Using the Java EE Service Engine to Create a Composite Application



Sun Microsystems, Inc. 4150 Network Circle Santa Clara, CA 95054 U.S.A.

Part No: 821–0236 June 2009 Copyright 2009 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more U.S. patents or pending patent applications in the U.S. and in other countries.

U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

This distribution may include materials developed by third parties.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, the Solaris logo, the Java Coffee Cup logo, docs.sun.com, Java, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. or its subsidiaries in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and SunTM Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Products covered by and information contained in this publication are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical or biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Contents

Using the Java EE Service Engine to Create a Composite Application	5
Tutorial Requirements	6
Prerequisites	e
System Requirements	6
Tutorial Scenario	6
Configuring the Tutorial Environment	7
Starting the GlassFish Software	7
▼ To Check the Status of the GlassFish V2 Application Server in the NetBeans IDE	7
igvee To Register the GlassFish V2 Application Server with the NetBeans IDE	8
igvee To Start the GlassFish V2 Application Server in the NetBeans IDE	8
Enabling the JBI Framework	9
Creating an EJB Module Project	9
igvee To Create the EJB Module Project	9
▼ To Create a WSDL Document	12
▼ To Create a Web Service from WSDL	15
igvee To Clean and Build the EJB Module Project	19
Creating a Composite Application Project	19
igvee To Create a Composite Application Project	19
Building and Deploying the Composite Application Project	22
igvee To Build and Deploy the Composite Application	22
Testing the Composite Application	23
▼ To Add a Test Case	23
▼ To Run the Test Case	26
Summary	27

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 is designed to show 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 6
- "Tutorial Scenario" on page 6

What You Need to Do

This tutorial includes instructions on how to perform these tasks:

- "Configuring the Tutorial Environment" on page 7
- "Starting the GlassFish Software" on page 7
- "Enabling the JBI Framework" on page 9
- "Creating a Composite Application Project" on page 19
- "Building and Deploying the Composite Application Project" on page 22
- "Testing the Composite Application" on page 23

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.

System Requirements

This tutorial assumes that your system meets the requirements specified in the System Requirements topic of the NetBeans IDE 6.1 Release Notes. NetBeans IDE runs on operating systems that support the Java VM (Virtual Machine).

Tutorial Scenario

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



Configuring the Tutorial Environment

Before you deploy your composite application, the application server and the appropriate JBI components must be configured correctly and must be running.

The following software is required for this tutorial:

- GlassFish ESB (Installation Instructions) includes the following:
 - GlassFish V2 Update Release 2 (UR2)
 - NetBeans IDE 6.1
 - Open ESB core components
 - Java Business Integration (JBI) service engines
 - Java Business Integration (JBI) binding components
 - Java Business Integration (JBI) component tooling
- JDK (Java Development Kit) 6

Note – Glassfish ESB installer installs all the components including NetBeans IDE. To download only the NetBeans IDE, see NetBeans IDE Download.

Note – You must have the JDK (Java Development Kit) software installed and JAVA_HOME set as an environment variable, prior to installing the GlassFish ESB or the installation will halt midway. See Installing the JDK Software and Setting JAVA_HOME for details.

Starting the GlassFish Software

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 V2 Application Server in the NetBeans IDE

- If the Services window is not visible, on the NetBeans IDE menubar 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 V2 Application Server with the NetBeans IDE" on page 8.

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 V2 Application Server in the NetBeans IDE" on page 8.

To Register the GlassFish V2 Application Server with the NetBeans IDE

- 1 If the Services window is not visible, on the NetBeans IDE menubar choose $Window \rightarrow Services$.
- 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 and the password is adminadmin.

9 Click Finish.

To Start the GlassFish V2 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 \rightarrow JBI \rightarrow 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

- 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.

iteps	Name and Loca	ation	
Choose Project Name and Location	Project Name:	Hello]
Server and Settings	Project Location:	$\label{eq:c:Documents} C:\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$	Browse
	Project Folder:	C:\Documents and Settings\Administrator\My Documents\NetBeansProjects\Hello]
	Use Dedicate	d Folder for Storing Libraries	
	Libraries Folder:		Browse
	✔ Set as Main P	Different users and projects can share the same compilation libraries (se roject	e Help for deta
	☑ Set as Main P	Different users and projects can share the same compilation libraries (se	e Help for deta

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

New EJB Module	X
Steps	Server and Settings
 Choose Project Name and Location 	Add to Enterprise Application: <pre> </pre>
3. Server and Settings	Server: GlassFish V2 Add
	Java EE Version: Java EE 5
	< Back Next > Finish Cancel Help

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.

0 N	🕽 New WSDL Document 🛛 🛛 🔀				
Ste	ps	Name and L	ocatio	n	
1. 2.	Choose File Type Name and Location	File Name: n	ewWSI	DL	
3. 4.	Abstract Configuration Concrete Configuration	Project:	Hello	Proute Contract	
		Created File:	d Sett	ings\Administrator\My Documents\NetBeansProjects\Hello\src\java\newWSDL.v	vsdl
		Target Names WSDL Type:	pace:	http://j2ee.netbeans.org/wsdl/Hello/newWSDL O Abstract WSDL Document O Concrete WSDL Document	
		Binding:		SOAP	~
-		Туре:		RPC Literal	~
				< Back Next > Finish Cancel Help	

- 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.

🕽 New WSDL Document 🛛 🛛 🔀				
Ste	ps	Abstract Configu	ration	
1.	Choose File Type	Port Type Name:	newWSDLPortType	
2. 3.	Name and Location Abstract Configuration	Operation Name:	newWSDLOperation	
4.	Concrete Configuration	Operation Type:	Request-Response Operation	~
		Input:	Message Part Name	Element Or Type
			in	xsd:string]
			Add Remove	
		Output:	Message Part Name	Element Or Type
			out	xsd:string
			Add Remove	
		Fault:	Message Part Name	Element Or Type
	<u>.</u>			
			Add Remove	
		Generate par	tnerlinktype automatically.	
			< Back N	ext > Finish Cancel Help

- 8 Click Next.
- 9 Choose the defaults and click Finish on the Concrete Configuration page.

N	lew WSDL Document		×
Ste	:ps	Concrete Configuration	
1. 2. 3. 4.	Choose File Type Name and Location Abstract Configuration Concrete Configuration	Binding Name: HelloWSDLBinding Service Name: HelloWSDLService Port Name: HelloWSDLPort	
-			
		< Back Next > Finish Cancel H	telp

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.

New File	🕽 New File 🛛 🛛 🔀			
Steps	Choose File Type			
1. Choose File Type	Project: 🚫 Hello			
	Categories: ESB Categories: ESB Categories: Categories: ESB Categories: DavaBeans Objects Dunit Persistence Web Services Mub GlassFish Other Description:	File Types: Web Service Web Service from WSDL Web Service Client Secure Token Service (STS) Logical Handler Message Handler		
	Creates a skaleton web service from WSD semantically encapsulate discrete function such as SOAP. The web services created to JSR-109 and J2EE 1.4 standards.	L file. Web services are reusable software components that ality. Web services are accessible over standard protocols any this template are deployed and run according to the		
	< Back	Next > Finish Cancel Help		

- 3 Click Next.
- 4 Type the Web Service Name as HelloWebWSDL and the Package name as Hello1.

New Web Service from WSD		
Steps	Name and Location	
Choose File Type Name and Location	Web Service Name: HelloWebWSDL	
	Project: Hello	
	Location: Source Packages	~
	Package: Hello1	~
	Select Local WSDL File or Enter WSDL URL:	
	Browse	
	If WSDL file defines more services and ports press Browse button to select port from which web service will be generated.	
	Web Service Port: Browse	
	Use Provider	
A CONTRACTOR		
	O Enter the name of the WSDL file representing the service you wish to use.	
	< Back Next > Finish Cancel	Help

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

- 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

🕡 New Project		
Steps	Choose Project	
 Choose Project 	Categories: Java Web Enterprise NetBeans Modules SOA General Samples	Projects: Composite Application BPEL Module XSLT Module Data Mashup Module
	Description: Creates an empty Composite Application p other types of Java Business Integration (Note: For more infomation, press the Help Composite Application Projects .	roject, which may include multiple BPEL Modules and JBI) modules.
	< Back	Next > Finish Cancel Help

- 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.

N	🕡 New Composite Application Project 🛛 🛛 🔀			
Ste	ps	Name and Locati	ion	
1. 2.	Choose Project Name and Location	Project Name:	HelloCompositeApp	
		Project Location:	ants and Settings\Administrator\My Documents\NetBeansProjects	Browse
		Project Folder:	Iministrator\My Documents\NetBeansProjects\HelloCompositeApp	
		Set as Main Pr	roject	
			< Back Next > Finish Cancel	Help

- 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.

🛓 Select Pro	ject		
Look in: 🛅 N	etBeansProjects	~	1 🕫 📂
	dule1	~	Project Name:
🗄 🛅 BpelMoo	dule2		Hello
🗄 🗁 🛅 BpelMoo	dule3		
🕀 🗁 🛅 BpelMoo	dule4		Project JAR Files:
🗄 🕀 🔁 🔁 🕀 🔁	dule5		dist/Hello.jar
🗄 🦳 Compos	siteApp1	=	
E Compos	siteApp2 siteApp3	=	
E Compos	siteApp4		
🗄 🖷 Hello			
🗄 🖷 🔚 HelloCo	mpositeApp		
🗄 🗁 🛅 Hellofro	mWSDL		
HelloWs	5DL		
🕀 💼 pollwrite	e	Y	
File name:	C:\Documents and Settings\Administrator\My Documents\	NetB	BeansProjects\Hello Add Project JAR Files
Files of type:	Project Folder		Cancel

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

New Test Case	X
New Test Case Steps . Enter the Test Case Name . Select the WSDL Document . Select the Operation to Test	Select the WSDL Document WSDL Documents: HelloCompositeApp - Process Files Hello - Hello Hello - Hello Hello - Gonf Hello - Java HelloWSDL.wsdl
	Selected WSDL: My Documents\NetBeansProjects\Hello\src\java\HelloWSDL.wsdl
	< Back Next > Finish Cancel Help

- 4 Click Next.
- 5 From the Select the Operation to Test page, select HelloWSDLOperation and click Finish.

New Test Case	
Steps	Select the Operation to Test
 Enter the Test Case Name Select the WSDL Document Select the Operation to Test 	Binding Operations: HelloWSDLPort (Binding="HelloWSDLBinding") HelloWSDLOperation
	Selected Operation: elloWSDLOperationRequest): HelloWSDLOperationResponse
< Back Next > Finish Cancel Help	

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.

6 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