5 Apache CXF and Tomcat Server

Objectives

After completing this chapter, you should be able to:

1. Develop a Java Web Application
2. Package a Web Application
3. Deploy a Web Application to Apache Tomcat 7 server

Deploying a Web Service using JDK alone is not adequate for large and complex systems. In this chapter, we will learn to develop and deploy Web Services using Apache CXF Web Services and Apache Tomcat server 7. Tomcat is an open source software program that implements Java Servlet and Java Server Page (JSP) technologies.

Apache CXF can be downloaded from Apache's website http://cxf.apache.org. Follow the download and installation instructions for your machine. The CXF WS project includes many of its Java libraries. The list of required libraries is shown in this section.

5.1 Configuration Parameters

The default port used for the Tomcat server is 8080. We keep this default for the Tomcat server that hosts the sample application we developed in the previous chapter. In fact, we use all of the default values that come with the Tomcat server.

5.2 Apache Tomcat Server

To run a Tomcat 7 server, go to the apache-tomcat-7.0.12/bin directory and run startup.cmd or startup.sh.

5.3 Develop CXF Web Service

Developing a Web Service using the Apache CXF software package is similar to that of the reference implementation that comes with the JDK.

5.3.1 Class Diagram

Consider the following updated class diagram. In this diagram, we added two classes – EmployeeDataIf. java and EmployeeData.java. The latter implements the interface of the former.

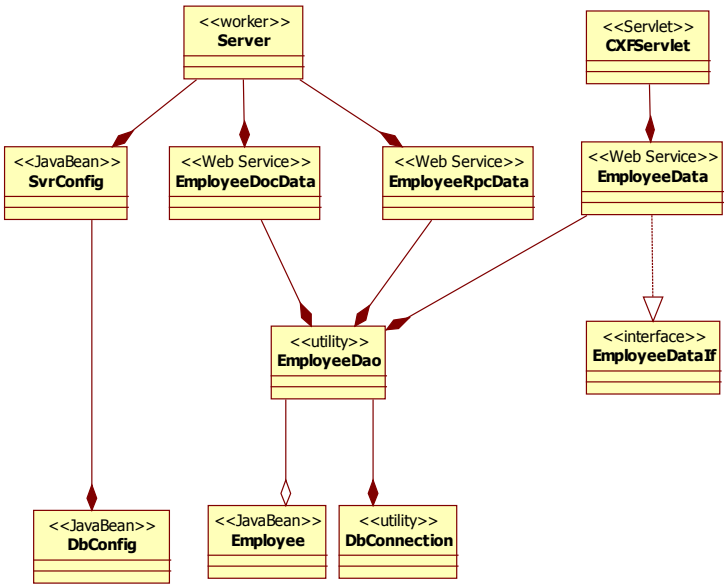

Figure 5-1. *Class Diagram for a CXF Web Service*

5.3.2 Deployment Diagram

We will deploy the Java Web Application onto the Tomcat 7 server.

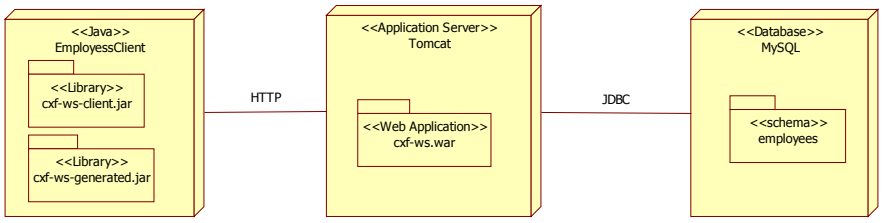

Figure 5-2. *Deployment Diagram for a CXF Web Service*

Developing and deploying Web Services involves the following major activities:

1. Create a dynamic Web application.
2. Create Web Services in Java.
3. Package the WAR file.
4. Deploy the WAR file to the Tomcat 7 server.

Figure 5-3. *Activities for Creating a Web Service Application*

5.3.3 Create a Dynamic Web Application

From Eclipse, create a new dynamic Web Project. The steps are as follows:

- Choose File → New → Other … → Web → Dynamic Web Project
- Choose all default values and name the project 'cxf-ws'.
- Choose cxf-ws project from the left panel, choose File → New → Folder. Name new folder lib.
- Now, we need the following Java libraries:

```
- mysql-connector-java-5.1.24-bin.jar: This contains the JDBC
  driver for MySQL database.
- java-ws.jar: This contains the main logic for accessing
  the database.
- data-svc.jar: This contains the data access classes.
```

- Import these files (from previous projects and from the Apache CXF distribution) into the lib folder we created earlier:

```
mysql-connector-java-5.1.24-bin.jar
ws-ch-1.jar
aopalliance-1.0.jar
asm-3.3.jar
axis.jar
commons-dbcp-1.4.jar
commons-discovery-0.2.jar
commons-logging-1.1.1.jar
commons-logging.jar
cxf-2.4.0.jar
geronimo-activation_1.1_spec-1.1.jar
geronimo-annotation_1.0_spec-1.1.1.jar
geronimo-javamail_1.4_spec-1.7.1.jar
geronimo-servlet_3.0_spec-1.0.jar
geronimo-ws-metadata_2.0_spec-1.1.3.jar
jaxb-api-2.2.1.jar
jaxb-impl-2.2.1.1.jar
jaxrpc.jar
neethi-3.0.0.jar
saaj-api-1.3.jar
saaj-impl-1.3.2.jar
spring-aop-3.0.5.RELEASE.jar
spring-asm-3.0.5.RELEASE.jar
spring-beans-3.0.5.RELEASE.jar
spring-context-3.0.5.RELEASE.jar
spring-core-3.0.5.RELEASE.jar
spring-expression-3.0.5.RELEASE.jar
spring-web-3.0.5.RELEASE.jar
stax2-api-3.1.1.jar
woodstox-core-asl-4.1.1.jar
wsdl4j-1.6.2.jar
wsdl4j.jar
xml-resolver-1.2.jar
xmlschema-core-2.0.jar
```

After we have finished coding Java classes for cxf-ws, the project should look like this:

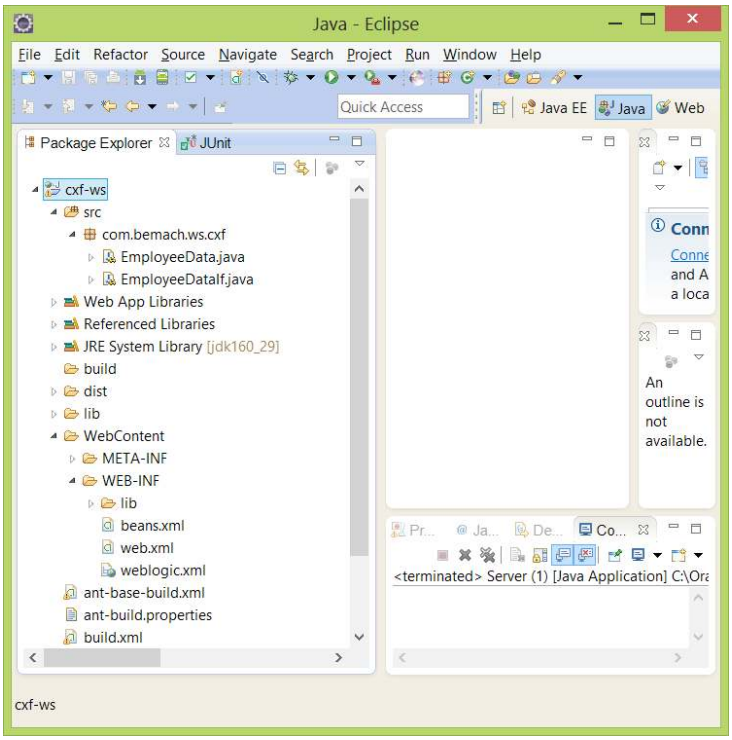

Figure 5-4. *Screenshot of a Dynamic Web Project*

5.3.4 Create Web Service

From the `cxf-ws/Java Resources/src` directory, create a Java package called '`com.bemach.ws.cxf`'. From this package, create one Java interface and one Java class.

5.3.4.1 EmployeeDataIf.java

We introduced another approach to create a Web Service using Java interface. This gives us a cleaner way to specify the service interface prior to implementing the Web Service. All WS annotations that were normally used for a Java class are used in a similar way.

Listing 5-1. EmployeeDataIf.java Class with Web Service Annotations

```
package com.bemach.ws.cxf;
/**
 * 2013 (C) BEM, Inc., Fairfax, Virginia
 *
 * Unless required by applicable law or agreed to in writing,
 * software distributed is distributed on an
 * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * KIND, either express or implied.
 *
 */

import java.sql.SQLException;

import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.xml.soap.SOAPException;

import com.bemach.data.Employee;

@WebService
@SOAPBinding(style=SOAPBinding.Style.DOCUMENT)
public interface EmployeeDataIf {
   @WebMethod
   Employee getEmployee(@WebParam(name="emplNo")long emplNo) throws
SOAPException, SQLException;

   @WebMethod
   long createEmployee(@WebParam (name="employee")Employee employee) throws
SOAPException, SQLException;

   @WebMethod
   boolean updateEmployee(@WebParam (name="employee")Employee employee) throws
SOAPException, SQLException;

   @WebMethod
   boolean deleteEmployee(@WebParam(name="emplNo")long emplNo) throws
SOAPException, SQLException;
}
```

The service will contain four basic operations:

1. The getEmployee method, which returns an employee of a given employee number. If not found, an exception is thrown;
2. The createEmployee method, which returns an employee number if created successfully. If failed, the method returns -1;
3. The updateEmployee method, which returns a Boolean value of true if successful, otherwise false; and
4. The deleteEmployee method, which returns a Boolean value of true if successful, otherwise false.

5.3.4.2 EmployeeData.java

This class is a wrapper class of the EmployeeDao.java class that we have seen earlier. It implements the WS interface EmployeeDataIf.

Listing 5-2. EmployeeData.java: An implementation of a Web Service Interface

```
package com.bemach.ws.cxf;
/**
 * 2013 (C) BEM, Inc., Fairfax, Virginia
 *
 * Unless required by applicable law or agreed to in writing,
 * software distributed is distributed on an
 * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * KIND, either express or implied.
 *
 */

import java.sql.SQLException;
import java.util.logging.Logger;

import javax.jws.WebService;
import javax.xml.soap.SOAPException;

import com.bemach.data.DbConfig;
import com.bemach.data.Employee;
import com.bemach.data.EmployeeDao;

@WebService(endpointInterface="com.bemach.ws.cxf.EmployeeDataIf")
public class EmployeeData implements EmployeeDataIf {
   private static final Logger LOG = Logger.getLogger(EmployeeData.class.getName());
   private static DbConfig cfg = new DbConfig();

   public static DbConfig getCfg() {
      return cfg;
   }

   public static void setCfg(DbConfig cfg) {
      EmployeeData.cfg = cfg;
   }

   public EmployeeData(DbConfig cfg) {
      EmployeeData.cfg = cfg;
   }

   public EmployeeData() {
   }

   @Override
   public Employee getEmployee(long emplNo) throws SOAPException, SQLException
{
      LOG.info("read employee");
      EmployeeDao dao = new EmployeeDao(cfg);
      Employee employee;
      employee = dao.getEmpl((int) emplNo);

      if (employee == null) {
         throw new SOAPException("GetEmployee: No such employee!");
      }
```

```
        return employee;
    }

    @Override
    public long createEmployee(Employee employee) throws SOAPException, SQLException {
        LOG.info("create employee");
        EmployeeDao dao = new EmployeeDao(cfg);
        return dao.createEmpl(employee);
    }

    @Override
    public boolean updateEmployee(Employee employee) throws SOAPException,
SQLException {
        LOG.info("update employee.");
        EmployeeDao dao = new EmployeeDao(cfg);
        return dao.updateEmpl(employee);
    }

    @Override
    public boolean deleteEmployee(long emplNo) throws SOAPException, SQLException {
        LOG.info("delete employee.");
        EmployeeDao dao = new EmployeeDao(cfg);
        return dao.deleteEmpl((int)emplNo);
    }
}
```

A default configuration is used. The EmployeeData constructor can be used to modify the configuration cfg object if necessary.

5.3.5 Package a WAR

The two important files – beans.xml and web.xml – are stored in the WEB-INF directory of the WAR file. These files assist the Tomcat Server and CXF to implement the Web Services that perform the actual work. Remember, we implement the sample Web Service using a Web application to be deployed on a Web application server (Tomcat or others).

5.3.5.1 Create web.xml

Listing 5-3. Content of web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
   <context-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>WEB-INF/beans.xml</param-value>
   </context-param>

   <listener>
      <listener-class>
         org.springframework.web.context.ContextLoaderListener
      </listener-class>
   </listener>

   <servlet>
      <servlet-name>CXFServlet</servlet-name>
      <display-name>CXF Servlet</display-name>
      <servlet-class>
         org.apache.cxf.transport.servlet.CXFServlet
      </servlet-class>
      <load-on-startup>1</load-on-startup>
   </servlet>

   <servlet-mapping>
      <servlet-name>CXFServlet</servlet-name>
      <url-pattern>/*</url-pattern>
   </servlet-mapping>
</web-app>
```

5.3.5.2 Create beans.xml

Listing 5-4. Content of beans.xml

```
<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">

       <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-servlet.xml" />

       <jaxws:endpoint id="com.bemach.ws.cxf.EmployeeDataIf"
implementor="com.bemach.ws.cxf.EmployeeData"address="/employees" />
</beans>
```

5.3.5.3 Build Web Application (WAR)

This build script creates a Java Web application (WAR file) packed into a file called 'cxf-ws.war'. The application is stored in the dist directory.

Listing 5-5. Content of build.xml for cxf-ws Dynamic Web Project

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<project name="cxf-ws" basedir="." default="dist">

   <property environment="env" />
   <path id="classpath.base">
      <fileset dir="./lib" includes="**/*.jar" />
   </path>

   <path id="classpath.compile">
      <path refid="classpath.base" />
   </path>

   <target name="init">
      <mkdir dir="./bin" />
      <mkdir dir="./dist" />
   </target>

   <target name="compile" depends="init" description="compile the source ">
      <javac srcdir="./src" destdir="./bin" debug="true">
         <classpath refid="classpath.compile" />
      </javac>
   </target>

   <target name="copylibs" depends="compile">
      <echo message="copying libraries ..."/>
      <copy todir="WebContent/WEB-INF/lib">
         <fileset dir="./lib">
            <include name="*.jar"/>
         </fileset>
      </copy>
   </target>

   <target name="dist" depends="buildwar">
      <echo message="Building ..."/>
   </target>

   <target name="buildwar" depends="copylibs">
      <echo message="building war ...."/>
      <tstamp/>
      <manifest file="MANIFEST.MF">
            <attribute name="Built-By" value="Kiet T. Tran"/>
         <attribute name="Build-Version" value="1"/>
         <attribute name="Build-Subversion" value="0"/>
         <attribute name="Built-Date" value="April 28, 2013"/>
      </manifest>
```

```
      <war destfile="./dist/cxf-ws.war" webxml="WebContent/WEB-INF/web.xml"
update="true"
            manifest="MANIFEST.MF">
         <classes dir="./bin" />
         <fileset dir="WebContent">
            <exclude name="WEB-INF/web.xml" />
         </fileset>
      </war>
      <delete file="MANIFEST.MF"/>
   </target>

   <target name="clean">
      <echo message="cleaning ...."/>
      <delete dir="./dist"/>
      <delete dir="./bin"/>
   </target>
</project>
```

5.4 Deploy the Service

When you install Tomcat 7, the Tomcat server home directory has the following directories:

- bin
- conf
- lib
- logs
- temp
- webapps
- work

We are most interested in the bin and the webapps directories for deploying and publishing our Web Services. To run Tomcat, go to the bin directory and run startup.cmd (WINDOWS) or startup.sh (UNIX).

To deploy the Web application that contains the sample Web Services, copy the cxf-web.war to the webapps directory. You need not restart the Tomcat server each time you deploy the Web application.

5.5 Testing services with SOAPUI

The testing of the new Web Service that is running on the Apache Tomcat server is similar to that of the Java WS Endpoint deployment described earlier. The Web Service endpoint is:

http://localhost:8080/cxf-ws/employees?WSDL

5.5.1 Check WSDL

Listing 5-6. A WSDL of a CXF Web Service Application

```
<wsdl:definitions xmlns:ns1="http://schemas.xmlsoap.org/soap/http"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://cxf.ws.bemach.com/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   name="EmployeeDataService" targetNamespace="http://cxf.ws.bemach.com/">
   <wsdl:types>
      <xs:schema xmlns:ns1="http://bemach.com" xmlns:tns="http://cxf.ws.bemach.com/"
         xmlns:xs="http://www.w3.org/2001/XMLSchema"
attributeFormDefault="unqualified"
         elementFormDefault="unqualified"
targetNamespace="http://cxf.ws.bemach.com/">
         <xs:import namespace="http://bemach.com" />
         <xs:element name="createEmployee" type="tns:createEmployee" />
         <xs:element name="createEmployeeResponse" type="tns:createEmployeeResponse"
/>
         <xs:element name="deleteEmployee" type="tns:deleteEmployee" />
         <xs:element name="deleteEmployeeResponse" type="tns:deleteEmployeeResponse"
/>
         <xs:element name="getEmployee" type="tns:getEmployee" />
         <xs:element name="getEmployeeResponse" type="tns:getEmployeeResponse" />
         <xs:element name="updateEmployee" type="tns:updateEmployee" />
         <xs:element name="updateEmployeeResponse" type="tns:updateEmployeeResponse"
/>
         <xs:complexType name="createEmployee">
            <xs:sequence>
               <xs:element minOccurs="0" name="employee" type="tns:employee" />
            </xs:sequence>
         </xs:complexType>
         <xs:complexType name="employee">
            <xs:sequence>
               <xs:element name="emplNo" type="xs:long" />
               <xs:element minOccurs="0" name="firstName" type="xs:string" />
               <xs:element minOccurs="0" name="lastName" type="xs:string" />
               <xs:element minOccurs="0" name="birthDate" type="xs:dateTime" />
               <xs:element minOccurs="0" name="gender" type="xs:string" />
               <xs:element minOccurs="0" name="hireDate" type="xs:dateTime" />
            </xs:sequence>
         </xs:complexType>
         <xs:complexType name="createEmployeeResponse">
            <xs:sequence>
               <xs:element name="return" type="xs:long" />
            </xs:sequence>
         </xs:complexType>
         <xs:complexType name="deleteEmployee">
            <xs:sequence>
               <xs:element name="emplNo" type="xs:long" />
            </xs:sequence>
         </xs:complexType>
```

```
        <xs:complexType name="deleteEmployeeResponse">
           <xs:sequence>
              <xs:element name="return" type="xs:boolean" />
           </xs:sequence>
        </xs:complexType>
        <xs:complexType name="getEmployee">
           <xs:sequence>
              <xs:element name="emplNo" type="xs:long" />
           </xs:sequence>
        </xs:complexType>
        <xs:complexType name="getEmployeeResponse">
           <xs:sequence>
              <xs:element minOccurs="0" name="return" type="tns:employee" />
           </xs:sequence>
        </xs:complexType>
        <xs:complexType name="updateEmployee">
           <xs:sequence>
              <xs:element minOccurs="0" name="employee" type="tns:employee" />
           </xs:sequence>
        </xs:complexType>
        <xs:complexType name="updateEmployeeResponse">
           <xs:sequence>
              <xs:element name="return" type="xs:boolean" />
           </xs:sequence>
```

```
         </xs:complexType>
         <xs:complexType name="SOAPException">
            <xs:sequence />
         </xs:complexType>
         <xs:element name="SOAPException" type="tns:SOAPException" />
      </xs:schema>
      <xs:schema xmlns:ns1="http://cxf.ws.bemach.com/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
         targetNamespace="http://bemach.com" version="1.0">
         <xs:import namespace="http://cxf.ws.bemach.com/" />
         <xs:element name="EmployeeService" type="ns1:employee" />
      </xs:schema>
   </wsdl:types>
   <wsdl:message name="createEmployeeResponse">
      <wsdl:part element="tns:createEmployeeResponse" name="parameters"></wsdl:part>
   </wsdl:message>
   <wsdl:message name="updateEmployee">
      <wsdl:part element="tns:updateEmployee" name="parameters"></wsdl:part>
   </wsdl:message>
   <wsdl:message name="getEmployeeResponse">
      <wsdl:part element="tns:getEmployeeResponse" name="parameters"></wsdl:part>
   </wsdl:message>
   <wsdl:message name="SOAPException">
      <wsdl:part element="tns:SOAPException" name="SOAPException"></wsdl:part>
   </wsdl:message>
   <wsdl:message name="updateEmployeeResponse">
      <wsdl:part element="tns:updateEmployeeResponse" name="parameters"></wsdl:part>
   </wsdl:message>
   <wsdl:message name="deleteEmployeeResponse">
      <wsdl:part element="tns:deleteEmployeeResponse" name="parameters"></
wsdl:part>
   </wsdl:message>
   <wsdl:message name="getEmployee">
      <wsdl:part element="tns:getEmployee" name="parameters"></wsdl:part>
   </wsdl:message>
   <wsdl:message name="createEmployee">
      <wsdl:part element="tns:createEmployee" name="parameters"></wsdl:part>
   </wsdl:message>
   <wsdl:message name="deleteEmployee">
      <wsdl:part element="tns:deleteEmployee" name="parameters"></wsdl:part>
   </wsdl:message>
   <wsdl:portType name="EmployeeDataIf">
      <wsdl:operation name="createEmployee">
         <wsdl:input message="tns:createEmployee"
name="createEmployee"></wsdl:input>
         <wsdl:output message="tns:createEmployeeResponse"
name="createEmployeeResponse"></wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="deleteEmployee">
         <wsdl:input message="tns:deleteEmployee"
name="deleteEmployee"></wsdl:input>
         <wsdl:output message="tns:deleteEmployeeResponse"
```

```
name="deleteEmployeeResponse"></wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getEmployee">
         <wsdl:input message="tns:getEmployee"
name="getEmployee"></wsdl:input>
         <wsdl:output message="tns:getEmployeeResponse"
name="getEmployeeResponse"></wsdl:output>
         <wsdl:fault message="tns:SOAPException" name="SOAPException"></wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="updateEmployee">
         <wsdl:input message="tns:updateEmployee"
name="updateEmployee"></wsdl:input>
         <wsdl:output message="tns:updateEmployeeResponse"
name="updateEmployeeResponse"></wsdl:output>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="EmployeeDataServiceSoapBinding" type="tns:EmployeeDataIf">
      <soap:binding style="document"
         transport="http://schemas.xmlsoap.org/soap/http" />
      <wsdl:operation name="createEmployee">
         <soap:operation soapAction="" style="document" />
         <wsdl:input name="createEmployee">
            <soap:body use="literal" />
         </wsdl:input>
         <wsdl:output name="createEmployeeResponse">
            <soap:body use="literal" />
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="deleteEmployee">
         <soap:operation soapAction="" style="document" />
         <wsdl:input name="deleteEmployee">
            <soap:body use="literal" />
         </wsdl:input>
         <wsdl:output name="deleteEmployeeResponse">
            <soap:body use="literal" />
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getEmployee">
         <soap:operation soapAction="" style="document" />
         <wsdl:input name="getEmployee">
            <soap:body use="literal" />
         </wsdl:input>
         <wsdl:output name="getEmployeeResponse">
            <soap:body use="literal" />
         </wsdl:output>
         <wsdl:fault name="SOAPException">
            <soap:fault name="SOAPException" use="literal" />
         </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="updateEmployee">
         <soap:operation soapAction="" style="document" />
         <wsdl:input name="updateEmployee">
            <soap:body use="literal" />
```

```
         </wsdl:input>
         <wsdl:output name="updateEmployeeResponse">
            <soap:body use="literal" />
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="EmployeeDataService">
      <wsdl:port binding="tns:EmployeeDataServiceSoapBinding"
         name="EmployeeDataPort">
         <soap:address location="http://localhost:8080/cxf-ws/employees" />
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

5.5.2 SOAPUI project

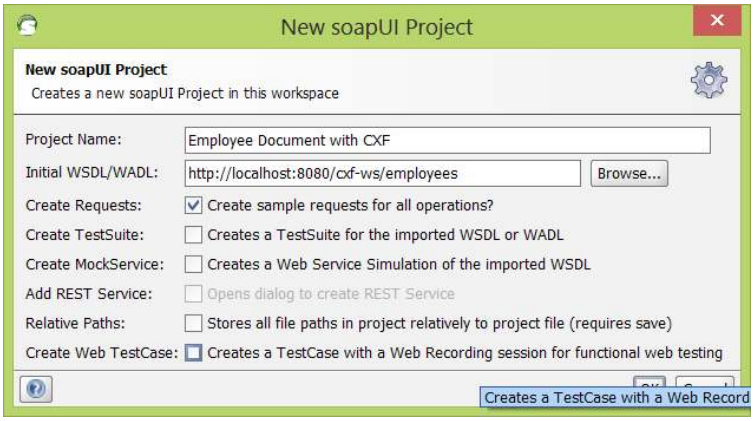

Figure 5-5. *Create a New SOAPUI Project*

The figure below shows a result of a call to the getEmployee operation.

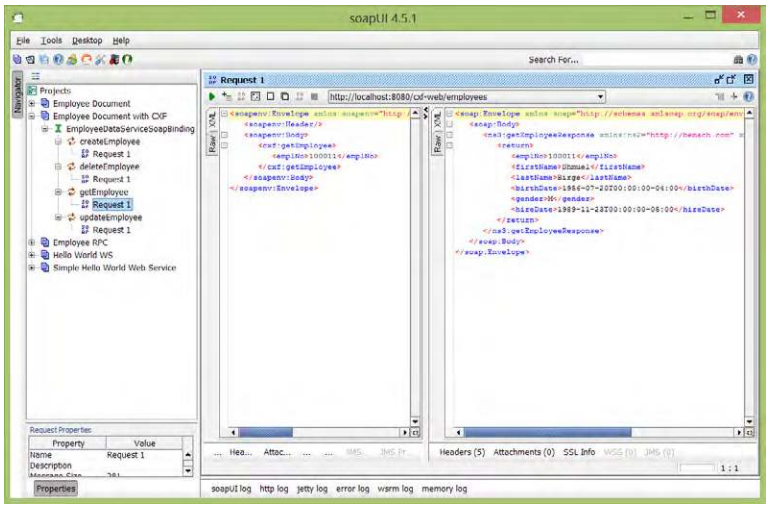

Figure 5-6. *Executing a Web Service Operation*

5.6 Develop a Web Service Consumer

5.6.1 Create Client Stub with wsimport

First, create a Java Project under Eclipse IDE and call it 'cxf-ws-client'.

1. At the command prompt, go to the Java Project for Eclipse called 'cxf-ws-client'.
2. Create a folder called 'generated'.
3. To generate Web Service stubs, run the following command:

```
wsimport -d . http://localhost:8080/cxf-ws/employees?WSDL
```

4. To create a Java library, run the following command:

```
jar -cvf ../lib/cxf-ws-generated.jar *
```

5. Now, verify the content of the created jar:

```
jar -tf ../lib/cxf-ws-generated.jar *
```

The content of the library should look like this:

```
META-INF/
META-INF/MANIFEST.MF
com/
com/bemach/
com/bemach/ObjectFactory.class
com/bemach/ws/
com/bemach/ws/cxf/
com/bemach/ws/cxf/Employee.class
com/bemach/ws/cxf/EmployeeDataIf.class
com/bemach/ws/cxf/EmployeeDataService.class
com/bemach/ws/cxf/ObjectFactory.class
com/bemach/ws/cxf/package-info.class
com/bemach/ws/cxf/SOAPException.class
com/bemach/ws/cxf/SOAPException_Exception.class
```

5.6.2 Create Client Code

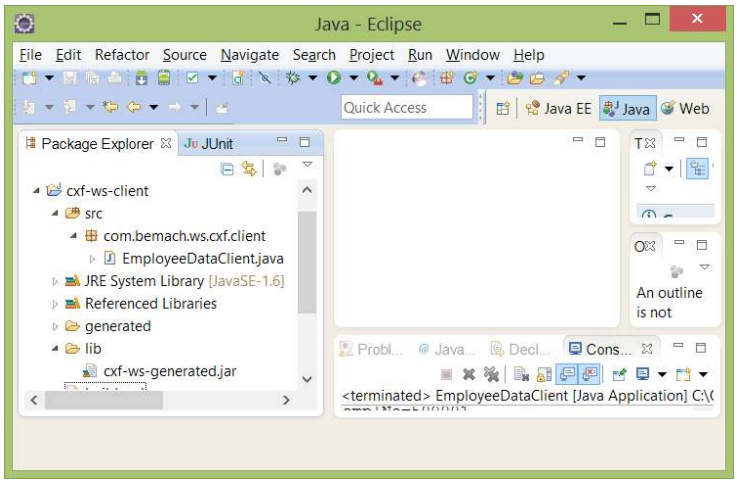

Figure 5-7. *Screenshot of cxf-ws-client Java Project*

5.6.2.1 EmployeeDataClient.java

This Java class contains code that can invoke the four operations provided by the EmployeeDataService hosted by the Tomcat 7 server.

Listing 5-7. EmployeeDataClient.java class

```
package com.bemach.ws.cxf.client;
/**
 * 2013 (C) BEM, Inc., Fairfax, Virginia
 *
 * Unless required by applicable law or agreed to in writing,
 * software distributed is distributed on an
 * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * KIND, either express or implied.
 *
 */

import java.net.MalformedURLException;
import java.net.URL;
import java.util.logging.Logger;

import javax.xml.namespace.QName;
import javax.xml.ws.Service;

import com.bemach.ws.cxf.Employee;
import com.bemach.ws.cxf.EmployeeDataIf;
import com.bemach.ws.cxf.EmployeeDataService;
import com.bemach.ws.cxf.SOAPException_Exception;

/**
 * This class is a wrapper class for accessing a remote web service.
 *
 * @author ktran
 * @version 1.0
 */
public class EmployeeDataClient {
   private static final Logger LOG = Logger.getLogger(EmployeeDataClient.class.
getName());
   private EmployeeDataIf emplDs = null;

   /**
    * This method constructs the EmployeeDataClient object.
    *
    * @param urlStr - The URL of the Web Service
    * @param targetNs - Namespace of the target web service
    * @param name - the name of the web service.
    * @throws MalformedURLException - if an error occurs, exception is thrown.
    *
    */
   public EmployeeDataClient(String urlStr, String targetNs, String name)
         throws MalformedURLException {
      LOG.info("Constructor ...");
      QName qName = new QName(targetNs, name);
      URL url = new URL(urlStr);
      Service service = EmployeeDataService.create(url, qName);
      emplDs = service.getPort(EmployeeDataIf.class);
   }
```

```
    /**
     * This method gets an employee record based on a unique id.
     *
     * @param id - unique employee id.
     * @return - an employee record.
     * @throws SOAPException_Exception - an exception is thrown.
     */
    public Employee get(long id) throws SOAPException_Exception {
        return emplDs.getEmployee(id);
    }

    /**
     * This method creates an employee record.
     *
     * @param empl
     * @return
     */
    public long create(Employee empl) {
        return emplDs.createEmployee(empl);
    }

    /**
     * This method deletes an employee record based on a unique id.
     *
     * @param id
     * @return
     */
    public boolean delete(long id) {
        return emplDs.deleteEmployee(id);
    }

    /**
     * This method updates an employee record with a new one.
     *
     * @param empl
     * @return
     */
    public boolean update(Employee empl) {
        return emplDs.updateEmployee(empl);
    }

    /**
     * This is the main entrance of the code.
     *
     * @param args
     * @throws MalformedURLException
     * @throws SOAPException_Exception
     */
```

```
   public static void main(String[] args)
         throws MalformedURLException, SOAPException_Exception {
      LOG.info("Calling Employee (CXF) data service ... ");
      String targetNameSpace = "http://cxf.ws.bemach.com/";
      String name = "EmployeeDataService";
      String portNo = args[0];
      String urlStr = String.format("http://localhost:%s/cxf-ws/
employees?WSDL",portNo);

      EmployeeDataClient cli = new EmployeeDataClient(urlStr, targetNameSpace,
name);
      LOG.info("URL="+urlStr);

      long oldEmplNo = Integer.valueOf(args[1]);
      Employee empl = cli.get(oldEmplNo);
      LOG.info("last="+empl.getLastName());
      LOG.info("hire="+empl.getHireDate());
      LOG.info("last="+empl.getLastName());
      LOG.info("first="+empl.getFirstName());

      empl.setFirstName("Silvester");
      empl.setLastName("Johnny");
      long newEmplNo = cli.create(empl);
      LOG.info("emplNo="+newEmplNo);

      Employee newEmpl = cli.get(newEmplNo);

      newEmpl.setLastName("New-name");
      newEmpl.setEmplNo(newEmplNo);
      boolean status = cli.update(newEmpl);
      LOG.info("update:"+status);
      LOG.info("last="+newEmpl.getLastName());
      LOG.info("first="+newEmpl.getFirstName());

      status = cli.delete(newEmplNo);
      LOG.info("deleteEmployee:"+status);
      LOG.info("Exit!");
   }
}
```

In this class, we provide the port number as the first program argument. The Tomcat port number is 8080.

5.6.3 Build a Java library

This build script creates a Java library called 'cxf-ws-client.jar'. It requires the cxf-ws-generated.jar library that is stored in the dist directory.

Listing 5-8. Content of build.xml for cxf-ws-client Java Project

```
<project name="cxf-ws-client" default="dist" basedir=".">
   <description>
         Client for employees data service.
      Assumed that cxf-ws-generated.jar is generated and built elsehwere
   </description>

   <!- set global properties for this build ->
   <property environment="env"/>
   <path id="classpath.base">
      <fileset dir="./lib" includes="**/*.jar" />
   </path>

   <path id="classpath.compile">
      <path refid="classpath.base"/>
   </path>

   <target name="init">
      <mkdir dir="./bin"/>
      <mkdir dir="./dist"/>
   </target>

   <target name="compile" depends="init" description="compile the source " >
      <javac srcdir="./src" destdir="./bin" debug="true">
         <classpath refid="classpath.compile" />
      </javac>
   </target>

   <target name="dist" depends="compile" description="generate the distribution" >
      <!- Create the distribution directory ->
      <jar jarfile="./dist/cxf-ws-client.jar" basedir="./bin"/>

   </target>

   <target name="clean" description="clean up" >
      <!- Delete the ${build} directory trees ->
      <delete dir="./dist"/>
      <delete dir="./bin"/>
   </target>
</project>
```

5.6.4 Run the Client Application

To run the client application, go to the cxf-ws-client project directory. At the command prompt, run the following command:

```
java -cp ./lib/cxf-ws-generated.jar;./dist/cxf-ws-client.jar com.bemach.
ws.cxf.client.EmployeeDataClient 8080
```

The output is as follows:

```
Calling Employee (CXF) data service ...
URL=http://localhost:8080/cxf-ws/employees?WSDL
last=Fecello
hire=1986-06-26T00:00:00-04:00
last=Fecello
first=Silvester
emplNo=500001
update:true
last=New-name
first=Silvester
deleteEmployee:true
Exit!
```