Exponiendo Una Ruta Camel Blueprint Como Servicio Web en Red Hat JBoss Fuse

Autor: Carlos Carreño, ccarrenovi@gmail.com
Dic, 2016

Resumen

En este tutorial mostraremos, como exponer una ruta Camel creada con el estándar OSGi blueprint como un Servicio Web con contrato WSDL. Para completar el procedimiento es necesario conocer la programación con Camel por tanto recomendamos tomar el curso JB421 Desarrollo con Camel en Red Hat JBoss Fuse.

Procedimiento

Creación del Proyecto con Maven

1. Crea el proyecto maven desde el arquetipo camel-archetype-blueprint

   $mkdir myprojects

   $cd myprojects

   $mvn archetype:generate -DarchetypeGroupId=org.apache.camel.archetypes \
   > -DarchetypeArtifactId=camel-archetype-blueprint -DgroupId=com.fuse.demo \
   > -DartifactId=FuseWebService -DarchetypeVersion=2.12.0 -DinteractiveMode=false

2. Importa el proyecto en Red Hat JBoss Developer Studio, como se indica en las siguientes figuras.

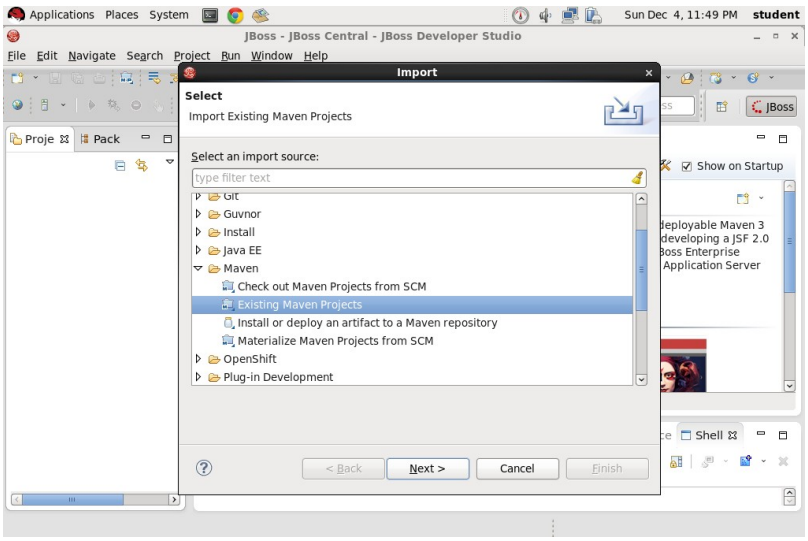

Figura 1

Selecciona “Existing Maven Projects” navega y selecciona la carpeta del proyecto FuseWebService creado en el punto 1, Haz clic en “Next”.

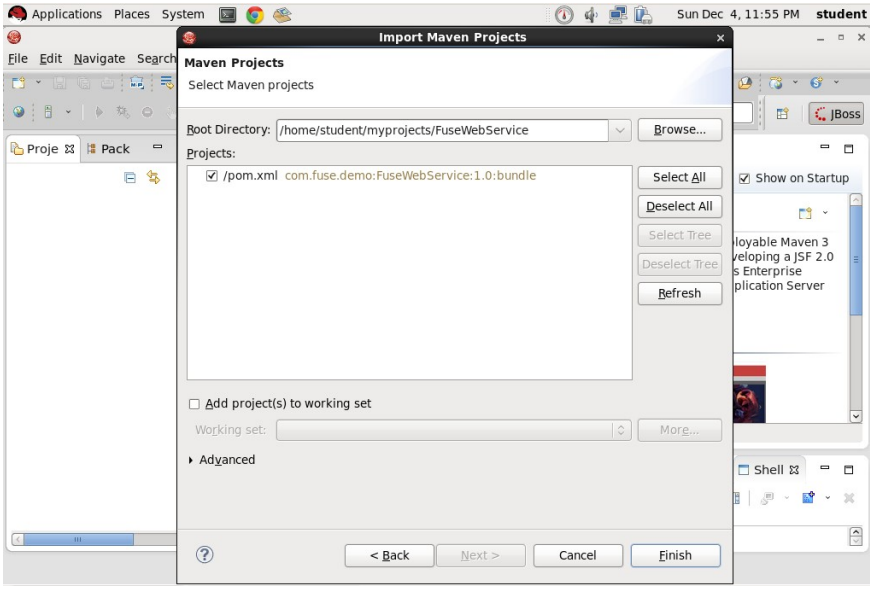

Figura 2

Asegurate de selección el checkbox del archivo pom.xml como se aprecia en la figura 2. Haz clic en “Finish”.

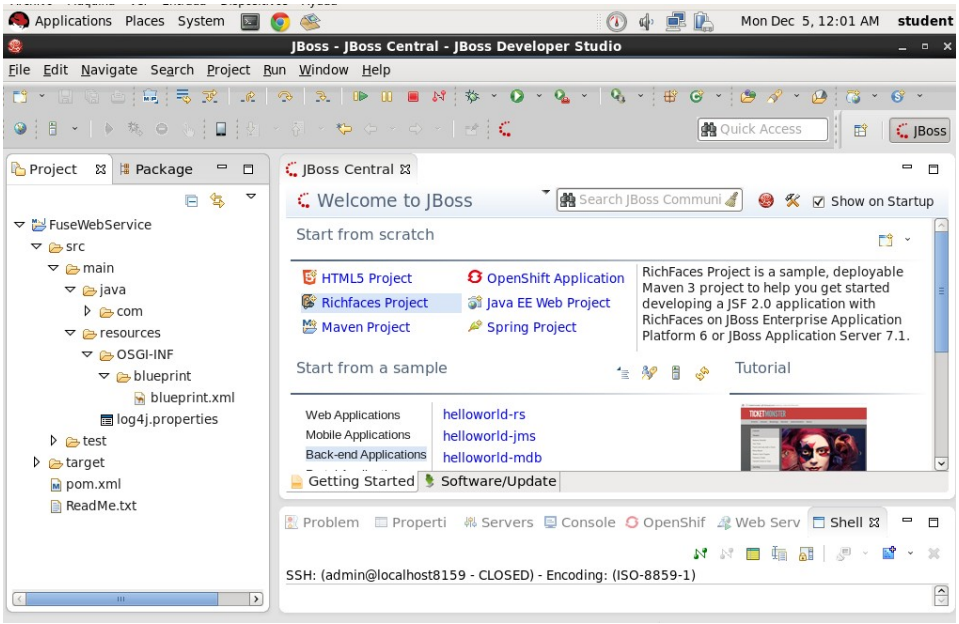

Figura 3

Dependencias Maven del Proyecto

3. Edita el archivo pom.xml para que contenga las dependencias necesarias para usa Apache cxf y utilizar camel:run

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">

 <modelVersion>4.0.0</modelVersion>

 <groupId>com.fuse.demo</groupId>
 <artifactId>FuseWebService</artifactId>
 <version>1.0</version>

 <packaging>bundle</packaging>

 <name>Ruta Camel Como Simple Web Service</name>
 <url>http://www.bamtech.com.pe</url>

 <properties>
  <camel-version>2.17.0</camel-version>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
 </properties>

 <dependencies>
  <dependency>
   <groupId>org.apache.camel</groupId>
   <artifactId>camel-core</artifactId>
   <version>${camel-version}</version>
  </dependency>
  <dependency>
   <groupId>org.apache.camel</groupId>
   <artifactId>camel-blueprint</artifactId>
   <version>${camel-version}</version>
  </dependency>

  <!-- logging -->
  <dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>slf4j-api</artifactId>
   <version>1.7.5</version>
  </dependency>
  <dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>slf4j-log4j12</artifactId>
   <version>1.7.5</version>
  </dependency>
```

```
 <dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>jcl-over-slf4j</artifactId>
  <version>1.7.5</version>
 </dependency>
 <dependency>
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  <version>1.2.17</version>
 </dependency>

 <!-- testing -->
 <dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-test-blueprint</artifactId>
   <version>${camel-version}</version>
  <scope>test</scope>
 </dependency>

 <!-- Camel CXF -->
 <dependency>
       <groupId>org.apache.camel</groupId>
       <artifactId>camel-cxf</artifactId>
       <version>${camel-version}</version>
 </dependency>
<dependency>
          <groupId>org.apache.cxf</groupId>
          <artifactId>cxf-rt-transports-http-jetty</artifactId>
          <version>3.1.8</version>
 </dependency>

 <!-- need for the cmd camel:run on blueprint camel route -->
 <dependency>
    <groupId>org.ow2.asm</groupId>
    <artifactId>asm-commons</artifactId>
    <version>5.0.3</version>
    <scope>runtime</scope>
 </dependency>
 <dependency>
    <groupId>org.osgi</groupId>
    <artifactId>org.osgi.core</artifactId>
    <version>5.0.0</version>
    <scope>runtime</scope>
 </dependency>
 <dependency>
    <groupId>org.osgi</groupId>
    <artifactId>org.osgi.compendium</artifactId>
    <version>5.0.0</version>
    <scope>runtime</scope>
 </dependency>

</dependencies>
```

```
  <build>
    <defaultGoal>install</defaultGoal>

    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>2.5.1</version>
        <configuration>
          <source>1.6</source>
          <target>1.6</target>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-resources-plugin</artifactId>
        <version>2.6</version>
        <configuration>
          <encoding>UTF-8</encoding>
        </configuration>
      </plugin>

      <!-- to generate the MANIFEST-FILE of the bundle -->
      <plugin>
        <groupId>org.apache.felix</groupId>
        <artifactId>maven-bundle-plugin</artifactId>
        <version>2.3.7</version>
        <extensions>true</extensions>
        <configuration>
          <instructions>
            <Bundle-SymbolicName>FuseWebService</Bundle-SymbolicName>
            <Private-Package>com.fuse.demo.*</Private-Package>
            <Import-Package>*</Import-Package>
          </instructions>
        </configuration>
      </plugin>

       <!-- to run the example using mvn camel:run -->
      <plugin>
        <groupId>org.apache.camel</groupId>
        <artifactId>camel-maven-plugin</artifactId>
        <version>${camel-version}</version>
        <configuration>
          <useBlueprint>true</useBlueprint>
        </configuration>
      </plugin>
    </plugins>
  </build>

</project>
```

Creación del Bean del Servicio

4. Crea las clases Product, ProductService y ProductServiceImp en el paquete com.fuse.demo del proyecto.

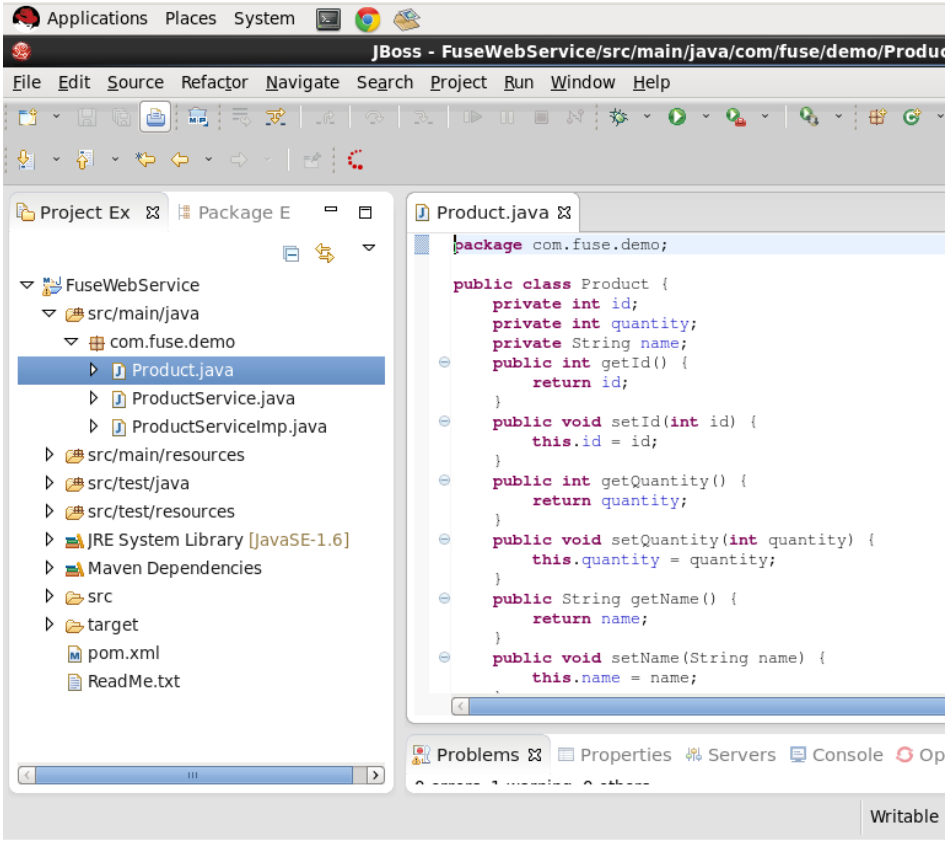

Figura 4

Product.java

```
package com.fuse.demo;

public class Product {
   private int id;
   private int quantity;
   private String name;
        public int getId() {
                return id;
        }
        public void setId(int id) {
                this.id = id;
        }
        public int getQuantity() {
                return quantity;
        }
        public void setQuantity(int quantity) {
                this.quantity = quantity;
```

```
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public Product(int id, int quantity, String name) {
		super();
		this.id = id;
		this.quantity = quantity;
		this.name = name;
	}

	// Importante!
	// Es necesario para desplegar en Fuse es obligatorio un constructor sin argumentos
	public Product() {

	}



}
```

ProductService.java

```
package com.fuse.demo;

public interface ProductService {
	public Product findProduct(String id);
   public String deleteProduct(String id);
}
```

ProductServiceImp.java

```
package com.fuse.demo;

import java.util.HashMap;
import java.util.Map;

public class ProductServiceImp implements ProductService {

	Map<String ,Product> store = new HashMap<String, Product>();

	public ProductServiceImp(){
		store.put("1", new Product(1,10,"Chocolate"));
		store.put("2", new Product(1,12,"Coca Cola"));
	}

	@Override
	public Product findProduct(String id) {
		return store.get(id);
```

```
    }

    @Override
    public String deleteProduct(String id) {
            Object obj = store.get(id);
   if (obj == null) {
    return "not Found!";
   } else {
    store.remove(id);
    return "Deleted!";
   }
    }

}
```

Creación del Endpoint y Las Rutas

5. Edita el archivo blueprint.xml y define el Endpoint y las rutas camel.

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



  <cxf:cxfEndpoint id="productEndpoint"
            address="http://localhost:9001/productService"
        serviceClass="com.fuse.demo.ProductService" />

  <bean id="beanProcessor" class="com.fuse.demo.ProductServiceImp" />

  <camelContext trace="false" id="blueprintContext"
xmlns="http://camel.apache.org/schema/blueprint">
    <route id="cxfRoute">
      <from uri="cxf:bean:productEndpoint"/>
      <log message="${header.operationName}"/>
      <recipientList>
         <simple>direct:${header.operationName}</simple>
      </recipientList>
    </route>
    <route id="findProductRoute">
      <from uri="direct:findProduct"/>
      <bean method="findProduct" ref="beanProcessor"/>
    </route>
    <route id="deleteProductRoute">
      <from uri="direct:deleteProduct"/>
      <bean method="deleteProduct" ref="beanProcessor"/>
    </route>
  </camelContext>

</blueprint>
```

6. Compila el proyecto e instala el bundle en el repositorio local.

```
[student@instructor FuseWebService]$ pwd
/home/student/myprojects/FuseWebService
```

[student@instructor FuseWebService]$ mvn clean package -Dmaven.test.skip=true install

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Ruta Camel Como Simple Web Service 1.0
[INFO] ------------------------------------------------------------------------
[INFO] …
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 9.689s
[INFO] Finished at: Mon Dec 05 00:41:49 PET 2016
[INFO] Final Memory: 23M/57M
[INFO] ------------------------------------------------------------------------
[student@instructor FuseWebService]$
```

Despliegue de la Aplicación en JBoss Fuse

7. Desde la consola de JBoss Fuse instala la aplicación

Figura 5

Ejecuta los siguientes comandos:

$osgi:install mvn:com.fuse.demo/FuseWebService/1.0

Bundle ID: 255

$osgi:list

[246] [Active] [ ] [ ] [60] hawtio :: hawtio-web (1.2.0.redhat-379)
[247] [Active] [Created] [ ] [ ] [60] hawtio :: hawtio-json-schema-mbean (1.2.0.redhat-379)
[248] [Active] [ ] [ ] [ ] [60] JLine (2.11.0)
[249] [Active] [Created] [ ] [ ] [60] hawtio :: Karaf terminal plugin (1.2.0.redhat-379)
[250] [Active] [Created] [ ] [ ] [60] hawtio :: hawtio-maven-indexer (1.2.0.redhat-379)
[255] [Installed] [ ] [ ] [ ] [60] Ruta Camel Como Simple Web Service (1.0.0)

$osgi:start 255

$osgi:list

[248] [Active] [ ] [ ] [ ] [60] JLine (2.11.0)
[249] [Active] [Created] [ ] [ ] [60] hawtio :: Karaf terminal plugin (1.2.0.redhat-379)
[250] [Active] [Created] [ ] [ ] [60] hawtio :: hawtio-maven-indexer (1.2.0.redhat-379)
[255] [Active] [Created] [ ] [ ] [60] Ruta Camel Como Simple Web Service (1.0.0)

Nota: El numero 255 es el numero de bundle asignado a la aplicación instalada

Test del Servicio Web

8. Abre la vista “Web Service Tester” en JBoss Developer Studio

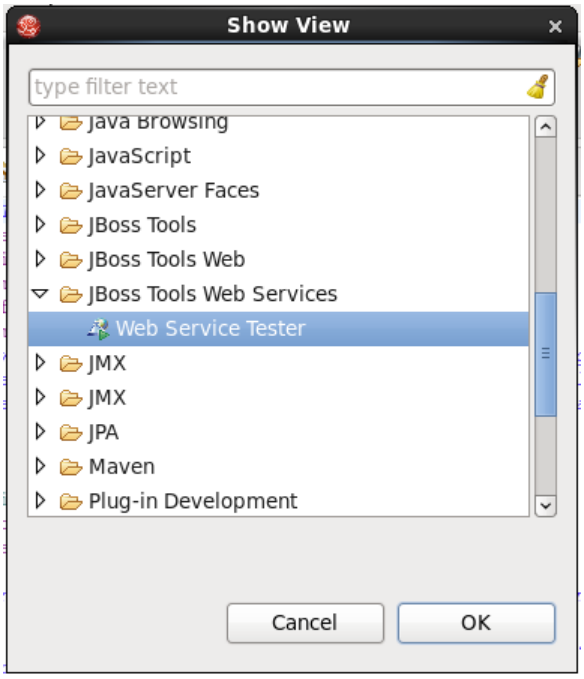

Figura 6

9. En “Web Service Tester” ingresa el uri del endpoint del Servicio Web, selecciona JAX-WS y luego el metodo a probar según indica la figura 7.

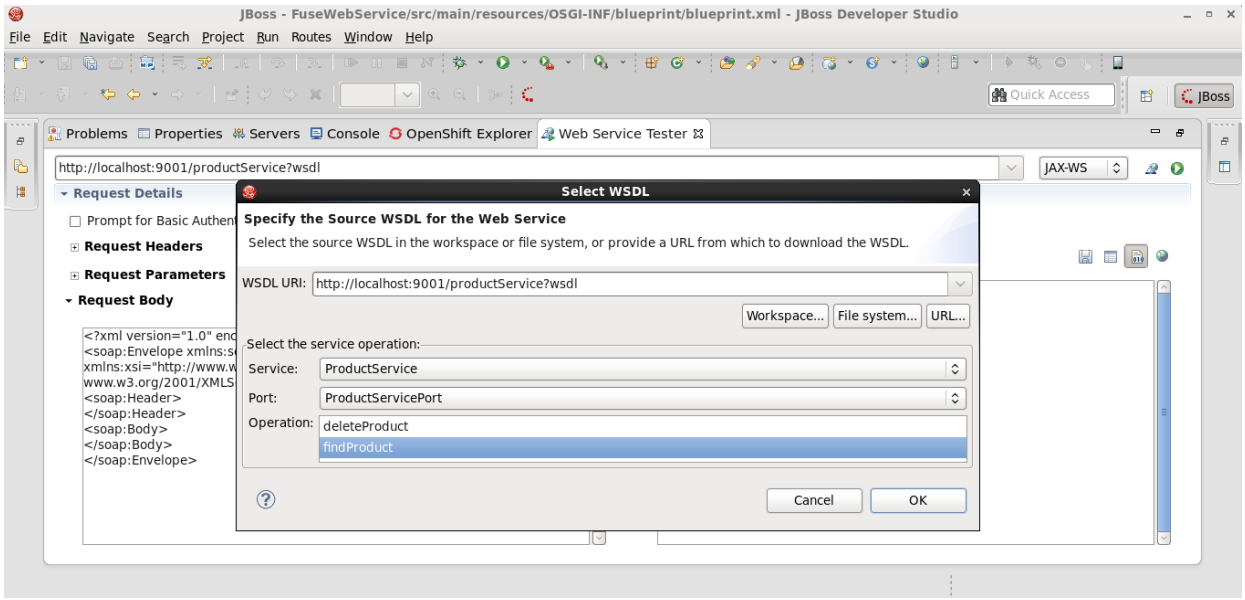

Figura 7

En Request Details, ingresa el valor de 1 el argumento del SOAP Message, asi <arg0>1</arg0>

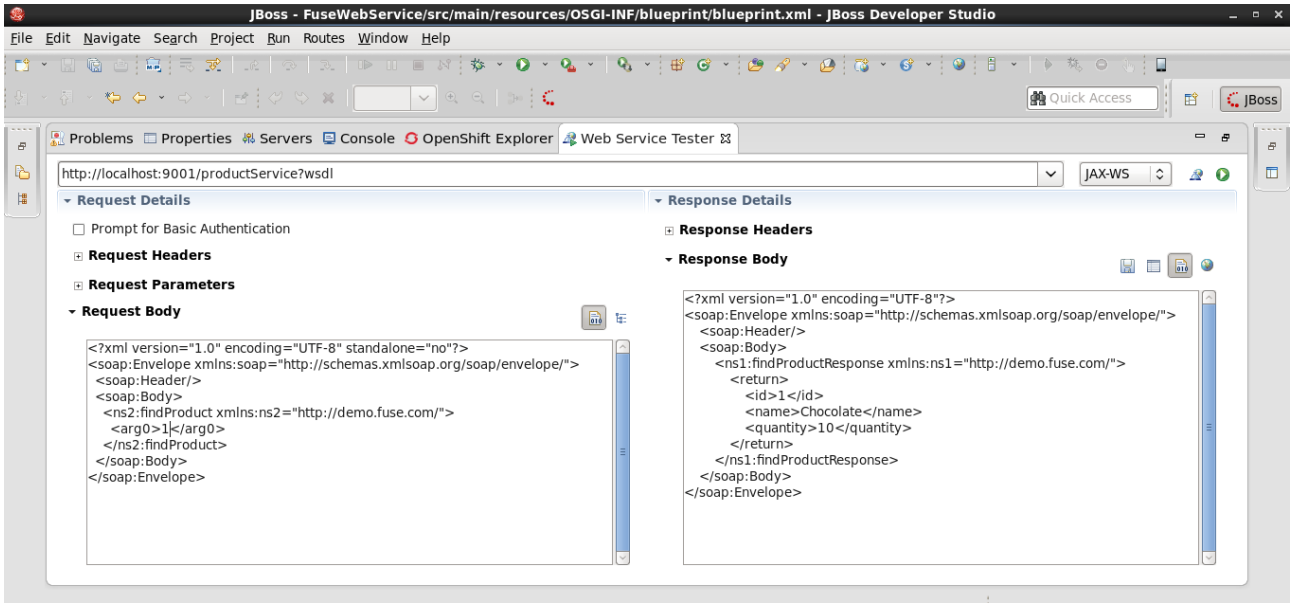
JBoss - FuseWebService/src/main/resources/OSGI-INF/blueprint/blueprint.xml - JBoss Developer Studio
File Edit Navigate Search Project Run Routes Window Help
Quick Access
JBoss
Problems Properties Servers Console OpenShift Explorer Web Service Tester
http://localhost:9001/productService?wsdl
JAX-WS
Request Details
Prompt for Basic Authentication
Request Headers
Request Parameters
Request Body
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Header/>
<soap:Body>
<ns2:findProduct xmlns:ns2="http://demo.fuse.com/">
<arg0>1</arg0>
</ns2:findProduct>
</soap:Body>
</soap:Envelope>
Response Details
Response Headers
Response Body
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Header/>
<soap:Body>
<ns1:findProductResponse xmlns:ns1="http://demo.fuse.com/">
<return>
<id>1</id>
<name>Chocolate</name>
<quantity>10</quantity>
</return>
</ns1:findProductResponse>
</soap:Body>
</soap:Envelope>

Figura 8

Monitoreo del Web Service con JBoss Fuse

10. Ingresa a la consola web de JBoss Fuse

    Abre en un brower la url http://localhost:8181/hawtio/

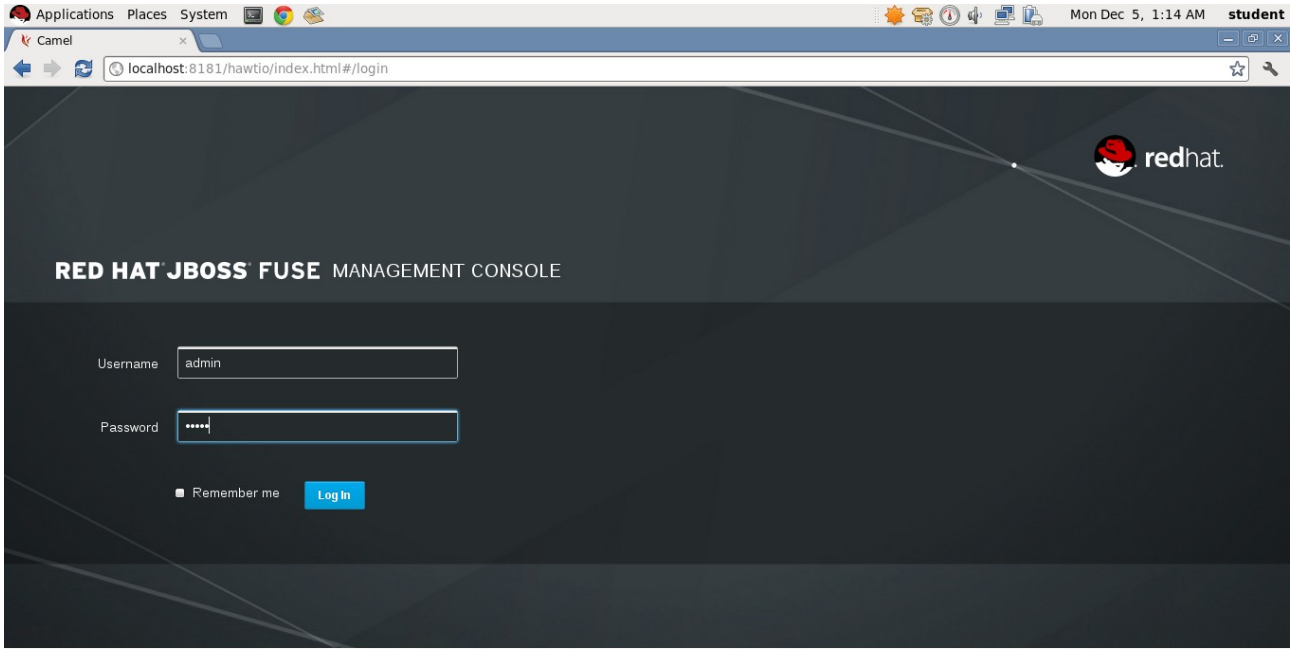

Figura 9

Ingresa con el usuario admin y clave admin. Haz clic en “login”.

11. Monitorea el procesamiento de las rutas

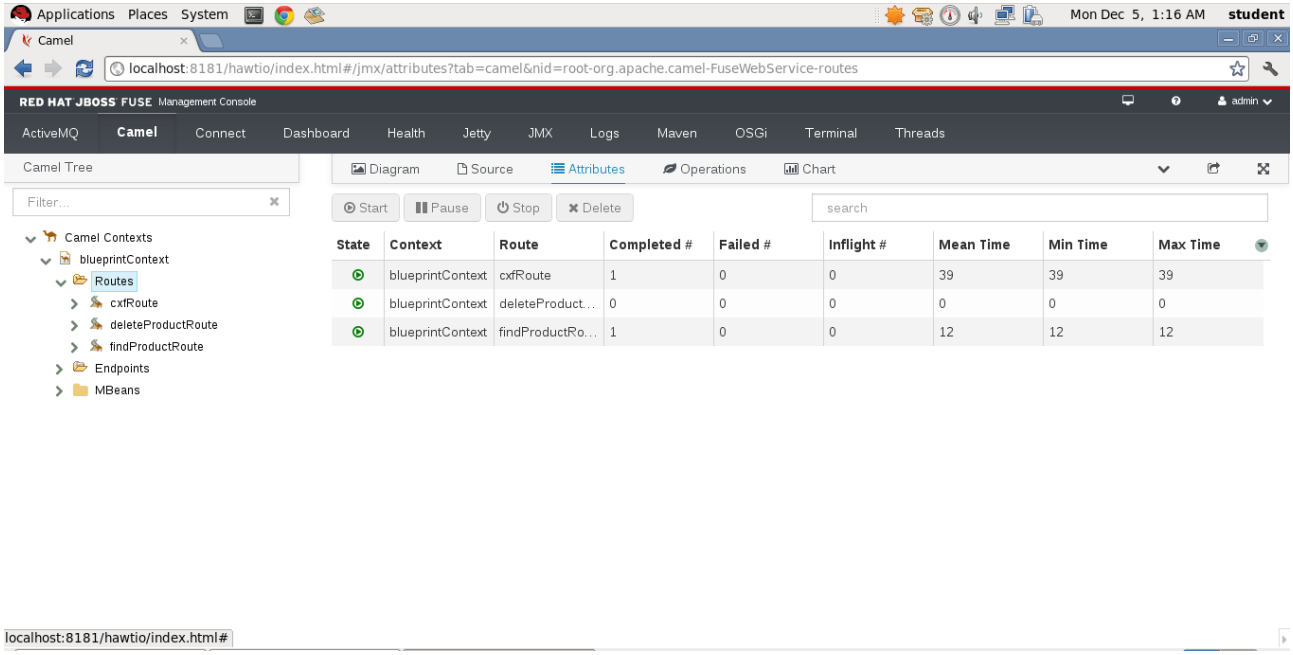

Figura 10

12. Muestra el diagrama de las rutas

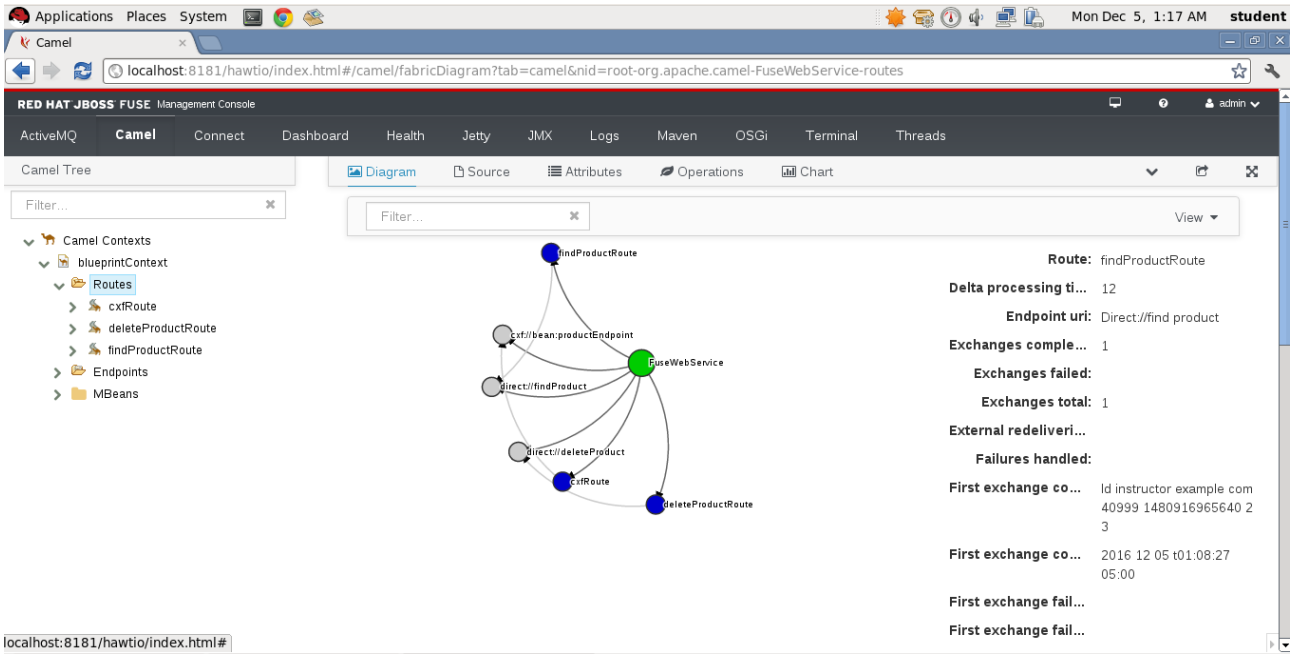

Figura 11

13. Diagrama de la ruta.

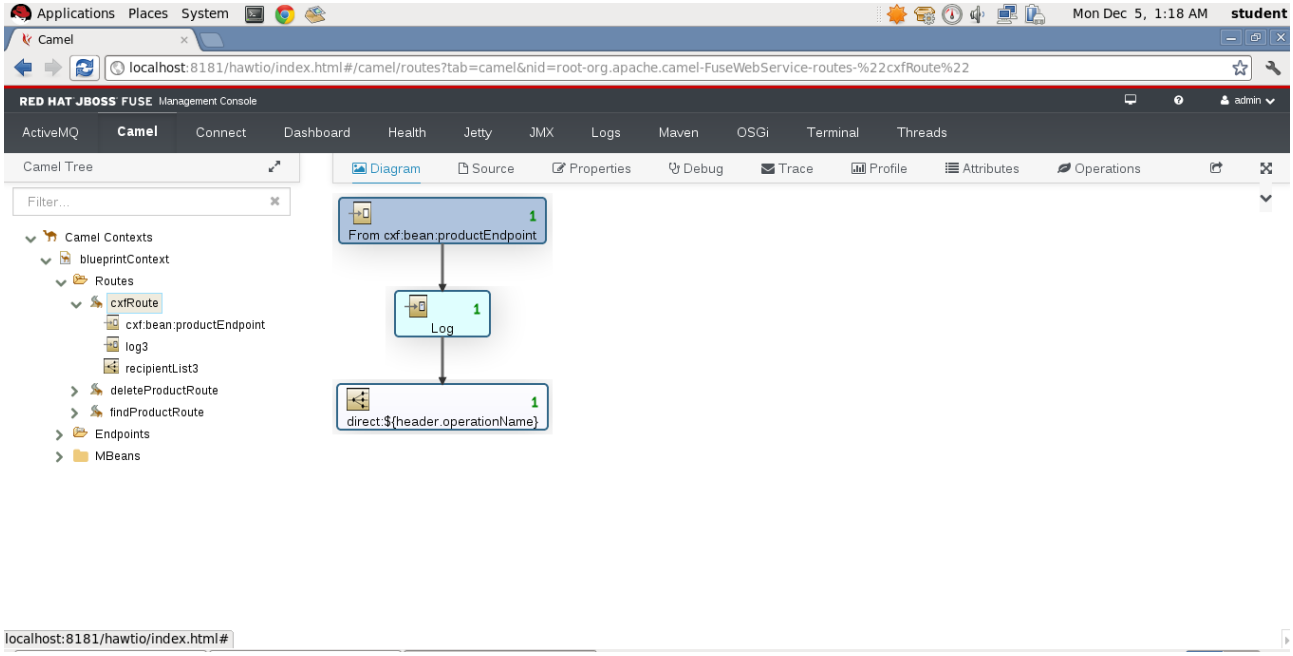

Figura 12

Felicitaciones!, haz deplegado correctamente la ruta camel como servicio.