Red Hat Fuse 7.12

Tooling ユーザーガイド

CodeReady Studio の FuseTooling を使用して、Fuse アプリケーションを開発およびデプロイします

Last Updated: 2023-11-24

Red Hat Fuse 7.12 Tooling ユーザーガイド

CodeReady Studio の FuseTooling を使用して、Fuse アプリケーションを開発およびデプロイします

法律上の通知

概要

このガイドでは、Red Hat Fuse Tooling の使用方法について説明します。Fuse Tooling は、統合アプリケーションの設計、開発、テスト、およびデバッグを実行する際の生産性向上を目的として設計された開発者ツールを提供します。

目次

はじめに 6

多様性を受け入れるオープンソースの強化 7

パート I. アプリケーションの開発 8

第1章 新しい FUSE INTEGRATION プロジェクトの作成 9
概要 9
作業を開始する前に 9
プロジェクト名とワークスペースの指定 9
プロジェクトデプロイメント環境の設定 10
新しいターゲットランタイムの作成 (オプション) 12
プロジェクトテンプレートの選択 15
MAVEN 依存関係エラーの解決 19

第2章 ルートエディターを使用したルーティングコンテキストの編集 21
2.1. ルートへのパターンの追加 21
2.2. パターンの設定 22
2.3. ルートからパターンを削除 23
2.4. ルーティングコンテキストへのルートの追加 23
2.5. ルートの削除 25
2.6. グローバルエンドポイント、データ形式、BEAN の追加 26
2.7. ルートエディターの設定 43

第3章 REST DSL コンポーネントの表示と編集 47
3.1. REST DSL コンポーネントのグラフィック表現の表示 47
3.2. グラフィカルビューでの REST DSL コンポーネントの編集 48
3.3. REST DSL ソースコードの表示と編集 49

第4章 新しい APACHE CAMEL JUNIT テストケースの作成 51
概要 51
前提条件 51
既存の JUNIT テストケースの削除 51
SRC/TEST/JAVA フォルダーの作成およびビルドパスへの追加 51
JUNIT テストケースの作成 52

第5章 RED HAT FUSETOOLING 内でのルートの実行 55
5.1. ローカル CAMEL コンテキストとしてルートを実行 55
5.2. MAVEN を使用したルートの実行 55
5.3. ランタイムプロファイルの操作 56

第6章 FUSE ON OPENSHIFT の使用 62
6.1. RED HAT CONTAINER DEVELOPMENT KIT サーバーの追加 63
6.2. CONTAINER DEVELOPMENT ENVIRONMENT (CDE) および仮想 OPENSHIFT サーバーの起動 65
6.3. 新しい OPENSHIFT プロジェクトの作成 66
6.4. 新規 FUSE INTEGRATION プロジェクトの作成 68
6.5. FUSE INTEGRATION プロジェクトの OPENSHIFT へのデプロイ 73
6.6. OPENSHIFT WEB コンソールへのアクセス 79

第7章 RED HAT FUSE SAP TOOL SUITE の使用 81
7.1. RED HAT FUSE SAP TOOL SUITE のインストール 81
SAP 宛先接続の作成とテスト 81
7.2. SAP サーバー接続の作成とテスト 84
7.3. 宛先接続およびサーバー接続の削除 85

7.4. 新規 SAP エンドポイントの作成 86

第8章 データ変換のスタートガイド 88
8.1. データ変換サンプルプロジェクトの作成 88
8.2. CAMEL ルートへのデータ変換ノードの追加 89
8.3. ソースデータ項目をターゲットデータ項目にマッピング 95
8.4. 変換テストファイルの作成と JUNIT テストの実行 98
8.5. 定数変数のデータ項目へのマッピング 99
8.6. 式のデータ項目へのマッピング 100
8.7. マップされたデータ項目へのカスタム変換の追加 104
8.8. 単純なデータ項目をコレクションのデータ項目にマッピング 108
8.9. マップされたデータ項目へのビルトイン関数の追加 110
8.10. データ変換を使用して FUSE INTEGRATION プロジェクトを RED HAT FUSE サーバーに公開 113

第9章 FUSE ONLINE インテグレーション用エクステンションの開発 115
9.1. タスクの概要 115
9.2. 前提条件 116
9.3. カスタムコネクターの作成 116
9.4. カスタムステップの作成 119
9.5. FUSE ONLINE エクステンション JAR ファイルのビルド 120
9.6. JAR ファイルを FUSE ONLINE ユーザーに提供 122

第10章 新規 CAMEL XML ファイルの作成 123
概要 123
手順 123

第11章 CAMEL バージョンの変更 125

第12章 既存 MAVEN プロジェクトのインポート 126
概要 126
手順 126

パート II. ルーティングコンテキストのデバッグ 127

第13章 ブレークポイントの設定 128
概要 128
無条件ブレークポイントの設定 128
条件付きブレークポイントの設定 128
ブレークポイントの無効化 129
ブレークポイントの削除 130
関連トピック 130

第14章 CAMEL デバッガーの実行 131
手順 131
ルーティングコンテキストを介したメッセージエクスチェンジの進捗監視 132
関連トピック 133

第15章 CAMEL デバッガーの停止 134
概要 134
CAMEL デバッガーを閉じる 134
関連トピック 134

第16章 変数値の変更 135
概要 135
手順 135
関連トピック 136

第17章 ウォッチリストへの変数の追加 **137**
概要 137
手順 137
関連トピック 138

第18章 実行中のコンテキストでのブレークポイントの無効化 **139**
概要 139
ブレークポイントビューでのブレークポイントの無効化と有効化 139

パート III. アプリケーションの監視とテスト **141**

第19章 JMX ナビゲーター **142**
19.1. JMX でのプロセスの表示 143
19.2. JMX サーバーの追加 143

第20章 コンポーネントの JMX 統計情報の表示 **145**
概要 145
手順 145

第21章 メッセージの参照 **147**
概要 147
手順 147
関連トピック 148

第22章 ルートのトレース **149**
22.1. ルートトレース用テストメッセージの作成 149
22.2. ルートトレースの有効化 150
22.3. ルーティングコンテキストを介したメッセージのトレース 151
22.4. ルートトレースの無効化 152

第23章 JMS 宛先の管理 **153**
23.1. JMS 宛先の追加 153
23.2. JMS 宛先の削除 153

第24章 ルーティングエンドポイントの管理 **155**
24.1. ルーティングエンドポイントの追加 155
24.2. ルーティングエンドポイントの削除 155

第25章 実行中のルートの編集 **157**
概要 157
実行中のルートの変更および結果の評価 157
ルート編集セッションの終了 159
関連トピック 159

第26章 ルーティングコンテキストの管理 **160**
26.1. ルーティングコンテキストの一時停止操作 160
26.2. ルーティングコンテキストの再開操作 160

パート IV. コンテナーへのアプリケーションの公開 **162**

第27章 サーバーの管理 **163**
27.1. サーバーの追加 163
27.2. サーバーの起動 167
27.3. 稼働中のサーバーへの接続 168
27.4. サーバーからの接続解除 170
27.5. サーバーの停止 170
27.6. サーバーの削除 171

第28章 FUSE INTEGRATION プロジェクトのサーバーへの公開 **172**
概要 172
リソース変更時に FUSE プロジェクトを自動公開 172
FUSE プロジェクトの手動公開 176
プロジェクトのサーバーへの公開を確認 177

付録A FUSE INTEGRATION パースペクティブ **179**

付録B DEBUG パースペクティブ **185**

はじめに

Red Hat Fuse Tooling は、Red Hat CodeReady Studio 内で統合アプリケーションを開発するプロセスを簡素化および合理化する Eclipse ベースの IDE です。Fuse Tooling は、以下で動作するように特別に設計された一連の開発者ツールを提供します。

- Red Hat Fuse
- Red Hat JBoss EAP
- Apache Camel
- Apache CXF
- Apache Karaf
- Spring Boot

このガイドでは、FuseTooling を使用して以下を行う方法について説明します。

- Maven 依存関係を含むアプリケーションのプロジェクトを作成する
- エンタープライズ統合パターンを接続および設定してルートを構築する
- エンドポイントとルートを参照する
- 実行中のルートにメッセージをドラッグアンドドロップする
- JMX を介してランタイムプロセスを参照および可視化する
- ローカルで実行されている Camel コンテキストとルートをデバッグする
- 以下の方法でアプリケーションをテストする
  - Apache Camel ルートで JUnit テストケースを作成および使用する
  - JMX を使用して実行中のコンポーネントを分析する
  - Apache Camel ルートを介してメッセージをトレースする
- アプリケーションをデプロイする

新規ユーザー用に、Tooling チュートリアル は、サンプル Camel アプリケーションを作成、デバッグ、テスト、およびデプロイするためのステップバイステップの手順を提供します。

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、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

パート I. アプリケーションの開発

第1章 新しい FUSE INTEGRATION プロジェクトの作成

概要

新しい Fuse Integration プロジェクトの作成には、次の主な手順が含まれます。

- プロジェクト名とワークスペースの指定
- プロジェクトデプロイメント環境の設定
- プロジェクトテンプレートの選択
- 必要な場合: Maven 依存関係エラーの解決

ツールは、プロジェクトを設定した後、必要なすべての Maven 依存関係をダウンロードし、プロジェクトの実行と公開に必要な POM ファイルを作成します。

作業を開始する前に

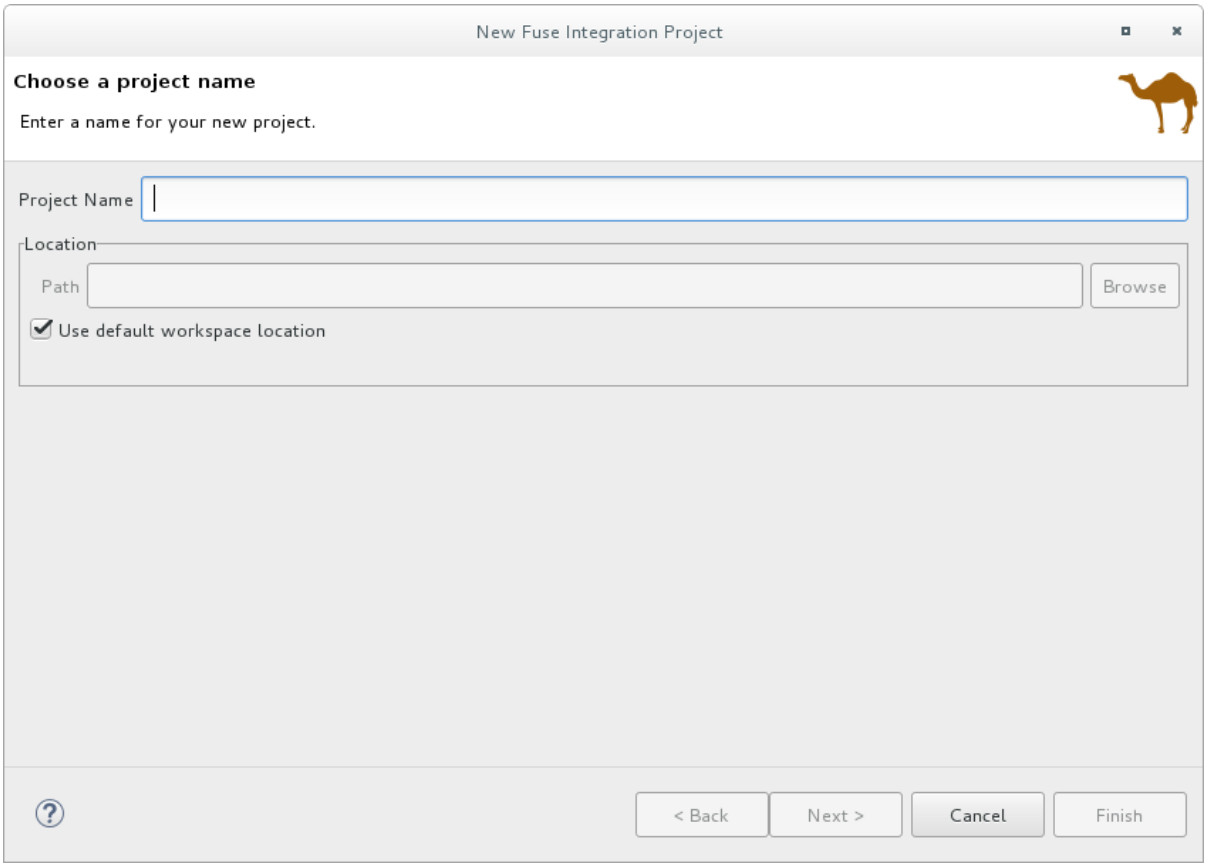
新しい Fuse Integration プロジェクトを作成する前に、次の情報が必要です。

- ターゲットランタイム環境: OpenShift 上の Fuse またはスタンドアロンの Fuse (Spring Boot、Karaf 上の Fuse、または EAP 上の Fuse)
- Camel バージョン (ツールでデフォルト以外が使用される場合)

プロジェクト名とワークスペースの指定

新しい Fuse Integration プロジェクトを作成するには、次の手順を実行します。

1. **New → Project → Red Hat Fuse → Fuse Integration Project** を選択して、**New Fuse Integration Project** ウィザードを開きます。
ウィザードが開き、**Location** ペインの **Use default workspace location** オプションが選択されています。



2. **Project Name** に、新規プロジェクトの名前を入力します (例: **MySampleProject**)。
3. プロジェクトのデータを保存するワークスペースの場所を指定します。
   - デフォルトのワークスペースを使用するには、**Use default workspace location** オプションを有効のままにします。
   - 別の場所を使用するには、**Use default workspace location** オプションをオフにし、**Path** フィールドで場所を指定します。
     

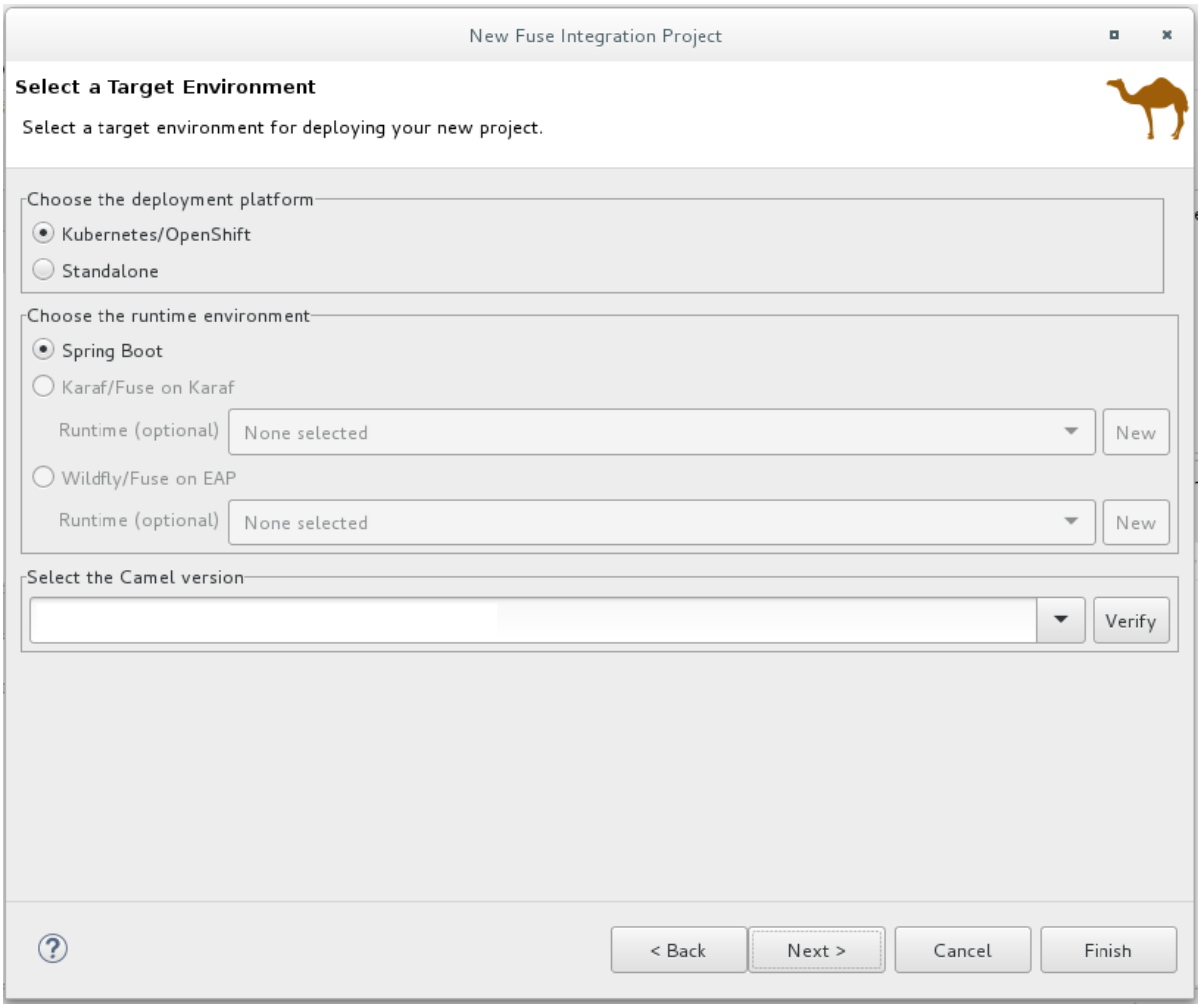
     をクリックし、簡単に別のワークスペースを検索して選択します。
4. **Next** をクリックして **Select a Target Environment** ページを開きます。

プロジェクトデプロイメント環境の設定

新規プロジェクトを作成する際に、プロジェクトのターゲットデプロイメント環境を指定して、プロジェクトが実行時に必要なリソースを確保します。デプロイメントプラットフォームと Camel バージョンを選択する必要があります。オプションで、ランタイム設定を指定できます。

Select a Target Environment ページを開いた状態で、次の操作を行います。

1. プロジェクトを、**Kubernetes/OpenShift** と **Standalone** のどちらのプラットフォーム上にデプロイするか選択します。



デプロイメントプラットフォームに **Kubernetes/OpenShift** を選択すると、**Sprint Boot** ランタイムが自動的に選択されます。その場合、ステップ 3 に進みます。

2. デプロイメントプラットフォームに **Standalone** を選択した場合、以下を実行します。

   a. ターゲットランタイム環境を選択します。

      - **Spring Boot**
      - **Karaf/Karaf 上の Fuse**
      - **Wildfly/EAP 上の Fuse**

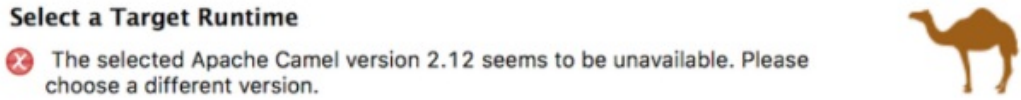
   b. Karaf および EAP スタンドアロンランタイム環境の場合、ランタイム設定として次のいずれかのオプションを選択します。

      - **None selected** オプションを選択します (後でランタイム設定を定義できます)。
      - ドロップダウンメニューから既存のランタイム設定を選択します。
      - 「新しいターゲットランタイムの作成 (オプション)」 の説明に従って、新規ランタイム設定を作成します。

3. **Select the Camel version for your new project**ペインで、ランタイムに関連付けられているデフォルトの Camel バージョンを選択するか、次の方法でデフォルトを変更します。

   - ドロップダウンリストから Camel バージョンを選択します。Fuse Tooling は、リストされている製品化バージョンをサポートしています。

- 製品化されていない (サポートされていない) バージョンを試す場合は、別の Camel バージョンを入力します。
  Verify ボタンをクリックすると、ツールが指定したバージョンにアクセスできるか確認できます。そうでない場合は、次の例のような通知が **Select a Target Runtime** ページのヘッダーに表示されます。



注記

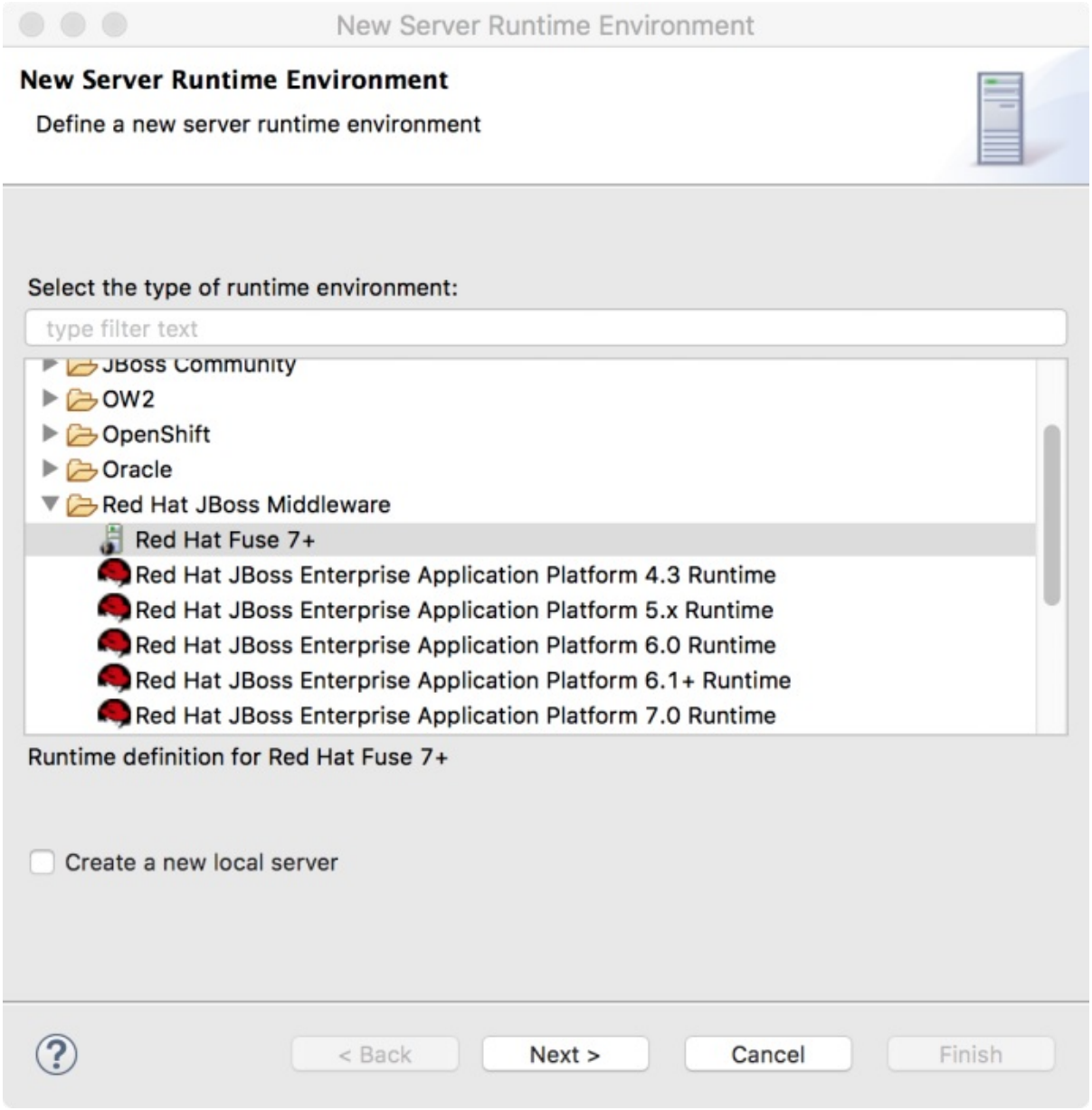
プロジェクトを作成、設定、および保存した後、Camel のバージョンを変更できます。11章*Camel バージョンの変更*を参照してください。

4. 新しい Fuse Integration プロジェクトのベースとなるランタイム環境と Camel バージョンを選択したら、**Next** をクリックしてウィザードの **Advanced Project Setup** ページを開き、「プロジェクトテンプレートの選択」 の手順に従います。

新しいターゲットランタイムの作成 (オプション)

Karaf および EAP スタンドアロンランタイム環境の場合、オプションで、**New Fuse Integration Project** ウィザードから新しいランタイム設定を作成できます。

1. ウィザードの **Select a Target Runtime** ページで、**New** をクリックして **New server runtime environment** ページを開きます。

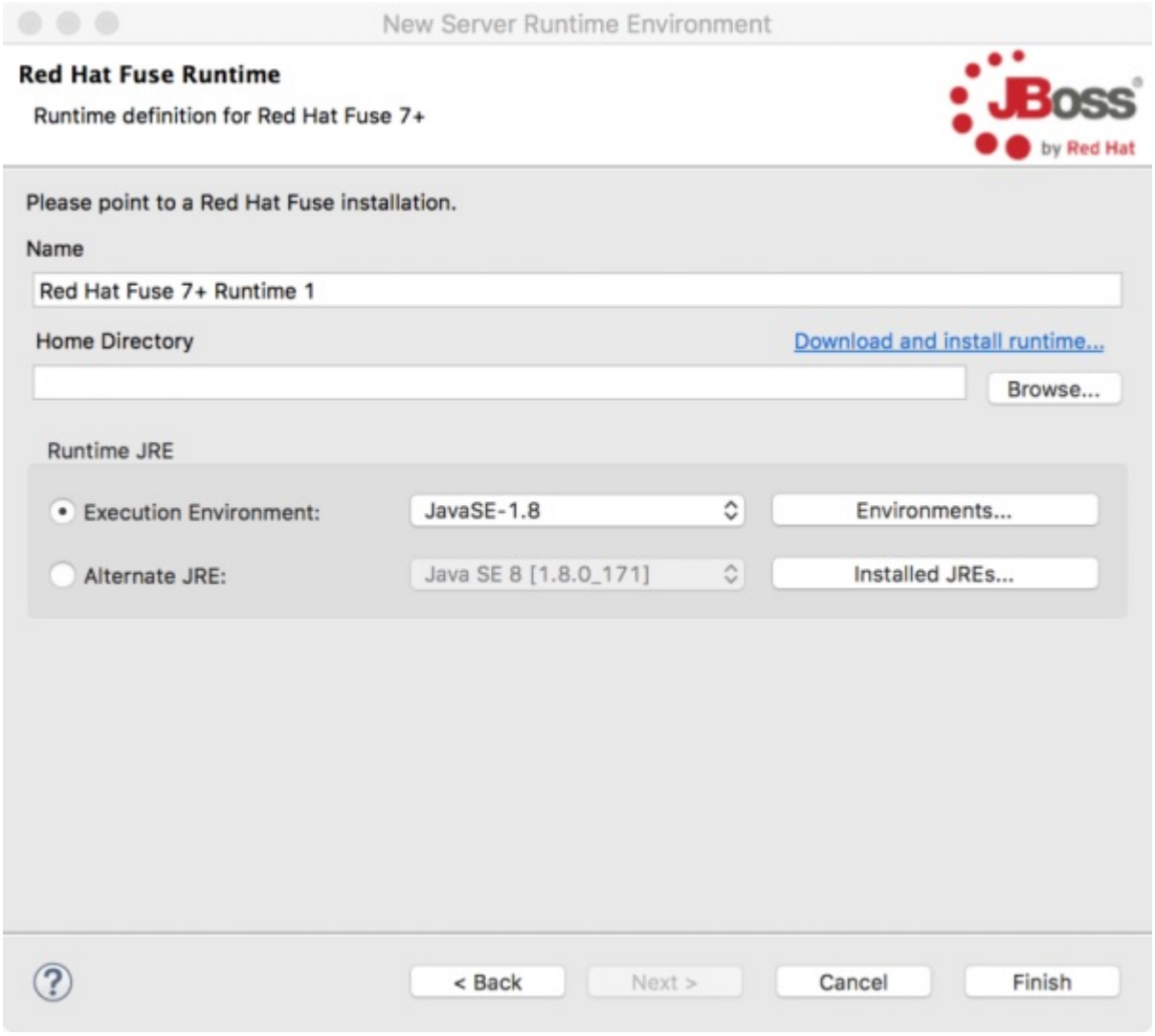


2. **Red Hat JBoss Middleware** フォルダーを展開し、Red Hat Fuse のランタイム環境を選択します。
Create a new local serverオプションをオフのままにします。プロジェクトを公開する準備ができたら、後でローカルサーバーを作成できます (「サーバーの追加」 を参照)。

注記

Create a new local serverオプションをオンにすると、**New Fuse Integration Project** ウィザードに Fuse サーバーランタイムを定義および設定するための追加手順の説明が表示されます (「サーバーの追加」 の説明を参照)。次に、プロジェクトをビルドするときに、**Fuse Integration** パースペクティブの **Servers** ビューにもサーバーランタイムが追加されます。

3. **Next** をクリックして、サーバーの **New Server Runtime Environment**ページを開きます。



4. サーバーランタイムの **Name**、**Home Directory**、**Execution Environment** を指定します。

- **Name** – デフォルトを使用するか、ランタイム環境の新しい名前を入力します。
- **Home Directory** – **Browse** ボタンをクリックして、サーバーランタイムのインストールディレクトリーを見つけて選択します。

注記

サーバーがマシンにインストールされていない場合は、ここで **Download and install runtime** リンクをクリックし、サイトのダウンロード手順に従ってサーバーをインストールできます。サイトによっては、ダウンロードプロセスを続行する前に、有効なクレデンシャルの提供が必要なこともあります。

- **Runtime JRE: Execution Environment**– デフォルトを使用するか、ドロップダウンリストから別の JavaSE バージョンを選択します。
目的のバージョンがリストに表示されない場合は、**Environments** ボタンをクリックして、そのリストからバージョンを選択します。選択した JRE バージョンがマシンにインストールされている必要があります。

注記

Fuse 7.x には JRE バージョン 1.8 が必要です。

- **Runtime JRE: Alternate JRE**- プロジェクトで異なるバージョンの Java が必要な場合は、このオプションを使用できます。

5. **Finish** をクリックして、**New Fuse Integration Project** ウィザードの **Select a Target Runtme** ページに戻ります。
   新しく設定されたターゲットランタイムが **Target Runtime** ペインのドロップダウンメニューに表示され、ランタイムでサポートされている Camel バージョンが **Camel Version** ペインにグレー表示されます。

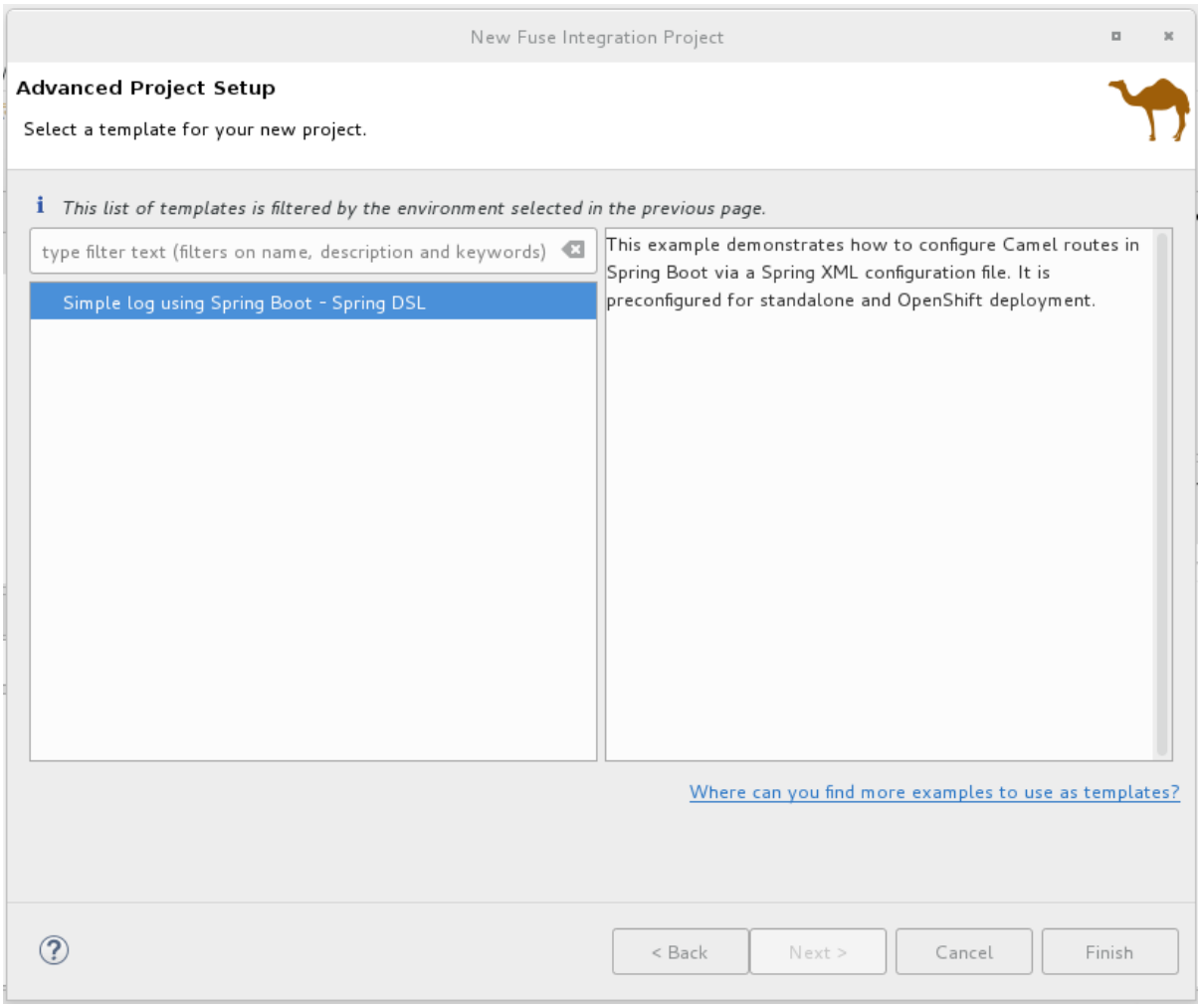
   Fuse Integration プロジェクトを作成した後は、Camel のバージョンを変更できます。11章*Camel バージョンの変更*を参照してください。

6. **Next** をクリックして、「プロジェクトテンプレートの選択」 で説明されているようにプロジェクトのテンプレートを指定します。

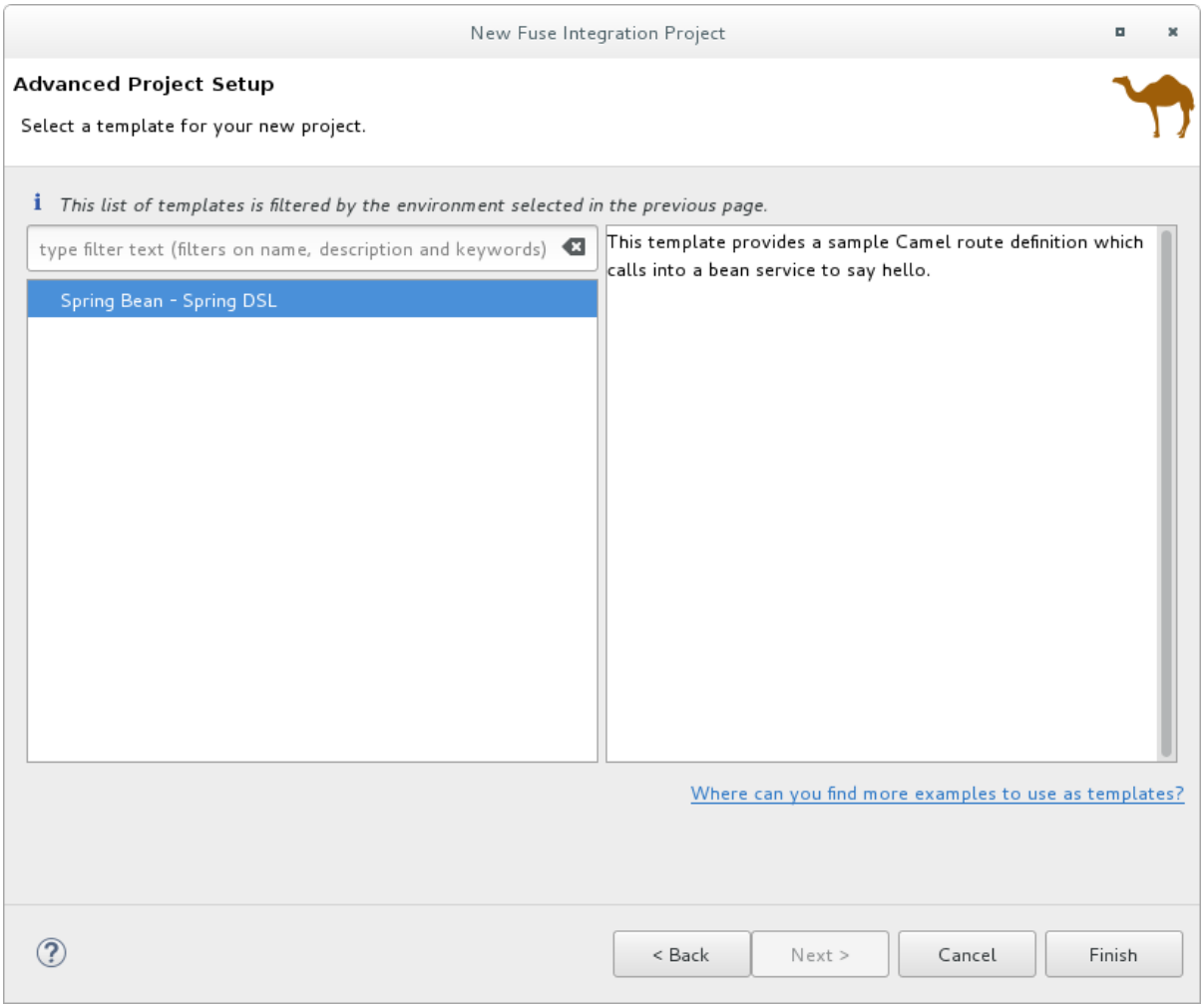
プロジェクトテンプレートの選択

Advanced Project Setup ページには、新しいプロジェクトの開始点として使用できるテンプレートのリストが表示されます。テンプレートは一般的なユースケースに基づいており、サンプルコードとデータが提供されているため、すぐに使い始めることができます。使用可能なテンプレートのリストは、前のページで選択したランタイム環境によって異なります。テンプレートを選択すると、右側のペインにその説明が表示されます。

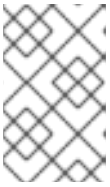
- **Fuse on OpenShift** には、Spring XML 設定ファイルを使用して Spring Boot で Camel ルートを設定する方法を示すテンプレートが 1 つあります。このテンプレートは Fuse Integration プロジェクトを作成します。また、**2.18.1.redhat-000012** より新しい Camel バージョンが必要です。
  このテンプレートは、OpenShift サーバーで実行されるプロジェクトを作成し、Spring DSL のみサポートします。このテンプレートの使用の詳細については、6章*Fuse on OpenShift の使用*を参照してください。



- **Wildfly** または **Fuse on EAP** の場合、こんにちはと言うために Bean サービスを呼び出すサンプル Camel ルートを提供するテンプレートが 1 つあります。このテンプレートは、Red Hat EAP サーバーで実行されるプロジェクトを作成し、Spring DSL のみサポートします。



- **Karaf** または **Fuse on Karaf** の場合、テンプレートを選択できます。サポートされている 3 つのドメイン固有言語 (DSL) のいずれかに基づいてスケルトン Camel コンテキストのルーティングファイルを作成する空のプロジェクトを作成するか、事前定義されたテンプレート (それぞれ一般的なユースケースに基づく) を使用できます。個々のテンプレートがすべての DSL オプションをサポートしているとは限りません。

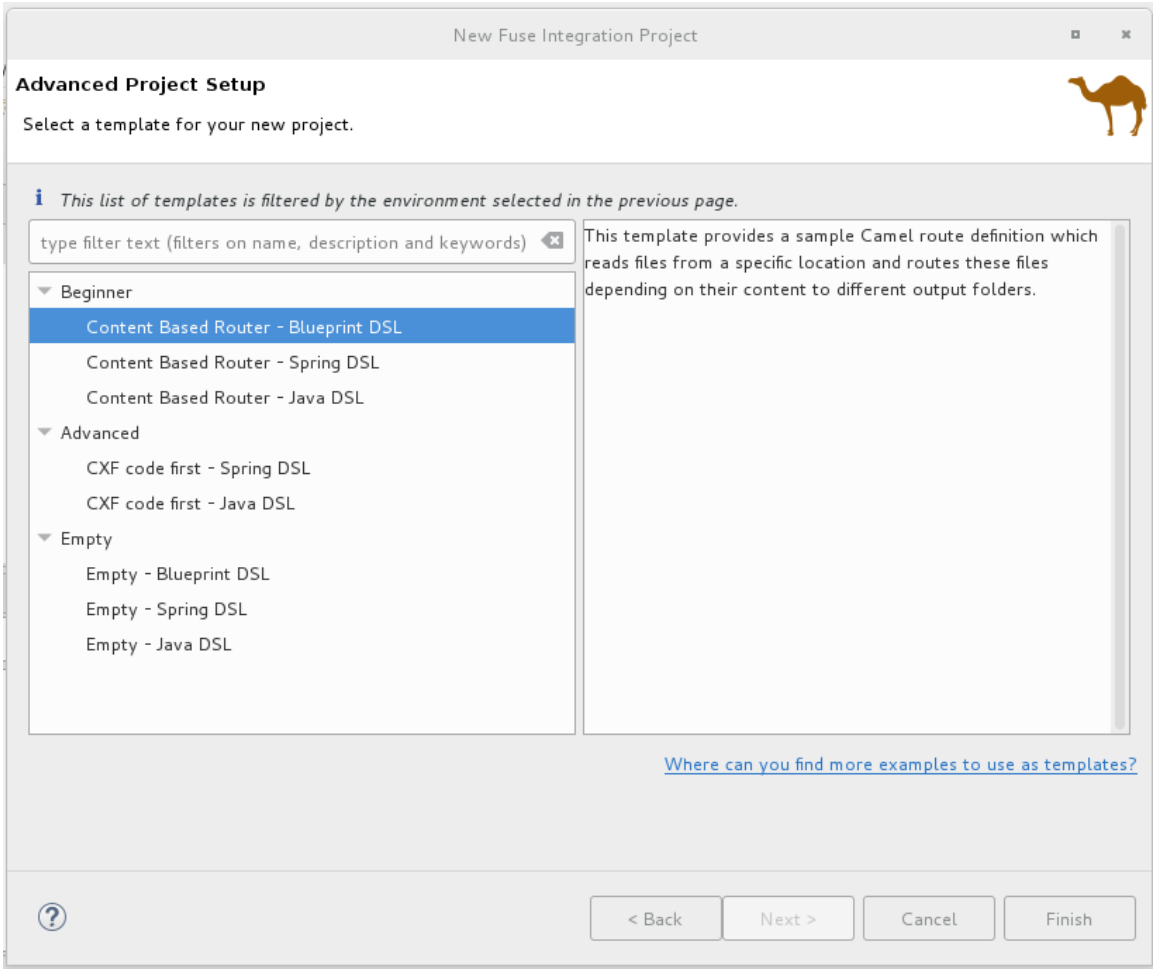


注記

Java DSL の場合、ツールはツールの Java エディターで編集することができる **CamelRoute.java** ファイルを生成しますが、グラフィカルダイアグラム表現を生成しません。

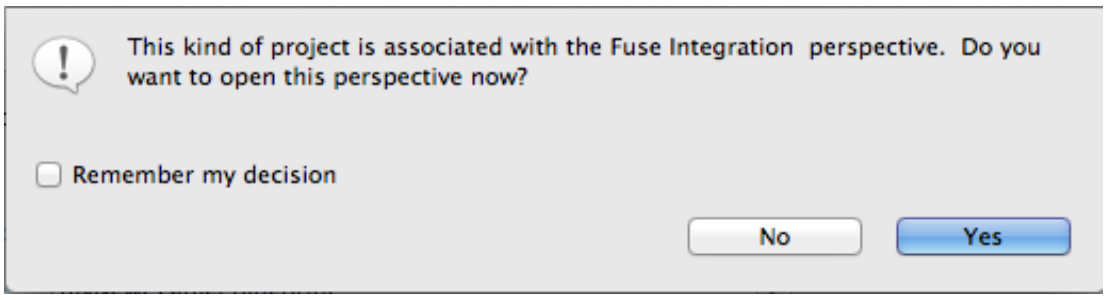
  - **Content Based Router** – 特定の場所からファイルを読み取り、メッセージの内容に応じて異なる出力フォルダーにルーティングするサンプル Camel ルートを提供します。
    このテンプレートは、Red Hat Fuse サーバーで実行されるプロジェクトを作成し、3 つの DSL をすべてサポートします。

  - **CXF code first** – CXF Web サービス呼び出しによって開始されるサンプル Camel ルートを提供します。
    このテンプレートは、Red Hat Fuse サーバーで実行されるプロジェクトを作成し、Spring および Java DSL のみサポートします。

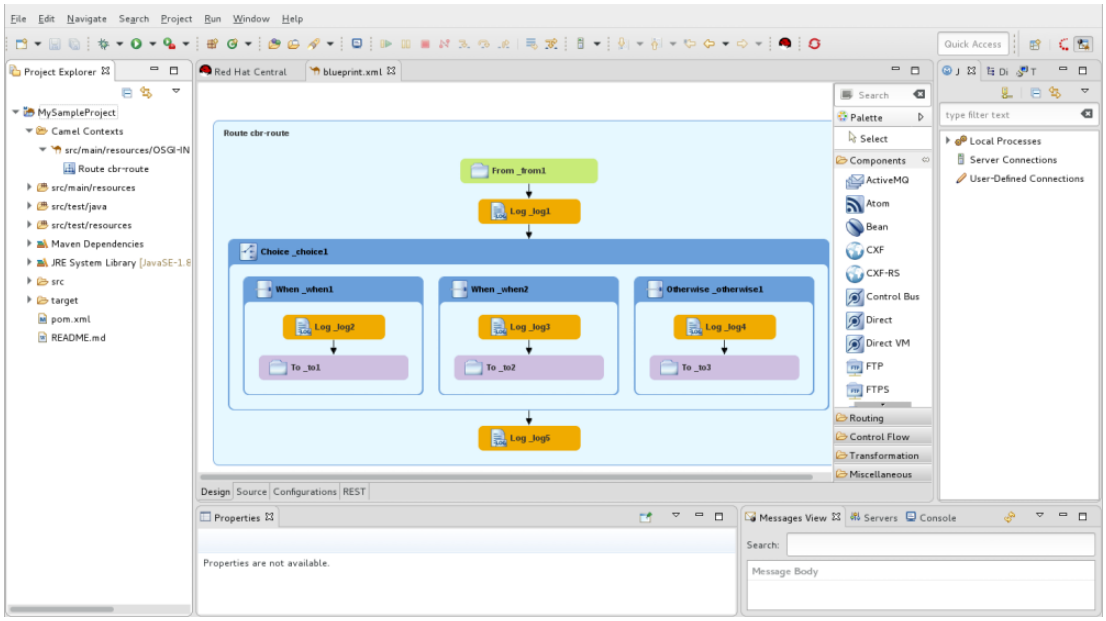


1. リストからテンプレートを選択します。
2. **Finish** をクリックします。
   ツールはプロジェクトの構築を開始し、それを **Project Explorer** ビューに追加します。

   Fuse Integration パースペクティブがまだ開いていない場合、ツールは今すぐそれに切り替えるかどうかを尋ねます。

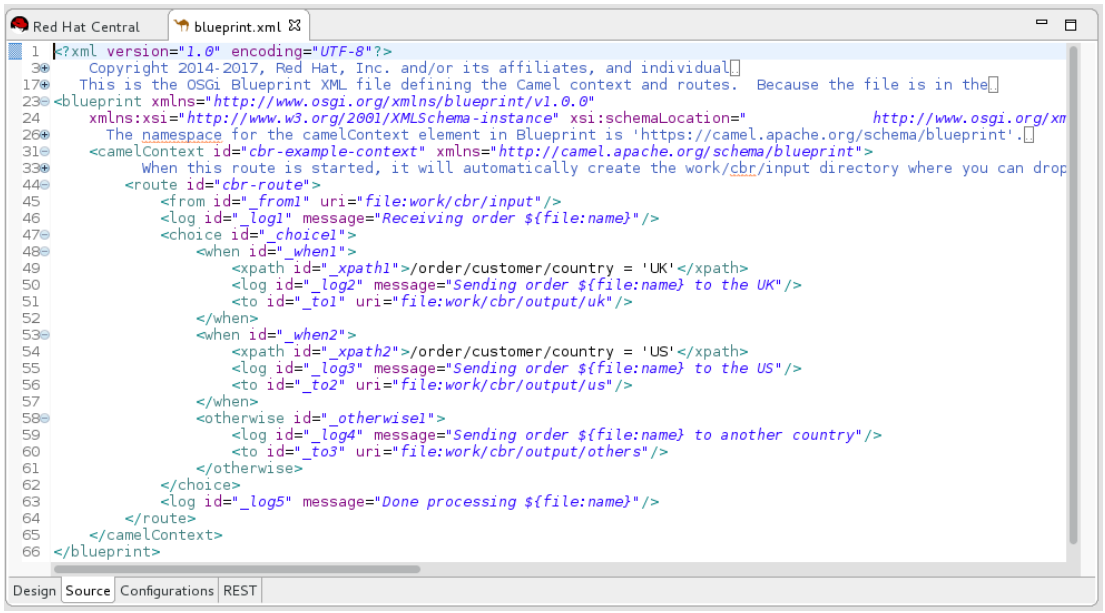


3. **Yes** をクリックして、**Fuse Integration** パースペクティブで新しいプロジェクトを開きます。

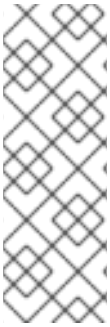


プロジェクトが **Project Explorer** ビューに表示されます。デフォルトでは、プロジェクトには Apache Camel コンテキスト (XML) ファイルが含まれています。

4. キャンバスの下部にある **Source** タブをクリックして、生成された Camel コンテキストファイルを表示します。



```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Copyright 2014-2017, Red Hat, Inc. and/or its affiliates, and individual
    This is the OSGi Blueprint XML file defining the Camel context and routes.  Because the file is in the
-->
<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/xm
      The namespace for the camelContext element in Blueprint is 'https://camel.apache.org/schema/blueprint'.
    <camelContext id="cbr-example-context" xmlns="http://camel.apache.org/schema/blueprint">
        When this route is started, it will automatically create the work/cbr/input directory where you can drop
        <route id="cbr-route">
            <from id="_from1" uri="file:work/cbr/input"/>
            <log id="_log1" message="Receiving order ${file:name}"/>
            <choice id="_choice1">
                <when id="_when1">
                    <xpath id="_xpath1">/order/customer/country = 'UK'</xpath>
                    <log id="_log2" message="Sending order ${file:name} to the UK"/>
                    <to id="_to1" uri="file:work/cbr/output/uk"/>
                </when>
                <when id="_when2">
                    <xpath id="_xpath2">/order/customer/country = 'US'</xpath>
                    <log id="_log3" message="Sending order ${file:name} to the US"/>
                    <to id="_to2" uri="file:work/cbr/output/us"/>
                </when>
                <otherwise id="_otherwise1">
                    <log id="_log4" message="Sending order ${file:name} to another country"/>
                    <to id="_to3" uri="file:work/cbr/output/others"/>
                </otherwise>
            </choice>
            <log id="_log5" message="Done processing ${file:name}"/>
        </route>
    </camelContext>
</blueprint>
```



注記

プロジェクトに別の新しい Camel コンテキストファイルを追加する場合は、10章 *新規 Camel XML ファイルの作成* を参照してください。

CXF を使用するプロジェクトをビルドする場合、Java ファイルでビルドプロセスが自動的に実行されて WSDL ファイルが生成されるようにできます。そのためには、プロジェクトの **.pom** ファイルで **java2ws** Maven プラグインを設定します。Apache CXF 開発ガイドの Maven ツーリングリファレンスの java2ws を参照してください。

MAVEN 依存関係エラーの解決

新しい Fuse Integration プロジェクトを作成した後、Maven 依存関係エラーが発生することがあります。

他の場合でも発生する可能性はありますが、通常は、プロセスが完了する前にプロジェクトのビルドをキャンセルした場合に発生します。この方法でプロセスを中断すると、プロジェクトのすべての依存関係が Maven リポジトリーからダウンロードできなくなることがよくあります。これには時間がかかる場合があります。

多くの場合、これらの依存関係エラーは、Maven 依存関係を次のように更新することで解決できます。

1. **Project Explorer** ビューで、ルートプロジェクトを右クリックしてコンテキストメニューを開きます。
2. **Maven → Update Project** を選択します。
3. **Update Maven Project** ウィザードで、以下を実行します。
   - ウィザードのリストに複数のプロジェクトが表示されている場合は、更新するプロジェクトを選択します。
   - **Force Update of Snapshots/Releases** オプションをクリックして有効にします。
4. **OK** をクリックします。
   欠落している依存関係が Maven リポジトリーからダウンロードされる際に、ワークベンチの右下隅に進行状況を示すバーが表示されます。

第2章 ルートエディターを使用したルーティングコンテキストの編集

> The following sections describe how to edit a routing context.

2.1. ルートへのパターンの追加

ルートは、接続されたパターンのシーケンスで設定されます。これは、**Route** コンテナーノード内のキャンバスに配置されると **ノード** と呼ばれます。完全なルートは通常、開始エンドポイント、一連の処理ノード、および 1 つ以上の宛先エンドポイントで設定されます。

キャンバス上のルートコンテナーにパターンを追加すると、パターンはノードのタイプを示す色になります。

- 青: コンテキストファイルのルート要素に対応するルートコンテナー、および **when** やロジックを完了する他の EIP が含まれる **otherwise** EIP 等の他のコンテナーノード
- 緑: ルートに入力するデータを入力するコンシューマーエンドポイント
- オレンジ: データ転送ルートのフローをルーティング、変換、処理、または制御する EIP
- 紫: ルートを出るデータを出力するプロデューサーエンドポイント

手順

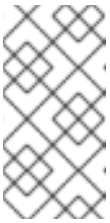
ルートにパターンを追加するには、以下を実行します。

1. **Palette** で、ルートに追加するパターンを見つけます。
2. 以下の方法のいずれかを使用します。
   - **Palette** でパターンをクリックしてから、キャンバスでルートコンテナーをクリックします。
   - パターンをターゲット **Route** コンテナーにドラッグし、ドロップする。
     または、発信接続がない既存のノード、または 2 つのノード間に存在する接続にパターンを追加して、ツールが関係するすべてのノード間の接続を自動的にワイヤリングするようにもできます。

     ツールは、その結果として得られる接続が有効かどうかをチェックし、ターゲットにパターンを追加することを許可または禁止します。有効な接続の場合、ターゲットがノードであるか接続であるかによって、ツールの動作が異なります。

     - **既存ノード** の場合、ツールは新しいノードをターゲットノードの発信側 (エディターの設定方法 に応じてその下または右側) に追加し、それらの間の接続を自動的にワイヤリングします。
     - **既存の接続** の場合、ツールは接続された 2 つのノードの間に新しいノードを挿入し、3 つのノード間の接続を自動的に再度ワイヤリングします。
3. オプションで、2 つのノードを手動で接続できます。
   a. キャンバスの **Route** コンテナーで、ソースノードを選択して、そのコネクター矢印を表示します。

b. ソースノードのコネクター矢印 () をターゲットノードにドラッグし、マウスボタンを解放してコネクターをドロップします。



注記

すべてのノードを接続できるわけではありません。ソースノードを無効なターゲットノードに接続しようとすると、ツールは ⊘ シンボルをマウスカーソルに付けて表示し、コネクターをターゲットノードに付ける操作に失敗します。

4. ルートコンテナー内にパターンを追加した後、有効な接続を確立できる限り、ルートコンテナー内の別の場所またはキャンバス上の別のルートコンテナーにパターンをドラッグできます。移動によって別の有効な接続を確立できる限り、すでに接続されている既存のノードを再配置することもできます。
こちら をクリックすると、エンドポイントを再配置する方法を説明する短い動画を閲覧できます。

5. **File → Save** を選択します。ツールは、ルートが完全であるかどうかに関係なく、ルートをコンテキストファイルに保存します。

Route コンテナーのキャンバスに新しいパターンが表示され、選択したノードになります。**Properties** ビューには、編集可能な新しいノードのプロパティーのリストが表示されます。

レイアウト方向の変更

あるノードを別のノードに接続すると、ツールはルートエディターのレイアウト設定に従ってレイアウトを更新します。デフォルトは **Down** です。

ルートエディターのレイアウト設定にアクセスするには、以下を実行します。

- Linux および Windows マシンでは、**Windows → Preferences → Fuse Tooling → Editor → Choose the layout direction for the diagram editor**を選択します。

関連トピック

- 「パターンの設定」
- 「ルートからパターンを削除」

2.2. パターンの設定

概要

ほとんどのパターンでは、明示的な設定が必要です。たとえば、エンドポイントには明示的に **URI** を入力する必要があります。

ツールの **Properties** ビューには、特定のパターンがサポートするすべての設定詳細を一覧表示するフォームがあります。**Properties** ビューには、次の便利な機能もあります。

- すべての必須プロパティーに値があることを検証する機能
- 指定された値がプロパティーの正しいデータ型であることを検証する機能

- 値のセットが固定されているプロパティーのドロップダウンリスト
- Apache Camel Spring 設定から利用可能な Bean 参照が入力されたドロップダウンリスト

手順

パターンを設定するには、以下を実行します。

1. キャンバスで、設定するノードを選択します。
   Properties ビューには、選択したノードのすべてのプロパティーが一覧表示され、編集できます。EIP の場合、**Details** タブには、パターンのプロパティーがすべて一覧表示されます。**Components** ドロワーのコンポーネントの場合、**Details** タブには一般的なプロパティーと値が必要なプロパティーが一覧表示され、**Advamced** タブには機能ごとにグループ化された追加のプロパティーが一覧表示されます。

   Documentation タブには、パターンとそのそのプロパティーが記載されています。
2. **Properties** ビューのフィールドを編集して、ノードを設定します。
3. 完了したら、メニューバーから **File → Save** を選択して作業を保存します。

2.3. ルートからパターンを削除

概要

ルートを開発および更新するときに、ルートのノードを 1 つ以上削除する必要がある場合があります。ノードの アイコンを使用すると、これを簡単に実行できます。キャンバスからノードを削除すると、ルート内の他のノードとの接続もすべて削除され、そのノードはコンテキストファイル内の対応するルート要素から削除されます。

注記

ノードのコンテキストメニューを開いて **Remove** を選択しても、ノードを削除できます。

手順

ルートからノードを削除するには、以下を実行します。

1. 削除するノードを選択します。
2. その アイコンをクリックします。
3. この要素を削除するかどうかを確認するメッセージが表示されたら、**Yes** をクリックします。

ノードとそのすべての接続がキャンバスから削除され、ノードがコンテキストファイルの対応するルート要素から削除されます。

関連トピック

- 「ルートへのパターンの追加」

2.4. ルーティングコンテキストへのルートの追加

概要

XML コンテキストファイル内の camelContext 要素は、ルーティングコンテキストを作成します。camelContext 要素には 1 つ以上のルートが含まれ、キャンバスに **Route** コンテナーノードとして表示される各ルートは、生成された camelContext 要素のルート要素にマッピングされます。

手順

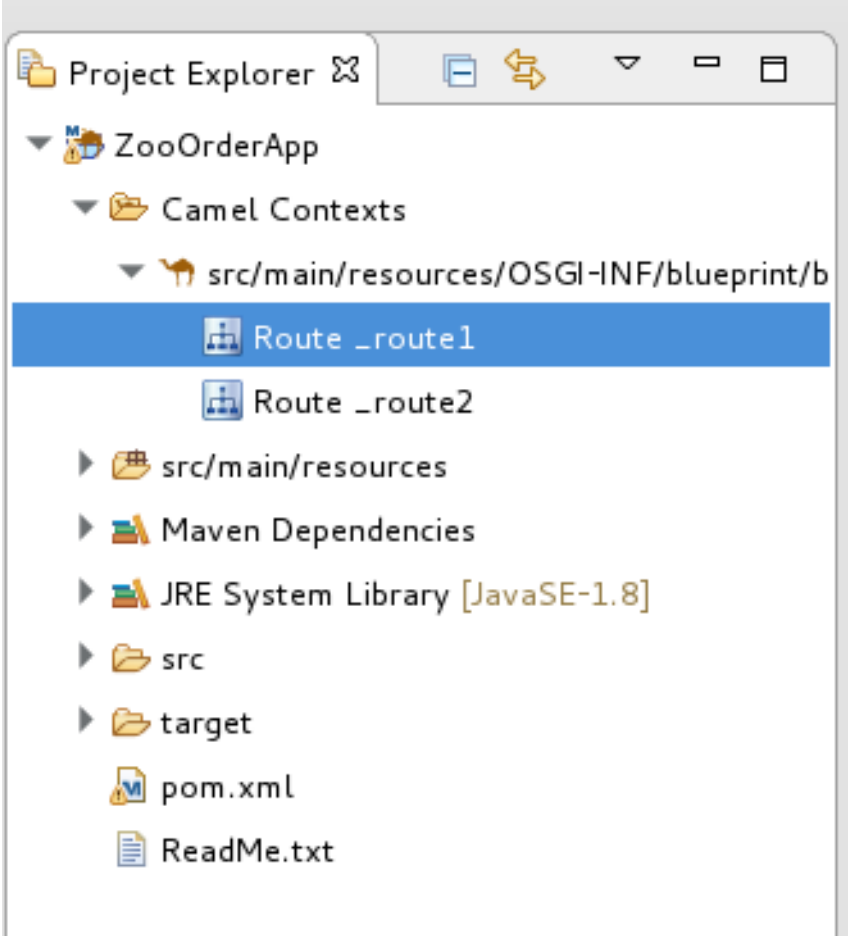
camelContext に別のルートを追加するには、以下を実行します。

1. **Design** タブで、次のいずれかを実行します。
   - **Palette** の **Routing** ドロワーで **Route** パターンをクリックし、ルートを配置するキャンバスをクリックする。
   - **Palette** の **Routing** ドロワーから **Route** パターンをドラッグしてキャンバスにドロップする。
     Properties ビューには、編集可能な新しいルートのプロパティーのリストが表示されます。
2. **Properties** ビューで、以下を入力します。
   - ルートの **Id** フィールドの新しいルートの ID (例: **Route2**)

     注記

     ツールは、キャンバスにドロップされた EIP およびコンポーネントパターンに ID を自動的に割り当てます。これらの自動生成された ID を独自の ID に置き換えて、プロジェクト内のルートを区別することができます。
   - **Description** フィールドのルートの説明
   - 必要に応じて、その他のプロパティーの値。必要なプロパティーはアスタリスク (*) で示されます。
3. メニューバーで、**File → Save** を選択して、ルーティングコンテキストファイルに加えた変更を保存します。
4. 複数のルートを切り替えるには、**Project Explorer** ビューでプロジェクトの **Camel Contexts** フォルダーにあるエントリーをクリックして、キャンバスで表示するルートを選択します。



5. スペースが許す範囲でコンテキスト内のすべてのルートを表示するには、**Project Explorer** ビューでコンテキストファイルエントリーをクリックします。

6. キャンバスにルートを追加するときにツールによって生成されたコードを表示するには、**Source** タブをクリックします。

注記

また、camelContext 要素内の既存リストに <route/> 要素を追加すると、**Source** タブでルートを追加できます。

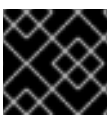
2.5. ルートの削除

概要

場合によっては、ルーティングコンテキストからルート全体を削除する必要があります。Route コンテナーの アイコンを使用すると、これを簡単に実行できます。ルートを削除すると、ルートコンテナー内のすべてのノードも削除され、コンテキストファイル内の対応するルート要素が削除されます。

注記

ルートコンテナーのコンテキストメニューで **Remove** を選択しても、ルートを削除できます。



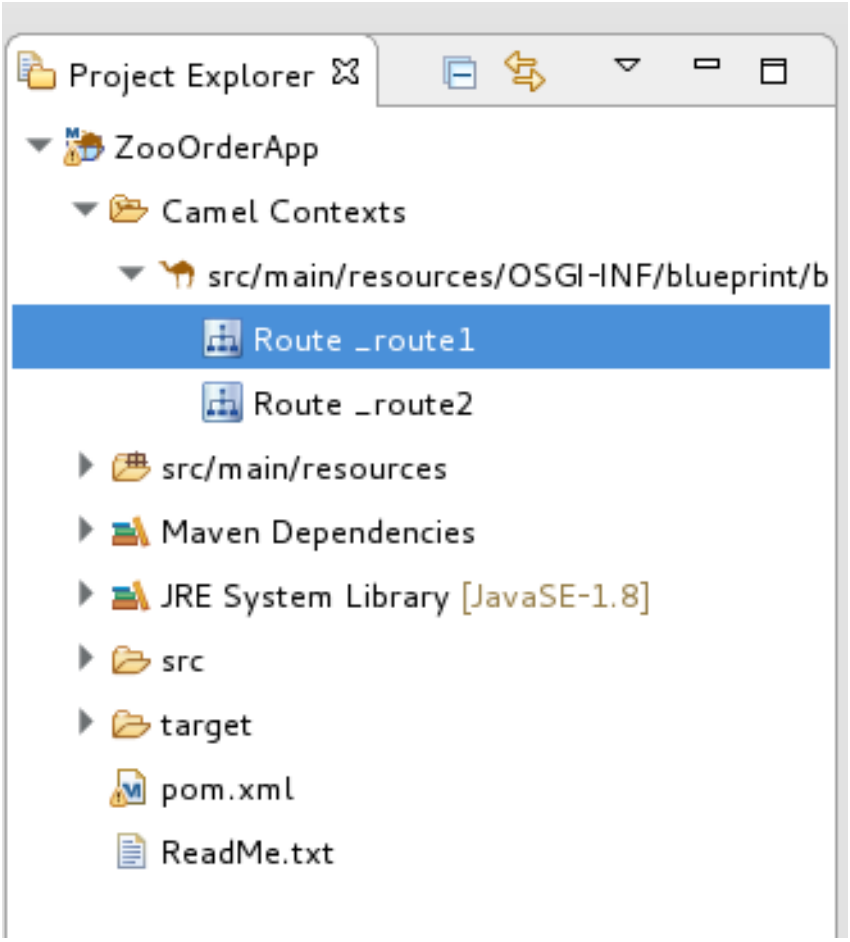
重要

この操作を元に戻すことはできません。

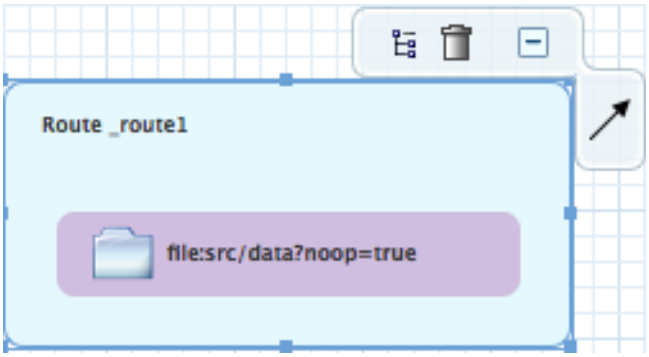
手順

ルートを削除するには、以下を実行します。

1. ルーティングコンテキストに複数のルートが含まれている場合は、最初に **Project Explorer** ビューで削除するルートを選択します。



2. キャンバスで Route コンテナーの アイコンをクリックします。



3. この要素を削除するかどうかを確認するメッセージが表示されたら、**Yes** をクリックします。

キャンバス、コンテキストファイル、および **Project Explorer** ビューからルートが削除されます。

2.6. グローバルエンドポイント、データ形式、BEAN の追加

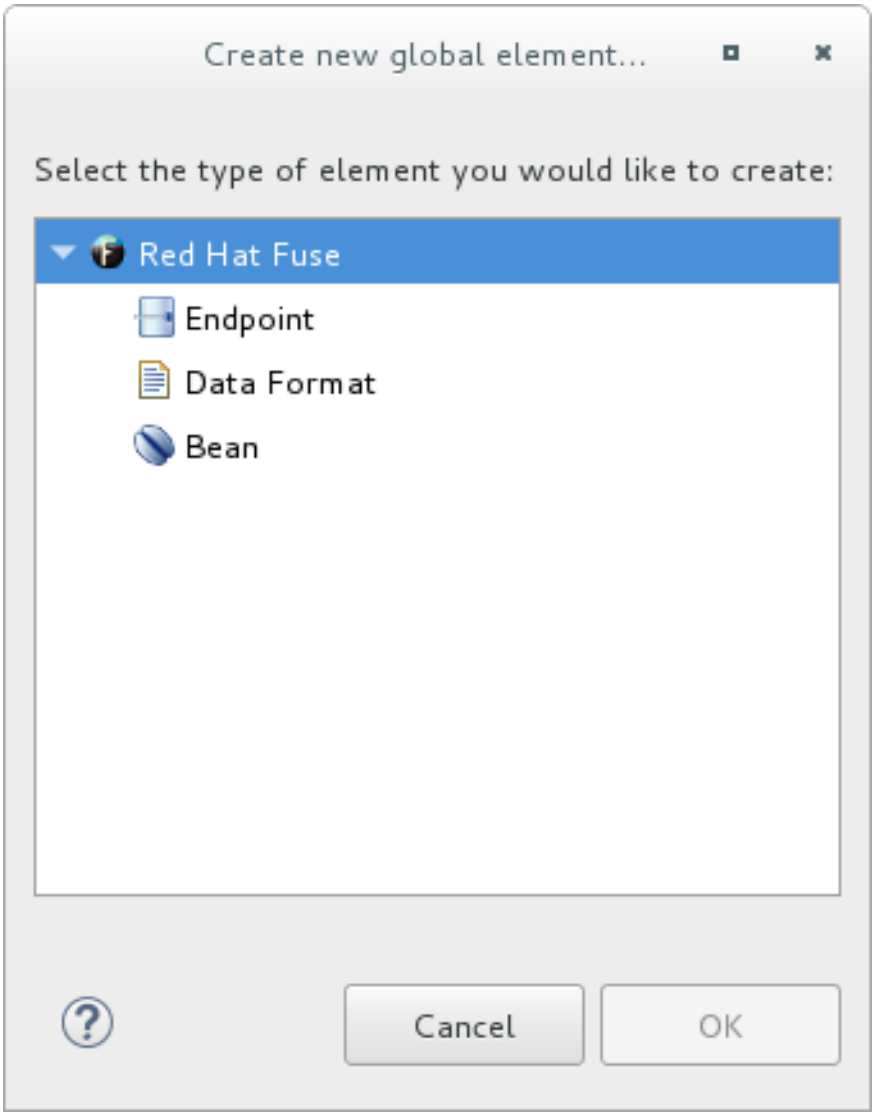
概要

一部のルートは、グローバルエンドポイント、グローバルデータ形式、またはグローバル Bean によって提供される共有設定に依存しています。ルートエディターの **Configurations** タブを使用して、プロジェクトのルーティングコンテキストファイルにグローバル要素を追加できます。

ルーティングコンテキストファイルにグローバル要素を追加するには、以下を実行します。

1. ルートエディターでルーティングコンテキストファイルを開きます。
2. グローバル設定がある場合は、ルートエディターの下部にある **Configurations** タブをクリックしてそれを表示します。

3. **Add** をクリックして、**Create a new global element** ダイアログボックスを開きます。

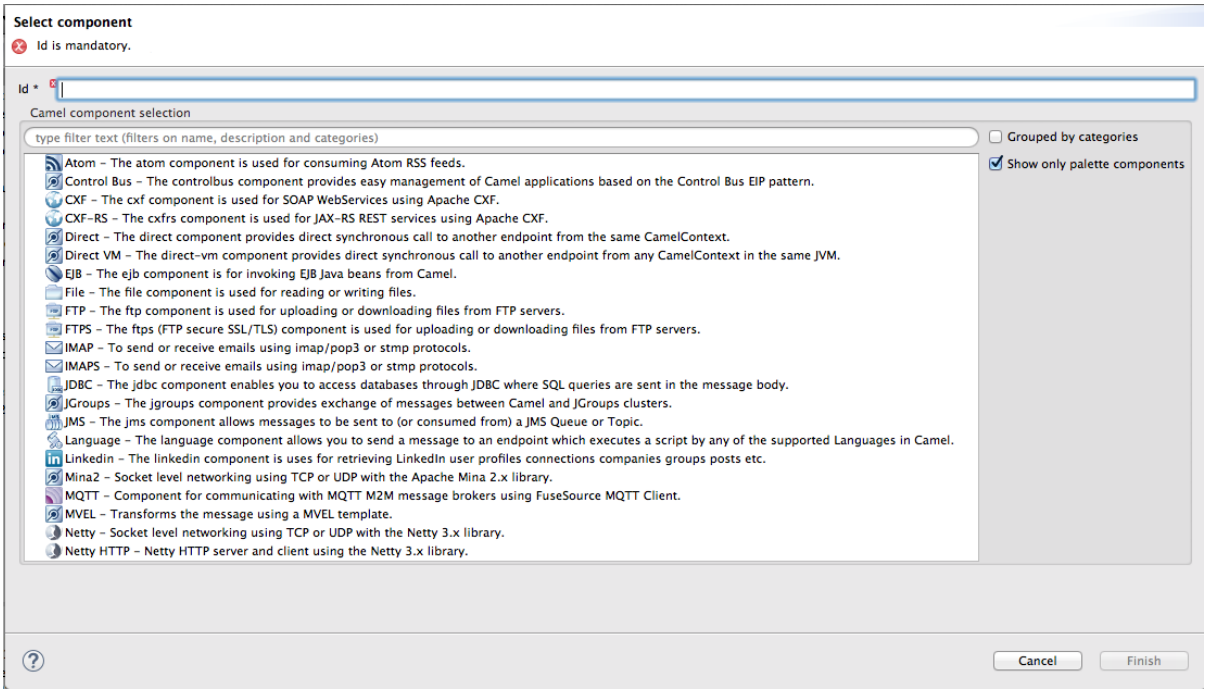


オプションは次のとおりです。

- エンドポイント:「グローバルエンドポイントの追加」 を参照してください。
- データ形式:「グローバルデータ形式の追加」 を参照してください。
- Bean:「グローバル Bean の追加」 を参照してください。

グローバルエンドポイントの追加

1. **Create a new global element**ダイアログボックスで **Endpoint** を選択し、**OK** をクリックして **Select component** ダイアログボックスを開きます。

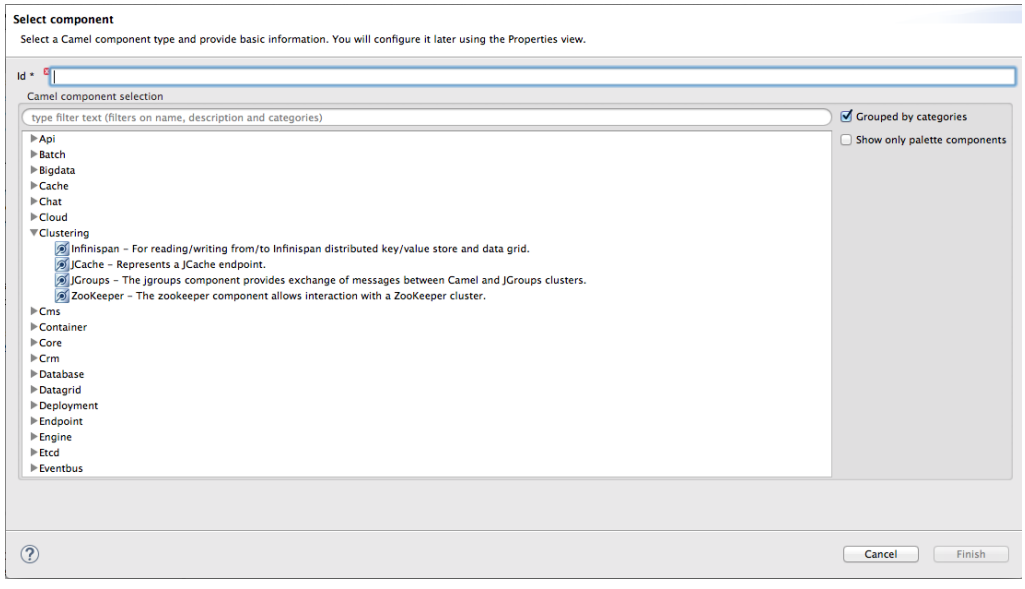


注記

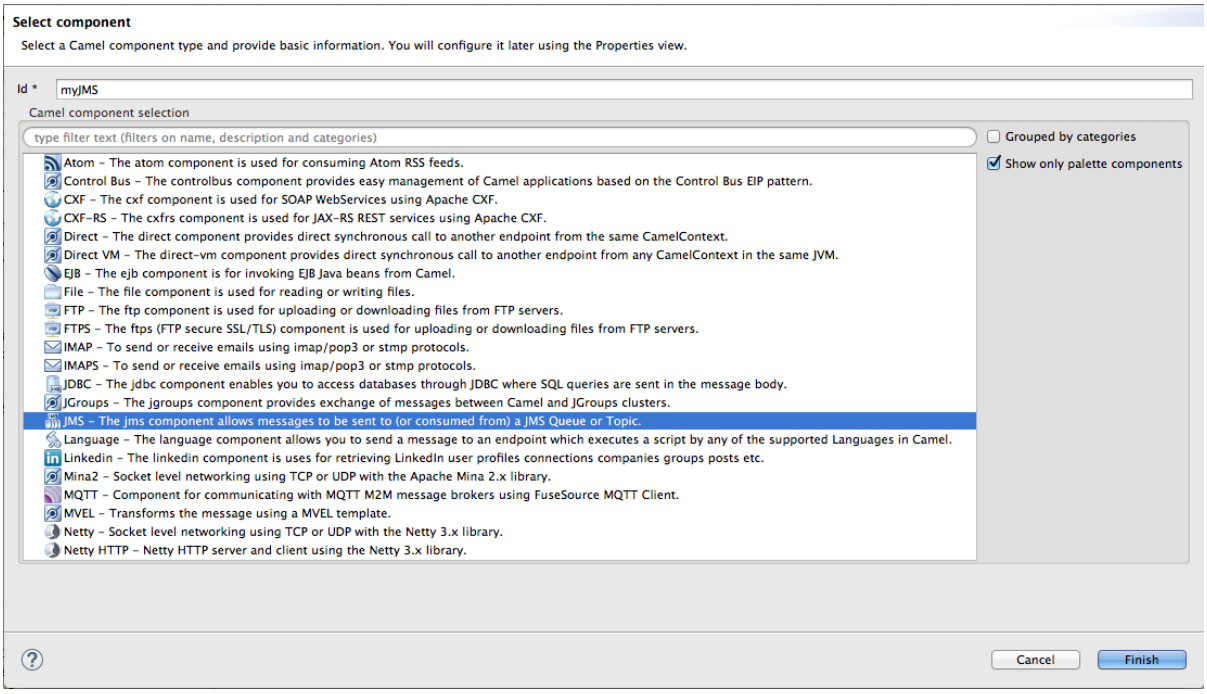
デフォルトでは、**Select component** ダイアログボックスが開き、**Show only palette components** オプションが有効になっています。使用可能なすべてのコンポーネントを表示するには、このオプションのチェックを外します。

注記

Grouped by categories オプションは、コンポーネントをタイプ別にグループ化します。

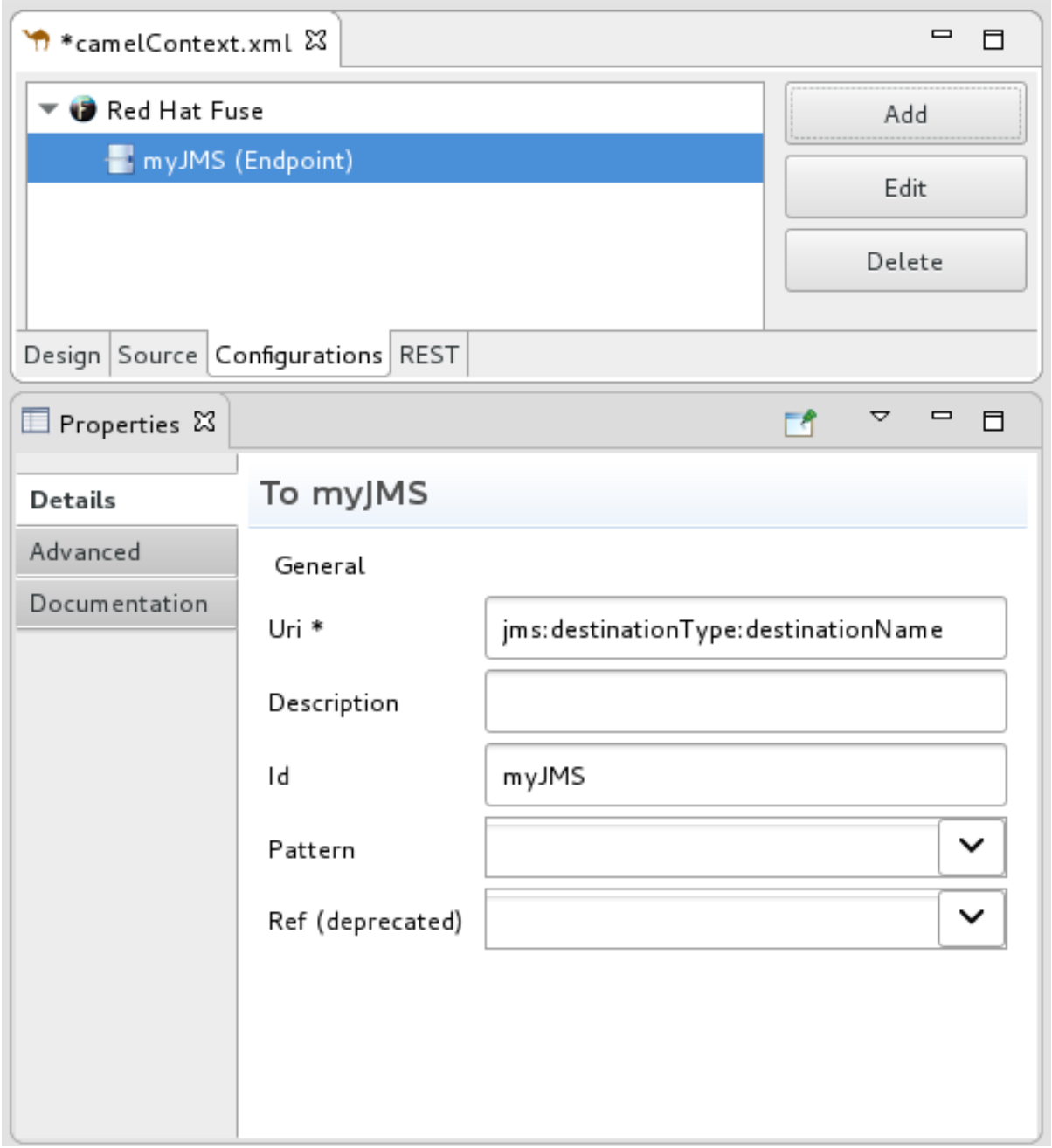


2. **Select component** ダイアログで Camel コンポーネントのリストをスクロールして、コンテキストファイルに追加するコンポーネントを見つけて選択し、次に **Id** フィールドにその ID を入力します。



この例では、JMS コンポーネントが選択され、**myJMS** が Id の値になります。

3. **Finish** をクリックします。



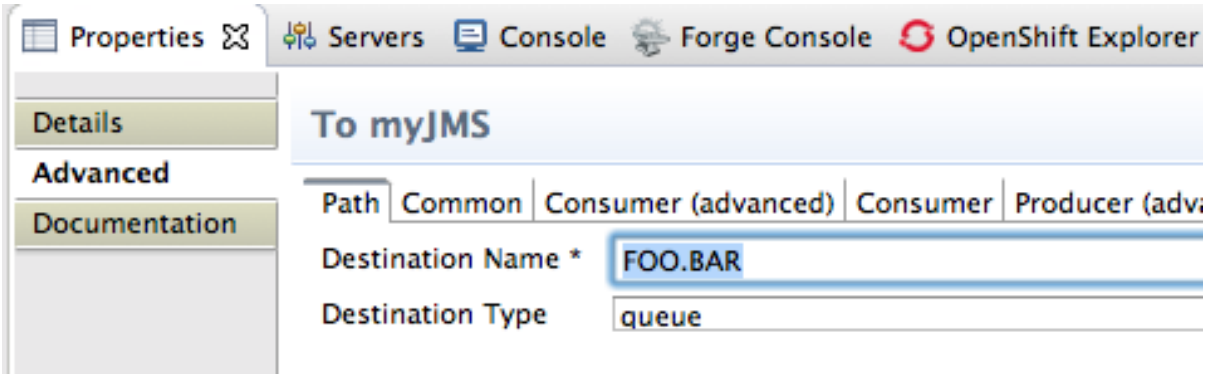
これで、必要に応じて **Properties** ビューでプロパティーを設定できるようになりました。

ツールによって、[globalEndptSelect]でコンポーネントの **Id** フィールドに入力した値が **Id** に自動入力されます。この例では、Camel はコンポーネントのスキーマ (ここでは **jms:**) で始まる **uri** (必須フィールド) を構築しますが、**destinationName** と **destinationType** を指定してコンポーネントの **uri** を完了する必要があります。

注記

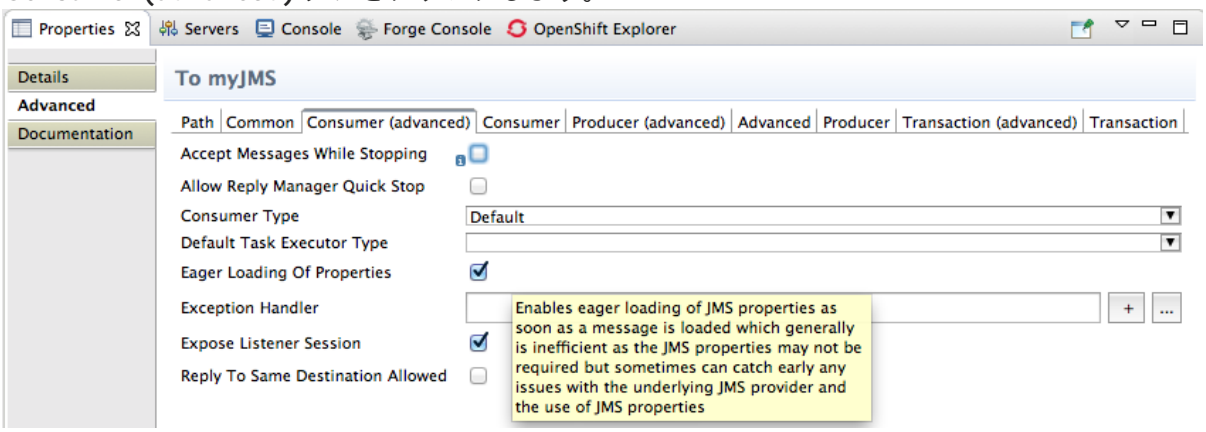
JMS コンポーネントの場合、宛先のタイプはデフォルトで **queue** に設定されます。このデフォルト値は、**Destination Name** (必須フィールド) に値を入力するまで、**Details** ページの **uri** フィールドに表示されません。

4. コンポーネントの URI を完了するには、**Advanced** をクリックします。
5. **Destination Name** フィールドに、宛先エンドポイントの名前を入力します (例: **FOO.BAR**)。**Destination Type** フィールドにエンドポイント宛先のタイプを入力します (例: **queue**、**topic**、**temp:queue**、または **temp:topic**)。



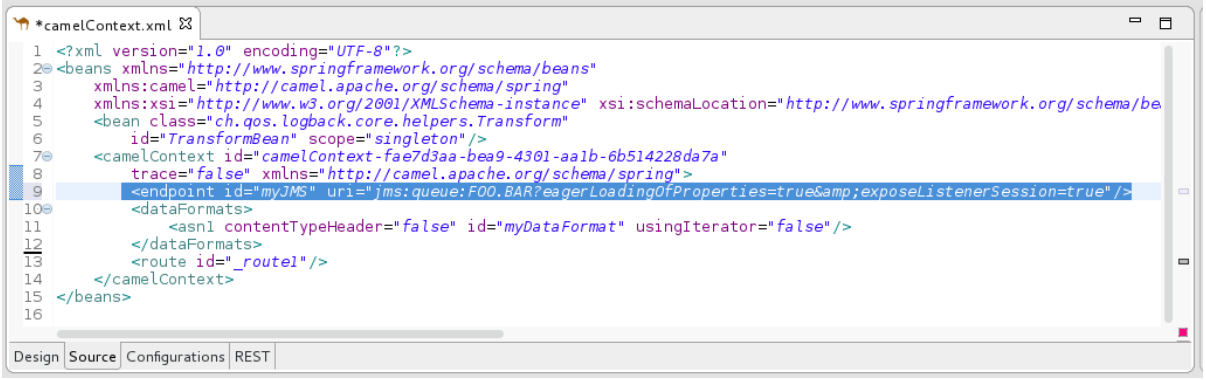
Properties ビューの Details タブと Advanced タブから、特定のコンポーネントの設定に使用できるすべてのプロパティーにアクセスできます。

6. Consumer (advanced) タブをクリックします。



プロパティーの Eager Loading Of Properties と Expose Listener Session を有効にします。

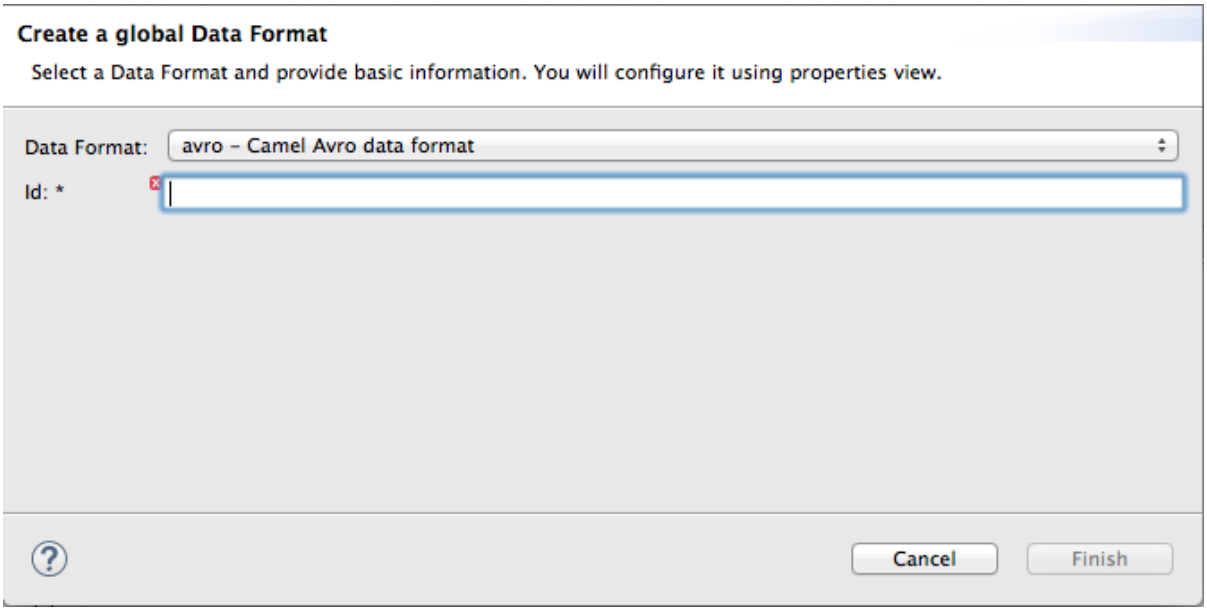
7. ルートエディターで Source タブに切り替えて、最初のルート要素の前にツールがコンテキストファイル (この例では、設定された JMS エンドポイント) に追加したコードを確認します。



8. 完了したら、メニューバーで File → Save を選択して変更を保存します。

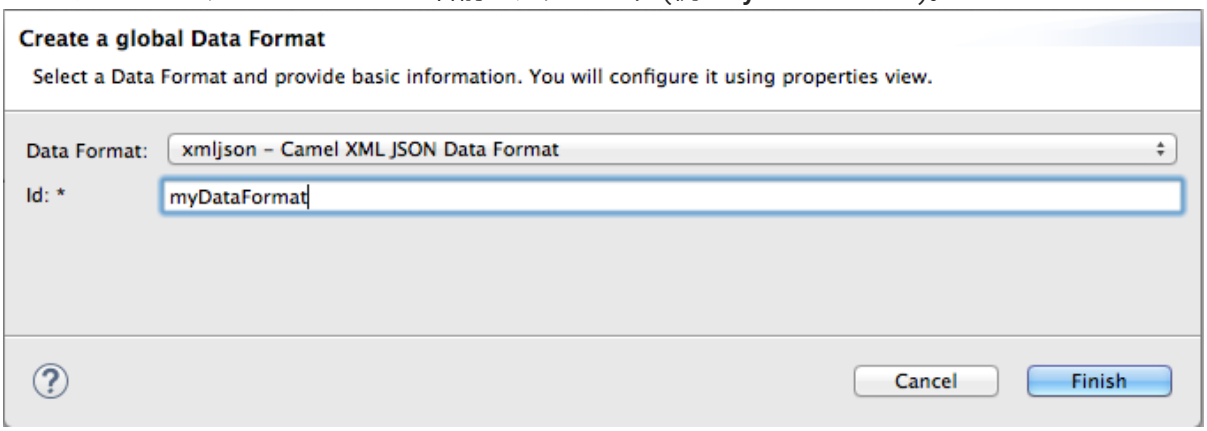
グローバルデータ形式の追加

1. Create a new global element ダイアログボックスで、Data Format を選択し、OK をクリックして Create a global Data Format ダイアログボックスを開きます。

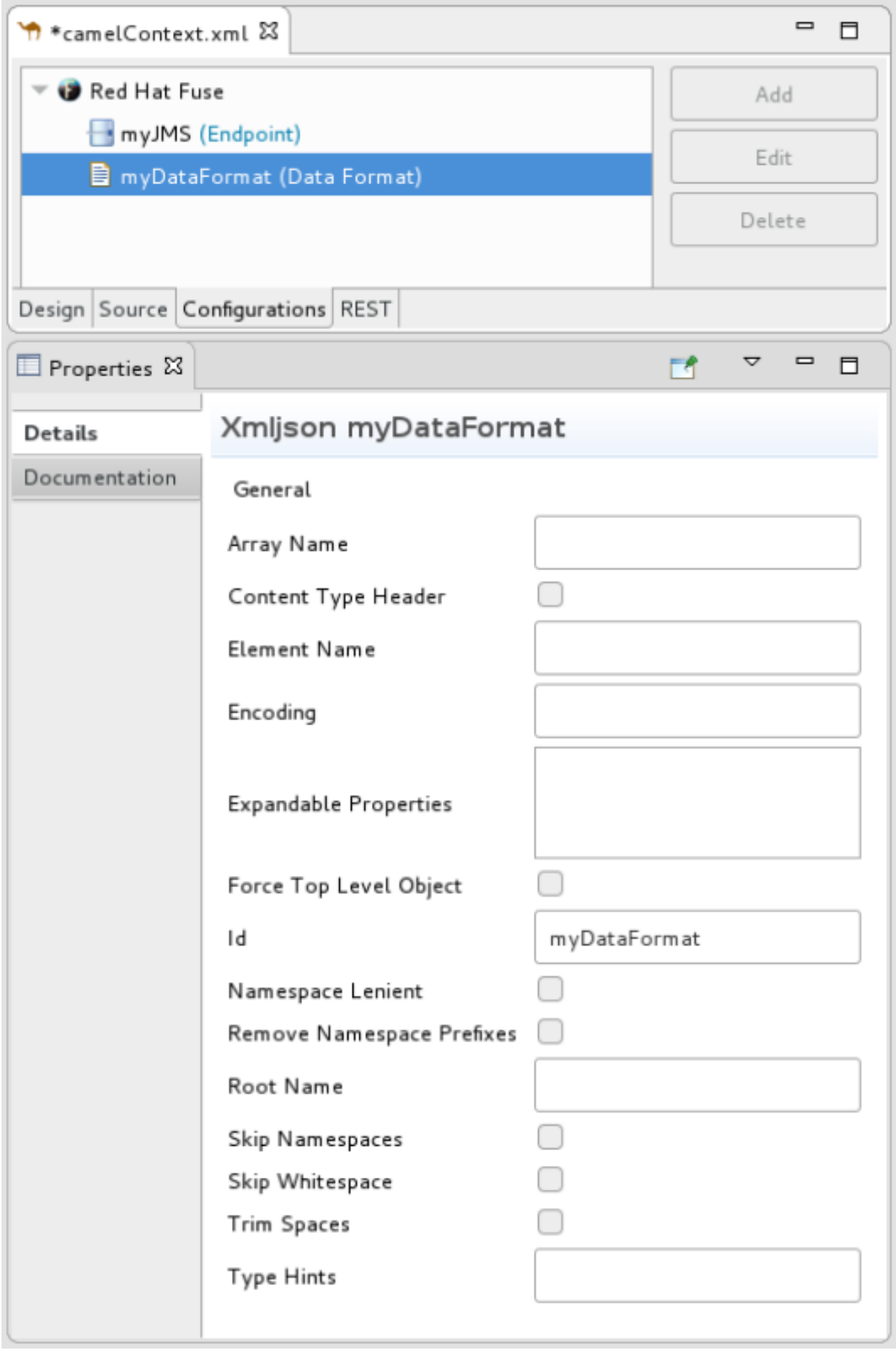


データフォーマットのデフォルトは、利用可能なフォーマットリストの一番上の **avro** です。

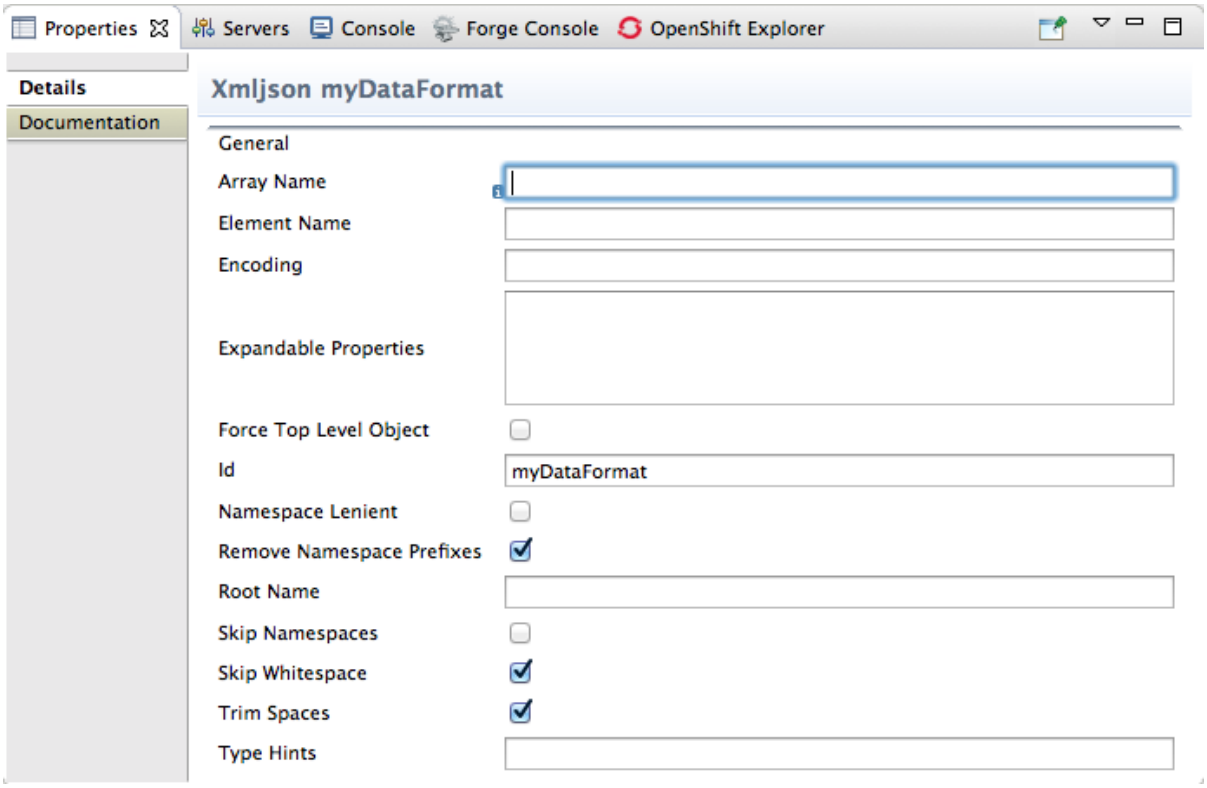
2. **Data Format** ドロップダウンメニューを開き、**xmljson** などの形式を選択します。
3. **Id** フィールドに、フォーマットの名前を入力します (例: **myDataFormat**)。



4. **Finish** をクリックします。



5. **Properties** ビューで、プロジェクトに応じてプロパティー値を設定します。以下はその例です。



6. ルートエディターで、**Source** タブをクリックして、ツールがコンテキストファイルに追加したコードを確認します。この例では、設定された xmljson データ形式は最初のルート要素の前にあります。

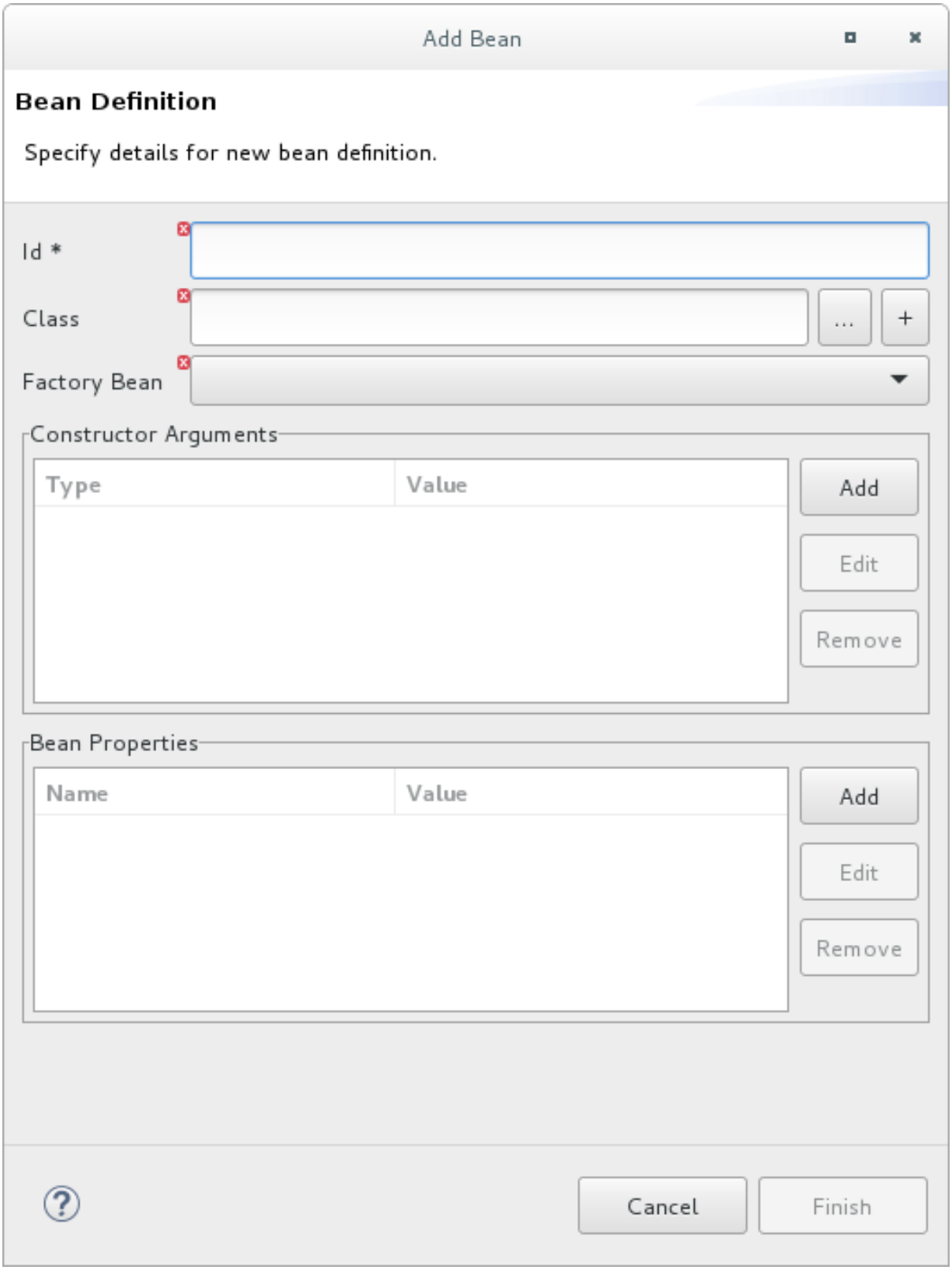
7. 完了したら、メニューバーで **File → Save** を選択して変更を保存します。

グローバル Bean の追加

グローバル Bean は、ルート内のどこからでも参照できるルート外の Bean 定義を有効にします。**Bean** コンポーネントをパレットからルートにコピーすると、**Properties** ビューの **Ref** ドロップダウンで定義済みのグローバル Bean を見つけることができます。**Bean** コンポーネントが参照するグローバル Bean を選択します。

グローバル Bean 要素を追加するには、以下を実行します。

1. **Create a new global element**ウィンドウで、**Bean** を選択し、**OK** をクリックして **Bean Definition** ダイアログボックスを開きます。



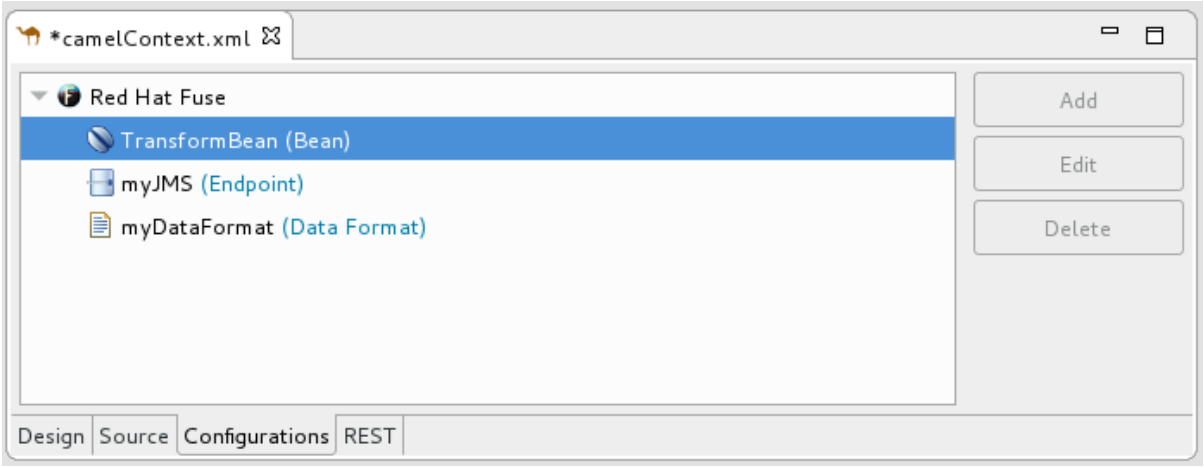
2. **Id** フィールドに、グローバル Bean の ID (例: **TransformBean**) を入力します。ID は、設定内で一意である必要があります。

3. Bean クラスまたはファクトリー Bean を識別します。
ファクトリー Bean を指定するには、ファクトリークラスが指定された別のグローバル Bean をすでに追加している必要があります。次に、そのグローバル Bean を選択して、グローバル Bean ファクトリーとして宣言できます。Bean ファクトリークラスの 1 つのインスタンスがランタイムにあります。他のグローバル Bean は、そのクラスのファクトリーメソッドを呼び出して、他のクラスの独自のインスタンスを作成できます。

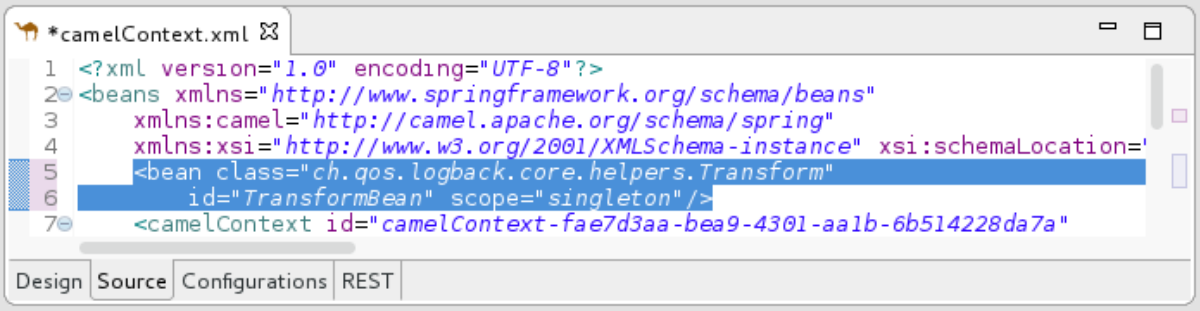
Class フィールドに入力するには、次のいずれかを実行します。

- プロジェクトまたは参照プロジェクトにあるクラスの名前を入力します。
- ... をクリックして、プロジェクトまたは参照されるプロジェクトのクラスに移動し、選択する。
- + をクリックして新しい Bean クラスを定義し、それをグローバル Bean として追加します。

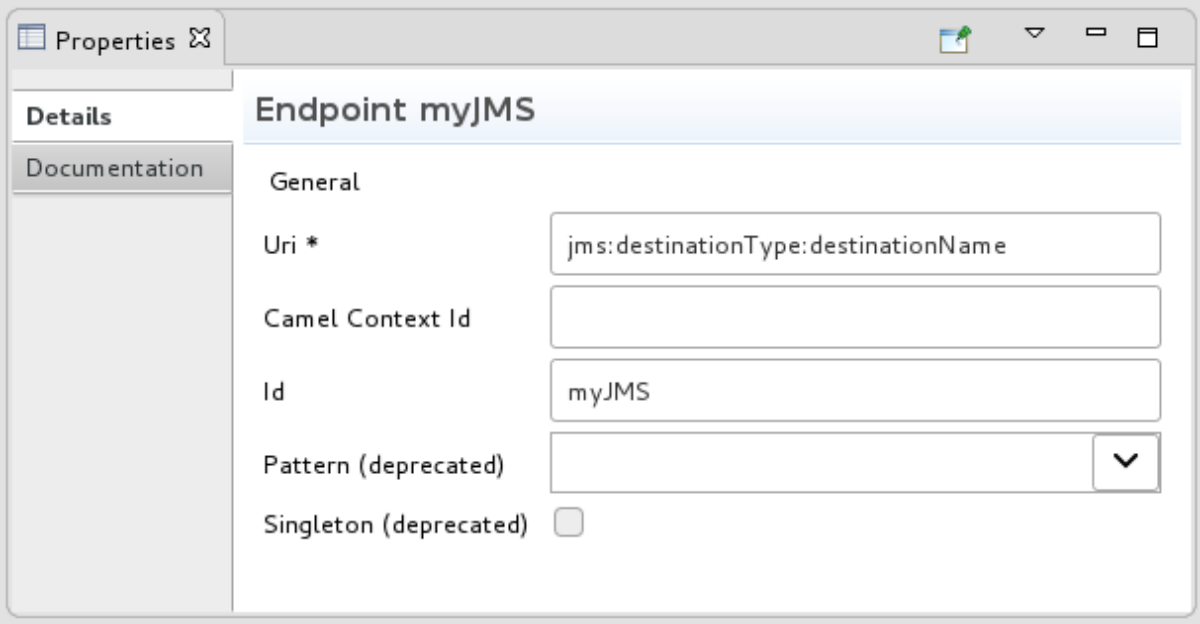
4. 追加する Bean で 1 つ以上の引数が必要な場合、**Constructor Arguments** セクションで各因数に対して以下を実行します。
   a. **Add** をクリックします。
   b. 必要に応じて、**Type** フィールドに引数のタイプを入力します。デフォルトは **java.lang.String** です。
   c. **Value** フィールドに、引数の値を入力します。
   d. **OK** をクリックします。
5. オプションで、グローバル Bean にアクセス可能な 1 つ以上のプロパティーを指定します。**Bean Properties** セクションで、プロパティーごとに以下を実行します。
   a. **Add** をクリックします。
   b. **Name** フィールドにプロパティー名を入力します。
   c. **Value** フィールドに、プロパティーの値を入力します。
   d. **OK** をクリックします。
6. **Finish** をクリックして、グローバル Bean を設定に追加します。指定したグローバル Bean ID は、次のように **Configurations** タブに表示されます。

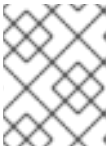


7. **Source** タブに移動し、ツールがコンテキストファイルに追加した **bean** 要素を表示します。以下に例を示します。



8. **Configurations** タブをクリックしてグローバル要素のリストに戻り、グローバル Bean を選択して、その標準プロパティーを **Properties** 次に例を示します。





注記

グローバル Bean を追加したときに指定したプロパティーを表示または編集するには、**Configurations** タブで Bean を選択し、**Edit** をクリックします。

9. 必要に応じてグローバル Bean プロパティーを設定します。

- **Depends-on** は、このグローバル Bean の前に作成する必要がある Bean を識別するために使用できる文字列です。依存 Bean の ID (名前) を指定します。たとえば、**TransformBean** を追加し、**Depends-on** を **ChangeCaseBean** に設定すると、**ChangeCaseBean** を作成する必要があり、**TransformBean** を作成できます。Bean が破棄されると、**TransformBean** が最初に破棄されます。

- **Factory-method** は、グローバル Bean がファクトリークラスである場合にのみ役立ちます。この状況では、Bean が参照されるときに呼び出される静的ファクトリーメソッドを指定または選択します。

- **スコープは singleton** または **prototype** です。デフォルトは **singleton** で、Bean が呼び出されるたびに Camel が同じ Bean のインスタンスを使用することを示します。Bean が呼び出されるたびに Camel が Bean の新規インスタンスを作成するようにするには、**prototype** を指定します。

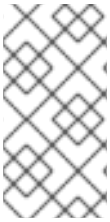
- **init-method**: Bean の参照時に呼び出す Bean の **init()** メソッドを指定または選択できます。

- **Destroy-method** を使用すると、Bean によって実行される処理が完了したときに呼び出す Bean の破棄メソッドを指定または選択できます。

10. 完了したら、メニューバーで **File → Save** を選択して変更を保存します。

グローバル要素の削除

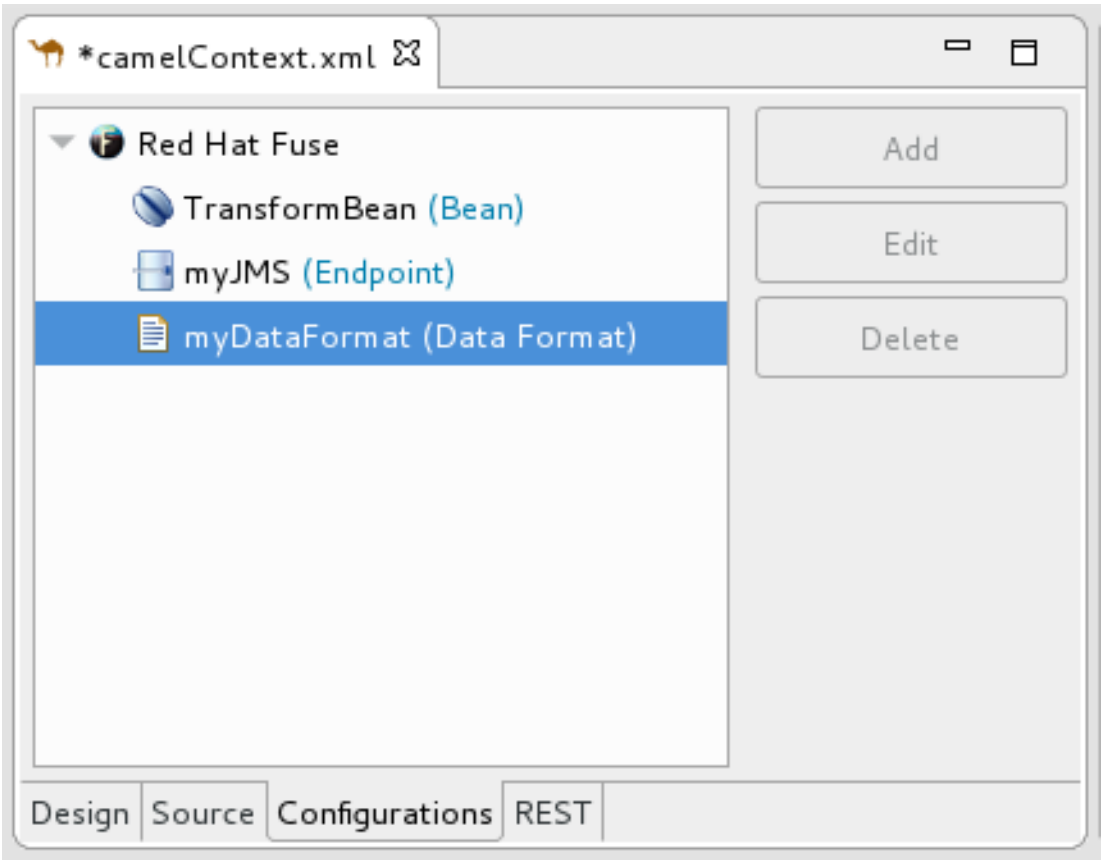
以前にルーティングコンテキストに追加されたエンドポイント、データ形式、または Bean を削除する場合でも、手順は同じです。



注記

グローバル要素の削除を元に戻す操作は実行できません。設定に保持したいグローバル要素を誤って削除した場合、保存せずにコンテキストファイルを閉じることで削除を元に戻すことができる場合があります。これが不可能な場合は、誤って削除したグローバル要素を再度追加してください。

1. **Configurations** タブで、削除するグローバル要素を選択します。
たとえば、「グローバルデータ形式の追加」で追加したデータフォーマット **myDataFormat** を削除する場合は、以下のようになります。



2. **Delete** をクリックします。
グローバル要素 **myDataFormat** は **Configurations** タブから消えます。

3. **Source** タブに切り替えて、ツールがルーティングコンテキストから XML コードを削除したことを確認します。

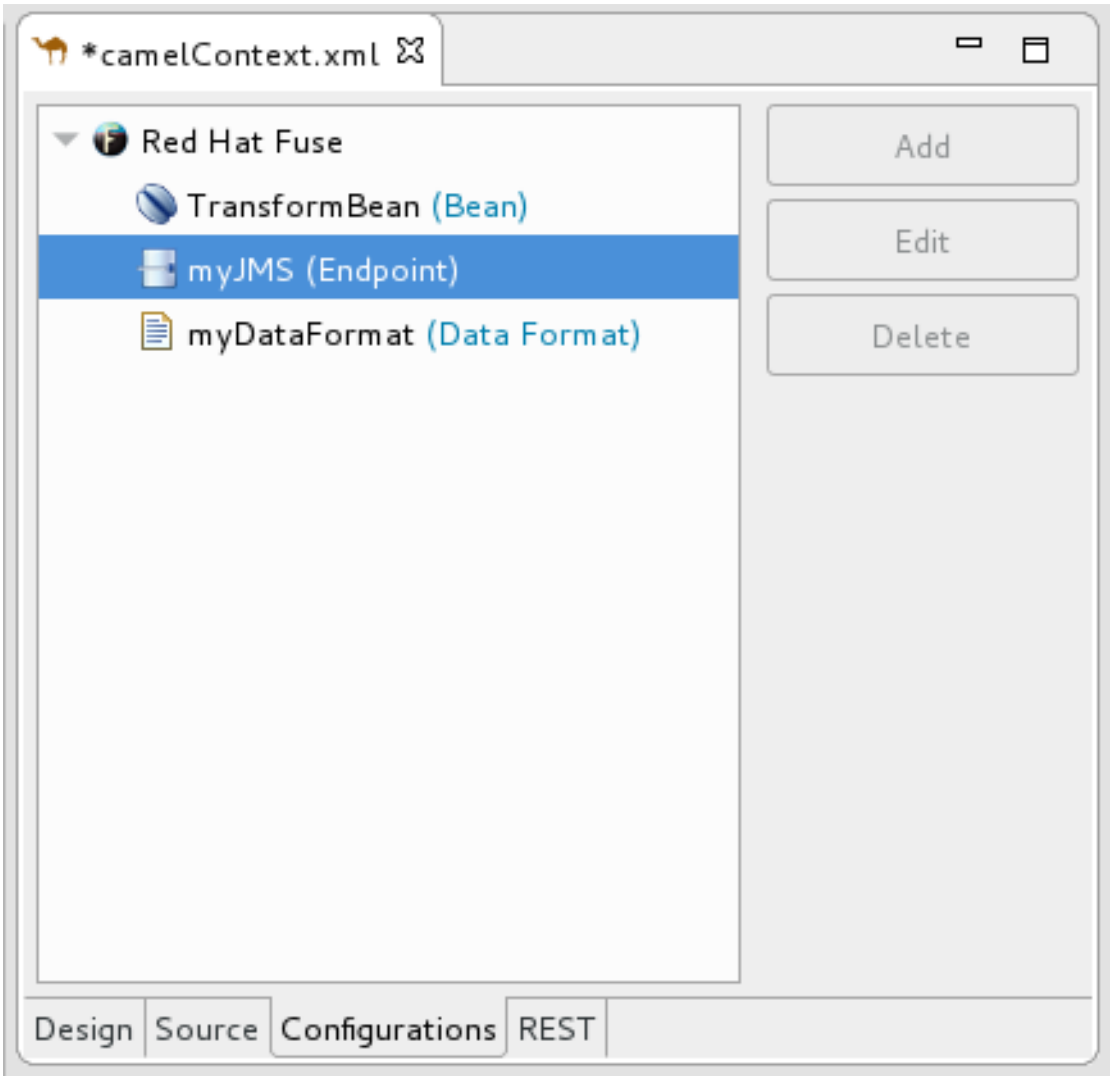
4. 完了したら、メニューバーで **File → Save** を選択して変更を保存します。

グローバル要素の編集

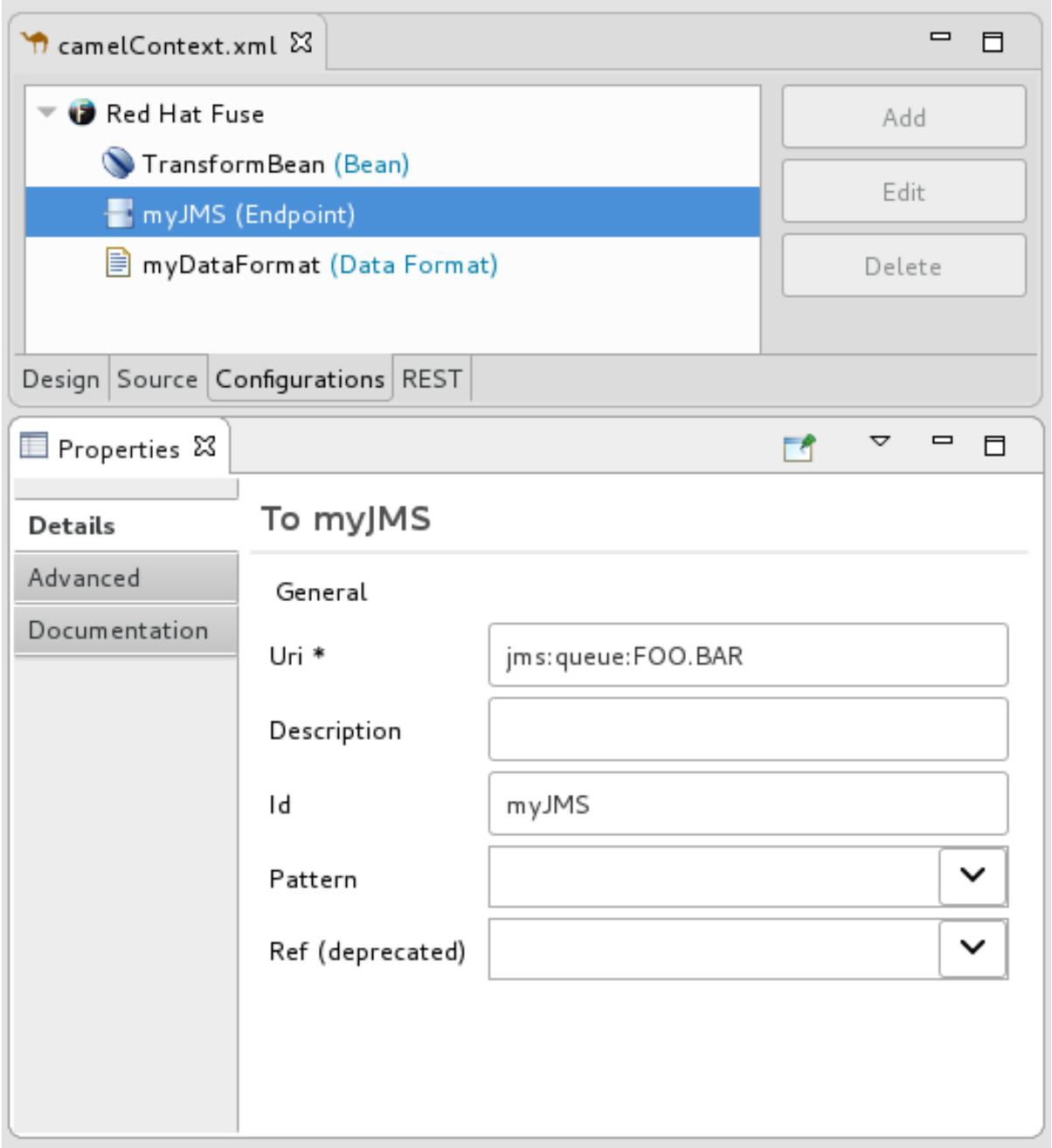
手順は、ルーティングコンテキストに追加したエンドポイント、データ形式、または Bean のプロパティーを変更する場合でも同じです。

通常、グローバル要素の ID は変更しません。グローバル要素が実行中のルートですでに使用されている場合、ID を変更すると、グローバル要素への参照が壊れることがあります。

1. **Configurations** タブで、編集するグローバル要素を選択します。
   たとえば、「グローバルエンドポイントの追加」で追加したエンドポイント **myJMS** を編集するには、以下のように選択します。

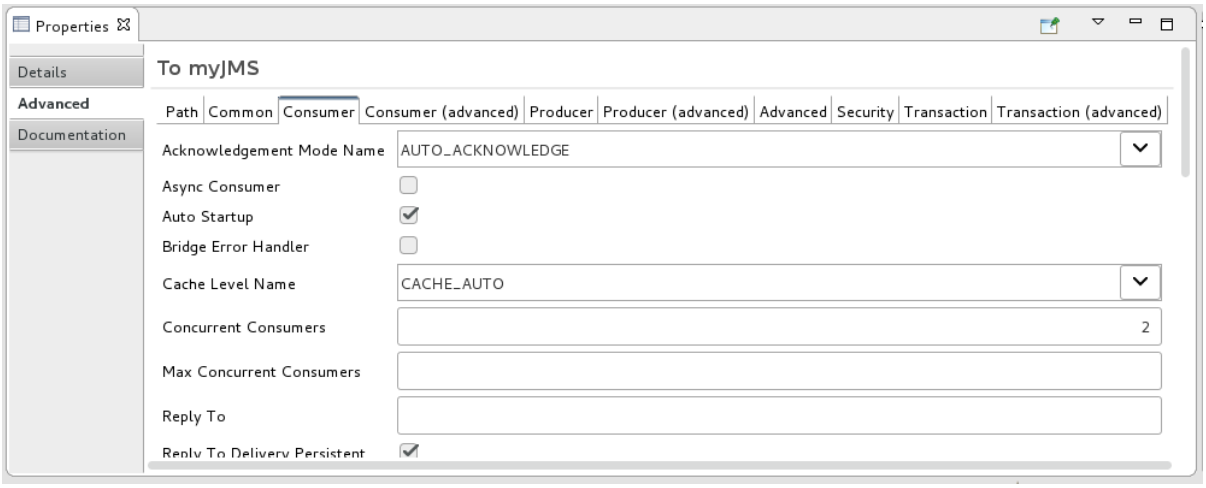


2. **Edit** をクリックします。



Properties ビューで、必要に応じて要素のプロパティーを変更します。

3. たとえば、**Advanced → Consumer** タブを開き、**Concurrent Consumers** の値を **2** に変更します。



4. ルートエディターで、**Source** タブをクリックして、ツールがプロパティー **concurrentConsumers=2** をルーティングコンテキストに追加したことを確認します。

5. 完了したら、メニューバーで **File → Save** を選択して変更を保存します。

2.7. ルートエディターの設定

概要

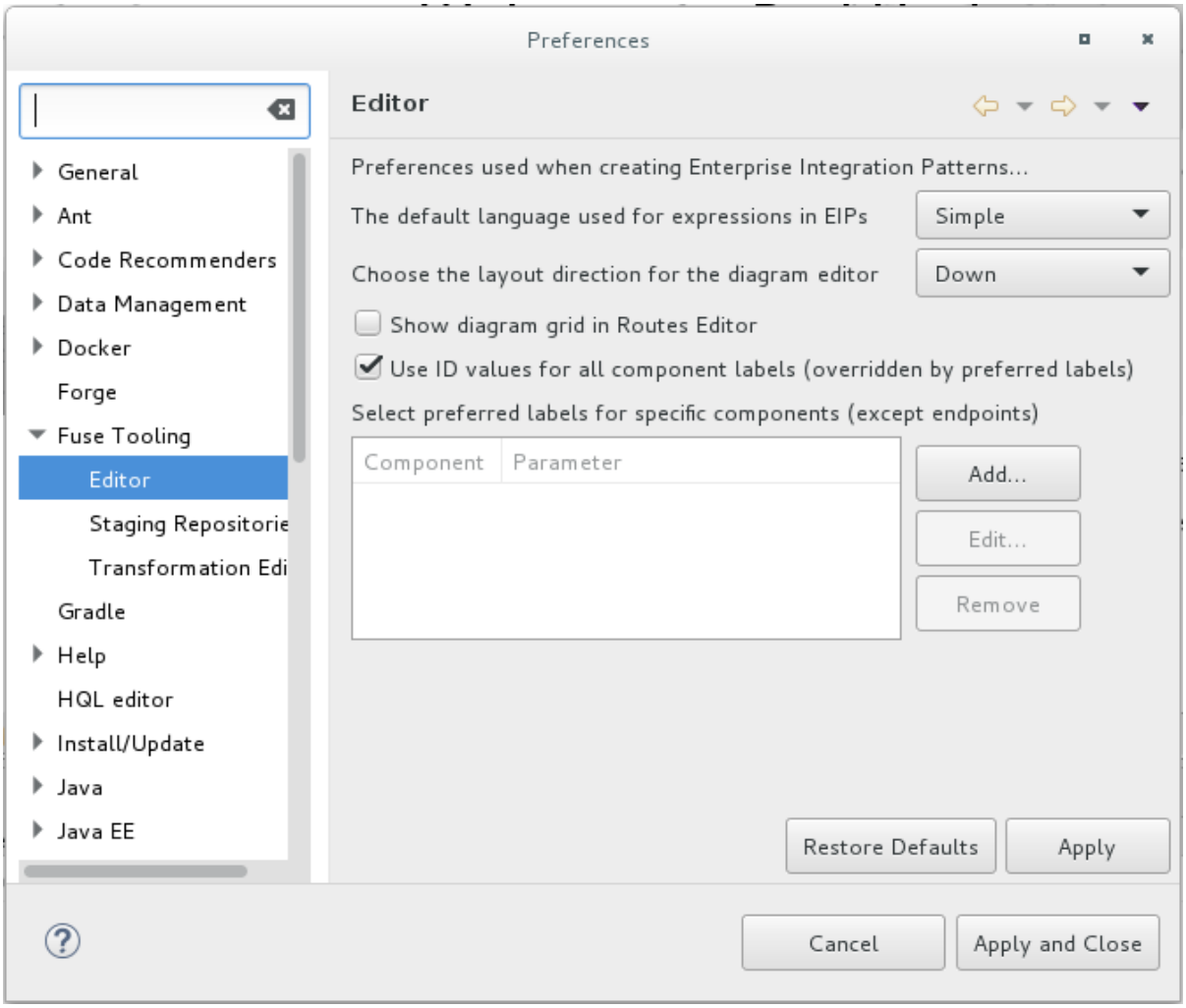
Fuse 設定を使用して、ルートエディターの動作とユーザーインターフェイスのオプションを指定できます。

- エンタープライズ統合パターン (EIP) の式に使用するデフォルト言語
- ルートを作成するときにデザインキャンバス上でパターンが流れる方向 (右または下)
- デザインキャンバスがキャンバスの背景にグリッドオーバーレイを表示するかどうか
- デザインキャンバス上のノードにラベルを付ける方法

手順

ルートエディターを設定するには、以下を実行します。

1. **Editor** 設定ウィンドウを開きます。
   - Linux および Windows マシンでは、**Windows → Preferences → Fuse Tooling → Editor** を選択します。



2. エンタープライズ統合パターン (EIP) コンポーネントの式に使用するデフォルトの言語を選択するには、ドロップダウンリストから言語を選択します。デフォルトは **Simple** です。

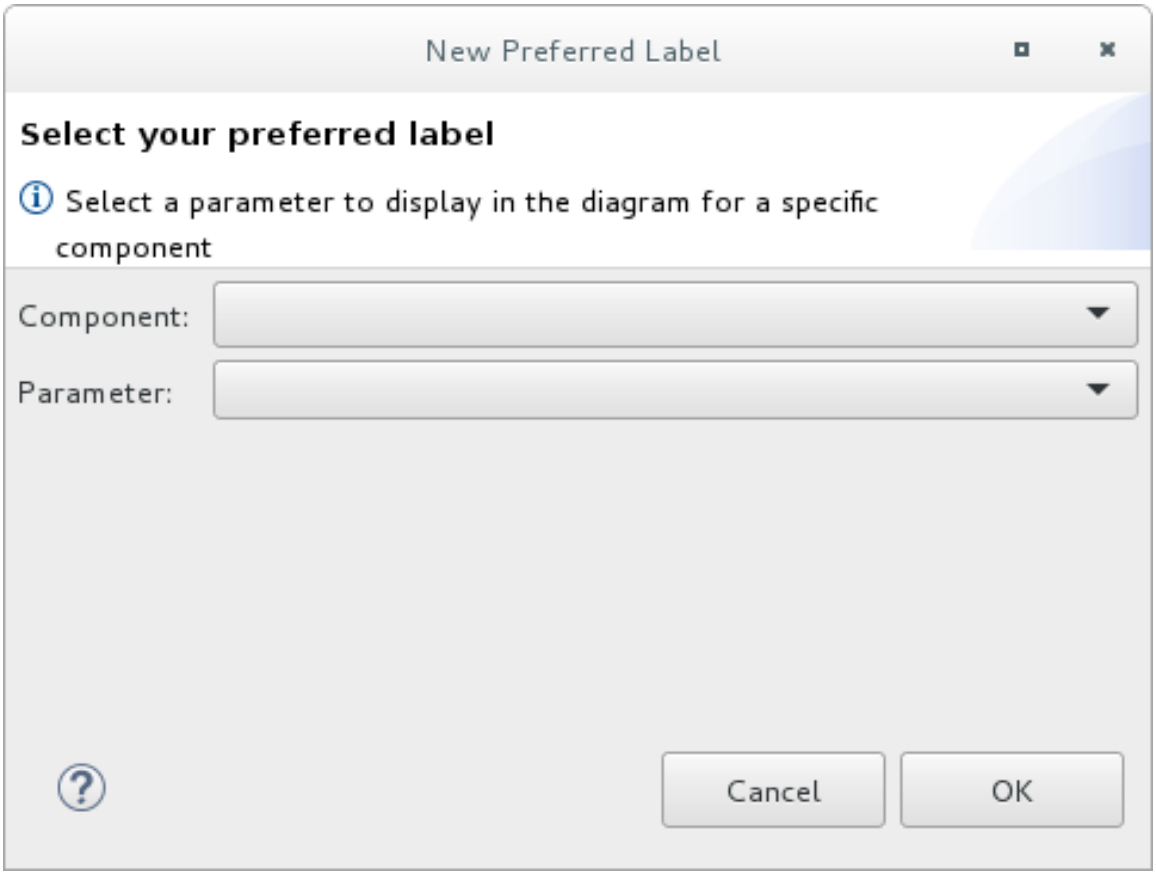
3. ルートエディターでルート内のパターンを整列させる方向を指定するには、**Down** または **Right** を選択します。デフォルトは **Down** です。

4. キャンバスの背景でグリッドオーバーレイの表示を有効または無効にするには、**Show diagram grid in Routes Editor** の横のチェックボックスをオンにします。デフォルトは **enabled** です。

5. ルートエディターの **Design** タブでコンポーネント ID をラベルとして使用することを有効または無効にするには、**Use ID values for component labels**の横のチェックボックスをオンにします。デフォルトは **disabled** です。
このオプションをチェックし、コンポーネントに使用するラベルを指定すると (手順 6 を参照)、ID 値の代わりに指定したラベルがそのコンポーネントに使用されます。

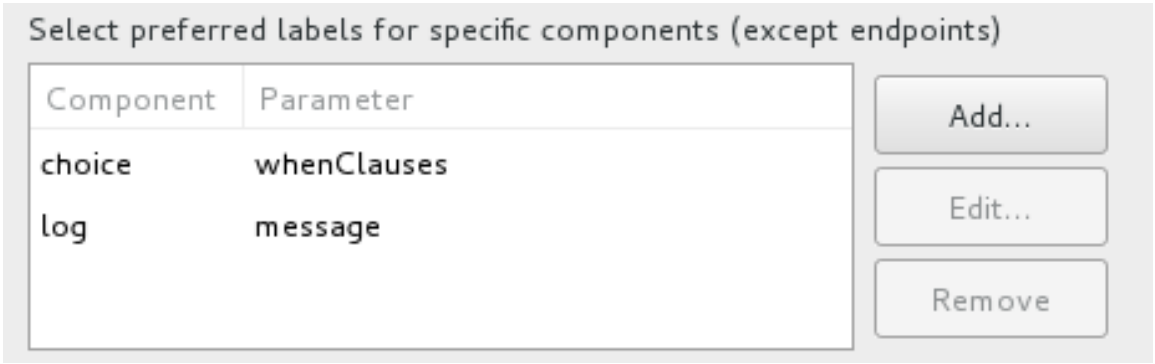
6. ルートエディターの **Design** タブで、コンポーネント (**File** ノードなどのエンドポイントを除く) のラベルとしてパラメーターを使用するには、次の手順に従います。

   a. **Preferred labels** セクションで、**Add** をクリックします。**New Preferred Label** ダイアログボックスが開きます。



b. **Component** を選択してから、コンポーネントのラベルとして使用する **Parameter** を選択します。

c. **OK** をクリックします。コンポーネントとパラメーターのペアは、エディターの設定ウィンドウに一覧表示されます。

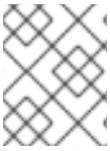


オプションで、コンポーネントラベルを **編集** および **削除** できます。

注記

Use ID values for component labelsオプションをオンにすると、**Preferred labels** セクションにリストされているコンポーネントを除くすべてのコンポーネントに適用されます。

7. **Apply and Close** をクリックして変更を **Editor** 設定に適用し、**Preferences** ウィンドウを閉じます。



注記

Editor 設定ダイアログボックスに戻って **Restore Defaults** をクリックすると、いつでもルートエディターの元のデフォルトに戻すことができます。

第3章 REST DSL コンポーネントの表示と編集

Apache Camel は、REST サービスを定義するために複数のアプローチをサポートします。特に、Apache Camel は REST DSL (Domain Specific Language) を提供します。これは、REST コンポーネントを抽象化でき、OpenAPI 2.0 specification とも統合できるシンプルながらも強力な API です。OpenAPI (以前の Swagger) は、ベンダーに依存しない、移植可能な API サービスのオープン記述形式です。

Camel Rest DSL の使用に関する詳細は、Apache Camel 開発ガイド の「REST サービスの定義」の章を参照してください。

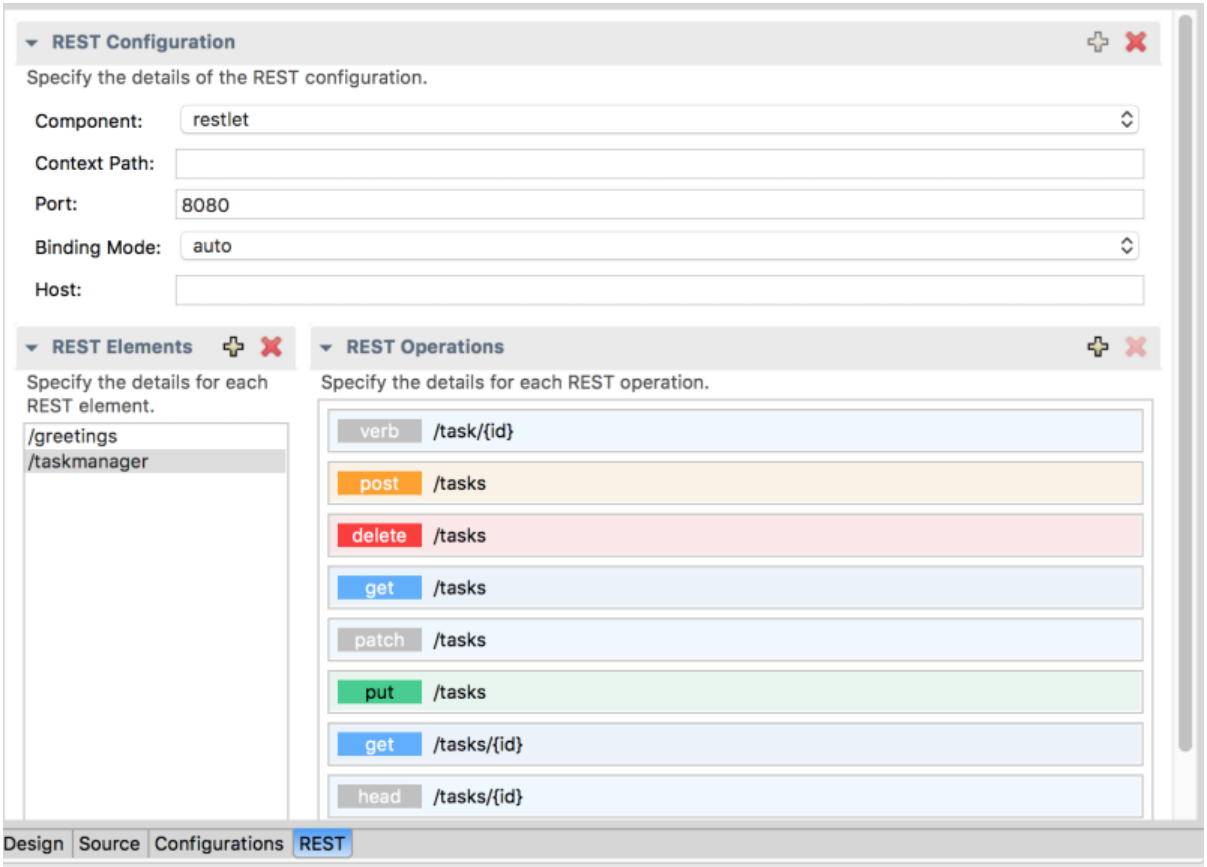
Fuse Tooling を使用すると、Camel コンテキストファイルにある Rest DSL コンポーネントを表示および編集できます。

https://access.redhat.com/articles/4296981 で説明されているように、REST API を OpenAPI クライアントに公開するように Fuse Integration プロジェクトを設定することもできます。

3.1. REST DSL コンポーネントのグラフィック表現の表示

Camel コンテキストファイルの REST DSL コンポーネントをグラフィカルモードで表示するには、以下を実行します。

1. ルートエディターで Camel コンテキストファイルを開きます。
2. **REST** タブをクリックして、Rest DSL コンポーネントを表示します。

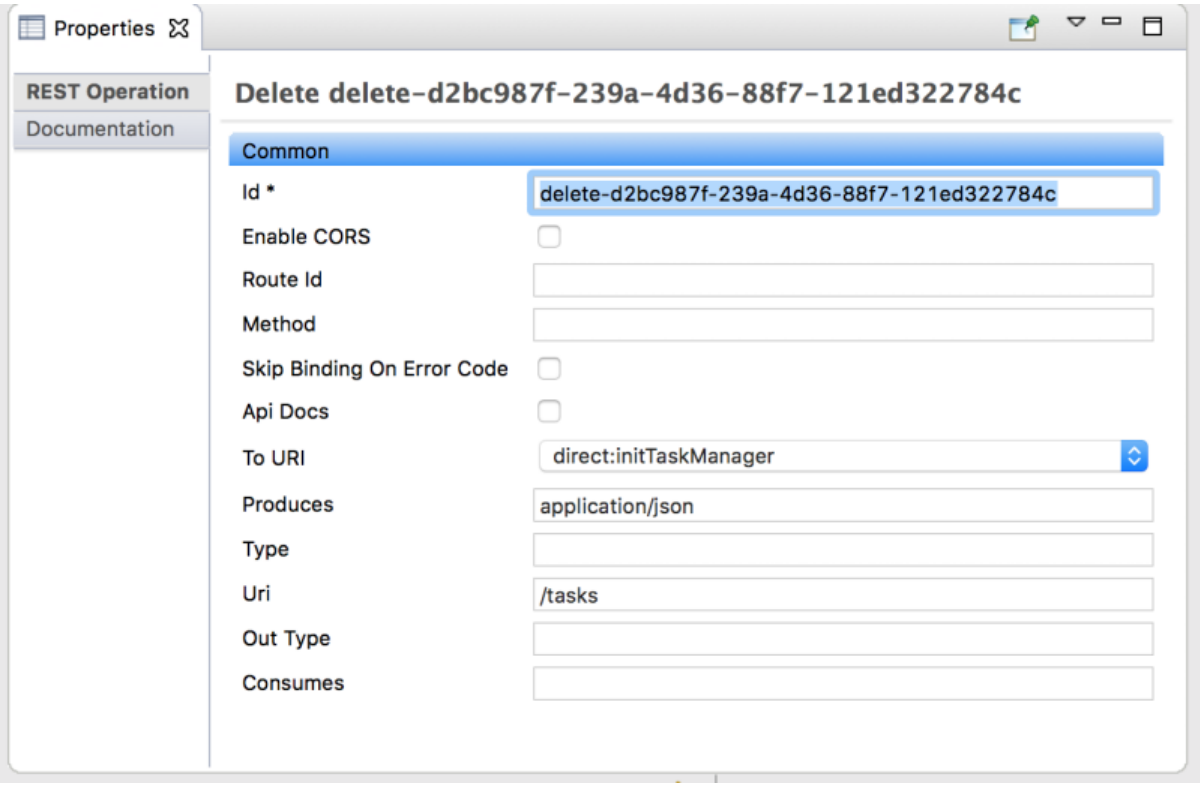


REST Configuration セクションには、次の設定オプションが表示されます。

- **Component** – REST トランスポートに使用する Camel コンポーネント。

- **Context Path** – REST サービスの主要なコンテキストパス。このオプションは、サーブレットのように、Web アプリケーションがコンテキストパスを使用してデプロイされるコンポーネントに使用できます。
- **Port** – REST サービスを公開するポート番号。
- **Binding Mode** – JSON または XML 形式のメッセージのバインディングモード。可能な値は、**off** (デフォルト)、**auto**、**json**、**xml**、または **json_xml** です。
- **Host** – REST サービスの公開に使用するホスト名。

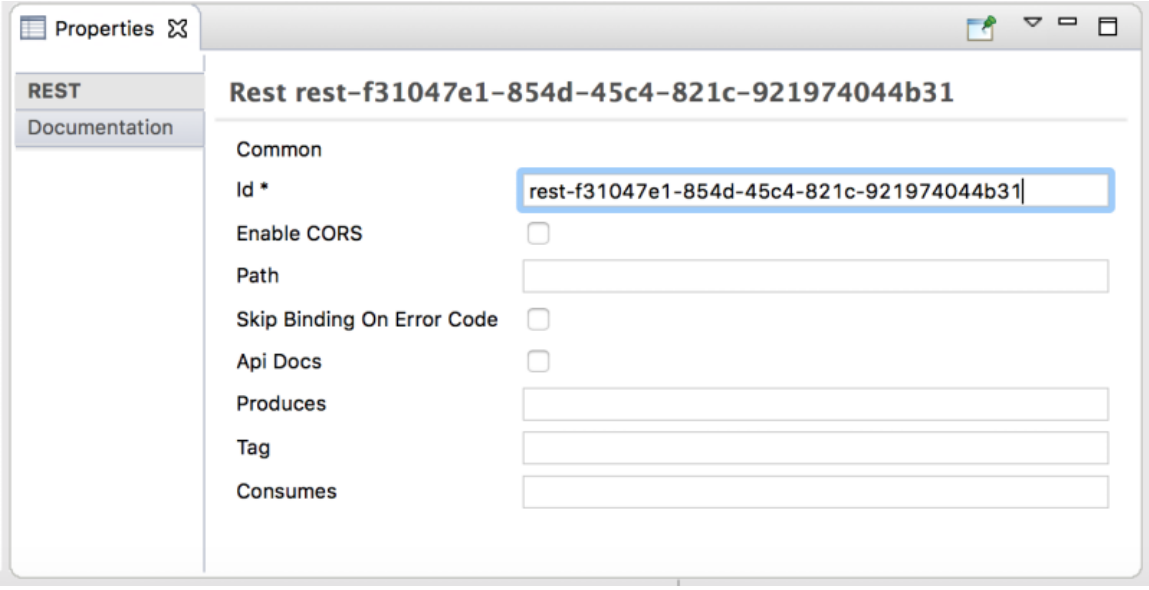
3. REST 要素をクリックして、**REST Operations** セクションで関連する操作 (例: **GET**、**POST**、**PUT**、および **DELETE**) を表示します。
4. REST 要素または REST 操作をクリックして、**Properties** ビューにそのプロパティーを表示します。



3.2. グラフィカルビューでの REST DSL コンポーネントの編集

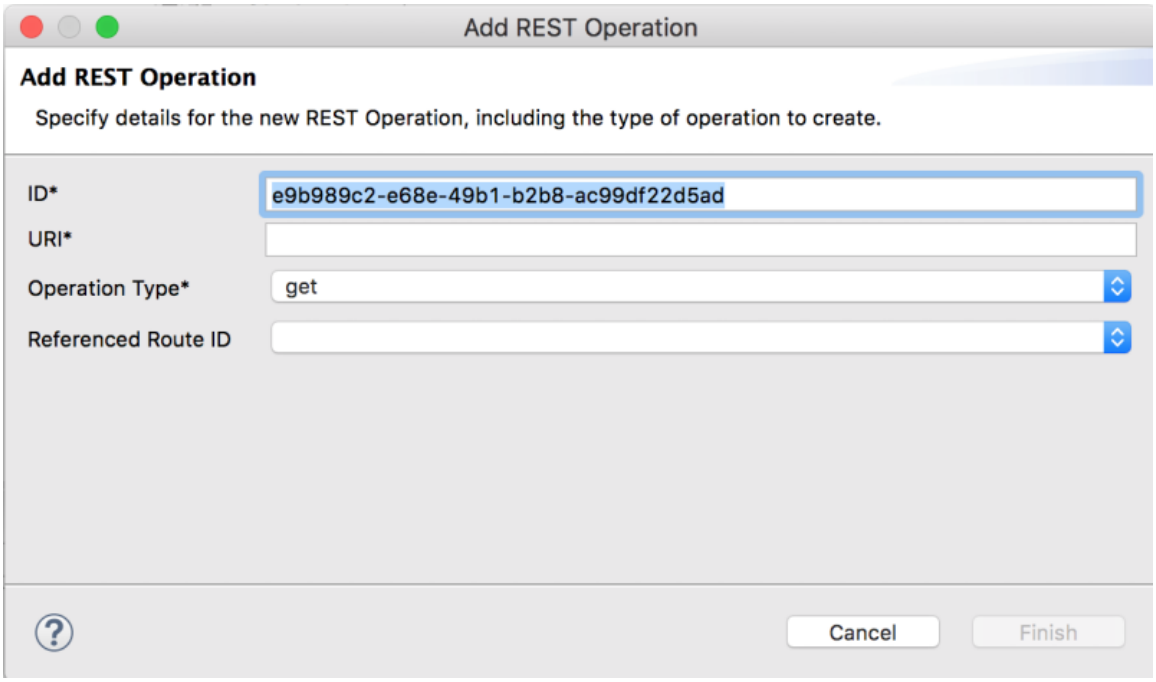
REST タブで、プロジェクトの Camel コンテキストファイルの REST 要素を 追加、編集、または削除できます。

- 新しい REST 要素を追加するには、以下を実行します。
  1. **REST elements** セクションで、+ ボタンをクリックします。REST 要素が **REST elements** のリストに追加されます。
  2. **Properties** ビューで、REST 要素のプロパティーを編集します。



- REST オペレーションを REST 要素に追加するには、以下を実行します。

  1. **REST elements** のリストで、REST 要素を選択します。
  2. **REST operations** セクションで、+ ボタンをクリックします。
     Add REST Operation ダイアログボックスが開きます。



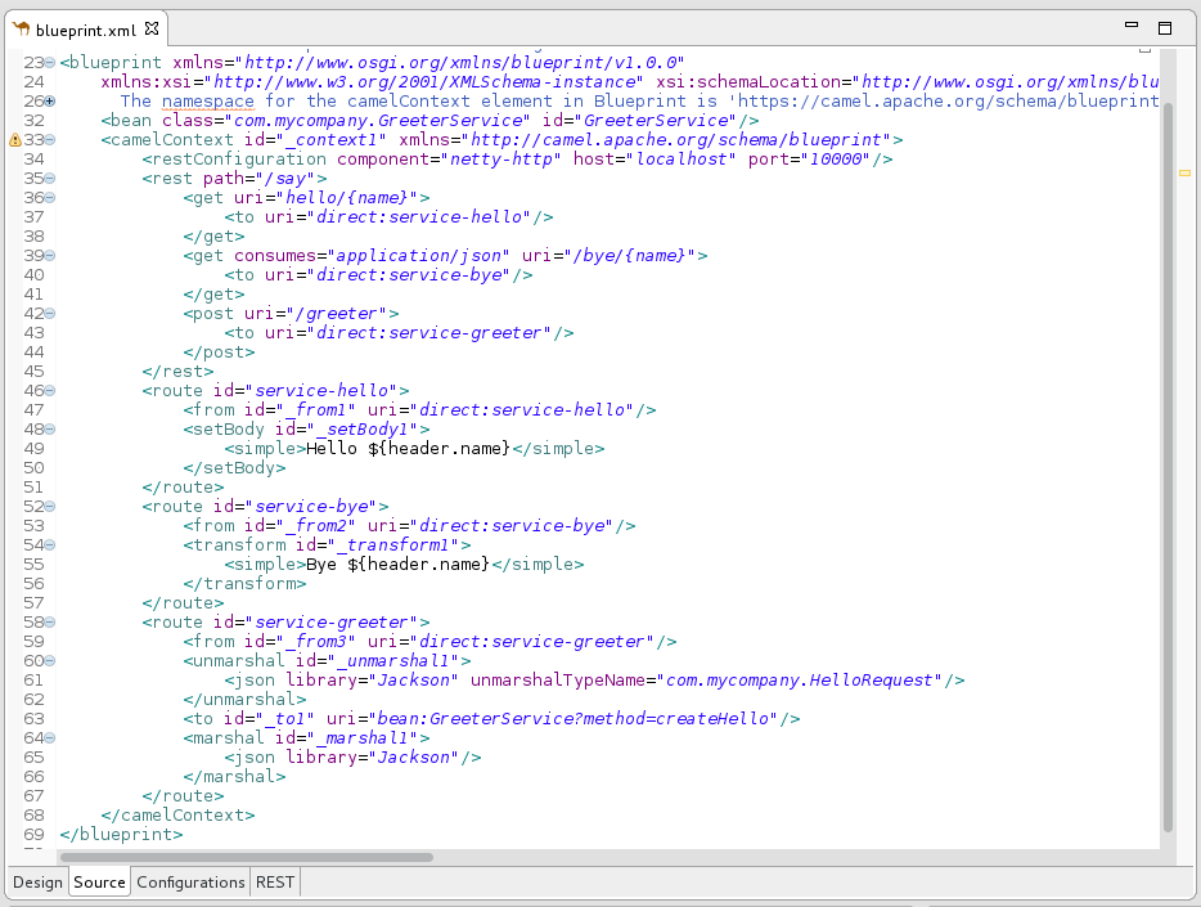
  3. **ID**、**URI**、および **Operation Type** を指定します。必要に応じて、**Referenced Route ID** を選択します。
  4. **Finish** をクリックします。新しいオペレーションは、選択した REST 要素の REST オペレーションリストに表示されます。

- REST 要素またはオペレーションを編集するには、**REST** タブでそれらを選択し、**Properties** タブでそのプロパティー値を編集します。
- 選択した REST 要素またはオペレーションを削除するには、**x** ボタンをクリックします。

3.3. REST DSL ソースコードの表示と編集

Source タブで、Rest DSL コンポーネントを表示および編集することもできます。

1. ルートエディターで Camel コンテキストファイルを開きます。
2. ルートエディターの **Source** タブをクリックして、コードを編集します。



3. 必要に応じて、**REST** タブをクリックして、グラフィカルビューで変更を確認します。
4. 変更を保存するには、**File → Save** を選択します。

第4章 新しい APACHE CAMEL JUNIT テストケースの作成

概要

ルートをテストする一般的な方法は、JUnit を使用することです。デザインタイムツールには、ルートの JUnit テストケースの作成を簡素化するウィザードが含まれています。ウィザードは、指定したエンドポイントを使用して、テストの開始点コードと設定を生成します。

注記

boilerplate JUnit テストケースを作成したら、それを変更して、作成または変更したルートに固有の期待値とアサーションを追加する必要があります。これにより、テストはルートに対して有効になります。

前提条件

新しい JUnit テストケースを作成する前に、予備的なタスクを実行する必要があります。

- 既存の JUnit テストケースを置き換える場合は、新しいテストケースを作成する前にそれを削除する必要があります。「既存の JUnit テストケースの削除」を参照してください。
- テストケースのないプロジェクトに新しい JUnit テストケースを作成する場合には、まずビルドパスに含まれるテストケースの **project_root/src/test/java** フォルダーを作成する必要があります。「**src/test/java** フォルダーの作成およびビルドパスへの追加」を参照してください。

既存の JUNIT テストケースの削除

1. **Project Explorer** ビューでプロジェクトのルートノードを展開し、**<root_project>/src/test/java** フォルダーを表示します。
2. **/src/test/java** フォルダーで JUnit テストケースファイルを見つけます。
   プロジェクトがベースとする DSL に応じて、JUnit テストケースファイルには **BlueprintXmlTest.java** または **CamelContextXmlTest.java** という名前が付けられます。
3. JUnit テストケースの **.java** ファイルを右クリックしてコンテキストメニューを開き、**Delete** を選択します。
   JUnit テストケースの **.java** ファイルは、**Project Explorer** ビューから消えます。

   これで 新しい JUnit テストケースの作成 が可能になります。

SRC/TEST/JAVA フォルダーの作成およびビルドパスへの追加

1. **Project Explorer** ビューで、プロジェクトのルートを右クリックしてコンテキストメニューを開きます。
2. **New → Folder** を選択して **Create a new folder resource**ウィザードを開きます。
3. ウィザードのプロジェクトツリーペインで、プロジェクトのルートノードを展開し、**src** フォルダーを選択します。
   <project_root>/src が **Enter or select the parent folder**フィールドに表示されるのを確認してください。

4. **Folder name** に **/test/java** を入力します。このフォルダーには、作成した新しい JUnit テストケースが格納されます。

5. **Finish** をクリックします。
Project Explorer ビューでは、新しい **src/test/java** フォルダーが **src/main/resources** フォルダーの下に表示されます。このフォルダーがクラスパス上にあることを確認するには、コンテキストメニューを開き、**Build Path** を選択します。**Remove from Build Path**がメニューオプションである場合は、**src/test/java** フォルダーがクラスパスにあります。

   これで 新しい JUnit テストケースの作成 が可能になります。

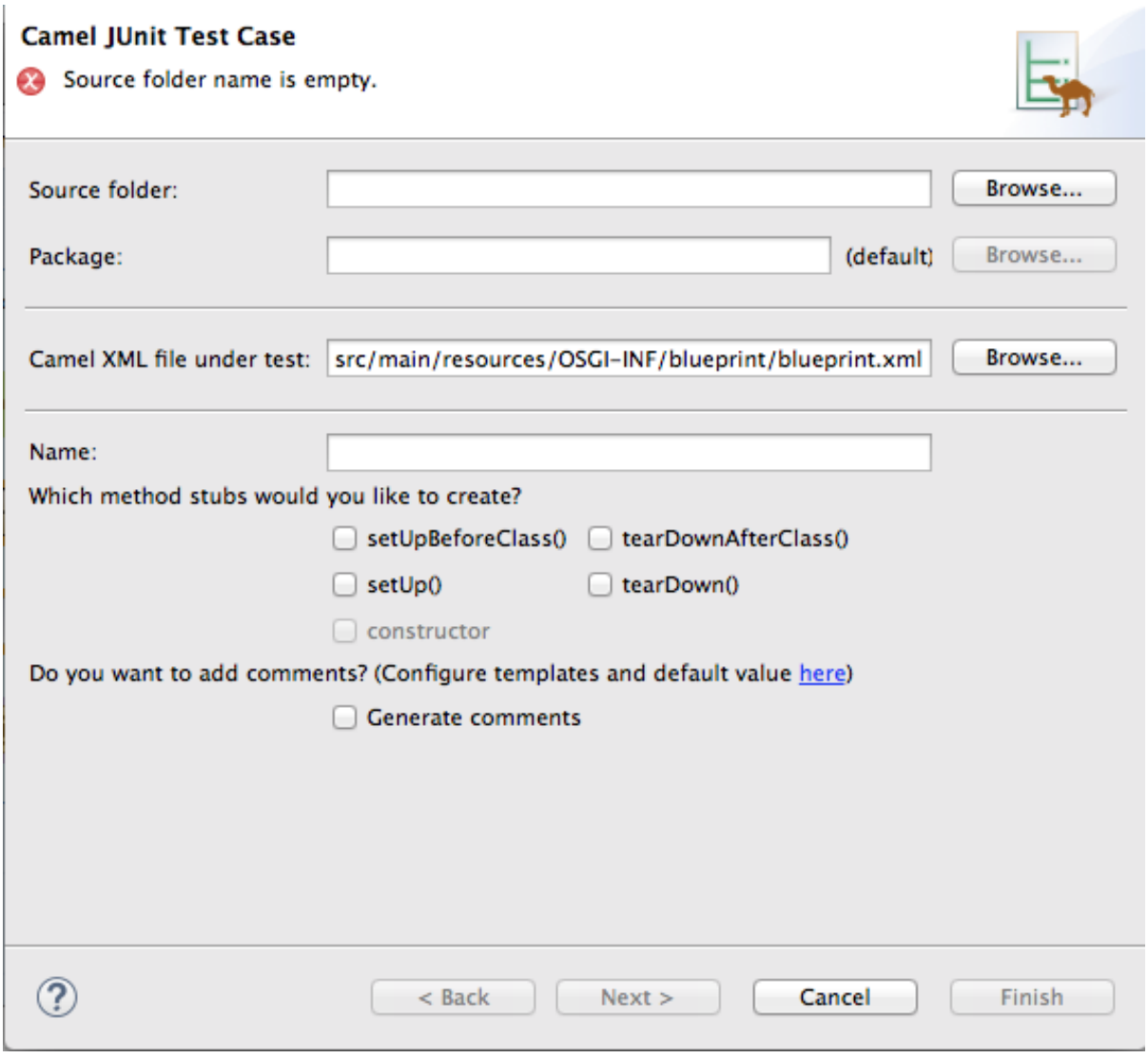
JUNIT テストケースの作成

ルートの新しい JUnit テストケースを作成するには、以下を実行します。

1. **Project Explorer** ビューで、プロジェクトのルーティングコンテキスト **.xml** ファイルを選択します。

2. 右クリックしてコンテキストメニューを開き、**New → Camel Test Case** を選択して、図 4.1「新しい Camel JUnit テストケースウィザード」 で示すとおり **New Camel JUnit Test Case** ウィザードを開きます。

図4.1新しい Camel JUnit テストケースウィザード



または、メニューバーから **File** → **New** → **Other** > **Fuse** > **Camel Test Case** を選択して、ウィザードを開くこともできます。

3. **Source folder** で、テストケースのソースコードのデフォルトの場所を受け入れるか、別の場所を入力します。
Browse... をクリックして場所を検索できます。

4. **Package** で、生成されたテストコードのデフォルトのパッケージ名を受け入れるか、別のパッケージ名を入力します。
Browse... をクリックしてパッケージを検索できます。

5. **Camel XML file under test**で、テストするルートを含むルーティングコンテキストファイルのデフォルトのパス名を受け入れるか、別のパス名を入力します。
Browse... をクリックしてコンテキストファイルを検索できます。

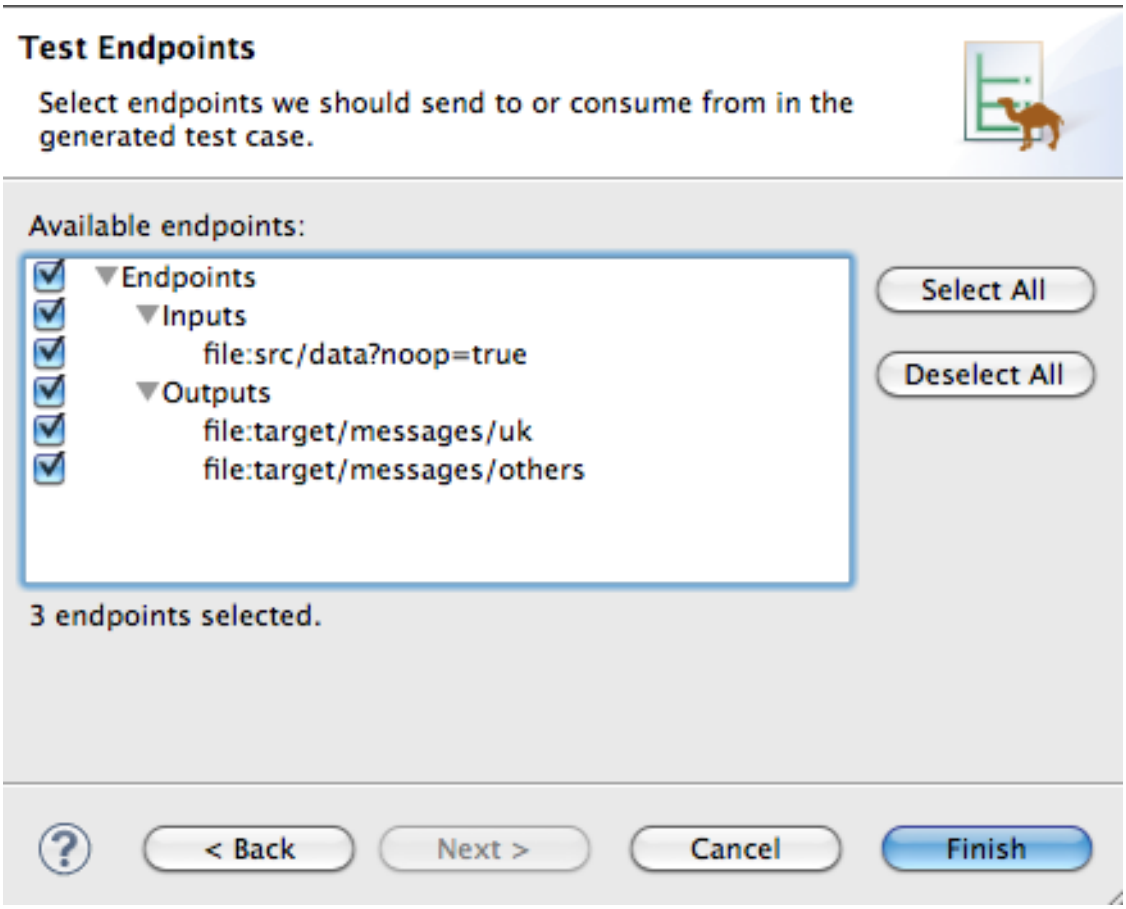
6. **Name** に、生成されたテストクラスのデフォルトの名前を受け入れるか、別の名前を入力します。

7. 生成されたコードに含めるメソッドスタブを選択します。

8. 生成されたコードにデフォルトの生成されたコメントを含める場合は、**Generate comments** ボックスをオンにします。

9. Next > をクリックし、**Test Endpoints** ページを開きます。例えば、図4.2「新しい Camel JUnit のテストケースページ」は、選択されたルートの入力ファイルと出力ファイルのエンドポイントを示しています。

図4.2 新しい Camel JUnit のテストケースページ



10. **Available endpoints** で、テストするエンドポイントを選択します。選択したエンドポイントの横にあるチェックボックスをクリックして、選択を解除します。
11. Finish をクリックします。

注記

プロンプトが表示されたら、ビルドパスに JUnit を追加します。

テストのアーティファクトはプロジェクトに追加され、**Project Explorer** ビューで **src/test/java** の下に表示されます。テストケースを実装するクラスが Java エディターで開きます。

第5章 RED HAT FUSETOOLING 内でのルートの実行

ツールを使用してルートを実行するには、次の 2 つの方法があります。

- 「ローカル Camel コンテキストとしてルートを実行」
- 「Maven を使用したルートの実行」

5.1. ローカル CAMEL コンテキストとしてルートを実行

概要

Apache Camel ルートを実行する最も簡単な方法は、**Local Camel Context** として実行する方法です。この方法では、**Project Explorer** ビューのコンテキストメニューから直接ルートを起動できます。コンテキストメニューからルートを実行すると、ツールによってランタイムプロファイルが自動的に作成されます。ルートを実行するためのカスタムランタイムプロファイルを作成することもできます。

ルートは、コマンドラインから直接呼び出されたかのように実行され、Apache Camel の埋め込み Spring コンテナーを使用します。ランタイムプロファイルを編集して、いくつかのランタイムパラメーターを設定できます。

手順

ローカル Camel コンテキストとしてルートを実行するには、以下の手順に従います。

1. **Project Explorer** ビューで、ルーティングコンテキストファイルを選択します。
2. それを右クリックしてコンテキストメニューを開き、**Run As → Local Camel Context** を選択します。

注記

Local Camel Context (without tests) を選択すると、ツールは検証テストを実行せずにプロジェクトを実行するように指示されます。こちらの方が速い場合があります。

結果

Console ビューには、ルートの実行から生成された出力が表示されます。

関連トピック

- 「ローカル Camel コンテキストランタイムプロファイルの編集」

5.2. MAVEN を使用したルートの実行

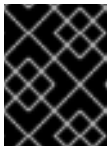
概要

ルートを含むプロジェクトが Maven プロジェクトの場合、m2e プラグインを使用してルートを実行できます。このオプションを使用すると、ルートが実行される前に Maven の目標を実行できます。

手順

Maven を使用してルートを実行するには、以下を行います。

1. **Project Explorer** ビューで、プロジェクトのルートを選択します。
2. それを右クリックしてコンテキストメニューを開き、**Run As → Maven build** を選択します。
   a. Maven を使用してプロジェクトを初めて実行すると、**Edit Configuration and launch** エディターが開き、Maven ランタイムプロファイルを作成できます。
   ランタイムプロファイルを作成するには、**Maven** タブで次の手順を実行します。
      i. Apache Camel プロジェクトのルートディレクトリーが **Base directory:** フィールドに表示されていることを確認します。
      たとえば、Linux では、プロジェクトのルートは ~/**workspace/simple-router** のようになります。
      ii. **Goals:** フィールドに **camel:run** を入力します。



重要

Java DSL を使用してプロジェクトを作成した場合は、**Goals:** フィールドに **exec:java** を入力します。

      iii. **Apply**、**Run** の順にクリックします。
   b. 実行の間で変更しない限り、後続の Maven 実行ではこのプロファイルが使用されます。

結果

Console ビューには、Maven 実行からの出力が表示されます。

関連トピック

- 「Maven ランタイムプロファイルの編集」

5.3. ランタイムプロファイルの操作

Red Hat Fuse Tooling は、各プロジェクトのランタイム環境に関する情報を **ランタイムプロファイル** に保存します。ランタイムプロファイルは、呼び出す Maven の目標、使用する Java ランタイム環境、設定する必要のあるシステム変数などの情報を追跡します。プロジェクトには、複数のランタイムプロファイルを含めることができます。

5.3.1. ローカル Camel コンテキストランタイムプロファイルの編集

概要

Local Camel Context ランタイムプロファイルは、ルートを実行するために Apache Camel を呼び出す方法を設定します。**Local Camel Context** ランタイムプロファイルは、ルートが定義されているコンテキストファイルの名前、呼び出す **main** の名前、JVM に渡すコマンドラインオプション、使用する JRE、使用するクラスパス、設定する必要のある環境変数、その他の情報などを保存します。

Local Camel Context ランタイムプロファイルのランタイム設定エディターには、次のタブが含まれています。

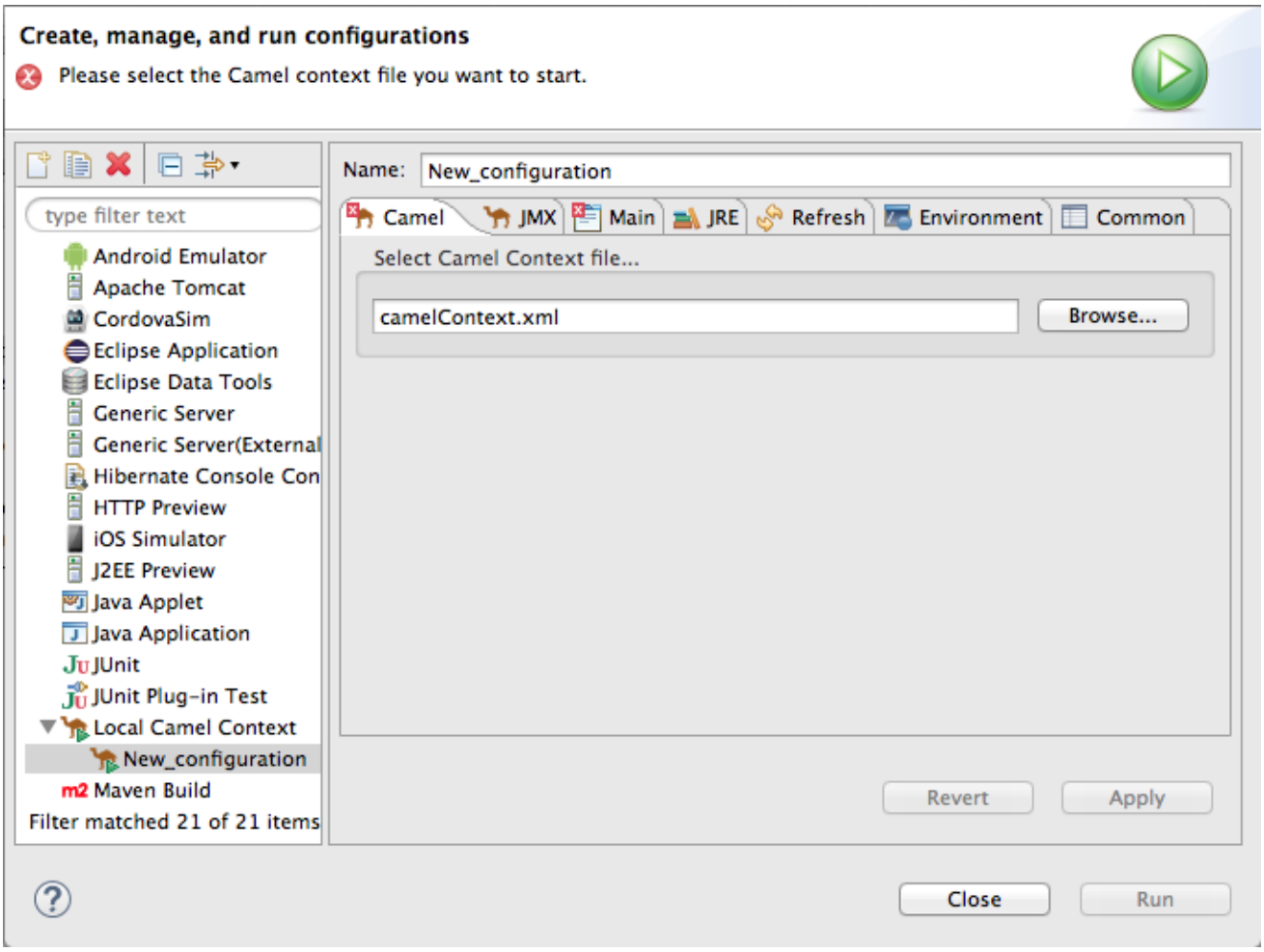
- **Camel Context File** – 新しい設定の名前とルートを含むルーティングコンテキストファイルのフルパスを指定します。
- **JMX** – JMX URI と、それにアクセスするために使用するユーザー名とパスワード (オプション) を含む、JMX 接続の詳細を指定します。
- **Main** – プロジェクトのベースディレクトリーの完全修飾名、ベースディレクトリーを見つけるためのいくつかのオプション、ルートを実行する前に実行する必要のある目標、使用する Maven ランタイムのバージョンを指定します。
- **JRE** – JVM の起動時に使用する JRE およびコマンドライン引数を指定します。
- **Refresh** – 実行が終了した後に Maven がプロジェクトのリソースファイルを更新する方法を指定します。
- **Environment** – 設定する必要のある環境変数を指定します。
- **Common** – プロファイルの保存方法と出力の表示方法を指定します。

Apache Camel ルートが **Local Camel Context** として初めて実行されるとき、Red Hat Fuse Tooling はルーティングコンテキストファイル用に、編集を必要としないデフォルトのランタイムプロファイルを作成します。

Local Camel Context のランタイム設定エディターへのアクセス

1. **Project Explorer** ビューで、カスタムランタイムプロファイルを編集または作成する Camel コンテキストファイルを選択します。
2. それを右クリックしてコンテキストメニューを開き、**Run As → Run Configurations** を選択して **Run Configurations** ダイアログボックスを開きます。
3. **コンテキスト選択** ペインで、**Local Camel Context** を選択してから、**コンテキスト選択** ペインの左上にある  をクリックします。
4. **Name** フィールドに、ランタイムプロファイルの新しい名前を入力します。

図5.1 Loca Camel Context のランタイム設定エディター



camel コンテキストファイルの設定

Camel Context File タブには 1 つのフィールド **Select Camel Context file...** があります。ルート定義を含むルーティングコンテキストファイルへのフルパスを入力します。

Browse ボタンは、**Open Resource** ダイアログボックスにアクセスします。これにより、ターゲットルーティングコンテキストファイルを簡単に見つけることができます。このダイアログボックスは、Apache Camel ルートを含むファイルを検索するように事前設定されています。

コマンドラインオプションの変更

デフォルトでは、JVM には以下のコマンドラインオプションのみ渡されます。

> -fa **context-file**

カスタムメインクラスを使用している場合は、別のオプションを渡す必要がある場合があります。そうするには、**Main** タブで **Add** ボタンをクリックして、パラメーターの名前と値を入力します。**Add Parameter** ダイアログボックスの **Variables...** ボタンをクリックして、選択可能な変数のリストを表示できます。

JVM 固有の引数を追加または変更するには、**JRE** タブの **VM arguments** フィールドを編集します。

出力の送信先の変更

デフォルトでは、ルートの実行から生成された出力は **Console** ビューに送信されます。ただし、これをファイルにリダイレクトすることもできます。

出力をファイルにリダイレクトするには、以下を実行します。

1. **Common** タブを選択します。
2. **Standard Input and Output** ペインで、**Output File:** フィールドの横にあるチェックボックスをクリックし、出力送信先のファイルへのパスを入力します。
Workspace ボタン、**File System** ボタン、および **Variables** ボタンを使用すると、出力ファイルへのパスを簡単に構築できます。

関連トピック

- 「ローカル Camel コンテキストとしてルートを実行」

5.3.2. Maven ランタイムプロファイルの編集

概要

Maven ランタイムプロファイルは、Maven が Apache Camel を呼び出す方法を設定します。Maven ランタイムプロファイルには、実行する Maven 目標、使用する Maven プロファイル、使用する Maven のバージョン、使用する JRE、使用するクラスパス、設定する必要のある環境変数、およびその他のいくつかの情報が格納されています。

重要

Maven を使用して Apache Camel ルートを初めて実行するときは、そのデフォルトランタイムプロファイルを作成する必要があります。

Fuse ランタイムプロファイルのランタイム設定エディターには、次のタブが含まれています。

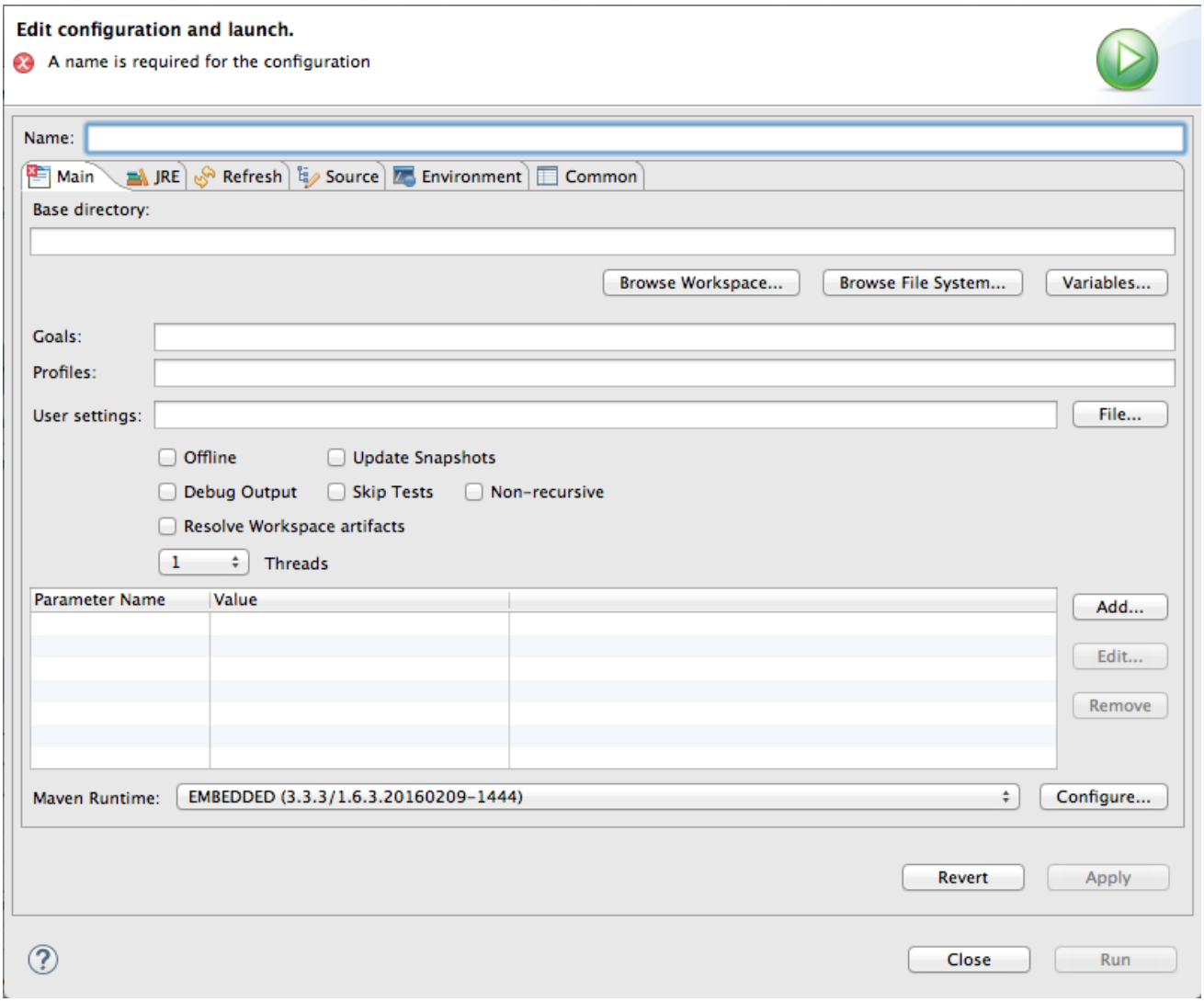
- **Main** – 新しい設定の名前、プロジェクトのベースディレクトリーの完全修飾名、ベースディレクトリーを見つけるためのいくつかのオプション、ルートを実行する前に実行する必要のある目標、使用する Maven ランタイムのバージョンを指定します。
- **JRE** – JVM の起動時に使用する JRE およびコマンドライン引数を指定します。
- **Refresh** – 実行が終了した後に Maven がプロジェクトのリソースファイルを更新する方法を指定します。
- **Source** – プロジェクトに必要な追加のソースの場所を指定します。
- **Environment** – 設定する必要のある環境変数を指定します。
- **Common** – プロファイルの保存方法と出力の表示方法を指定します。

Maven ランタイム設定エディターへのアクセス

1. **Project Explorer** ビューで、カスタムランタイムプロファイルを編集または作成するプロジェクトのルートを選択します。
2. それを右クリックしてコンテキストメニューを開き、**Run As → Run Configurations** を選択して **Run Configurations** ダイアログボックスを開きます。

3. **コンテキスト選択** ペインで、**Maven Build** を選択してから、**コンテキスト選択** ペインの左上にある  をクリックします。

図5.2 Maven のランタイム設定エディター



Maven ゴールの変更

ルートを実行するときに最も一般的に使用されるゴールは、**camel:run** です。これは、独自の JVM で実行されている Spring コンテナーにルートをロードします。

Apache Camel プラグインは、Maven が使用するのと同じ JVM に Spring コンテナーをロードする **camel:embedded** ゴールもサポートしています。これの利点は、ルートのブートストラップが速くなることです。

Java DSL に基づくプロジェクトは、**exec:java** ゴールを使用します。

POM に他のゴールが含まれる場合、**Main** タブの **Maven Runtime** フィールドの横にある **Configure...** ボタンをクリックして、使用する Maven ゴールを変更できます。**Installations** ダイアログボックスで、**Global settings for <selected_runtime> installation** フィールドを編集します。

Maven バージョンの変更

デフォルトでは、Red Hat Fuse Tooling for Eclipse は Eclipse に組み込まれている m2e を使用します。別の Maven バージョンを使用する場合、または開発マシンに新しいバージョンをインストールする場合は、**Main** タブの **Maven Runtime** ドロップダウンメニューから選択できます。

出力送信先の変更

デフォルトでは、ルート実行からの出力は **Console** ビューに送信されます。ただし、これをファイルにリダイレクトすることもできます。

出力をファイルにリダイレクトするには、以下を実行します。

1. **Common** タブを選択します。
2. **Output File:** フィールドの横にあるチェックボックスをクリックして、出力を送信するファイルへのパスを入力します。
   Workspace ボタン、**File System** ボタン、および **Variables** ボタンを使用すると、出力ファイルへのパスを簡単に構築できます。

関連トピック

- 「Maven を使用したルートの実行」

第6章 FUSE ON OPENSHIFT の使用

Fuse on OpenShift (Fuse Integration Services 7.0 以降の名称) は、Fuse アプリケーションを OpenShift Container Platform にデプロイすることができます。

重要

Fuse Integration プロジェクト (Fuse on OpenShift プロジェクト) の場合、Fuse Tooling では Red Hat Container Development Kit (CDK) v3.**x** をインストールする必要があります。手順については、Getting Started Guide を参照してください。このガイドで指定されている前提条件に加えて、Red Hat アカウントを持っていない場合は、それを確立する必要があります。Red Hat Container Development Kit で提供される仮想 OpenShift インスタンスを開始するには、Red Hat ユーザー名とパスワードが必要です。

Red Hat Customer Portal に登録することで、簡単にアカウントを取得できます。ホワイトバナーの右上隅にある  をクリックし、**ご自身の Red Hat アカウントにログイン** ページの 

をクリックします。

Fuse Tooling を使用すると、s2i バイナリーワークフローを使用して Fuse Integration プロジェクトを開発および展開できます。このワークフローでは、ツールがプロジェクトをローカルでビルドし、それをイメージストリームにアセンブルしてから、そのイメージストリームを OpenShift にプッシュし、そこで Docker コンテナーのビルドに使用します。Docker コンテナーがビルドされると、OpenShift はそれを Pod にデプロイします。

重要

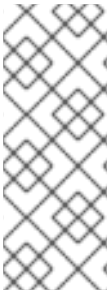
Fuse Tooling は、S2I バイナリーワークフローでのみ機能し、SpringBoot フレームワークに基づくプロジェクトでのみ機能します。

注記

Fuse Tooling はツールを使用して作成された Fuse Integration プロジェクトをリモート OpenShift サーバーにデプロイできますが、この章では、Red Hat Container Development Kit (CDK) v3.**x** を使用してローカルにインストールされた仮想 OpenShift インスタンスに Fuse Integration プロジェクトを作成およびデプロイする方法について説明します。

次のセクションでは、最初の Fuse Integration プロジェクトを作成してデプロイする方法について説明します。

- 「Red Hat Container Development Kit サーバーの追加」
- 「Container Development Environment (CDE) および仮想 OpenShift サーバーの起動」
- 「新しい OpenShift プロジェクトの作成」
- 「新規 Fuse Integration プロジェクトの作成」
- 「Fuse Integration プロジェクトの OpenShift へのデプロイ」



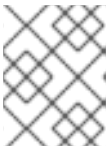
注記

Fuse Integration プロジェクトをローカル Camel コンテキストとして実行し、「ローカル Camel コンテキストとしてルートを実行」 を参照してから、**JMX Navigator** ビューに接続できます。このビューでは、ルーティングコンテキストを関しおよびテストできます。Fuse Integration プロジェクトで Camel デバッガーを実行し (パートII「ルーティングコンテキストのデバッグ」)、ルーティングコンテキストの論理エラーを公開して修正することもできます。

6.1. RED HAT CONTAINER DEVELOPMENT KIT サーバーの追加

Red Hat Container Development Kit を **Servers** ビューに追加するには、以下を実行します。

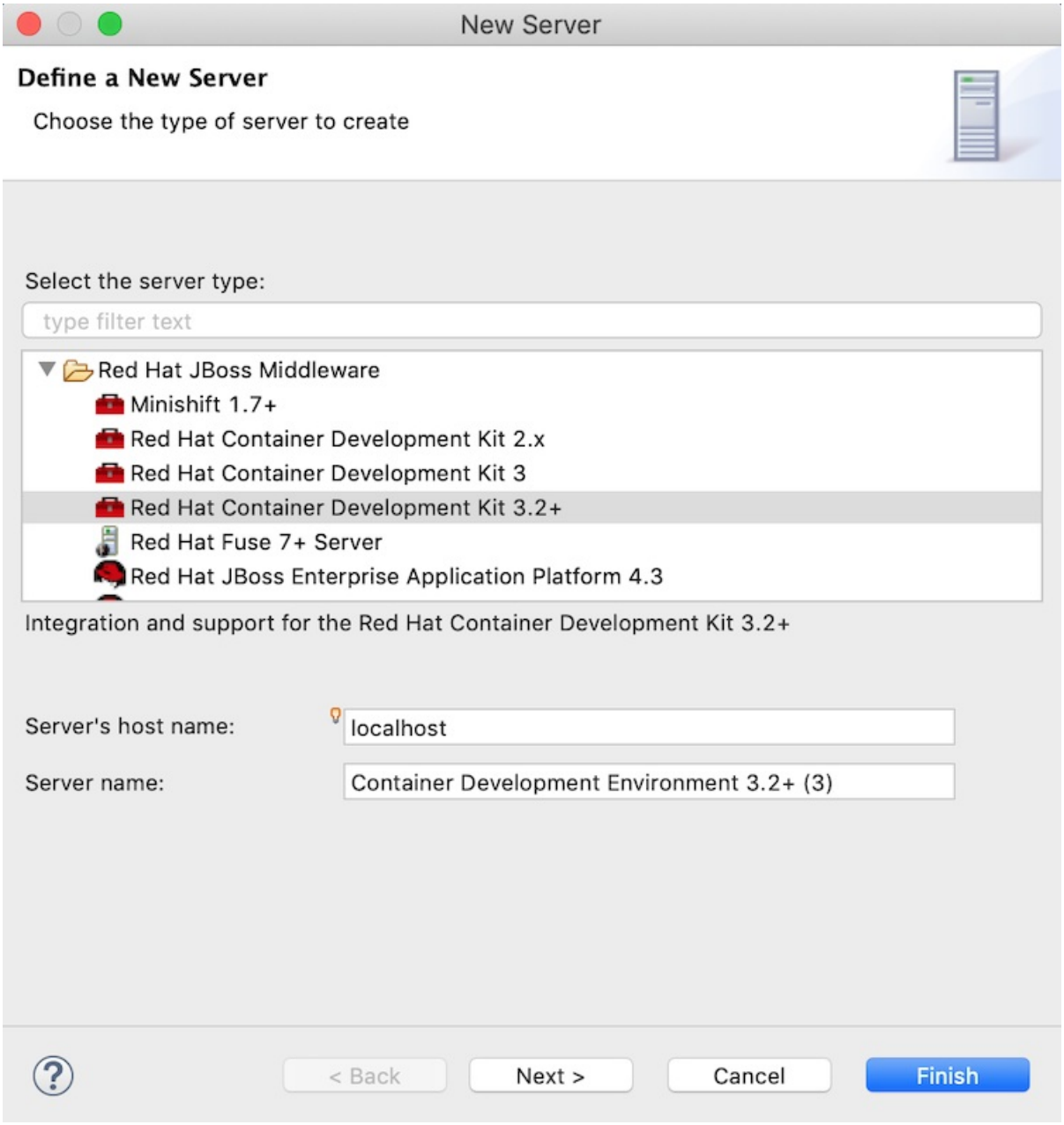
1. 必要に応じて、**Window → Perspective → Open Perspective → Fuse Integration** の順に選択して **Fuse Integration** パースペクティブに切り替えます。



注記

この手順で説明されているビューが開かない場合は、**Window → Show View → Other** を選択し、開くビューの名前を選択します。

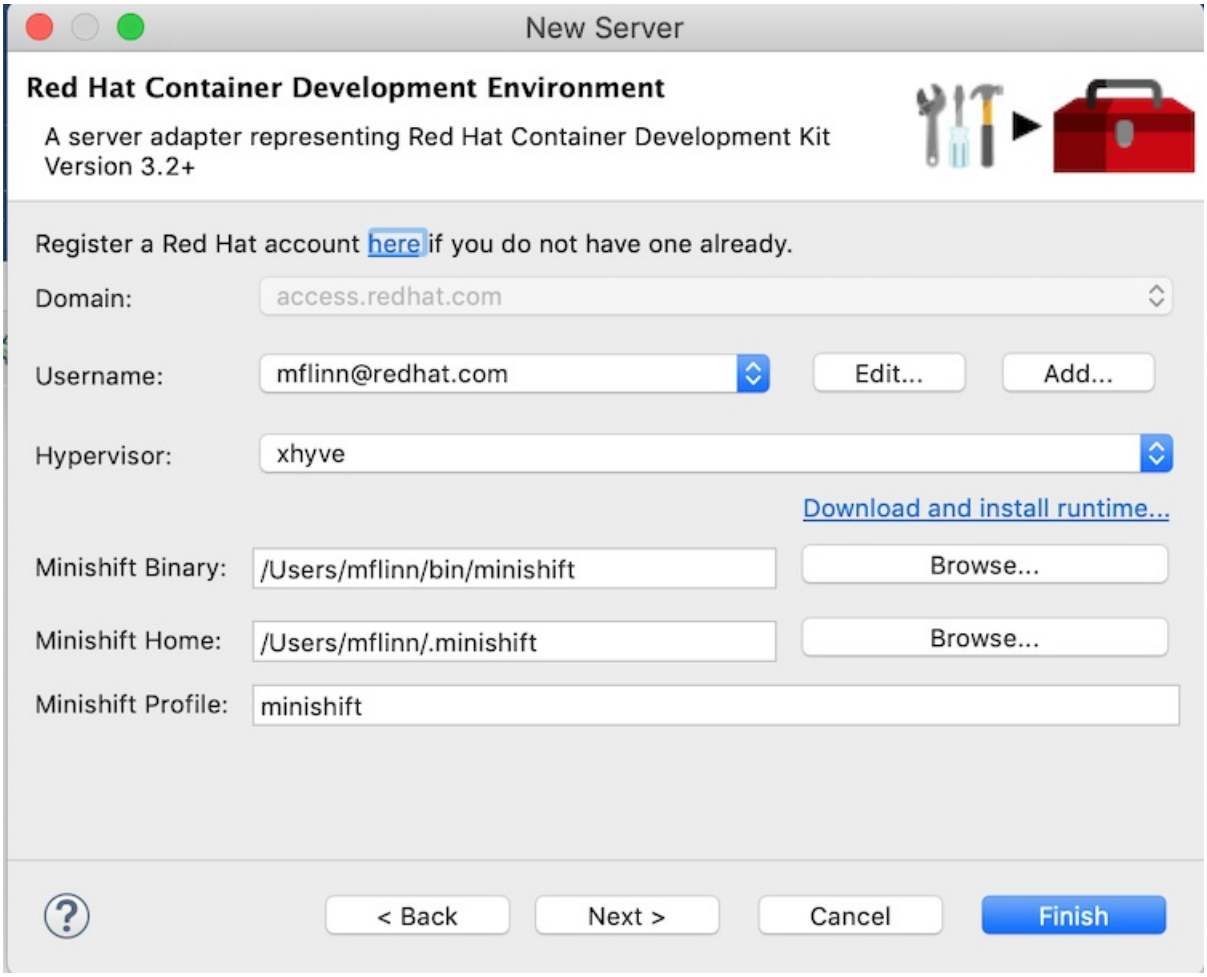
2. **Servers** ビューで **No servers are available. Click this link to create a new server...**のリンクをクリックし、**Define a New Server** ウィザードを開きます。このリンクは、**Servers** ビューにサーバーエントリーが含まれない場合にのみ表示されます。
それ以外の場合は、**Servers** ビューを右クリックしてコンテキストメニューを開き、**New → Server** を選択して **Define a New Server** ウィザードを開きます。



3. **Red Hat JBoss Middleware** → **Red Hat Container Development Kit 3.2+** を選択します。次のデフォルト値を受け入れます。
   - Server's hostname: **localhost**
   - Server name: **Container Development Environment**
4. **Next** をクリックして **Red Hat Container Development Environment**ページを開きます。
5. **MiniShift Binary** の横にある **Browse** をクリックして、Red Hat Container Development Kit 3. **x** をインストールした場所に移動し、**Open** をクリックします。
6. **Username** の横にある **Add** をクリックし、**Add a Credential** ページを開きます。
7. クレデンシャルを次のように設定します。
   - **Username** – Red Hat アカウントへのログインに使用する名前を入力します。
   - **Always prompt for password** – そのままにします (無効)。

- **Password** – Red Hat アカウントへのログインに使用するパスワードを入力します。

8. **OK** をクリックして、**Red Hat Container Development Environment**ページに戻ります。このページにデータが入力されています。以下に例を示します。

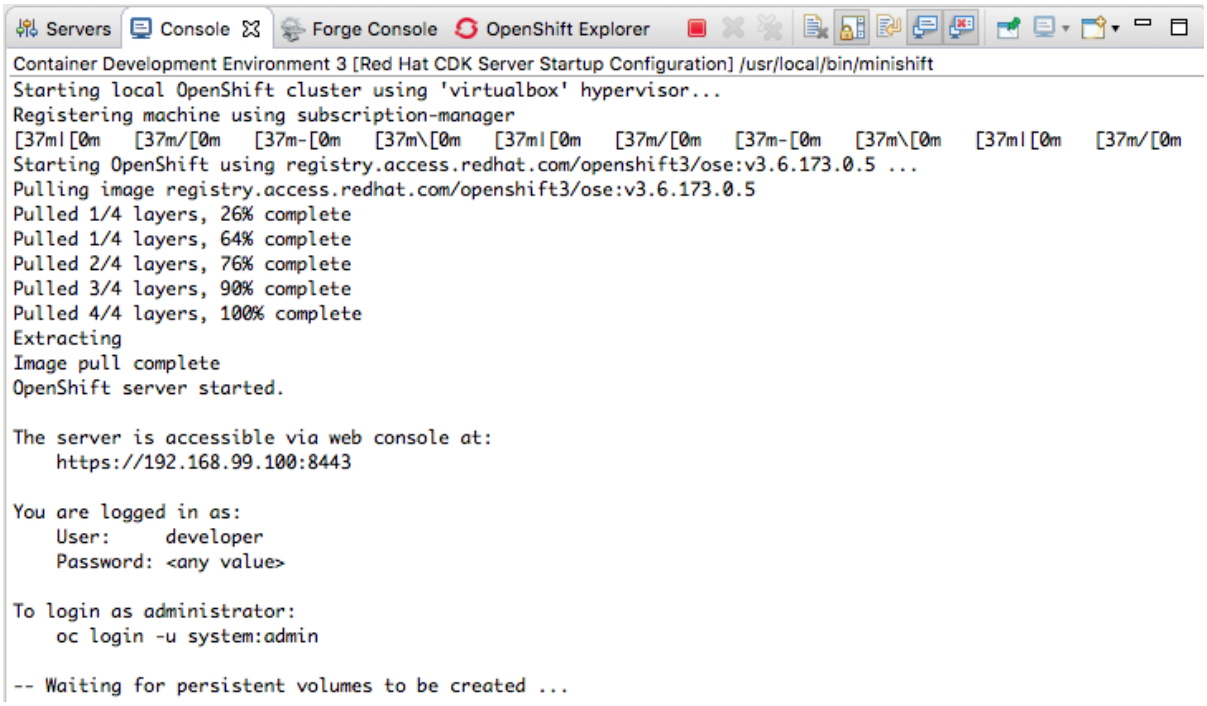


9. **Finish** をクリックします。**Container Development Environment 3.2+ [Stopped, Synchronized]** が **Servers** ビューに表示されます。**Container Development Environment 3.2+** は、CDK 3.**x** サーバーを追加する場合のデフォルトサーバー名です。

6.2. CONTAINER DEVELOPMENT ENVIRONMENT (CDE) および仮想 OPENSHIFT サーバーの起動

Container Development Environment (CDE) を起動すると、仮想 OpenShift サーバーも起動します。CDE を停止すると、仮想 OpenShift サーバーも停止します。

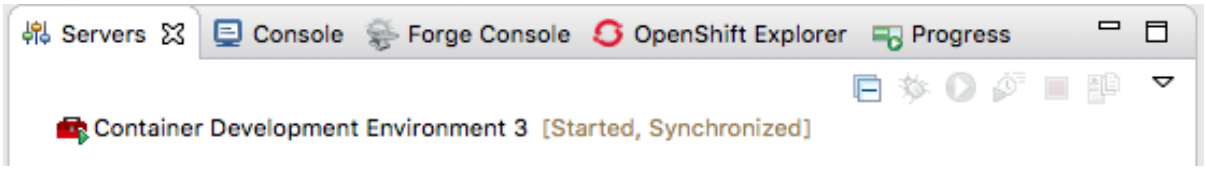
1. **Servers** ビューで **Container Development Environment 3 [stopped, Synchronized]**を選択し、**Servers** メニューバーの  をクリックします。
Console ビューが開き、起動プロセスのステータスが表示されます。



注記

CDE は、最初の起動時に、信頼できない SSL 証明書を受け入れるかどうかを尋ねます。**Yes** をクリックします。

起動プロセスが終了すると、**Servers** ビューに次のように表示されます。



2. **OpenShift Explorer** ビューに切り替えます。
仮想 OpenShift サーバーインスタンス **developer** も実行されています。

https://192.168.99.100:8443 は、OpenShift 開発者 Web コンソールの URL の例です。インストールにより、インスタンスの URL が表示されます。詳細については 「OpenShift Web コンソールへのアクセス」 を参照してください。

6.3. 新しい OPENSHIFT プロジェクトの作成

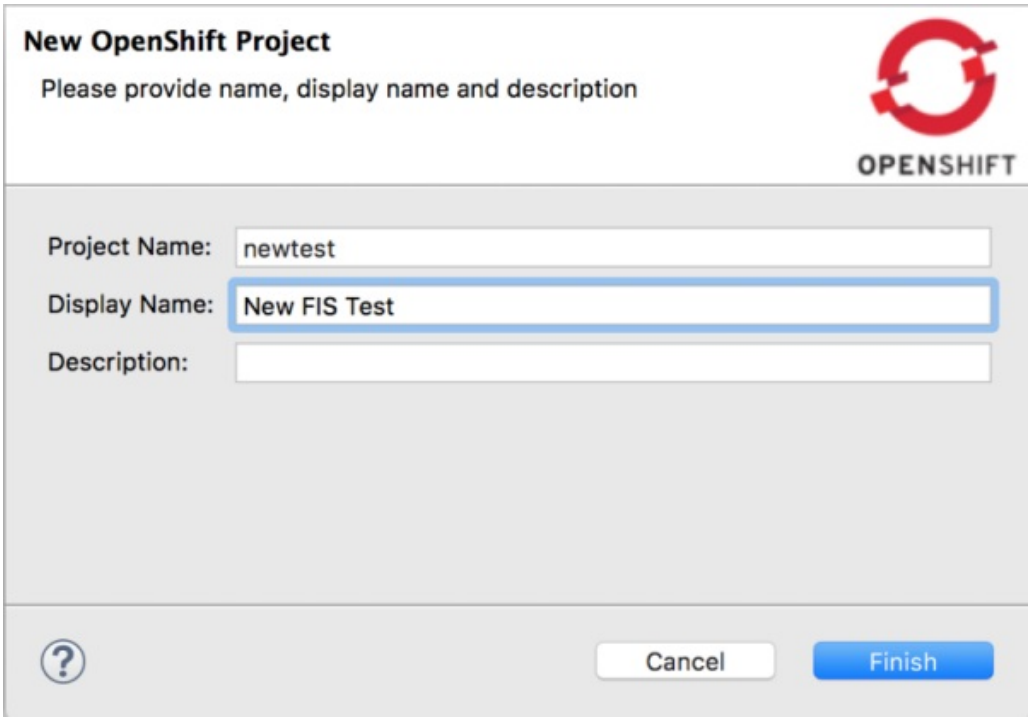
Fuse Integration プロジェクトを OpenShift にデプロイすると、ここで作成した OpenShift プロジェクトに公開されます。

1. **OpenShift Explorer** ビューで、**developer** エントリーを右クリックして、コンテキストメニューを開きます。
2. **New → Project** を選択して、**New OpenShift Project** ウィザードを開きます。
3. 新規プロジェクトのプロパティーを以下のように設定します。

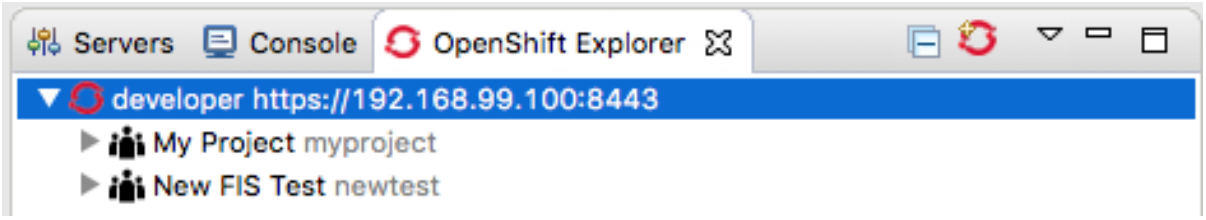
- **Project Name** フィールドに、仮想 OpenShift サーバーにおけるプロジェクトの namespace の名前を入力します。
  小文字、数字、ダッシュのみ有効です。

- **Display Name** フィールドに、仮想 OpenShift Web コンソールの **Overview** ページに表示する名前を入力します。

- **Description** フィールドはそのままにしておきます。
  以下に例を示します。



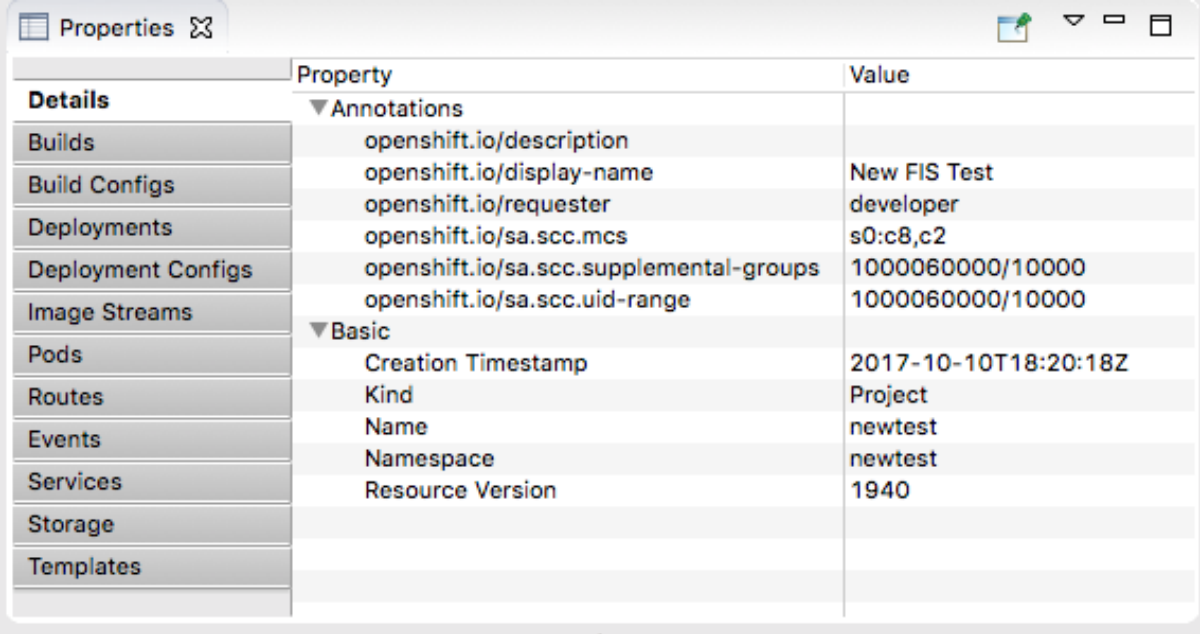
4. **Finish** をクリックします。
   新しい OpenShift プロジェクト (この例では **New FIS Test newtest**) は、**OpenShift Explorer** タブの、この例では **developer** https://192.168.99.100:8443 の下に表示されます。



注記

MyProject myproject は、OpenShift に含まれている初期サンプルプロジェクトです。

OpenShift Explorer ビューで **New FIS Test newtest** を選択すると、**Properties** ビューにプロジェクトの詳細が表示されます。以下に例を示します。



注記

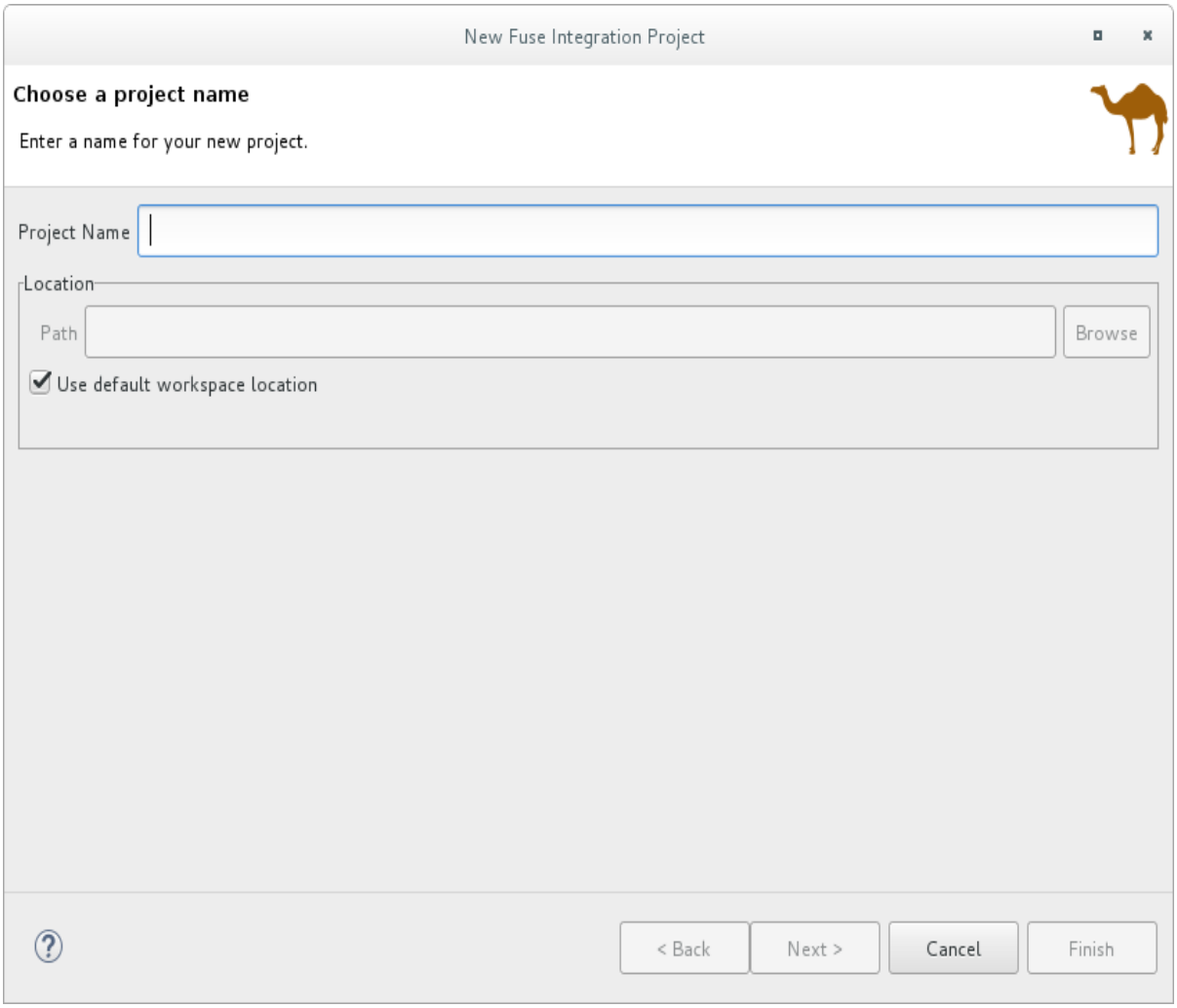
プロジェクトを OpenShift にデプロイすると、**Properties** ビューは OpenShift Web コンソールが行うのと同じプロジェクトに関する情報を収集して表示します。

6.4. 新規 FUSE INTEGRATION プロジェクトの作成

新規 Fuse Integration プロジェクトを作成する前に、ステージングリポジトリーを有効にする必要があります。一部の Maven アーティファクトはデフォルトの Maven リポジトリーにないため、この設定が必要です。ステージングリポジトリーを有効にするには、**Window → Preferences → Fuse Tooling → Staging Repositories** の順に選択します。

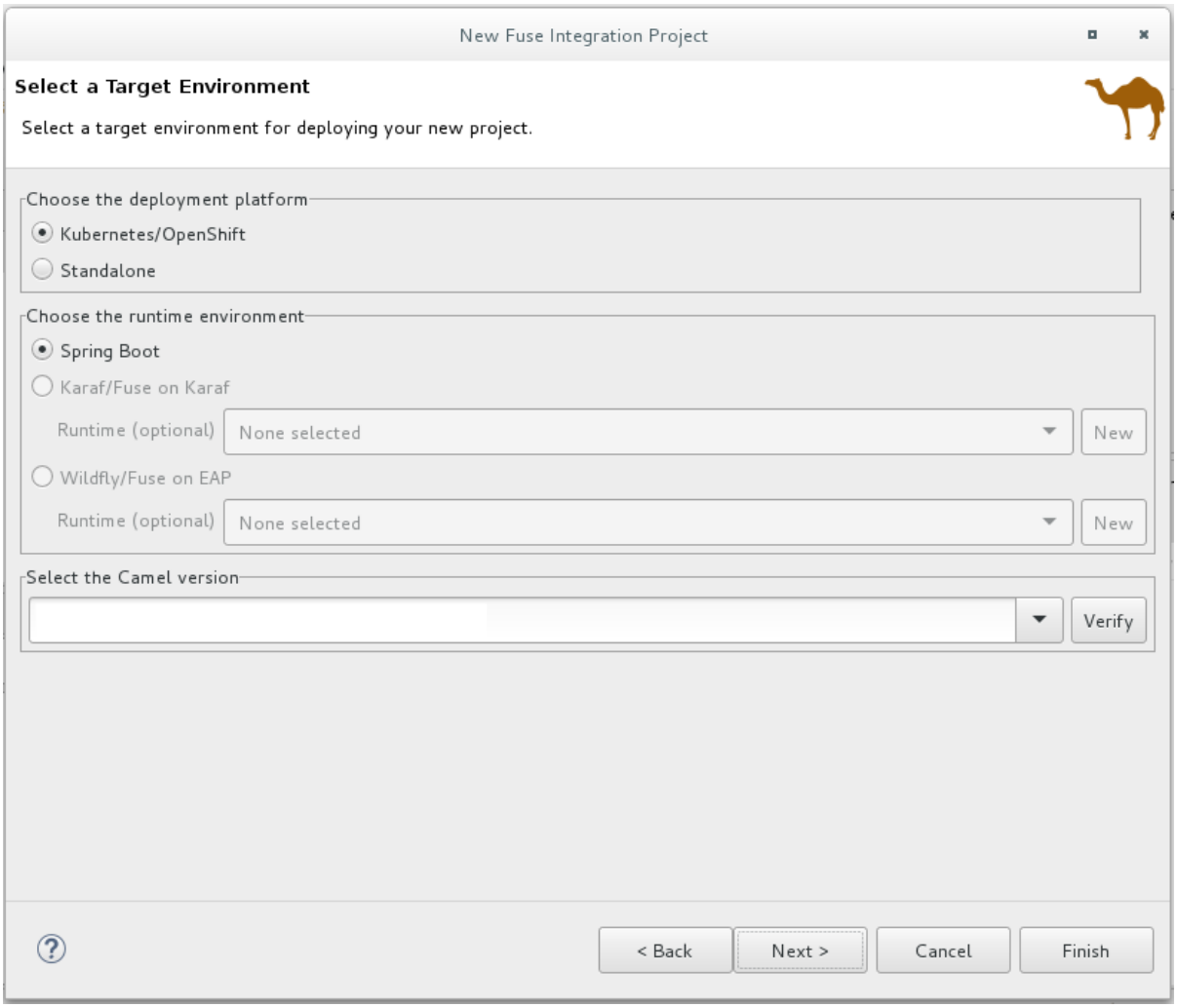
Fuse Integration プロジェクトを作成するには、**Spring Boot on OpenShift** テンプレートを使用します。

1. **Project Explorer** ビューで右クリックしてコンテキストメニューを開き、**New → Fuse Integration Project** を選択してウィザードの **Choose a project name** ページを開きます。



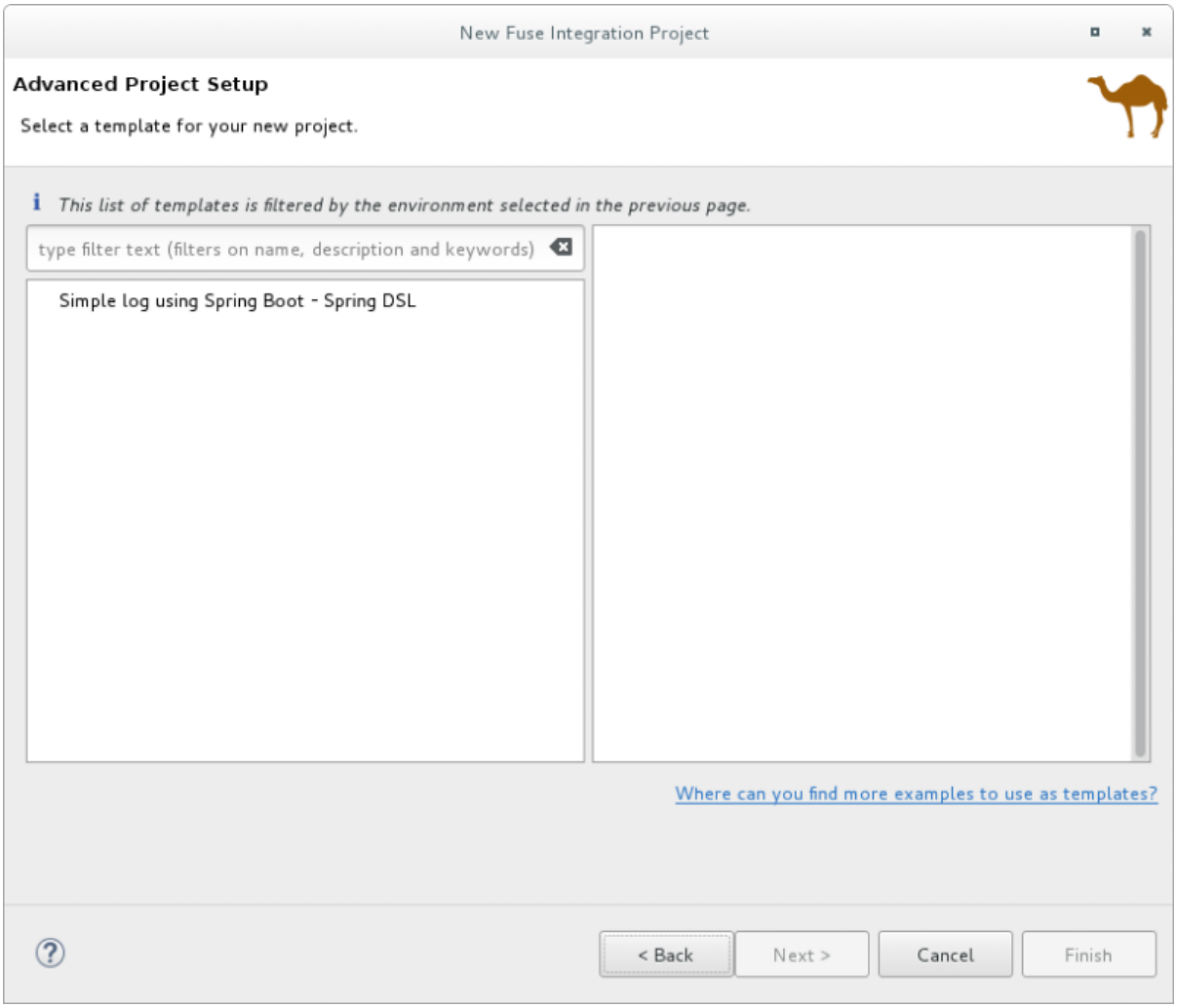
2. **Project Name** フィールドに、使用しているワークスペースに固有の名前を入力します (例: **myFISproject**)。
   他のオプションでデフォルト値を使用します。

3. **Next** をクリックして、**Select a Target Runtime** ページを開きます。

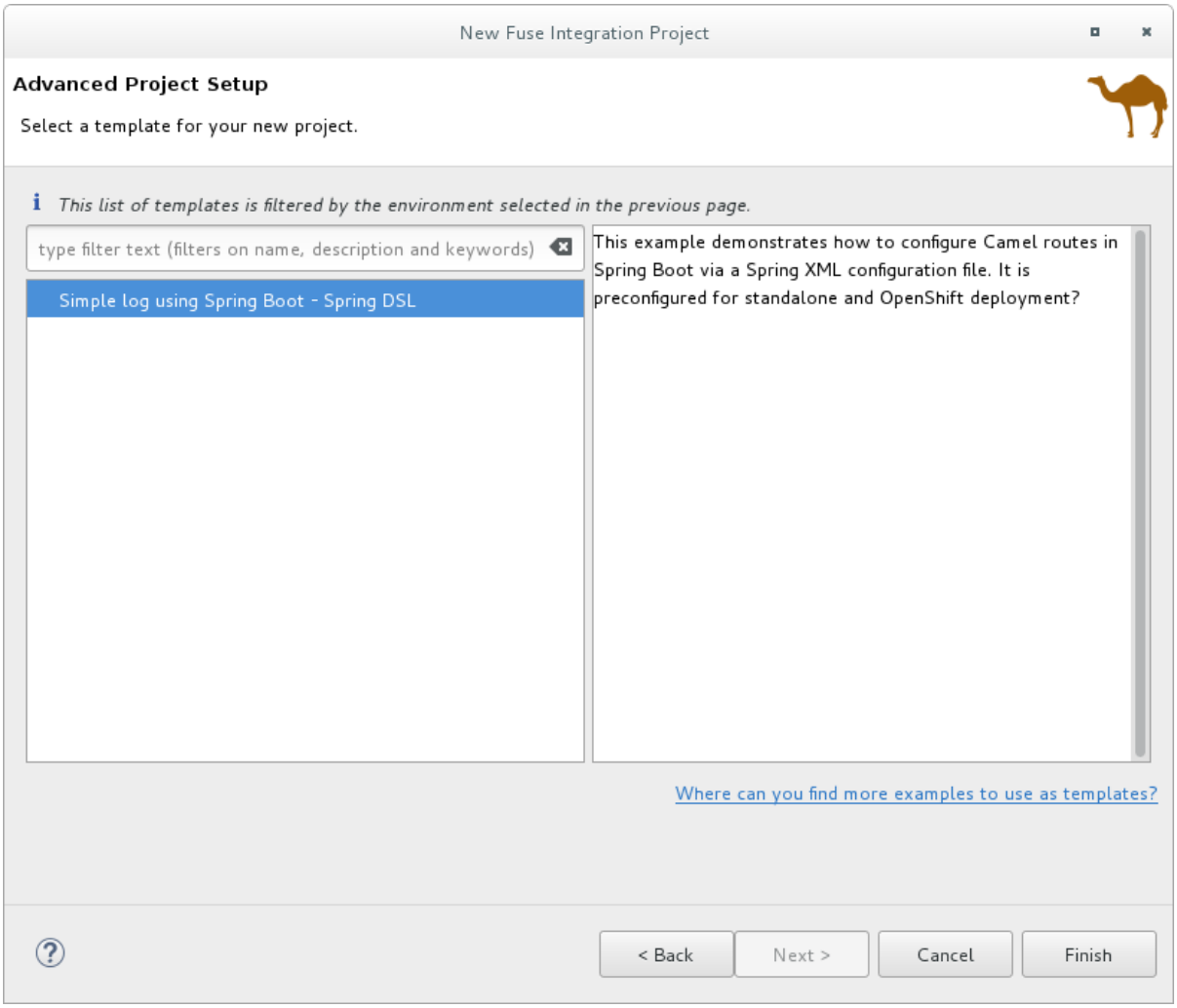


Target Runtime (**No Runtime selected**) および **Camel Version** はデフォルト値をそのまま使用します。

4. **Next** をクリックして **Advanced Project Setup** ページを開きます。



5. **Simple log using Spring Boot - Spring DSL**テンプレートを選択します。



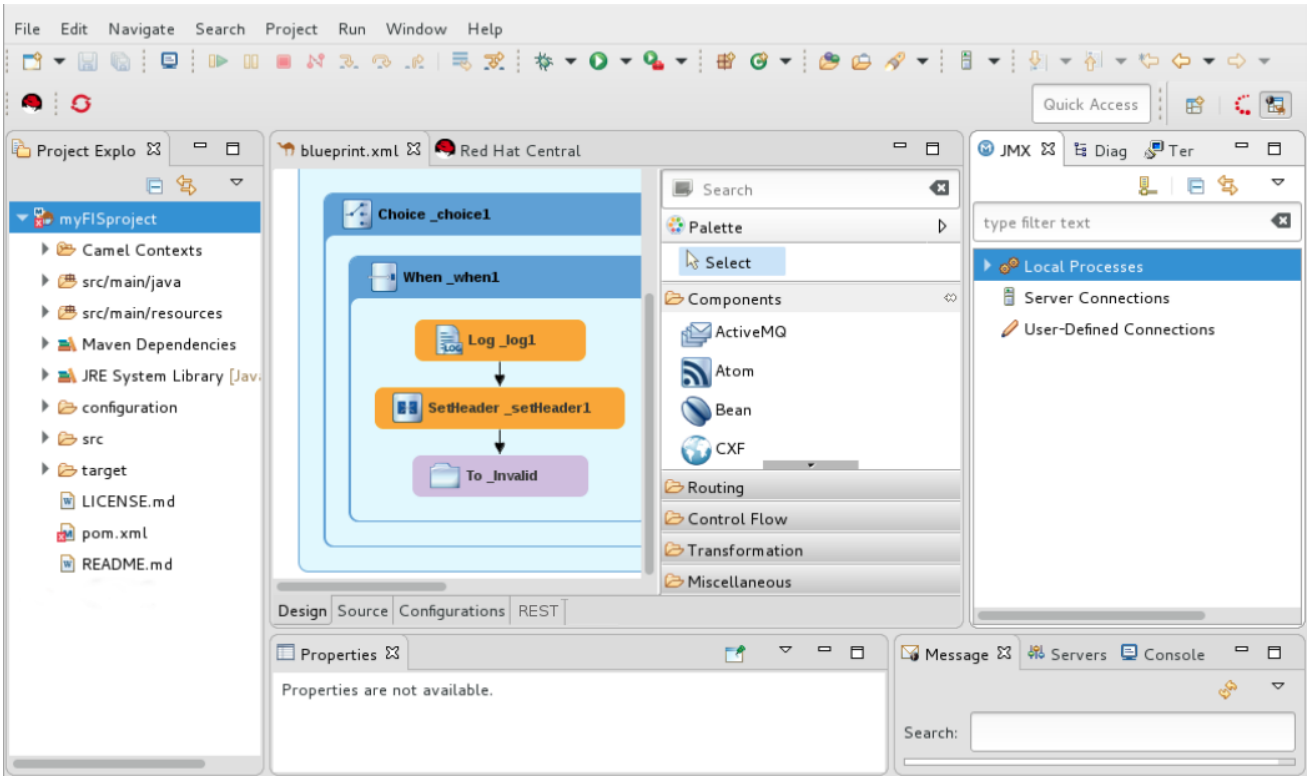
6. **Finish** をクリックします。

注記

初めての Fuse Integration プロジェクトでダウンロードされる依存関係の数が多いため、プロジェクトのビルドには時間がかかる場合があります。

Fuse Integration パースペクティブがまだ開いていない場合、Developer Studio により、ここで開くかどうかを示すようにプロンプトが表示されます。**Yes** をクリックします。

ビルドが完了すると、**Fuse Integration** パースペクティブにプロジェクトが表示されます。以下はその例です。

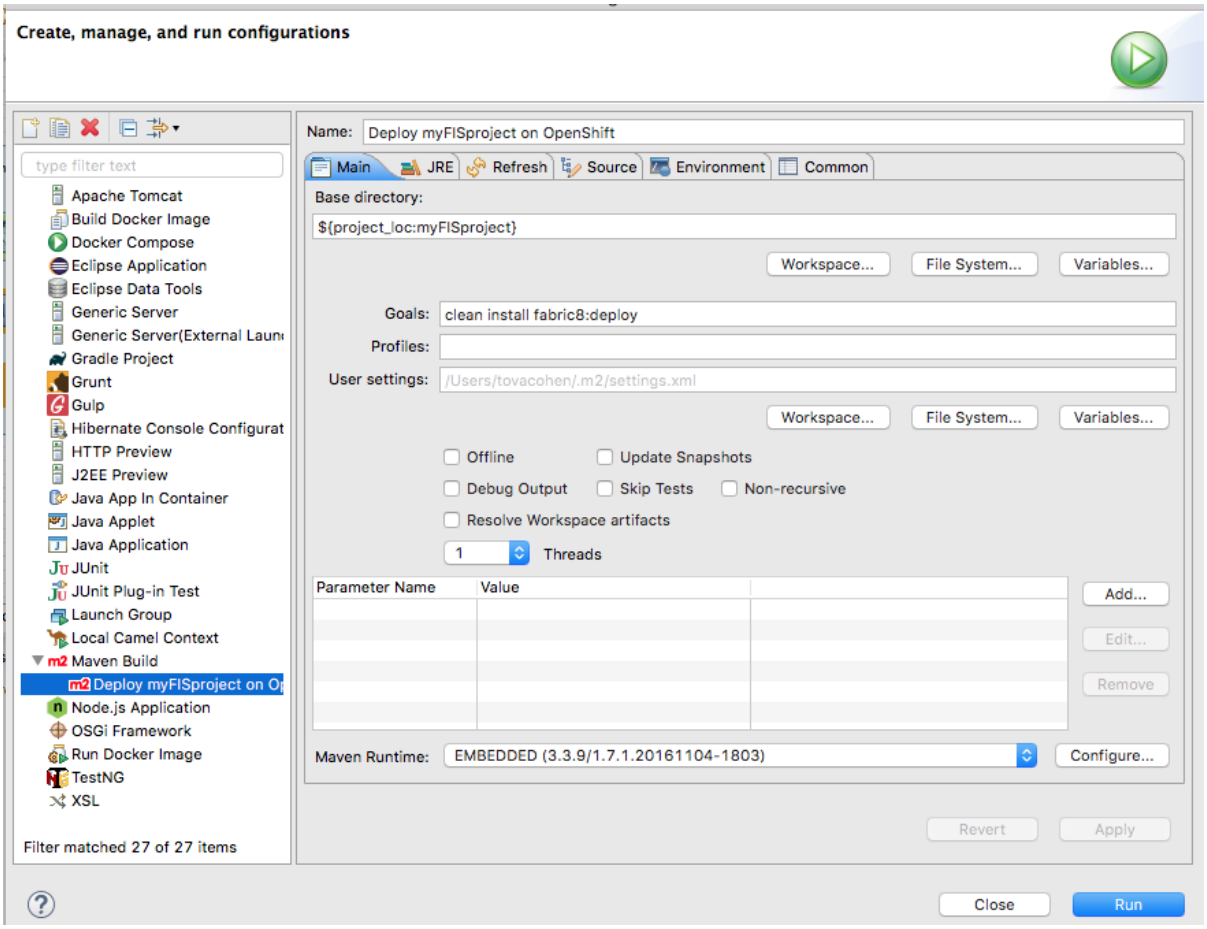


この時点で、以下を実行できます。

- OpenShift へのプロジェクトのデプロイ
- 「ローカル Camel コンテキストとしてルートを実行」 ルーティングコンテキストがローカルマシンで正常に実行されていることの確認
  JMX Navigator ビューで実行中のコンテキスト (「ローカル JMX サーバーでのプロセスの表示」 を参照) に接続すると、ルートコンポーネントを監視し、ルートが想定どおりに実行されるかどうかをテストできます。
  - ルートコンポーネントの JMX 統計の表示 (20章*コンポーネントの JMX 統計情報の表示* を参照)
  - 実行中のルートの表示 (24章*ルーティングエンドポイントの管理*)
  - 実行中のルートの一時停止/再開 (26章*ルーティングコンテキストの管理*)
  - 実行中のルートでのトレースの開始/停止 (22章*ルートのトレース*)
- ロジックエラーの検出と修正を目的とした、プロジェクトの **camel-context.xml** ファイルでの Camel デバッガーの実行 (パートII「ルーティングコンテキストのデバッグ」 を参照)

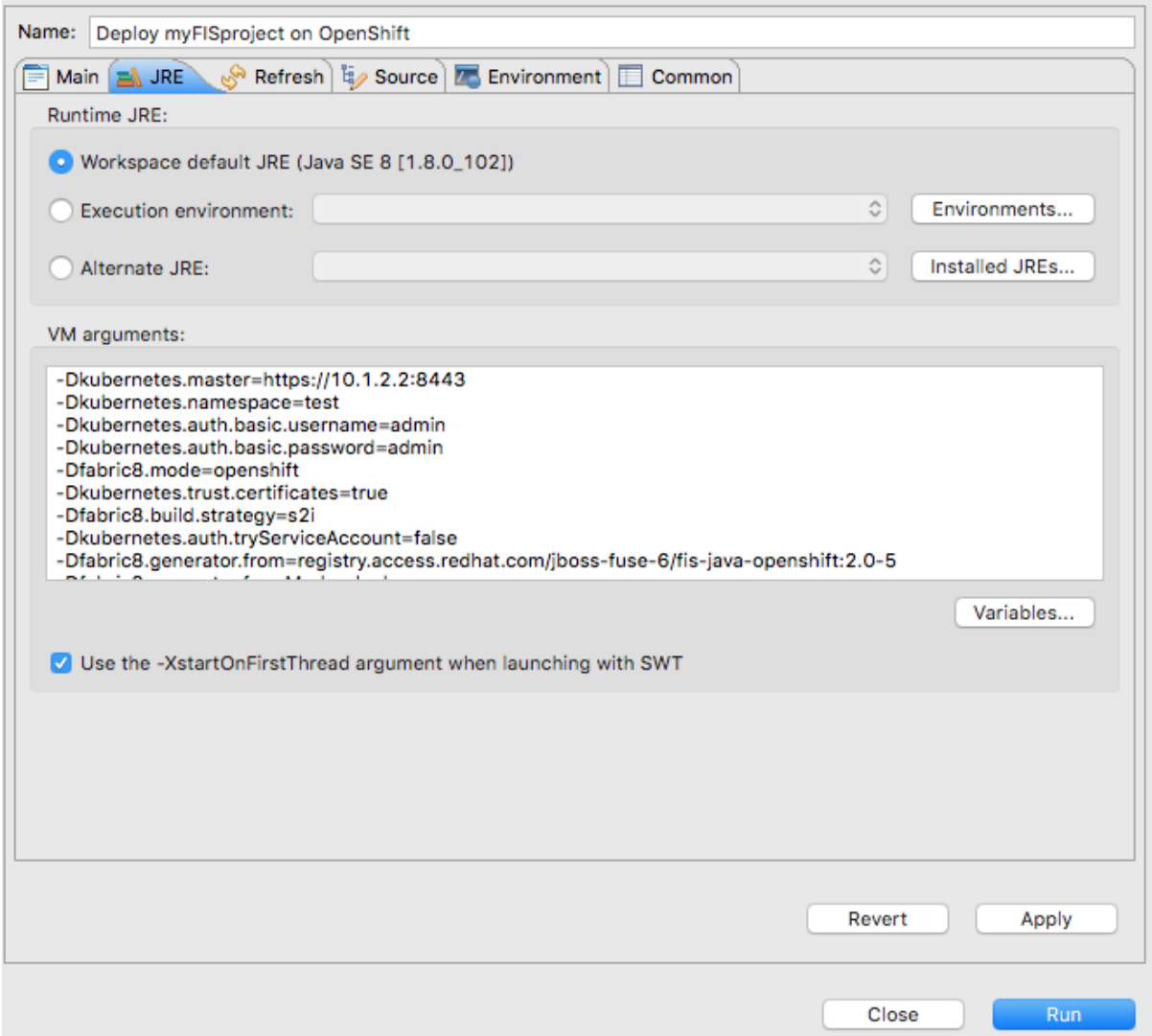
6.5. FUSE INTEGRATION プロジェクトの OPENSHIFT へのデプロイ

1. **Project Explorer** ビューで、プロジェクトのルート (この例では **myFISproject**) を右クリックして、コンテキストメニューを開きます。
2. **Run As** → **Run Configurations** の順に選択し、**Run Configurations** ウィザードを開きます。
3. サイドバーメニューで、**Maven Build** → **Deploy <projectname> on OpenShift** (この例では **Deploy myFISproject on OpenShift**) を選択し、プロジェクトのデフォルトラン設定を開きます。

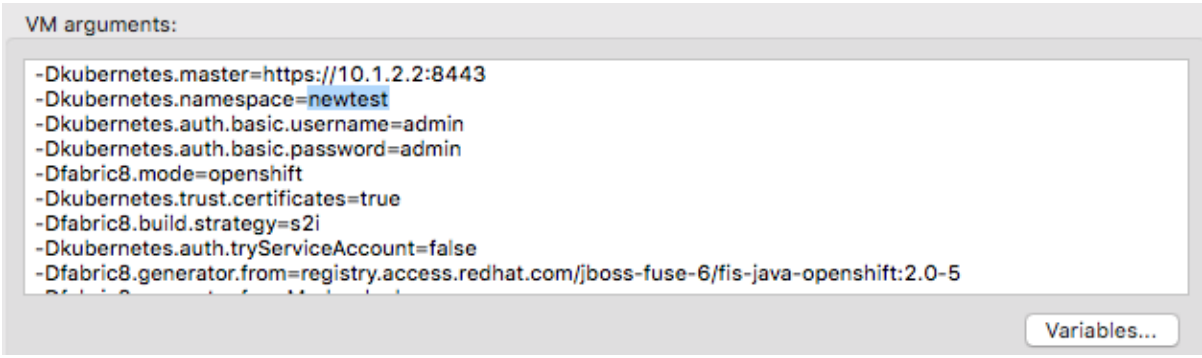


Main タブのデフォルト設定をそのまま使用します。

4. JRE タブを開いて、VM 引数にアクセスします。



5. **VM arguments** ペインで、**-Dkubernetes.namespace=test** 引数の値を変更して、OpenShift プロジェクトの作成時に使用したプロジェクト名 (「新しい OpenShift プロジェクトの作成」の OpenShift のプロジェクト名) に一致させます。
この例では、デフォルト値の **test** を **newtest** に変更します。



OpenShift の設定によっては、それをサポートするために他の VM 引数を変更する必要がある場合があります。

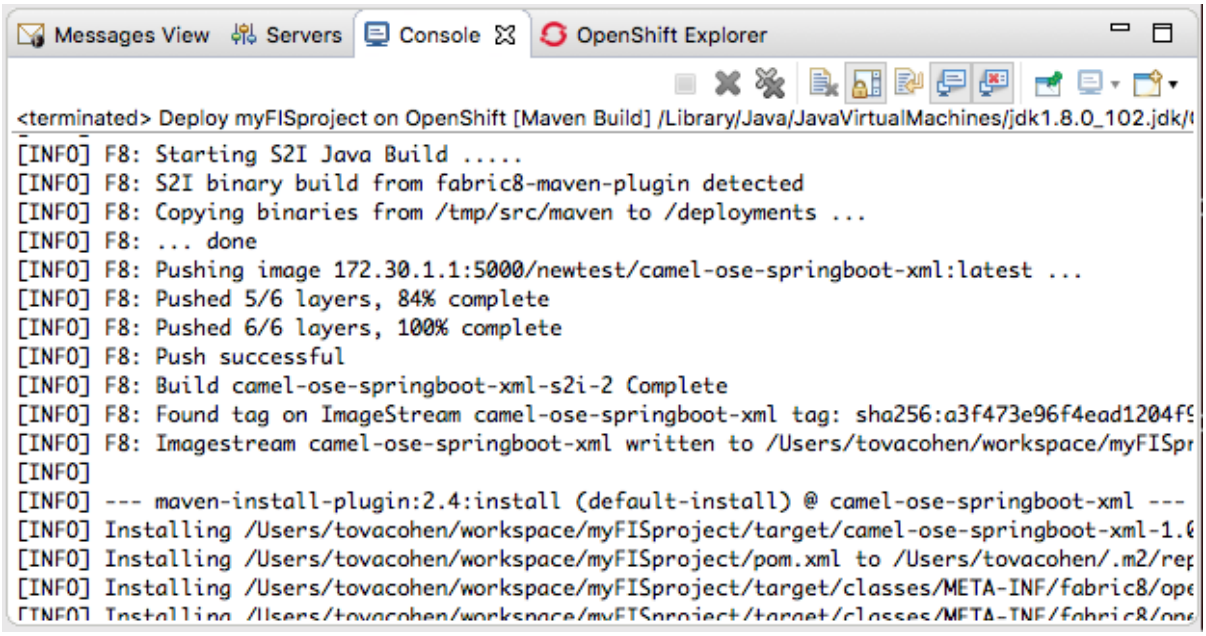
- **-Dkubernetes.master=https://192.168.99.1:8443**
複数の OpenShift インスタンスを実行する場合、またはリモートインスタンスを使用する場合は、デプロイメントの対象となる OpenShift インスタンスの URL を指定する必要があります。上記の URL はその例です。

- **-Dkubernetes.trust.certificates=true**

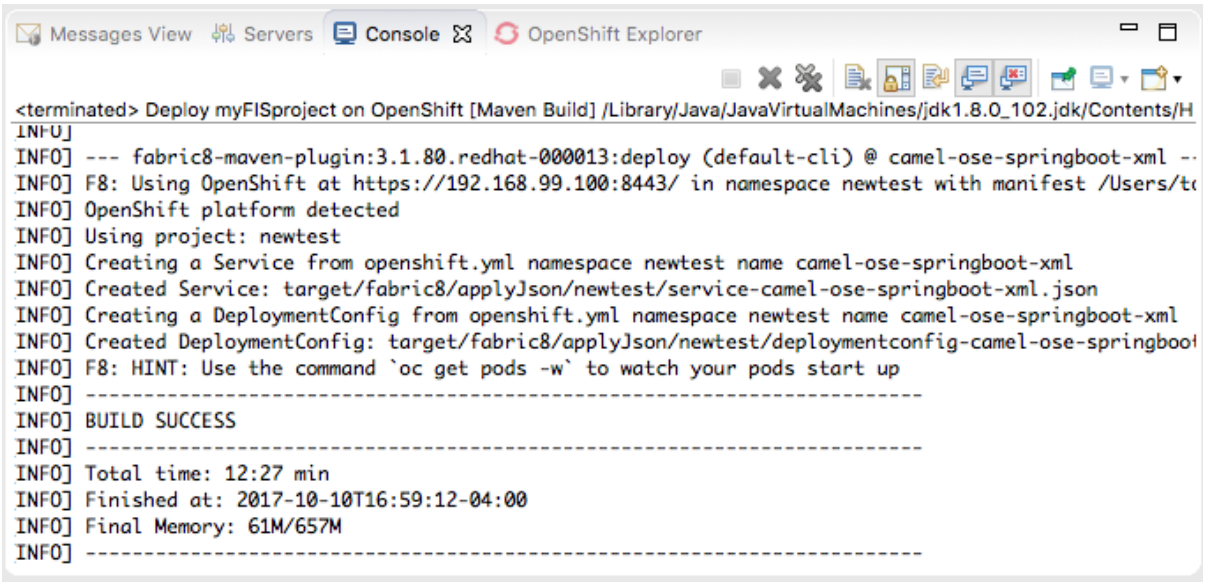
- CDK を使用する場合、この引数は必須です。**true** に設定したままにしてください。
- 有効な SSL 証明書を持つ OpenShift インスタンスを使用している場合は、この引数の値を **false** に変更します。

6. **Apply** をクリックしてから、**Run** をクリックします。
ダウンロードする依存関係の数が多いため、初回の展開には時間がかかる場合があります。コンピューターの速度とインターネット接続が要因になっています。通常、初回の展開が完了するまでに 25〜35 分かかります。

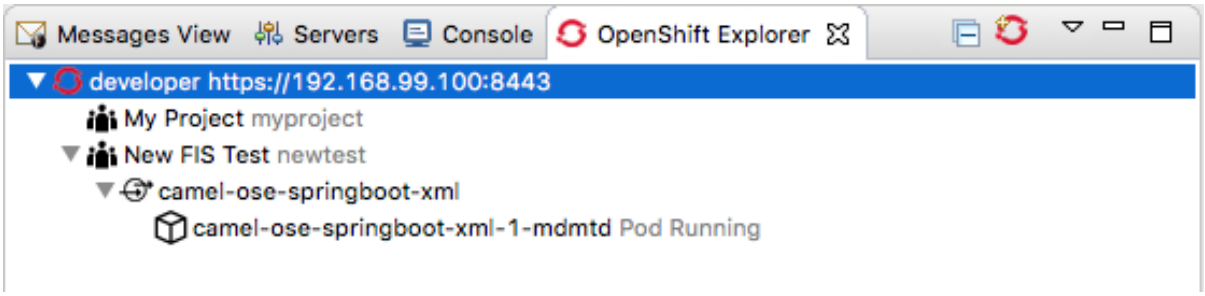
   Console ビューでは、デプロイプロセスの進行状況を追跡できます。以下の出力のエントリー Pushing image 172.30.1 ... * は、プロジェクトが正常にビルドされ、アプリケーションイメージが OpenShift にプッシュされて Docker コンテナーのビルドに使用されることを示します。



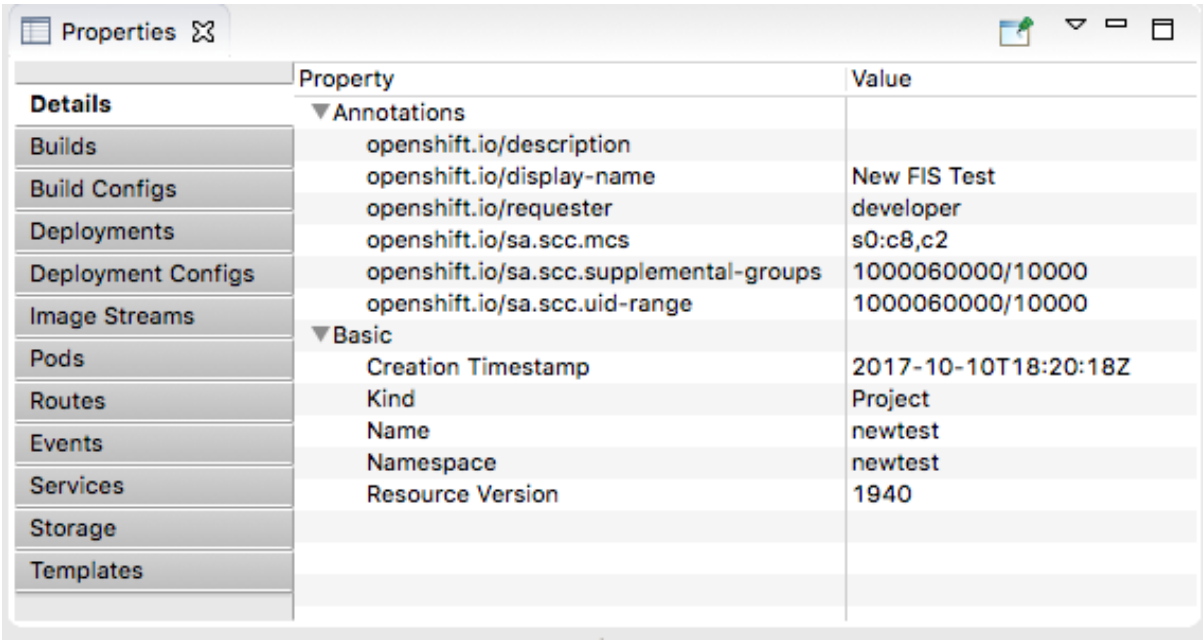
   デプロイメントが正常に完了すると、**Console** ビューに **BUILD SUCCESS** が表示されます。



7. **OpenShift Explorer** ビューに切り替えて、**New FIS Test newtest**を選択します。

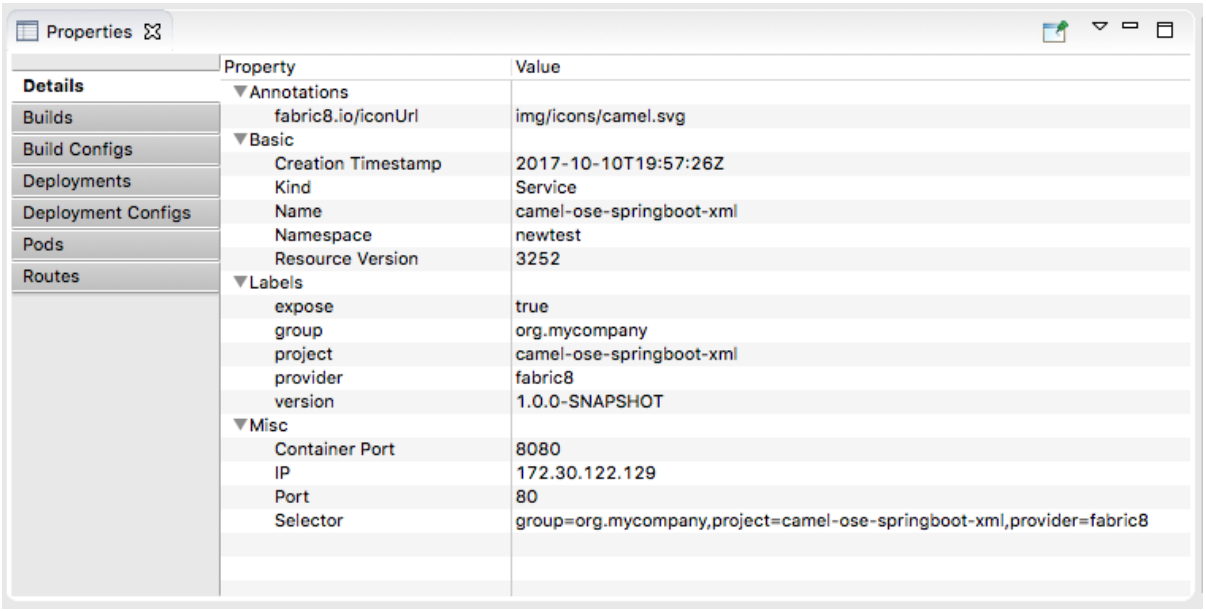


Properties ビューの **Details** ページには、プロジェクトのプロパティー値がすべて表示されます。



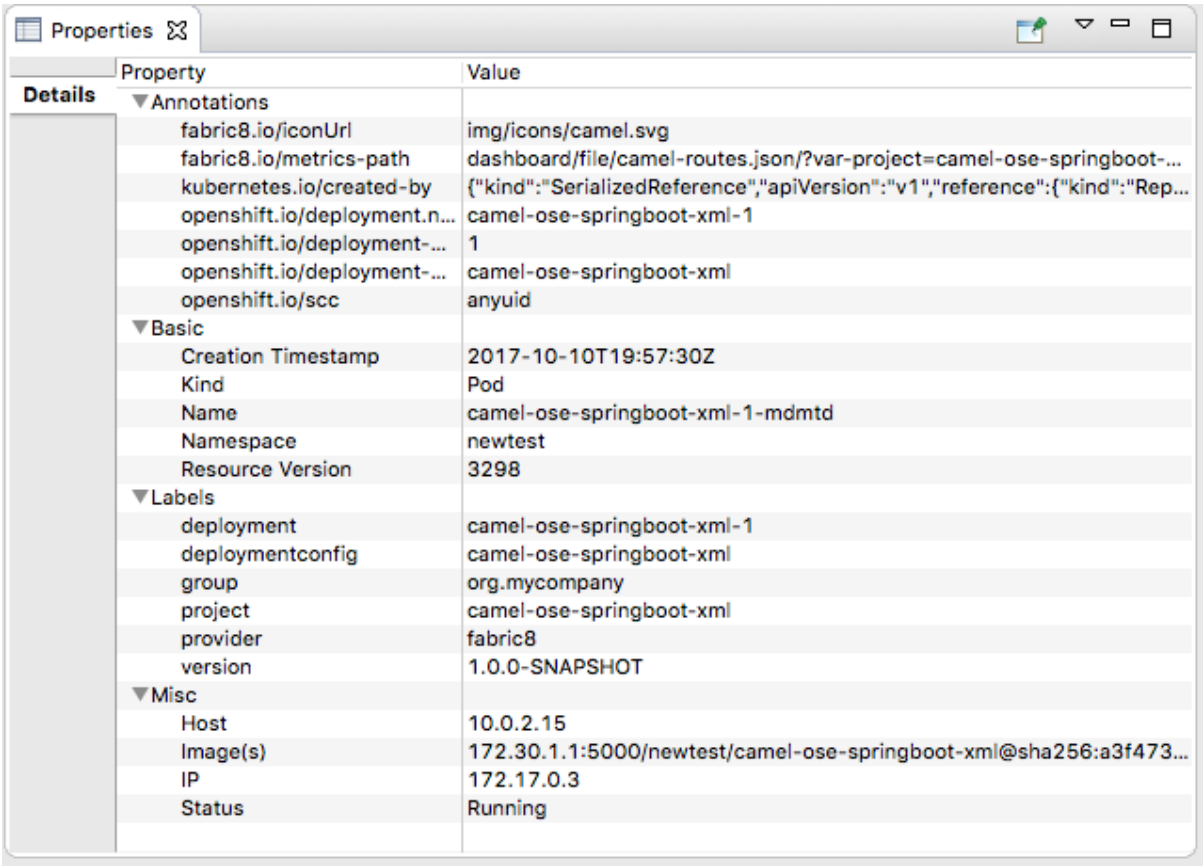
他のタブ (**Builds**、**Build Configs**、**Deployments** ...) を開き、プロジェクトの他のプロパティーを表示します。**Properties** ビューには、OpenShift Web コンソールと同じ情報が表示されます。

8. **OpenShift Explorer** ビューで **camel-ose-springboot-xml** を選択し、**Properties** ビューでその詳細を表示します。

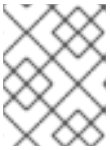


他のタブをスクロールして、デプロイメント設定の他のプロパティーを表示します。

9. OpenShift Explorer ビューで **camel-ose-springboot-xml-1-mdmtd Pod Running** を選択し、**Properties** ビューで実行中のインスタンスの詳細を表示します。



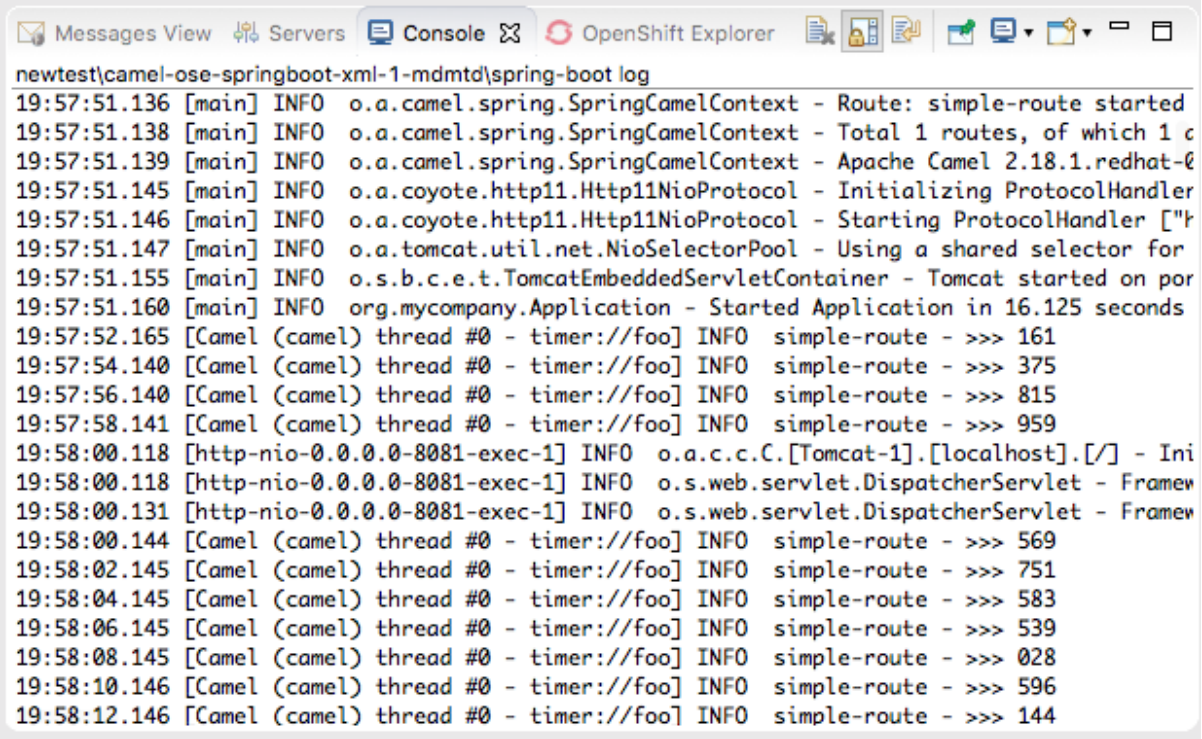
10. OpenShift Explorer ビューで **camel-ose-springboot-xml-1-mdmtd Pod Running** を右クリックし、続いて**Pod Logs...** を選択します。



注記

プロンプトが表示されたら、インストールされた **oc** 実行可能ファイルへのパスを入力します。Pod ログを取得する必要があります。

Console ビューが自動的に開き、実行中の Pod からのログが表示されます。



Console ビューのメニューバーの  をクリックしてセッションを終了し、コンソールの出力をクリアします。

6.6. OPENSHIFT WEB コンソールへのアクセス

注記

この情報は、Red Hat Container Development Kit のインストールにのみ適用されます。

OpenShift Web コンソールにアクセスするには、ブラウザーを開き、インスタンスとマシンに固有の OpenShift サーバーの URL を入力します。たとえば、ブラウザーのアドレスフィールドに **https://192.168.99.100:8443** を入力します。

デフォルトのクレデンシャルを使用して、開発者または管理者として Web コンソールにログインできます。

- デフォルトの開発者ロール
  開発者ユーザーは、自分のプロジェクトと、OpenShift v3 の機能を示す提供された OpenShift サンプルプロジェクトのみを表示できます。開発者ユーザーは、OpenShift にデプロイされている、自分が所有する任意のプロジェクトを作成、編集、および削除できます。
  - Username: **developer**
  - Password: **developer**
- デフォルトの管理者ロール
  管理者ユーザーは、OpenShift (CDK) 上のすべてのプロジェクトを表示およびアクセスできます。管理者ユーザーは、OpenShift にデプロイされた任意のプロジェクトを作成、編集、および削除できます。
  - Username: **admin**
  - Password: **admin**

OpenShift Web コンソールの使用ついて、詳しくは スタートガイド を参照してください。

第7章 RED HAT FUSE SAP TOOL SUITE の使用

Red Hat Fuse SAP Tool Suite を使用すると、Camel ルートをリモートの SAP アプリケーション・サーバーと統合できます。リモート機能呼び出し (RFC) および中間文書 (IDoc) の送受信をサポートするために、さまざまな SAP コンポーネントが提供されています。SAP Tool Suite は、SAP の JCo および IDoc クライアントライブラリーに依存しています。これらのライブラリーをインストールして使用するには、SAP Service Marketplace アカウントが必要です。

7.1. RED HAT FUSE SAP TOOL SUITE のインストール

概要

Red Hat Fuse SAP Tool Suite には、Edit SAP Connection Configuration ダイアログボックスがあり、SAP アプリケーションサーバーと宛先接続を作成および管理するのに役立ちます。このスイートは、SAP によって個別にライセンスされているサードパーティーの JCo および IDoc クライアントライブラリーを必要とするため、デフォルトではインストールされていません。

SAP ツールのプラットフォーム制限

SAP ツールスイートはサードパーティーの JCo および IDoc ライブラリーに依存しているため、これらのライブラリーがサポートするプラットフォームにのみインストールできます。SAP ツールのプラットフォーム制限の詳細は、Red Hat Fuse 7.12 でサポートされる構成 を参照してください。

前提条件

- Fuse SAP Tool Suite をインストールする前に、JCo および IDoc ライブラリーを http://service.sap.com/connectors からダウンロードする必要があります。
- オペレーティングシステムに適切な JCo および IDoc ライブラリーを決定するには、Red Hat Fuse Supported Configurations ページを参照してください。
- JCo および IDoc ライブラリーをダウンロードするには、SAP Service Marketplace アカウントが必要です。
- このインストール手順では、ダウンロードしたファイルをアーカイブ形式のままにしておくことができます。内容を抽出する必要はありません。

SAP 宛先接続の作成とテスト

概要

Fuse SAP Tool スイートでは、Edit SAP Connection Configuration ダイアログボックスを使用して、SAP アプリケーション宛先接続を作成および管理できます。このセクションでは、SAP 宛先接続を作成およびテストする方法について説明します。

手順

SAP 宛先接続を作成してテストするには、次の手順を実行します。

1. ルートエディターのグローバル **Configurations** タブに移動し、**Add** をクリックします。**Create new global element** ビューが表示されます。

2. SAP で、作成する接続のタイプを選択します。SAP Connection を選択し、OK をクリックします。
   Edit SAP Connection Configuration ダイアログボックスが表示されます。これにより、宛先およびサーバー接続設定を作成、編集、削除できます。

3. 新しい Destination Data Store を作成するには、Add Destination タブをクリックします。
   Create Destination ダイアログボックスが表示されます。

4. Destination Name フィールドに宛先の名前を入力し、Ok をクリックします。

5. Properties ダイアログボックスで以下を実行します。

   a. Basic タブをクリックして、SAP 宛先への接続に必要な基本プロパティーを設定します。このタブで、次のプロパティーフィールドに入力して接続を設定します。

      - SAP Application Server
      - SAP System Number
      - SAP Client
      - Logon User
      - Logon Password
      - Logon Language

   b. Connection タブをクリックして、SAP 宛先への接続に必要な値を追加します。次のプロパティーフィールドに入力して、接続を設定します。

      - SAP System Number
      - SAP Router String
      - SAP Application Server
      - SAP Message Server
      - SAP Message Server Port
      - Gateway Host
      - Gateway Port
      - SAP System ID
      - SAP Application Server Group

   c. Authenticate タブをクリックして、SAP 宛先の検証に必要な値を追加します。次のプロパティーフィールドに入力して、接続を設定します。

      - SAP Authentication type
      - SAP Client
      - Logon User
      - Logon User Alias

- Logon Password
- SAP SSO Logon Ticket
- SAP X509 Login Ticket
- Logon Language

d. Special タブをクリックします。このタブで、次のプロパティーフィールドに入力して接続を設定します。

- Select CPIC Trace
- Initial Codepage

e. Pool タブをクリックし、次のプロパティーフィールドに入力して接続を設定します。

- Connection Pool Peak Limit
- Connection Pool Capacity
- Connection Pool Expiration Time
- Connection Pool Expire Check Period
- Connection Pool Max Get Client Time

f. SNC タブをクリックし、次のプロパティーフィールドに入力して接続を設定します。

- SNC Partner Name
- SNC Level of Security
- SNC Name
- SNC Library Path

g. Repository タブをクリックし、次のプロパティーフィールドに入力して接続を設定します。

- Repository Destination
- Repository Logon User
- Repository Logon Password

注記

これらの設定について詳しい情報が必要な場合は、SAP のドキュメントを参照してください。

6. これで、宛先接続をテストする準備が整いました。Edit SAP Connection Configuration ダイアログボックスで宛先名を右クリックし、Test を選択します。
Test Destination Connection ダイアログボックスが開きます。

7. ダイアログボックスでは、現在の宛先設定を使用して SAP Destination Data Store に接続します。テストが成功すると、ステータス領域に次のメッセージが表示されます。

> Connection test for destination 'YourDestination' succeeded.

それ以外の場合は、ステータス領域にエラーレポートが表示されます。

8. **Close** をクリックして、**Test Destination Connection** ダイアログボックスを閉じます。
9. **Finish** をクリックします。新しく作成された **SAP Destination Connection** が SAP の下に表示されます。

7.2. SAP サーバー接続の作成とテスト

概要

Fuse SAP Tool スイートでは、Edit SAP Connection Configuration ダイアログボックスを使用して、SAP アプリケーションサーバー接続を作成および管理できます。このセクションでは、SAP サーバー接続を作成およびテストする方法について説明します。

手順

SAP サーバー接続を作成してテストするには、次の手順を実行します。

1. ルートエディターのグローバル **Configurations** タブに移動し、**Add** をクリックします。**Create new global element** ビューが表示されます。
2. **SAP** で、作成する接続のタイプを選択します。**SAP Connection** を選択し、**OK** をクリックします。
   Edit SAP Connection Configuration ダイアログボックスが表示されます。これにより、宛先およびサーバー接続設定を作成、編集、削除できます。
3. 新しい **Server Data Store** を作成するには、**Add Server** タブをクリックします。
   Create Server ダイアログボックスが表示されます。
4. **Server Name** フィールドにサーバーの名前を入力し、**Ok** をクリックします。
5. **Properties** ダイアログボックスで以下を実行します。
   a. **Mandatory** タブをクリックして、SAP サーバーへの接続に必要な基本プロパティーを設定します。このタブで、次のプロパティーフィールドに入力して接続を設定します。
      - **Gateway Host**
      - **Gateway Port**
      - **Program ID**
      - **Repository Destination**
      - **Connection Count**
   b. **Optional** タブをクリックし、次のプロパティーフィールドに入力して接続を設定します。
      - **SAP Router String**
      - **Worker Thread Count**
      - **Minimum Worker Thread Count**

- **Maximum Startup Delay**
- **Repository Map**

c. **SNC** タブをクリックし、次のプロパティーフィールドに入力して接続を設定します。

- **SNC Level of Security**
- **SNC Name**
- **SNC Library Path**

注記

設定の詳細については、SAP のドキュメントを参照してください。

6. これで、サーバー接続をテストする準備が整いました。**Edit SAP Connection Configuration** ダイアログボックスでサーバー名を右クリックし、**Test** を選択します。
Test Server Connection ダイアログボックスが開きます。

7. ダイアログボックスでは、現在のサーバー設定を使用して **SAP Server Data Store** に接続します。テストが成功すると、ステータス領域に次のメッセージが表示されます。

```
Server state: STARTED
Server state: ALIVE
```

テストが失敗した場合、サーバーのステータスは **DEAD** として報告されます。

8. **Stop** をクリックして、テストサーバーをシャットダウンします。

9. **Close** をクリックして、**Test Server Connection** ダイアログボックスを閉じます。

10. **Finish** をクリックします。新しく作成された **SAP Server Connection** が SAP の下に表示されます。

7.3. 宛先接続およびサーバー接続の削除

概要

このセクションでは、Edit SAP Connection Configuration ダイアログボックスで SAP 宛先接続およびサーバー接続を削除する方法について説明します。

手順

宛先接続とサーバー接続を削除する場合は、次の手順を実行します。

1. ルートエディターのグローバル **Configurations** タブに移動し、**Add** をクリックします。
Create new global element ビューが表示されます。

2. **SAP** で **SAP Connection** を選択し、**Ok** をクリックします。
Edit SAP Connection Configuration ダイアログボックスが表示されます。これにより、宛先およびサーバー接続設定を作成、編集、削除できます。

3. **Edit SAP Connection Configuration** ダイアログボックスで、削除する Destination and Server Data Stores を選択します。

4. **Delete** をクリックします。選択した接続が削除されます。
最後に、**Finish** をクリックします。すべての変更が保存されます。

7.4. 新規 SAP エンドポイントの作成

概要

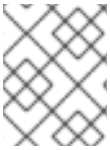
ルートエディターのコンポーネントパレットを使用して、Edit SAP Connection Configuration ダイアログで SAP コンポーネントをルートに追加できます。

注記

SAP Connection ビューを使用している場合は、必ず必要な SAP 接続設定データを Blueprint XML または Spring XML コードに貼り付けてください。

前提条件

Edit SAP Connection Configuration ダイアログを使用して、いくつかの SAP 宛先接続またはサーバー接続 (あるいはその両方) が作成済みである必要があります。



注記

SAP Connection ビューを使用している場合は、この設定を適切なタイプ (Blueprint XML または Spring XML) のファイルにエクスポートします。

手順

新しい SAP エンドポイントを作成するには、次の手順を実行します。

1. 使用する Fuse プロジェクトと Camel XML ファイル (Blueprint XML または Spring XML 形式のいずれか) が既に存在することを前提としています。

2. ルートエディターで Camel XML ファイルを開きます。Red Hat Fuse SAP Tool Suite をすでにインストールしている場合は、ルートエディターの **Components** パレットの下に SAP コンポーネントが表示されるはずです。次の SAP コンポーネントは、ツールスイートによって提供されます。

   - **SAP IDoc Destination**
   - **SAP IDoc List Destination**
   - **SAP IDoc List Server**
   - **SAP qRFC Destination**
   - **SAP Queued IDoc Destination**
   - **SAP Queued IDoc List Destination**
   - **SAP sRFC Destination**
   - **SAP sRFC Server**

- SAP tRFC Destination
- SAP tRFC Server
  ルートエディターの **Design** タブで、これらのコンポーネントのいずれかをキャンバスにドラッグし、現在の **camelContext** に新しい SAP エンドポイントを作成します。

注記

SAP Netweaver コンポーネントは、Red Hat Fuse SAP Tool Suite に属していません。Apache Camel プロジェクトでホストされています。

3. キャンバスの下部にある **Source** タブをクリックして、ルートエディターの **Source** タブに移動します。ルートの XML ソースが表示されます。
4. SAP エンドポイント URI を指定する場合、宛先名またはサーバー接続名のいずれかを URI 形式で埋め込む必要があります。たとえば、**sap-srfc-destination** コンポーネントの URI 形式は以下のとおりです。

```
sap-srfc-destination:destinationName:rfcName
```

特定の宛先を参照するには、関連する **entry** 要素の **key** 属性の値をこの URI の **destinationName** として使用します。

第8章 データ変換のスタートガイド

システムとデータのインテグレーションに伴う課題の1つは、コンポーネントシステムがさまざまなデータ形式で動作することが多いことです。つまり、受信システムが認識できる形式 (または言語) にメッセージを変換しなければ、システムから別のシステムにメッセージを送信できません。データ変換とは、この変換を表す用語です。

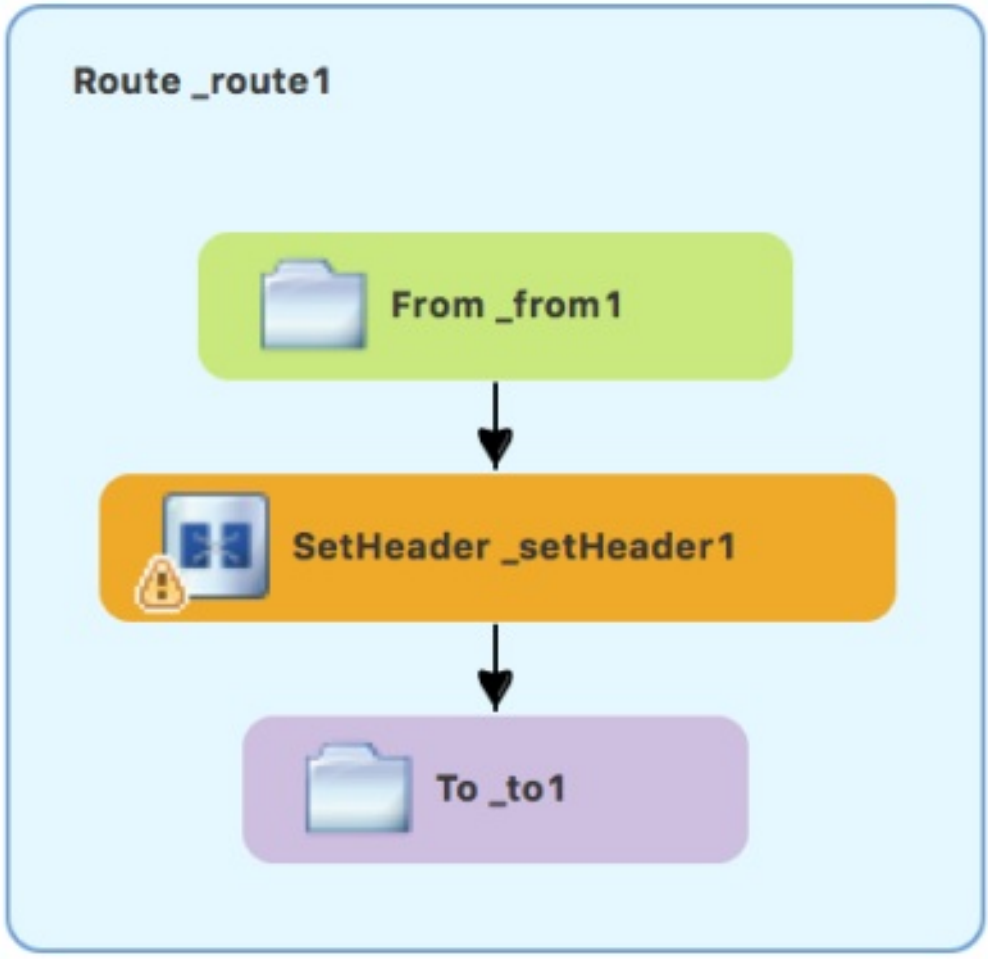
この章では、事前定義された Camel ルートにデータ変換を含める方法を説明します。Camel ルートは、XML データを生成するソースエンドポイントから JSON データを消費するターゲットエンドポイントにメッセージを送信します。ソースの XML データ形式をターゲットの JSON データ形式にマップするデータ変換コンポーネントを追加して定義します。

8.1. データ変換サンプルプロジェクトの作成

1. 新規 Fuse Integration プロジェクトを作成します (**File → New → Fuse Integration Project** を選択します)。
   ウィザードで次の情報を入力します。
   - プロジェクト名: **starter**
   - デプロイメントプラットフォーム: **Standalone**
   - ランタイム環境: **Karaf/Fuse on Karaf**
   - Camel バージョン: デフォルトを使用
   - テンプレート: **Empty - Blueprint DSL**
2. 準備済みデータサンプルを **https://github.com/FuseByExample/fuse-tooling-tutorials/archive/user-guide-11.1.zip** からダウンロードします。
3. **data** フォルダーとそこに含まれる 3 つのファイルを、**user-guide-11.1.zip** アーカイブから Fuse Integration プロジェクトの **src** ディレクトリー (**starter/src/data**) に展開します。
4. **Project Explorer** ビューで、スタータープロジェクトを展開します。
5. **Camel Contexts → src/main/resources/OSGI-INF/blueprint/blueprint.xml** をダブルクリックして、ルートエディターの **Design** タブでルートを開きます。
6. **Source** タブをクリックして基礎となる XML を表示します。
7. **<route id="_route1"/>** を以下のコードに置き換えます。

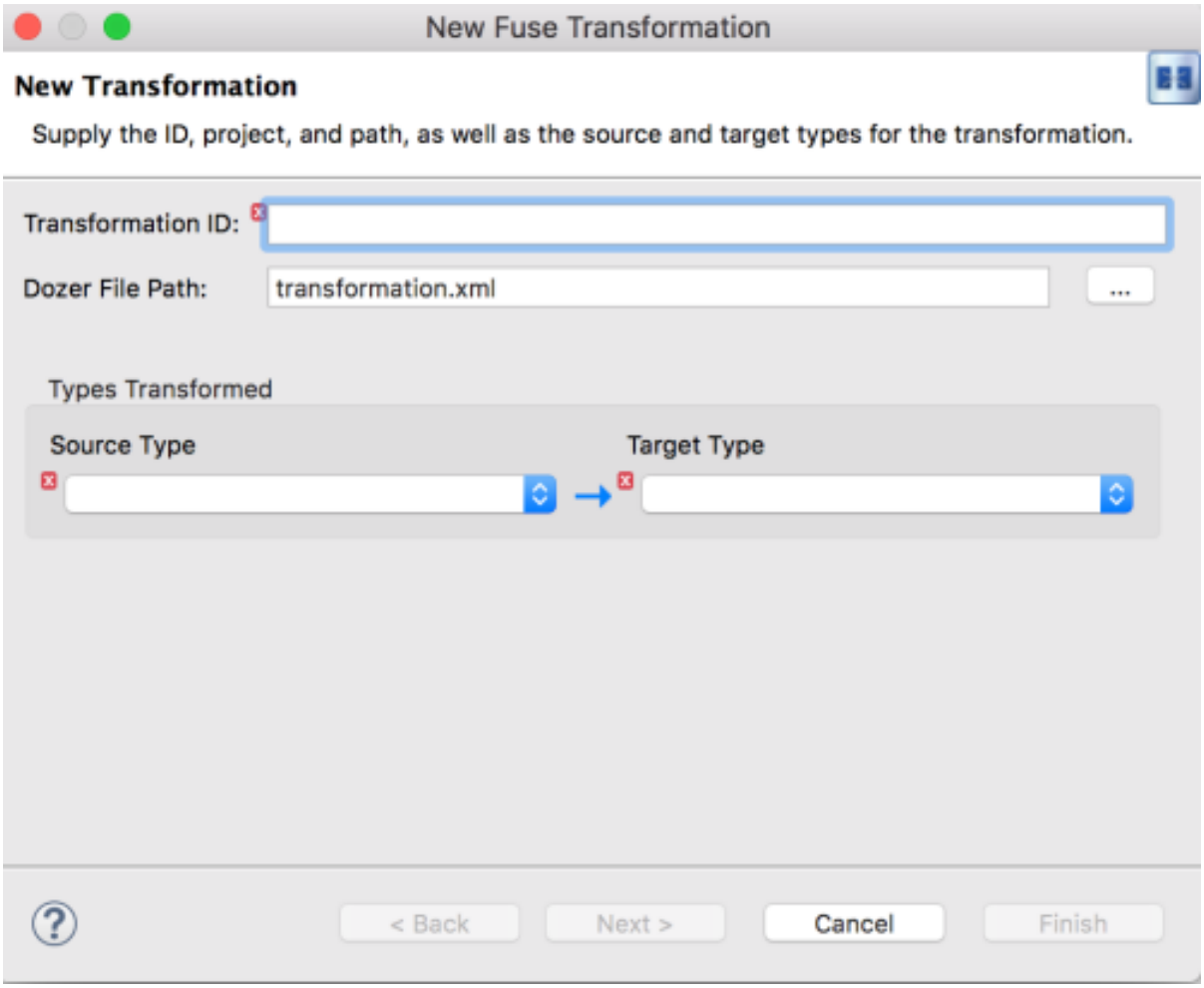
```
<route id="_route1">
  <from id="_from1" uri="file:src/data?fileName=abc-order.xml&amp;noop=true"/>
  <setHeader headerName="approvalID" id="_setHeader1">
    <simple>AUTO_OK</simple>
  </setHeader>
  <to id="_to1" uri="file:target/messages?fileName=xyz-order.json"/>
</route>
```

8. **Design** タブをクリックして、ルートのグラフィカル表示に戻ります。

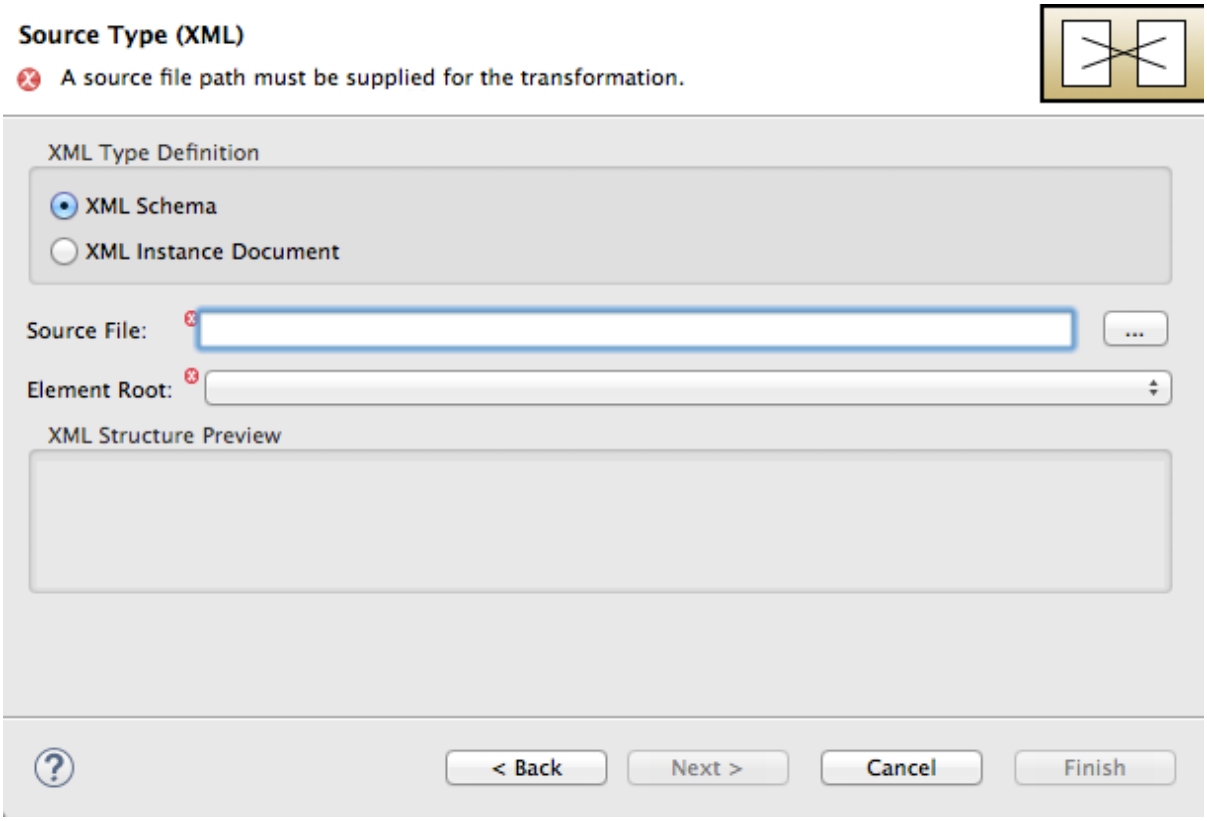


8.2. CAMEL ルートへのデータ変換ノードの追加

1. **Palette** で、**Transformation** ドロワーを展開します。
2. **Data Transformation** パターンをクリックし、キャンバスで **SetHeader _setHeader1** と **To_to1** ノード間の矢印をクリックします。
   New Transformation ウィザードが開き、**Dozer File Path** フィールドが自動入力されます。



3. 残りのフィールドに入力します。
   - Transformation ID フィールド: **xml2json** を入力します。
   - Source Type は、ドロップダウンメニューから XML を選択します。
   - Target Type は、ドロップダウンメニューから JSON を選択します。
4. Next をクリックします。
   Source Type (XML) 定義ページが開き、XML Schema (デフォルト) または XML Instance Document の例を指定して、ソースデータのタイプ定義を指定します。

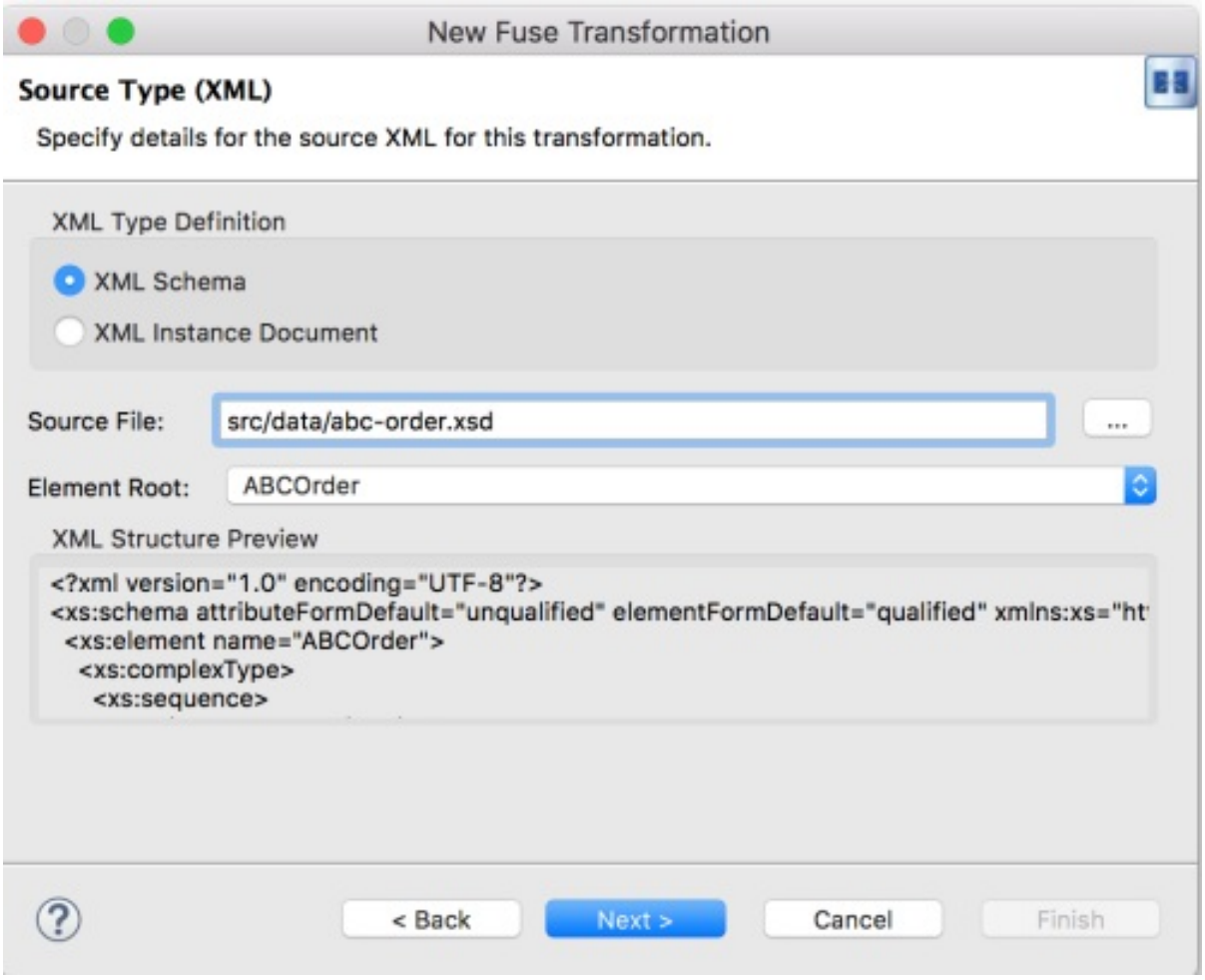


5. **XML Schema** は有効のままにします。

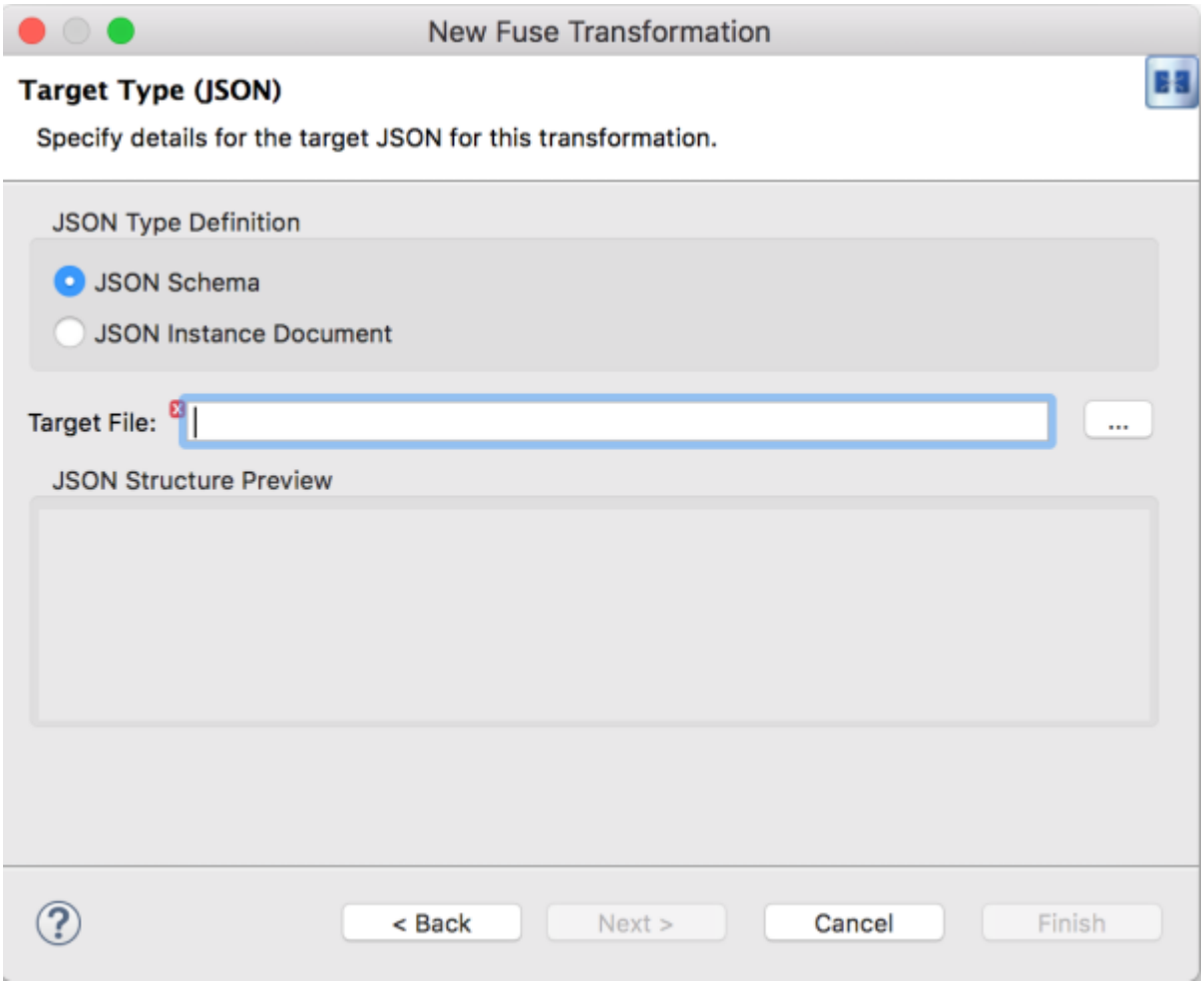
6. **Source file** での場合は、ソースデータの種類定義に使用する XML スキーマファイルまたは XML インスタンスファイルの場所を参照して、選択します (ここでは **abc-order.xsd**)。
XML Structure Preview ペインには、XML 構造のプレビューが表示されます。

7. **Element root** フィールドに **ABCOrder** を入力します。
ツールはこのテキストを使用して、マップするソースデータ項目を表示するペインにラベルを付けます。

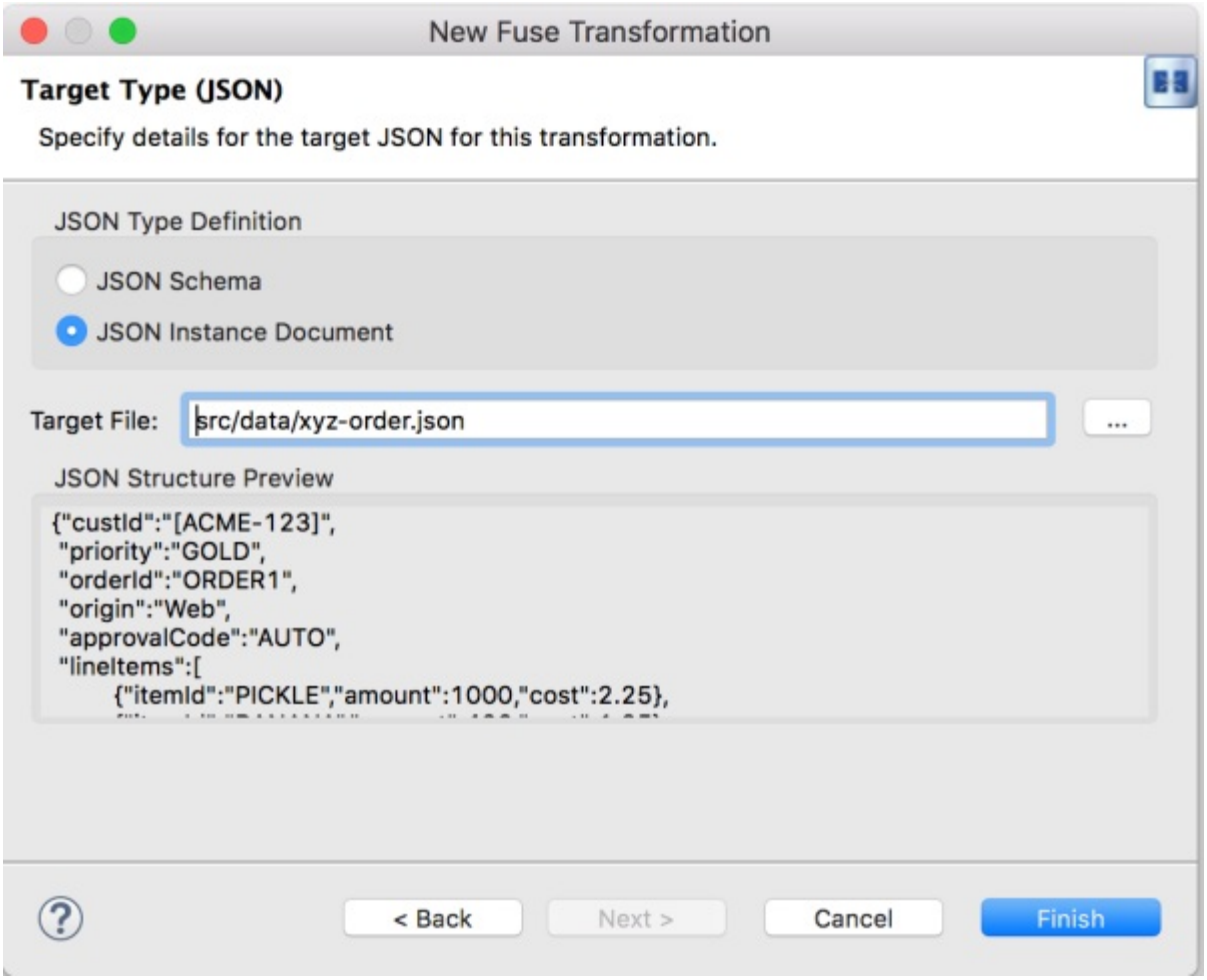
   Source Type (XML) 定義ページは以下のようになります。



8. **Next** をクリックして **Target Type (JSON)** 定義ページを開きます。ここで、ターゲットデータのタイプ定義を指定します。

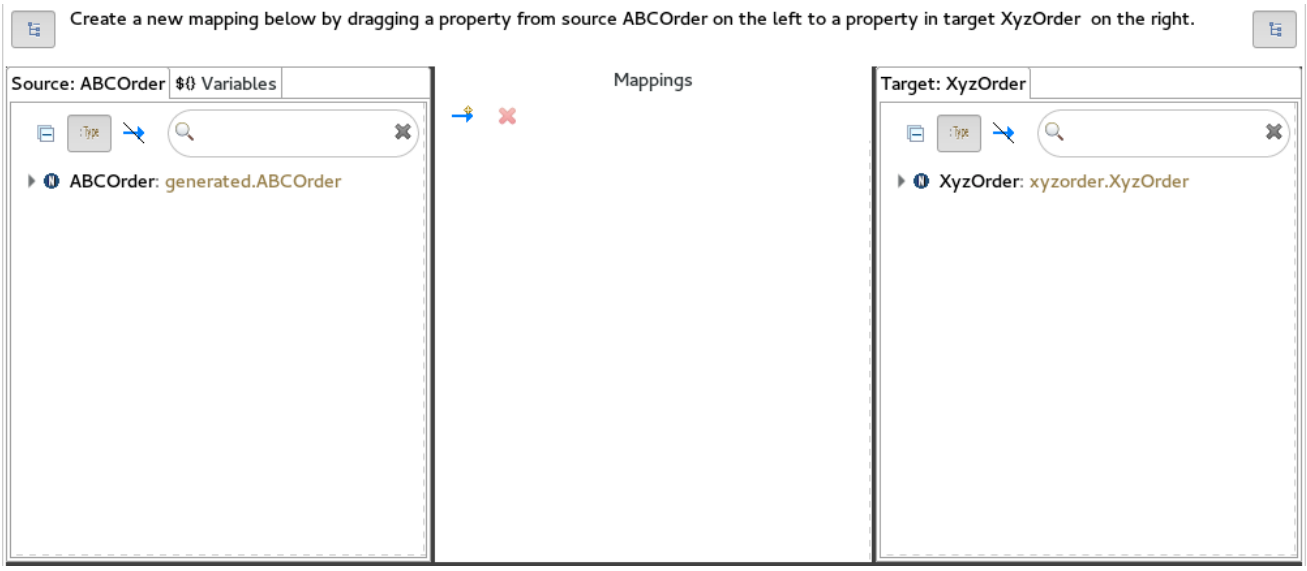


9. JSON Instance Document をクリックします。
Target File フィールドに、**xyz-order.json** インスタンスドキュメントへのパスを入力します。または、このドキュメントを参照します。**JSON Structure Preview** ペインには、JSON データ構造のプレビューが表示されます。



10. **Finish** をクリックします。

変換エディターが開きます。ここで、XML ソースのデータ項目を JSON ターゲットのデータ項目にマップできます。

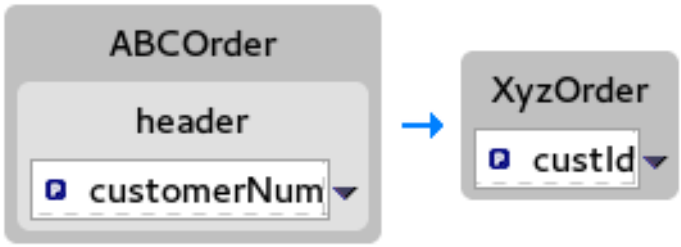


変換エディターは、次の 3 つのパネルで設定されています。

- **Source** – ソースの利用可能なデータ項目が一覧表示されます
- **Mappings** – ソースとターゲットのデータ項目間のマッピングが表示されます

- **Target** – ターゲットの利用可能なデータ項目が一覧表示されます

さらに、エディターの 3 つのパネルのすぐ下にあるエディターの詳細ペイン (最初のマッピングが作成されると表示) には、現在選択されているマッピング済みのソースデータ項目とターゲットデータ項目の両方の階層先がグラフィカル表示されます。以下に例を示します。

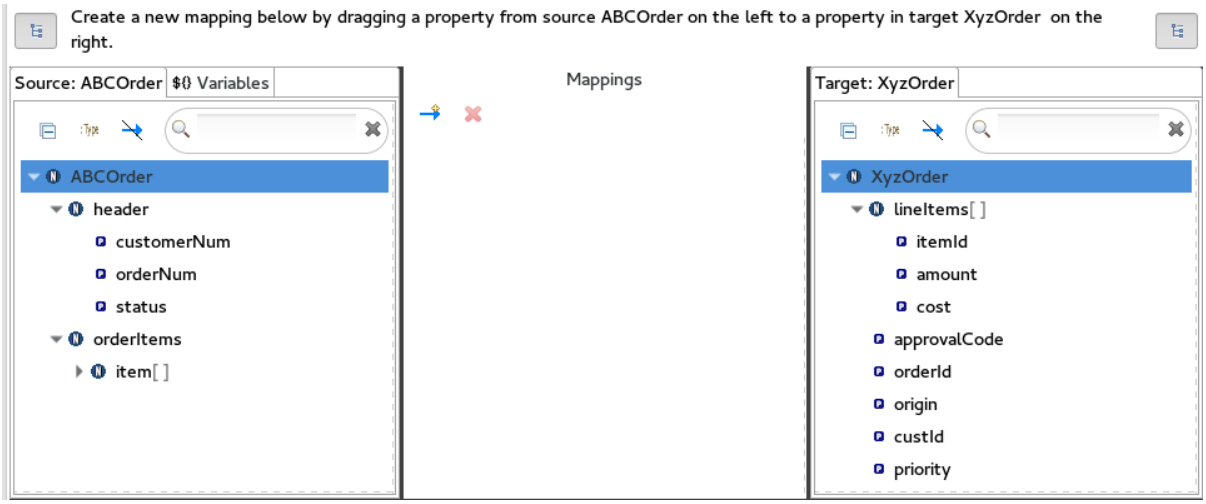


詳細ペインを使用して、選択したソースおよびターゲットのデータ項目のマッピングをカスタマイズできます。

- **Set property** – 既存のマッピングを変更するか、単純なデータ項目をコレクション内の 1 つにマップします (「単純なデータ項目をコレクションのデータ項目にマッピング」 を参照)。
- **Set variable** – データ項目の定数値を指定します (「定数変数のデータ項目へのマッピング」 を参照)。
- **Set expression** – データ項目を指定された式の動的評価にマップします (「式のデータ項目へのマッピング」 を参照)。
- **Add transformation** – ビルトイン関数を使用して、マップされたデータ項目の値を変更します (「マップされたデータ項目へのビルトイン関数の追加」 を参照)。
- **Add custom transformation** – 作成した Java メソッドまたは以前に作成したメソッドを使用して、マップされたデータ項目の値を変更します (「マップされたデータ項目へのカスタム変換の追加」 を参照)。

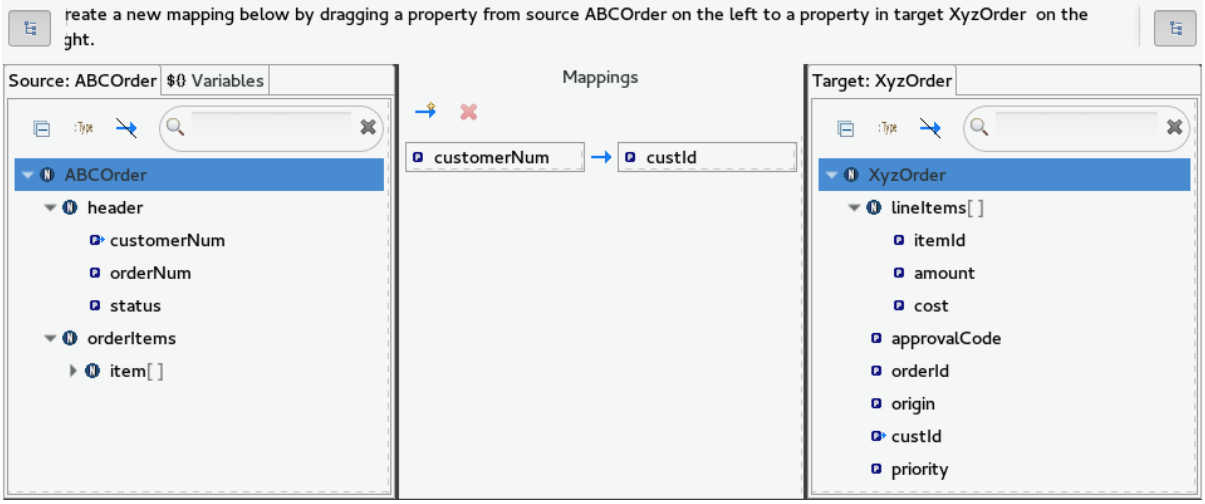
8.3. ソースデータ項目をターゲットデータ項目にマッピング

1. **Mappings** パネルの左右にある **Source** パネルおよび **Target** パネルのすべての項目を展開します。



2. **Source** パネルからデータ項目をドラッグし、**Target** パネルの対応するデータ項目にドロップします。

たとえば、Source パネルから **customerNum** データ項目をドラッグし、Target パネルの **custId** データ項目にドロップします。



マッピングは Mappings パネルに表示され、Source と Target の両方のデータ項目の詳細が下の詳細ペインに表示されます。

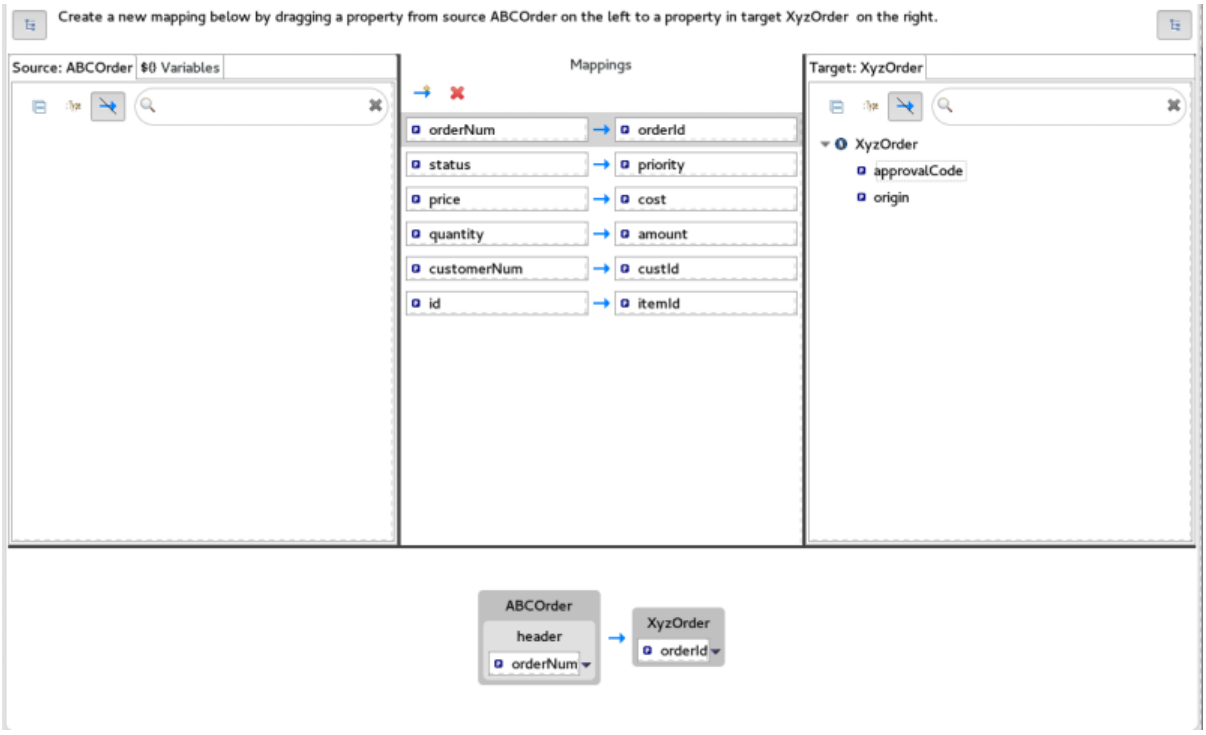
3. すべての基本的なマッピングが完了するまで、ソースデータ項目を対応するターゲットデータ項目にドラッグアンドドロップします。
starter の例では、マッピングする残りのデータ項目は以下のようになります。

ソース	ターゲット
orderNum	**orderId**
status	**priority**
id	**itemId**
price	**cost**
quantity	**amount**

注記

コレクション (リストまたはセットを含むデータ項目) を非コレクションデータ項目にマップしたりその逆にマップしたりすることは可能ですが、コレクションを他のコレクションにマップすることはできません。

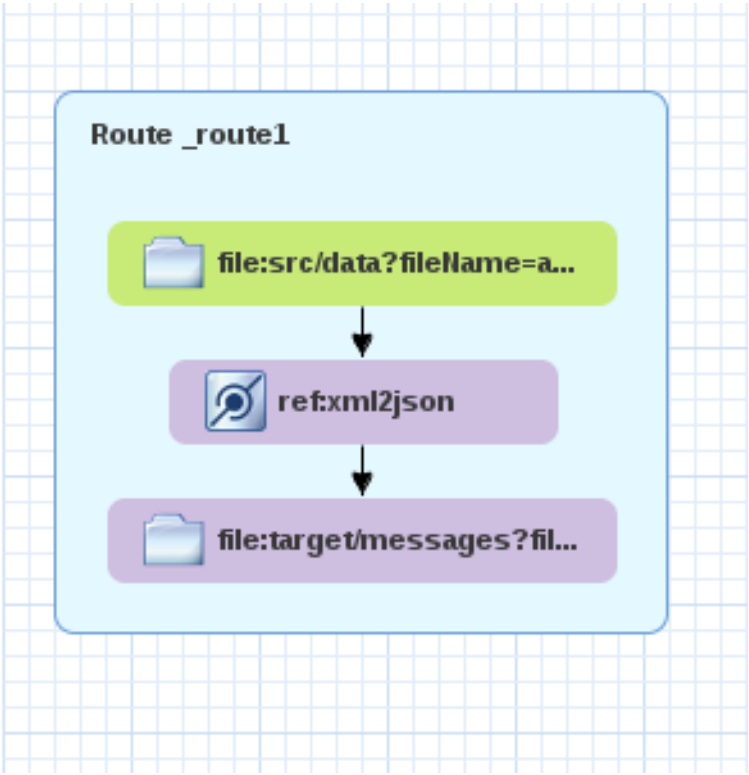
4. Source パネルと Target パネルの両方で  をクリックし、すべてのデータ項目がマッピングされているかどうかを素早く判別します。



マッピングされていないデータ項目のみが **Source** および **Target** パネルに表示されます。

starter の例では、マッピングされていない残りの **Target** 属性は **approvalCode** および **origin** です。

5. **blueprint.xml** タブをクリックして、ルートのグラフィカル表示に戻ります。



6. **File → Save** をクリックします。

変換テストを作成した後、変換ファイルに対して JUnit テストを実行できます。詳細は、「変換テストファイルの作成と JUnit テストの実行」 を参照してください。この時点でこれを行うと、**Console** ビューに次の出力が表示されます。

ソース XML データの場合:

```
<?xml version="1.0" encoding="UTF-8"?>
<ABCOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:java="http://java.sun.com">
    <header>
        <status>GOLD</status>
        <customer-num>ACME-123</customer-num>
        <order-num>ORDER1</order-num>
    </header>
    <order-items>
        <item id="PICKLE">
            <price>2.25</price>
            <quantity>1000</quantity>
        </item>
        <item id="BANANA">
            <price>1.25</price>
            <quantity>400</quantity>
        </item>
    </order-items>
</ABCOrder>
```

ターゲット JSON データの場合:

```
{"custId":"ACME-123","priority":"GOLD","orderId":"ORDER1","lineItems":[{"itemId":"PICKLE",
"amount":1000,"cost":2.25},{"itemId":"BANANA","amount":400,"cost":1.25
```

8.4. 変換テストファイルの作成と JUNIT テストの実行

1. **Project Explorer** ビューで **starter** プロジェクトを右クリックし、**New → Other → Fuse Tooling → Fuse Transformation Test** の順に選択します。
2. **Next** を選択して **New Transformation Test** ウィザードを開きます。
3. **New Transformation Test** ウィザードで、以下の値を設定します。

フィールド	値
Package	**example**
Camel File Path	**OSGI-INF/blueprint/blueprint.xml**
Transformation ID	**xml2json**

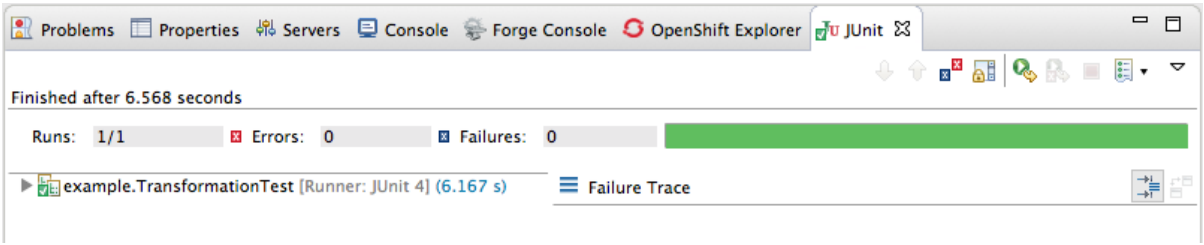
4. **Finish** をクリックします。
5. **Project Explorer** ビューで **starter/src/test/java/example** に移動し、**TransformationTest.java** ファイルを開きます。
6. 以下のコードを **transform** メソッドに追加します。

```
startEndpoint.sendBodyAndHeader(readFile("src/data/abc-order.xml"), "approvalID",
"AUTO_OK");
```

7. **File → Save** をクリックします。
   これで、チュートリアルの任意の時点で、変換ファイルに対して JUnit テストを実行できます。

8. **Project Explorer** ビューで、**starter** プロジェクトを展開して、**/src/test/java/example/TransformationTest.java** ファイルを公開します。

9. それを右クリックしてコンテキストメニューを開き、**Run as JUnit Test** を選択します。
   JUnit Test ペインが開き、テストのステータスが表示されます。ワークスペースが乱雑になるのを防ぐには、**Console** ビューの近くにある下部パネルでペインをドラッグアンドドロップします。



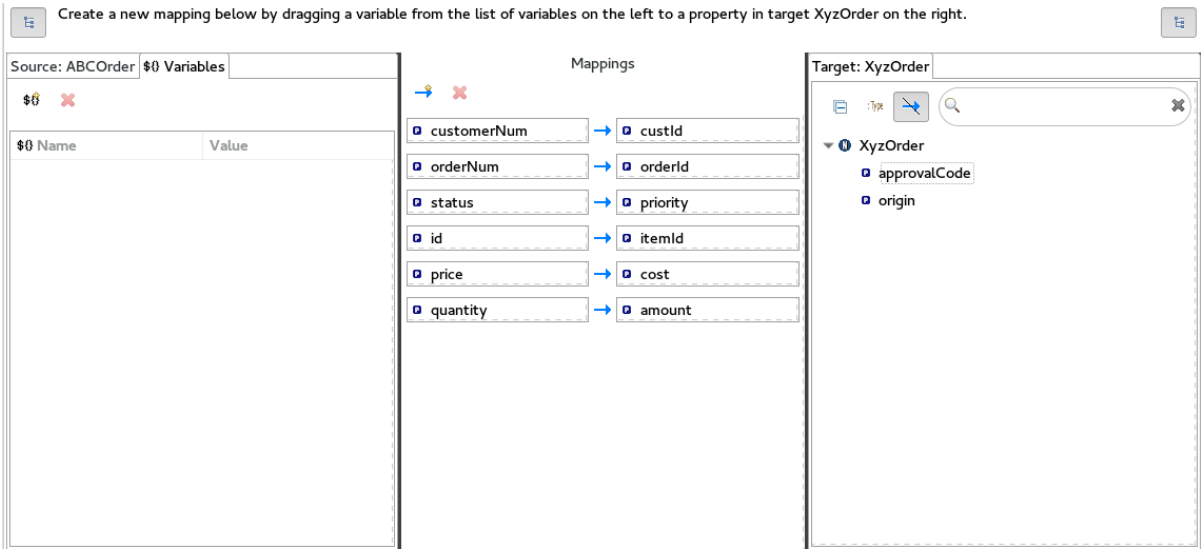
10. **Console** ビューを開き、ログ出力を表示します。

8.5. 定数変数のデータ項目へのマッピング

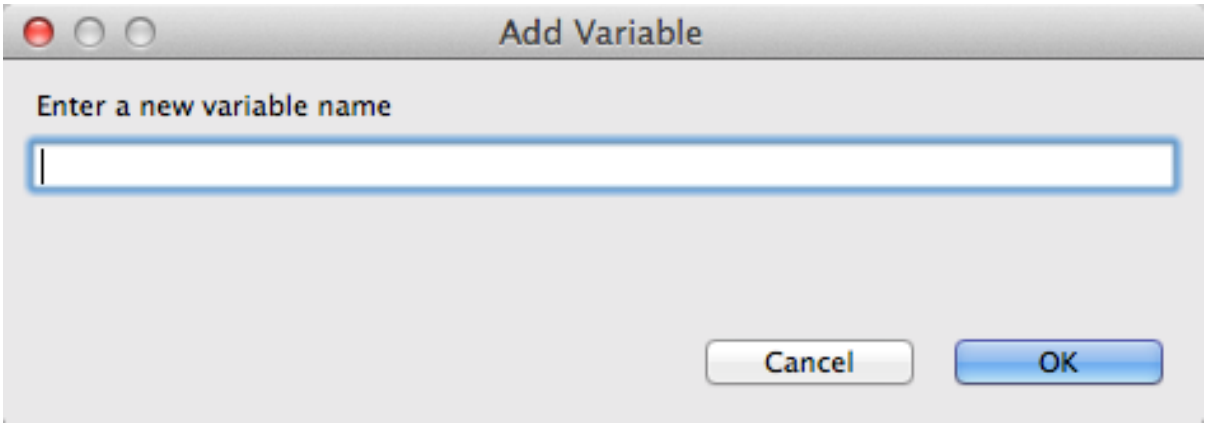
ソース/ターゲットデータ項目に対応するターゲット/ソースデータ項目がない場合は、定数変数を既存のデータ項目にマップできます。

starter の例では、ターゲットデータ項目 **origin** には対応するソースデータ項目がありません。**origin** 属性を定数変数にマッピングするには、以下を実行します。

1. **Source** パネルで **Variables** ビューをクリックします。



2. **Variables** ビューで $0 をクリックし、**Enter a new variable name** ダイアログを開きます。



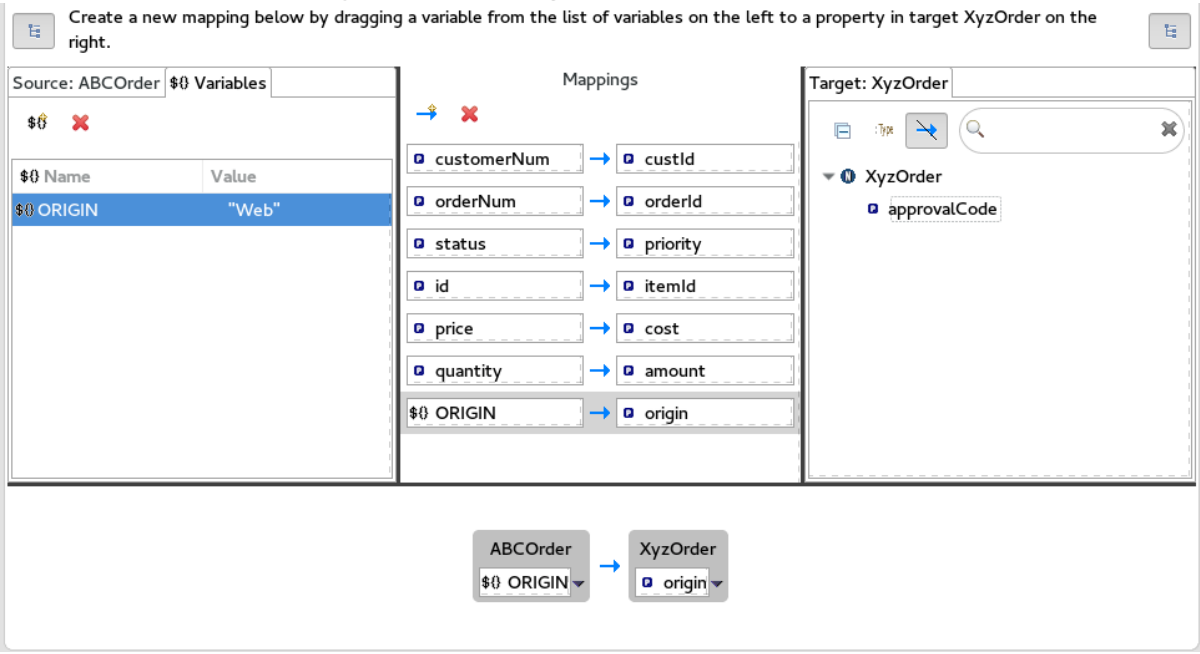
3. 作成する変数の名前を入力します。
starter の例では、**ORIGIN** と入力します。

4. **OK** をクリックします。
新規作成された変数 **ORIGIN** は、Variables ビューの Name 列および Value 列のデフォルト値 **ORIGIN** に表示されます。

5. デフォルト値をクリックして編集し、値を **Web** に変更します。

6. **Enter** キーを押します。

7. 新しい変数 **ORIGIN** を Target パネルの **origin** データ項目にドラッグしてドロップします。



変数 **$(ORIGIN)** の新しいマッピングが **Mappings** パネルと、詳細ペインに表示されます。

8. **TransformationTest.java** ファイルで JUnit テストを実行します。詳細は、「変換テストファイルの作成と JUnit テストの実行」 を参照してください。
Console ビューには、JSON 形式の出力データが表示されます。

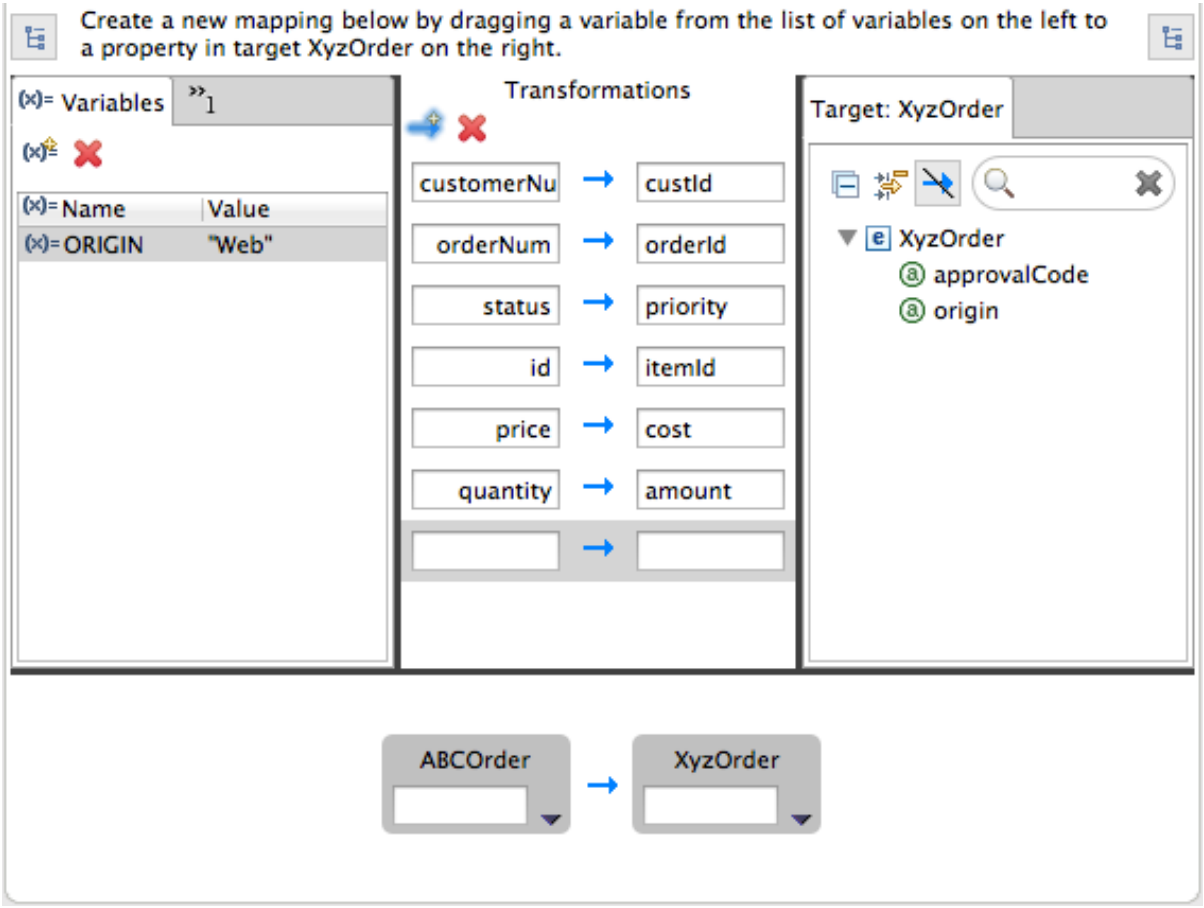
```
{"custId":"ACME-123","priority":"GOLD","orderId":"ORDER1","origin":"Web",
"approvalCode":"AUTO_OK","lineItems":[{"itemId":"PICKLE","amount":1000,"cost":2.25},
{"itemId":"BANANA","amount":400,"cost":1.25}]}
```

8.6. 式のデータ項目へのマッピング

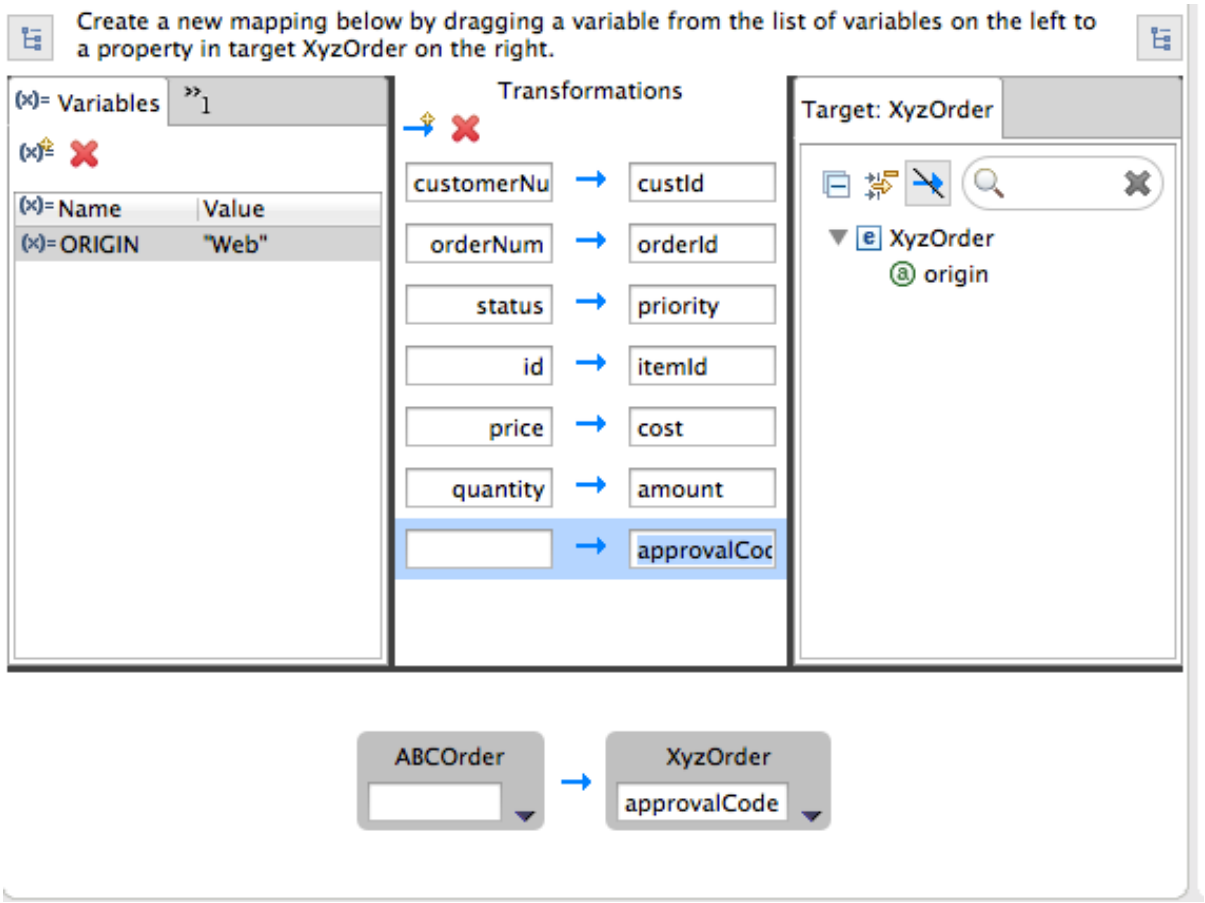
この機能により、ターゲットデータ項目を Camel 言語式の動的評価にマップできます。

対応するソースデータアイテムがないターゲット **approvalCode** データアイテムを使用します。

1. をクリックし、空の変換マッピングを **Mappings** パネルに追加します。

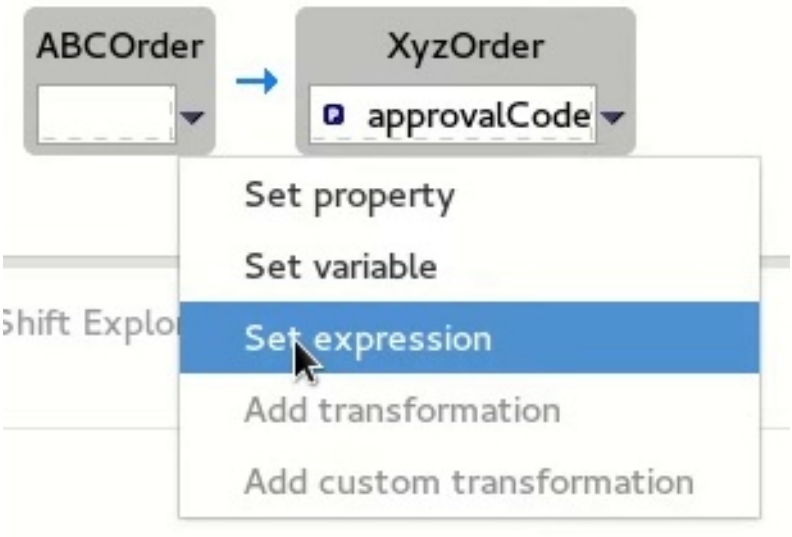


2. **Target** パネルから、**approvalCode** データ項目を **Mappings** パネルで新しく作成されたマッピングのターゲットフィールドにドラッグアンドドロップします。



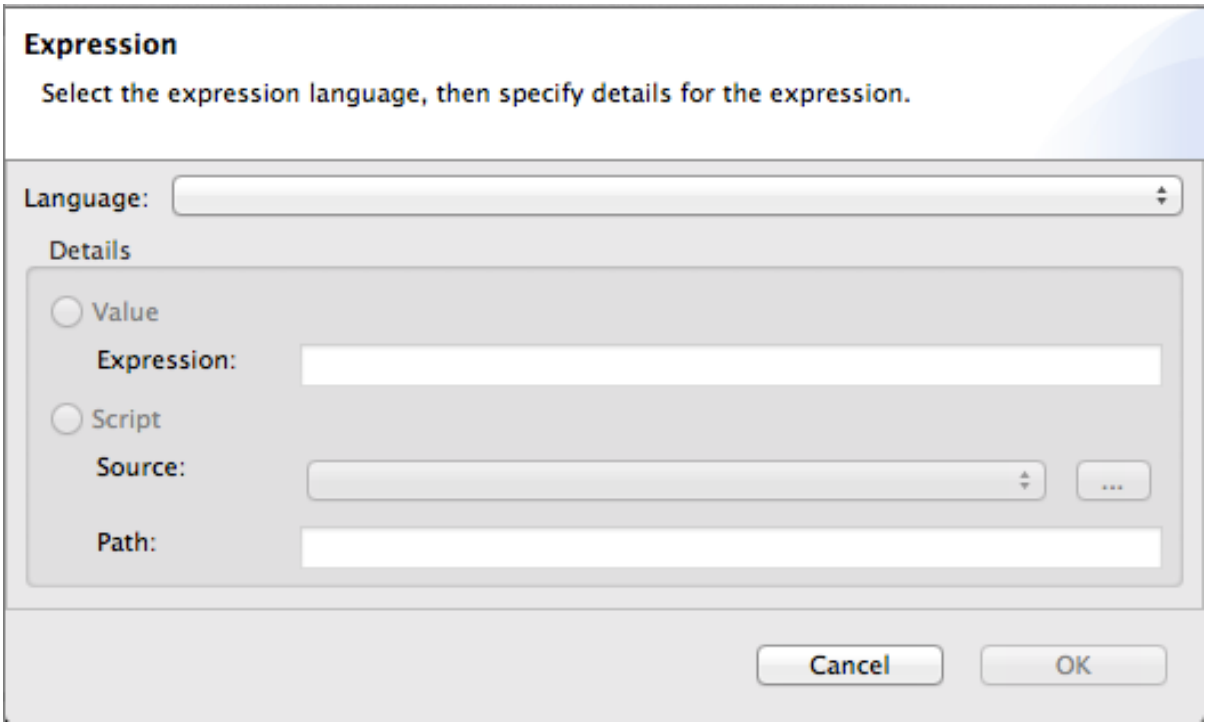
詳細ペインのターゲットボックスにも、**approvalCode** データ項目が表示されます。

3. 詳細ペインで、**ABCOrder** ソースボックスの ▾ をクリックして、ドロップダウンメニューを開きます。



メニューオプションは、選択したデータ項目のデータタイプによって異なります。利用可能なオプションは太字になっています。

4. **Set expression** を選択して、**Expression** ダイアログボックスを開きます。

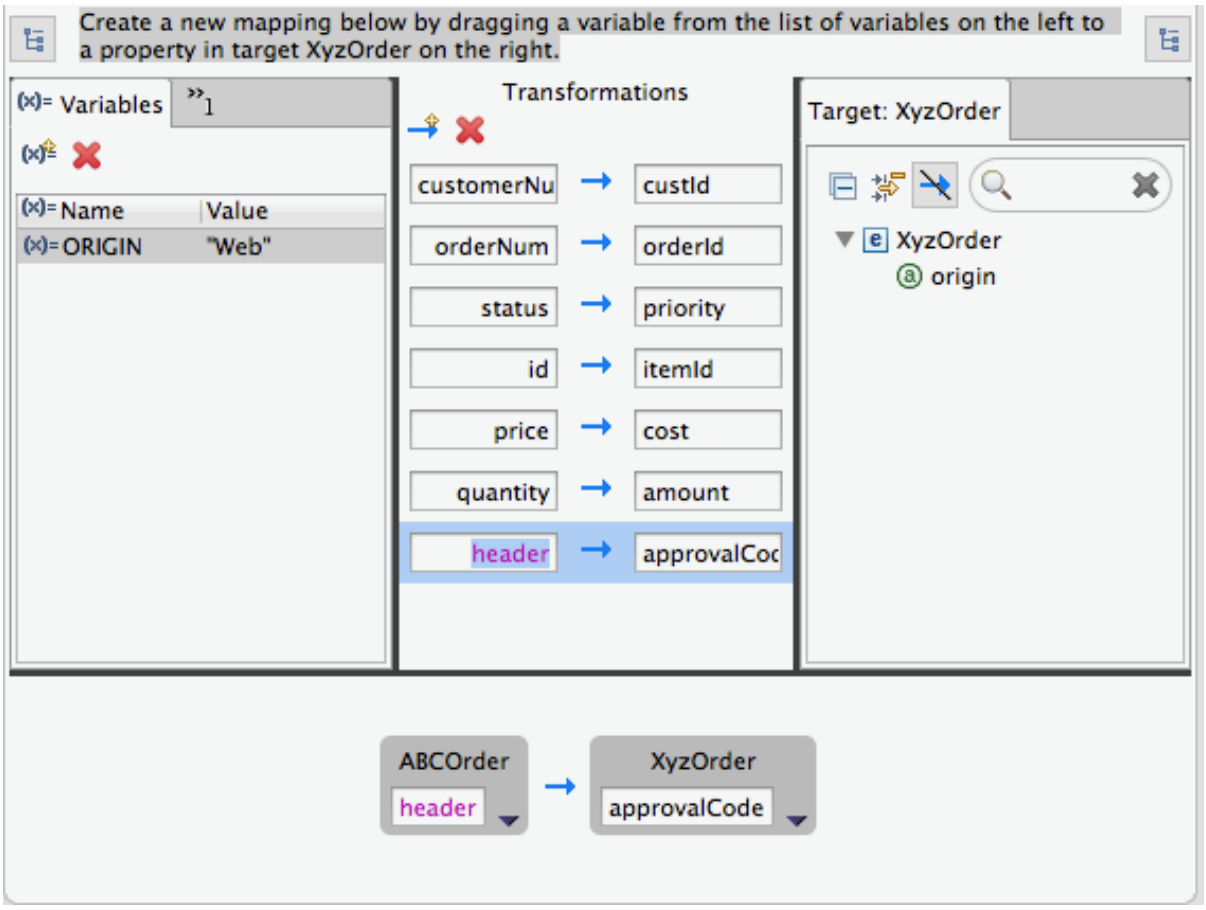


5. **Language** で、利用可能な言語の一覧から使用する式言語を選択します。利用可能なオプションは、データ項目のデータタイプによって異なります。
   starter の例では、**Header** を選択します。

6. 詳細ペインで、使用する式のソースを選択します。
   オプションは **Value** および **Script** です。

   starter の例では、**Value** をクリックしてから **ApprovalID** を入力します。

7. **OK** をクリックします。



Mappings パネルと詳細ペインの両方に、ターゲットデータ項目 **approvalCode** の新しいマッピングが表示されます。

8. **TransformationTest.java** ファイルで JUnit テストを実行します。詳細は、「変換テストファイルの作成と JUnit テストの実行」 を参照してください。
Console ビューには、JSON 形式の出力データが表示されます。

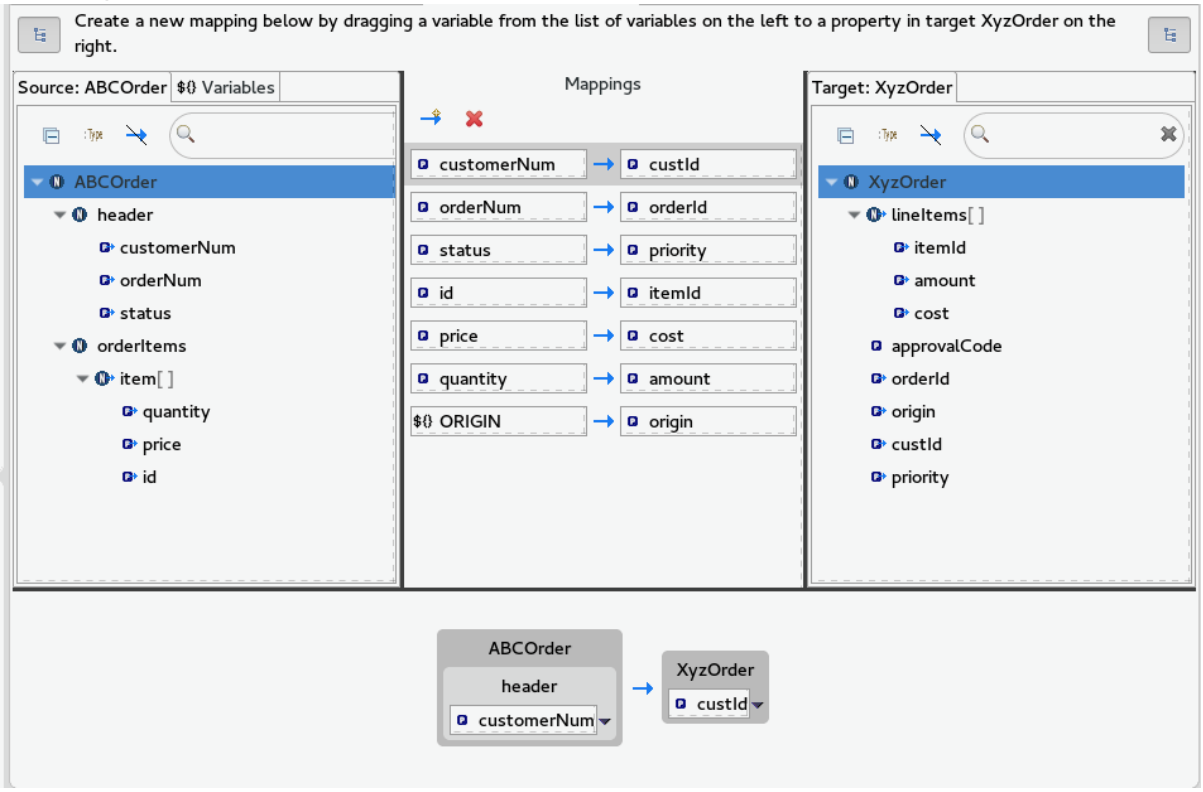
```
{"custId":"ACME-123","priority":"GOLD","orderId":"ORDER1","origin":"Web",
"approvalCode":"AUTO_OK","lineItems":[{"itemId":"PICKLE","amount":1000,"cost":2.25},
{"itemId":"BANANA","amount":400,"cost":1.25}]}
```

8.7. マップされたデータ項目へのカスタム変換の追加

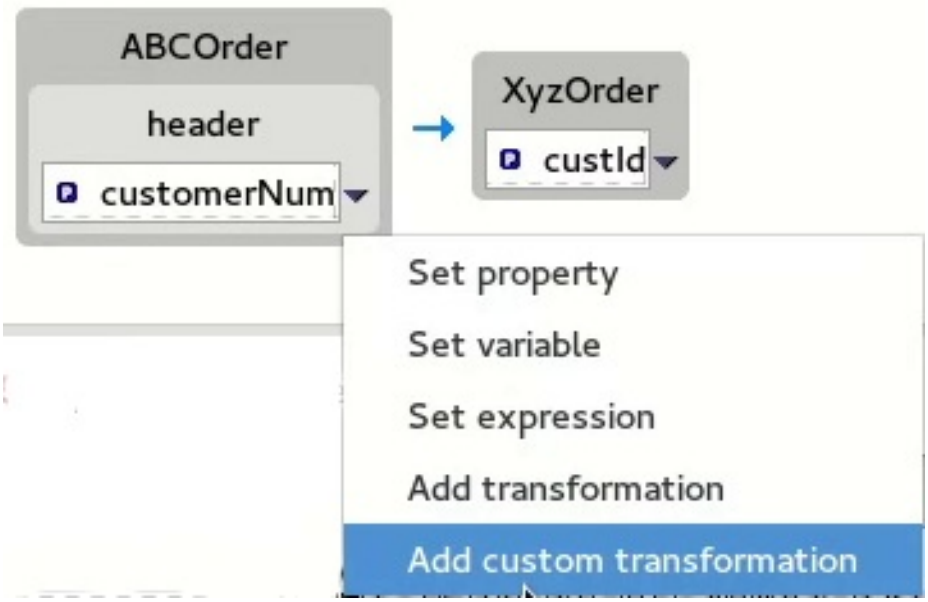
ターゲットシステムの要件を満たしていなければ、ソースデータアイテムのフォーマットを変更する必要がある場合もあります。

たとえば、すべての顧客 ID を括弧で囲むというターゲットシステムの要件を満たすには、以下を実行します。

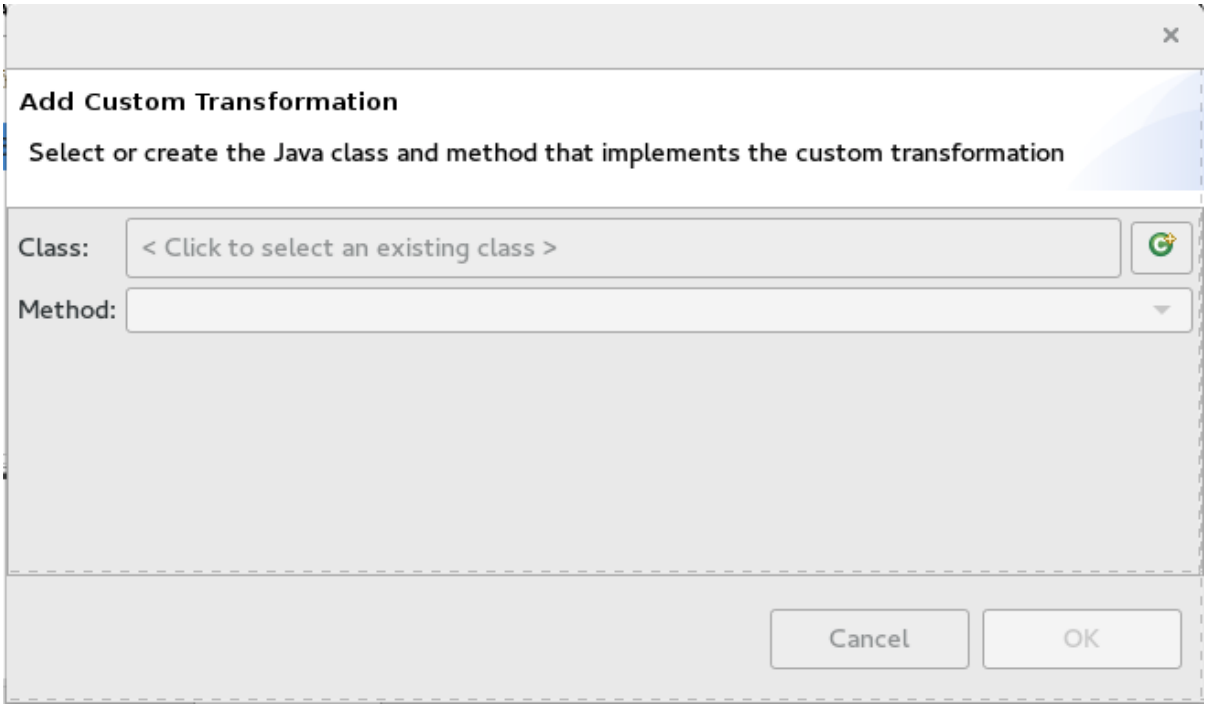
1. Mappings パネルで **customerNum** マッピングを選択し、詳細ペインに反映させます。



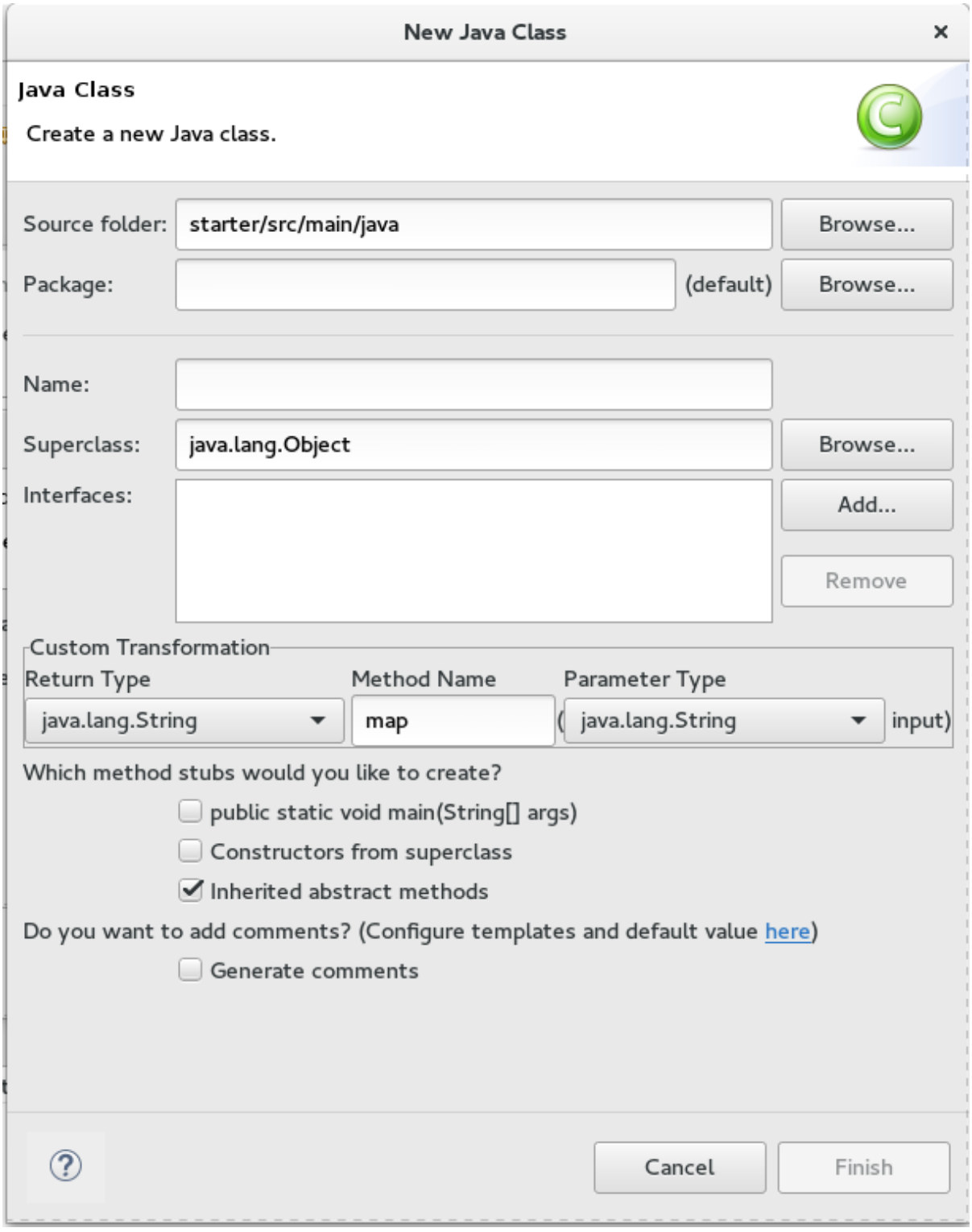
2. 詳細ペインで、**ABCOrder** ソースボックスの をクリックして、ドロップダウンメニューを開きます。



3. **Add custom transformation** を選択して、**Add Custom Transformation** ページを開きます。



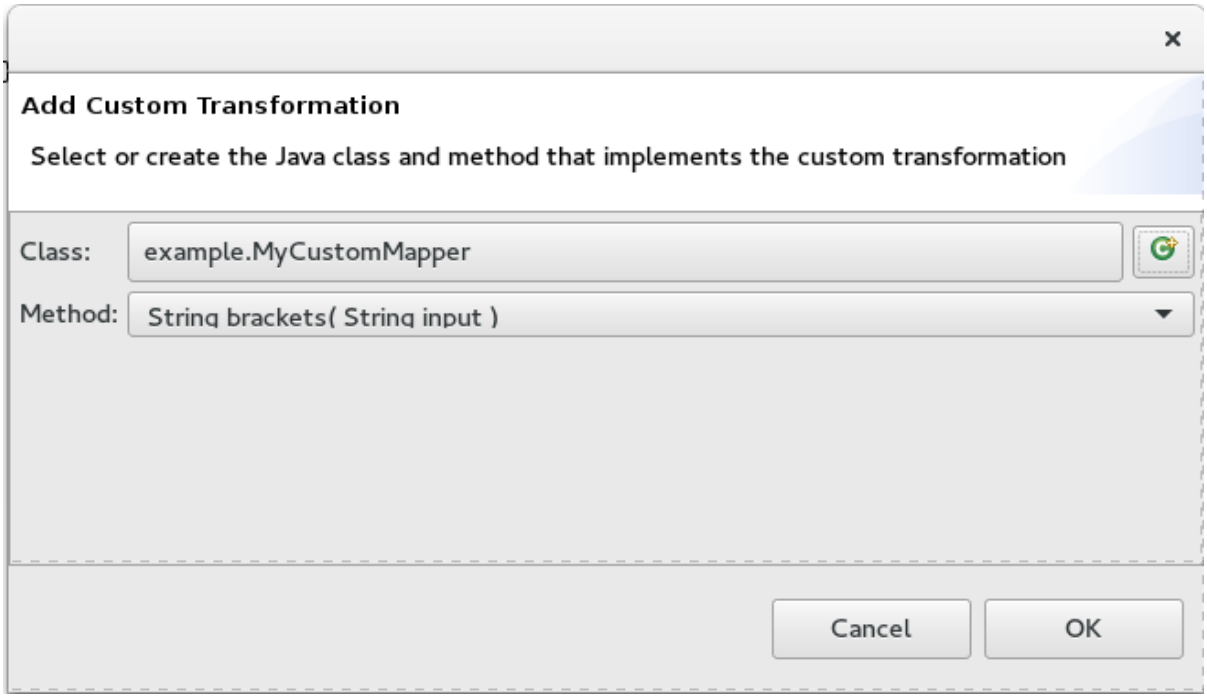
4. **Class** フィールドの横にある  をクリックして、**Create a New Java Class** ウィザードを開きます。



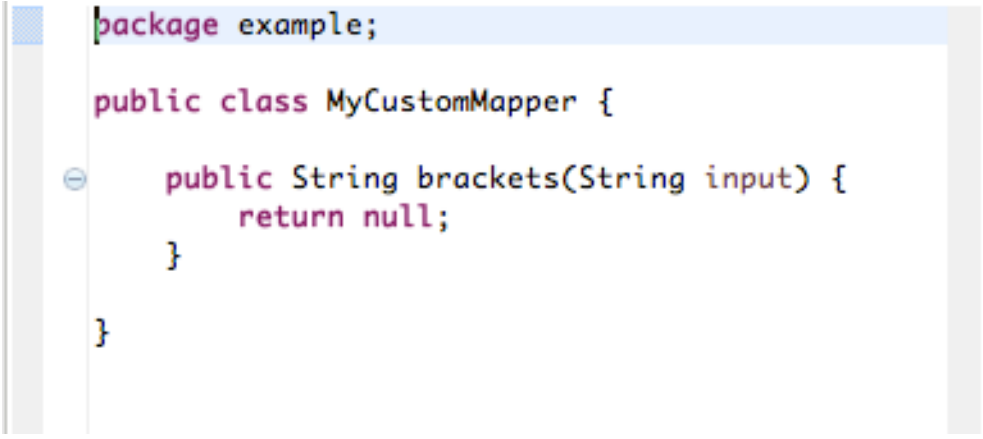
5. 以下のフィールドを変更します。

   - Package: **example** を入力します。
   - Name: **MyCustomMapper** を入力します。
   - Method Name: **map** を **brackets** に変更します。
     その他のフィールドはすべてそのままにします。

6. **Finish** をクリックします。
   Add Custom Transformation ページが開き、**Class** および **Method** フィールドが自動的に入力されます。



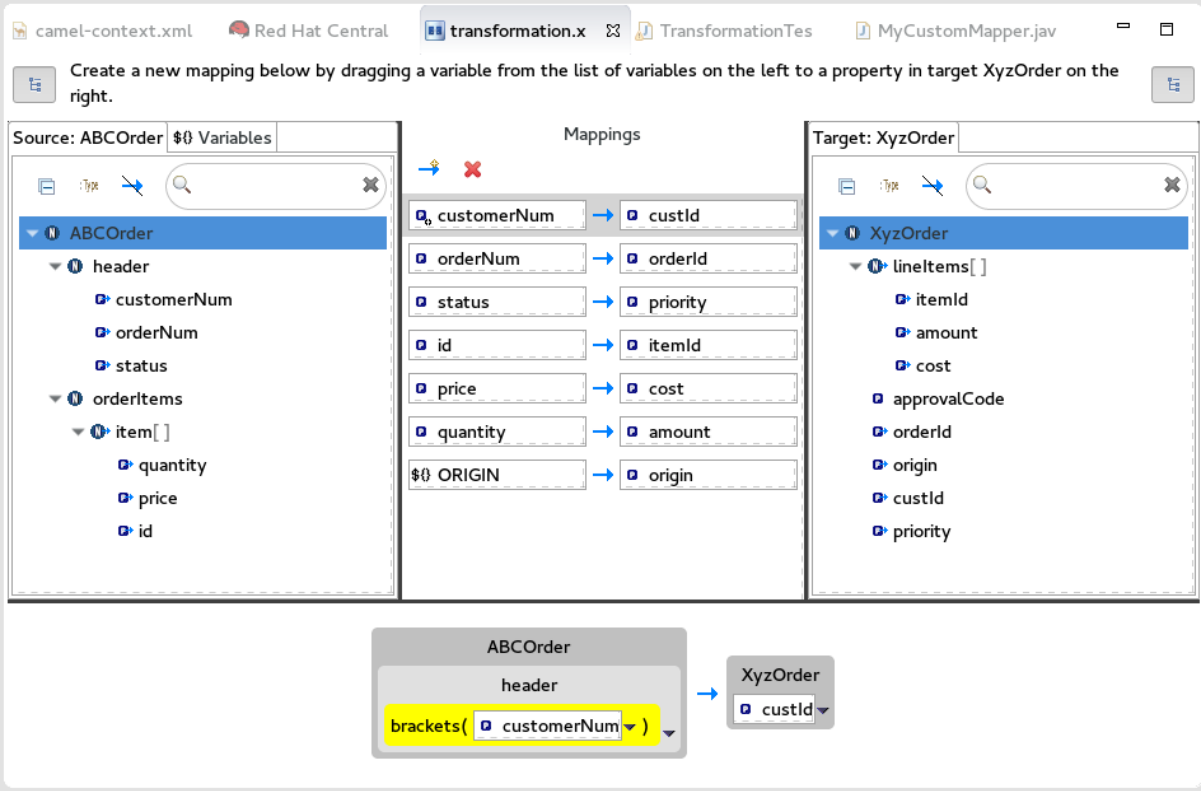
7. **OK** をクリックして Java エディターで **MyCustomMapper.java** ファイルを開きます。



8. **brackets** メソッドを編集して、最後の行 **return null;** を以下のように変更します。

```
return "[" + input + "]";
```

9. **transformation.xml** タブをクリックし、変換エディターに戻ります。



詳細ペインは、**brackets** メソッドが **customerNum** のデータ項目に関連付けられていることを示しています。

この **brackets** メソッドは、ターゲットシステムに送信される前にソース入力で実行されます。

10. **TransformationTest.java** ファイルで JUnit テストを実行します。詳細は、「変換テストファイルの作成と JUnit テストの実行」 を参照してください。
Console ビューには、JSON 形式の出力データが表示されます。

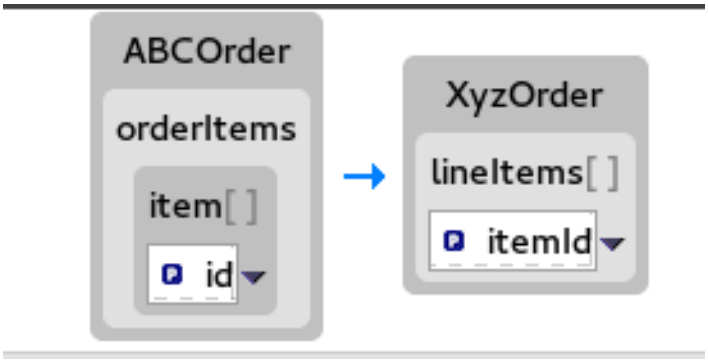
```
{"custId":"[ACME-123]","priority":"GOLD","orderId":"ORDER1","origin":"Web",
"approvalCode":"AUTO_OK","lineItems":[{"itemId":"PICKLE","amount":1000,"cost":2.25},
{"itemId":"BANANA","amount":400,"cost":1.25}]}
```

8.8. 単純なデータ項目をコレクションのデータ項目にマッピング

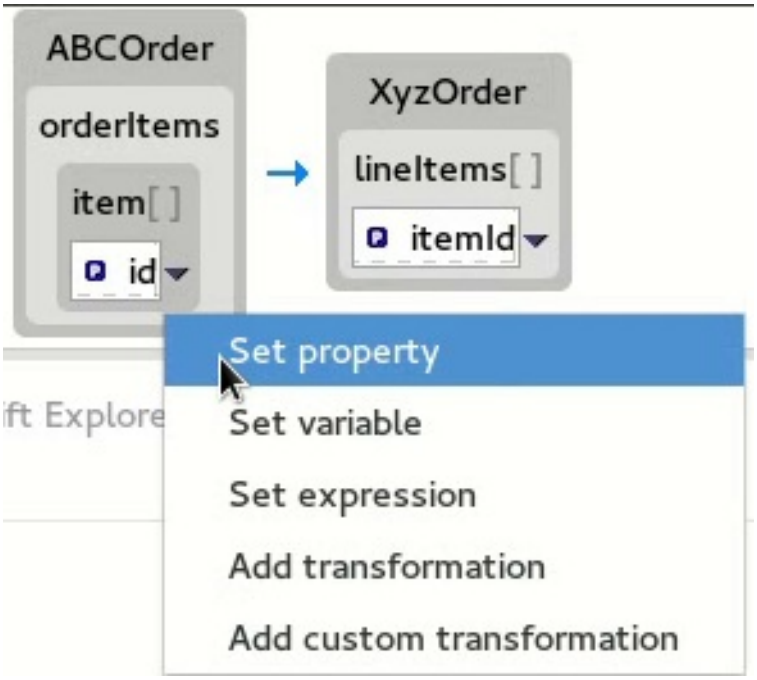
このチュートリアルでは、Source のすべての **id** を Target の **itemId** にマッピングする既存のマッピングを変更します。新しいマッピングは、Source の **customerNum** データ項目を、Target の **lineItems** コレクションの 2 番目の項目である **itemId** にマッピングします。

この変更により、ソースの **id** は Target の **itemId** にマッピングされません。

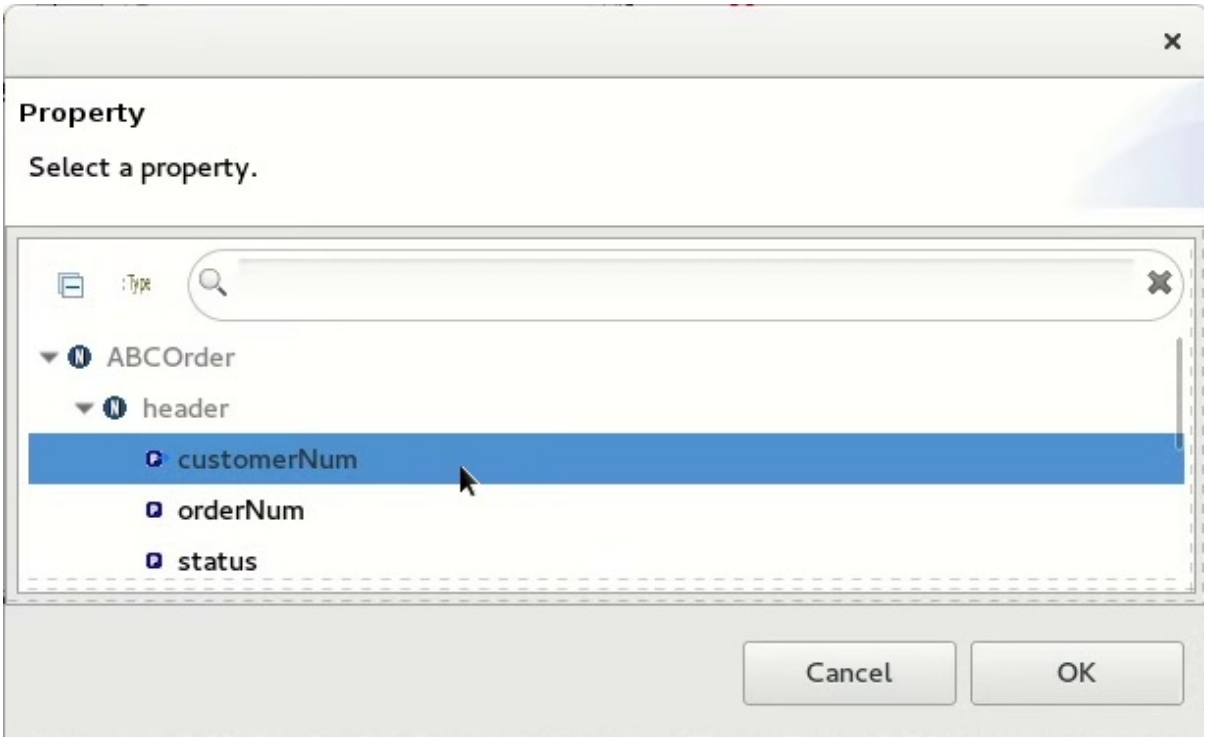
1. **Mappings** パネルでマッピング **id** - > **itemId** を選択し、詳細ペインにマッピングを表示します。



2. Source ボックスで ▾ をクリックし、ドロップダウンメニューを開き、**Set property** を選択します。



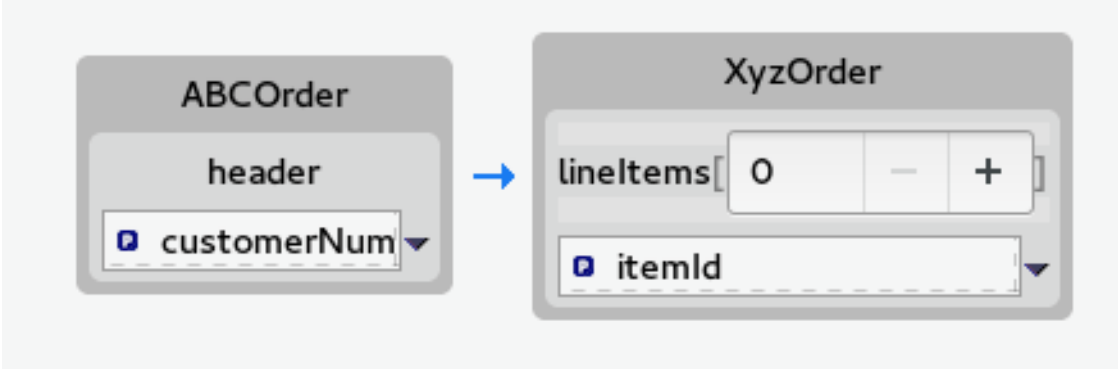
3. **Select a property** ページで **header** ノードを展開し、**customerNum** を選択します。OK をクリックして変更を保存します。



4. 詳細ペインには、**XyzOrder** に **lineItems** フィールドがあることが表示されるようになりました。**lineItems** の横にあるトグルボタンをクリックして、その値を **1** に増やします。

注記

インデックスはゼロがベースであるため、**1** の値はコレクション内の 2 番目のインスタンスである **itemId** を選択します。



詳細ペインには、**lineItems** コレクションの 2 番目の項目である **itemId** にマッピングされた **customerNum** が表示されます。

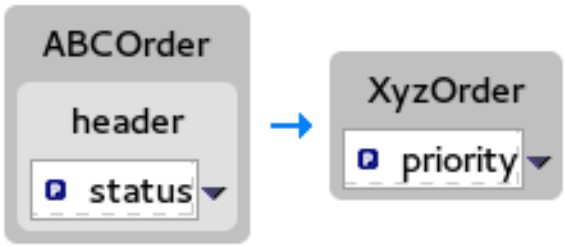
5. **TransformationTest.java** ファイルで JUnit テストを実行します。詳細は、「変換テストファイルの作成と JUnit テストの実行」 を参照してください。
Console ビューには、JSON 形式の出力データが表示されます。

```
{"custId":"[ACME-123]","priority":"GOLD","orderId":"ORDER1","origin":"Web",
"approvalCode":"AUTO_OK","lineItems":[{"amount":1000,"cost":2.25},
{"itemId":"ACME-123","amount":400,"cost":1.25}]}
```

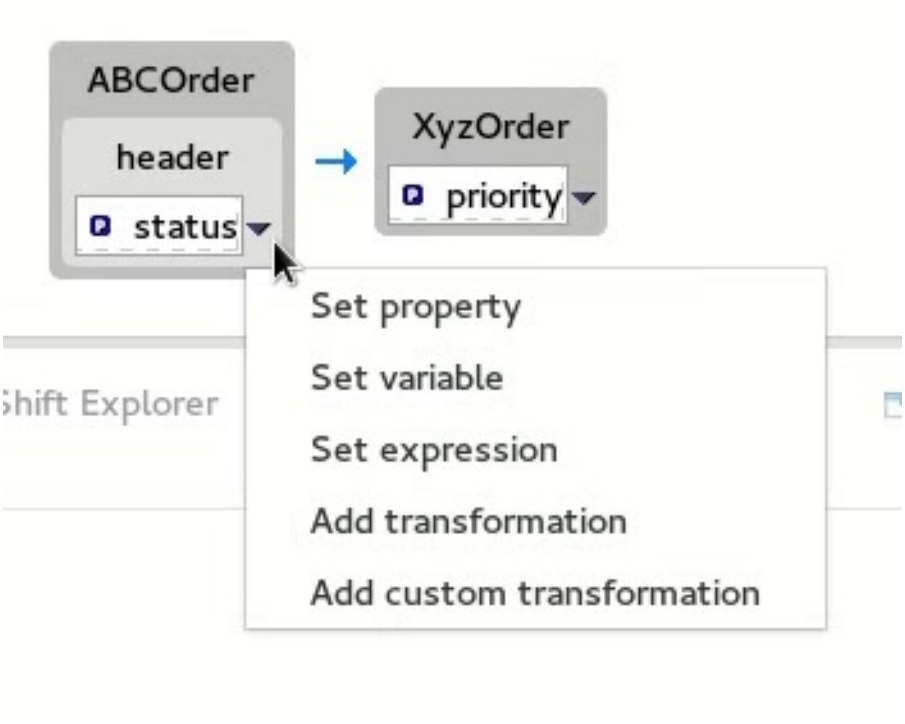
8.9. マップされたデータ項目へのビルトイン関数の追加

ビルトインの文字列関連関数を使用して、マップされたデータ項目に変換を適用できます。

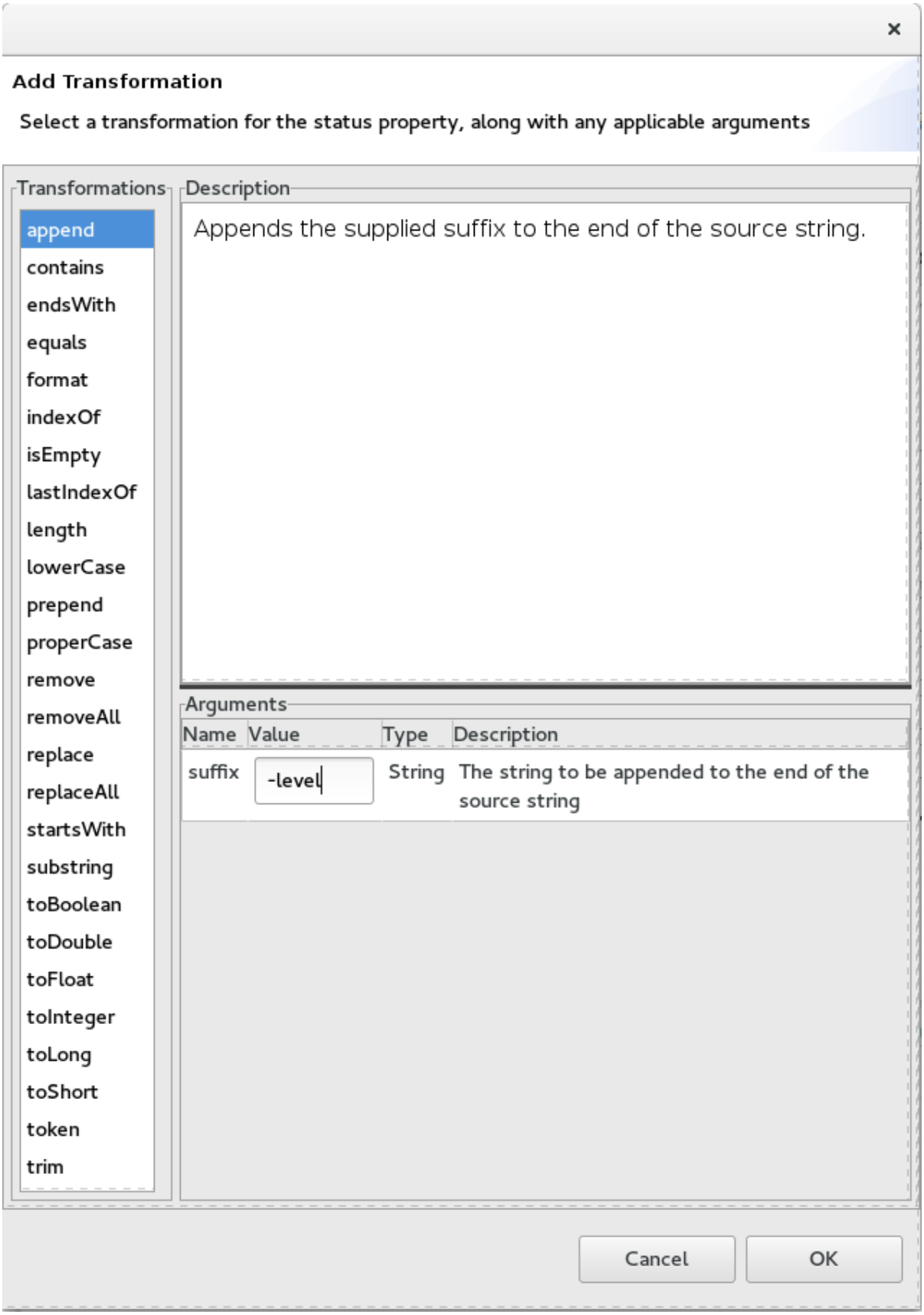
1. **Transformations** パネルで **status** から **priority** へのマッピングを選択し、詳細ペインに反映させます。



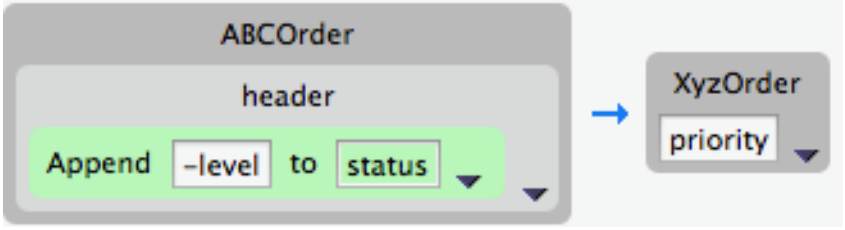
2. Source ボックスで ▾ をクリックし、ドロップダウンメニューを開き、**Add transformation** を選択します。



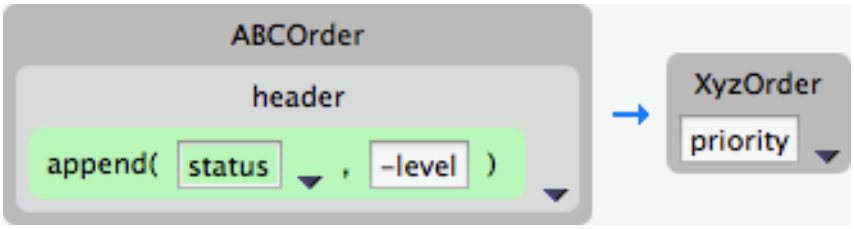
3. Transformations ペインで **append** を選択し、Arguments ペイン で **suffix** の値に **-level** を入力します。
   この **append** 関数は、**status** 文字列をターゲットの **priority** データ項目にマッピングする前に、指定された接尾辞を文字列の最後に追加します。



4. OK をクリックします。



デフォルトでは、詳細ペインには、**append** 関数を **status** データ項目に追加した結果がユーザーフレンドリーな形式で表示されます。このフォーマットを変更するには、Source ボックスの右端の ▾ をクリックし、**Show standard format** を選択します。



5. **TransformationTest.java** ファイルで JUnit テストを実行します。詳細は、「変換テストファイルの作成と JUnit テストの実行」 を参照してください。
Console ビューには、JSON 形式の出力データが表示されます。

```
{"custId":"[ACME-123]","priority":"GOLD-level","orderId":"ORDER1","origin":"Web",
"approvalCode":"AUTO_OK","lineItems":[{"amount":1000,"cost":2.25},{"itemId":"ACME-123",
"amount":400,"cost":1.25}]}
```

8.10. データ変換を使用して FUSE INTEGRATION プロジェクトを RED HAT FUSE サーバーに公開

データ変換プロジェクトを Fuse サーバーに公開する前に (28章*Fuse Integration プロジェクトのサーバーへの公開*を参照)、Fuse ランタイムに以下の機能をインストールする必要があります。

- **camel-dozer**
- **camel-jackson**
- **camel-jaxb**

Fuse ランタイムに必要な機能をインストールするには、以下を行います。

1. まだの場合は **Fuse Integration** パースペクティブに切り替えます。
2. 必要な場合は、Fuse サーバーを **Servers** リストに追加します (「サーバーの追加」 を参照)。
3. Fuse Server (「サーバーの起動」 を参照) を起動し、JBoss Fuse シェルが **Terminal** ビューに表示されるまで待ちます。
4. 必要な **camel-** 関数ごとに、**JBossFuse:admin@root>** のプロンプトで以下を入力します。
features:install camel-<featureName>

   ここで、featureName は、**dozer**、**jackson**、または **jaxb** のいずれかです。
5. 各機能が正常にインストールされていることを確認するには、**JBossFuse:admin@root>** のプロンプトで以下のコマンドを入力します。
features:list --ordered --installed

   出力一覧に先ほどインストールした camel 機能が表示されるはずです。

```
[installed  ] [2.17.0.redhat-630159] camel-dozer                camel-2.17.0.redhat-630159
[installed  ] [2.17.0.redhat-630159] camel-exec                 camel-2.17.0.redhat-630159
[installed  ] [2.17.0.redhat-630159] camel-ftp                  camel-2.17.0.redhat-630159
[installed  ] [2.17.0.redhat-630159] camel-jackson              camel-2.17.0.redhat-630159
[installed  ] [2.17.0.redhat-630159] camel-jasypt               camel-2.17.0.redhat-630159
[installed  ] [2.17.0.redhat-630159] camel-jaxb                 camel-2.17.0.redhat-630159
```

第9章 FUSE ONLINE インテグレーション用エクステンションの開発

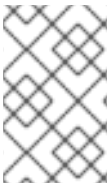
Fuse Online は、アプリケーションを統合するための Web インターフェイスを提供する Red Hat Fuse 機能です。ビジネスエンジニアは、コードを作成せずに、Fuse Online を使用してアプリケーションに接続し、任意で異なるアプリケーションへのコネクション間のデータで操作できます。インテグレーターが必要とする機能を Fuse Online が提供しない場合、開発者は必要な動作を定義するエクステンションを作成できます。

Fuse Tooling を使用して、Fuse Online で使用する機能を提供するエクステンションを開発できます。エクステンションは、以下を定義します。

- インテグレーションでコネクション間のデータを操作する 1 つ以上のカスタム **ステップ** または
- 1 つのカスタム **コネクター**

Fuse Online では、**コネクター** はデータの取得元または送信先となる特定のアプリケーションを表します。各コネクターは、その特定のアプリケーションへの接続を作成するためのテンプレートです。たとえば、Salesforce コネクターは Salesforce への接続を作成するためのテンプレートです。Fuse Online で Fuse Online ユーザーが必要とするコネクターが提供されていない場合、カスタムコネクターを定義するエクステンションを開発できます。

Fuse Online では、インテグレーションのコネクション間で発生するデータ操作を **ステップ** と呼びます。Fuse Online は、データのフィルタリングやマッピングなどの操作のステップを提供します。Fuse Online のビルトインステップでは提供されない方法で接続間のデータを操作するために、1 つ以上のカスタムステップを定義する Fuse Online エクステンションを開発できます。



注記

任意の IDE でエクステンションを開発することもできます。Fuse Tooling と別の IDE のどちらを使用するかは、完全に個人が選択できます。IDE でのエクステンションの開発に関する情報は、Fuse Online でのアプリケーションの統合 を参照してください。

9.1. タスクの概要

以下は、Fuse Online エクステンションを開発するためのタスクの概要です。

1. Fuse Online エクステンションプロジェクトを作成し、エクステンションタイプとして **Custom Connector** または **Custom Step** を選択します。
2. エクステンションのタイプに応じて、エクステンションのコードを記述します。
   - **Custom Connector**: ベース Camel コンポーネント、コネクターアイコン、グローバルコネクタープロパティー、およびコネクターアクションを定義します。
   - **Custom Step**: ルートの追加、アクションの定義、および依存関係の指定を行います。
3. **.jar** ファイルをビルドします。
4. **.jar** ファイルを Fuse Online ユーザーに提供します。

Fuse Online ユーザーは **.jar** ファイルを Fuse Online にアップロードし、カスタムコネクターまたはカスタムステップを使用できるようにします。Fuse Online とインテグレーションの作成方法は、Fuse Online でのアプリケーションの統合 を参照してください。

9.2. 前提条件

開始する前に、以下の情報および知識が必要です。

- Fuse Online カスタムコネクターまたはステップに必要な機能の説明 (Fuse Online ユーザーから)。
- エクステンションの Fuse Online バージョン番号。
- カスタムコネクターの場合、PNG または SVG 形式のアイコンイメージファイル。Fuse Online は、このアイコンをインテグレーションのフローを表示するときに使用します。アイコンを指定しない場合、エクステンションが含まれる .jar がアップロードされると Fuse Online が 1 つ作成します。
- 以下に精通している必要があります。
  - Fuse Online
  - Spring Boot XML または Java
  - Apache Camel ルート (ルートベースのステップエクステンションを作成する場合)
  - JSON
  - Maven

9.3. カスタムコネクターの作成

Fuse Online では、カスタムコネクターは 1 つ以上の接続設定パラメーター、1 つ以上の接続アクション、およびアクションの任意の設定パラメーターで設定されます。

カスタムコネクターを開発するためのタスクの概要を以下に示します。

1. Fuse Online エクステンションプロジェクトを作成し、エクステンションタイプに **Custom Connector** を選択します。
2. エクステンションのコードを記述します。ベース Camel コンポーネント、コネクターアイコン、グローバルコネクタープロパティー、およびコネクターアクションを定義します。

9.3.1. カスタムコネクターのコードの記述

Fuse Online エクステンションプロジェクトを作成した後、Fuse Online ユーザーによって提供される必要な機能の説明を基にして、カスタムコネクター要素を定義するコードを記述します。表9.1「カスタムコネクター要素」 の表は、Fuse Tooling で作成したカスタムコネクター要素が Fuse Online 要素にどのように対応するか示しています。

表9.1 カスタムコネクター要素

Fuse Tooling 要素	Fuse Online 要素	説明
グローバル (最上位) プロパティー	接続設定パラメーター	Fuse Online ユーザーがこのコネクターから接続を作成すると、ユーザーは接続設定の一部としてこのプロパティーの値を指定します。

Fuse Tooling 要素	Fuse Online 要素	説明
Action	接続アクション	Fuse Online では、Fuse Online ユーザーはこのコネクターから作成された接続に対して、アクションの 1 つを選択します。
アクションで定義されたプロパティー	アクション設定パラメーター	Fuse Online ユーザーが接続で実行するアクションを設定する場合、Fuse Online ユーザーはこのプロパティーの値をアクション設定の一部として指定します。

Fuse Online のカスタムコネクターを実装するコードを記述するには、以下を実行します。

1. エディタービューで syndesis-extension-definition.json ファイルを開き、グローバルプロパティー、カスタムコネクターが実行できるアクション、および各アクションのプロパティーを定義するコードを記述します。
   各 **グローバルプロパティー** は Fuse Online の接続設定パラメーターに対応します。各アクションプロパティーは Fuse Online コネクションアクション設定パラメーターに対応します。Fuse Online では、ユーザーがカスタムコネクターを選択すると、Fuse Online が各接続設定パラメーターの値を要求します。カスタムコネクターは、OAuth プロトコルを使用するアプリケーションのカスタムコネクターであることがあります。この場合、OAuth クライアント ID のグローバルプロパティーと OAuth クライアントシークレットの別のグローバルプロパティーを指定するようにしてください。Fuse Online ユーザーは、このコネクターから作成された接続が機能するように、これらのパラメーターの値を指定する必要があります。

   各 **コネクターアクション** はベース Camel コンポーネントスキームを宣言します。

   New Fuse Online Extension Project ウィザードによって提供される例では、**telegram** Camel コンポーネントスキームを使用します。

```
{
  "schemaVersion" : "v1",
  "name" : "Example Fuse Online Extension",
  "extensionId" : "fuse.online.extension.example",
  "version" : "1.0.0",
  "actions" : [ {
    "id" : "io.syndesis:telegram-chat-from-action",
    "name" : "Chat Messages",
    "description" : "Receive all messages sent to the chat bot",
    "descriptor" : {
      "componentScheme" : "telegram",
      "inputDataShape" : {
        "kind" : "none"
      },
      "outputDataShape" : {
        "kind" : "java",
        "type" : "org.apache.camel.component.telegram.model.IncomingMessage"
      },
      "configuredProperties" : {
        "type" : "bots"
      }
```

```
    },
    "actionType" : "connector",
    "pattern" : "From"
  }, {
    "id" : "io.syndesis:telegram-chat-to-action",
    "name" : "Send a chat Messages",
    "description" : "Send messages to the chat (through the bot).",
    "descriptor" : {
      "componentScheme" : "telegram",
      "inputDataShape" : {
        "kind" : "java",
        "type" : "java.lang.String"
      },
      "outputDataShape" : {
        "kind" : "none"
      },
      "propertyDefinitionSteps" : [ {
        "description" : "Chat id",
        "name" : "chatId",
        "properties" : {
          "chatId" : {
            "kind" : "parameter",
            "displayName" : "Chat Id",
            "type" : "string",
            "javaType" : "String",
            "description" : "The telegram's Chat Id, if not set will use CamelTelegramChatId from
the incoming exchange."
          }
        }
      } ],
      "configuredProperties" : {
        "type" : "bots"
      }
    },
    "actionType" : "connector",
    "pattern" : "To"
  } ],
  "properties" : {
    "authorizationToken" : {
      "kind" : "property",
      "displayName" : "Authorization Token",
      "group" : "security",
      "label" : "security",
      "required" : true,
      "type" : "string",
      "javaType" : "java.lang.String",
      "secret" : true,
      "description" : "Telegram Bot Authorization Token"
    }
  }
}
```

2. カスタムコネクターに追加の依存関係が必要な場合は、その依存関係をプロジェクトの **pom.xml** ファイルに追加します。依存関係のデフォルトのスコープは runtime です。Red Hat が出荷する依存関係を追加する場合は、指定されたとおりスコープを定義します。以下はその例です。

```
<dependencies>
  <dependency>
      <groupId>org.apache.camel</groupId>
      <artifactId>camel-telegram</artifactId>
      <scope>provided</scope>
  </dependency>
   </dependencies>
```

カスタムコネクターのコードの作成が終了したら、「Fuse Online エクステンション JAR ファイルのビルド」の説明に従って **.jar** ファイルをビルドします。

9.4. カスタムステップの作成

Fuse Online エクステンションプロジェクトを作成した後、Fuse Online ユーザーによって提供される必要な機能の説明を基にして、カスタムステップを定義するコードを記述します。1 つのエクステンション内では、複数のカスタムステップを定義できます。また、各カスタムステップを Camel ルートまたは Java Bean で定義できます。

9.4.1. カスタムステップのコードの記述

Fuse Online エクステンションプロジェクトを作成した後、Fuse Online ユーザーによって提供される必要な機能の説明を基にして、カスタムステップを定義するコードを記述します。

表9.2「カスタムステップ要素」 は、Fuse Tooling で作成したカスタムステップ要素が Fuse Online 要素にどのように対応するか示しています。

表9.2 カスタムステップ要素

Fuse Tooling 要素	Fuse Online 要素	説明
Action	カスタムステップ	Fuse Online では、ユーザーがステップエクステンションをインポートした後、**Choose a step** ページにカスタムステップが表示されます。
アクションで定義されたプロパティー	カスタムステップ設定パラメーター	Fuse Online では、ユーザーがカスタムステップを選択すると、Fuse Online が設定パラメーターの値を要求します。

Fuse Online のカスタムステップを実装するコードを記述するには、以下を実行します。

1. Camel ルートベースのステップの場合、**extension.xml** ファイルで、エクステンションの目的に対応するルートを作成します。各ルートのエントリーポイントは、ステップ 2 で説明されているように、**syndesis-extension-definition.json** ファイルで定義するエントリーポイントと一致する必要があります。
   Java Bean ベースのステップの場合は、**java** ファイルを編集します。

2. **syndesis-extension-definition.json** ファイルで、アクションとそのプロパティーを定義するコードを作成します。各エントリーポイントに新しいアクションが必要です。

作成する各アクションは Fuse Online のカスタムステップに対応します。アクションごとに、異なるタイプのコードを使用できます。つまり、あるアクションに Camel ルートを、別のアクションに Java Bean を使用することができます。

各プロパティーは Fuse Online のステップ設定パラメーターに対応します。Fuse Online では、ユーザーがカスタムステップを選択すると、Fuse Online が設定パラメーターの値を要求します。たとえば、カスタムログステップにはログに送信する情報量を示す level パラメーターが含まれる場合があります。

エクステンションをアップロードし、そのカスタムステップをインテグレーションに追加した後に Fuse Online でユーザーが入力するプロパティーを含むエクステンションメタデータが含まれる **.json** ファイルのテンプレートは次のとおりです。

```
{
  "actions": [
    {
      "actionType": "extension",
      "id": "${actionId}",
      "name": "Action Name",
      "description": "Action Description",
      "tags": [
          "xml"
      ],
      "descriptor": {
        "kind": "ENDPOINT|BEAN|STEP",
        "entrypoint": "direct:${actionId}",
        "inputDataShape": {
          "kind": "any"
        },
        "outputDataShape": {
          "kind": "any"
        },
        "propertyDefinitionSteps": []
      }
    }
  ],
  "tags": [
    "feature",
    "experimental"
  ]
}
```

注記

このリリースではタグは無視されます。これらは今後使用するために予約されています。

3. エクステンションの依存関係を編集するには、エディターで 'pom.xml' ファイルを開きます。依存関係を追加する場合は、そのスコープを定義する必要があります。

カスタムステップのコード作成が終了したら、「Fuse Online エクステンション JAR ファイルのビルド」の説明に従って **.jar** ファイルをビルドします。

9.5. FUSE ONLINE エクステンション JAR ファイルのビルド

エクステンションの **.jar** ファイルをビルドするには、以下を行います。

1. **Project Explorer** ビューで、プロジェクトを右クリックします。
2. コンテキストメニューから **Run As → Maven clean verify** と選択します。
3. **Console** ビューでは、ビルドの進捗を監視できます。
4. ビルドが完了したら、**Project Explorer** ビューのターゲットフォルダーを更新します (プロジェクトを選択し、**F5** を押します)。
5. **Project Explorer** ビューでターゲットフォルダーを開き、生成された **.jar** ファイルを確認します。
   .jar ファイルの名前は Maven のデフォルト **${artifactId}-${version}.jar** に従います。

   例: **custom:step-camel-1.0.0.jar**

   この **.jar** ファイルは、エクステンション、必要な依存関係、そのメタデータ (Extension Id、Name、Version、Tags、および Description) を定義します。以下に例を示します。

```
{
    "schemaVersion" : "v1",
    "name" : "Example Fuse Online Extension",
    "description" : "Logs a message body with a prefix",
    "extensionId" : "fuse.online.extension.example",
    "version" : "1.0.0",
    "actions" : [ {
        "id" : "Log-body",
        "name" : "Log Body",
        "description" : "A simple xml Body Log with a prefix",
        "descriptor" : {
            "kind" : "ENDPOINT",
            "entrypoint" : "direct:log-xml",
            "resource" : "classpath:META-INF/syndesis/extensions/log-body-action.xml",
            "inputDataShape" : {
                "kind" : "any"
            },
            "outputDataShape" : {
                "kind" : "any"
            },
            "propertyDefinitionSteps" : [ {
                "description" : "Define your Log message",
                "name" : "Log Body",
                "properties" : {
                    "prefix" : {
                        "componentProperty" : false,
                        "deprecated" : false,
                        "description" : "The Log body prefix message",
                        "displayName" : "Log Prefix",
                        "javaType" : "String",
                        "kind" : "parameter",
                        "required" : false,
                        "secret" : false,
                        "type" : "string"
                    }
                }
```

```
            } ]
        },
        "tags" : [ "xml" ],
        "actionType" : "step"
    } ],
    "dependencies" : [ {
        "type" : "MAVEN",
        "id" : "io.syndesis.extension:extension-api:jar:1.3.0.fuse-000014"
    } ],
    "extensionType" : "Steps"
}
```

9.6. JAR ファイルを FUSE ONLINE ユーザーに提供

以下を Fuse Online ユーザーに提供します。

- **.jar** ファイル
- エクステンションを説明するドキュメント。ステップエクステンションでは、ステップエクステンションの各アクションが入力として必要とする、または出力として提供する (データマッピング用) データ形状に関する情報を含めます。

Fuse Online では、ユーザーは Fuse Online でのアプリケーションの統合 の説明どおりに **.jar** ファイルをアップロードします。

第10章 新規 CAMEL XML ファイルの作成

概要

Apache Camel は、camelContext 要素が含まれる XML ファイルにルートを保存します。新しい Fuse Integration プロジェクトを作成する場合、ツールはデフォルトでプロジェクトの一部として Apache Camel コンテキスト (XML) ファイルを提供します。

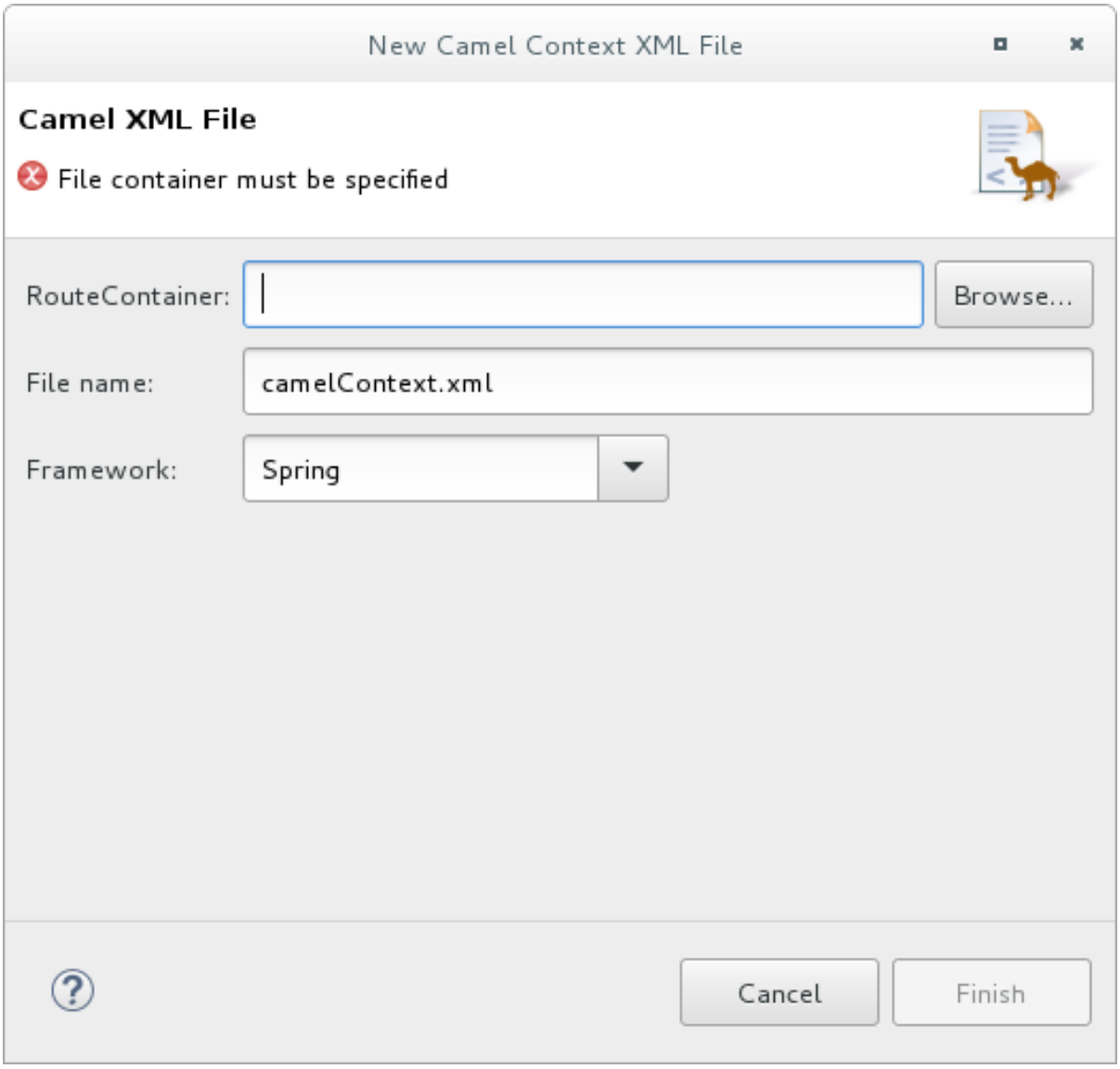
必要な事前設定済み namespace およびテンプレート camelContext 要素が含まれる新しい Camel XML ファイルを追加することもできます。

手順

新しい Apache Camel コンテキストファイルをプロジェクトに追加するには、以下を実行します。

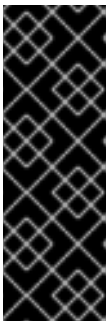
1. 図10.1「Camel XML ファイルウィザード」 に示されるように、メインメニューから **File → New → Camel XML File** を選択し、**Camel XML File** ウィザードを開きます。

図10.1 Camel XML ファイルウィザード



2. **RouteContainer** で、新規ファイルの場所を入力するか、デフォルトを使用します。

をクリックして、適切な場所を検索できます。



重要

Spring フレームワークと OSGi Blueprint フレームワークでは、すべての Apache Camel ファイルをプロジェクトの **META-INF** または **OSGI-INF** フォルダーの特定の場所に配置する必要があります。

- Spring - **projectName/src/main/resources/META-INF/spring/**
- OSGi Blueprint - **projectName/src/main/resources/OSGI-INF/blueprint/**

3. **File Name** に、新しいコンテキストファイルの名前を入力するか、デフォルト (**camelContext.xml**) を受け入れます。
ファイル名にはスペースや特殊文字を含めることはできません。また、JVM 内で一意でなければなりません。

4. **Framework** では、デフォルトを使用するか、ルートが使用するフレームワークを選択します。
   - **Spring** – Spring コンテナーや OSGi 以外のコンテナーでデプロイされるか、スタンドアロンアプリケーションとしてデプロイされるルート用の [default]
   - **OSGi Blueprint** – OSGi コンテナーにデプロイされるルート用
   - **Routes**: 読み込み既存の **camelContext** に追加できるルート用

5. **Finish** をクリックします。
新しいコンテキストファイルがプロジェクトに追加され、ルートエディターで開かれます。

第11章 CAMEL バージョンの変更

Fuse ツールプロジェクトで作業する場合、使用する Camel のバージョンを変更することをお勧めします。そうすると、たとえば、より新しいバージョンの Camel でサポートされる機能を使用する場合や、コミュニティーバージョンを使用する場合に役に立ちます。

プロジェクトが使用する Camel バージョンを変更するには、以下を実行します。

1. **Project Explorer** で、Camel バージョンを変更するプロジェクトを右クリックし、**Configure → Change Camel Version** を選択します。

2. **Change Camel Version** ウィンドウで、**Camel Version** フィールドの右側にある下矢印をクリックし、利用可能な Camel バージョンを表示します。
コミュニティーバージョンの Apache Camel を使用するには、そのバージョン番号 (例: **2.19.2**) を入力します。

3. 必要なバージョンを選択または入力し、**Finish** をクリックします。

Fuse Tooling は、選択したバージョンが利用可能で、Fuse Tooling によってサポートされるかどうかを確認します。確認されると、Fuse Tooling は Camel バージョンを変更し、プロジェクトの更新された **pom.xml** ファイルを保存します。選択した Camel バージョンが利用できない、またはサポートされていない場合、エラーメッセージが表示されます。

pom.xml ファイルの **<camel.version>** 要素で、プロジェクトの Camel バージョンを確認できます。

第12章 既存 MAVEN プロジェクトのインポート

概要

たとえば、アプリケーション開発のテンプレートや開始点として使用するために、既存プロジェクトのインポートが必要な場合があります。

たとえば、New Fuse Integration Project ウィザードは、以下の Github リポジトリーを参照します。

- https://github.com/apache/camel/tree/master/examples
- https://github.com/fabric8-quickstarts
- https://github.com/wildfly-extras/wildfly-camel-examples
- https://github.com/jboss-fuse/quickstarts

手順

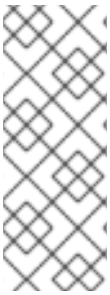
既存の Maven プロジェクトをインポートするには、以下を実行します。

1. **File → Import → Maven → Existing Maven Projects**の順に選択し、**Next** をクリックします。
2. ルートディレクトリーには、ダウンロードしたサンプルプロジェクトが含まれているフォルダーを選択します。
3. プロジェクト一覧で、インポートするプロジェクトを確認してから **Finish** をクリックします。

パート II. ルーティングコンテキストのデバッグ

Camel デバッガーには、ローカルおよびリモートで実行されているルーティングコンテキストのデバッグに使用する機能が多数含まれています。

- ルートエディターでノードに条件付きおよび無条件のブレークポイントを設定
- デバッガーの自動起動および **Debug** パースペクティブへの切り替え
- 実行中のルーティングコンテキストとの対話
  - ブレークポイント間の切り替えでメッセージインスタンスの変数の値を迅速に比較
  - 対象の変数の値を確認して変更
  - デバッグセッション全体で追跡するために監視一覧に対象の変数を追加
  - ブレークポイントを無効にして再度有効化
  - ルーティングコンテキストランタイムでのメッセージフローをグラフィカルに追跡
  - コンソールログを確認して、Camel およびデバッガーアクションを追跡



注記

Camel デバッガーを実行する前に、ルートエディターのキャンバスに表示される対象ノードにブレークポイントを設定する必要があります。その後、プロジェクトのルーティングコンテキストの **.xml** ファイルで Camel デバッガーを実行し、そのコンテキスト内のロジックエラーを見つけ、修正できます。Camel デバッガーを呼び出すと、デバッグモードでルーティングコンテキストが実行され、Debug Perspective が開きます。

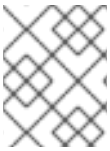
第13章 ブレークポイントの設定

概要

ブレークポイントを設定するには、ルートエディターの **Design** タブでプロジェクトのルーティングコンテキストの **.xml** ファイルを開く必要があります。

Camel デバッガーは、2 種類のブレークポイントをサポートします。

- 無条件ブレークポイント – デバッグセッション中にブレークポイントが検出されるたびにトリガーされます。
- 条件付きブレークポイント – デバッグセッション中にブレークポイントの指定条件が満たされた場合にのみトリガーされます。



注記

コンシューマーエンドポイントまたは **when** もしくは **otherwise** ノードにブレークポイントを設定することはできません。

無条件ブレークポイントの設定

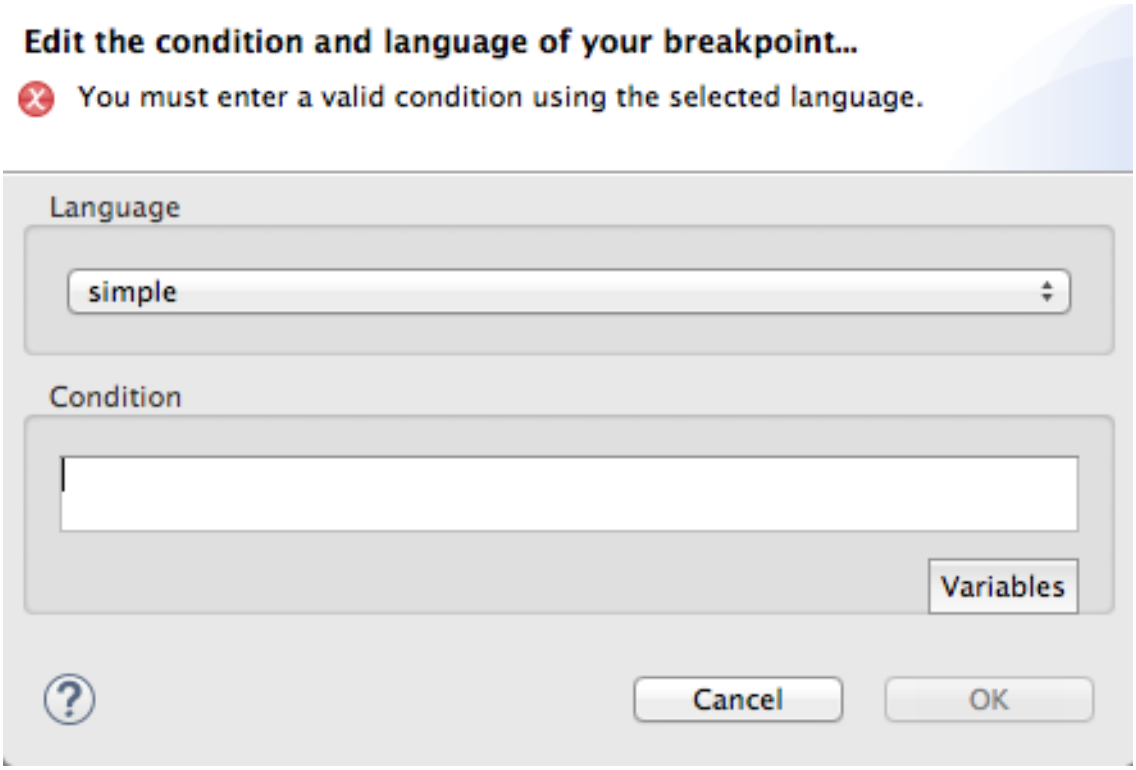
ルーティングコンテキストが **Design** タブのキャンバスに表示されます。

1. デバッグセッション中に状態を検査するノードを選択します。
2.  アイコンをクリックして、無条件ブレークポイントを設定します。
3. 無条件ブレークポイントを設定する各ノードで、この手順を繰り返します。

条件付きブレークポイントの設定

ルーティングコンテキストが **Design** タブのキャンバスに表示されます。

1. デバッグセッション中に状態を検査するノードを選択します。
2.  アイコンをクリックして、条件付きブレークポイントを設定し、**Edit the condition and language of your breakpoint...** ダイアログを開きます。



3. **Language** ドロップダウンメニューをクリックし、ブレークポイントをトリガーする条件を作成するために使用する式の言語を選択します。
Fuse Tooling は、24 の式言語をサポートしています。これらの言語には、条件式を作成するための変数が含まれている場合も、含まれていない場合もあります。

4. **Variables** をクリックし、選択した言語でサポートされる変数の一覧を表示します。
リストが表示された場合は、1 つ以上の変数を順番に選択して、ブレークポイントをトリガーするための条件を作成します。**Condition** テキストボックスに選択した変数が表示されます。

<nothing available> が表示された場合には、**Condition** テキストボックスに式を直接入力します。

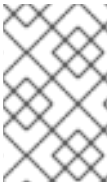
5. 条件付きブレークポイントを設定する各ノードで、[condBpFirst] から[condBpLast] の手順を繰り返します。

ブレークポイントの無効化

ブレークポイントを一時的に無効にしてそのままにし、後で再度有効にできます。 ボタンは、デバッグセッション中に無効にしたブレークポイントをスキップします。

ブレークポイントを無効にするには、キャンバスのノードを選択し、 アイコンをクリックします。ブレークポイントは灰色になり、無効化されたことを示します。

無効にしたブレークポイントを有効にするには、キャンバスのノードを選択し、 アイコンをクリックします。無効にしたブレークポイントが条件付きか無条件かに応じて黄色または赤色に変化し、再度有効化されたことを示します。



注記

セッションのデバッグ中にブレークポイントを無効にしてから再度有効にすることもできます。詳細は、18章*実行中のコンテキストでのブレークポイントの無効化* を参照してください。

ブレークポイントの削除

個々のブレークポイントまたはすべてのブレークポイントを削除できます。

- 個別のブレークポイントを削除するには、ルートコンテナーでブレークポイントを削除するノードを選択し、その  アイコンをクリックします。
- 特定のルートにあるすべてのブレークポイントを削除するには、対象のルートのコンテナーを右クリックし、 
  Delete all breakpoints を選択します。
- すべてのルートですべてのブレークポイントを削除するには、キャンバスを右クリックして 
  Delete all breakpoints を選択します。

関連トピック

- 14章*Camel デバッガーの実行*

第14章 CAMEL デバッガーの実行

Camel デバッガーを、ローカルで実行中のルーティングコンテキストで実行できます。

注記

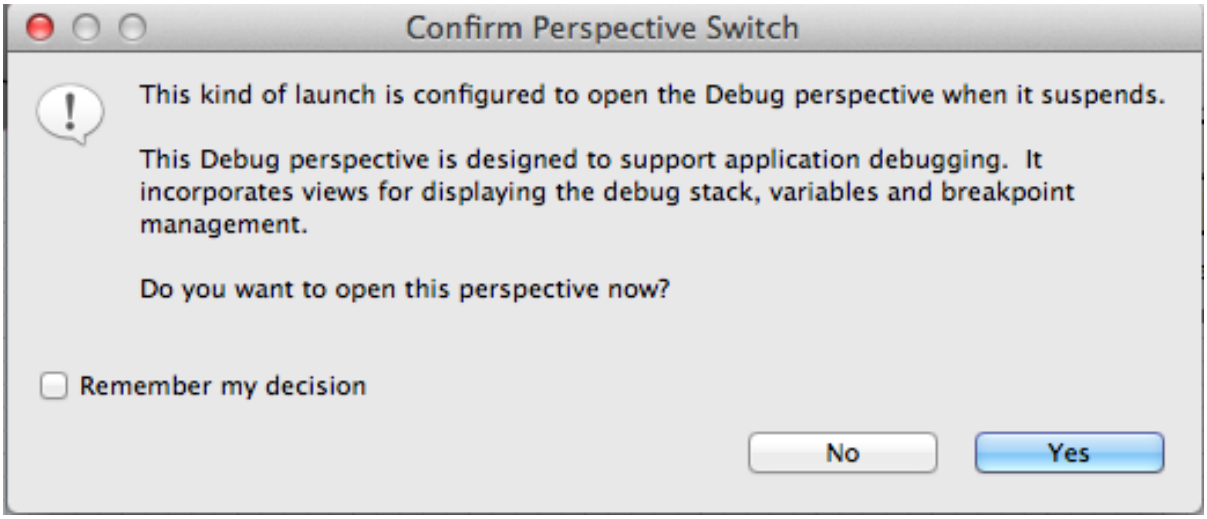
- リモートデバッグはサポートされなくなりました。リモートデバッグの場合は、Jolokia を設定し、Jolokia 経由で特定の JMX 接続を作成する必要があります。
- プロジェクトに Java コードが含まれている場合は、標準の Eclipse Java デバッグツールを使用してデバッグできます。

Camel デバッガーを起動する前に、ルーティングコンテキストファイルでブレークポイントを設定する必要があります。

手順

1. **Project Explorer** ビューで、デバッグするルーティングコンテキストファイルを選択します。
2. 選択したファイルを右クリックしてコンテキストメニューを開き、**Debug As → Local Camel Context** の順に選択します。
   Fuse Tooling は Camel ルートをビルドし、Apache Camel を起動し、ルーティングコンテキストを起動し、JMX を有効にし、ルーティングコンテキストでルートを起動し、ブレークポイントをノードに追加し、Camel デバッガーを有効にします。

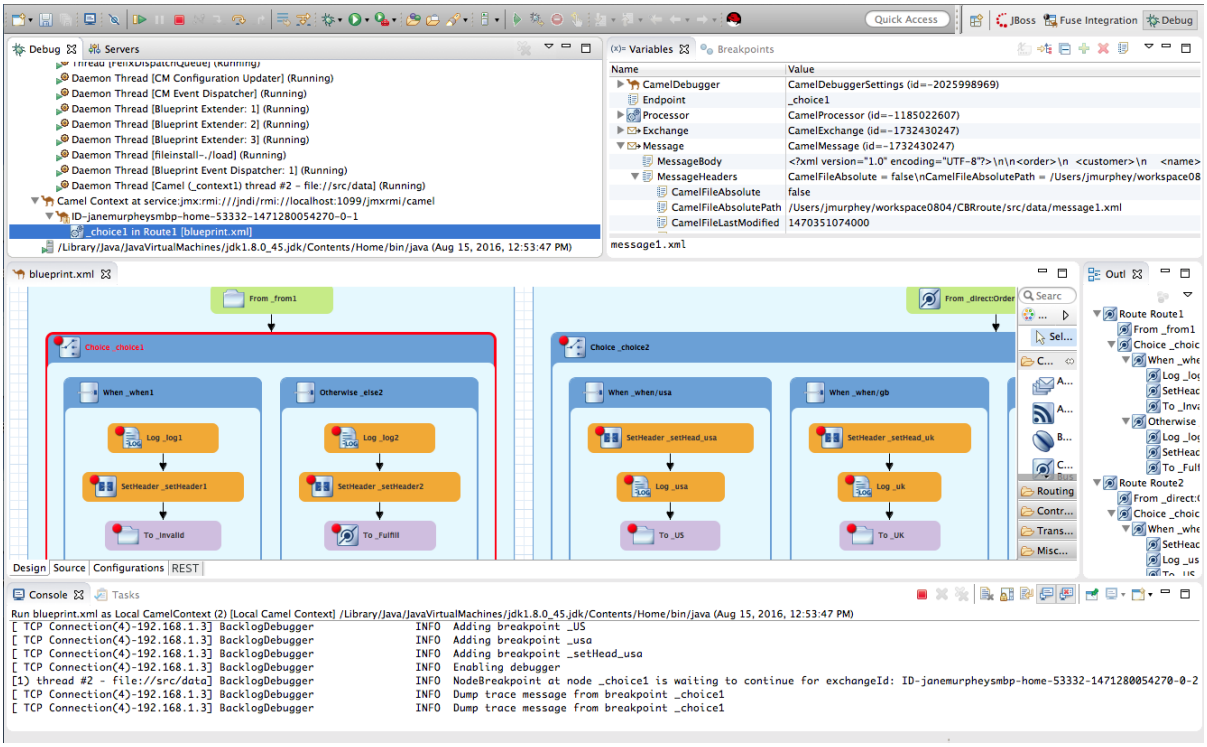
   Camel デバッガーは、最初のブレークポイントヒット (メッセージの受信) でルーティングコンテキストの実行を一時停止し、**Debug** パースペクティブを開くかどうかを尋ねます。

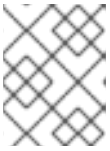


3. **Yes** をクリックして **Debug** パースペクティブを開きます。
   Debug パースペクティブが開き、実行中のルーティングコンテキストで最初に発生した最初のブレークポイントでルーティングコンテキストが一時停止されます。

重要

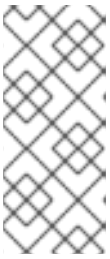
ブレークポイントは、デバッガーが自動的に再開した後に最大 5 分間保持され、次のブレークポイントまたはルーティングコンテキストの最後に移動します。





注記

コンソールの出力を表示する場合、パースペクティブをオンにしたときに開いていなければ **Console** ビューを開きます。

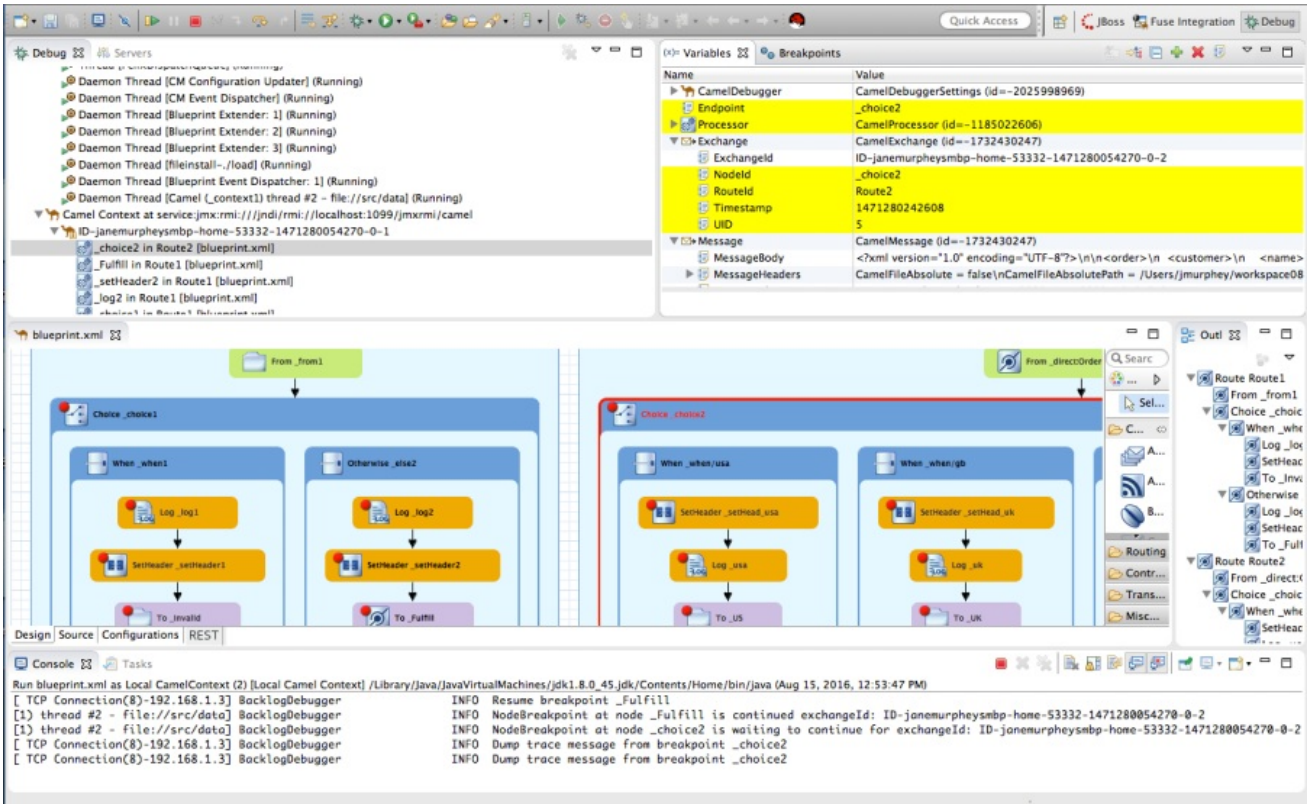


注記

デフォルトでは、**Debug** パースペクティブには **Outline** ビューが表示されます。このビューは、実行中のルーティングコンテキストで個別のルートを切り替える手段を提供します。ルーティングコンテキストに単一のルートが含まれている場合は、**Outline** ビューを閉じると、他のビューを展開するためのスペースが解放され、デバッガー出力へのアクセスと調査が容易になります。

ルーティングコンテキストを介したメッセージエクスチェンジの進捗監視

(ステップオーバー) をクリックし、ルーティングコンテキストで実行される次のノードにジャンプします。 (再開) をクリックし、ルーティングコンテキストの次のアクティブなブレークポイントで実行を継続します。



関連トピック

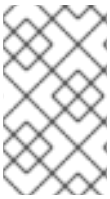
- *15章Camel デバッガーの停止*

第15章 CAMEL デバッガーの停止

概要

Camel デバッガーを停止するには、デバッグセッションが終了したら、メニューバーの  を 1 回クリックします。それ例外の場合は、  を 2 回クリックします。1 回目で現在実行しているノードスレッドを終了し、2 回目で **Camel Context** スレッドを終了します (どちらも **Debug** ビューに表示されます)。

+

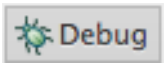


注記

Camel デバッガーを終了するとコンソールも終了しますが、出力はクリアされません。出力をクリアするには、**コンソール** ビューのメニューバーの  (コンソールのクリア) をクリックします。

CAMEL デバッガーを閉じる

プロジェクトのデバッグが終了したら、**Debug** パースペクティブを閉じて、ワークベンチのスペースを増やすことができます。

これを行うには、ツールバーの 

を右クリックし、**Close** を選択します。

関連トピック

- *14章Camel デバッガーの実行*

第16章 変数値の変更

概要

Camel デバッガーがブレークポイントに到達すると、Variables ビューにはルーティングコンテキストのその時点で利用可能なすべての変数の値が表示されます。一部の変数は編集可能で、値を変更できます。これにより、アプリケーションがプログラム状態の変更をどのように処理するか確認できます。

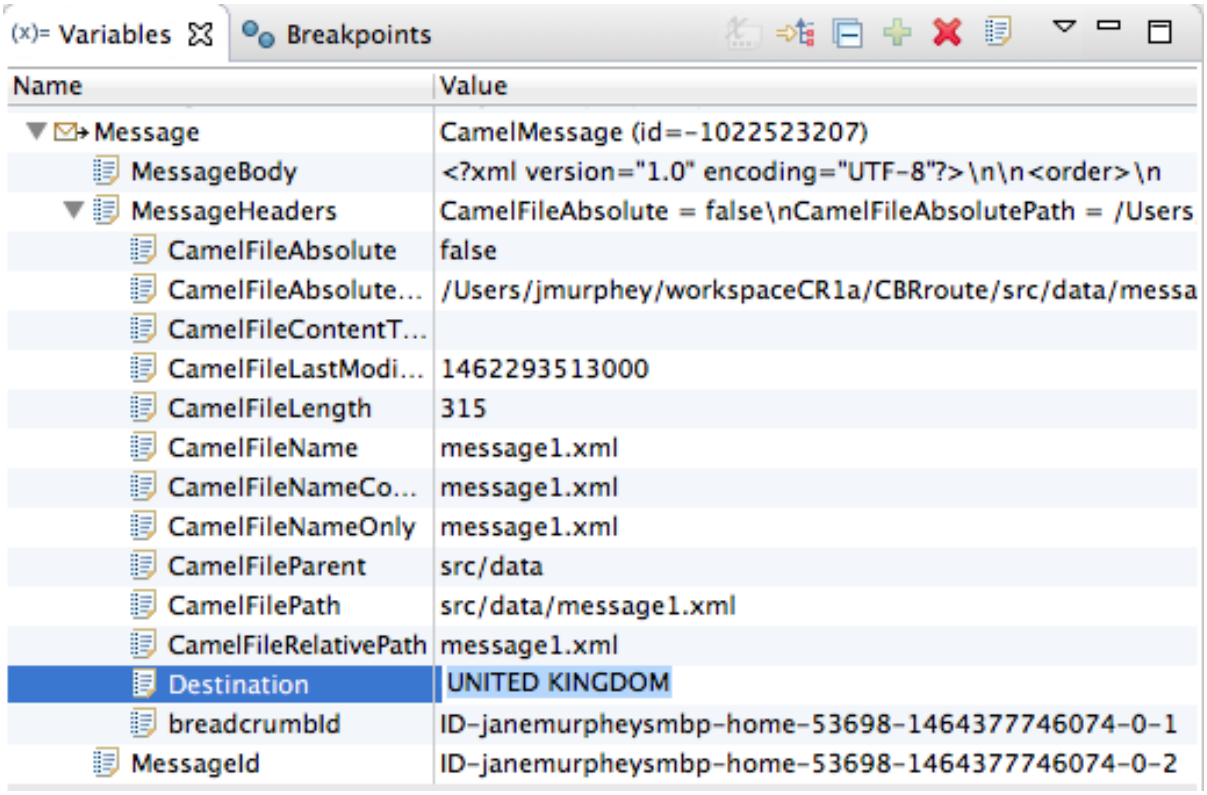
注記

すべての変数が編集可能というわけではありません。編集可能な変数のコンテキストメニューには、**Change Value...** オプションが表示されます。

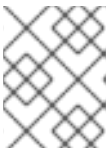
手順

変数の値を変更するには、以下を実行します。

1. 必要に応じてデバッガーを起動します。14章*Camel デバッガーの実行*を参照してください。
2. **Variables** ビューで、変更する値の変数を選択し、その **Value** フィールドをクリックします。



変数の **value** フィールドが薄い青の網掛け表示になり、編集モードであることを示します。



注記

または、変数を右クリックしてコンテキストメニューを開き、**Change Value...** を選択して値を編集します。

3. 新しい値を入力してから、**Enter** をクリックします。
Console ビューには、変数の値の変更に関する **INFO** レベルのログエントリーが表示されます (たとえば、**Breakpoint at node to1 is updating message header on exchangeId: ID-dhcp-**

97-16-bos-redhat-com-52574-1417298894070-0-2 with header: Destination and value: UNITED KINGDOM)。

4. ブレークポイントをステップピングし、メッセージが想定どおりに処理されているかどうかを確認します。各ステップで **Debug** ビュー、**Variables** ビュー、および **Console** の出力を確認します。

関連トピック

- 18章*実行中のコンテキストでのブレークポイントの無効化*
- 17章 *ウォッチリストへの変数の追加*

第17章 ウォッチリストへの変数の追加

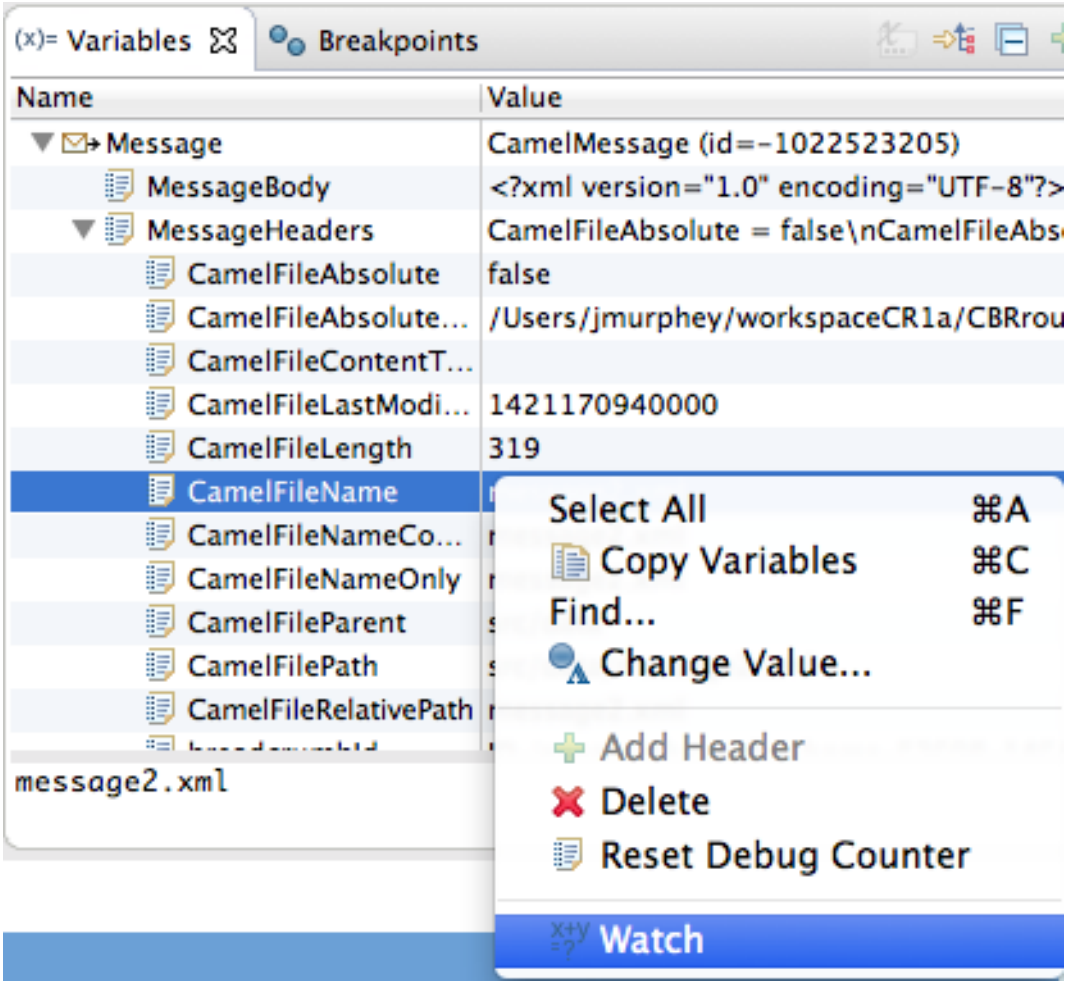
概要

変数をウォッチリストに追加すると、特定の変数にフォーカスして、ルーティングコンテキストを通過する際に、値が想定どおりに変化するかどうかを確認できます。

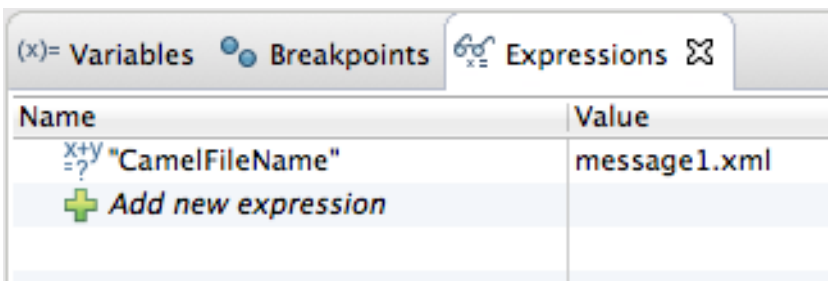
手順

ウォッチリストに変数を追加するには、以下を実行します。

1. 必要に応じてデバッガーを起動します。14章*Camel デバッガーの実行*を参照してください。
2. **Variables** ビューで、追跡する変数を右クリックして、コンテキストメニューを開きます。



3. **Watch** を選択します。
新しいビューである **Expressions** が、**Breakpoints** ビューの横に開きます。**Expressions** ビューには、監視される変数の名前と現在の値が表示されます。以下はその例です。



4. [watch1] および [watch2] を繰り返し、ウォッチリストに変数を追加します。

注記

追加する変数は、削除されるまでウォッチリストに残ります。変数の監視を停止するには、リストで変数を右クリックしてコンテキストメニューを開き、**Remove** をクリックします。

5. **Expressions** ビューが開いたら、ルーティングコンテキストでステップを実行し、ウォッチリストの各変数がルートの各ステップに到達した際にどのように変化するか追跡します。

関連トピック

- *16章 変数値の変更*

第18章 実行中のコンテキストでのブレークポイントの無効化

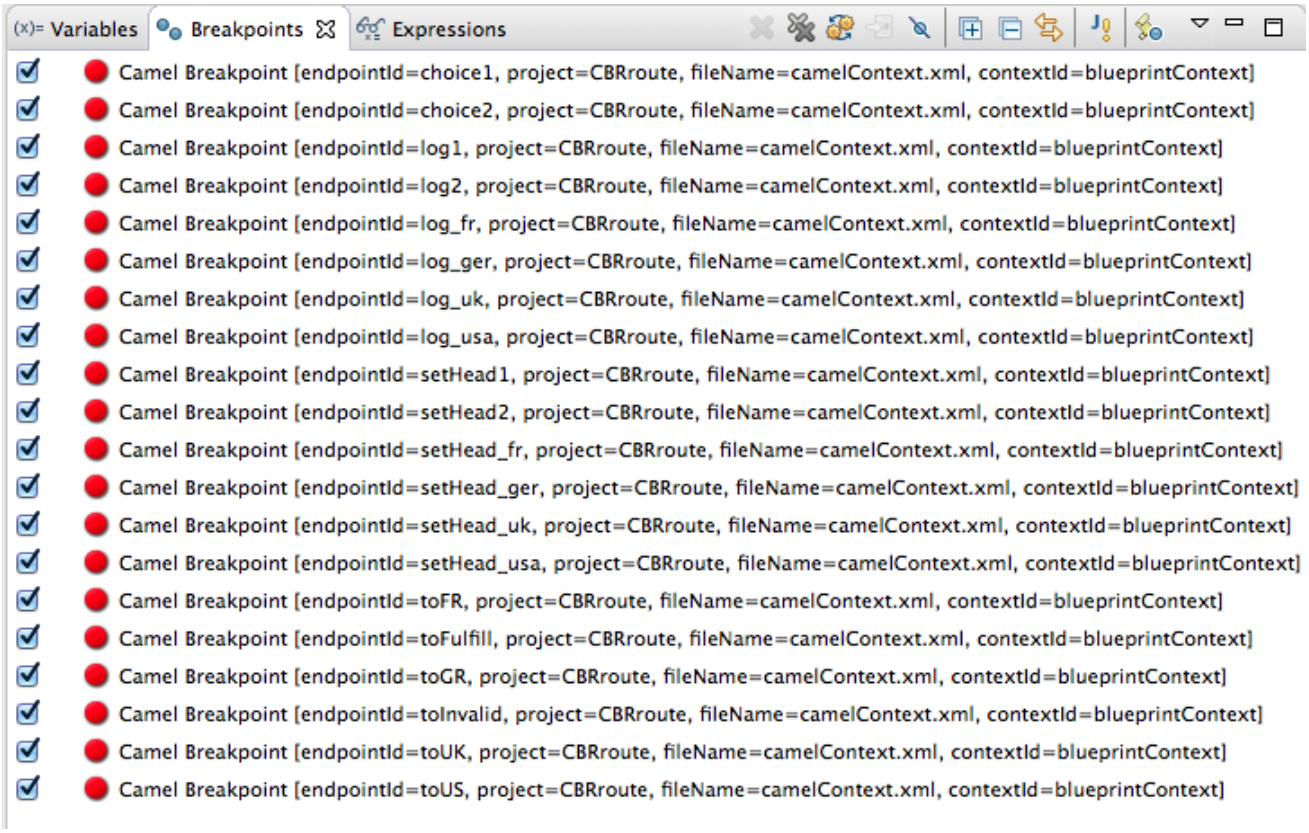
概要

Breakpoints ビューで、実行中のルーティングコンテキストのブレークポイントを無効にして再度有効にできます。

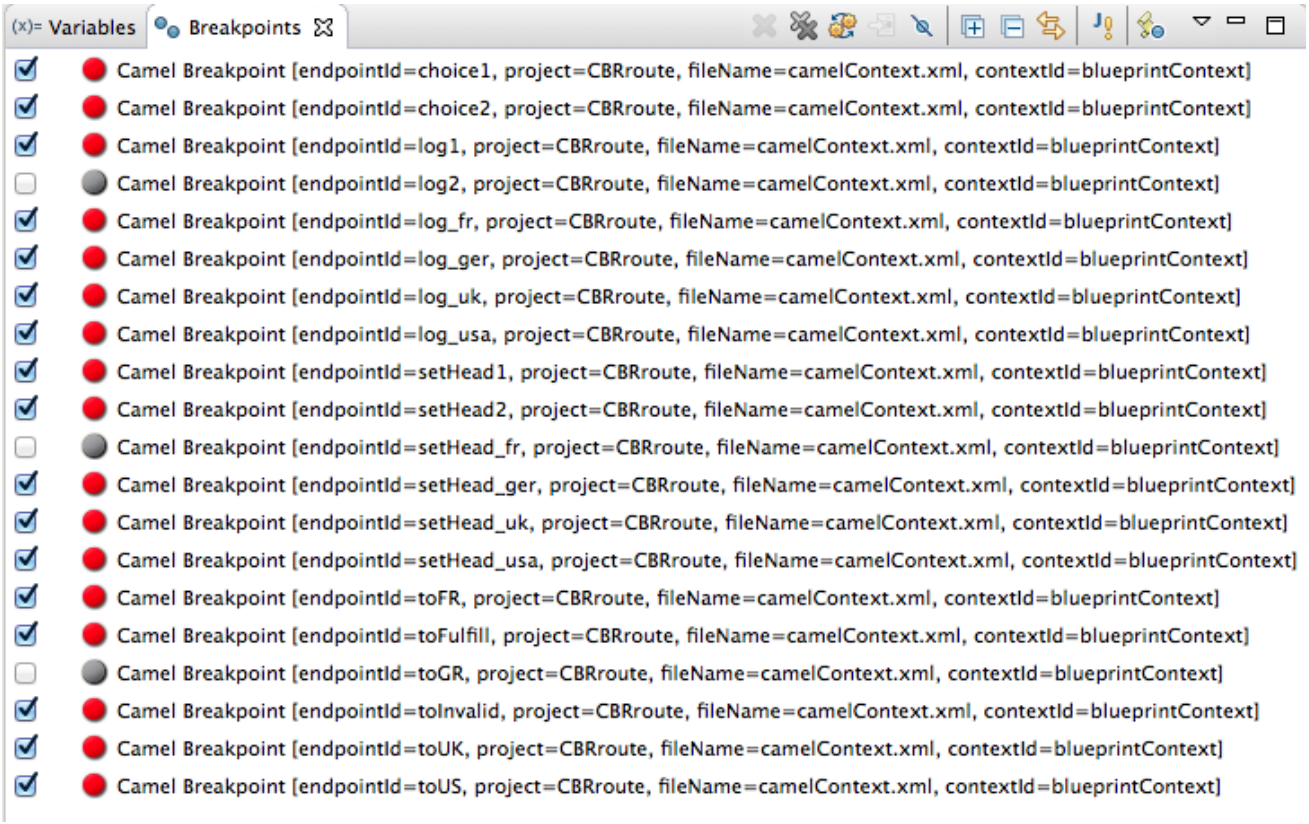
ブレークポイントが無効になっている場合、 ボタンにより、デバッガーはデバッグセッション時にそのブレークポイントをスキップします。

ブレークポイントビューでのブレークポイントの無効化と有効化

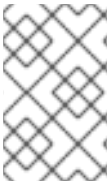
ブレークポイントがすべて有効な状態で、**Breakpoints** ビューが開きます。



ブレークポイントを無効にするには、そのチェックボックスの選択を解除します。



無効にする各ブレークポイントについて、Console ビューに **INFO** レベルのログエントリーが表示され、無効になったことが示されます (例: **Removing breakpoint log2**)。同様に、再度有効にする各ブレークポイントについて、Console ビューに **INFO** レベルのログエントリーが表示され、有効になったことが示されます (例: **Adding breakpoint log2**)。

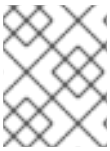


注記

無効にしたブレークポイントを再度有効にするには、チェックボックスをクリックします。Console ビューに **INFO** レベルのログエントリーが表示され、ブレークポイントが選択したノードに追加されたことが示されます。

パート III. アプリケーションの監視とテスト

JMX Navigator ビューは、Fuse アプリケーションを監視およびテストする方法を多数提供します。



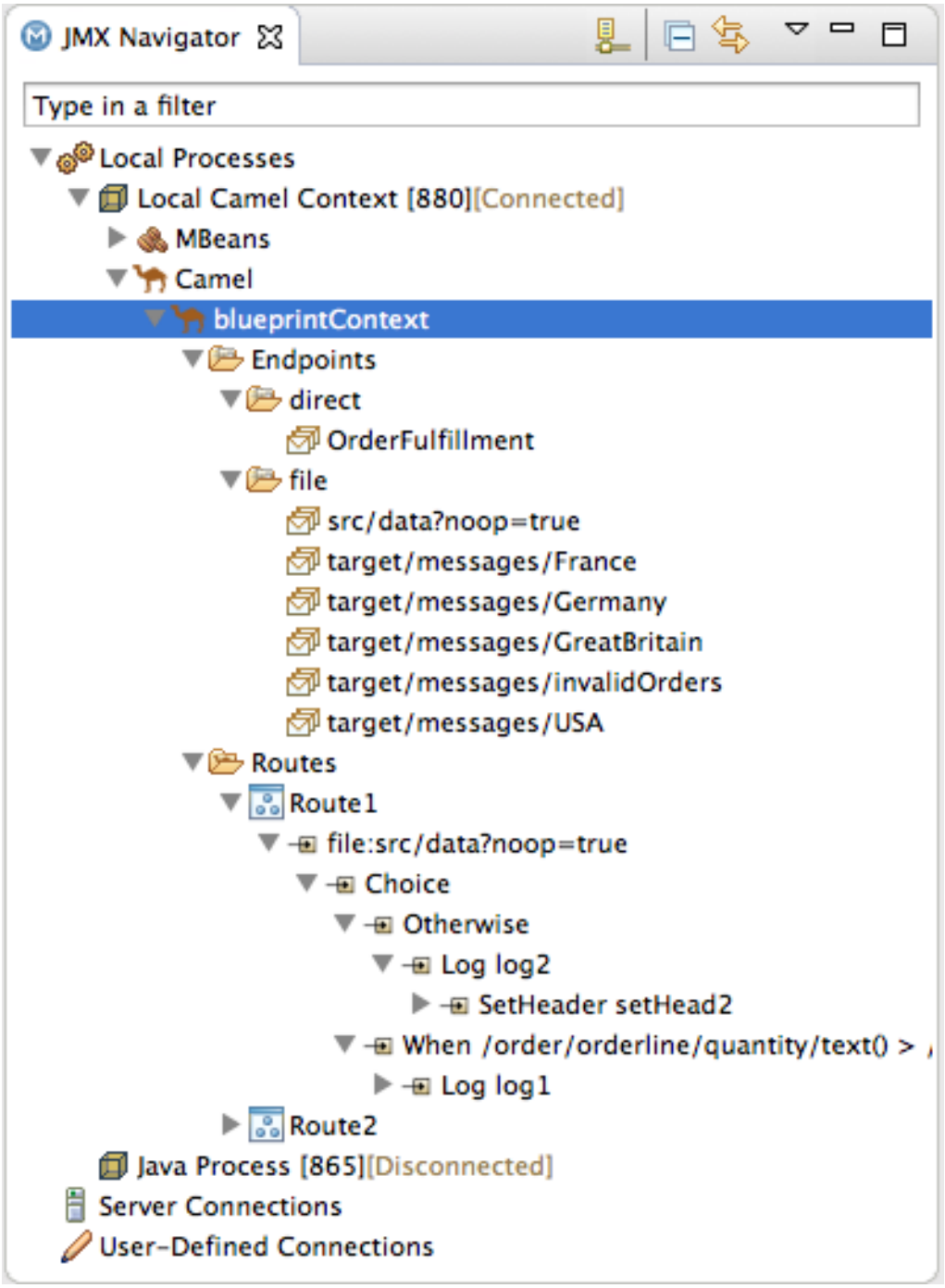
注記

Fuse の管理 に記載のとおり、Fuse Console で Fuse アプリケーションを監視することもできます。

第19章 JMX ナビゲーター

図19.1「JMX ナビゲータービュー」 に示される **JMX Navigator** ビューには、アプリケーションで実行されているすべてのプロセスが表示され、監視機能とテスト機能のインタラクションが促進されます。**Fuse Integration** パースペクティブの他の領域は、**JMX Navigator** ビューで選択したノードに関連する情報を表示します。**JMX Navigator** ビューでは、そのコンテキストメニューで、ルートトレースを有効にして JMS 宛先を追加するために必要なコマンドが提供されます。

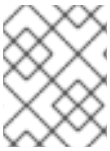
図19.1 JMX ナビゲータービュー



デフォルトでは、**JMX Navigator** ビューはローカルマシンで実行されているすべての JMX サーバーを検出し、以下のカテゴリーに一覧表示します。

- ローカルプロセス
- サーバー接続

- ユーザー定義の接続



注記

サーバーの JMX URL を使用すると、他の JMX サーバーを追加できます。詳細は、「JMX サーバーの追加」 を参照してください。

19.1. JMX でのプロセスの表示

概要

JMX Navigator ビューには、一連のツリー内の既知のプロセスがすべて一覧表示されます。各ツリーのルートは JMX サーバーです。

一覧の最初のツリーは、ローカルマシンで実行しているすべての JMX サーバーが含まれる特別な **Local Processes** ツリーです。JMX サーバーの 1 つに接続し、含まれるプロセスを確認する必要があります。

ローカル JMX サーバーでのプロセスの表示

ローカル JMX サーバーのプロセスに関する情報を表示するには、以下を行います。

1. **JMX Navigator** ビューで **Local Processes** を展開します。
2. **Local Processes** で、最上位エントリーのいずれかをダブルクリックして接続します。
3. エントリーの横に表示される ▶ アイコンをクリックし、JVM で実行されているコンポーネントの一覧を表示します。

代替 JMX サーバーでのプロセスの表示

代替 JMX サーバーのプロセスに関する情報を表示するには、以下を実行します。

1. **JMX Navigator** ビューに JMX サーバーを 「JMX サーバーの追加」 します。
2. **JMX Navigator** ビューで、エントリーの横にある ▶ アイコンを使用してサーバーのエントリーを展開します。JVM で実行されている JMX サーバーのコンポーネント一覧が表示されます。

19.2. JMX サーバーの追加

概要

JMX Navigator ビューで、ツリーの **Local Processes** ブランチのにローカル JMX サーバーの一覧が表示されます。他のマシンにデプロイされたコンポーネントを表示する場合は、特定の JMX サーバーに接続する必要がある場合があります。

JMX サーバーを追加するには、追加するサーバーの JMX URL を知っている必要があります。

手順

JMX サーバーを **JMX Navigator** ビューに追加するには、以下を行います。

1. **JMX Navigator** ビューで、**New Connection**  をクリックします。
2. **Create a new JMX connection**ウィザードで **Default JMX Connection** を選択します。
3. **Next** をクリックします。
4. **Advanced** タブを選択します。
5. **Name** フィールドに、JMX サーバーの名前を入力します。
   名前には任意の文字列を指定できます。これは、**JMX Navigator** ツリーのエントリーにラベルを付けるために使用されます。
6. **JMX URL** フィールドに、サーバーの JMX URL を入力します。
7. JMX サーバーに認証が必要な場合は、**Username** フィールドおよび **Password** フィールドにユーザー名とパスワードを入力します。
8. **Finish** をクリックします。
   新しい JMX サーバーが、**User-Defined Connections** ツリーにブランチとして表示されます。

第20章 コンポーネントの JMX 統計情報の表示

概要

このツールは、Fuse コンポーネントによって報告されるすべての JMX 統計情報を収集し、**Properties** ビューに表示します。この統計情報により、インテグレーションアプリケーションで何が起こっているかについて重要な洞察を得ることができます。

JMX 統計情報は、**Properties**、**Processor**、**Profile** の 3 つのカテゴリーに分類されます。

手順

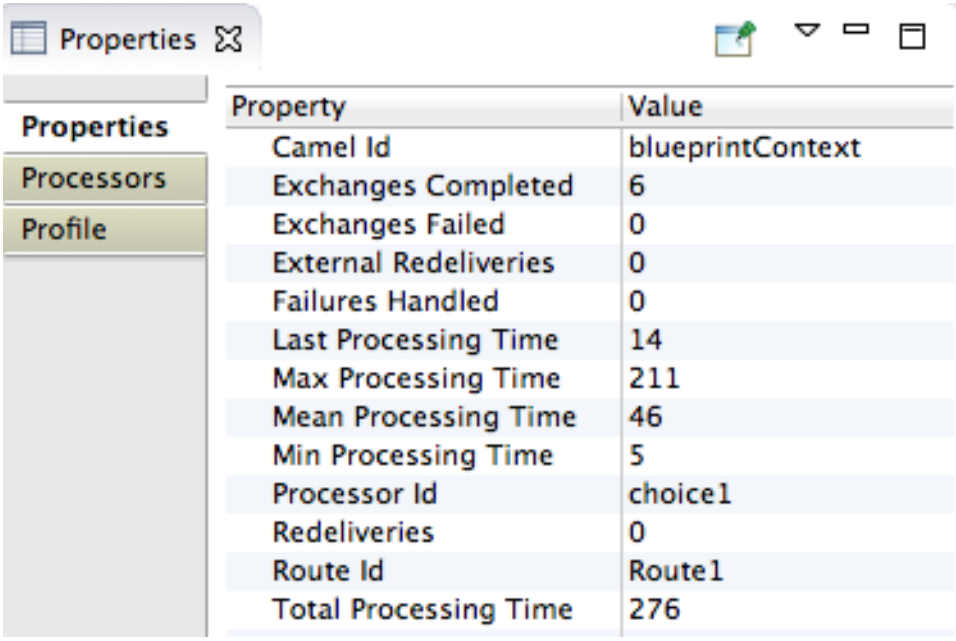
Fuse コンポーネントの統計情報を表示するには、以下を行います。

1. **JMX Navigator** ビューで、コンポーネントのノードを見つけます。
   下位コンポーネントを見つけるには、ツリー上でノードを展開する必要がある場合があります。

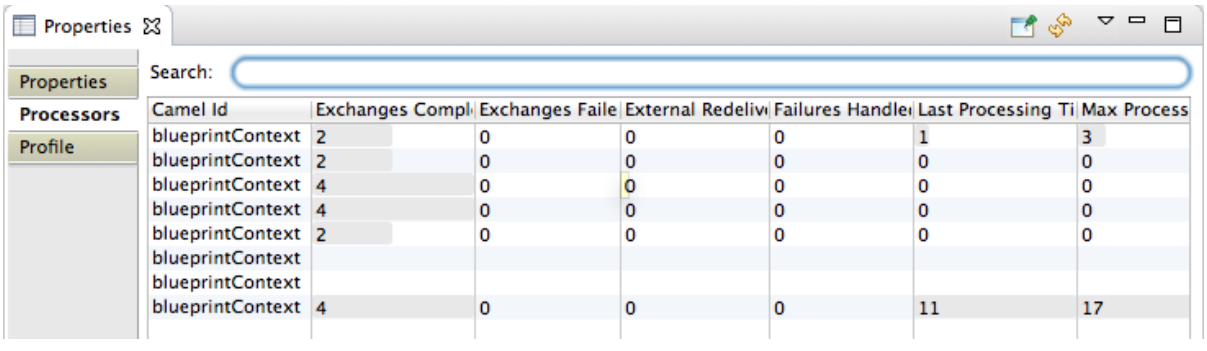
2. 統計情報を確認する Fuse コンポーネントのノードを選択します。

3. **Properties** ビューを開きます。

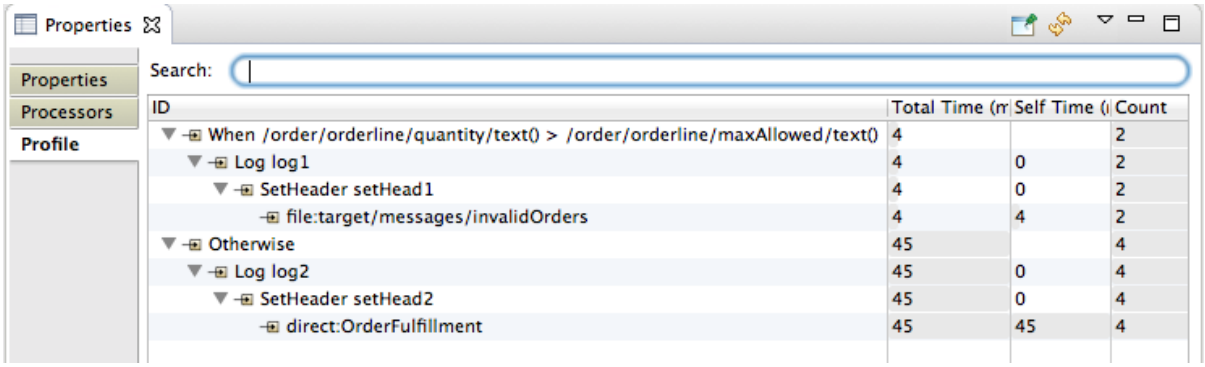
4. **Properties** ページには、選択したコンポーネントの JMX プロパティーが表示されます。



5. **Processors** をクリックし、選択したコンポーネントのエクスチェンジメトリクスを確認します。



6. **Profile** をクリックして、選択したノードとそのサブノードのメッセージメトリクスを確認します。



第21章 メッセージの参照

概要

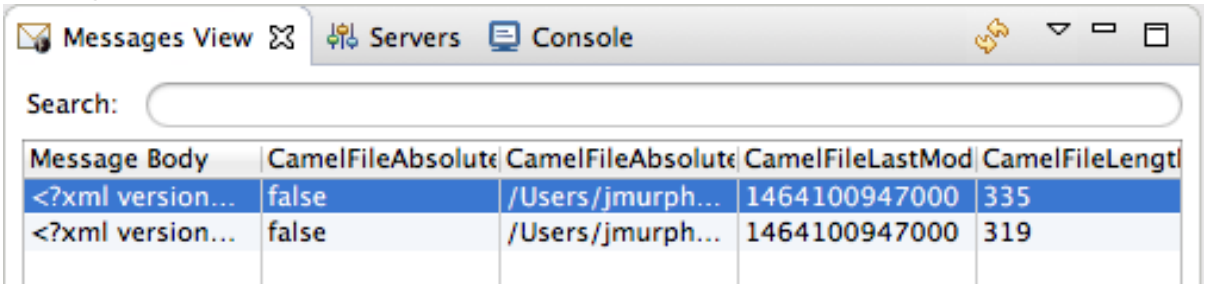
分散環境でアプリケーションをデバッグする際の重要なツールは、アプリケーションの JMS 宛先およびルートエンドポイントに格納されているすべてのメッセージを確認します。ツールは以下を参照できます。

- JMS 宛先
- JMS ルーティングエンドポイント
- Apache Camel ルーティングエンドポイント
- SEDA ルーティングエンドポイント
- ルーティングエンドポイントの参照
- Mock ルーティングエンドポイント
- 仮想マシンルーティングエンドポイント
- DataSet ルーティングエンドポイント

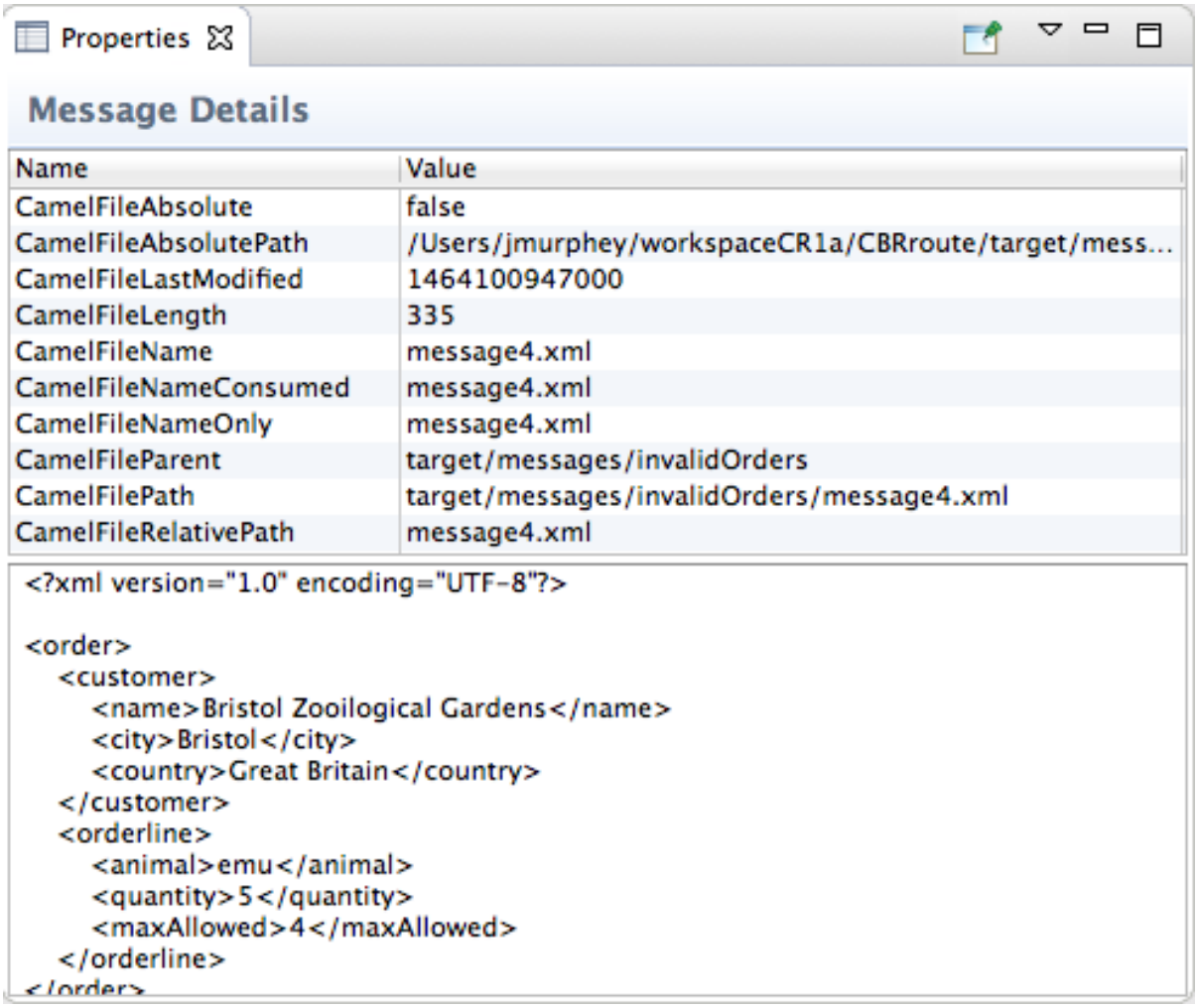
手順

メッセージを参照するには、以下を行います。

1. **JMX Navigator** ビューで、参照する JMS 宛先またはエンドポイントを選択します。
   メッセージの一覧は **Messages View** に表示されます。
2. **Messages View** で、検査する個別のメッセージを選択します。



   メッセージの詳細とコンテンツが **Properties** ビューに表示されます。



関連トピック

- 「ルーティングコンテキストを介したメッセージのトレース」

第22章 ルートのトレース

ルートをデバッグするには、以下の 2 つの問題のいずれかを解決する必要がある場合があります。

- メッセージが適切に変換されない。
- メッセージが宛先エンドポイントに到達しない。

ルートを介して 1 つ以上のテストメッセージをトレースすることは、このような問題の原因を見つける最も簡単な方法です。

ツールのルートトレース機能を使用すると、メッセージがルートを通過するパスを監視し、メッセージがプロセッサーからプロセッサーに渡されるときにどのように変換されるかを確認できます。

Diagram View はルートをグラフィカル表示します。これにより、メッセージが通過するパスを表示できます。ルート内の各プロセッサーでは、ルートの起動後に処理されたすべてのメッセージの平均処理時間 (ミリ秒単位) と、ルートの起動後に処理されたメッセージ数も表示されます。

Messages View には、**JMX Navigator** ツリーで選択した JMS 宛先またはルートエンドポイントによって処理されたメッセージが表示されます。**Messages View** で個別のメッセージトレースを選択すると、**Properties** ビューにメッセージの詳細な情報とコンテンツが表示され、**Diagram View** で関連ノードが強調表示されます。

ルートを使用したメッセージのトレースには、以下のステップが含まれます。

1. 「ルートトレース用テストメッセージの作成」
2. 「ルートトレースの有効化」
3. 「ルーティングコンテキストを介したメッセージのトレース」
4. 「ルートトレースの無効化」

22.1. ルートトレース用テストメッセージの作成

概要

ルートトレースは、あらゆる種類のメッセージ構造で機能します。**Fuse Message** ウィザードは空の **.xml** メッセージを作成し、任意のメッセージ構造を設定することができます。

注記

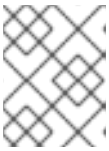
テストメッセージを保存するフォルダーが存在しない場合は、メッセージを作成する前に作成する必要があります。

テストメッセージを格納する新規フォルダーの作成

新規フォルダーを作成するには、以下を実行します。

1. **Project Explorer** ビューで、プロジェクトルートを右クリックしてコンテキストメニューを開きます。
2. **New → Folder** を選択して **New Folder** ウィザードを開きます。
   プロジェクトルートが **Enter or select the parent folder**に表示されます。

3. プロジェクト階層のグラフィカル表示でノードを展開し、親フォルダーとして使用するノードを選択します。
4. **Folder name** フィールドに新規フォルダーの名前を入力します。
5. **Finish** をクリックします。
新規フォルダーが **Project Explorer** ビューの選択した親フォルダーの下に表示されます。



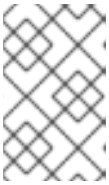
注記

新規フォルダーが表示されない場合は、親 foler を右クリックし、**Refresh** を選択します。

テストメッセージの作成

テストメッセージを作成するには、以下を実行します。

1. **Project Explorer** ビューで、プロジェクトを右クリックしてコンテキストメニューを開きます。
2. **New → Fuse Message** を選択して **New File** ウィザードを開きます。
3. プロジェクト階層のグラフィカル表示でノードを展開し、新しいテストメッセージを保存するフォルダーを選択します。
4. **File name** フィールドにメッセージの名前を入力するか、デフォルト (**message.xml**) を受け入れます。
5. **Finish** をクリックします。
XML エディターで新しいメッセージが表示されます。
6. メッセージの内容 (ボディーとヘッダーテキストの両方) を入力します。



注記

入力したヘッダーテキストによっては、**No grammar constraints (DTD or XML Schema) referenced in the document** という警告が表示される場合があります。この警告は無視しても問題ありません。

関連トピック

- 「ルーティングコンテキストを介したメッセージのトレース」

22.2. ルートトレースの有効化

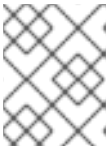
概要

ルーティングコンテキストでメッセージをトレースする前に、ルーティングコンテキストのルートトレースを有効にする必要があります。

手順

ルーティングコンテキストでトレースを有効にするには、以下を実行します。

1. **JMX Navigator** ビューで、トレースを開始する実行中のルーティングコンテキストを選択します。



注記

コンテキストで任意のルートを選択し、コンテキスト全体でトレースを開始できます。

2. 選択したルーティングコンテキストを右クリックしてコンテキストメニューを開き、**Start Tracing** を選択してトレースを開始します。
コンテキストメニューで **Stop Tracing Context** が有効になっている場合、トレースはすでにアクティブです。

関連トピック

- 「ルーティングコンテキストを介したメッセージのトレース」
- 「ルートトレースの無効化」

22.3. ルーティングコンテキストを介したメッセージのトレース

概要

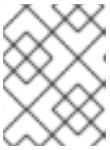
ルーティングコンテキストで何が起こるかを確認する最善の方法は、停止ごとにメッセージに対して何が起こるかを監視することです。このツールは、実行中のルーティングコンテキストにメッセージをドロップし、メッセージが通過するパスを追跡するメカニズムを提供します。

手順

ルーティングコンテキストを介してメッセージをトレースするには、以下を実行します。

1. 「ルートトレース用テストメッセージの作成」 の説明に従って、テストメッセージを 1 つ以上作成します。

2. **Project Explorer** ビューで、プロジェクトの Camel コンテキストファイルを右クリックしてコンテキストメニューを開き、**Run As → Local Camel Context (Tests なし)** を選択します。



注記

プロジェクトの包括的な JUnit テストを作成していない場合は、これを **Local Camel Context** として実行しないでください。

3. 「ルートトレースの有効化」 の説明に従って、実行中のルーティングコンテキストのトレースを有効にします。

4. **Project Explorer** ビューから、テストメッセージの 1 つを **JMX Navigator** ビューのルーティングコンテキスト開始点にドラッグします。

5. **JMX Navigator** ビューで、トレース対象のルーティングコンテキストを選択します。
ツールは、**Messages View** に、トレースされたコンテキストの各段階でメッセージを表すメッセージインスタンスを設定します。

   Diagram View には、選択したルーティングコンテキストがグラフィカル表示されます。

6. **Messages View** で、メッセージインスタンスの 1 つを選択します。
Properties ビューには、メッセージインスタンスの詳細と内容が表示されます。

   Diagram View では、選択したメッセージインスタンスに対応するルートステップが強調表示されます。ルートステップが処理ステップである場合、ツールは既存のパスにタイミングと処理のメトリクスをタグ付けします。

7. 必要に応じて、この手順を繰り返します。

関連トピック

- 「ルートトレース用テストメッセージの作成」
- 「ルートトレースの有効化」
- 「ルートトレースの無効化」

22.4. ルートトレースの無効化

概要

ルーティングコンテキストでルートのデバッグが完了したら、トレースを無効にする必要があります。

重要

トレースを無効にすると、トレースが停止し、ルーティングコンテキスト内のすべてのルートのトレースデータがフラッシュされます。つまり、過去のトレースセッションは確認できなくなります。

手順

ルーティングコンテキストのトレースを停止するには、以下を実行します。

1. **JMX Navigator** ビューで、トレースを無効にする実行中のルーティングコンテキストを選択します。

注記

コンテキストで任意のルートを選択し、コンテキストのトレースを停止できます。

2. 選択したルーティングコンテキストを右クリックしてコンテキストメニューを開き、**Stop Tracing Context** を選択します。
Start Tracing がコンテキストメニューに表示される場合、そのルーティングコンテキストに対するトレースは有効になっていません。

関連トピック

- 「ルートトレースの有効化」
- 「ルーティングコンテキストを介したメッセージのトレース」

第23章 JMS 宛先の管理

JMX Navigator ビューを使用すると、Red Hat Fuse の実行中のインスタンスで JMS 宛先を追加または削除できます。

重要

これらの変更は、ブローカーの再起動後は維持されません。

23.1. JMS 宛先の追加

概要

新しいシナリオをテストする場合は、新しい JMS 宛先をブローカーの 1 つに追加すると便利です。

手順

JMS 宛先をブローカーに追加するには、以下を実行します。

1. **JMX Navigator** ビューの宛先を追加するブローカーノードで、**Queues** の子または **Topics** の子を選択します。
2. 選択したノードを右クリックしてコンテキストメニューを開き、**Create Queue** または **Create Topic** のいずれかを選択します。
3. **Create Queue** または **Create Topic** のダイアログボックスで、新しい宛先の名前を入力します。
4. **OK** をクリックします。
5. **Queues** または **Topics** の子を右クリックし、**Refresh** を選択します。
Queues または **Topics** 子のの下にある **JMX Navigator** ビューに新しい宛先が表示されます。

関連トピック

- 「JMS 宛先の削除」

23.2. JMS 宛先の削除

概要

フェイルオーバーのシナリオ、または障害処理に関連するその他のシナリオをテストする場合は、JMS 宛先を簡単に削除できると便利です。

手順

JMS 宛先を削除するには、以下を実行します。

1. **JMX Navigator** ビューの **Queues** または **Topics** の子で、削除する JMS 宛先を選択します。
2. 選択した宛先を右クリックしてコンテキストメニューを開き、**Delete Queue/Topic** を選択します。

関連トピック

- 「JMS 宛先の追加」

第24章 ルーティングエンドポイントの管理

JMX Navigatorビューを使用すると、ルーティングエンドポイントを追加または削除できます。

重要

これらの変更は、ルーティングコンテキストの再起動後は維持されません。

24.1. ルーティングエンドポイントの追加

概要

新しいシナリオをテストする場合には、新しいエンドポイントをルーティングコンテキストに追加することをお勧めします。

手順

エンドポイントをルーティングコンテキストに追加するには、以下を実行します。

1. **JMX Navigator**ビューのルーティングコンテキストノードで、エンドポイントを追加する**Endpoints**の子を選択します。
2. 選択したノードを右クリックしてコンテキストメニューを開き、**Create Endpoint**を選択します。
3. **Create Endpoint**ダイアログで、新しいエンドポイントを定義するURLを入力します (例: **file://target/messages/validOrders**)。
4. **OK**をクリックします。
5. ルーティングコンテキストノードを右クリックし、**Refresh**を選択します。
   新しい宛先は、**Endpoints**ノードの**JMX Navigator**ビューで、エンドポイントのタイプに対応するフォルダーに表示されます (例: **file**)。

関連トピック

- 「ルーティングエンドポイントの削除」

24.2. ルーティングエンドポイントの削除

概要

フェイルオーバーのシナリオ、または障害処理に関連するその他のシナリオをテストする場合は、ルーティングコンテキストからエンドポイントを削除できると便利です。

手順

ルーティングエンドポイントを削除するには、以下を実行します。

1. **JMX Navigator**ビューで、削除するエンドポイントを選択します。

2. 選択したエンドポイントを右クリックしてコンテキストメニューを開き、**Delete Endpoint** を選択します。
ツールがエンドポイントを削除します。

3. 削除したエンドポイントをビューから削除するには、**Endpoints** ノードを右クリックし、**Refresh** を選択します。
エンドポイントは **JMX Navigator** ビューから消えます。

注記

プロジェクトを再実行することなくエンドポイントのノードを **Project Explorer** ビューから削除するには、ノードを右クリックして **Delete** を選択し、明示的に削除する必要があります。ビューから削除するには、プロジェクト表示を更新します。

関連トピック

- 「ルーティングエンドポイントの追加」

第25章 実行中のルートの編集

概要

プロジェクトのルーティングコンテキストを変更せずに、実行中のルートへの変更を試すことができます。

これを行うには、以下を行います。

- **JMX Navigator** ビューで、実行中のルーティングコンテキストの **Edit Routes** オプションを有効にします。
  これにより、ルートエディターでその in-memory モデル (**Remote CamelContext: <camelContextId>**) が開きます。

- ルートエディターで、ルーティングコンテキストの in-memory モデルを変更します。同時に、Camel デバッガーとそのすべての機能を使用するために、関連するノードにブレークポイントを設定できます。
  in-memory インメモリーモデルを編集して、ノードの追加、削除、または再配置、既存ノードのプロパティーの追加または削除、既存ノードに設定されているプロパティー値の変更を行えます。実行中のコンテキストを更新して **Debug** パースペクティブに結果を表示するには、in-memory モデルに加えた変更を保存する必要があります。

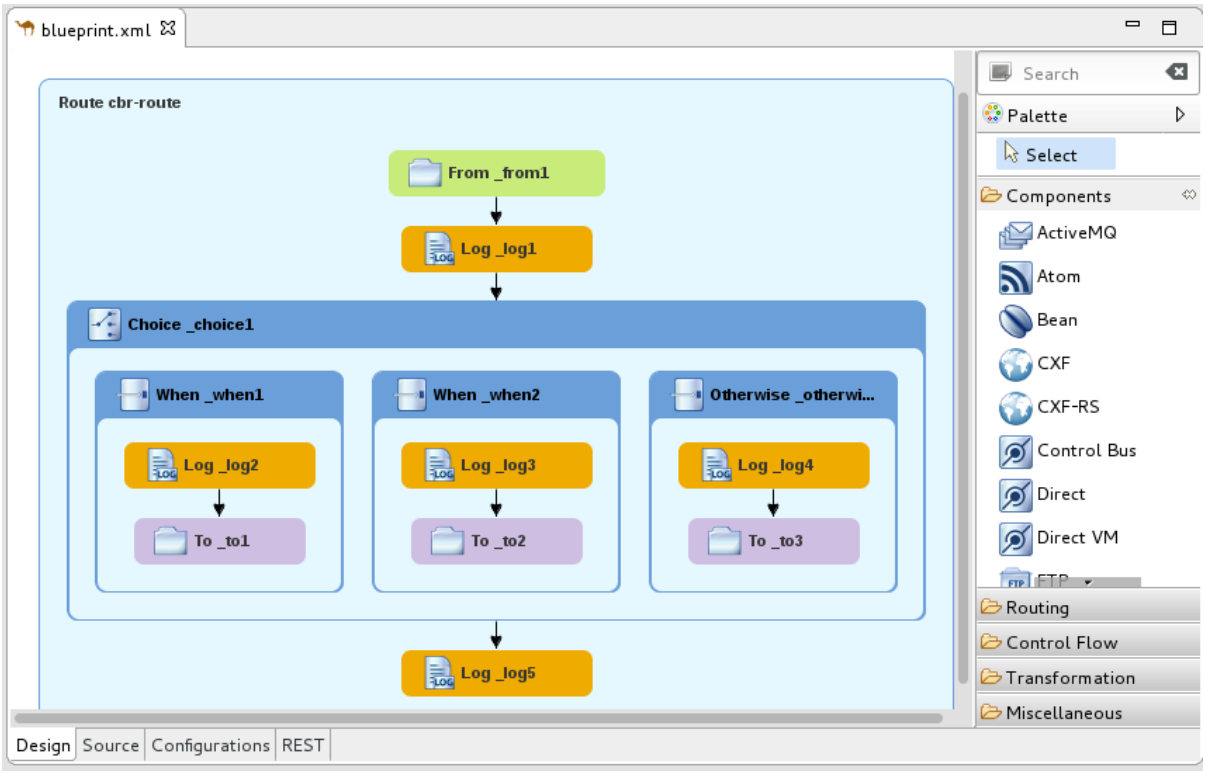
- **JMX Navigator** ビューで、実行中のルーティングコンテキストにメッセージをドロップするか、タイマー、ActiveMQ、ファイル、またはその他の継続的な入力ノードからメッセージが到着するのを待ちます。

- **Debug** パースペクティブでは、結果を評価し、Camel デバッガーを使用してルーティングコンテキストの詳細を把握します。

実行中のルートの変更および結果の評価

1. **JMX Navigator** ビューで、編集するルートが含まれるルーティングコンテキストを選択します。

2. 選択したルーティングコンテキストを右クリックしてコンテキストメニューを開き、**Edit Routes** を選択します。
   ルートエディターがルーティングコンテキストの in-memory モデル (**Remote CamelContext: <contextId>**) を開き、コンテキスト内のすべてのルートを表示します。以下はその例です。



注記

<contextId> は、プロジェクトのルーティングコンテキストの **xml** ファイルの **camelContext** 要素の ID です。 **Fuse → Content Based Router** のビルトインテンプレートをベースとするこの例では、ID は **cbr-example-context** です。

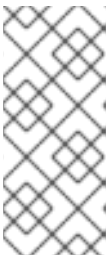
3. *2章ルートエディターを使用したルーティングコンテキストの編集* の説明に従ってルートを編集し、**File → Save** を選択して in-memory モデルに加えた変更を保存し、実行中のルーティングコンテキストを更新します。

4. *13章 ブレークポイントの設定* の説明に従って、関連するノードにブレークポイントを設定します。

5. **JMX Navigator** ビューで、実行中のルーティングコンテキストの入力ノードにメッセージをドロップします。
プロジェクトにテストメッセージが含まれていない場合は、「ルートトレース用テストメッセージの作成」 の説明に従って作成できます。

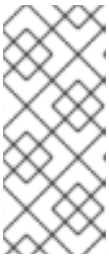
6. **Yes** をクリックし、**Debug** パースペクティブに切り替えます。

7. Camel デバッガーで、通常どおり (*14章Camel デバッガーの実行* を参照) ブレークポイントを介してメッセージをステップし、変更によって生成される結果を確認します。
Camel デバッガーは、**Edit Routes** モードでも通常のデバッグモードと同じように動作するため、メッセージがルーティングコンテキストを通過している間、Camel デバッガーの任意の機能を使用できます。



注記

メッセージがルーティングコンテキストの最後に到達すると、デバッガーが停止します。デバッグを続行するには、**Fuse Integration** パースペクティブに戻し、**JMX Navigator** ビューの入力ノードに別のメッセージをドロップします。そうするたびに、ツールは **Debug** パースペクティブへの切り替えを確認するように要求します。



注記

ルート編集セッション中に、実行中のルーティングコンテキストへの接続が失われる可能性があります。この場合は、**JMX Navigator** ビューに次のように表示されます: **Local Processes → maven[xxxx][Disconnected]**。セッションを続行するには、実行中のルーティングコンテキストに再接続し、**JMX Navigator** ビューでこれを選択してから **Edit Routes** を再度選択する必要があります。

ルート編集セッションの終了

1. **Debug** パースペクティブの **Debug** ビューで、**Remote Camel Debug - camelContext--<contextId>--xxxxxxxxxxxxxxxxxx.xml [Remote Camel Context]** スレッドを選択し、メニューバーの  をクリックしてデバッグセッションを終了します。

2. **コンソール** ビューのメニューバーで  をクリックし、ルーティングコンテキストを終了します。

3. コンソール出力をクリアする場合は、**Console** ビューのメニューバーの  をクリックします。

4. **Fuse Integration** パースペクティブに切り替え、ルートエディターで **Remote CamelContext: <contextId>** タブの  をクリックし、ルーティングコンテキストファイルのメモリー内モデルを閉じます。

関連トピック

- *2章ルートエディターを使用したルーティングコンテキストの編集*
- パートII「ルーティングコンテキストのデバッグ」

第26章 ルーティングコンテキストの管理

JMX Navigator ビューを使用すると、実行中のルーティングコンテキストを一時停止および再開できます。

26.1. ルーティングコンテキストの一時停止操作

概要

このツールを使用すると、JMX Navigator ビューでルーティングコンテキストの操作を一時停止できます。コンテキスト操作を一時停止すると、コンテキストのすべてのルートが正常にシャットダウンされますが、操作を再開できるようにメモリーにロードされた状態を維持します。

手順

ルーティングコンテキストの操作を一時停止するには、以下を実行します。

1. JMX Navigator ビューで、プロジェクトの Camel Contexts ノードを展開し、一時停止する操作のルーティングコンテキストを選択します。
2. 選択したルーティングコンテキストを右クリックしてコンテキストメニューを開き、Suspend Camel Context を選択します。

注記

コンテキストメニューに Resume Camel Context が表示される場合、コンテキストの操作はすでに一時停止されています。

関連トピック

- 「ルーティングコンテキストの再開操作」

26.2. ルーティングコンテキストの再開操作

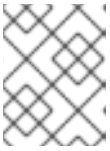
概要

ツールを使用すると、JMX Navigator ビューで一時停止されたルーティングコンテキストの操作を再開できます。コンテキストの操作を再開すると、コンテキスト内のすべてのルートが再開され、メッセージを処理できるようになります。

手順

ルーティングコンテキストの操作を再開するには、以下を実行します。

1. JMX Navigator ビューで、プロジェクトの Camel Contexts ノードを展開し、操作を再開するルーティングコンテキストを選択します。
2. 選択したコンテキストを右クリックしてコンテキストメニューを開き、Resume Camel Context を選択します。



注記

コンテキストメニューに **Suspend Camel Context** が表示される場合、コンテキストとそのルートは実行されています。

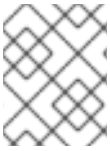
関連トピック

- 「ルーティングコンテキストの一時停止操作」

パート IV. コンテナーへのアプリケーションの公開

Fuse Integration プロジェクトをサーバーコンテナーに公開するには、最初にサーバーとそのランタイム定義をツールの **Server** リストに追加する必要があります。その後、プロジェクトをサーバーランタイムに割り当て、その公開オプションを設定できます。

第27章 サーバーの管理



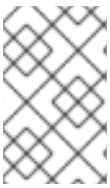
注記

Camel プロジェクトを Red Hat Fuse に公開するための詳細な手順については、28 章*Fuse Integration プロジェクトのサーバーへの公開* を参照してください。

27.1. サーバーの追加

概要

ツールを使用してサーバーを管理するには、サーバーを **Servers** リストに追加する必要があります。追加すると、サーバーは **Servers** ビューに表示されます。ここから、サーバーに接続して Fuse Integration プロジェクトを公開できます。



注記

Red Hat Fuse サーバーを追加する場合は、その **installDir/etc/users.properties** ファイルを編集し、**user=password,role** の形式でユーザー情報を追加して、ツールがサーバーへの SSH 接続を確立できるようにすることが推奨されます。

手順

新しいサーバーを **Servers** ビューに追加する方法は 3 つあります。

- Servers ビューで **No servers are available. Click this link to create a new server...**をクリックします。

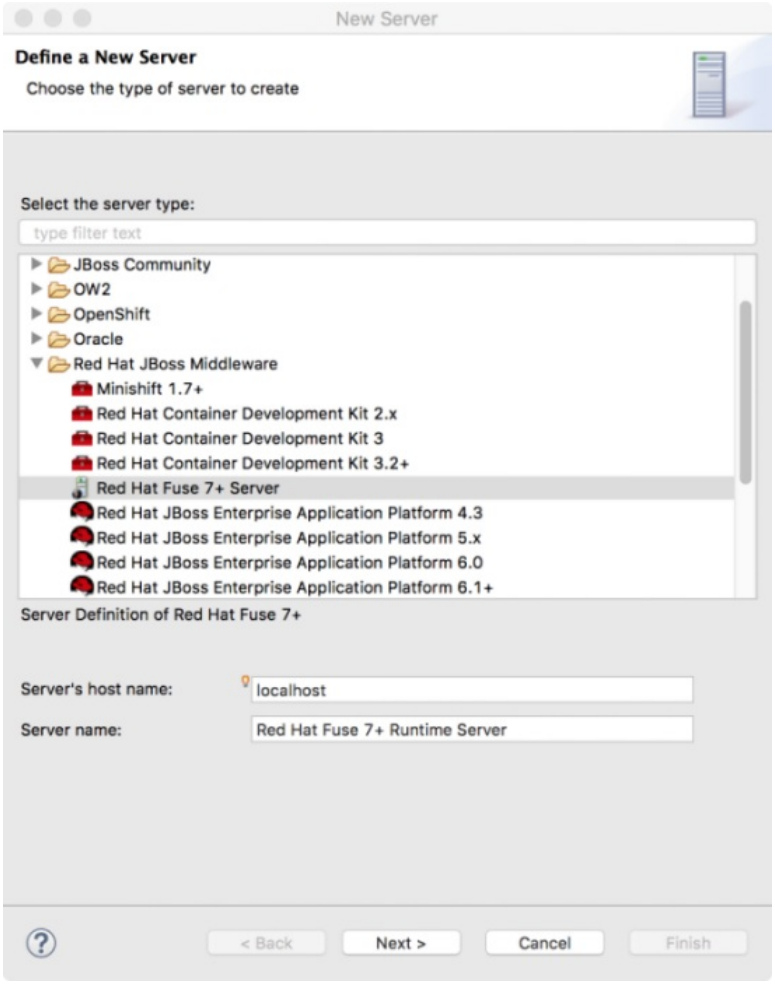
  注記

  このリンクは、サーバーが定義されていない場合にのみ **Servers** ビューに表示されます。プロジェクトを最初に作成したときにサーバーを定義して追加した場合は、**Servers** ビューにそのサーバーが表示されます。

- **Servers** ビューで右クリックしてコンテキストメニューを開き、**New → Server** の順に選択します。
- メニューバーで **File → New → Other → Server → Server** の順に選択します。

Define a New Server ダイアログボックスで、新しいサーバーを追加します。

1. **Red Hat JBoss Middleware** ノードを展開して、使用可能なサーバーオプション一覧を公開します。

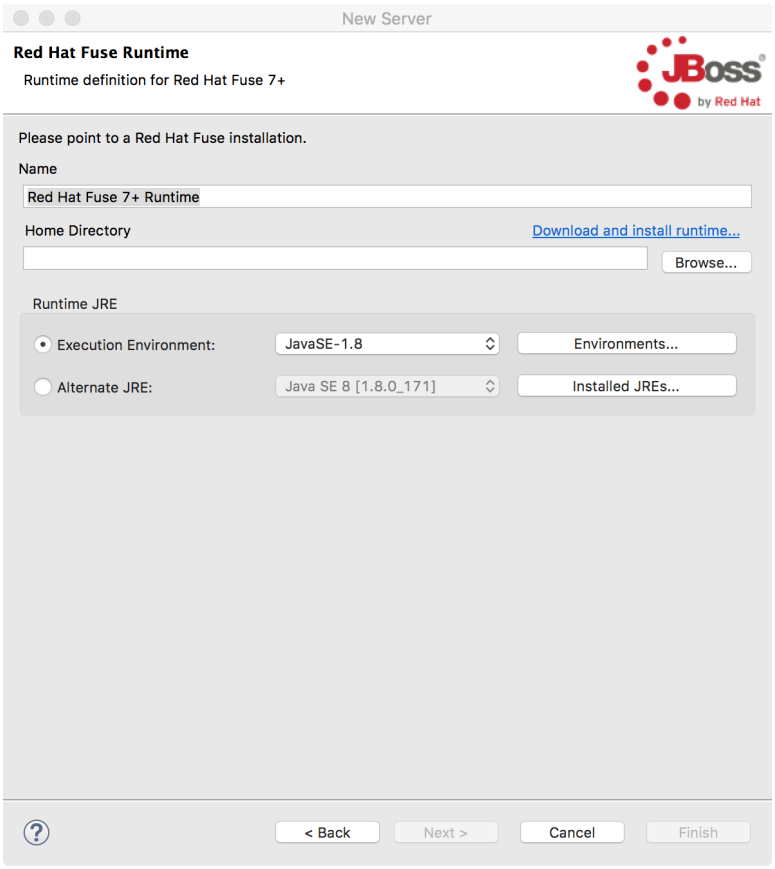


2. 追加するサーバーをクリックします。
3. **Server's host name** フィールドで、デフォルト (**localhost**) を受け入れます。

注記

localhost のアドレスは **0.0.0.0** です。

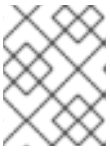
4. **Server name** フィールドで、デフォルトを使用するか、ランタイムサーバーの別の名前を入力します。
5. **Server runtime environment** で、デフォルトを使用するか、**Add** をクリックしてサーバーのランタイム定義ページを開きます。



注記

サーバーがマシンにインストールされていない場合は、**Download and install runtime** をクリックしサイトのダウンロード手順に従って、ここでサーバーをインストールできます。サイトによっては、ダウンロードプロセスを続行する前に、有効なクレデンシャルの提供が必要なこともあります。

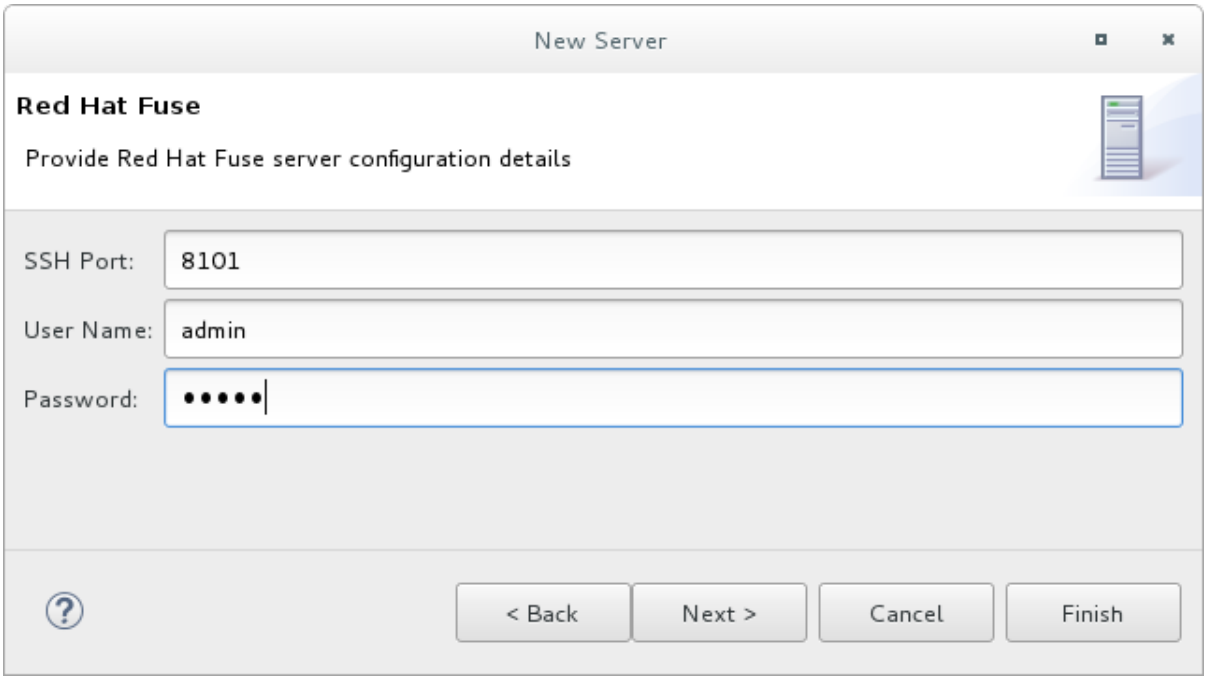
6. インストール **Name** のデフォルトを使用します。
7. **Home Directory** フィールドに、サーバーランタイムがインストールされているパスを入力するか、**Browse** をクリックして検索し、選択します。
8. **Execution Environment** の横にあるドロップダウンメニューからランタイム JRE を選択します。
一覧に目的のバージョンが表示されない場合は、**Environments** をクリックして、表示される一覧からバージョンを選択します。選択した JRE バージョンがマシンにインストールされている必要があります。



注記

必要な Java バージョンについては、Red Hat Fuse Supported Configurations を参照してください。

9. **Alternate JRE** オプションはそのままにしておきます。
10. **Next** をクリックしてサーバーのランタイム定義を保存し、その **Configuration details** ページを開きます。



11. **SSH Port** のデフォルト (**8101**) を受け入れます。
ランタイムは SSH ポートを使用して、サーバーの Karaf シェルに接続します。このデフォルトがご自分のセットアップに対して正しくない場合は、サーバーの **installDir/etc/org.apache.karaf.shell.cfg** ファイルを確認して正しいポート番号を検出できます。

12. **User Name** フィールドに、サーバーへのログインに使用する名前を入力します。
Red Hat Fuse では、これは Red Hat Fuse の **installDir/etc/users.properties** ファイルに保存されているユーザー名です。

注記

/etc/users.properties ファイルのデフォルトのユーザーがアクティベートされている場合 (アンコメント)、[servCnfigDetails] に記載のとおり、ツールは **User Name** および **Password** フィールドにデフォルトユーザーの名前およびパスワードを自動入力します。

ユーザーが設定されていない場合は、**user=password,role** のフォーマットを使用してそのファイルにユーザーを追加するか (例: **joe=secret,Administrator**)、karaf **jaas** コマンドセットを使用して設定できます。

- **jaas:realms**: レルムを一覧表示します。
- **jaas:manage --index 1**: 最初の (サーバー) レルムを編集します。
- **jaas:useradd <username> <password>**: ユーザーと関連するパスワードを追加します。
- **jaas:roleadd <username> Administrator**: 新規ユーザーのロールを指定します。
- **jaas:update**: 新しいユーザー情報を使用してレルムを更新します。

サーバーに対して **jaas** レルムがすでに選択されている場合は、コマンド **JBossFuse:karaf@root>jaas:users** を実行してユーザー名を検出できます。

13. **Password** フィールドには、サーバーにログインする際に **User name** で必要なパスワードを入力します。

14. **Finish** をクリックしてサーバー設定の詳細を保存します。
サーバーランタイムが **Servers** ビューに表示されます。

サーバーノードを展開すると、サーバーの JMX ノードが公開されます。

27.2. サーバーの起動

概要

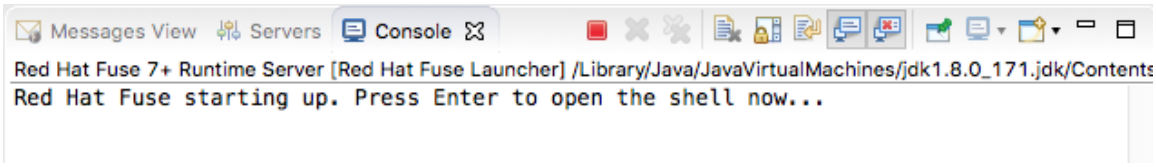
設定済みサーバーを起動すると、ツールはサーバーのリモート管理コンソールを **Terminal** ビューで開きます。これにより、アプリケーションのテスト中にコンテナーを簡単に管理できます。

手順

サーバーを起動するには、以下を実行します。

1. **Servers** ビューで、起動するサーバーを選択します。
2.  をクリックします。

- **Console** ビューが開き、コンテナーの起動中は待機するように求めるメッセージが表示されます。以下はその例です。

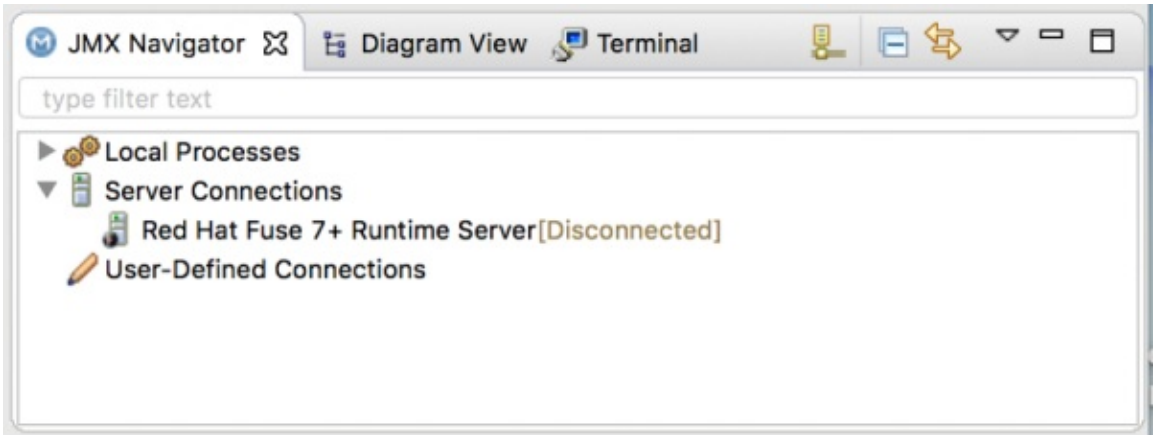


注記

リモートコンソールを開くためのユーザー名とパスワードを適切に設定しなかった場合は、ダイアログボックスが開き、適切な認証情報の入力が求められます。「サーバーの追加」を参照してください。

- コンテナーが起動すると、**Terminal** ビューが開き、コンテナーの管理コンソールが表示されます。
- 稼働中のサーバーが **Servers** ビューに表示されます。

- 稼働中のサーバーは、**Server Connections** の **JMX Navigator** ビューにも表示されます。



注記

サーバーがツールと同じマシン上で稼働している場合、そのサーバーは **Local Processes** の下にもエントリーがあります。

27.3. 稼働中のサーバーへの接続

概要

設定済みのサーバーを起動すると、**Servers** ビューと、**Server Connections** ノードの **JMX Navigator** ビューに表示されます。サーバーを表示するには、**Server Connections** ノードを展開する必要がある場合があります。

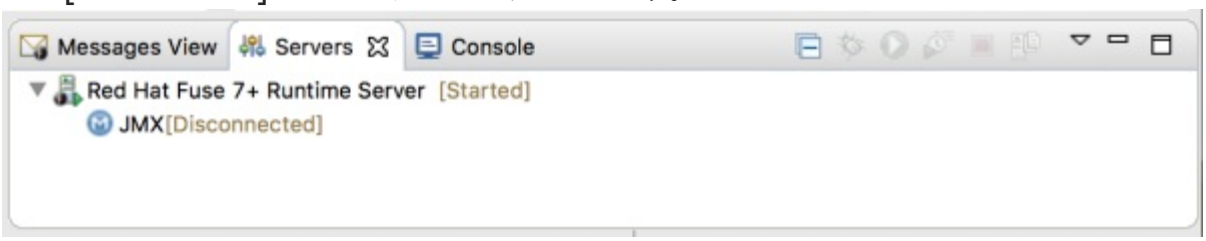
稼働中のサーバーで Fuse プロジェクトアプリケーションを公開およびテストするには、まず接続する必要があります。**Servers** ビューまたは **JMX Navigator** ビューのいずれかで、稼働中のサーバーに接続できます。

注記

Servers ビューと **JMX Navigator** ビューは、サーバー接続に関して同期されます。つまり、**Servers** ビューのサーバーに接続すると、**JMX Navigator** ビューでもサーバーに接続され、その逆も同様に接続されます。

Servers ビューで稼働中のサーバーへの接続

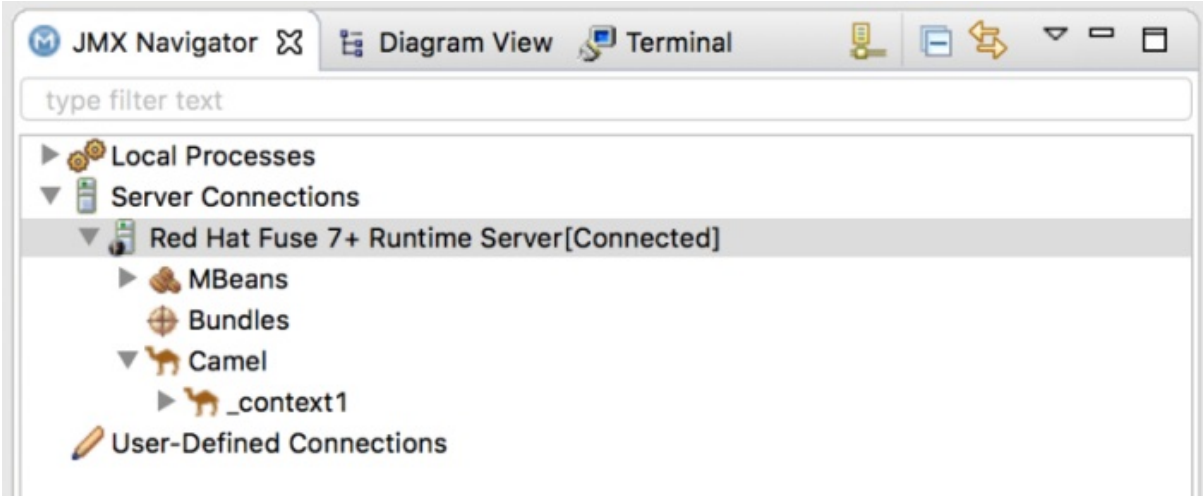
1. **Servers** ビューで、サーバーランタイムを展開して **JMX[Disconnected]** ノードを公開します。
2. **JMX[Disconnected]** ノードをダブルクリックします。



JMX Navigator ビューで稼働中のサーバーへの接続

1. **Server Connections** ノードの **JMX Navigator** ビューで、接続するサーバーを選択します。

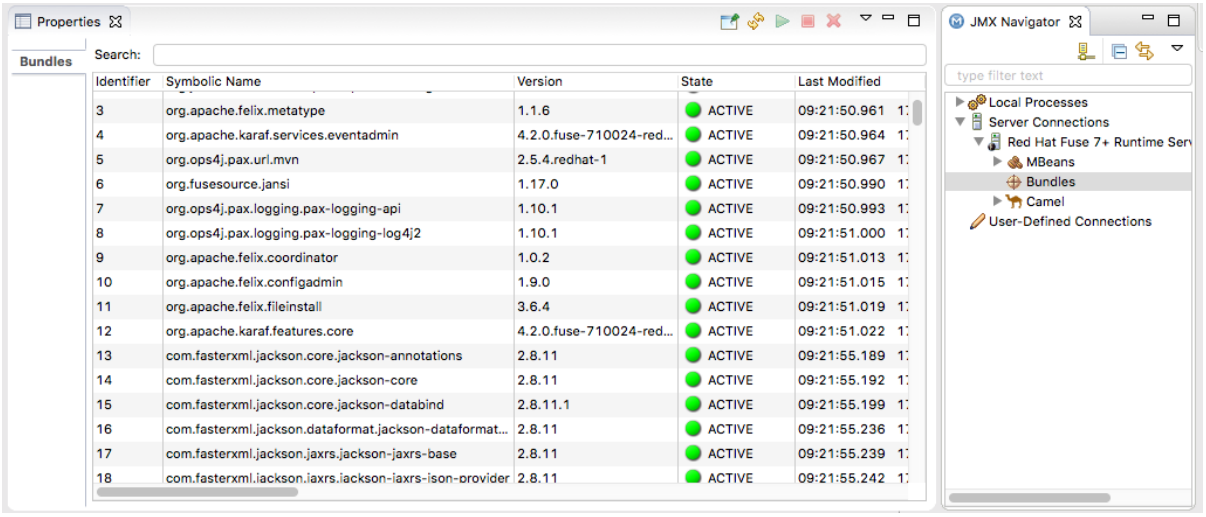
2. 選択したサーバーをダブルクリックします。



接続されたサーバーにインストールされているバンドルの表示

1. Servers ビューまたは JMX Navigator ビューで、サーバーランタイムツリーを展開して Bundles ノードを公開し、これを選択します。

2. このツールは、Properties ビューに、サーバーにインストールされているバンドルの一覧を表示します。



Properties ビューの Search ツールを使用し、わかっている場合は Symbolic Name または Identifier でバンドルを検索できます。記号名または識別子を入力すると一覧が更新され、現在の検索文字列に一致するバンドルのみが表示されます。

注記

または、Terminal ビューで **osgi:list** コマンドを実行し、Red Hat Fuse サーバーランタイムにインストールされているバンドルに関する生成されたリストを表示することもできます。ツールは、**osgi:list** コマンドが表示する OSGi バンドルに異なる命名スキームを使用します。

プロジェクトの **pom.xml** ファイルの **<build>** セクションで、バンドルのシンボリック名およびそのバンドル名 (OSGi) が **maven-bundle-plugin** エントリーに一覧表示されます。詳細については 「プロジェクトのサーバーへの公開を確認」 を参照してください。

27.4. サーバーからの接続解除

概要

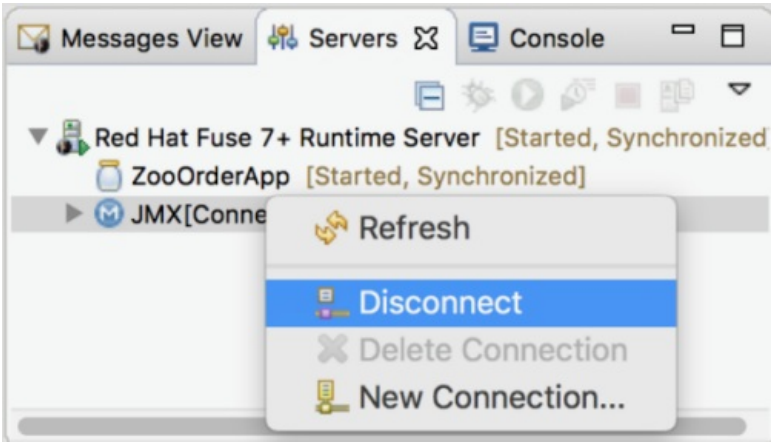
アプリケーションのテストが終了したら、サーバーを停止せずに接続を解除できます。

注記

Servers ビューと JMX Navigator ビューは、サーバー接続に関して同期されます。つまり、Servers ビューのサーバーからの接続を解除すると、JMX Navigator ビューでも接続解除されます。また、その逆も同様です。

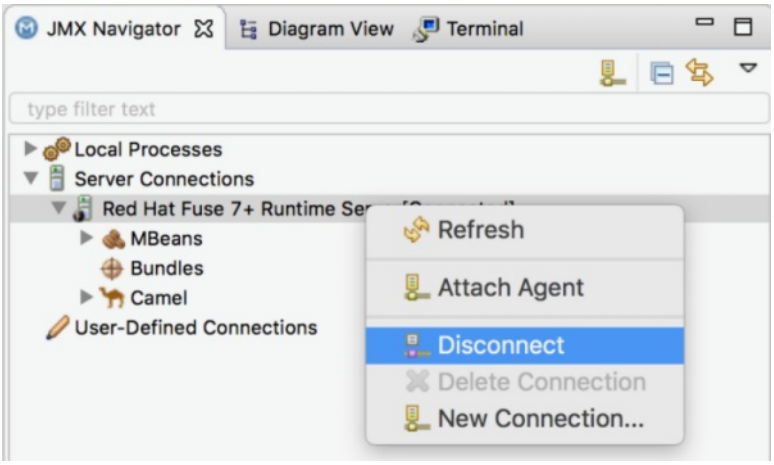
Servers ビューのサーバーからの接続解除

1. Servers ビューで、サーバーランタイムを展開して JMX[Connected] ノードを公開します。
2. JMX[Connected] ノードを右クリックしてコンテキストメニューを開き、Disconnect を選択します。



JMX Navigator ビューのサーバーからの接続解除

1. JMX Navigator ビューの Server Connections で、接続を解除するサーバーを選択します。
2. 選択したサーバーを右クリックしてコンテキストメニューを開き、Disconnect を選択します。



27.5. サーバーの停止

概要

Servers ビュー、または Terminal ビューのサーバーのリモートコンソールで、サーバーをシャットダウンできます。

Servers ビューの使用

サーバーを停止するには、以下を実行します。

1. Servers ビューで、停止するサーバーを選択します。
2.  をクリックします。

リモートコンソールの使用

サーバーを停止するには、以下を実行します。

1. サーバーのリモートコンソールをホストしている Terminal ビューを開きます。
2. **Ctrl**+**D** を押します。

27.6. サーバーの削除

概要

設定済みサーバーの処理が終了した場合、またはサーバーを誤って設定した場合は、サーバーとその設定を削除できます。

まず、Servers ビューまたは JMX Navigator ビューからサーバーを削除します。次に、サーバー設定を削除します。

サーバーの削除

1. Servers ビューで、削除するサーバーを右クリックしてコンテキストメニューを開きます。
2. **Delete** を選択します。
3. **OK** をクリックします。

サーバーの設定の削除

Linux および Windows マシンでは、**Window → Preferences** の順に選択します。

1. Server フォルダーを展開し、**Runtime Environments** を選択して Server Runtime Environments ページを開きます。
2. 一覧で、Servers ビューから以前に削除したサーバーのランタイム環境を選択し、**Remove** をクリックします。
3. **OK** をクリックします。

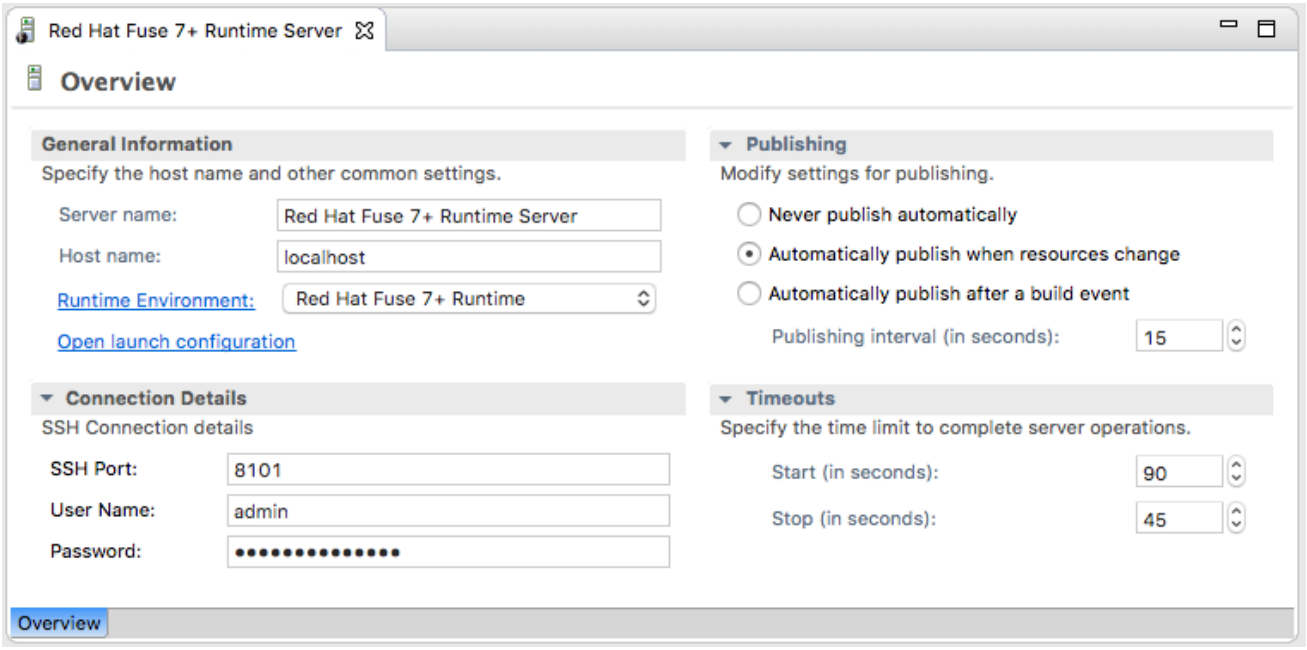
第28章 FUSE INTEGRATION プロジェクトのサーバーへの公開

Eclipse 公開メカニズムを使用して、Fuse Integration プロジェクトをサーバーランタイムにデプロイします。これを行うには、サーバーを **Fuse Integration** パースペクティブの **Servers** ビューに定義および追加する必要があります。詳しい手順を示すデモについては、以下を参照してください。

概要

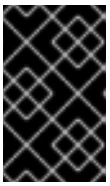
サポート対象サーバーを、割り当てられた Fuse プロジェクトを自動的に公開するか、publish コマンドを手動で呼び出した場合にのみ公開するように設定できます。

Servers ビューに追加された各サーバーランタイムには、その設定、接続、および公開の詳細が含まれる独自の **Overview** ページがあります。



Publishing を展開して、サーバーランタイムの公開オプションとデフォルト設定を公開する必要がある場合があります。

- **Never publish automatically** – プロジェクトを手動で公開する場合は、このオプションを選択する必要があります。



重要

また、サーバーの **Add and Remove** ページで、**If server started, publish changes immediately** オプションを無効にする必要があります (詳細については 「Fuse プロジェクトの手動公開」 を参照)。

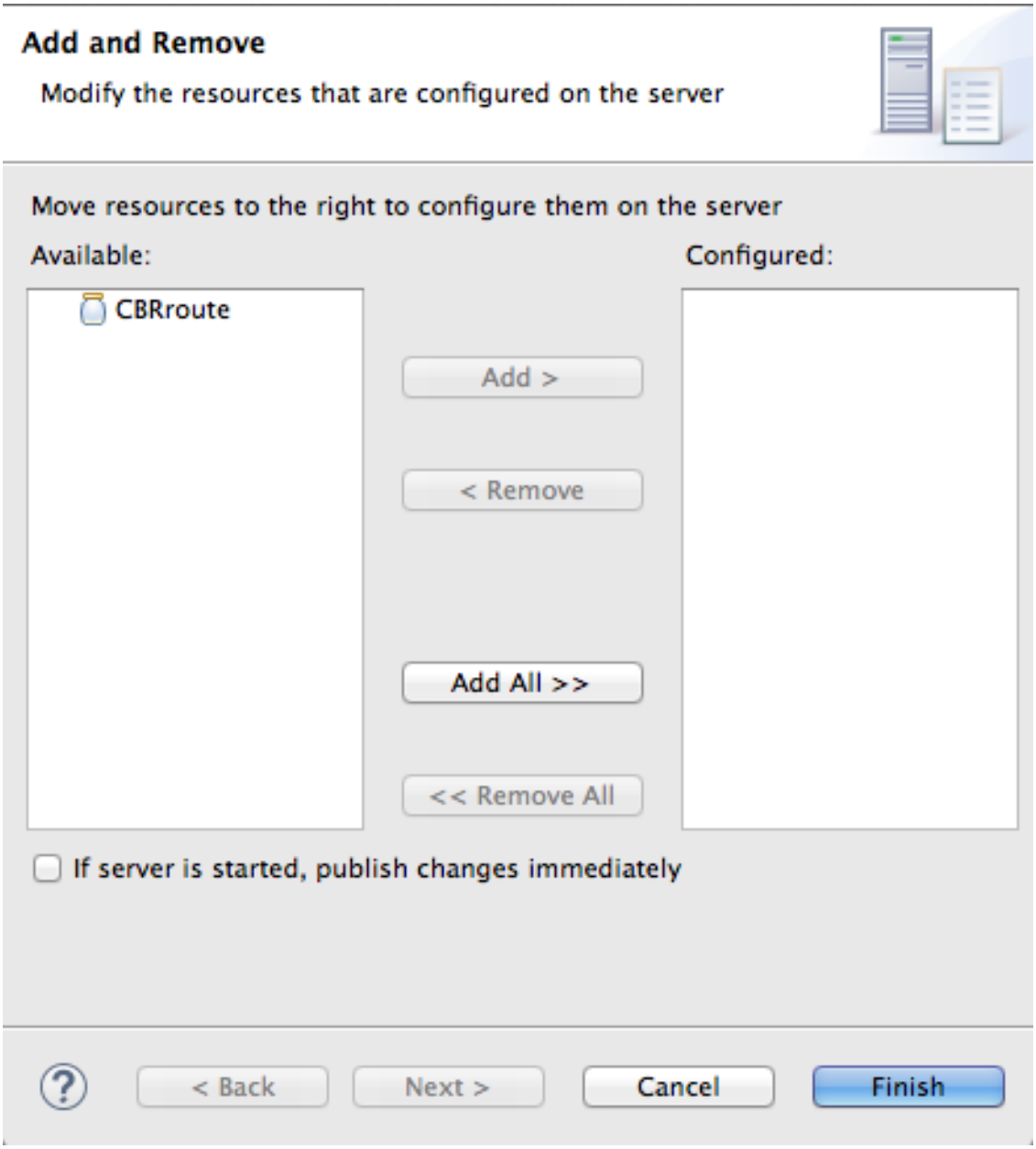
- **Automatically publish when resources change** – [default] Fuse プロジェクトへの変更を保存するときに、その Fuse プロジェクトを自動的に公開または再公開する場合は、このオプションを有効にします。プロジェクトの公開速度は、**Publishing interval** (デフォルトは 15 秒) によって異なります。

- **Automatically publish after a build event**– Fuse プロジェクトの場合、 **Automatically publish when resources change** と同じように機能します。

リソース変更時に FUSE プロジェクトを自動公開

サーバーランタイムのデフォルトの公開オプションは、**Automatically publish when resources change** です。

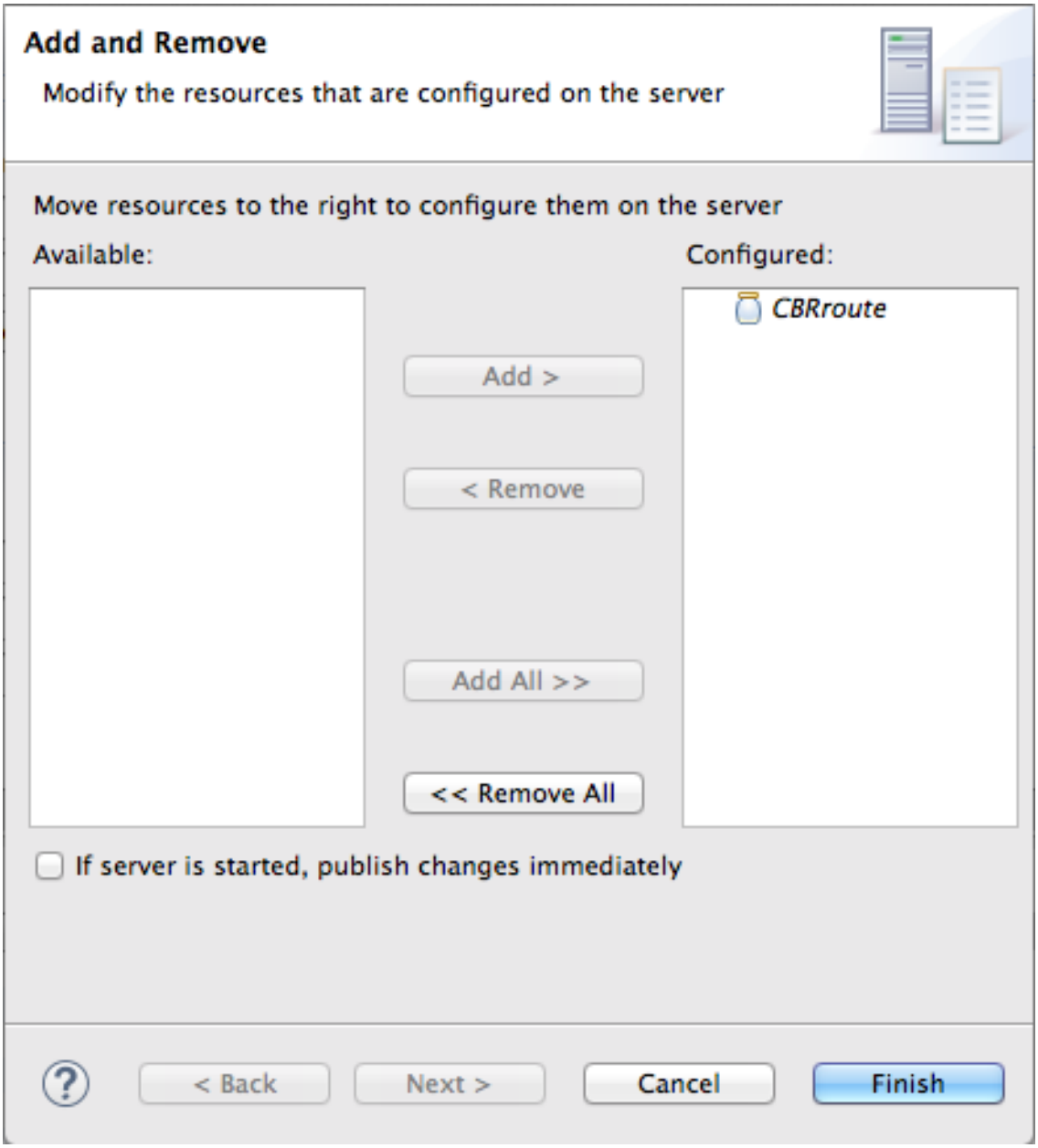
1. 必要に応じて、Fuse プロジェクトを公開するサーバーランタイムを起動します。詳細は、「サーバーの起動」 を参照してください。
2. **Servers** ビューで、サーバーランタイムをダブルクリックして **Overview** ページを開きます。
3. **Publishing** を展開し、**Automatically publish when resources change**を選択します。
4. 公開サイクルの間隔を増減するには、必要に応じて **Publishing interval (in seconds)** の横にあるラジオボタンをクリックします。
5. **Servers** ビューでサーバーランタイムを右クリックし、コンテキストメニューを開き、**Add and Remove** を選択します。



公開に使用できるすべてのリソースが **Available** 列に表示されます。

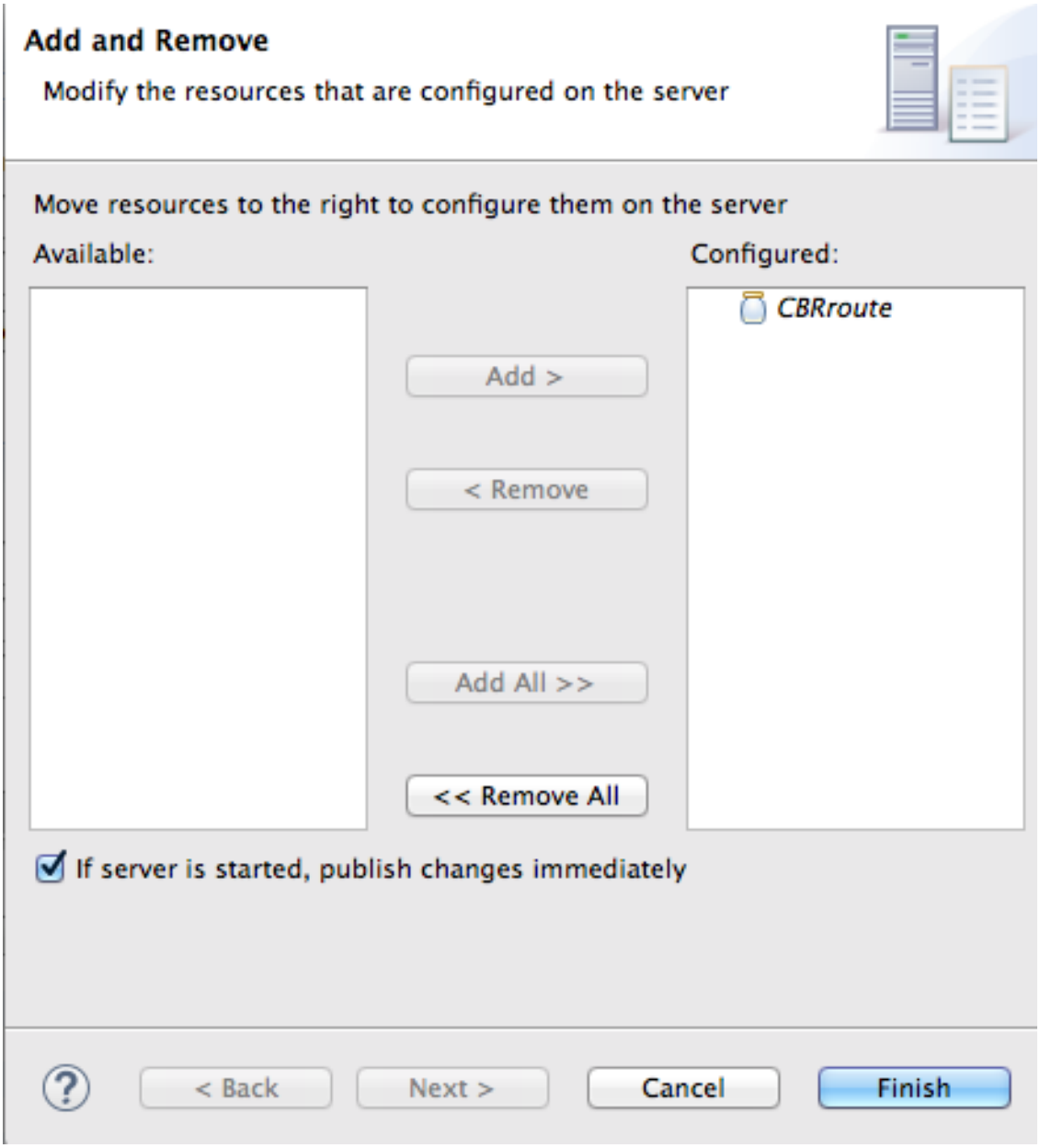
6. リソース (この場合は **CBRroute** Fuse プロジェクト) をサーバーランタイムに割り当てるには、以下を実施します。
   - ダブルクリックします。または

- これを選択して **Add** をクリックします。
選択したリソースは **Configured** 列に移動します。



この段階で、割り当てられたリソースが実際に公開される時間は、サーバーランタイムが実行されているかどうか、および **Publishing interval** の設定によって異なります。ただし、サーバーが停止している場合は、サーバー起動後にプロジェクトを手動で公開する必要があります (詳細については、「Fuse プロジェクトの手動公開」 を参照してください)。

7. **If server started, publish changes immediately**オプションをクリックして有効にします。



このオプションを使用すると、**Finish** をクリックすることで設定したプロジェクトが即座に公開されます。サーバーランタイムの **Overview 要** ページの **Automatically publish when resources change** オプションを使用すると、ローカルプロジェクトに加えられた変更が保存されるたびに設定済みのプロジェクトが再公開されます。

8. **Finish** をクリックします。
プロジェクトが **Servers** ビューのサーバーランタイムノードに表示され、サーバーランタイムのステータスが **[Started,Publishing...]** と報告されます。

公開が完了すると、サーバーランタイムとプロジェクトのステータスは **[Started,Synchronized]** と報告されます。

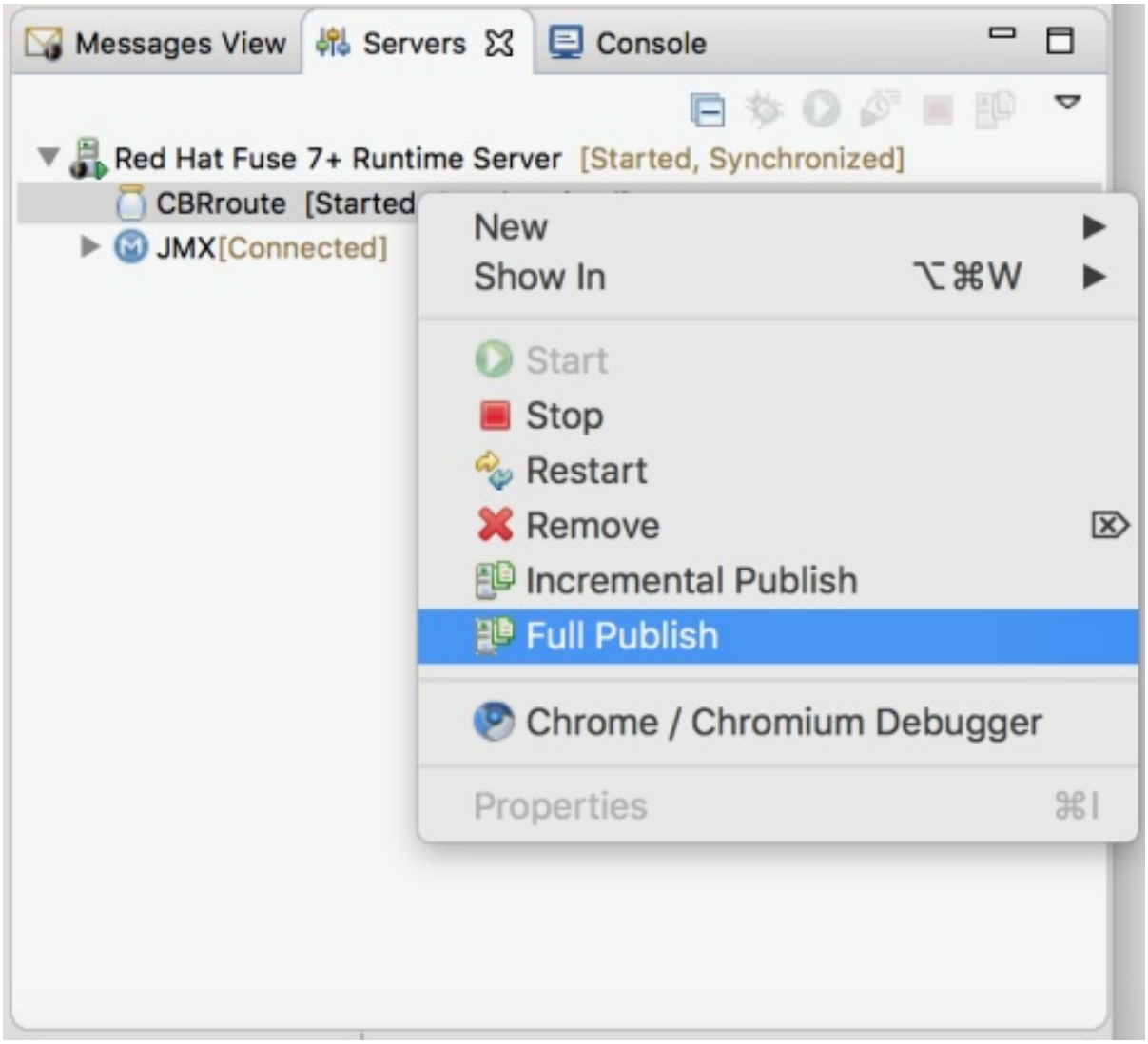
注記

サーバーランタイムの場合、**synchronized** は、サーバーのすべての公開済みリソースが、ローカルの対応する公開済みリソースと同じであることを意味します。公開済みリソースの場合、**Synchronized** は、ローカルの対応するリソースと同じであることを意味します。

FUSE プロジェクトの手動公開

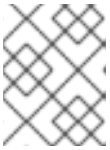
1. 必要に応じて、Fuse プロジェクトを公開するサーバーランタイムを起動します。詳細は、「サーバーの起動」 を参照してください。
2. **Servers** ビューで、サーバーランタイムをダブルクリックして **Overview** ページを開きます。
3. **Publishing** を展開し、**Never publish automatically** を選択します。
4. **File → Save** をクリックして、公開オプションの変更を保存します。
5. Fuse プロジェクトがすでにサーバーランタイムに割り当てられている場合は、**If server started, publish changes immediately** のオプションが無効であることを確認します。
   a. **Servers** ビューで、サーバーランタイムを右クリックしてコンテキストメニューを開きます。
   b. **Add and Remove...** をクリックしてサーバーの **Add および Remove** ページを開きます。
   c. **If server started, publish changes immediately**オプションが有効になっている場合は無効にします。
   d. [finish] に進みます。
6. Fuse プロジェクトがサーバーランタイムに割り当てられていない場合は、ここで割り当てます。
   a. 「リソース変更時に Fuse プロジェクトを自動公開」 の [startAssignResource] から [stopAssignResource] に従います。
   b. **If server started, publish changes immediately**オプションは有効にしないでください。
7. **Finish** をクリックします。
   プロジェクトが **Servers** ビューのサーバーランタイムノードに表示され、サーバーランタイムのステータスが **[Started]** と報告されます。

8. **Servers** ビューで、プロジェクトのノードを右クリックします。この例では、**CBRroute** Fuse プロジェクトを選択してコンテキストメニューを開きます。



9. **Full Publish** を選択します。
公開操作中に、サーバーランタイムとプロジェクトのステータスは **[Started,Republish]** と報告されます。

公開が完了すると、サーバーランタイムとプロジェクトの両方のステータスが **[Started,Synchronized]** と報告されます。



注記

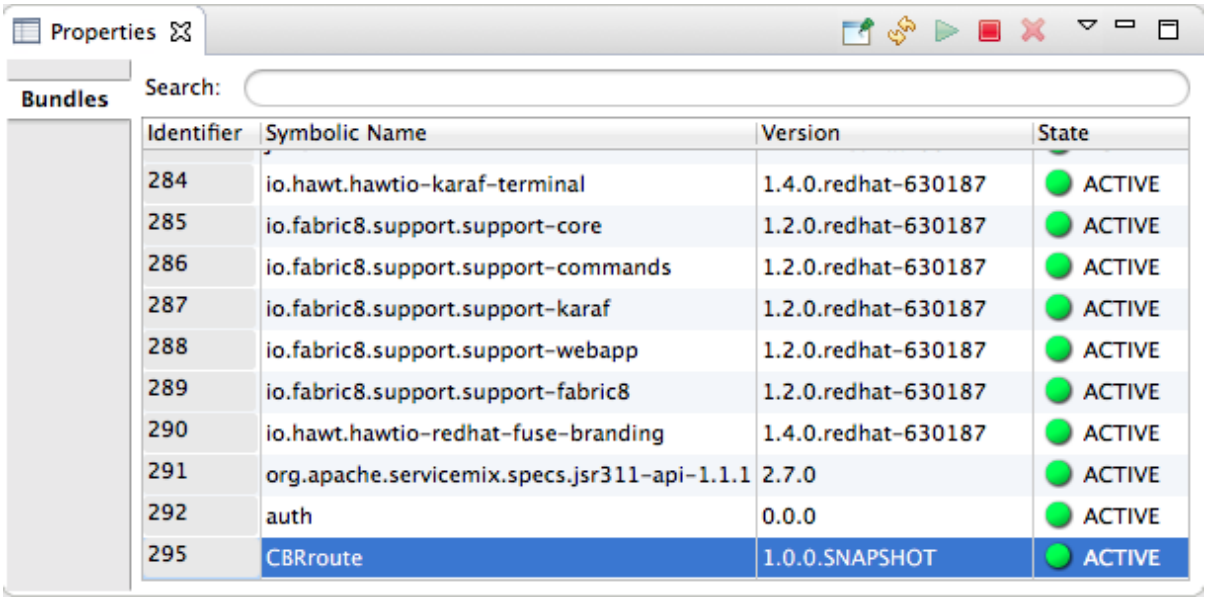
このツールは **Incremental Publish** オプションをサポートしません。**Incremental Publish** をクリックすると、完全公開になります。

プロジェクトのサーバーへの公開を確認

Fuse プロジェクトをサーバーランタイムに公開した後、サーバーに接続し、プロジェクトのバンドルがインストールされていることを確認できます。

1. サーバーランタイムに接続します。詳細は、「Servers ビューで稼働中のサーバーへの接続」を参照してください。

2. **Servers** ビューで、サーバーランタイムツリーを展開して **Bundles** ノードを公開し、これを選択します。
このツールは、**Properties** ビューに、サーバーにインストールされているバンドルの一覧を表示します。



3. プロジェクトのバンドルを検索するには、リストの下部までスクロールするか、**Properties** ビューの **Search** ボックスにバンドルの **Symbolic Name** を入力します。バンドルの **Symbolic Name** は、作成時にプロジェクトに付与した名前です。

注記

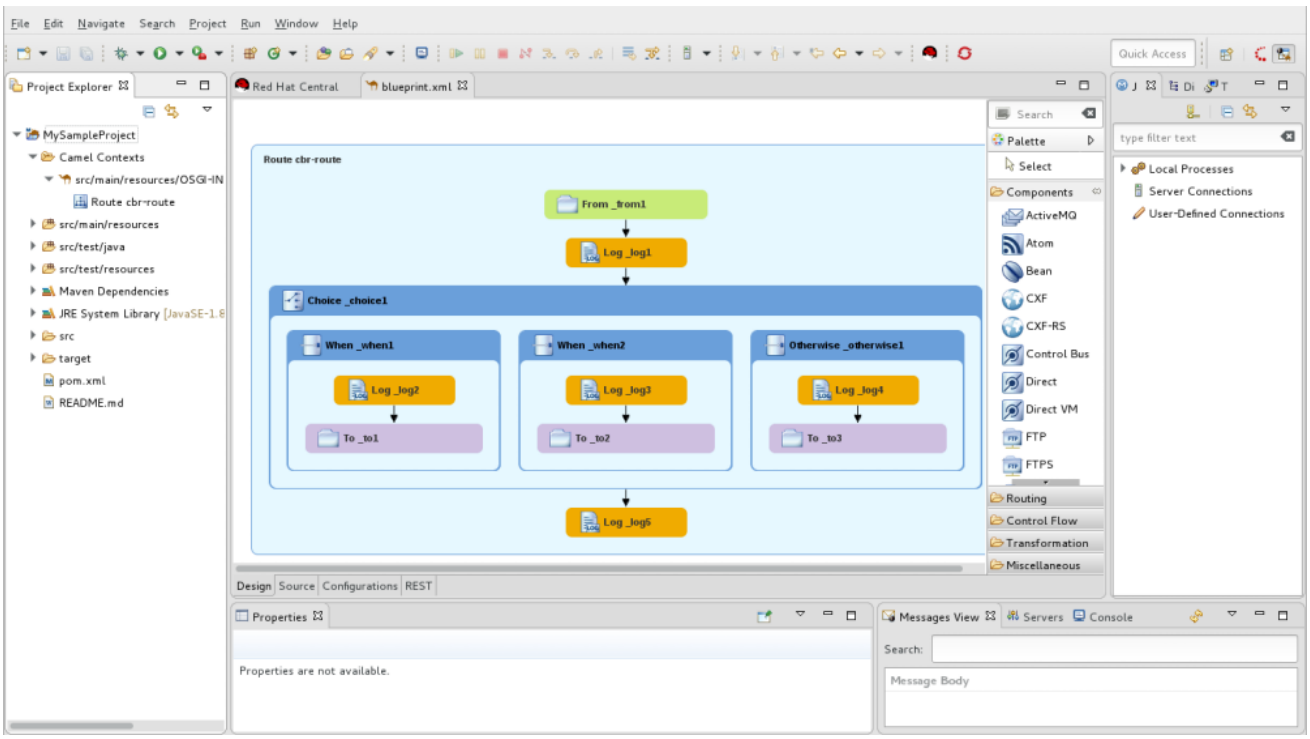
または、**Terminal** ビューで **osgi:list** コマンドを実行し、Fuse サーバーランタイムにインストールされているバンドルに関する生成されたリストを表示することもできます。ツールは、**osgi:list** コマンドが表示する OSGi バンドルに異なる命名スキームを使用します。

プロジェクトの **pom.xml** ファイルの **<build>** セクションで、バンドルのシンボリック名およびそのバンドル名 (OSGi) が **maven-bundle-plugin** エントリーに一覧表示されます。以下に例を示します。

```
<build>
  <defaultGoal>install</defaultGoal>
  <plugins>
    <plugin>
      <groupId>org.apache.felix</groupId>
      <artifactId>maven-bundle-plugin</artifactId>
      <version>${version.maven-bundle-plugin}</version>
      <extensions>true</extensions>
      <configuration>
        <instructions>
          <Bundle-SymbolicName>CBRroute</Bundle-SymbolicName>
          <Bundle-Name>Empty Camel Blueprint Example [CBRroute]</Bundle-Name></instructions>
      </configuration>
    </plugin>
```

付録A FUSE INTEGRATION パースペクティブ

Fuse Integration パースペクティブを使用して、インテグレーションアプリケーションの設計、監視、テスト、および公開を行います。

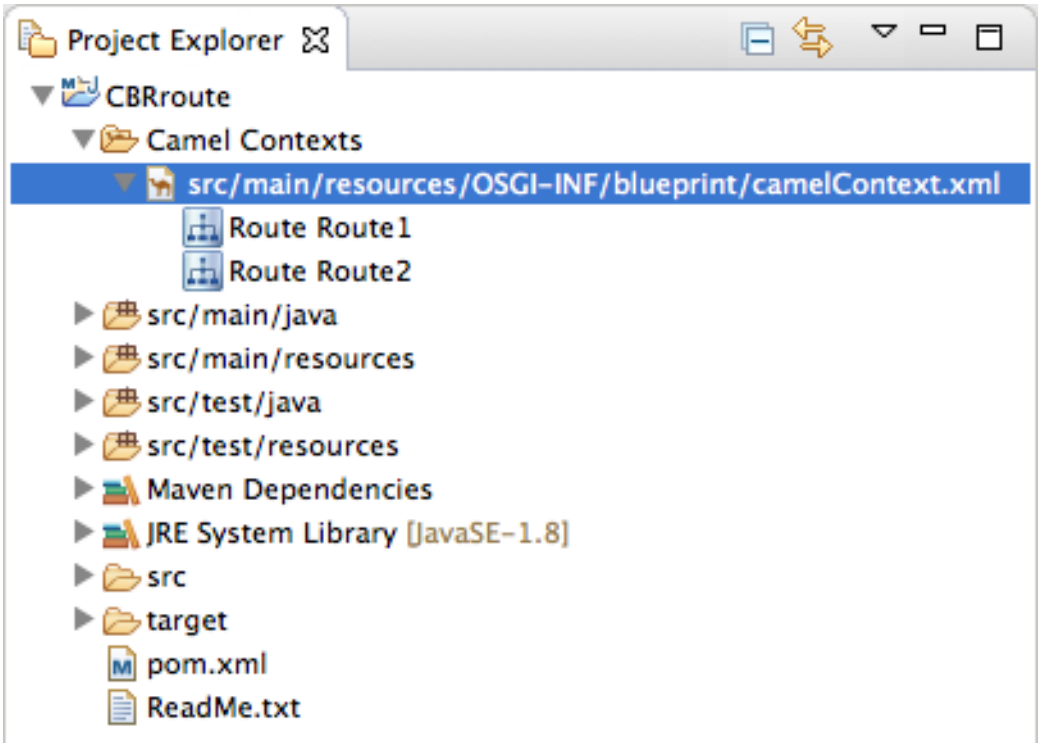


以下の方法で **Fuse Integration** パースペクティブを開くことができます。

- 新しい Fuse Integration プロジェクトを作成する場合 (*1章新しい Fuse Integration プロジェクトの作成* を参照)、ツールは **Fuse Integration** パースペクティブに切り替わります。
- ツールバーで  をクリックします。ツールバーで  アイコンが利用できない場合は、  をクリックし、利用可能なパースペクティブの一覧から **Fuse Integration** を選択します。
- **Window → Perspective → Open Perspective → Fuse Integration** の順に選択します。

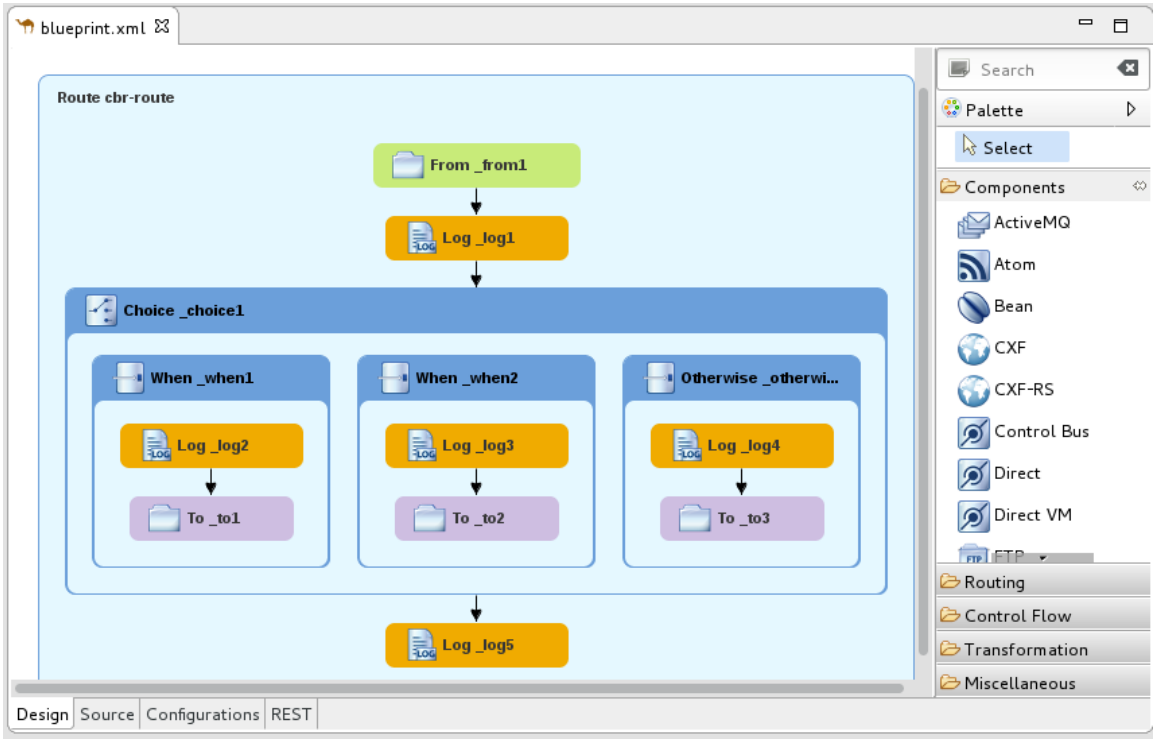
Fuse Integration パースペクティブは、９つのメインエリアで設定されます。

- **Project Explorer** ビュー
  ツールが認識するすべてのプロジェクトを表示します。各プロジェクトを設定するすべてのアーティファクトを表示できます。**Project Explorer** ビューでは、**Camel Contexts** ノードにプロジェクトのすべてのルーティングコンテキストの **.xml** ファイルも表示されます。これにより、プロジェクトに含まれるルーティングコンテキストファイルを見つけて開くことができます。**Project Explorer** ビューの各ルーティングコンテキストの **.xml** ファイルに、コンテキスト内で定義されたすべてのルートが表示されます。マルチルートコンテキストの場合、キャンバスで特定のルートにフォーカスできます。



- ルートエディター
  主な設計時ツールを提供し、以下の 3 つのタブで設定されます。

  - **Design**: ルートを構築する大きなグリッドエリアと、エンタープライズ統合パターン (EIP) および Camel コンポーネントを選択してキャンバスに接続し、ルートを形成するためのパレットが表示されます。



    キャンバスは、ルートエディターのワークベンチであり、多くの作業を行う場所です。接続された EIP および Camel コンポーネント (キャンバスに配置されるとノードと呼ばれます) で設定される 1 つ以上のルートがグラフィカル表示されます。

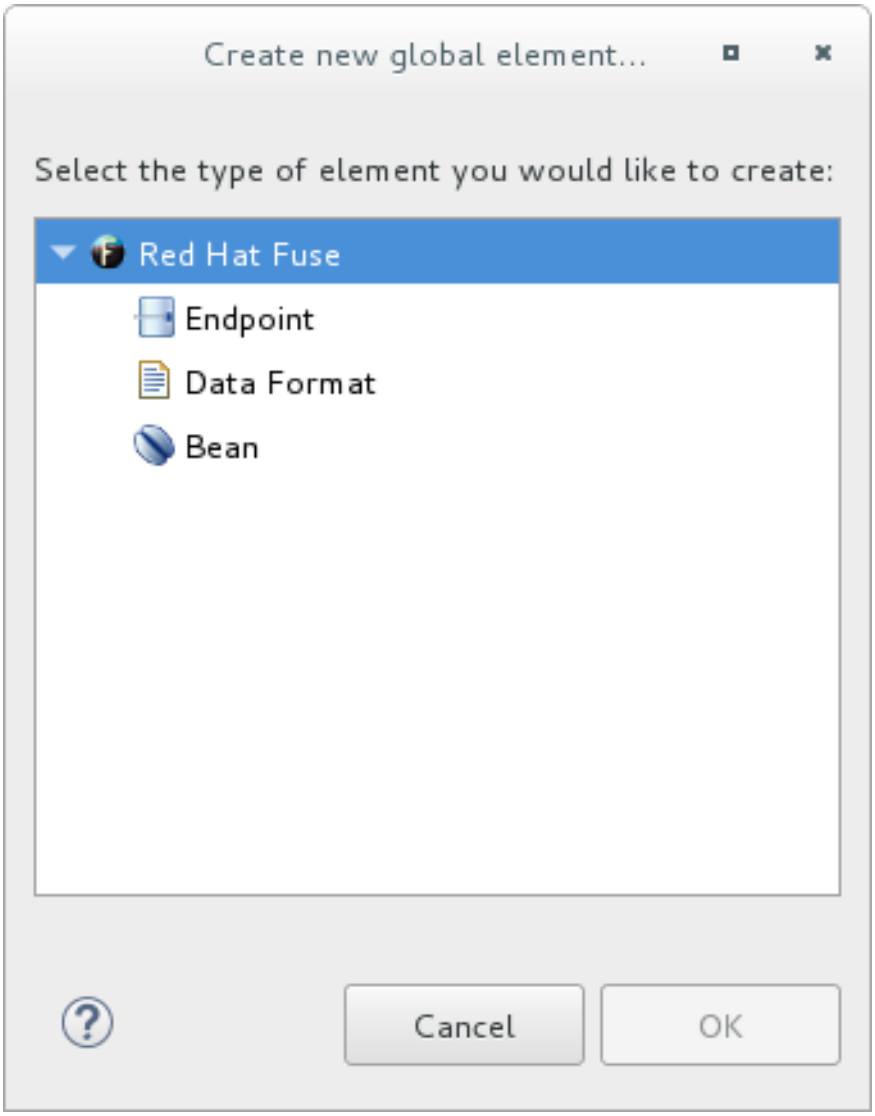
    キャンバスでノードを選択すると、**Properties** ビューに選択したノードに適用されるプロパティーが表示され、それを編集できるようになります。

Palette には、ルートを構築するために必要なすべてのパターンと Camel コンポーネントが含まれており、それらは機能 (**Components**、**Routing**、**Control Flow**、**Transformation**、**Miscellaneous**) に基づきグループ化されます。

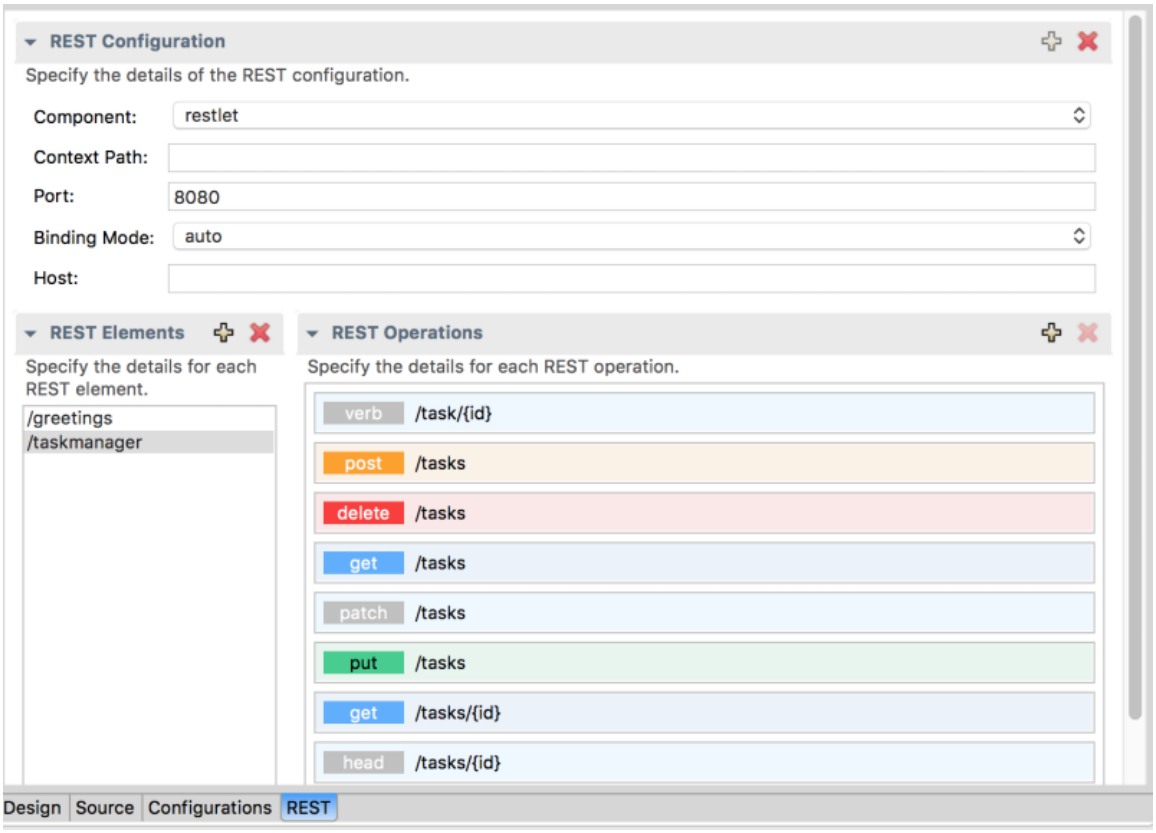
- **ソース**
  ルートエディターのキャンバスに構築されたルートの .xml ファイルの内容を表示します。

  Design タブと **Source** タブで、ルーティングコンテキストを編集できます。**Source** タブは、設定、コメント、または Bean を編集したり、ルーティングコンテキストファイルに追加したりする場合に便利です。コンテンツアシスト機能は、設定ファイルの使用に役立ちます。**Source** タブで **Ctrl**+**Space** を押すと、プロジェクトに挿入できる値の一覧が表示されます。

- **設定**
  共有設定 (グローバルエンドポイント、データ形式、Bean) を簡単にマルチルートルーティングコンテキストに追加する方法を提供します。詳細は、「グローバルエンドポイント、データ形式、Bean の追加」 を参照してください。



- **REST**
  Rest DSL コンポーネントのグラフィック表現を提供します。



- Properties ビュー
  キャンバスで選択したノードのプロパティーが表示されます。

- JMX Navigator ビュー
  JMX サーバーと監視するインフラストラクチャーの一覧が表示されます。これにより、JMX サーバーとそれらが監視しているプロセスを参照できます。また、Red Hat プロセスのインスタンスも特定できます。

  JMX Navigator ビューは、**Fuse Integration** パースペクティブのすべての監視およびテストアクティビティーを促進します。**Diagram View**、**Properties** ビュー、および **Messages View** に表示されるルートを決定します。ルートトレースの有効化、JMS 宛先の追加および削除、ルートの開始および一時停止を行うメニューコマンドも提供します。また、メッセージをルートにドラッグアンドドロップするためのターゲットでもあります。

  デフォルトでは、**JMX Navigator** ビューには、ローカルマシンで実行しているすべての Java プロセスが表示されます。必要に応じて JMX サーバーを追加して、他のマシンのインフラストラクチャーを表示できます。

- Diagram View
  JMX Navigator ビューで選択したノードを表すグラフィカルツリーが表示されます。プロセス、サーバー、エンドポイント、または他のノードを選択すると、**Diagram View** には、選択したノードがルートとして表示され、その子と孫に分岐します。

  ブローカーを選択すると、**Diagram View** には接続、トピック、およびキューの最大 3 つの子が表示されます。また、設定された接続と宛先が孫として表示されます。

  ルートを選択すると、**Diagram View** にルート内のすべてのノードが表示され、メッセージがルートを通過できるさまざまなパスが表示されます。また、ルートトレースが有効になっている場合、ルートの各処理ステップのタイミングメトリクスも表示されます。

- Messages View

ルートトレースが有効になっている場合に、選択した JMS 宛先または Apache Camel エンドポイントを通過したメッセージが一覧表示されます。

JMX Navigator ビューで JMS 宛先を選択すると、ビューにはその宛先にあるすべてのメッセージが一覧表示されます。

ルートトレースを有効にすると、**Messages View** にはトレース開始後にルートのノードを通過したすべてのメッセージが一覧表示されます。**Messages View** を設定して、対象のデータのみを好みの順序で表示できます。

Messages View のメッセージトレースを選択すると、その詳細 (メッセージの本文とすべてのメッセージヘッダー) が **Properties** ビューに表示されます。**Diagram View** では、選択したメッセージトレースに関連付けられたルートのステップが強調表示されます。

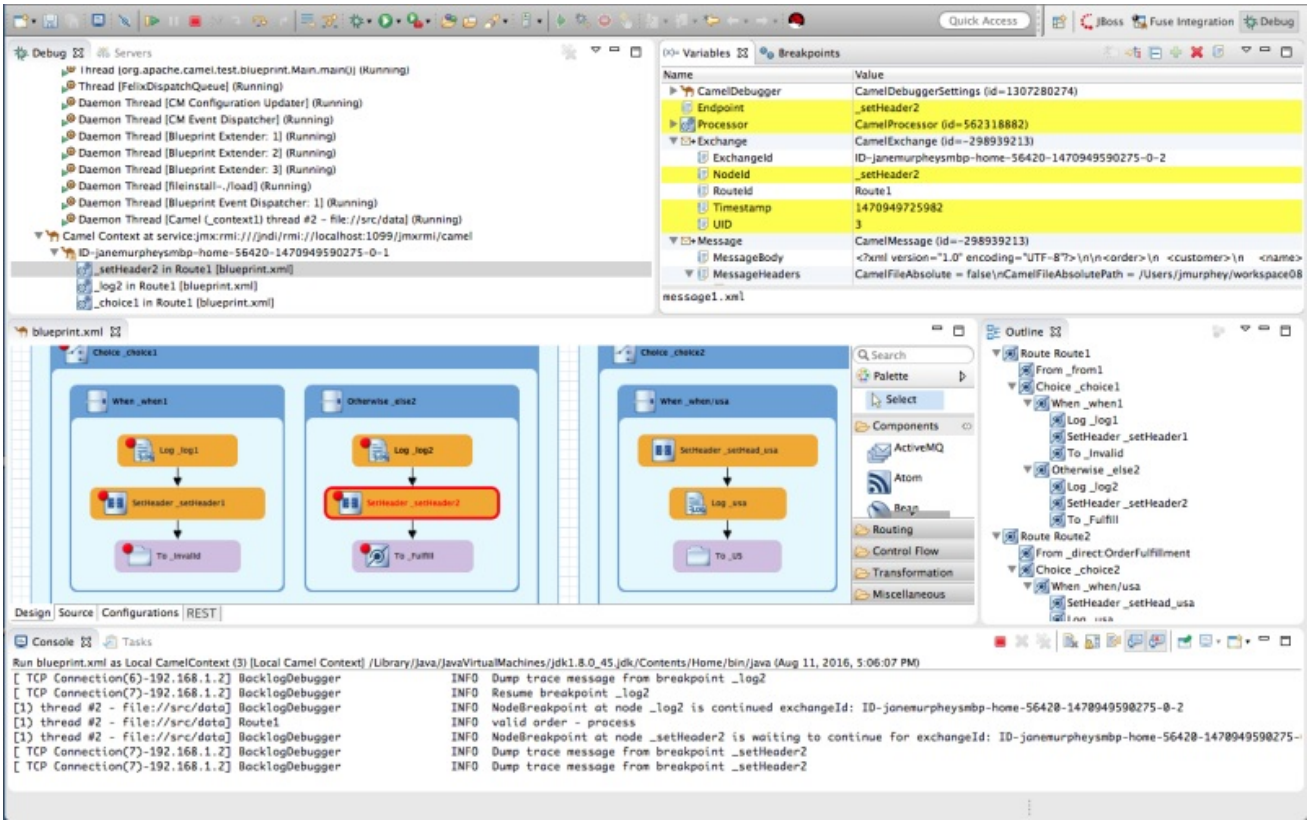
- **Servers** ビュー
  ツールが管理するサーバーの一覧が表示されます。ランタイムステータスが表示され、それらの追加、開始、停止、およびそれらへのプロジェクトの公開を制御できます。

- **Terminal** ビュー
  接続されているコンテナーのコマンドコンソールが表示されます。**Terminal** ビューでコマンドを入力してコンテナーを制御できます。

- **Console** ビュー
  最近実行したアクションのコンソール出力が表示されます。

付録B DEBUG パースペクティブ

Debug パースペクティブを使用して、実行中の Camel コンテキストを監視し、デバッグします。



- Debug ビュー
  実行中の Camel コンテキストの場合、**Debug** ビューにデバッグスタックが表示されます。

  Camel Context at service:jmx:rmi://jndi/rmi://localhost:1099/jmxrmi/camel エントリーの下に一覧表示される同じメッセージフロー内のブレークポイントを切り替えて、**Variables** ビュー内の変数の値を確認して比較できます。

  メッセージフローは一意のブレッドクラム ID で識別され、後続の各メッセージフローのブレッドクラム ID は 2 ずつ増えます。たとえば、最初のメッセージフローのブレッドクラム ID が **ID-janemurpheysmbp-home-54620-1470949590275-0-1** の場合、2 番目のメッセージフローのブレッドクラム ID は **ID-janemurpheysmbp-home-54620-1470949590275-0-3** になります。

- Variables ビュー
  ブレークポイントが設定されているルーティングコンテキスト内の各ノードでは、ブレークポイントに到達すると **Variables** ビューに利用可能な変数の値が表示されます。前のブレークポイント以降に値が変更された各変数は、黄色で強調表示されます。

  編集可能な変数の値を変更して、このような変更が想定される結果を生成するか確認して、ルーティングコンテキストの堅牢性をテストすることができます。

  ウォッチリストに変数を追加することもできるため、メッセージフローの想定されるポイントでそれらの値が想定どおりに変化するかどうかをすばやく簡単に確認できます。

- Breakpoints ビュー
  ルーティングコンテキストに設定されたブレークポイントの一覧が、有効/無効を含めて表示されます。個々のブレークポイントを有効化または無効化するには、それらのブレークポイントのチェックをオン (有効化) またはオフ (disabling) にします。これにより、問題のある動作をするルーティングコンテキストのノードに一時的にフォーカスすることができます。

 ボタンは無効化されたブレークポイントをスキップし、ルーティングコンテキスト内の次のアクティブなブレークポイントに移動します。一方、  ボタンは、ブレークポイントに関係なく、ルーティングコンテキスト内の次の実行ノードに移動します。

- **Camel Context.xml** ビュー
  実行中のルーティングコンテキストファイルをグラフィカルモードで表示します。ブレークポイントが設定されているノードの場合、設定されているブレークポイントのタイプと、ブレークポイントが有効か無効かが表示されます。ブレークポイントに到達すると、キャンバス上の対応するノードが赤で囲まれます。

  ノードの設定を確認するには、**Properties** ビューを開き、**camel Context.xml** のキャンバスのノードを選択します。

- **Console** ビュー
  Camel デバッガーがルーティングコンテキストを実行するときに生成されるログ出力を表示します。

- **Properties** ビュー
  CamelContext.xml のキャンバスで選択したノードに設定されたプロパティーを表示します。