Red Hat JBoss Fuse 6.0

Tutorials

Example integration applications

Last Updated: 2017-10-13

Red Hat JBoss Fuse 6.0 Tutorials

Example integration applications

JBoss A-MQ Docs Team
Content Services
fuse-docs-support@redhat.com

Legal Notice

Abstract

This guide contains examples of implementing several different types of applications in Red Hat JBoss Fuse.

Table of Contents

CHAPTER 1. DEPLOYING A WEB SERVICE INTO RED HAT JBOSS FUSE

Abstract

Here we will take an in depth look at the CXF OSGi example, a JAX-WS web service packaged as an OSGi bundle.

You are going to build and deploy a simple service that is based on a WSDL document. The source for this example can be found in the ***InstallDir*/examples/cxf-osgi** folder of your Red Hat JBoss Fuse installation.

The example uses the JBoss Fuse Maven tooling to build a bundle that contains the service implementation and all of the metadata needed to deploy it into the JBoss Fuse container.

The sample code includes a web page, **client.html**, that will allow you to access the service once it is exposed.

1.1. DEPLOYING THE EXAMPLE

Before you run the example, start Red Hat JBoss Fuse.

Installing a prebuilt version of the example

To install and run a prebuilt version of this example, enter the following command in the Red Hat JBoss Fuse console:

```
features:install examples-cxf-osgi
```

Building and deploying the example yourself

To build and deploy the example do the following:

1. If you installed the examples-cxf-osgi feature, first uninstall it by running the following from the Red Hat JBoss Fuse console:

   ```
   features:uninstall examples-cxf-osgi
   ```

2. In a separate command window or terminal, change to the root folder of the example.

   ```
   cd InstallDir/examples/cxf-osgi
   ```

3. Enter the following command:

   ```
   mvn install
   ```

   This command will build a bundle called **cxf-osgi-6.0.0.redhat-024.jar** and place it into the **target** folder of the example.

4. Copy the bundle to ***InstallDir*/deploy** to deploy it to the container.

Testing the example

To see if the example is running you can visit http://localhost:8181/cxf/HelloWorld?wsdl in your Web browser. You should see the WSDL shown in Example 1.1, “CXF OSGi Example WSDL”.

Example 1.1. CXF OSGi Example WSDL

```
<wsdl:definitions name="HelloWorldImplService"

targetNamespace="http://cxf.examples.servicemix.apache.org/">
  <wsdl:types>
    <xs:schema attributeFormDefault="unqualified"
               elementFormDefault="unqualified"

targetNamespace="http://cxf.examples.servicemix.apache.org/">
      <xs:complexType name="sayHi">
        <xs:sequence>
          <xs:element minOccurs="0" name="arg0" type="xs:string"/>
        </xs:sequence>
       </xs:complexType>
       <xs:complexType name="sayHiResponse">
         <xs:sequence>
           <xs:element minOccurs="0" name="return" type="xs:string"/>
         </xs:sequence>
       </xs:complexType>
       <xs:element name="sayHi" nillable="true" type="sayHi"/>
       <xs:element name="sayHiResponse" nillable="true"
type="sayHiResponse"/>
     </xs:schema>
  </wsdl:types>
  <wsdl:message name="sayHiResponse">
    <wsdl:part element="tns:sayHiResponse" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="sayHi">
    <wsdl:part element="tns:sayHi" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="HelloWorld">
    <wsdl:operation name="sayHi">
      <wsdl:input message="tns:sayHi" name="sayHi">
      </wsdl:input>
      <wsdl:output message="tns:sayHiResponse" name="sayHiResponse">
       </wsdl:output>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="HelloWorldImplServiceSoapBinding"
type="tns:HelloWorld">
    <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="sayHi">
        <soap:operation soapAction="" style="document"/>
        <wsdl:input name="sayHi">
          <soap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="sayHiResponse">
          <soap:body use="literal"/>
        </wsdl:output>
```

```
      </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="HelloWorldImplService">
    <wsdl:port binding="tns:HelloWorldImplServiceSoapBinding"
name="HelloWorldImplPort">
      <soap:address location="http://localhost:8181/cxf/HelloWorld"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

1.2. RUNNING A CLIENT

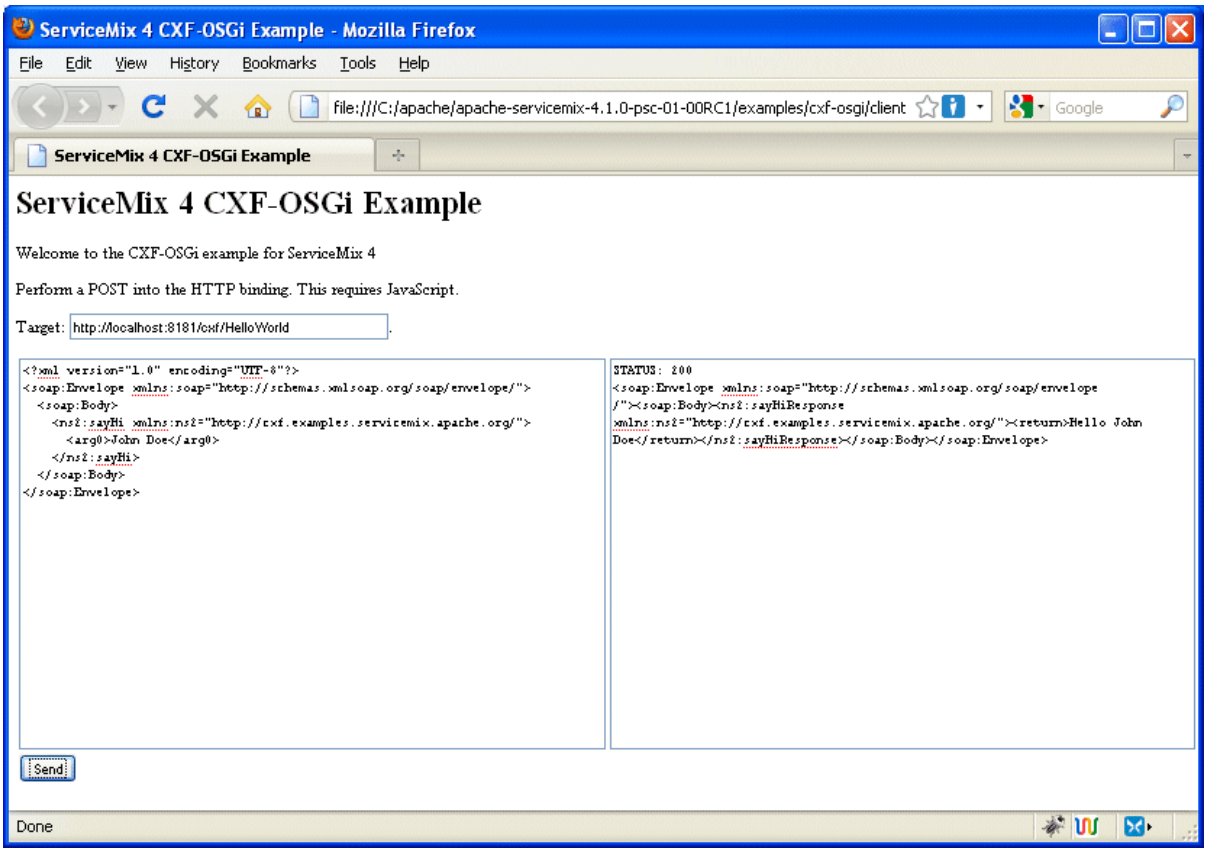
You can run either a web client or a Java client against the deployed web service.

Running the web client

To run the web client:

1. Open the **`client.html`**, which is located at the root of the **`cxf-osgi`** example folder.
2. Click the **Send** button to send a request.

Figure 1.1. Example Client HTML Screen



Once the request has been successfully sent, a response message appears in the right-hand panel of the web page

Running the Java client

To run the Java client:

1. In a command prompt or terminal, change to the ***InstallDir*/examples/cxf-osgi directory**.

2. Run the following command:

   ```
   mvn compile exec:java
   ```

   If the client request is successful, a response similar to the following is returned to the command window:

   ```
   the response is =====> <soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><soap:Body>
   <ns2:sayHiResponse
   xmlns:ns2="http://cxf.examples.servicemix.apache.org/">
   <return>Hello John Doe</return></ns2:sayHiResponse></soap:Body>
   </soap:Envelope>
   ```

1.3. UNDERSTANDING THE EXAMPLE

Overview

The example builds a simple HelloWorld service and packages it for deployment into Red Hat JBoss Fuse. The service is written using standard JAX-WS APIs. It implements a single operation **sayHi()**. Once deployed, the service is exposed as a SOAP/HTTP endpoint. The most interesting parts of the example are the Spring configuration used to configure the endpoint and the Maven POM used to build the bundle.

The Spring configuration provides the details needed to expose the service using SOAP/HTTP. It can also contain details used to configure advanced Apache CXF functionality.

The Maven POM, in addition to compiling the code, uses the bundle generation plug-in to package the resulting classes into an OSGi bundle. It contains all of the details needed by the Red Hat JBoss Fuse container to activate the bundle and deploy the packaged service.

Using the Maven tools

The Red Hat JBoss Fuse Maven tooling automates a number of the steps in packaging functionality for deployment into JBoss Fuse. In order to use the Maven OSGi tooling, you add the elements shown in Example 1.2, “POM Elements for Using Red Hat JBoss Fuse OSGi Tooling” to your POM file.

Example 1.2. POM Elements for Using Red Hat JBoss Fuse OSGi Tooling

```
...
<pluginRepositories>
  <pluginRepository>
     <id>fusesource.m2</id>
      <name>Open Source Community Release Repository</name>
       <url>http://repo.fusesource.com/maven2</url>
        <snapshots>
           <enabled>false</enabled>
         </snapshots>
           <releases>
```

```
            <enabled>true</enabled>
          </releases>
  </pluginRepository>
</pluginRepositories>
 ...
<build>
   <plugins>
      <plugin>
        <groupId>org.apache.felix</groupId>
        <artifactId>maven-bundle-plugin</artifactId>
         ...
       </plugin>
    </plugins>
</build>
  ...
```

These elements point Maven to the correct repositories to download the Red Hat JBoss Fuse Maven tooling and load the plug-in that implements the OSGi tooling.

The Spring configuration

The Red Hat JBoss Fuse container needs some details about a service before it can instantiate and endpoint for it. Apache CXF uses Spring based configuration to define endpoints for services. The configuration shown in Example 1.3, “OSGi Example Spring Configuration” is stored in the example's **\src\main\resources\META-INF\spring\beans.xml** file.

Example 1.3. OSGi Example Spring Configuration

```
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:jaxws="http://cxf.apache.org/jaxws"
 xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd">

1    <import resource="classpath:META-INF/cxf/cxf.xml" />
     <import resource="classpath:META-INF/cxf/cxf-extension-soap.xml"
/>
    <import resource="classpath:META-INF/cxf/cxf-extension-http.xml" />
    <import resource="classpath:META-INF/cxf/osgi/cxf-extension-
osgi.xml" />

2 <jaxws:endpoint id="helloWorld"

implementor="org.apache.servicemix.examples.cxf.HelloWorldImpl"
                address="/HelloWorld"/>

</beans>
```

The configuration shown in Example 1.3, “OSGi Example Spring Configuration” does the following:

1. Imports the required configuration to load the required parts of the Apache CXF runtime.
2. Configures the endpoint that exposes the service using the **`jaxws:endpoint`** element and its attributes.
   - **`id`** is an identifier used by the configuration mechanism.
   - **`implementor`** specifies the class that implements the service. It must be on the classpath.
   - **`address`** specifies the address at which the service will be exposed. This address is relative to the containers HTTP address with **`cxf`** appended to it.

The POM

Example 1.4. OSGi POM

```
<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>

    <parent>
        <groupId>org.apache.servicemix.examples</groupId>
        <artifactId>examples</artifactId>
        <version>6.0.0.redhat-024</version>
    </parent>

    <groupId>org.apache.servicemix.examples</groupId>
    <artifactId>cxf-osgi</artifactId>
    <packaging>bundle</packaging>
    <version>6.0.0.redhat-024</version>
    <name>Apache ServiceMix Example :: CXF OSGi</name>

<!-- Add ServiceMix repositories for snaphots and releases -->
 ...

    <dependencies>
        <dependency>
            <groupId>org.apache.geronimo.specs</groupId>
            <artifactId>geronimo-ws-metadata_2.0_spec</artifactId>
            <version>${geronimo.wsmetadata.version}</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.felix</groupId>
                <artifactId>maven-bundle-plugin</artifactId>
                <configuration>
                    <instructions>
                        <Bundle-SymbolicName>${pom.artifactId}</Bundle-
```

```
SymbolicName>
                        <Import-Package>
                            javax.jws,
                            javax.wsdl,
                            META-INF.cxf,
                            META-INF.cxf.osgi,
                            org.apache.cxf.bus,
                            org.apache.cxf.bus.spring,
                            org.apache.cxf.bus.resource,
                            org.apache.cxf.configuration.spring,
                            org.apache.cxf.resource,

org.apache.servicemix.cxf.transport.http_osgi,
                            org.springframework.beans.factory.config
                        </Import-Package>
                        <Private-
Package>org.apache.servicemix.examples.cxf</Private-Package>
                        <Require-Bundle>org.apache.cxf.cxf-
bundle</Require-Bundle>
                    </instructions>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

INDEX

M