Creating an Web Service from an enterprise bean using SOAP/JMS

Contents

Creating an Web Service from an enterprise bean using SOAP/JMS

Learning objectives

In this tutorial, you will learn to:

- Create a JMS server and server configuration
- Create the WSDL document named TestEJB.wsdl
- Deploy the Web service to the WebSphere Application Server
- Generate the Java client proxy
- Generate and launch the sample application

To complete this tutorial, you will need approximately 2 hours. If you decide to explore other facets of Web services while working on the tutorial, it could take longer to finish.

Prerequisites

In order to complete this tutorial end to end, you should be familiar with:

- Web services concepts
- JMS server concepts
- Basic EJB concepts

When you are ready, begin Lesson 1.1: Creating a server and server configuration for JMS.

Feedback

Related information

View the PDF version

Lesson 1.1: Creating a server and server configuration for JMS

In this lesson you will learn how to create a JMS server configuration for use with Web services.

You can import the required EAR file using the project interchange wizard:

Import the JMSEAR.ear

In the J2EE perspective notice JMSService. This project contains all the resources for EJB applications, including the enterprise bean, the remote interface, and EJB home. The JMSServiceRouter project has also been created. This is an EJB project that will contain the message router message-driven bean for the Web service.

Create a JMS Server

You can create a JMS server by following these steps:

1. From the **File** menu, select **New** → **Other** → **Server** → **Server** → **Next**.
2. Select **WebSphere v6.1 Server** as the server type. Click **Next**.
3. If this runtime has not been created in your workspace, you will be prompted to select the installation directory for the server. Click **Next**.
4. Accept the default server port and name. Click **Next**.

5. Select the JMSEAR from the list of available projects and click **Add** to target it to the server. Click **Finish**.
6. Wait for the server to start. once it has started the console will display `Server server1 open for e-business;`

Configuring the server to work with JMS

JMS settings for this server must be set in the WebSphere Application Server administrative console. The console can be launched through the Start menu on Windows, or through a Web browser at: `http://localhost:9060/ibm/console`

1. Once you have launched the console, select **Servers** → **Application Servers** to ensure that the server you created is listed.
2. In the left-hand pane, expand **Service Integration** → **Buses** and click **New**. Enter a unique name in the Name field (for example `WS_tutorial_bus`) and click **Next** and then **Finish**.
3. To associate the current server with the newly created integration bus, select the name of the bus you have just created, under **Topology** click **Bus members**. Click **Add** and select the server you want to associate the integration bus and then click **Next**. Select **File store** as the message persistence state and click **Next**. You can accept the default message store properties for this tutorial and click **Next**. If you are creating a JMS bus for your own Web service, select Help and search on ″File store settings″ for additional information about which settings are best for you. Click **Finish** to confirm.
4. Create a physical queue for the request message:
   a. In the left-hand pane, expand **Service Integration** → **Buses**. Select the bus created earlier.
   b. Under **Destination resources** click **Destinations**.
   c. On the destinations page click on **New** .
   d. Choose **Queue** as the destination type and click **Next**.
   e. Enter an identifier such as `ws_tutorial_queueJms`. Click **Next**
   f. Accept the default bus member. Click **Next**.
   g. Click **Finish** to confirm your changes, and then save your changes.
5. Assign JMS settings against the newly created queue:
   a. In the left-hand navigation panel, go to **Resources** → **JMS** → **JMS Providers**.
   b. From the Scope drop-down list select the server as your scope, and from the provider list select **Default messaging provider**.
   c. Under Additional Properties select **Queue**.
   d. Enter a name (for example `ws_tutorial_queueJms`) and JNDI name (for example`jms/ws_tutorial_queue`). In the connection pane, select the bus (`WS_tutorial_bus`) and Queue (`ws_tutorial_queueJms`) you created earlier.
   e. Click **OK** to save the changes.
6. Create a queue connection factory for the input queue:
   a. Go to **Resources** → **JMS** → **Queue connection factories**.
   b. From the Scope drop-down list select the server as your scope, and click **New**.
   c. Select the default messaging provider and click **OK**.
   d. Under General Properties enter a name (for example `WebServicesInput_QCF`) and a JNDI name (for example `jms/ws_tutorial_qcf`).
   e. In the Connection pane select the bus created earlier (`WS_tutorial_Bus`) as the bus name.
   f. Click **OK** to save the changes.
7. Create a queue connection factory for the reply queue:
   a. Go to **Resources** → **JMS** → **Queue connection factories**.
   b. From the Scope drop-down list select the server as your scope, and click **New**.
   c. Select the default messaging provider and click **OK**.

d. Under General Properties enter `WebServicesReply_QCF` as the name (you **must** use `WebServicesReply_QCF` for this field) and a JNDI name (for example`jms/WebServicesReplyQCF`). If you want to use a custom name for the reply queue connection factory you have to change the reference alias in the JMSServiceRouter deployment descriptor. This reference is set up when you run the Web service wizards. Thus if you decide to use a different JNDI name you have to go into this project and override the default setting.

e. In the Connection pane select the bus created earlier (`WS_tutorial_Bus`) as the bus name and click **OK** to save the changes.

8. A JMS activation specification is needed to bind the input queue and the listening message driven EJB:
   a. Go to **Resources** → **JMS** → **Activation specifications**.
   b. From the Scope drop-down list select the server as your scope, and click **New**.
   c. Select the default messaging provider and click **OK**.
   d. Enter a name (for example `ws_tutorial_JMSRouter`), and enter a JNDI name (for example `eis/ws_tutorial_JMSRouter`). In the Destination pane select **Queue** as the destination type, enter the destination JNDI name (`jms/ws_tutorial_queue`), and select the bus name (`WS_tutorial_Bus`).
   e. Click **OK** to save the changes.
9. Once you have added the required connection factories and queues or topics, you can stop and restart WebSphere Application Server v6.1 and return to the development workspace.

Lesson Checkpoint

Now you are ready to begin Lesson 1.2: Creating the Web service.

Feedback

Lesson 1.2: Creating the Web service

In this lesson you will learn how to create the bottom up EJB Web service.

Before you begin, you must complete Lesson 1.1: Creating a server and server configuration for JMS.

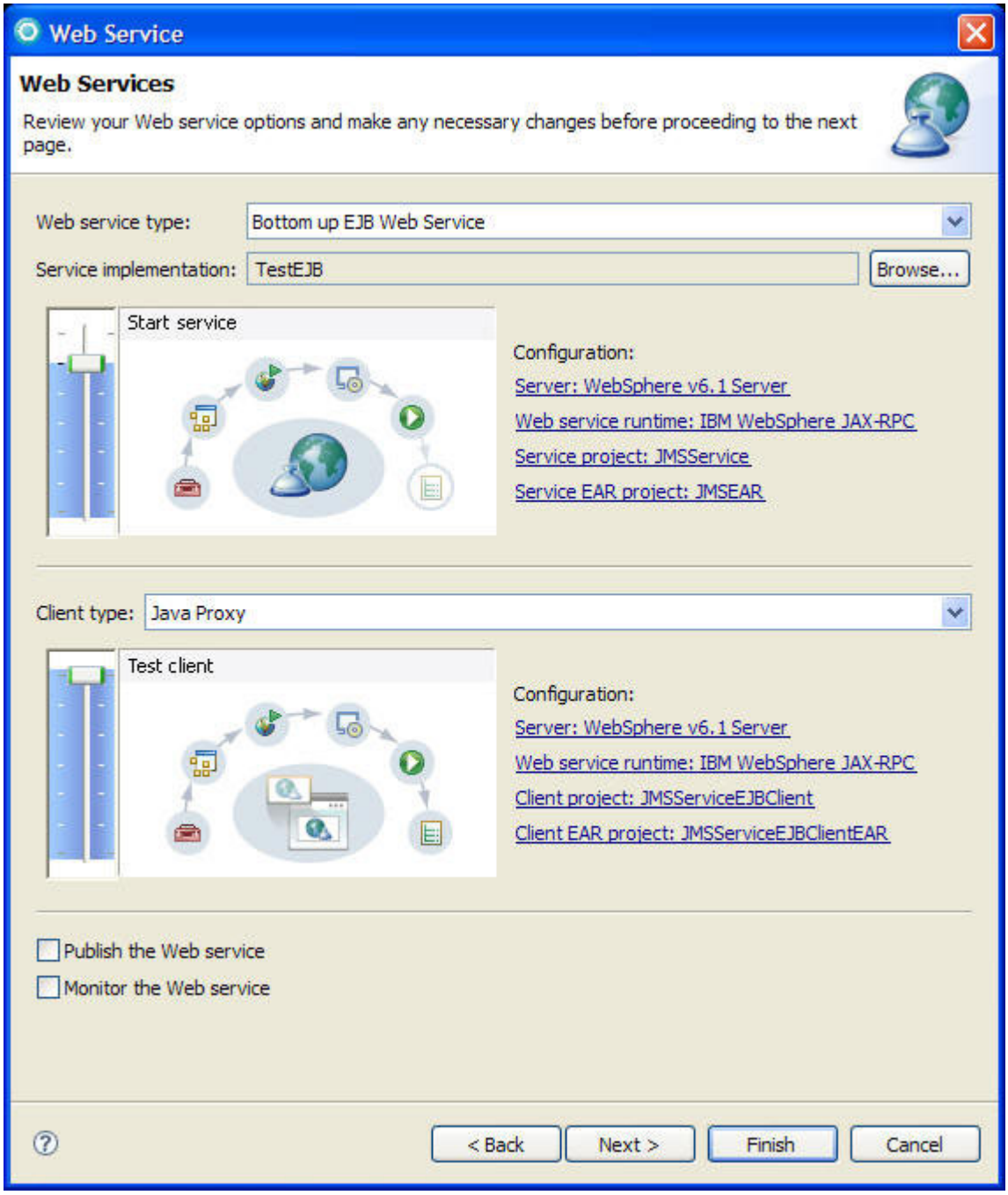
The WSDL document describes where the Web service is deployed and what operations this service provides. You can create the WSDL document, deployment descriptor file, proxy, and sample, by following these steps:

1. Click **File** → **New** → **Other**. Select **Web Services** in order to display the various Web service wizards. Select the **Web Service** wizard. Click **Next** to start the Web Service wizard.
2. In the **Web service type** field, select **Bottom up EJB Web service**.
3. Select your service implementation by clicking **Browse**. A window displays which has all the available enterprise beans listed. Select the TestEJB bean.
4. In order to create a service and a client and to start both of these on the server, move the slider for the service to the ″Start″ position, and the slider for the client to the ″Test″ position. This will create all the appropriate service and client code, projects, and router modules, associate them with the correct EAR, create the deployment code, install the EAR on the target server, and start the service. It will not test the service because the Web Services Explorer does not support Web services over JMS.
5. Ensure that the following server-side options are selected:
   - Server: WebSphere v6.1 Server
   - Runtime: IBM WebSphere JAX-RPC runtime environment
   - Service project: JMSService
   - Service EAR project: JMSEAR

   Ensure that the following client-side options are selected:

- Server: WebSphere v6.1 Server
- Runtime: IBM WebSphere JAX-RPC runtime environment
- Client project: JMSServiceEJBClient
- Client EAR project: JMSServiceEJBClientEAR

The projects which do not already exist will be created for you by the wizard. When you have selected all your options it should look similar to the following:

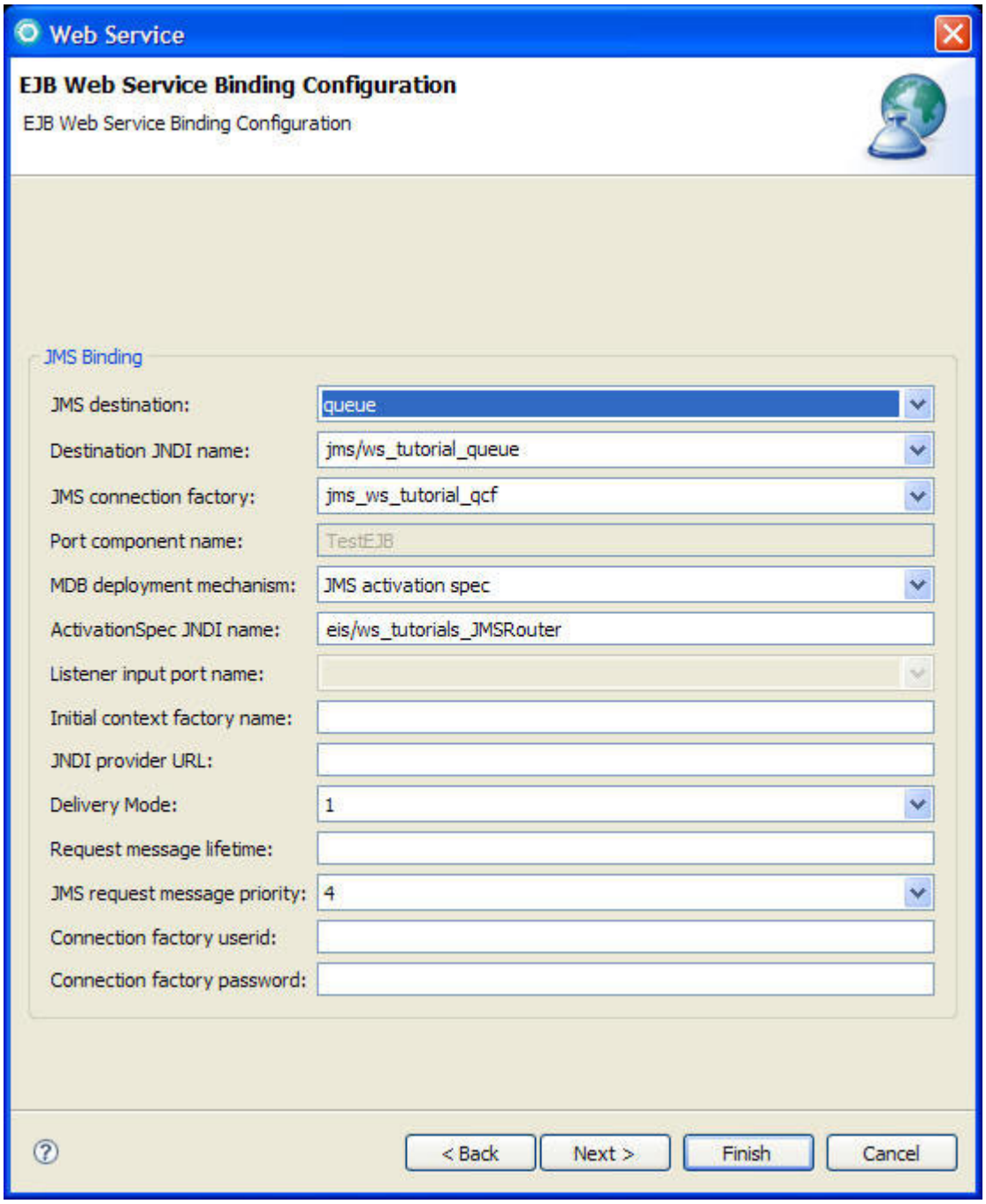


6. On the Web service EJB configuration page, select **JMS** as your transport method and clear the HTTP check box if it is selected. The JMSServiceRouter router project that you imported as part of the JMSEAR should be selected. Click **Next**. Because JMS is WS-I non-compliant, unless you have set

your WS-I compliance settings to **Ignore**, an error message displays warning you of the incompliance. If you click **Details** the reason for the warning message is shown. You can safely ignore this warning; click **Ignore**.

7. On the EJB Web service binding configuration page you will have to manually enter the following values to match the queues and connection factories created in the previous lesson:
   - Ensure that queue is selected as the JMS destination. This tutorial will not work for topics.
   - `jms/ws_tutorial_queue` as the Destination JNDI Name
   - `jms/ws_tutorial_qcf` as the JMS Connection Factory
   - The name of the port component is the target service name, therefore, TestEJB will be used as target service name.
   - `eis/ws_tutorial_JMSRouter` as the ActivationSpec JNDI Name

   The completed page should look similar to the following:



8. In the Web Service Java Bean Identity page of the wizard you can specify your Web service URI, scope, and the names of the generated files. You can also select the methods that will be included in your Web service, the encoding style, and configure security for your Web service. Click **Next** to accept the default values.

   Important: The Uniform Resource Identifier (URI) for your Web service is automatically generated by the wizard from the artifact you selected to turn into a Web service. The default base URI http://tempuri.org/ is used to construct a URI without any unique association to an entity. The host name tempuri comes from the WSDL specification and stands for temporary URI. Use the default base URI when you do not want to make the URI globally unique. It is not recommended to use http://tempuri.org/ as the base for stable fixed entities.

9. The Web Service Proxy page lists the location where the proxy code will be generated. The client proxy provides a remote procedure call interface to your Web service. Do not enable security for the generated proxy. Click **Next**.
10. Use the Web Service Client Test page to select the following options:
    - Select to generate a sample Web service sample JSP as your test facility.
    - Select the folder where the JSP will be located, and ensure all methods are included in the JSP.
    - Select **Run test on server** to start the server for you automatically.

    Click **Finish**.
11. The proxy JSP is launched in a Web browser at the following URL: `http://localhost:9080/JMSClient/sample/TestEJB/TestClient.jsp` You can use this sample application to test the Web service by selecting a method, entering a value for the method, and clicking **Invoke**. The result of the method - an echo of the string you entered in the text field - will display in the results pane.

Lesson Checkpoint

Finish your tutorial by reviewing the materials in the Summary

Feedback

Summary

You have just completed development and testing of a Web service that uses JMS transport.

In this tutorial, you used the Web service wizard to generate Web services description language (WSDL) document named TestEJB.wsdl from the TestEJB enterprise bean. You then deployed the Web service to the WebSphere Application Server and tested it through sample Web service JSPs.

When you have completed the tutorial and no longer require the Web projects for testing, right-click each project and select **Delete** to remove the projects from your workspace.

Additional resources

For more information about Web services, please consult the online help (**Help → Help Contents**).

Related information

developerWorks Web services
For more in-depth technical articles on Web services.