使用して API 定義のデフォルトのスケルトン実装を提供し、すべてのリソース操作に対応する Fuse Camel プロジェクトが含まれます。またプロジェクトには、プロジェクトの生成に使用した元の OpenAPI 定義ファイルも含まれます。

4.3. API DESIGNER で生成される CAMEL プロジェクトの完了

API Designer は、Camel の Rest DSL を使用して API 定義のデフォルトのスケルトン実装を提供し、すべてのリソース操作に対応する Fuse プロジェクトを生成します。Fuse 開発環境で、プロジェクトを完了します。

前提条件

- API Designer によって生成された **camel-project.zip** ファイルがある。
- (オプション) Fuse Tooling を使用して Red Hat Developer Studio をインストールしている。

手順

1. API Designer によって生成された **camel-project.zip** ファイルを一時フォルダーに解凍します。
2. Red Hat Developer Studio を開きます。
3. Developer Studio で **File → Import** を選択します。
4. **Import** ダイアログで、**Maven → Existing Maven Projects** を選択します。

5. エディタービューでプロジェクトの **camel-context.xml** ファイルを開きます。

6. **REST** タブをクリックして Rest DSL コンポーネントを編集します。
REST サービスの定義についての詳細は、Apache Camel Development Guide の Defining REST services セクションを参照してください。

   Swagger サポートで JAX-RS エンドポイントを拡張する方法については、Apache CXF Development Guide を参照してください。

   Fuse Tooling REST エディターの使用方法は、Tooling User Guide の Viewing and editing Rest DSL components セクションを参照してください。

7. Design タブで Camel ルートを編集します。
Camel ルートの編集に関する詳細は、Tooling User Guide の Editing a routing context in the route editor セクションを参照してください。

4.4. REST サービスのビルドとデプロイ

Fuse プロジェクトの完了後に、プロジェクトを OpenShift でビルドし、デプロイすることができます。

前提条件

- REST サービスを定義する完全なエラーのない Fuse プロジェクトがあります。
- Java 8 JDK (以降) および Maven 3.3.x (以降) をインストールしています。

手順

Minishift または Red Hat Container Development Kit などの単一ノードの OpenShift クラスターが インストールされ、実行中である 場合、プロジェクトをそこにデプロイできます。

このプロジェクトを実行中の単一ノードの OpenShift クラスターにデプロイするには、以下を実行します。

1. OpenShift クラスターにログインします。

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

2. プロジェクトの新規 OpenShift プロジェクトを作成します。たとえば、以下のコマンドは **test-deploy** という名前の新規プロジェクトを作成します。

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

3. ディレクトリーを Fuse Camel プロジェクトが含まれるフォルダーに変更します (例: **myworkspace/camel-project**)。

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

4. プロジェクトを OpenShift クラスターにビルドし、デプロイします。

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

5. ブラウザーで OpenShift コンソールを開き、プロジェクト (例: **test-deploy**) に移動します。**camel-project** アプリケーションの Pod が起動していることを確認できるまで待機します。

6. プロジェクトの **Overview** ページで、**camel-project** アプリケーションの URL を特定します。URL は **http://camel-project-MY_PROJECT_NAME.OPENSHIFT_IP_ADDR.nip.io** という形式を使用します。

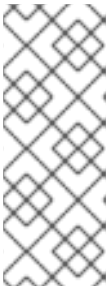
7. URL をクリックしてサービスにアクセスします。

第5章 3SCALE の検出についての API サービスの準備

Red Hat 3scale API Management を使用すると、パブリックインターネット上の API サービスへのアクセスを制御できます。3scale の機能には、サービスレベルアグリーメント (SLA) の実施、API バージョンの管理、セキュリティーおよび認証サービスの提供などが含まれます。Fuse は 3scale の **サービス検出**機能をサポートします。これにより、3scale 管理ポータル UI から Fuse サービスを簡単に検出できます。サービス検出を使用して、同じ OpenShift クラスターで実行されている Fuse アプリケーションをスキャンし、関連付けられた API 定義を 3scale に自動的にインポートできます。

前提条件

- API サービスを提供する Fuse アプリケーションが OpenShift でデプロイされ、実行されています。
- Fuse アプリケーションには、3scale によって検出されるように必須のアノテーションが付けられています。



注記

API Designer によって生成される Fuse プロジェクトは、必要なアノテーションを自動的に提供するように事前に設定されています。

API Designer によって生成されない Fuse プロジェクトの場合、API Designer によって生成されない Fuse プロジェクトのアノテーションの追加 の説明どおりに、プロジェクトを設定する必要があります。

- 3scale API Management システムは、検出される API サービスと **同じ** OpenShift クラスターにデプロイされます。

3scale で API サービスを検出する手順についての詳細は、Red Hat 3scale API Management 管理ポータルガイド のサービスディスカバリーを参照してください。

関連情報

- Red Hat 3scale API Management の製品ページ
- Red Hat 3scale API Management ドキュメント

5.1. API DESIGNER によって生成されない FUSE プロジェクトのアノテーションの追加

3scale が API サービスを検出できるようにするには、API サービスを提供する Fuse アプリケーションに、検出を可能にする Kubernetes サービスアノテーションが含まれている必要があります。これらのアノテーションは、OpenShift Maven Plugin の一部である Service Discovery Enricher により提供されます。

Apache Camel Rest DSL プロジェクトの場合、OpenShift Maven Plugin はデフォルトで Service Discovery Enricher を実行します。

API Designer によって生成される Fuse プロジェクトは、必要なアノテーションを自動的に提供するよう事前に設定されます。

手順

API Designer によって生成されていない Fuse Rest DSL プロジェクトの場合、プロジェクトを以下のように設定します。

1. 以下の例のように、Fuse プロジェクトの **pom.xml** ファイルを編集し、**openshift-maven-plugin** 依存関係を組み込みます。

```
<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>
```

   OpenShift Maven Plugin は、特定のプロジェクトレベルの条件が満たされる場合は Service Discovery Enricher を実行します (例: プロジェクトは Camel Rest DSL プロジェクトである必要があります)。API サービスアノテーション値のカスタマイズ の説明にあるように、 Enricher の動作をカスタマイズする必要がない場合は、Service Discovery Enricher を **pom.xml** ファイルで依存関係として指定する必要はありません。

2. Fuse Rest DSL プロジェクトの **camel-context.xml** ファイルで、以下の属性を **restConfiguration** 要素に指定します。

   - **scheme**: サービスがホストされる URL のスキームの部分です。http または https を指定できます。
   - **contextPath**: API サービスがホストされる URL のパスの部分です。
   - **apiContextPath**: API サービスの記述ドキュメントがホストされる場所へのパスです。ドキュメントがセルフホストされている場合は相対パスを指定でき、ドキュメントが外部でホストされる場合は完全な URL を指定できます。
     サンプル **camel-context.xml** ファイルからの以下の抜粋には、アノテーションの属性値が **restConfiguration** 要素に表示されています。

```
<?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"/>
...
```

Enricher は、これらの **restConfiguration** 要素の属性値で指定される情報を使用し、**discovery.3scale.net/scheme**、**discovery.3scale.net/path**、および

discovery.3scale.net/description-path アノテーションの値を作成します。これにより、Red Hat 3scale API Management管理ポータルガイド のサービスディスカバリーの説明にあるように、プロジェクトのデプロイされた OpenShift サービスを 3scale で検出可能にすることができます。

Enricher は以下のラベルとアノテーションを追加して、サービスが 3scale で検出できるようにします。

- **discovery.3scale.net** ラベル: デフォルトで、Enricher はこの値を true に設定します。3scale はセレクター定義を実行して検出を必要とするすべてのサービスを検索する際にこのラベルを使用します。
- また、以下のアノテーションが含まれている必要があります。
  - **discovery.3scale.net/discovery-version**: (オプション)3scale 検出プロセスのバージョンです。Enricher はデフォルトでこの値を v1 に設定します。
  - **discovery.3scale.net/scheme**: サービスがホストされる URL のスキーム部分です。Enricher は、**restConfiguration** 要素の **scheme** 属性で上書きしない限り、デフォルトの http を使用します。他の使用できる値は https です。
  - **discovery.3scale.net/path**: サービスがホストされる URL のパスの部分です。このアノテーションは、パスがルート/にある場合に省略されます。Enricher は、この値を **restConfiguration** 要素の **path** 属性から取得します。
  - **discovery.3scale.net/port**: サービスのポートです。Enricher はこの値を Kubernetes サービス定義から取得します。これには、公開するサービスのポート番号が含まれます。Kubernetes サービス定義が複数のサービスを公開する場合、Enricher は一覧表示される最初のポートを使用します。
  - **discovery.3scale.net/description-path**: (オプション) OpenAPI サービス記述ドキュメントへのパスです。Enricher は、この値を **restConfiguration** 要素の **contextPath** 属性から取得します。

API サービスアノテーション値のカスタマイズ の説明どおりに、Service Discovery Enricher の動作をカスタマイズできます。

5.2. API サービスアノテーション値のカスタマイズ

Maven Fabric8 プラグインは、デフォルトで Fabric8 Service Discovery Enricher を実行します。Enricher は、Red Hat 3scale API Management の **管理ポータルガイド** の サービスディスカバリーの使用 で説明されているように、アノテーションを Fuse Rest DSL プロジェクトの API サービスに追加して API サービスを 3scale で検出できるようにします。

Enricher は一部のアノテーションのデフォルト値を使用し、プロジェクトの **camel-context.xml** ファイルから他のアノテーションの値を取得します。

デフォルト値や **camel-context.xml** ファイルで定義される値は、Fuse プロジェクトの **pom.xml** ファイルや **service.yml** ファイルに値を定義して上書きできます。(両方のファイルで値を定義する場合、Enricher は **service.yml** ファイルから値を使用します。) Fabric8 Service Discovery Enricher に指定できる要素の説明については、Fabric8 Service Discovery Enricher elements を参照してください。

手順

Fuse プロジェクトの **pom.xml** ファイルにアノテーションの値を指定するには、以下を実行します。

1. 任意のエディターで Fuse プロジェクトの **pom.xml** ファイルを開きます。

2. 以下の例のように、**openshift-maven-plugin** の依存関係を見つけます。

```
<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. 以下の例のように、Fabric8 Service Discovery Enricher を openshift-maven プラグインに依存関係として追加します。

```
<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. 変更を保存します。

または、**src/main/fabric8/service.yml** フラグメントを使用して、以下の例のようにアノテーション値を上書きできます。

```
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 ENRICHER 要素

以下の表は、デフォルト値や **camel-context.xml** ファイルで定義された値を上書きする必要がある場合に Fabric8 Service Discovery Enricher に指定できる要素について説明しています。

これらの値は Fuse Rest DSL プロジェクトの **pom.xml** ファイルや **src/main/fabric8/service.yml** ファイルで定義できます。(両方のファイルで値を定義する場合、Enricher は **service.yml** ファイルから値を使用します。) 例は、Customizing the API service annotation values を参照してください。

表5.1 Fabric8 Service Discovery Enricher 要素

要素	説明	デフォルト
springDir	**camel-context.xml** ファイルが含まれる spring 設定ディレクトリーへのパスです。	Camel Rest DSL プロジェクトを認識するために使用される **/src/main/resources/spring** パスです。
scheme	サービスがホストされる URL のスキーム部分です。http または https を指定できます。	http
path	API サービスがホストされる URL のパスの部分です。	
port	API サービスがホストされる URL のポートの部分です。	80
descriptionPath	API サービス記述ドキュメントがホストされる場所へのパスです。ドキュメントがセルフホストされている場合は相対パスを指定でき、ドキュメントが外部でホストされる場合は完全な URL を指定できます。	
discoveryVersion	3scale 検出の実装のバージョンです。	v1

要素	説明	デフォルト
discoverable	**discovery.3scale.net** ラベルを **true** または **false** のいずれかに設定する要素です。 **true** に設定すると、3scale はこのサービスの検出を試行します。 **false** に設定すると、3scale はこのサービスの検出を試行しません。 この要素をスイッチとして使用し、false に設定することで 3scale 検出の統合を一時的にオフにできます。	値を指定しない場合、Enricher はサービスを検出可能にできるかどうかについて自動検出を試行します。Enricher がサービスが検出不可能であると判別すると、3scale はこのサービスの検出を試行しません。