

# **Oracle® Java CAPS Java EE Service Engine Tutorial**

Copyright © 2009, 2011, Oracle and/or its affiliates. All rights reserved.

#### **License Restrictions Warranty/Consequential Damages Disclaimer**

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

#### **Warranty Disclaimer**

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

#### **Restricted Rights Notice**

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

##### **U.S. GOVERNMENT RIGHTS**

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

#### **Hazardous Applications Notice**

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

#### **Trademark Notice**

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group in the United States and other countries.

#### **Third Party Content, Products, and Services Disclaimer**

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

# Contents

---

<b>Using the Java EE Service Engine to Create a Composite Application</b> .....	5
Tutorial Requirements .....	5
Prerequisites .....	6
Tutorial Scenario .....	6
Starting the GlassFish Application Server .....	6
▼ To Check the Status of the GlassFish Application Server in the NetBeans IDE .....	6
▼ To Register the GlassFish Application Server with the NetBeans IDE .....	7
▼ To Start the GlassFish Application Server in the NetBeans IDE .....	7
Enabling the JBI Framework .....	8
Creating an EJB Module Project .....	8
▼ To Create the EJB Module Project .....	9
▼ To Create a WSDL Document .....	11
▼ To Create a Web Service from WSDL .....	15
▼ To Clean and Build the EJB Module Project .....	18
Creating a Composite Application Project .....	18
▼ To Create a Composite Application Project .....	18
Building and Deploying the Composite Application Project .....	21
▼ To Build and Deploy the Composite Application .....	21
Testing the Composite Application .....	22
▼ To Add a Test Case .....	22
▼ To Run the Test Case .....	25
Summary .....	26



# Using the Java EE Service Engine to Create a Composite Application

---

This tutorial is designed to showcase the Java EE Service Engine's functionality. This example illustrates how the Java EE Service Engine implements a Web Service from a WSDL file accessing it through the HTTP Binding Component.

We will create a Composite Application so that Java EE SE will be used in the service and we will then run a test case to test our service. This tutorial also uses the features of the HTTP Binding Component.

## What You Need to Know

These topics provide information you need to know before you start the tutorial:

- [“Tutorial Requirements” on page 5](#)
- [“Tutorial Scenario” on page 6](#)

## What You Need to Do

This tutorial includes instructions on how to perform these tasks:

- [“Starting the GlassFish Application Server” on page 6](#)
- [“Enabling the JBI Framework” on page 8](#)
- [“Creating a Composite Application Project” on page 18](#)
- [“Building and Deploying the Composite Application Project” on page 21](#)
- [“Testing the Composite Application” on page 22](#)

## Tutorial Requirements

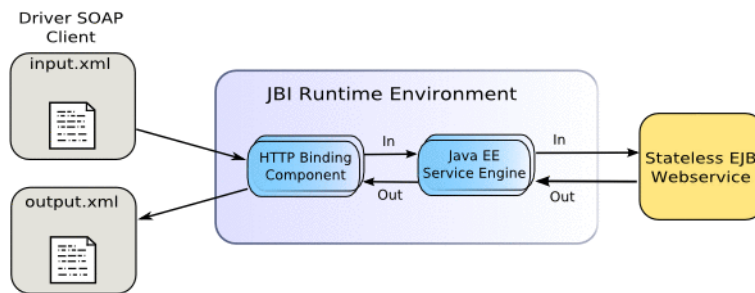
Before you proceed, make sure you review the requirements in this section.

## Prerequisites

This tutorial assumes that you have some basic programming experience with the Java language and platform, and knowledge of the NetBeans IDE.

## Tutorial Scenario

The following sample illustrates how Java EE Service Engine implements a Web Service from WSDL.



## Starting the GlassFish Application Server

The Java EE Service Engine starts together with GlassFish. Before deploying and performing test runs of a Composite Application project in the NetBeans IDE, make sure that the GlassFish Application Server is started.

### ▼ To Check the Status of the GlassFish Application Server in the NetBeans IDE

- 1 If the Services window is not visible, on the NetBeans IDE main menu choose **Window** → **Services**.
- 2 In the Services window, expand the **Servers** node.

The Servers node should contain a GlassFish V2 subnode. If the GlassFish V2 node is not visible, see [“To Register the GlassFish Application Server with the NetBeans IDE”](#) on page 7.

If a green arrow icon appears on the GlassFish V2 node, the server is running. If no green arrow icon appears, see [“To Start the GlassFish Application Server in the NetBeans IDE”](#) on page 7.

## ▼ To Register the GlassFish Application Server with the NetBeans IDE

- 1 If the Services window is not visible, on the NetBeans IDE main menu choose **Window** → **Services**.
- 2 In the Services window, right-click the **Servers** node and choose **Add Server** from the pop-up menu.  
The Add Server Instance dialog box opens.
- 3 In the **Choose Server** page of the dialog box, select **GlassFish V2** from the **Server** drop-down list.
- 4 If you want to change the server name that the IDE uses to identify the server, type it in the **Name** field.
- 5 Click **Next** .  
The Platform Location Folder page opens.
- 6 In the **Platform Location** field, click **Browse** and select the installation location of the application server.
- 7 Select the **Register Local Default Domain** option and click **Next**.
- 8 Type the user name and password for the domain's administrator.  
If you accepted the default values during the installation, the user name is `admin`, the password is specified during installation.
- 9 Click **Finish**.

## ▼ To Start the GlassFish Application Server in the NetBeans IDE

- 1 On the NetBeans IDE page, in the Services window, right-click the **GlassFish V2** node and choose **Start**.
- 2 Wait until the following message appears in the Output window:  
"Application server startup complete."  
When the server is running, the IDE displays a green arrow icon on the GlassFish V2 node.

The Java EE Service Engine is represented as `sun-javaee-engine` in the Services window of the IDE, under the `GlassFish V2` → `JBI` → `Service Engines` nodes.

## Enabling the JBI Framework

In some cases, you might have to enable the JBI framework to deploy a Java EE Service Engine component. The following command enables the JBI framework:

```
asadmin enable --user adminuser JBIFramework
```

## Creating an EJB Module Project

In this section you will create a new EJB module project called `Hello`. You will also create a WSDL document, a web service and then clean and build the EJB module project.

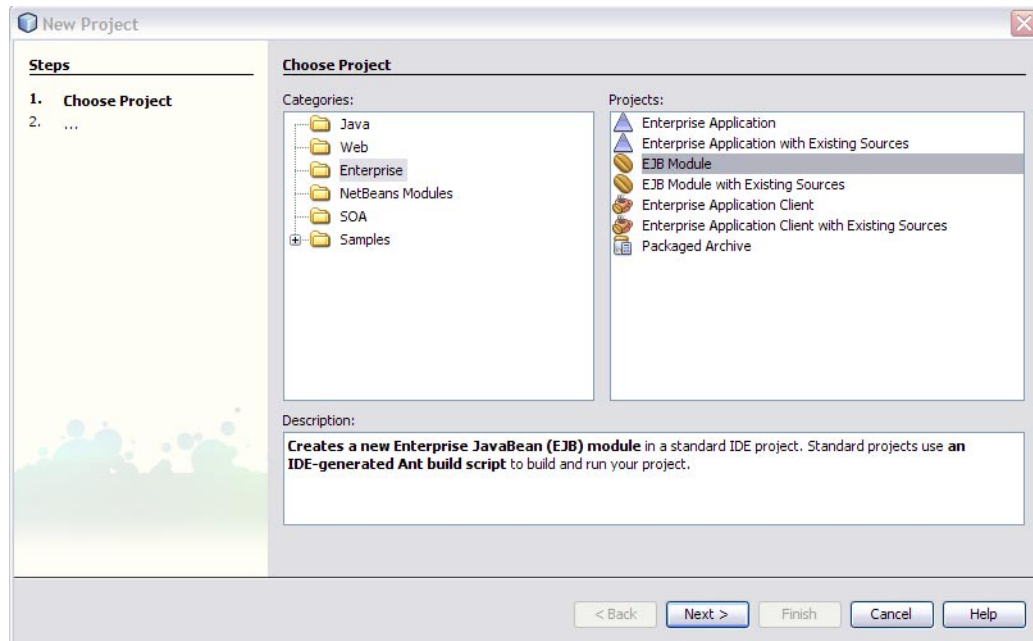


## ▼ To Create the EJB Module Project

- 1 From the NetBeans IDE's main menu, choose **File** → **New Project**.

The New Project wizard opens.

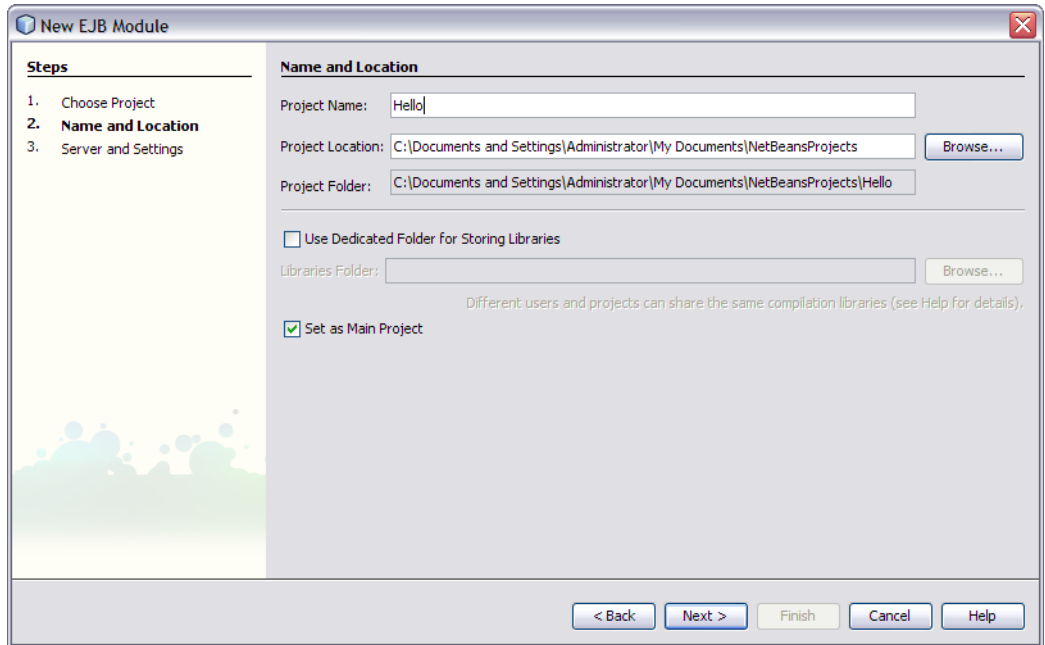
- 2 In the **Categories** list, select the **Enterprise** node and in the **Projects** list select **EJB Module**.



- 3 Click **Next**.

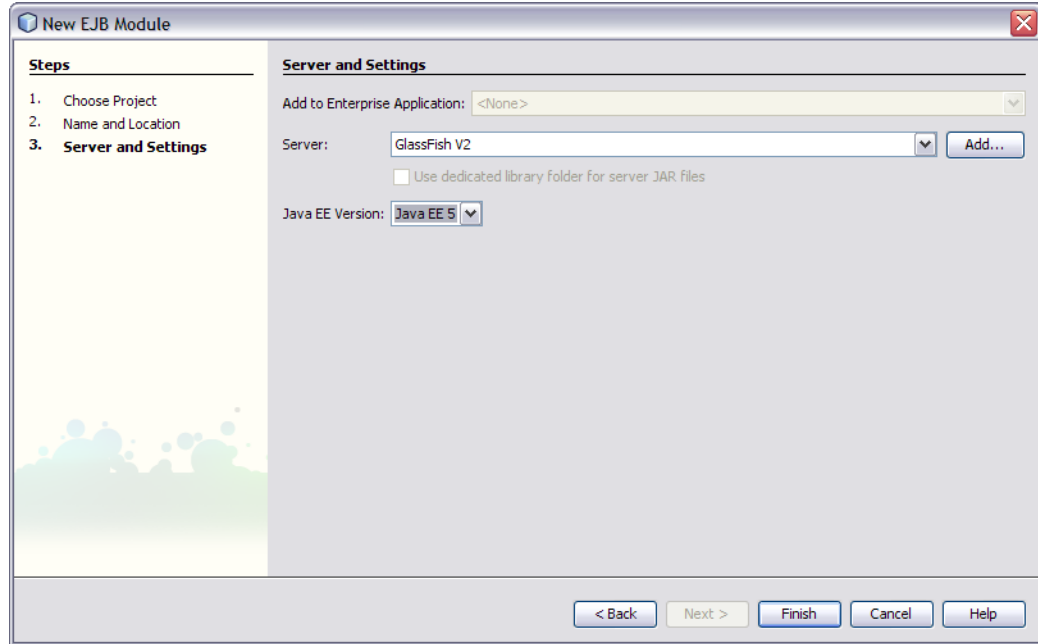
**4 In the Project Name field, type Hello.**

You may choose to change the Project Location or use the default location.



**5 Click Next.**

- 6 In the Server field select your server and in Java EE Version field select the appropriate version.



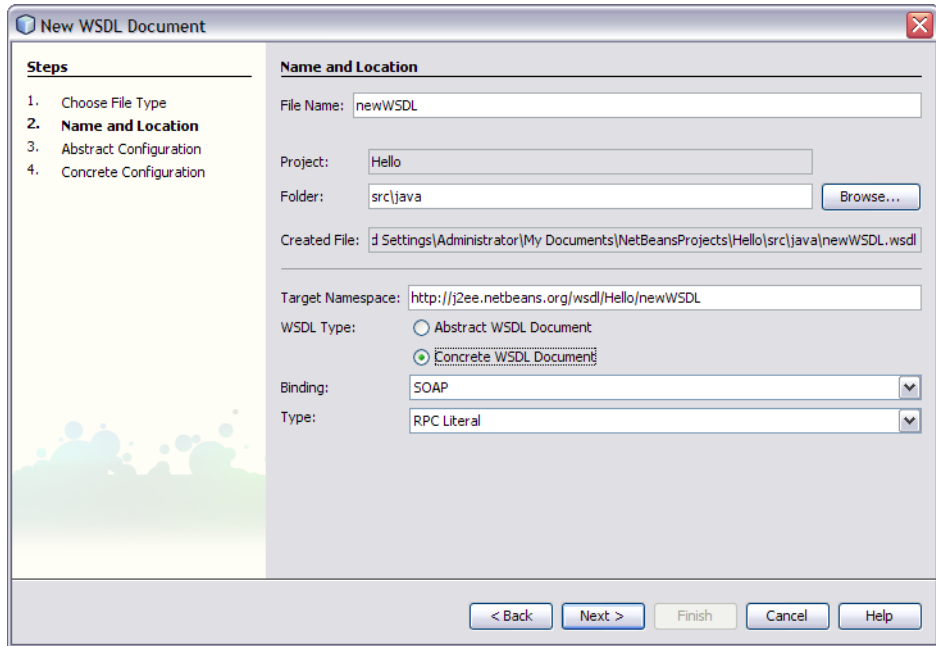
- 7 Click **Finish**.

The Projects window now contains a node for a EJB Module project called Hello.

## ▼ To Create a WSDL Document

- 1 In the Projects window of the IDE, right-click the Hello node and choose **New** → **WSDL Document**.
- 2 In the File Name field type HelloWSDL.
- 3 In the WSDL Type, select the **Concrete WSDL Document** option.

- 4 In the Binding field, select SOAP and in the Type field, select RPC Literal.



- 5 Click Next.
- 6 On the Abstract Configuration page, in Input, under the Message Part Name double-click Part1 and change the value to in and press Return.

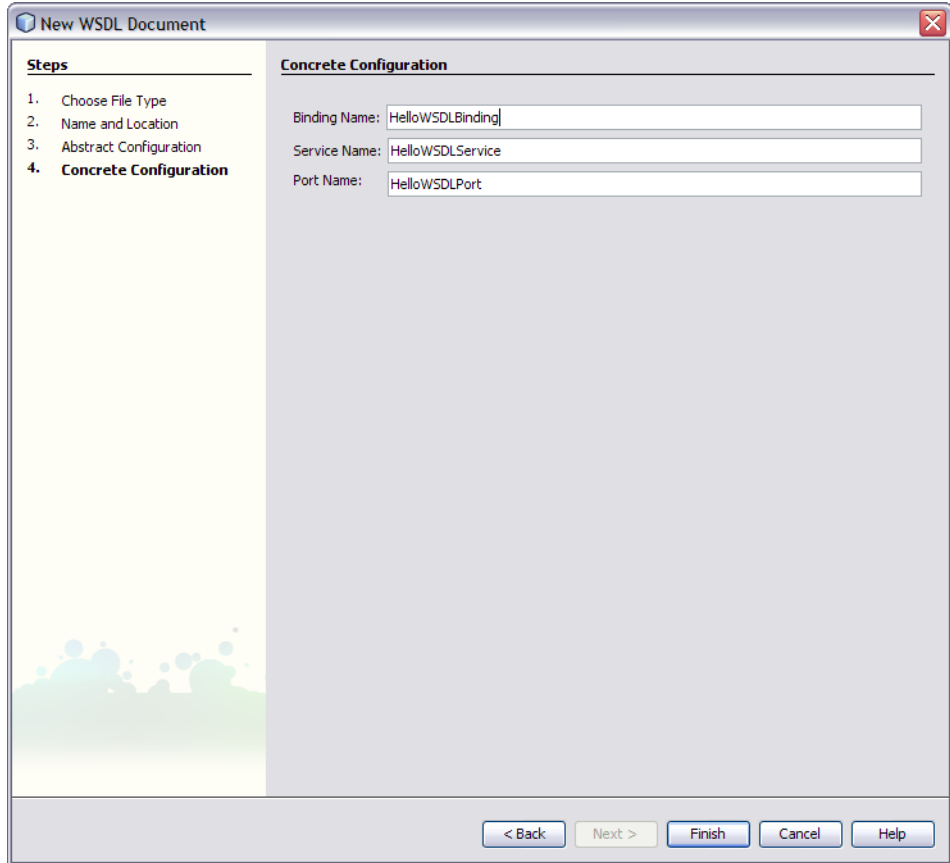
- 7 Doing the same way in Output, change Part2 to out under Message Part Name and press Return.

The screenshot shows the 'New WSDL Document' dialog box with the 'Abstract Configuration' tab selected. The 'Steps' pane on the left lists four steps: 1. Choose File Type, 2. Name and Location, 3. Abstract Configuration (highlighted), and 4. Concrete Configuration. The main configuration area contains the following fields and controls:

- Port Type Name: newWSDLPortType
- Operation Name: newWSDLOperation
- Operation Type: Request-Response Operation (dropdown menu)
- Input: A table with two columns: 'Message Part Name' and 'Element Or Type'. The first row contains 'in' and 'xsd:string'. Below the table are 'Add' and 'Remove' buttons.
- Output: A table with two columns: 'Message Part Name' and 'Element Or Type'. The first row contains 'out' and 'xsd:string'. Below the table are 'Add' and 'Remove' buttons.
- Fault: A table with two columns: 'Message Part Name' and 'Element Or Type'. It is currently empty. Below the table are 'Add' and 'Remove' buttons.
- At the bottom, there is a checked checkbox labeled 'Generate partnerlinktype automatically.'
- At the very bottom, there are navigation buttons: '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

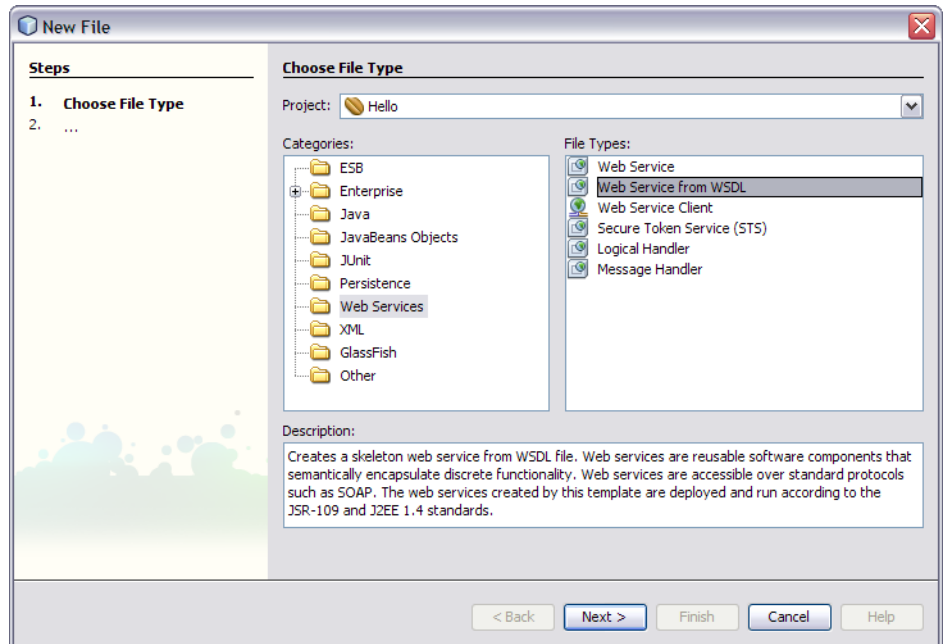
- 8 Click Next.

9 Choose the defaults and click **Finish** on the **Concrete Configuration** page.



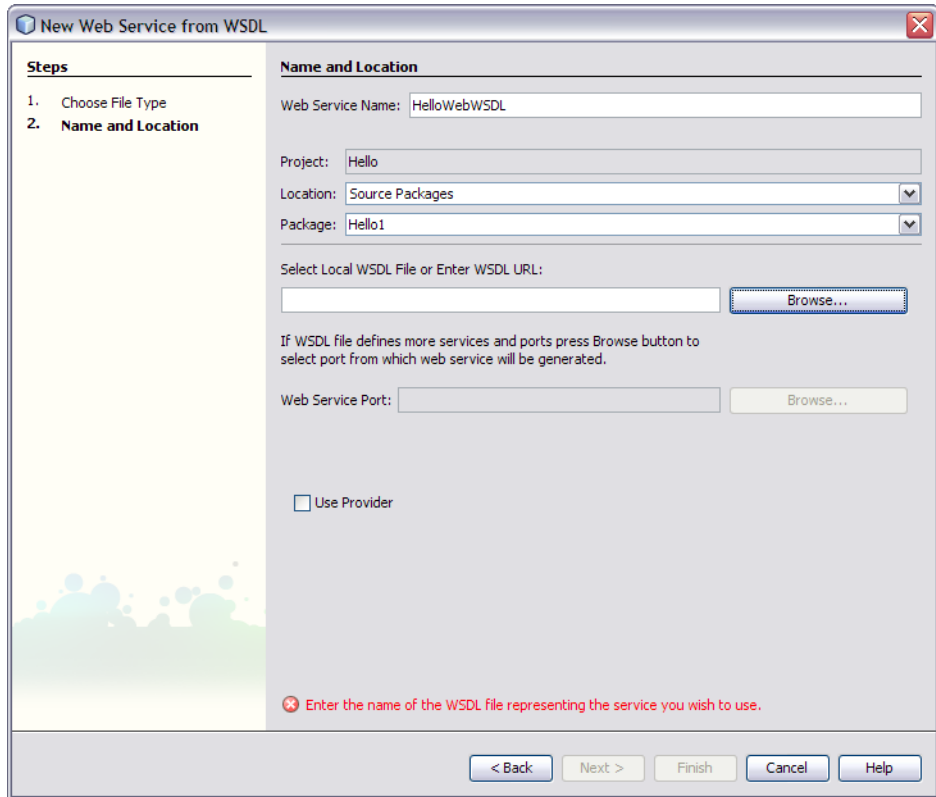
## ▼ To Create a Web Service from WSDL

- 1 In the Projects window of the IDE, right-click the Hello node and choose New → Other.
- 2 In the Categories list select Web Services and in File Types select Web Service from WSDL.



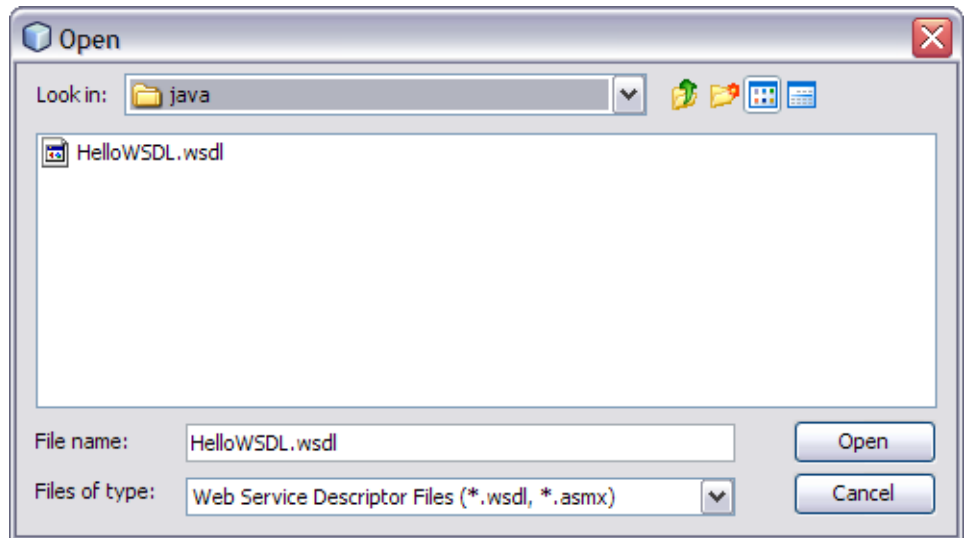
- 3 Click Next.

4 Type the Web Service Name as HelloWebWSDL and the Package name as Hello1.



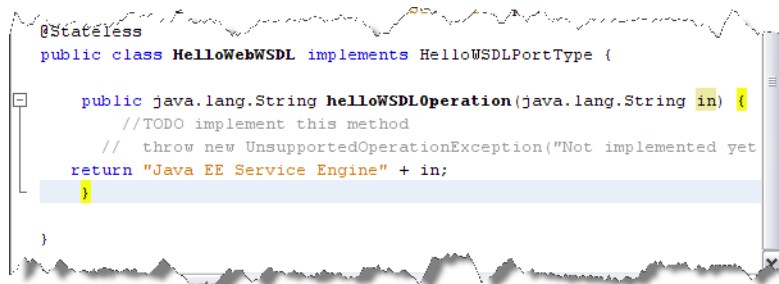


- 5 Click the Browse button to select local WSDL file or the WSDL URL and then click Open.



**Note** – This WSDL File or the WSDL URL is located in the NetBeans Projects folder. For example: C:\Documents and Settings\Administrator\My Documents\NetBeansProjects\Hello\src\java This path is valid if you have used the default project location while creating the EJB module project.

- 6 Click Finish.
- 7 Click the Source button and add the following line to the public class.  
return "Java EE Service Engine" + in;



- 8 From the NetBeans IDE toolbar click Save All button.

## ▼ To Clean and Build the EJB Module Project

- In the Projects window right-click the `Hello` node and choose `Clean and Build`.

When the build is complete the Output window reports `BUILD SUCCESSFUL`.

If the Output window is not visible, choose `Window` → `Output` → `Output`.

## Creating a Composite Application Project

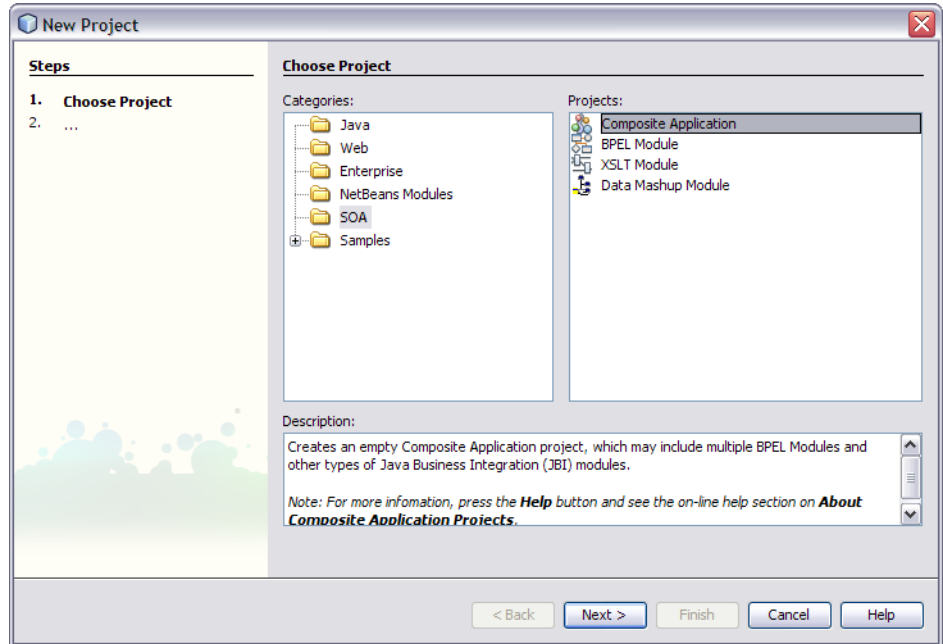
A EJB Module project is not directly deployable. You must first add a EJB Module project, as a JBI module, to a Composite Application project. You can then deploy the Composite Application project. Deploying the project makes the service assembly available to the application server and enables its service units to run.

## ▼ To Create a Composite Application Project

- 1 From the NetBeans IDE's main menu, choose `File` → `New Project`.

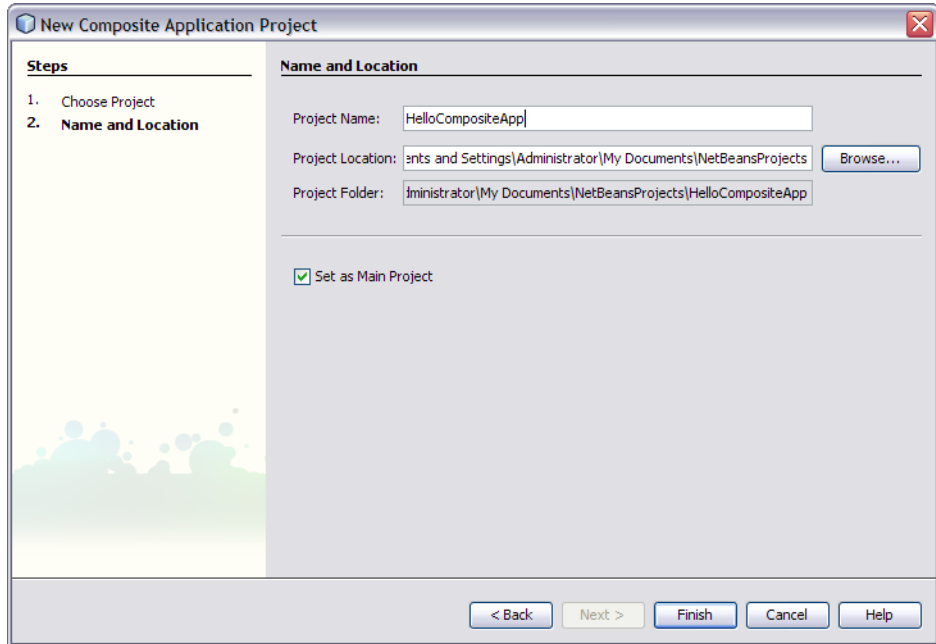
The New Project wizard opens.

- 
- 2 In the Categories list, select the SOA node and in the Projects list select Composite Application



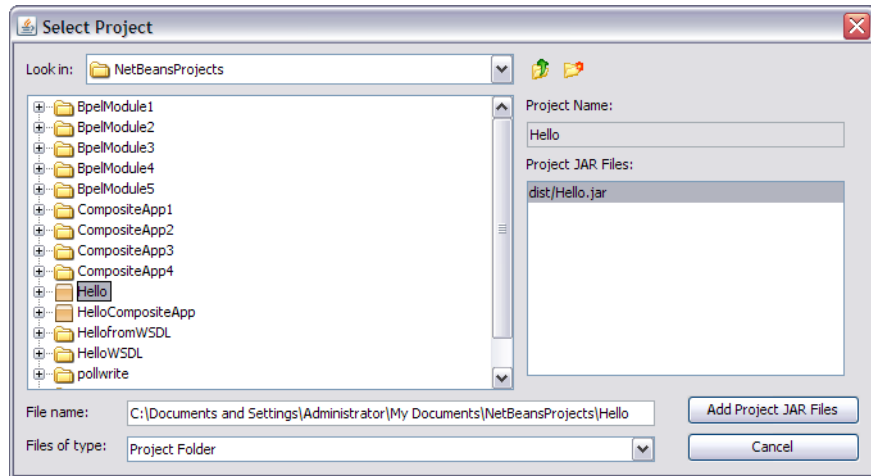
- 
- 
- 3 Click Next.

- 4 In the Name and Location page, change the project name to `HelloCompositeApp`, and specify the location of project files or just use the default location.



- 5 Leave the Set as Main Project option selected and click Finish.
- 6 To add the EJB Module as a JBI module to the Composite Application project, right-click `HelloCompositeApp` and choose Add JBI Module.  
The Select Project dialog box opens.

- 7 Select the `Hello` project you created earlier and select `dist/Hello.jar` under Project JAR Files and click **Add Project JAR Files** button.



The Select Project dialog box closes and the `Hello.jar` file is added to the JBI Modules node of the `HelloCompositeApp` Composite Application in the Projects window.

## Building and Deploying the Composite Application Project

Deploying the project makes the service assembly available to the application server, which allows its service units to run. Before you deploy the EJB Module project, you must add the JBI module to the deployment project.

### ▼ To Build and Deploy the Composite Application

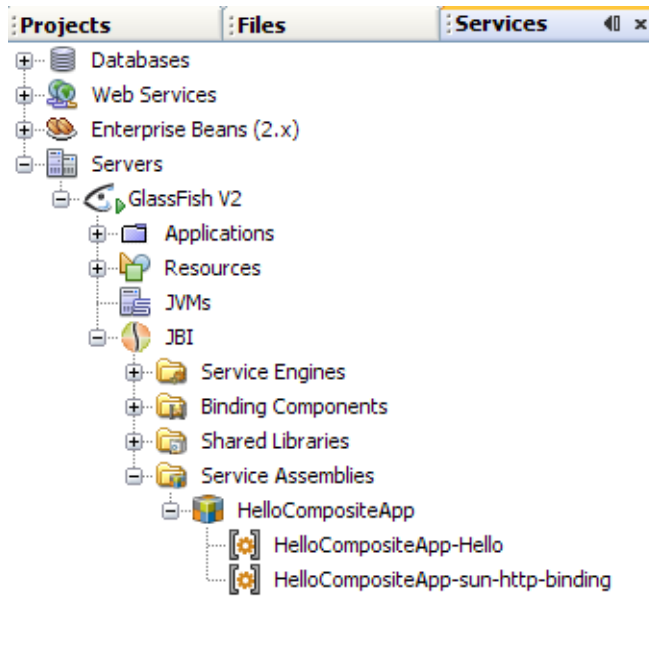
- 1 Right-click the `HelloCompositeApp` node, and choose **Build** from the pop-up menu.

When the build is complete the Output window reports **BUILD SUCCESSFUL**.

- 2 Right-click the `HelloCompositeApp` node, and choose **Deploy**.

Deployment is successful when you see the **BUILD SUCCESSFUL** message in the `build.xml` (run) tab of the Output window.

- 3 Open the Services window of the IDE and expand Servers → GlassFish V2 → JBI → Service Assemblies to see your new deployed Service Assembly.



If you do not see the deployed project, right-click the Service Assemblies node and choose Refresh.

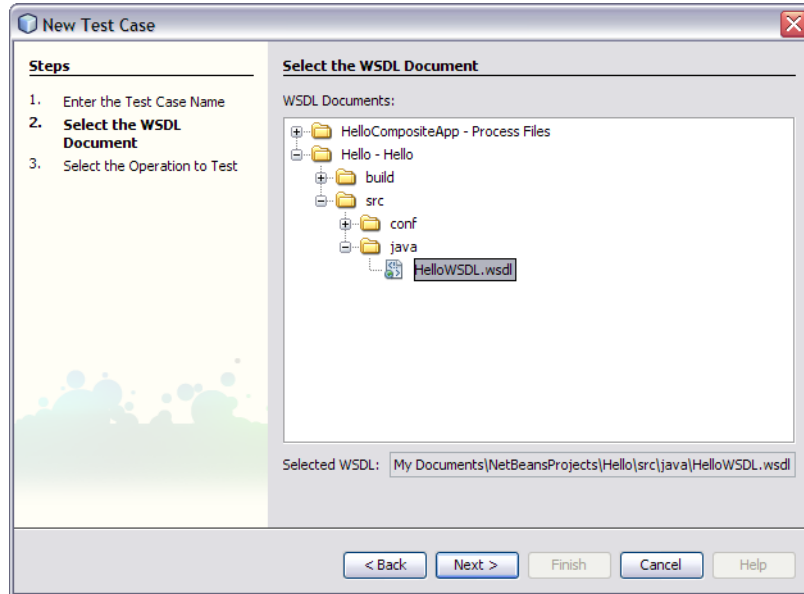
## Testing the Composite Application

You can enhance the Composite Application project by adding test cases, binding to the operation, supplying input, and then using the Tester. You have to first add a test case and then run the test case.

### ▼ To Add a Test Case

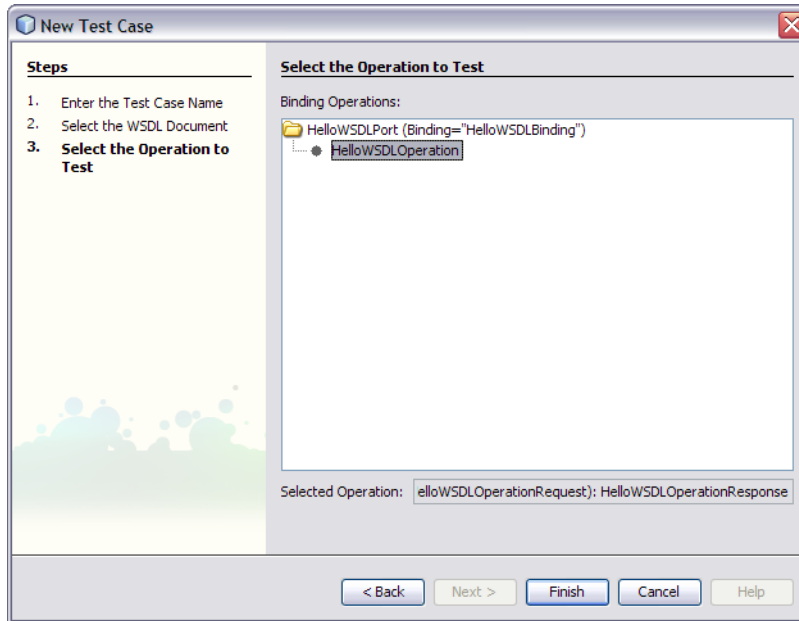
- 1 In the Projects window of the IDE, expand the HelloCompositeApp project node, right-click the Test node, and choose New Test Case from the pop-up menu.  
The New Test Case wizard opens.
- 2 Accept the default test case name, TestCase1, and click Next.

- From the Select the WSDL Document page, expand the Hello - Hello, src , java nodes, and select HelloWSDL.wSDL.



- Click Next.

- From the Select the Operation to Test page, select HelloWSDLOperation and click Finish.



A new TestCase1 node is added under the project's Test node in the Projects window, containing two subnodes, Input and Output.

The Source Editor appears containing the Input file, Input.xml

---

**Note** – If the Source Editor does not contain a tab for Input.xml, double-click the Input node in the Projects window to open the file.

---

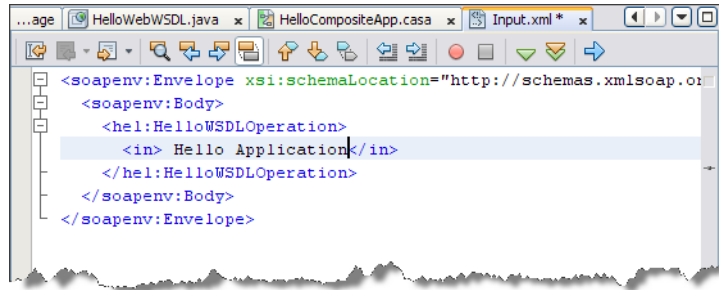
- From the Input.xml tab of the Source Editor, locate the line:

```
<in?string?</ in>
```



## 7 Replace the string `?string?` with `Hello Application`, so that it appears as:

```
<in> Hello Application</ in>
```



## 8 From the NetBeans IDE toolbar, click the Save All button.

## ▼ To Run the Test Case

### 1 In the Projects window, expand the `HelloCompositeApp` → `Test` → `TestCase1` nodes, right-click `TestCase1` for the specific test case, and then choose `Run`.

In the Output window the first run correctly reports that it failed. This happens because the output produced does not match the (empty) `Output.xml` file, and the file's null content is replaced with the output of the first run.

### 2 When the `Overwrite Empty Output` dialog box appears, click `Yes` to accept new output.

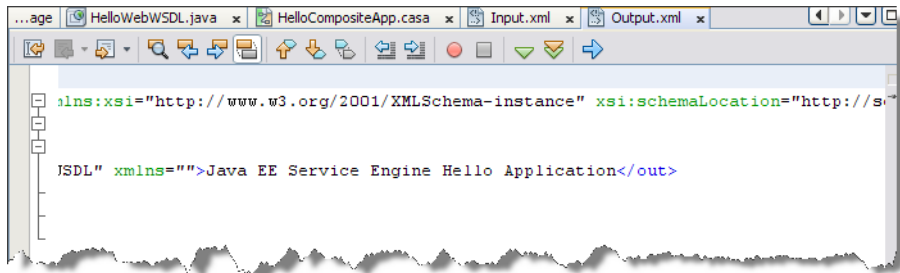
To compare the output with newly generated output, we have to right-click the generated Output file in the projects window and choose `Use Recent Result as Output` option. From the next run onwards the system compares the generated output with the `Output.xml` file and provides the result.

### 3 Run the test again.

The test case is compared to the current output file and succeeds.

#### 4 To check the output, double-click the Output node under TestCase1.

In the Output.xml tab, according to this tutorial example, the result should have a string as shown in the figure below.



## Summary

In this tutorial, you learned how the Java EE Service Engine implements a Web Service derived from WSDL, and you created and tested a composite application.

This tutorial demonstrates how to:

- Create a EJB Module project
- Create a WSDL document and a Web Service from WSDL
- Build and deploy a Composite Application project to GlassFish
- Create and run test cases