Introduction

Time required

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

Learning objectives

In this tutorial, you will learn to:

- Create a router project for the EJB Web service
- 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

Now you are ready to begin Exercise 1.1: Importing the required resources and building the required projects.

Exercise 1.1: Importing the required resources and building the required projects

Enabling Web service capabilities

To enable the capabilities required for Web services development:

1. On the Welcome page, check to see if Web services are enabled by looking for the Web services icon in the lower right-hand corner:  . If the Welcome page has been closed, you can reopen it from the Help menu.
2. If Web services are not enabled, select the icon in the lower right corner that looks like a person. This will generate a list of capabilities that you can select from.
3. Select the Web services icon in the top left corner:

The tools used in Web services development are now enabled.

Importing the EAR file

To import the EAR file, follow these steps:

1. From the **File** menu, click **Import** to open the Import wizard, then select **EAR file** and click **Next**.
2. Enter the following file in the **EAR File** field. You can click **Browse** to locate and select the file: `com.ibm.etools.webservice.buejb.tutorial.doc\resources\JMSEAR.ear`. This plugin is located in the directory where you installed the Rational Developer product in the `rad\eclipse\plugins` folder.
3. In the **EAR Project** field, type `JMSEAR` if it is not already prefilled. This is the Enterprise Application project which you will import the EAR file into. The project will be created when you complete the wizard.
4. Ensure that the target server selected is WebSphere Application Server v6.0.

5. Click **Finish**. The Enterprise Application project, JMSEAR and the EJB Module, JMSService are created.

In the J2EE perspective under the EJB Projects folder, 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.

Now you are ready to begin Exercise 1.2: Creating a server and server configuration for JMS.

Exercise 1.2: Creating a server and server configuration for JMS

Before you begin, you must complete Exercise 1.1: Importing the required resources and building the required projects.

To create a JMS server:

1. From the **File** menu, select **New** > **Other** > **Server** > **Server** > **Next**.
2. Select **WebSphere v6.0 Server** as the server type. Click **Next**.
3. Accept the default server port and name. Click **Next**.
4. Select the JMSEAR from the list of available projects and click **Add** to target it to the server. Click **Finish**.
5. 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 **OK**.
3. To associate the current server with the newly created integration bus, select the name of the bus you have just created, under **Additional Properties** click **Bus members**. Click **Add** and select the server you want to associate the integration bus and then click **Next**. Click **Finish** to confirm and click **Save** to save the changes.
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 **Additional Properties** click **Destinations**.
    c. Click on **New** and choose **Queue** as the destination type. Enter an identifier such as `ws_tutorial_queueJms`. Click **Next**.
    d. Accept the default bus member. Click **Next**.
    e. 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 Providers > Default Messaging**.
    b. Under **Destinations**, click **JMS Queue**, and then click on **New**.
    c. 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.
    d. Click **OK** and save the changes.
6. Create a queue connection factory for the input queue:
    a. Go to **Resources > JMS Providers > Default Messaging**.
    b. Under **Connection Factories** select **JMS queue connection factory**, click **New** and enter a name (for example `WebServicesInput_QCF`) and a JNDI name (for example

`jms/ws_tutorial_qcf`).
   c. Select the bus created earlier (`WS_tutorial_Bus`) as the bus name.
   d. Click **OK** and save the changes.
7. Create a queue connection factory for the reply queue:
   a. Go to **Resources > JMS Providers > Default Messaging**.
   b. Under **JMS queue connection factory** click **New** and enter a name (for example `WebServicesReply_QCF`) and a JNDI name (for example `jms/WebServicesReplyQCF`).
   c. Select the bus created earlier (`WS_tutorial_Bus`) as the bus name and click **OK** and 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 Providers > Default Messaging**.
   b. Under **Activation specifications**, click **JMS activation specification**, click **New**, and enter a name (for example `ws_tutorial_JMSRouter`), enter a JNDI name (for example `eis/ws_tutorial_JMSRouter`), select **Queue** as the destination type, enter the destination JNDI name (`jms/ws_tutorial_queue`), and select the bus name (`WS_tutorial_Bus`).
   c. Click **OK** and 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, and return to the Rational Developer product's workspace.

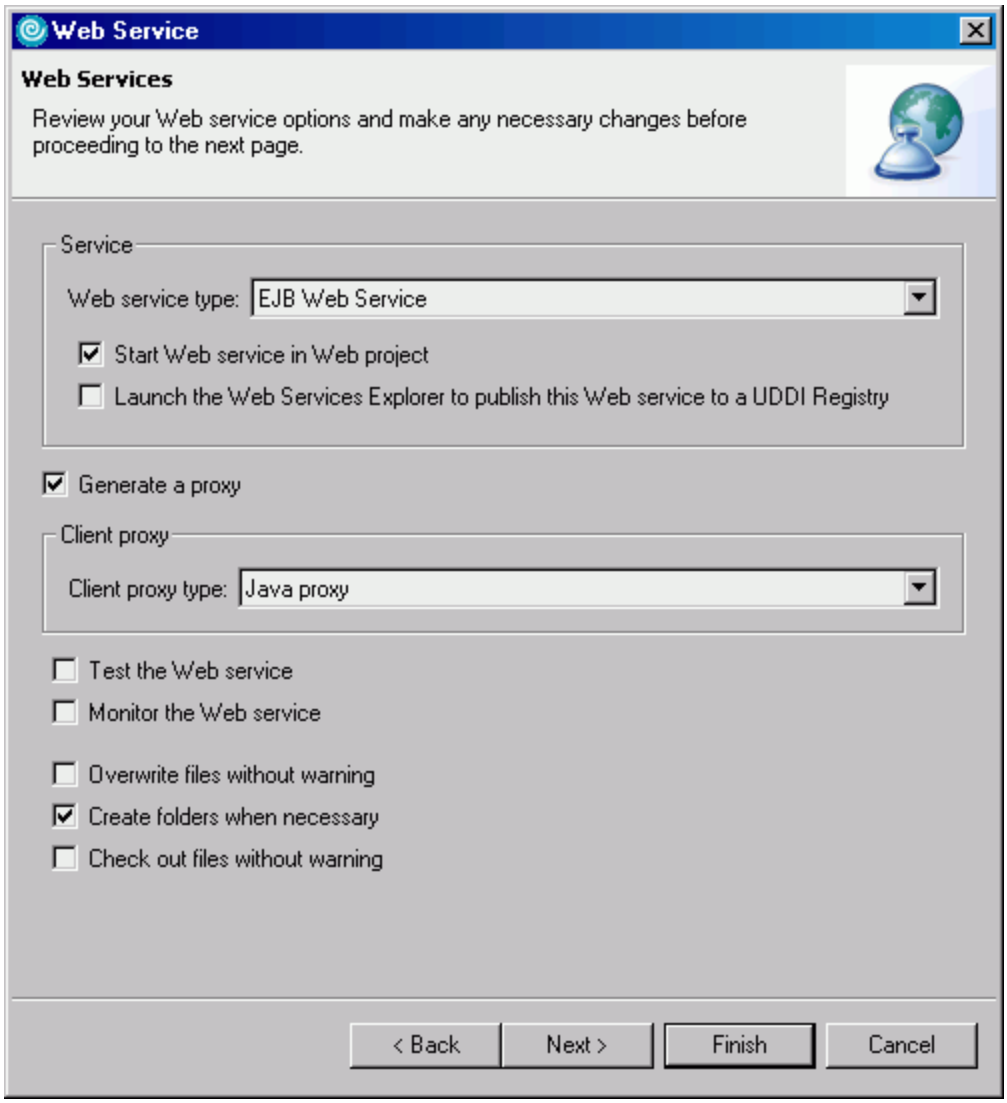
Now you are ready to begin Exercise 1.3: Creating the Web service.

Exercise 1.3: Creating the Web service

Before you begin, you must complete Exercise 1.2: Creating a server and server configuration for JMS.

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

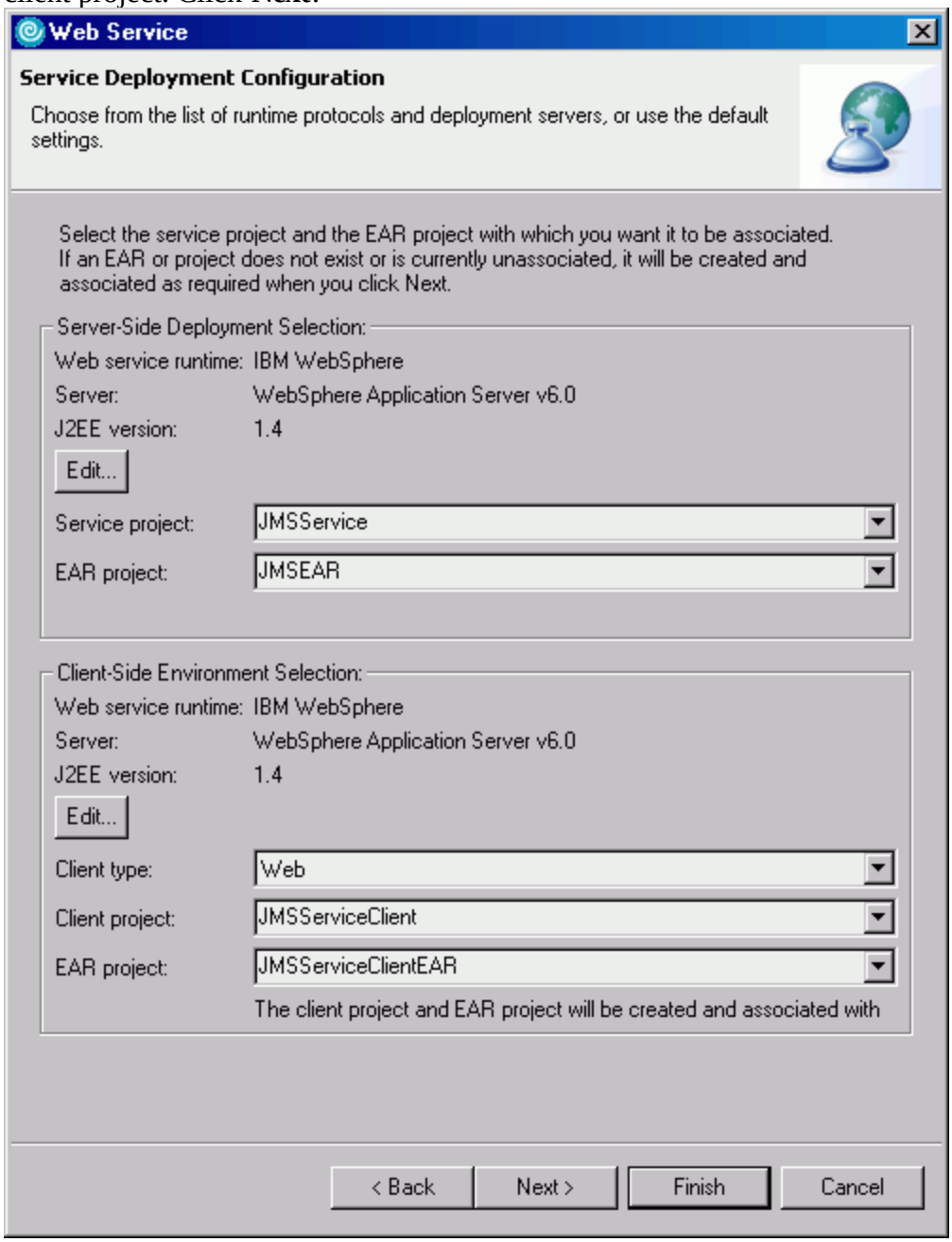
1. Click **File > New > Other**. Select **Web Services** in order to display the various Web service wizards. Select the **Web Service** wizard. Click **Next** to start the Web Service wizard.
2. In the **Web service type** field, ensure **EJB Web service** is displayed and that the following check boxes are selected:
   - Start Web services in Web project
   - Generate a proxy
   - Create folders when necessary



   Do not select Test the Web Service; the Web Services Explorer which is used to test Web services does not support SOAP over JMS.
3. On the Object selection page the JMSEAR should be prefilled, and the TestEJB bean should be

listed in the EJB Bean table. Select **TestEJB** and click **Next**.

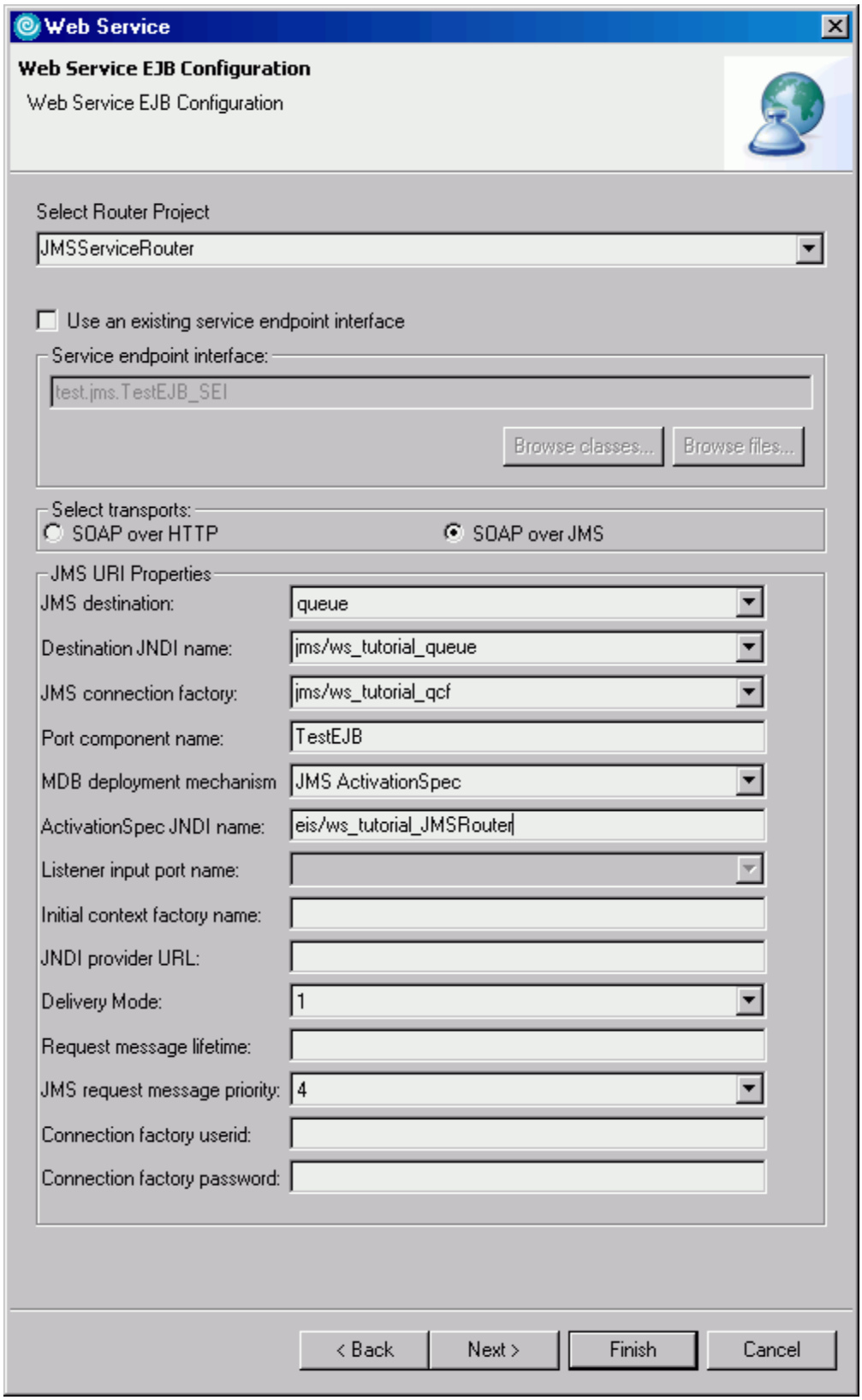
4. The Web Service Deployment Configurations page allows you to select from supported run-time protocols and deployment servers. You will use the default run-time environment, IBM WebSphere v6, and the server that you created in the previous task, WebSphere v6 Server. Ensure that the EJB, and client projects (where the Client type is Web) selected match those in the picture below. Since the wizard creates the client project for you, you can manually enter the name of the client project. Click **Next**.



5. On the Web service EJB configuration page, the Router project that you imported as part of the JMSEAR should be selected. Under **Select Transports**, select **SOAP over JMS**. Most of the required information in this section should be prefilled. You will have to manually enter the following values:
   - Ensure that `queue` is selected as the JMS destination. This sample will not work for topics.
   - `jms/ws_tutorial_queue` as the Destination JNDI Name
   - `jms/ws_tutorial_qcf` as the JMS Connection Factory

- `TestEJB` as the name of the port component to which the request will be dispatched. 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:



6. 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.

7. 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**.
8. In the Web Service Proxy page, ensure that **Generate a proxy** is selected. The client proxy provides a remote procedure call interface to your Web service. Do not enable security for the generated proxy. Click **Next**.
9. 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**.
10. 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.

Finish your tutorial by reviewing the materials in the Summary.

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.

For more information about Web services, WSDL, and SOAP, please consult the online help for WebSphere Studio (**Help > Help Contents**). For more in-depth technical articles on Web services, consult www.ibm.com/developerworks/webservices