6 Apache CXF and Oracle WebLogic Server

Objectives

After completing this chapter, you should be able to:

1. Understand the fundamentals of Oracle WebLogic 12
2. Create a basic WebLogic domain
3. Start an Administration Server
4. Use WebLogic to deploy a CXF WS application on the server
5. Test Web Service using a WebLogic Console

In this chapter, we take the same Web application – cxf-ws.war – and deploy it on a WebLogic Server version 12.

6.1 Oracle WebLogic Server 12

Oracle WebLogic Server (WLS) is also known as the 'Oracle Fusion Middleware'. The latest release of WLS is 12.1.1. Oracle WLS is an industrial-strength enterprise-ready Java platform, Enterprise Edition (Java EE) application server. It is the foundation for building Service-oriented Architectures (SOA) applications using Oracle software products.

Oracle WLS implements complete JEE 6 specification and provide a set of APIs for creating a variety of services: databases, messaging, and connections to external systems. Oracle WLS provides an environment capable of deploying mission-critical applications that are robust, secure, highly available and scalable.

Major advantages of using Oracle WLS are briefly described in the following sections:

6.1.1 Programming Modles

Oracle WLS comes with a set of tools that enable the following capabilities:

- Web Applications (JSP and Servlet)
- Web Services (JAX-WS, JAX-RPC, JAX-RS)
- XML Programming (JAXB)
- Java Messaging Service (JMS)
- Java Database Connectivity (JDBC)
- Resource Adapters (Enterprise Information Systems)

- Enterprise JavaBeans (EJB)
- Remote Method Invocation (RMI)
- Security APIs (Security Service Providers APIs)
- WebLogic Tuxedo Connectivity (WTC)

6.1.2 Highly Availability

Mission-critical applications can be supported with the following capabilities:

- WebLogic Server Clusters
- Work Managers
- Overload Protection
- Network Channels
- Weblogic Server Peristent Store
- Store-and-forward Services
- Enterprise-ready Development Tools
- Production Redeployment

6.2 Deployment Diagram

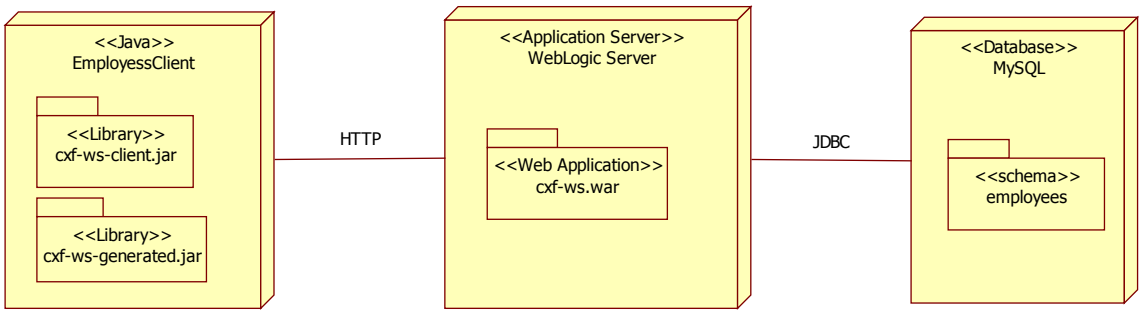

Figure 6-1. *Deployment Diagram for CXF Web Service Application and Oracle WebLogic Server*

6.3 Creating a WebLogic Domain

A WebLogic Server administration domain is a logical group of WLS resources. A domain is managed by a special type of server called an 'Administration Server'. This server instance is used for managing resources and configurations of these resources. Applications and services should not be deployed in an Administration Server; they should be deployed on Managed Server instances instead. A WLS domain may have one or more Managed Server instances.

Two or more Managed Servers can be grouped into a cluster. A domain can administer one or more clusters. For the sake of simplicity, we will deploy the CXF Web Service Application on an Administration Server. First, we create a WLS domain by following these steps:

1. Assuming that you installed Oracle WebLogic Server on Windows, go to C:\Oracle\Middleware\wlserver_12.1\common\bin.
2. Run config.cmd (or config.sh).
3. Follow the on-screen instructions to complete the creation of the domain.
4. Choose all default parameters.
5. Once complete, the domain is created and stored here: C:\Oracle\Middleware\user_projects\domains\base_domain

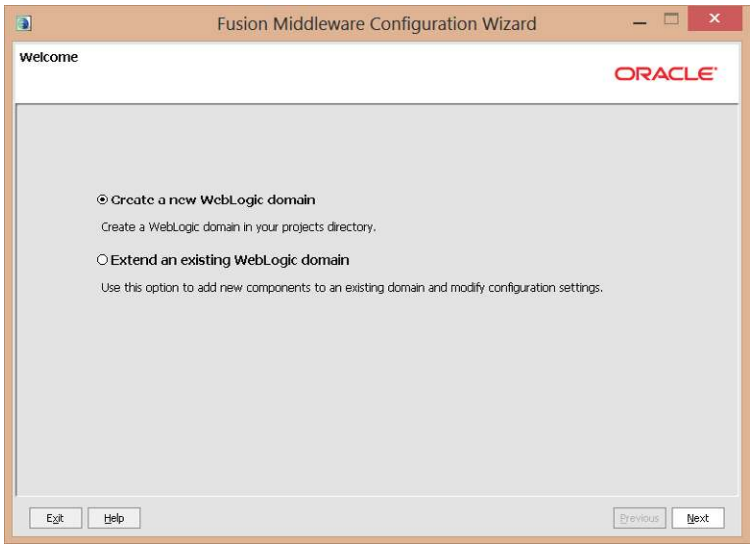

Figure 6-2. *Creating a WLS Domain*

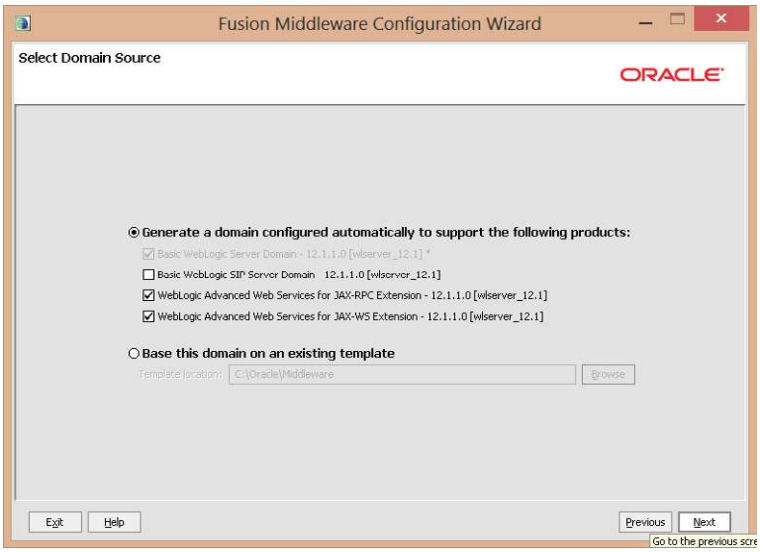

Figure 6-3. *Adding Extensions (JAX-WS and JAX-RPC)*

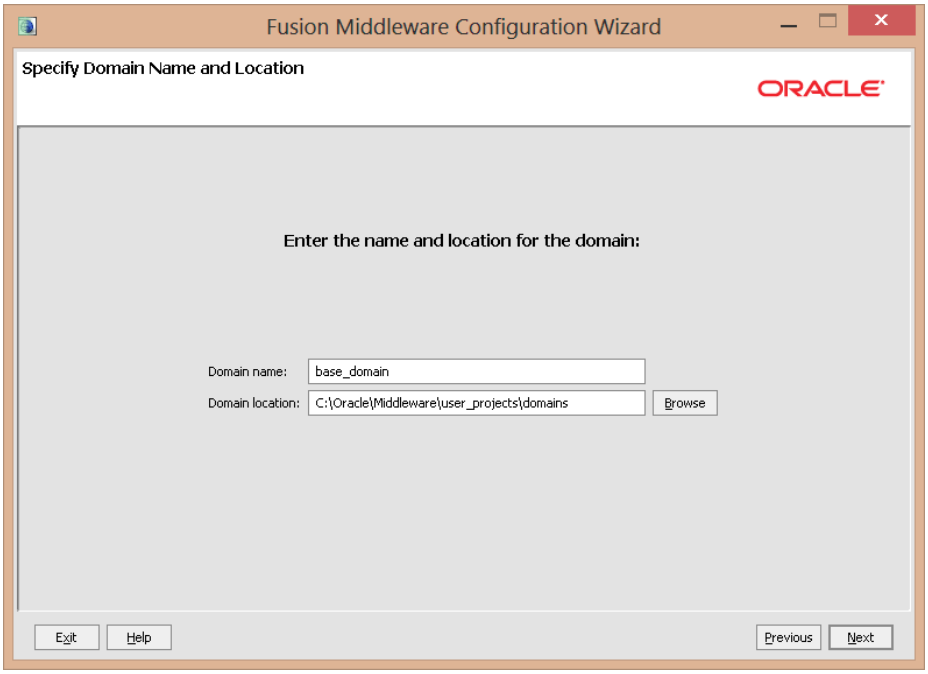

Figure 6-4. *Enter the Domain Name*

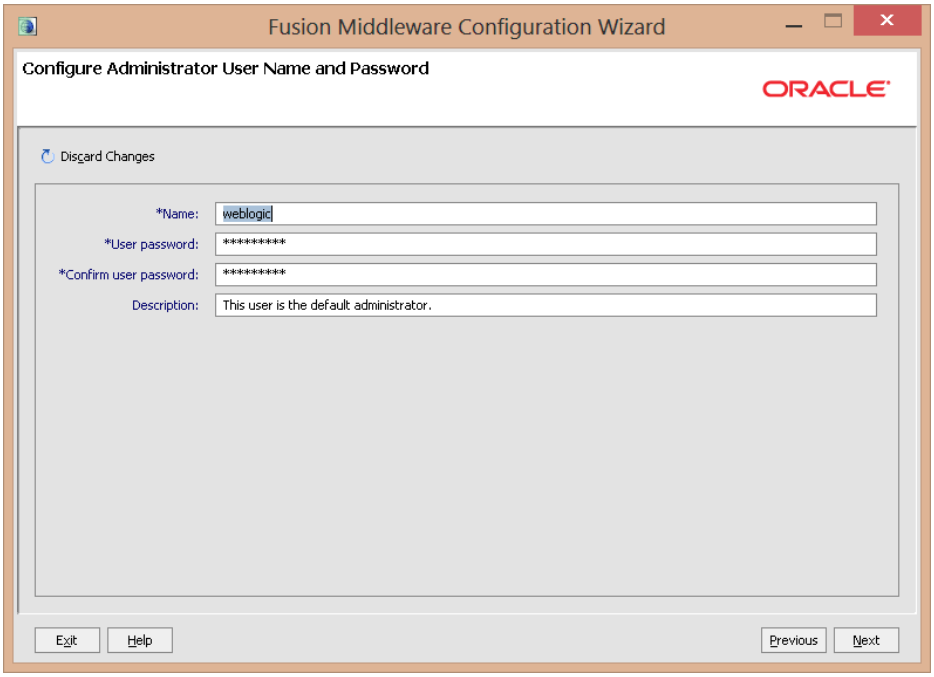

Figure 6-5. *Enter User ID and Password*

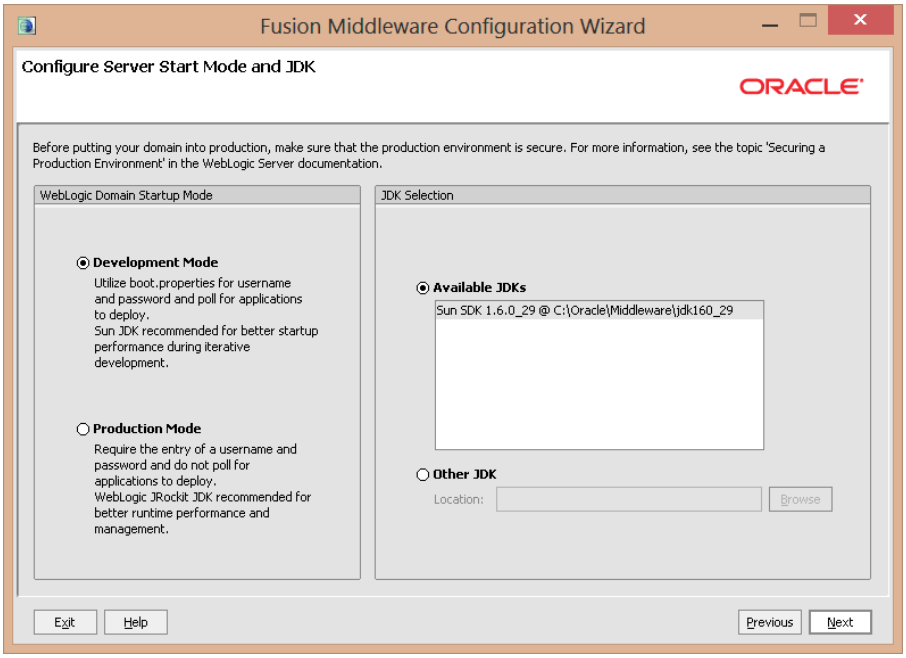

Figure 6-6. *Select a default JDK*

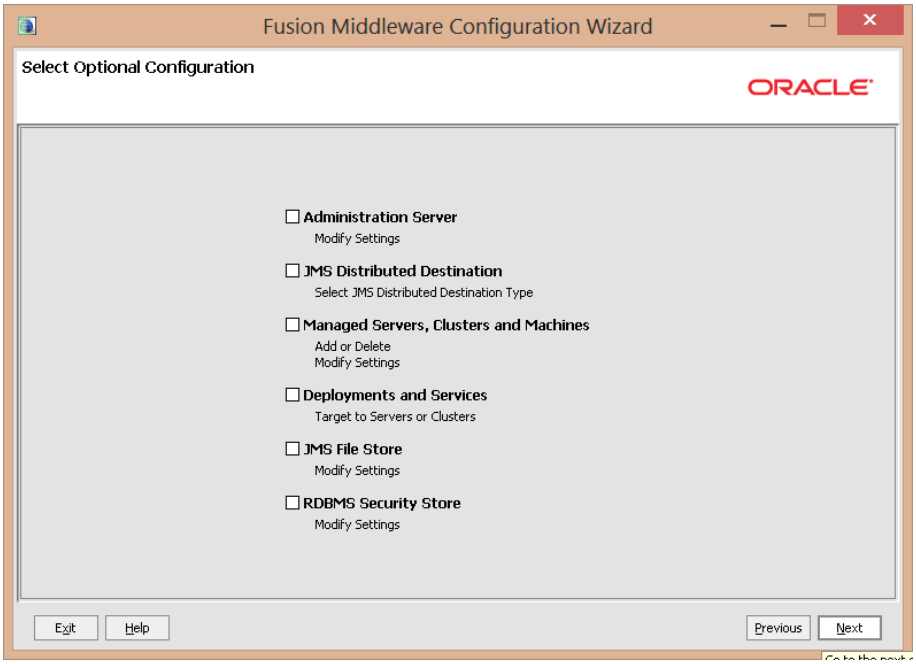

Figure 6-7. *Additional Configuration*

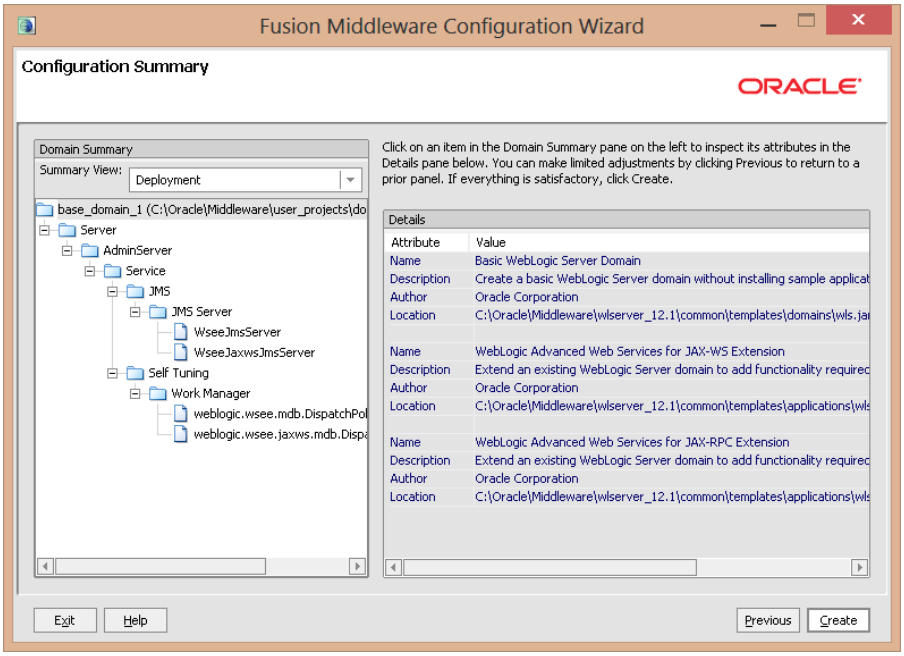

Figure 6-8. *Configuration Summary of the Domain*

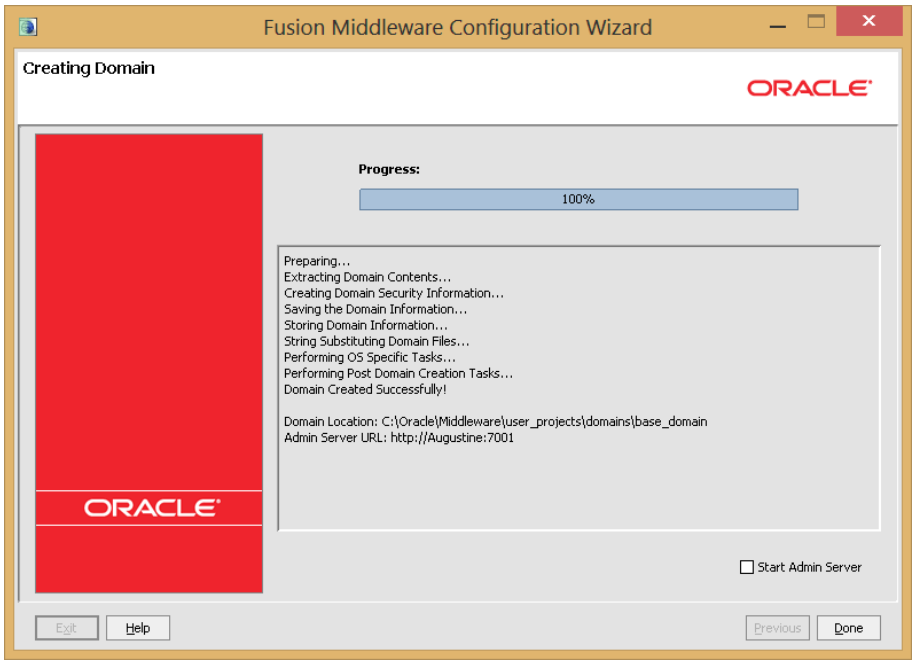

Figure 6-9. *Status of the Domain Creation*

6.3.1 Starting an Administration Server

To start an Administration Server:

1. Go to the domain directory:

   ```
   C:\Oracle\Middleware\user_projects\domains\base_domain
   ```

2. Run the following command:

   ```
   startweblogic.cmd
   ```

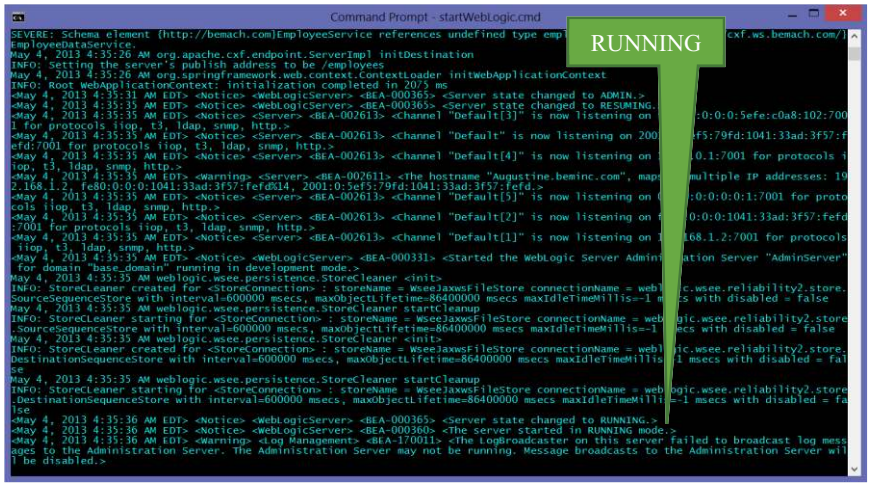

Figure 6-10. *Output of a WLS Administration Server*

Once the server has started successfully, it displays the following message in the command window:

```
<The server started in RUNNING mode.>
```

6.4 Deploy the Web Service

Make sure to include the weblogic.xml file in the WEB-INF directory of the cxf-ws project prior to building the Java Web Application.

6.4.1 weblogic.xml

This file contains WebLogic-specific configuration parameters. It is needed for deploying the CXF WS application on a WebLogic server.

Listing 6-1. Content of weblogic.xml to be included for cxf-ws.war Web Application

```
<?xml version="1.0" encoding="UTF-8"?>
<weblogic-web-app xmlns="http://xmlns.oracle.com/weblogic/weblogic-web-app"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.oracle.com/weblogic/weblogic-web-app
http://xmlns.oracle.com/weblogic/weblogic-web-app/1.0/weblogic-web-app.xsd">
      <context-root>cxf-ws</context-root>
      <jsp-descriptor>
            <precompile>true</precompile>
      </jsp-descriptor>

      <session-descriptor>
            <timeout-secs>900</timeout-secs>
            <invalidation-interval-secs>10</invalidation-interval-secs>
            <max-in-memory-sessions>500</max-in-memory-sessions>
      </session-descriptor>

</weblogic-web-app>
```

To deploy the CXF Web Application on a WebLogic Server, take the following steps:

1. Open a browser and go to http://localhost:7001/console. Login as weblogic/weblogic1 using the username/password that you defined during the domain creation step.

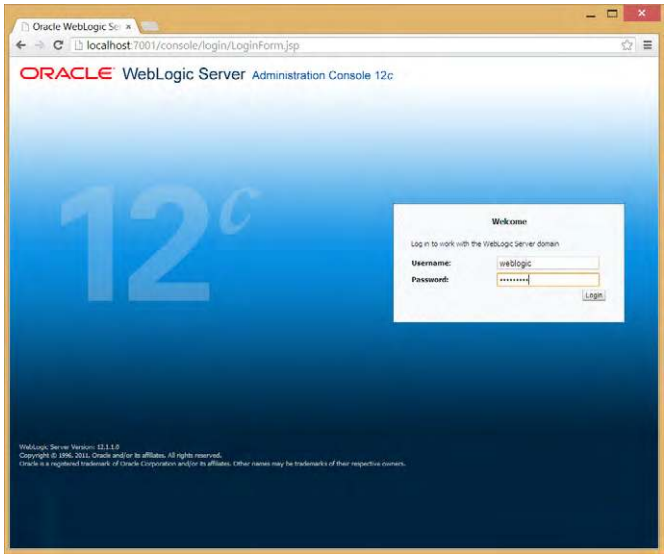

Figure 6-11. *OracleWLS Console Login Screen*

2. On the left panel, choose the `Deployments option`.

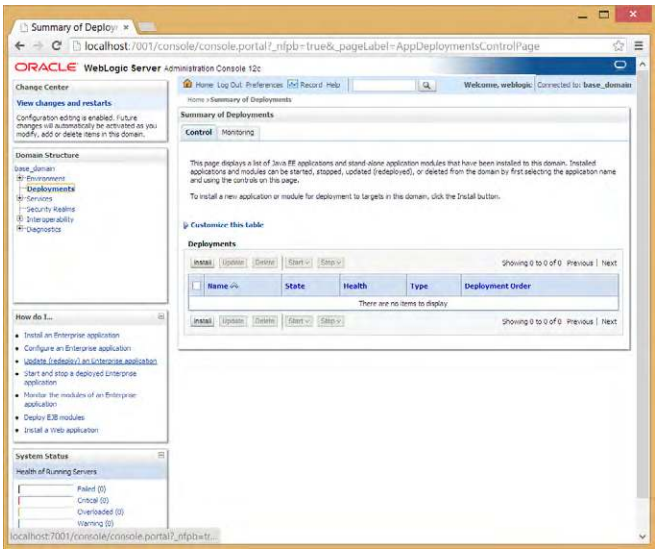

Figure 6-12. *Oracle WLS Deployment Screen*

3. On the right panel, click the Install button.
4. Choose the cxf-ws.war file located in the dist directory of the cxf-ws project.

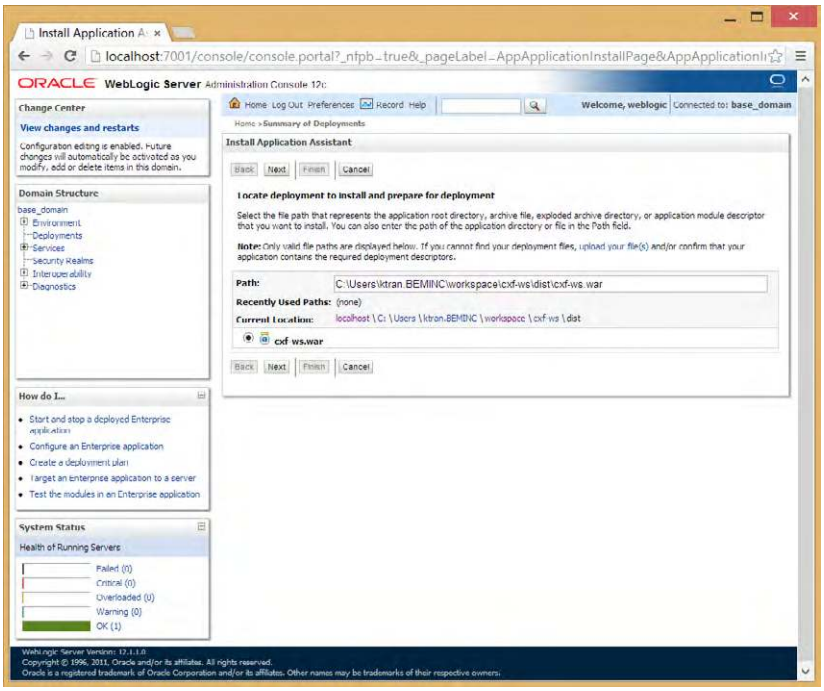

Figure 6-13. *Oracle WLS Install Application Screen*

5. Click 'Next'.

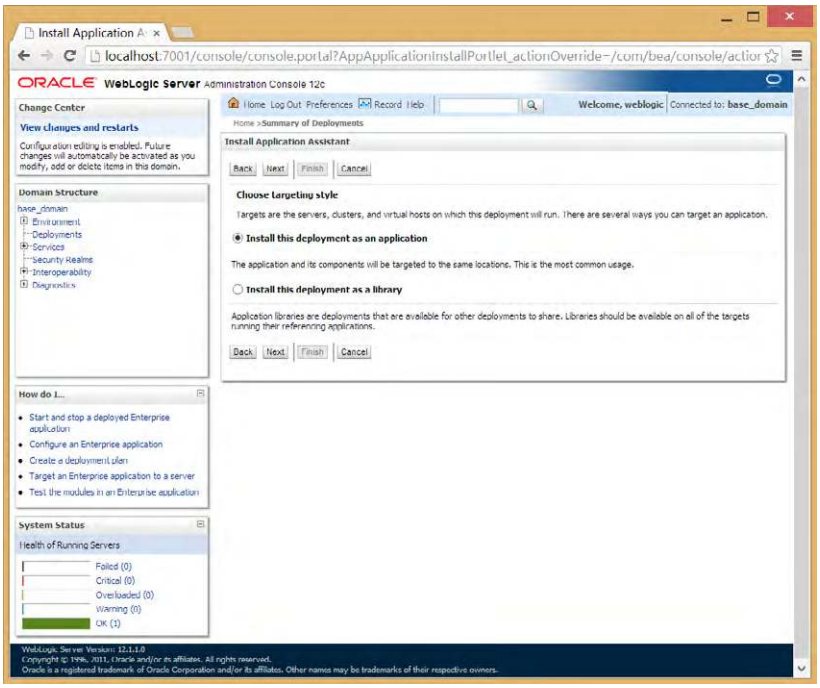

Figure 6-14. *Type of Deployment*

6. Click 'Next'.

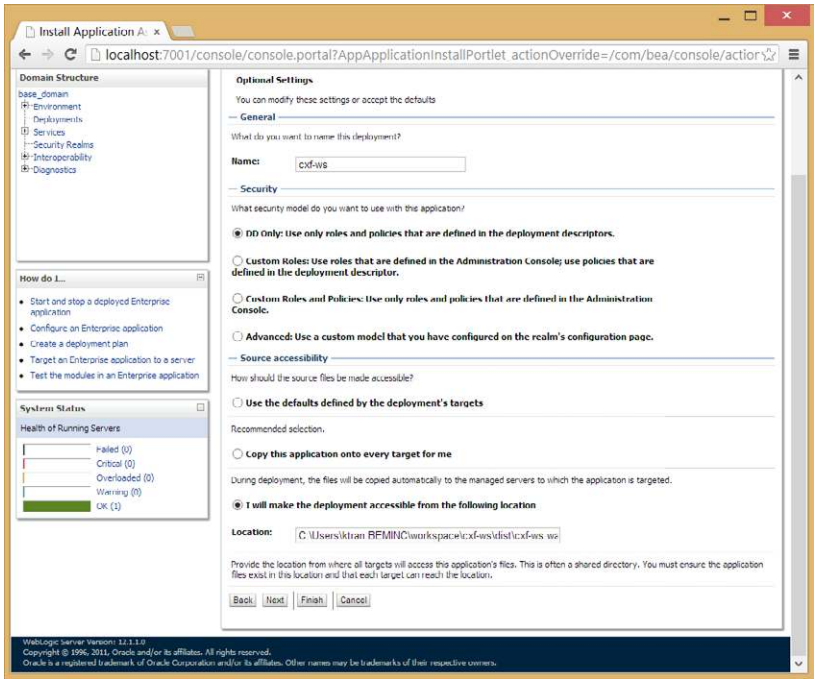

Figure 6-15. *Additional Settings for the Deployment Application Process*

7. Click 'Finish'.

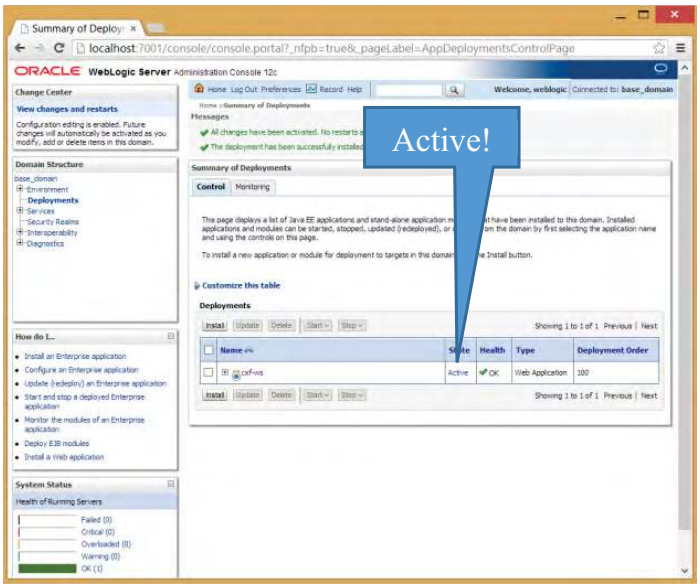

Figure 6-16. *Deployment Verification*

The state must show as 'Active'. Any other state can be a problem.

We are now ready to test the CXF Web Service that is hosted by an Oracle WebLogic Server.

6.5 Test CXF Web Service with WebLogic Test Tools

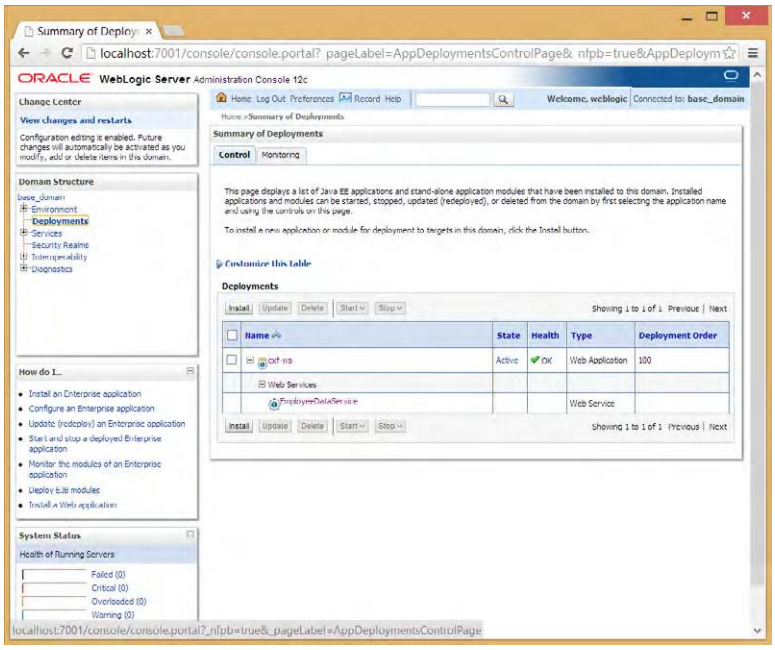

Figure 6-17. *Select the Web Service Application for Testing*

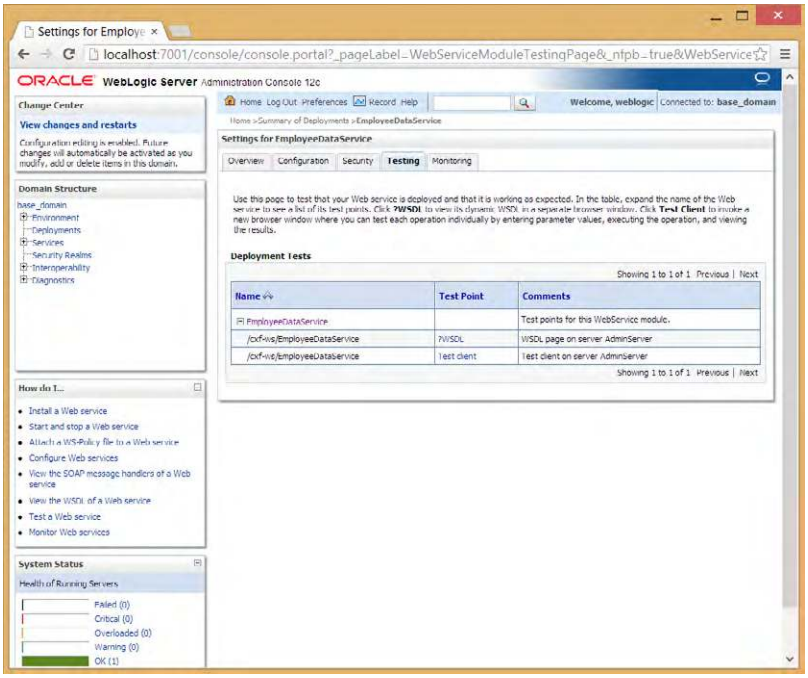

Figure 6-18. *Display of the Web Application*

Click on *Test Client URL*.

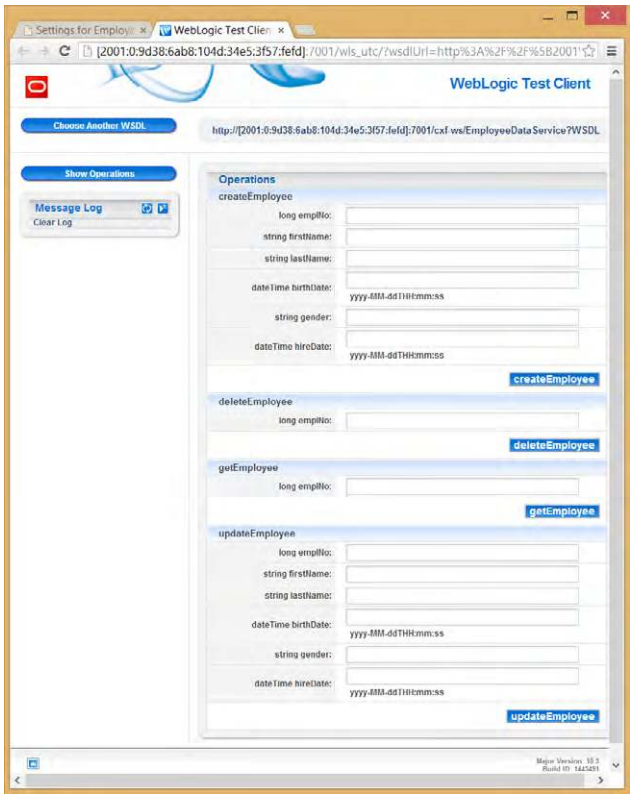

Figure 6-19. *WebLogic Test Client*

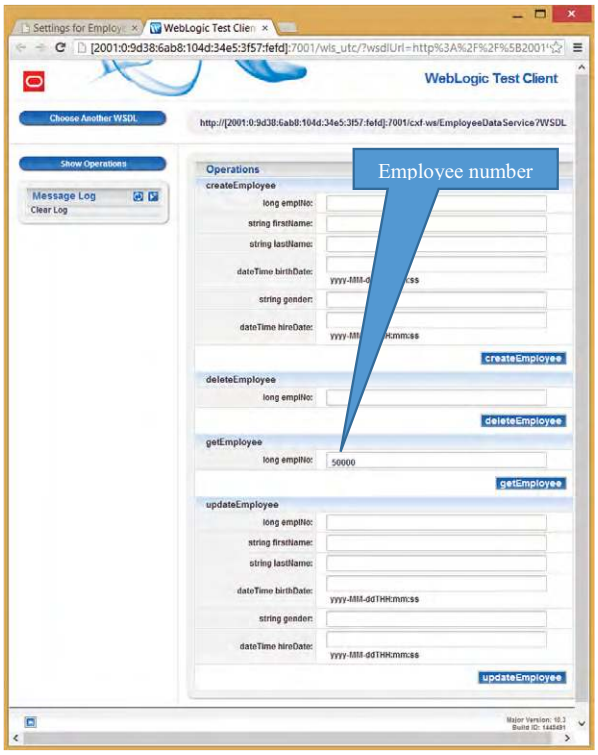

Figure 6-20. *Prepare to Run getEmployee Operation*

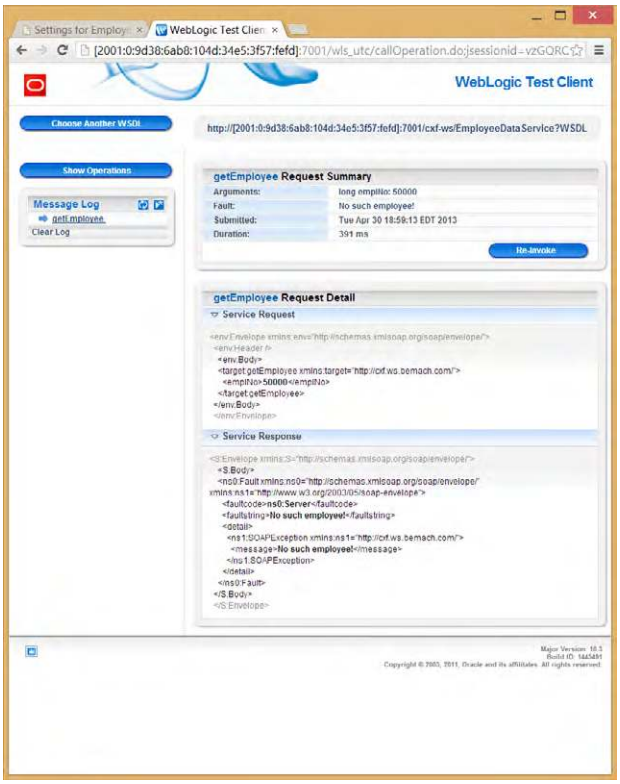

Figure 6-21. *Result of a call to getEmployee Operation*

6.5.1 Check WSDL

The WSDL is located at the following URL:

http://localhost:7001/cxf-ws/employees?wsdl

Listing 6-2. WSDL for CXF Web Application on Oracle WebLogic Server

```
<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>
      <xsd:schema xmlns="http://cxf.ws.bemach.com/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         attributeFormDefault="unqualified" elementFormDefault="unqualified"
         targetNamespace="http://cxf.ws.bemach.com/">
         <xsd:complexType name="employee">
            <xsd:sequence>
               <xsd:element name="emplNo" type="xsd:long" />
               <xsd:element minOccurs="0" name="firstName" type="xsd:string" />
               <xsd:element minOccurs="0" name="lastName" type="xsd:string" />
               <xsd:element minOccurs="0" name="birthDate" type="xsd:dateTime" />
               <xsd:element minOccurs="0" name="gender" type="xsd:string" />
               <xsd:element minOccurs="0" name="hireDate" type="xsd:dateTime" />
            </xsd:sequence>
         </xsd:complexType>
         <xsd:complexType name="SOAPException">
            <xsd:sequence />
         </xsd:complexType>
         <xsd:element name="SOAPException" type="SOAPException" />
      </xsd:schema>
      <xsd:schema xmlns:ns0="http://bemach.com"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         attributeFormDefault="unqualified" elementFormDefault="unqualified"
         targetNamespace="http://bemach.com">
         <xsd:import namespace="http://cxf.ws.bemach.com/" />
         <xsd:element name="EmployeeService" type="employee" />
      </xsd:schema>
   </wsdl:types>
   <wsdl:message name="createEmployeeResponse">
      <wsdl:part name="return" type="xsd:long"></wsdl:part>
   </wsdl:message>
   <wsdl:message name="getEmployeeResponse">
      <wsdl:part name="return" type="tns:employee"></wsdl:part>
   </wsdl:message>
   <wsdl:message name="updateEmployee">
      <wsdl:part name="employee" type="tns:employee"></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 name="return" type="xsd:boolean"></wsdl:part>
   </wsdl:message>
   <wsdl:message name="deleteEmployeeResponse">
      <wsdl:part name="return" type="xsd:boolean"></wsdl:part>
   </wsdl:message>
   <wsdl:message name="getEmployee">
      <wsdl:part name="emplNo" type="xsd:long"></wsdl:part>
   </wsdl:message>
   <wsdl:message name="createEmployee">
      <wsdl:part name="employee" type="tns:employee"></wsdl:part>
   </wsdl:message>
   <wsdl:message name="deleteEmployee">
      <wsdl:part name="emplNo" type="xsd:long"></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="updateEmployee">
         <wsdl:input message="tns:updateEmployee"
name="updateEmployee"></wsdl:input>
         <wsdl:output message="tns:updateEmployeeResponse"
name="updateEmployeeResponse"></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:portType>
   <wsdl:binding name="EmployeeDataServiceSoapBinding" type="tns:EmployeeDataIf">
      <soap:binding style="rpc"
         transport="http://schemas.xmlsoap.org/soap/http" />
      <wsdl:operation name="createEmployee">
         <soap:operation soapAction="" style="rpc" />
         <wsdl:input name="createEmployee">
            <soap:body namespace="http://cxf.ws.bemach.com/" use="literal" />
         </wsdl:input>
```

```
         <wsdl:output name="createEmployeeResponse">
            <soap:body namespace="http://cxf.ws.bemach.com/" use="literal" />
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="deleteEmployee">
         <soap:operation soapAction="" style="rpc" />
         <wsdl:input name="deleteEmployee">
            <soap:body namespace="http://cxf.ws.bemach.com/" use="literal" />
         </wsdl:input>
         <wsdl:output name="deleteEmployeeResponse">
            <soap:body namespace="http://cxf.ws.bemach.com/" use="literal" />
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getEmployee">
         <soap:operation soapAction="" style="rpc" />
         <wsdl:input name="getEmployee">
            <soap:body namespace="http://cxf.ws.bemach.com/" use="literal" />
         </wsdl:input>
         <wsdl:output name="getEmployeeResponse">
            <soap:body namespace="http://cxf.ws.bemach.com/" 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="rpc" />
         <wsdl:input name="updateEmployee">
            <soap:body namespace="http://cxf.ws.bemach.com/" use="literal" />
         </wsdl:input>
         <wsdl:output name="updateEmployeeResponse">
            <soap:body namespace="http://cxf.ws.bemach.com/" use="literal" />
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="EmployeeDataService">
      <wsdl:port binding="tns:EmployeeDataServiceSoapBinding"
         name="EmployeeDataPort">
         <soap:address location="http://localhost:7001/cxf-ws/employees" />
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

6.6 Run the Client Application

```
java -cp ./lib/cxf-ws-generated.jar;./dist/cxf-ws-client.jar com.
bemach.ws.cxf.client.EmployeeDataClient 7001
```

The output of this test will be printed on-screen as follows:

```
EmployeeDataClient 7001
Calling Employee (CXF) data service ...
URL=http://localhost:7001/cxf-ws/employees?WSDL
last=Fecello
hire=1986-06-26T00:00:00.0-04:00
last=Fecello
first=Silvester
emplNo=500001
update:true
last=New-name
first=Silvester
deleteEmployee:true
Exit!
```