Introduction to Service Component Architecture

What this exercise is about

The objective of this lab is to provide you with an understanding of how to use the WebSphere® Integration Developer V6 tools to build an application that is based upon the service component architecture (SCA).

Lab Requirements

List of system and software required for the student to complete the lab.

- WebSphere Integration Developer V6 installed
- WebSphere Process Server V6 test environment installed
- Sample code in the directory C:\Labfiles60 (Windows®) or /tmp/LabFiles60 (Linux® and AIX®)

What you should be able to do

At the end of this lab you should be able to:

- Create a new business integration module
- Use the interface editor to define a simple WSDL interface

- Use the assembly diagram editor to:
    - Define a simple service component with a single WSDL port type interface and a Java™ implementation
    - Define a stand-alone reference to the simple service component
    - Assemble a service module
- Use the SCA client programming model to invoke a service component from within a service client
- Test the service component invocation from a
    - Client JSP
    - Test framework

Introduction

The Service Component Architecture (SCA) is a service oriented component model for building business services that publish or operate on business data, and is the foundation upon which applications running on WebSphere Process Server V6 are built. In this exercise you will build a simple "Hello World" application that will provide an introduction to SCA and the authoring tools provided with WebSphere Integration Developer V6.

The Hello World application in this exercise consists of a single service component that is associated with a simple WSDL interface definition that includes a sendMessage operation. The service component in this application uses a Java implementation (POJO – Plain Old Java Object) to provide the sendMessage functionality. Once the simple service component is built you will define a stand-alone reference that will enable the HelloWorld service to be invoked by a service client (such as a JSP) using the SCA client programming model. The following is a diagram illustrating the SCA module you will build in this exercise:

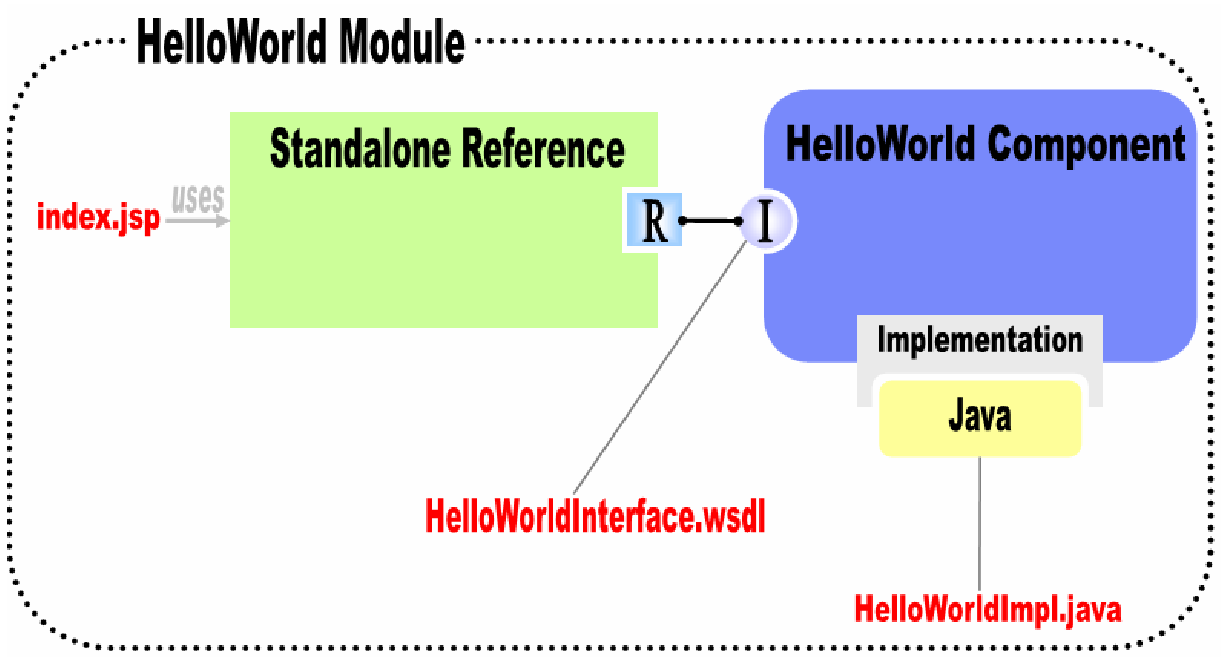


Figure 1

Exercise Instructions

Some instructions in this lab may be Windows operating-system specific. If you plan on running the lab on an operating-system other than Windows, you will need to run the appropriate commands, and use appropriate files (.sh vs. .bat) for your operating system. The directory locations are specified in the lab instructions using symbolic references, as follows:

Reference Variable	Windows Location	Linux Location
<LAB_NAME>	HelloWorld	HelloWorld
<WID_HOME>	C:\Program Files\IBM\WebSphere\ID\6.0	/opt/IBM/WebSphere/ID/6.0
<WPS_HOME>	<WID_HOME>\runtimes\bi_v6	<WID_HOME>/runtimes/bi_v6
<LAB_FILES>	C:\Labfiles60	/tmp/Labfiles60
<TEMP>	C:\temp	/tmp
<SOLUTION>	C:\Labfiles60\HelloWorld\Solution	/tmp/Labfiles60/HelloWorld/Solution

Windows users' note: When directory locations are passed as parameters to a Java program such as EJBDeploy or wsadmin, you must replace the backslashes with forward slashes to follow the Java convention. For example, C:\LabFiles60\ would be replaced by C:/LabFiles60/

Note that the previous table is relative to where you are running WebSphere Integration Developer. The following table is related to where you are running remote test environment:

Reference Variable	Example: Remote Windows test server location	Example: Remote z/OS® test server location	Input your values for the remote location of the test server
<SERVER_NAME>	server1	cl1sr01	
<WAS_HOME>	C:\Program Files\IBM\WebSphere\AppServer	/etc/cl1cell/AppServerNode1	
<HOSTNAME>	localhost	mvsxxx.rtp.raleigh.ibm.com	
<BOOTSTRAP_PORT>	2809	2809	
<TELNET_PORT>	N/A	1023	
<PROFILE_NAME>	AppSrv01	default	
<USERID>	N/A	cl1admin	
<PASSWORD>	N/A	fr1day	

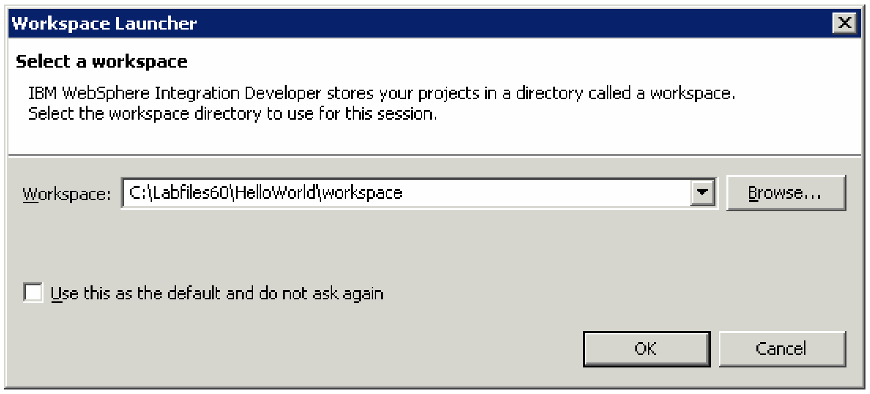
Instructions for using a remote testing environment, such as z/OS, AIX or Solaris, can be found at the end of this document, in the section "**Task: Adding Remote Server to WebSphere Integration Developer Test Environment**".

Part 1: Set up Development Environment

____ 1. Start WebSphere Integration Developer V6 with a new workspace

__ a. Select **Start** > **Programs** > **IBM WebSphere** > **Integration Developer V6.0.1** > **WebSphere Integration Developer V6.0.1** from the start menu.

__ b. Enter **<LAB_FILES>\<LAB_NAME>\workspace** for your workspace and click **OK w**hen prompted.



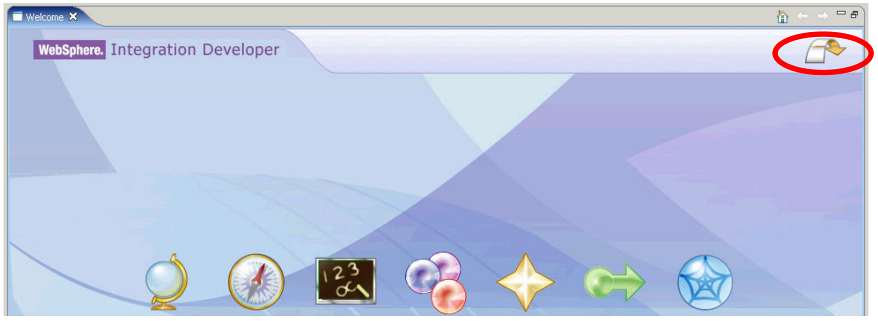
Workspace Launcher

Select a workspace

IBM WebSphere Integration Developer stores your projects in a directory called a workspace.
Select the workspace directory to use for this session.

Workspace: C:\Labfiles60\HelloWorld\workspace

__ c. Close the **Welcome page** after WebSphere Integration Developer V6.0 opens.

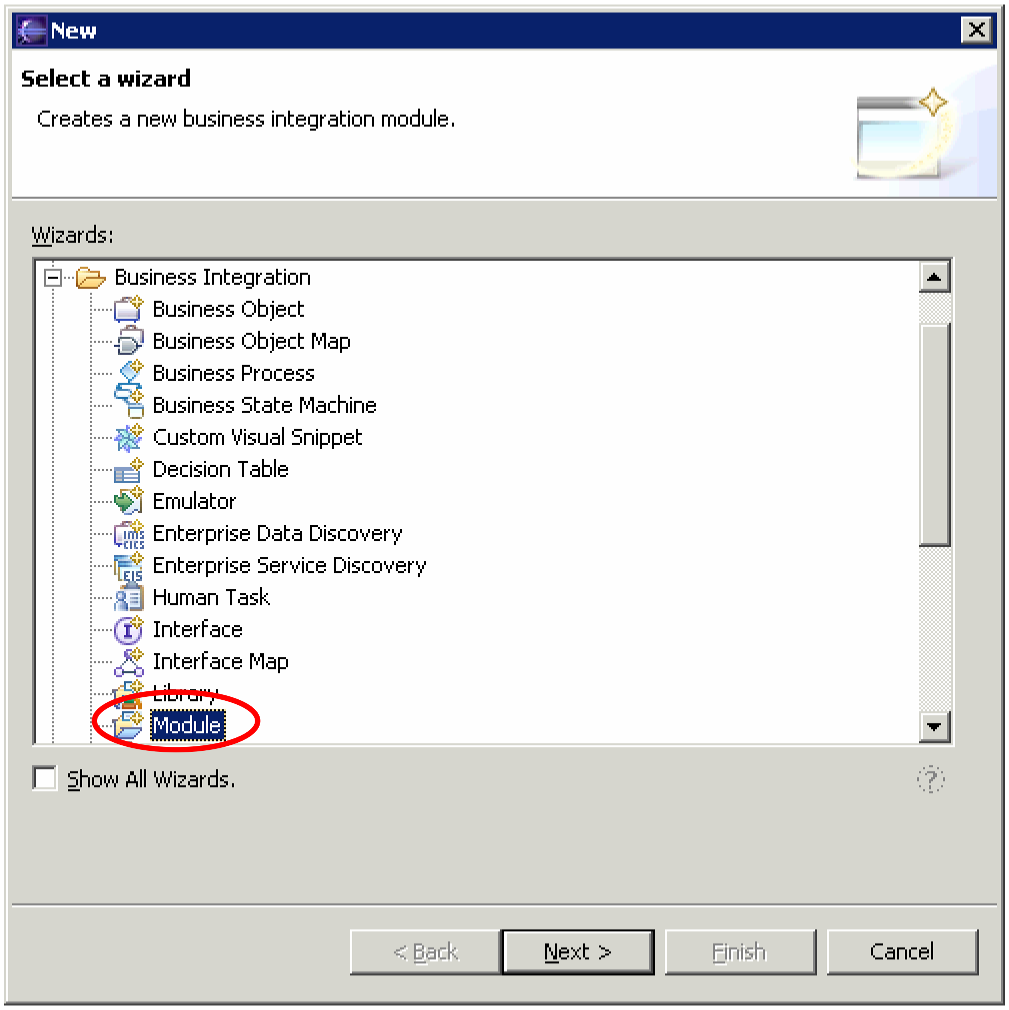


Part 2: Build the HelloWorld Module

____ 1. The fundamental project type for building SCA applications in WebSphere Integration Developer V6 is called a Module. Begin building the HelloWorld application by creating a new Module called **HelloWorld**.

__ a. Select **File > New > Other**.

__ b. Expand Business Integration and select **Module**.


New

Select a wizard

Creates a new business integration module.

__ c. Click **Next**.

__ d. Enter **HelloWorld** for the Module name and click **Finish**.

____ 2. The SCA component in this application is associated with a WSDL interface called HelloWorldInterface (See Figure 1 of the Introduction section). Create a new WSDL interface with this name.

__ a. Expand **HelloWorld** from the Business Integration view.

__ b. Right click on Interfaces and select **New > Interface**.

__ c. Enter **HelloWorldInterface** for the name and verify that the selected Module is **HelloWorld**.

__ d. Click **Finish**.

____ 3. The HelloWorldInterface has a single operation called sendMessage that takes a string message as input and returns a status string. Use the WSDL editor to define the sendMessage operation on the HelloWorldInterface.

__ a. After completing step 2 above, the WSDL editor should be open in your workspace.

__ b. Click the Add Request Response Operation icon ( ) at the top of the WSDL editor.

__ c. Change the operation name from operation1 to **sendMessage**.

NOTE: The name of the operation can be modified from either the properties view or within the interface editor itself.

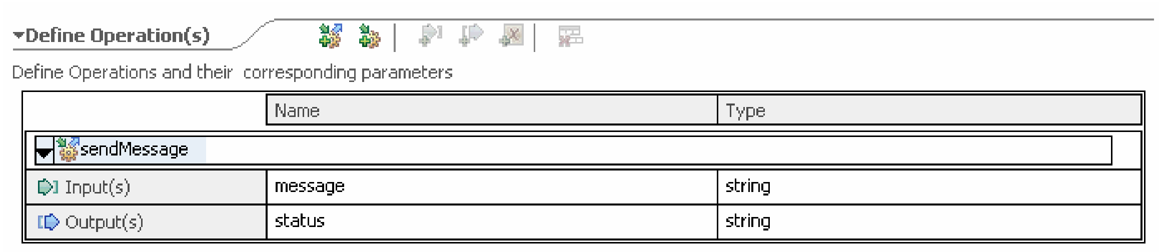
__ d. Click on the Add Input icon ( ) at the top of the WSDL editor.

__ e. Change the input1 input name to **message** and accept string as the type for the input parameter.

__ f. Click on the Add Output icon ( ) at the top of the WSDL editor.

__ g. Change the output1 output name to **status** and accept string as the type for the output parameter.

__ h. Verify that your interface matches the following diagram.



	Name	Type
sendMessage		
Input(s)	message	string
Output(s)	status	string

__ i. **Save** and **Close** the WSDL file.

____ 4. The SCA component used in this application uses a Java implementation for the WSDL interface defined in the previous step. Import the HelloWorldImpl.java file for this implementation.

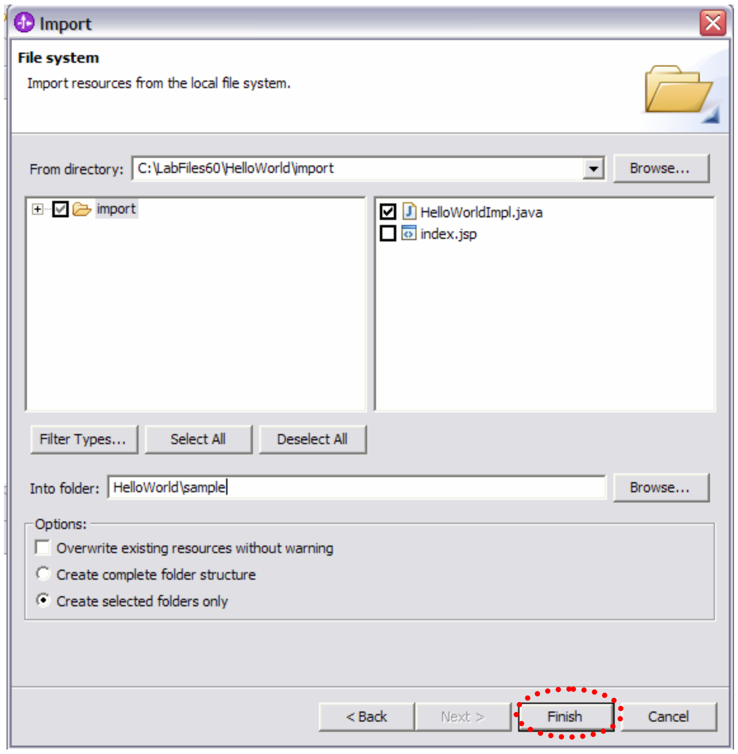
__ a. Select **File > Import** from the menu.

__ b. Select **File system** for the import source and click **Next**.

__ c. Use the Browse button to select **<LAB_FILES>\HelloWorld \import** for the “from” directory.

__ d. Check the box next to **HelloWorldImpl.java**.

__ e. Specify the '**Into folder**' as **HelloWorld\sample**.


Import

File system

Import resources from the local file system.

From directory: C:\LabFiles60\HelloWorld\import

☑ import

☑ HelloWorldImpl.java

☐ index.jsp

Into folder: HelloWorld\sample

Options:

☐ Overwrite existing resources without warning

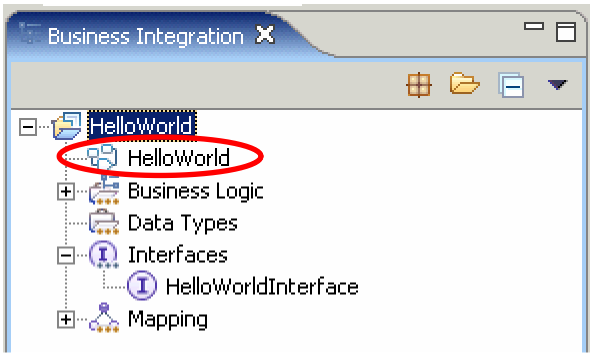
○ Create complete folder structure

● Create selected folders only

__ f. Click **Finish**.

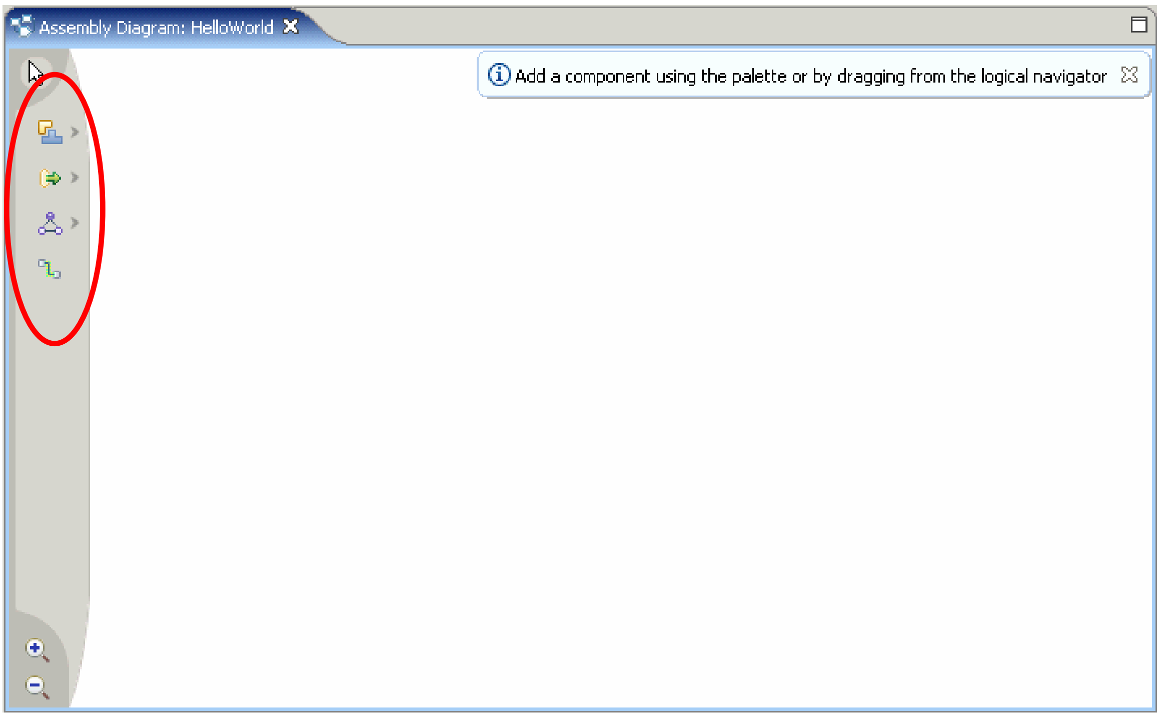
____ 5. The primary tool for composing an SCA-based application is the assembly editor. Open the assembly editor for the HelloWorld module and build the SCA module illustrated in Figure 1 of the Introduction section.

__ a. Go to the Business Integration view, expand **HelloWorld** and double click on HelloWorld to open the assembly editor for the HelloWorld module..



__ b. The assembly editor will open in your workspace.

NOTE: The assembly editor consists of a canvas area and a palette for selecting components to add to the diagram. The palette is located to the left of the canvas area. To locate some of the nested items that can be added to a diagram, click the gray **>** symbol next to the parent item on the palette.



Add a component using the palette or by dragging from the logical navigator

__ c. Add a service component to the canvas of the assembly diagram.

__ d. **NOTE:** To do this, locate and click the component icon ( ) from the palette and then click anywhere on the canvas area of the assembly diagram. In the assembly diagram, select the component added in the previous step.

__ e. Use the properties view to change the Display name and Name of the component from Component1 to **HelloWorld**. This property can be found on the General tab.

__ f. In the assembly diagram select the **HelloWorld** component and select the Add Interface icon ( ) from the hover over menu.

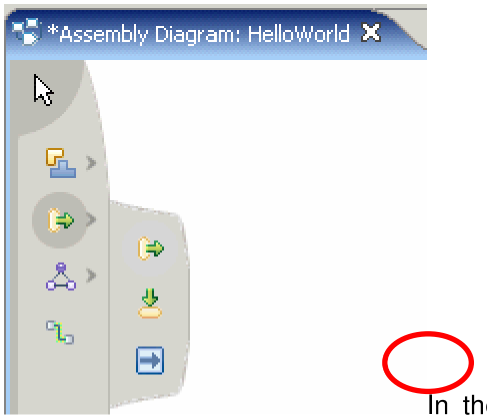
NOTE: The hover over menu will pop up if you position your mouse over a component selected in the diagram.

__ g. Select the **HelloWorldInterface** from the '**Matching interface**' list and click **OK**.

__ h. In the assembly diagram right click on the **HelloWorld** component and select **Select Implementation > Java** from the context menu.

__ i. When the Pick Implementation dialog appears, begin typing **HelloWorldImpl** into the "**Select entries**" box. Select the **HelloWorldImpl** from the "**Matching types**" lists when it appears and click **OK**.

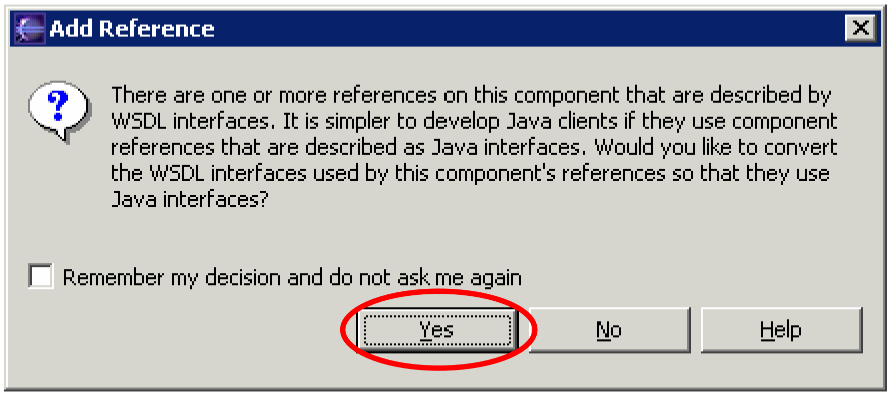
__ j. Return to the assembly diagram and add a stand-alone reference to the diagram. To do this, locate and click on the stand-alone reference icon ( ) on the palette and then click anywhere on the canvas of the assembly diagram. The following diagram indicates where to find the stand-alone reference icon.



__ k. In the assembly diagram select the stand-alone reference and click the Add Reference icon ( ) from the hover over menu.

__ l. In the Add Partner Reference dialog enter **HelloWorldInterfacePartner** for the name. Select the **HelloWorldInterface** from the 'Matching Interface' list and click **OK**.

__ m. Click **Yes** when prompted if you would like to convert the WSDL interface to Java interfaces.



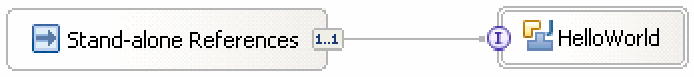
There are one or more references on this component that are described by WSDL interfaces. It is simpler to develop Java clients if they use component references that are described as Java interfaces. Would you like to convert the WSDL interfaces used by this component's references so that they use Java interfaces?

Remember my decision and do not ask me again

__ n. Create a wire between the stand-alone reference and the HelloWorld component. To do this, select the wire tool ( ) on the palette. Then, click on the reference box on the stand-alone reference ( ) followed by the HelloWorld component.

__ o. Click on the selection tool ( ) from the palette.

__ p. Verify that your assembly diagram looks like the following.



NOTE: You can automatically arrange the contents of the assembly diagram by right clicking on the assembly diagram and selecting **Arrange Contents Automatically**. You may need to press the Esc key or click again on the wire tool on the palette to right click in the assembly diagram.

__ q. (Optional) Click on the wire connecting the stand-alone reference and the HelloWorld component. Use the properties view to inspect the wire definition. Notice that the reference name is HelloWorldInterfacePartner. This name will be used in the client JSP file you will create in the following steps.

__ r. **Save** and **Close** the assembly diagram file.

____ 6. The HelloWorld application includes a web interface to allow users to invoke the HelloWorld service component from a web browser. Before importing the client JSP file that includes the appropriate client code to invoke the HelloWorld service, create a new web module to hold this client interface. . (Step 6 is required for 6.0.1 and beyond but is not required for version 6.0)

__ a. Select the **J2EE Perspective** and expand **Enterprise Applications.**

__ b. Right click on HelloWorldApp and select **New > Dynamic Web Project** from the list.

__ c. Enter **HelloWorldClient** for the name of the Dynamic Web project.

__ d. Click **Finish**.

Click **Yes** to confirm a switch to the Web Perspective.

NOTE: This will enable the Web Development capability within WebSphere Integration Developer for this workspace.

____ 7. For this exercise you will import the client JSP file that includes the appropriate client code to invoke the HelloWorld service component created in the previous steps.

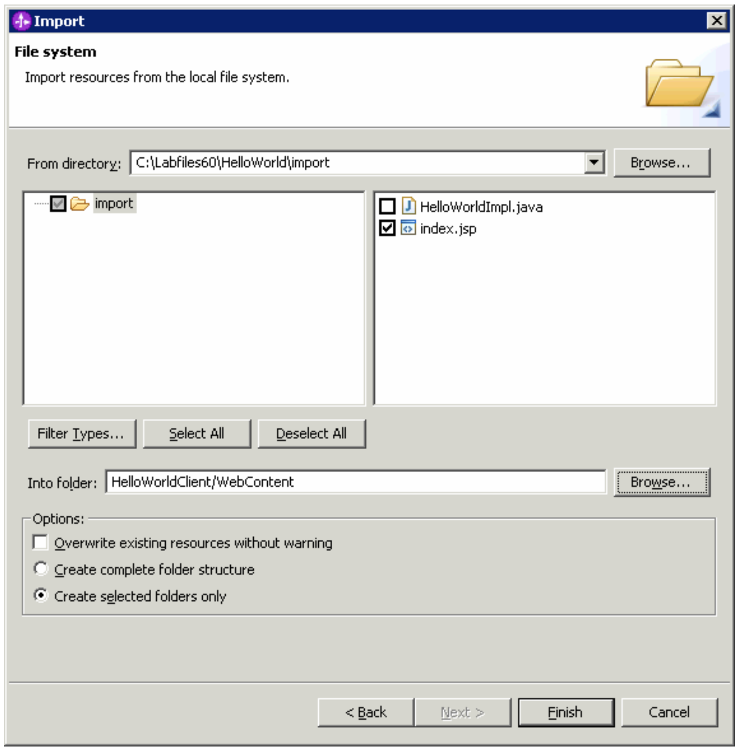
__ a. Select **File > Import** from the menu.

__ b. Select ‘File System’ and click **Next**.

__ c. From the File system import dialog click the **Browse** button and choose the following ‘**From directory**’: **<LAB_FILES>\HelloWorld\import**.

__ d. Check the box next to the index.jsp file.

__ e. Click the Browse button to select **HelloWorldClient**/**WebContent** as the '**Into folder**'.



From directory: C:\Labfiles60\HelloWorld\import

[✓] import

[ ] HelloWorldImpl.java
[✓] index.jsp

Into folder: HelloWorldClient/WebContent

[ ] Overwrite existing resources without warning
( ) Create complete folder structure
(●) Create selected folders only

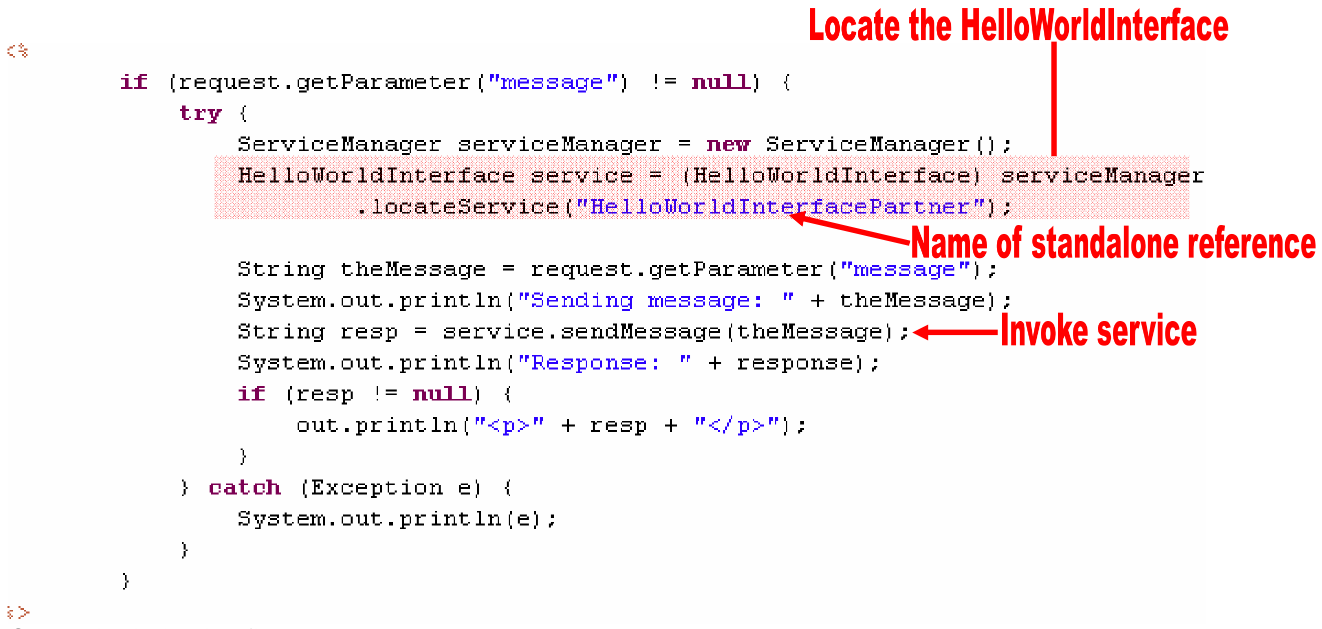
__ f. Click **Finish**.

____ 8. Open the index.jsp file and inspect the client code that is used to invoke the HelloWorld service component.

__ a. If you are in Business Integration perspective, switch to the Web Perspective by selecting **Window > Open Perspective > Other** and check the box next to Show all. Select **Web** and click **OK**.

__ b. Expand **Dynamic Web Projects > HelloWorldClient > WebContent** from the Project Explorer view and double click on **index.jsp**.

__ c. Click on the Source tab after the JSP editor opens and examine the following code.



```
<%
        if (request.getParameter("message") != null) {
            try {
                ServiceManager serviceManager = new ServiceManager();
                HelloWorldInterface service = (HelloWorldInterface) serviceManager
                        .locateService("HelloWorldInterfacePartner");

                String theMessage = request.getParameter("message");
                System.out.println("Sending message: " + theMessage);
                String resp = service.sendMessage(theMessage);
                System.out.println("Response: " + response);
                if (resp != null) {
                    out.println("<p>" + resp + "</p>");
                }
            } catch (Exception e) {
                System.out.println(e);
            }
        }
%>
```

__ d. **Close** the index.jsp file.

Part 3: Testing HelloWorld from the Web Interface

____ 1. Start the server.

__ a. If using a remote testing environment, follow the directions provided in **Task: Adding Remote Server to WebSphere Integration Developer Test Environment** at the end of this document to add a server to the WebSphere Integration Developer test environment and start it. This is especially true for z/OS, AIX, Solaris remote test environment, where the WebSphere Integration Developer will be remote to the test environment.

If using a local testing environment, right click on WebSphere Process Server V6.0 from the Server view and select **Start** from the context menu.

__ b. Wait until the sever status is indicated as **Started**.

____ 2. Add the HelloWorld project to the configured projects on the server.

__ a. From the Servers view right click on WebSphere Process Server V6.0 and select **Add and remove projects…** from the context menu.

__ b. Select **HelloWorldApp** from the available projects list and click **Add**.

__ c. Click **Finish**.

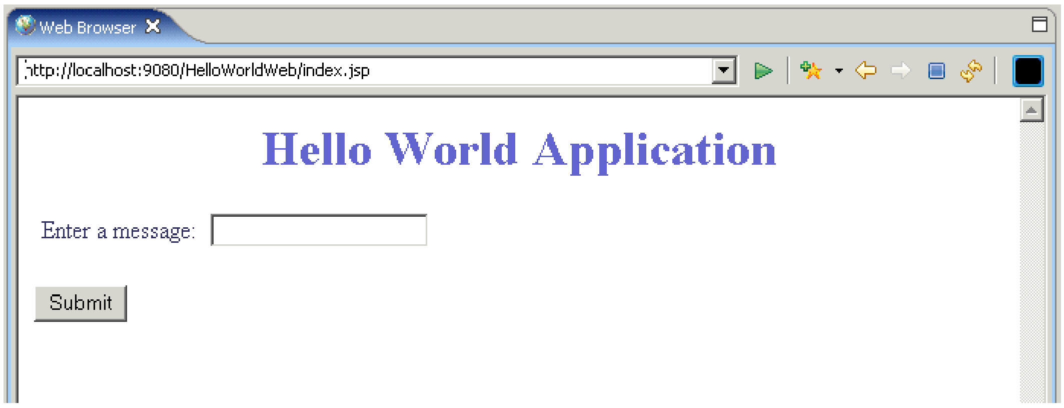
____ 3. Test the HelloWorld service component.

__ a. Open a web browser and enter the following URL.

`http://<HOSTNAME>:9080/HelloWorldClient/index.jsp`

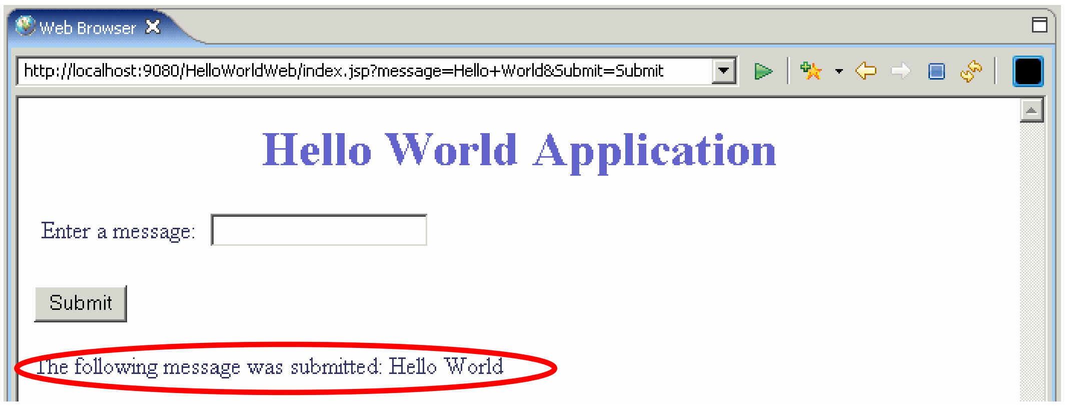
NOTE: To open the embedded browser within WebSphere Integration Developer V6 select **Window > Customize Perspective** from the menu. In the Customize Perspective dialog click on the Commands tab and scroll to the bottom of the available command groups list until you find Web Browser. Check the box next to Web Browser and click **OK**. Click the web browser icon (🌐) now in your toolbar to open the embedded web browser.

__ b. Verify that you see the following page displayed in the browser.



__ c. Enter **Hello World** for your message and click Submit.

__ d. Verify that you see the following output.



http://localhost:9080/HelloWorldWeb/index.jsp?message=Hello+World&Submit=Submit

Hello World Application

The following message was submitted: Hello World

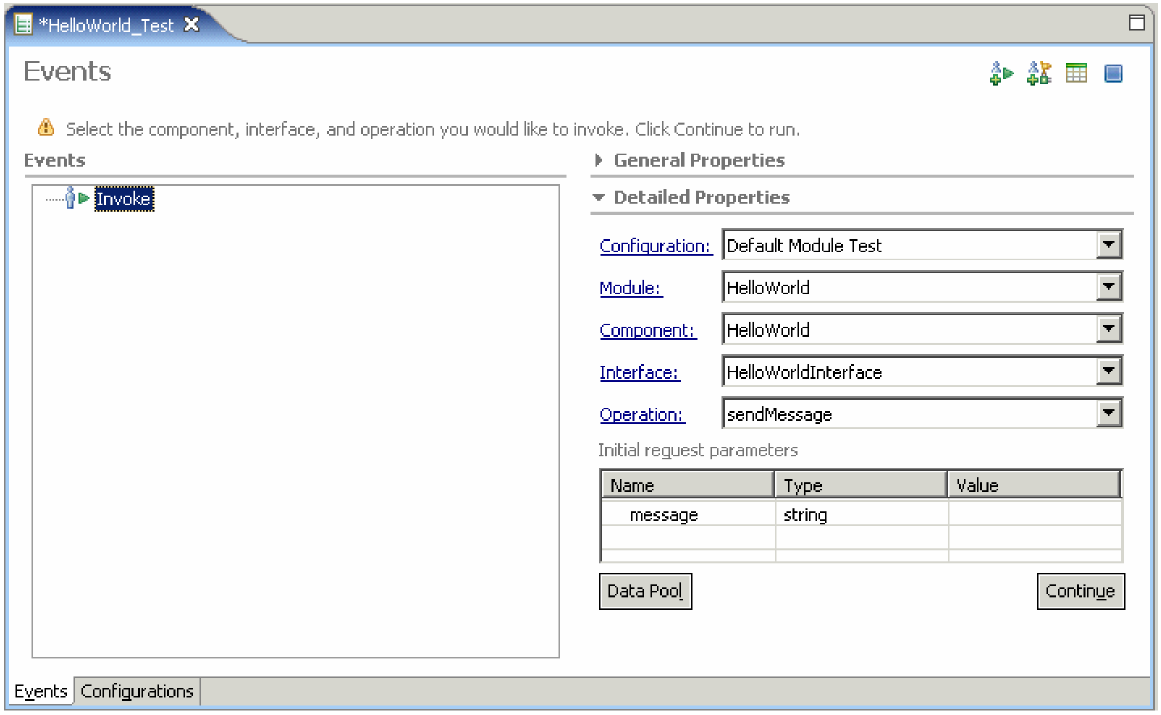
Part 4: Testing Using the Integration Test Client

____ 1. The integration test client can be invoked on a service component from the assembly diagram. Open the assembly diagram and launch a component test on the HelloWorld Service component.

__ a. Expand **HelloWorld** and double click on HelloWorld from the Business Integration view.

Right click on the HelloWorld world component and select **Test Component** from the context menu.

Verify that the Test Editor opens in your work area.



Select the component, interface, and operation you would like to invoke. Click Continue to run.

Configuration: Default Module Test
Module: HelloWorld
Component: HelloWorld
Interface: HelloWorldInterface
Operation: sendMessage

Initial request parameters

Name	Type	Value
message	string	

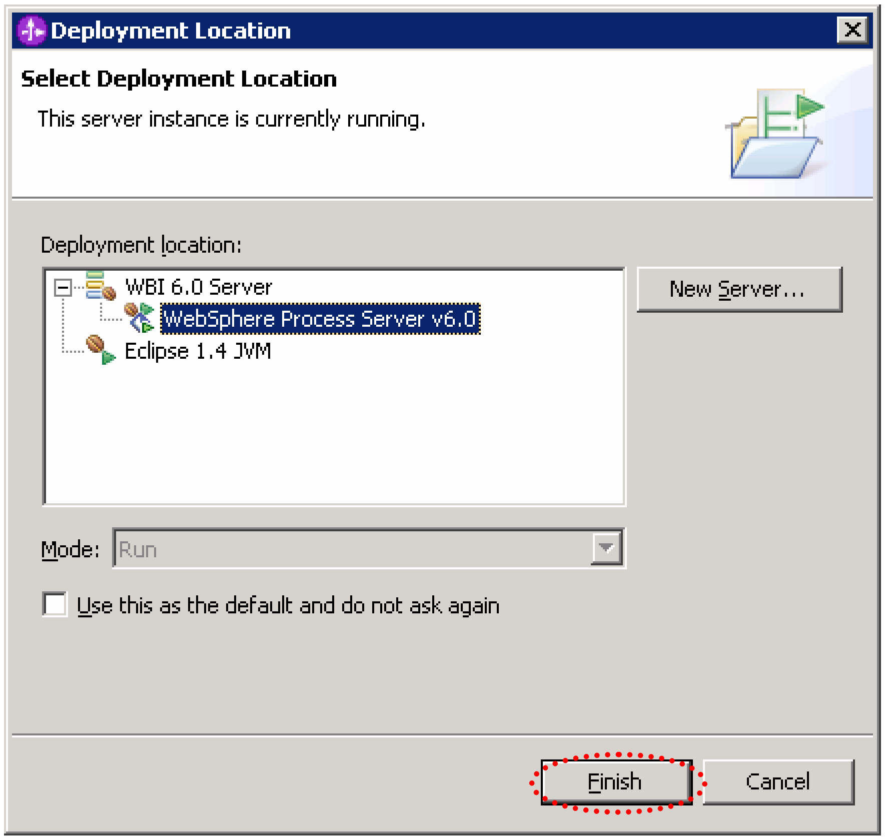
____ 2. For this test invoke the HelloWorld service component with the message “Hello World 2”.

Locate the table that lists the initial request parameters under the “Detailed Properties” section.

Enter **Hello World 2** for the message parameter value.

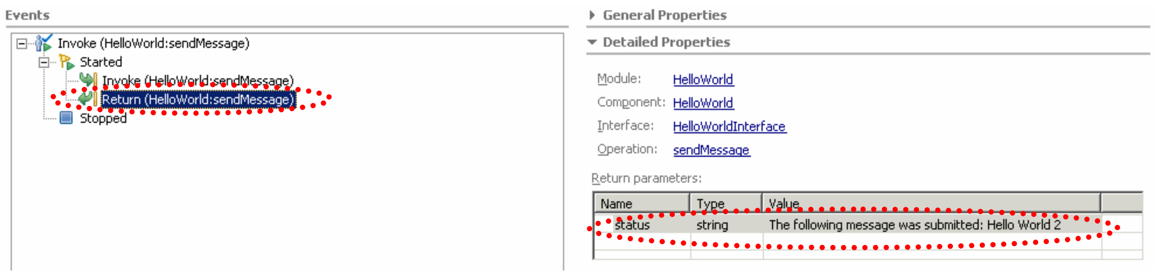
Click the **Continue** button underneath the parameters table to invoke the test.

Expand WBI 6.0 Server and click on **WPS Server v6.0** from the Deployment Location dialog appears.


Deployment Location
Select Deployment Location
This server instance is currently running.
Deployment location:
WBI 6.0 Server
WebSphere Process Server v6.0
Eclipse 1.4 JVM
New Server...
Mode: Run
Use this as the default and do not ask again
Finish
Cancel

Click **Finish**.

____ 3. Once the test has completed, verify that the invocation response is: “The following message was submitted: Hello World 2”.


Events
Invoke (HelloWorld:sendMessage)
Started
Invoke (HelloWorld:sendMessage)
Return (HelloWorld:sendMessage)
Stopped
General Properties
Detailed Properties

Module:	HelloWorld
Component:	HelloWorld
Interface:	HelloWorldInterface
Operation:	sendMessage

Return parameters:

Name	Type	Value
status	string	The following message was submitted: Hello World 2

____ 4. Close the test editor without saving the data file associated with the test run. Close the HelloWorld assembly diagram.

Part 5: Restore Server Configuration

____ 1. Remove the HelloWorld project from the configured projects on the server.

__ a. Right click on WebSphere Process Server v6.0 sever from the Server view and select **Add and remove projects…** from the context menu.

__ b. Select HelloWorldApp from the configured projects list and click **Remove**.

__ c. Click **Finish**.

____ 2. Stop the server.

__ a. Right click on WebSphere Process Server v6.0 server from the Servers view and select **Stop** from the context menu.

What you did in this exercise

In Part 1 of this exercise you set up the development environment. Next, in Part 2 you assembled the HelloWorld SCA module consisting of a single Java typed service component called HelloWorld that was associated with a simple WSDL interface. The SCA module also included a stand-alone reference that was used by a client JSP to invoke the HelloWorld service component. During the testing phase of this exercise you tested the HelloWorld application first from the Web interface, and then also with the end-to-end test client feature provided with WebSphere Integration Developer V6.

Solution Instructions

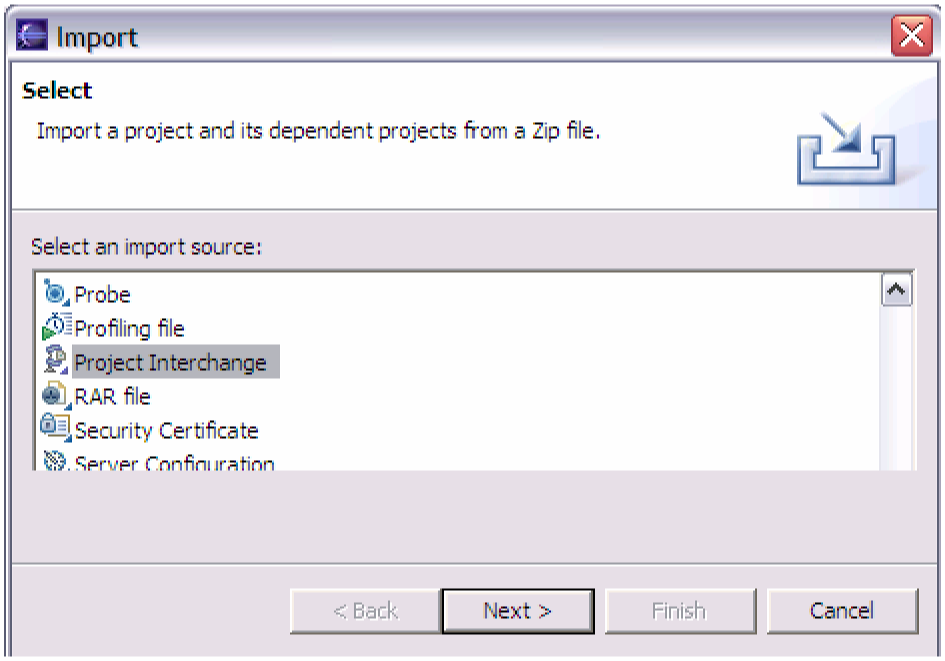
____ 1. Start WebSphere Integration Developer V6 with a new workspace

__ a. Follow the instructions outlined in Part 1 of this exercise

____ 2. Import the project interchange file **HelloWorld_PI.zip** from **<LAB_FILES>\<LAB_NAME>\Solution** directory.

__ a. Select **File > Import** from the menu

__ b. Select **Project Interchange** in the **Import** dialog and click **Next**



__ c. For the "**From zip file**', click on the **Browse** button and select the **HelloWorld_PI.zip** in the **<LAB_FILES>\<LAB_NAME>\Solution** directory.

__ d. Enter **<LAB_FILES>\<LAB_NAME>\workspace** for the **Project location root** and click **Next**

__ e. Select the check boxes next to **HelloWorld** and **HelloWorldClient**

__ f. Click **Finish**.

____ 3. Continue with Part 3 of this exercise.

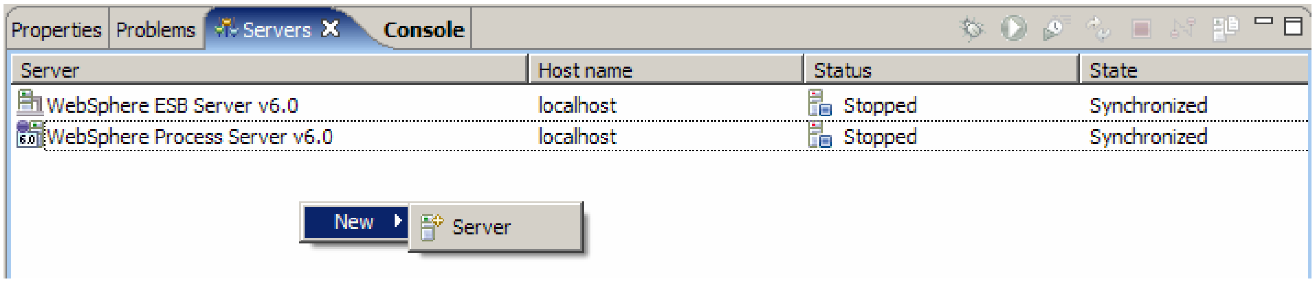
Task: Adding Remote Server to WebSphere Integration Developer Test Environment

This task describes how to add a remote server to the WebSphere Integration Developer Test environment. The sample will use a z/OS machine.

____ 1. Create a new remote server.

__ a. Right click on the background of the Servers view to access the pop-up menu.

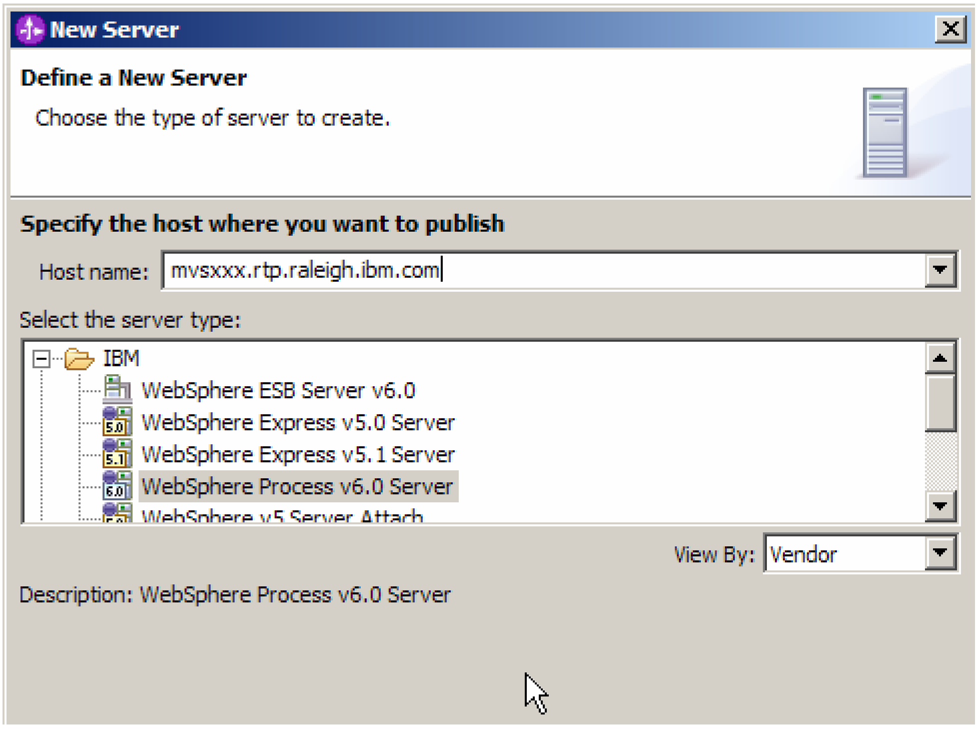
__ b. Select **New > Server.**



Server	Host name	Status	State
WebSphere ESB Server v6.0	localhost	Stopped	Synchronized
WebSphere Process Server v6.0	localhost	Stopped	Synchronized

__ c. Specify hostname to the remote server, **<HOSTNAME>**.

__ d. Ensure that '**WebSphere Process v6.0 Server**' is highlighted in the server type list.



Define a New Server

Choose the type of server to create.

Specify the host where you want to publish

Host name: mvsxxx.rtp.raleigh.ibm.com

Select the server type:

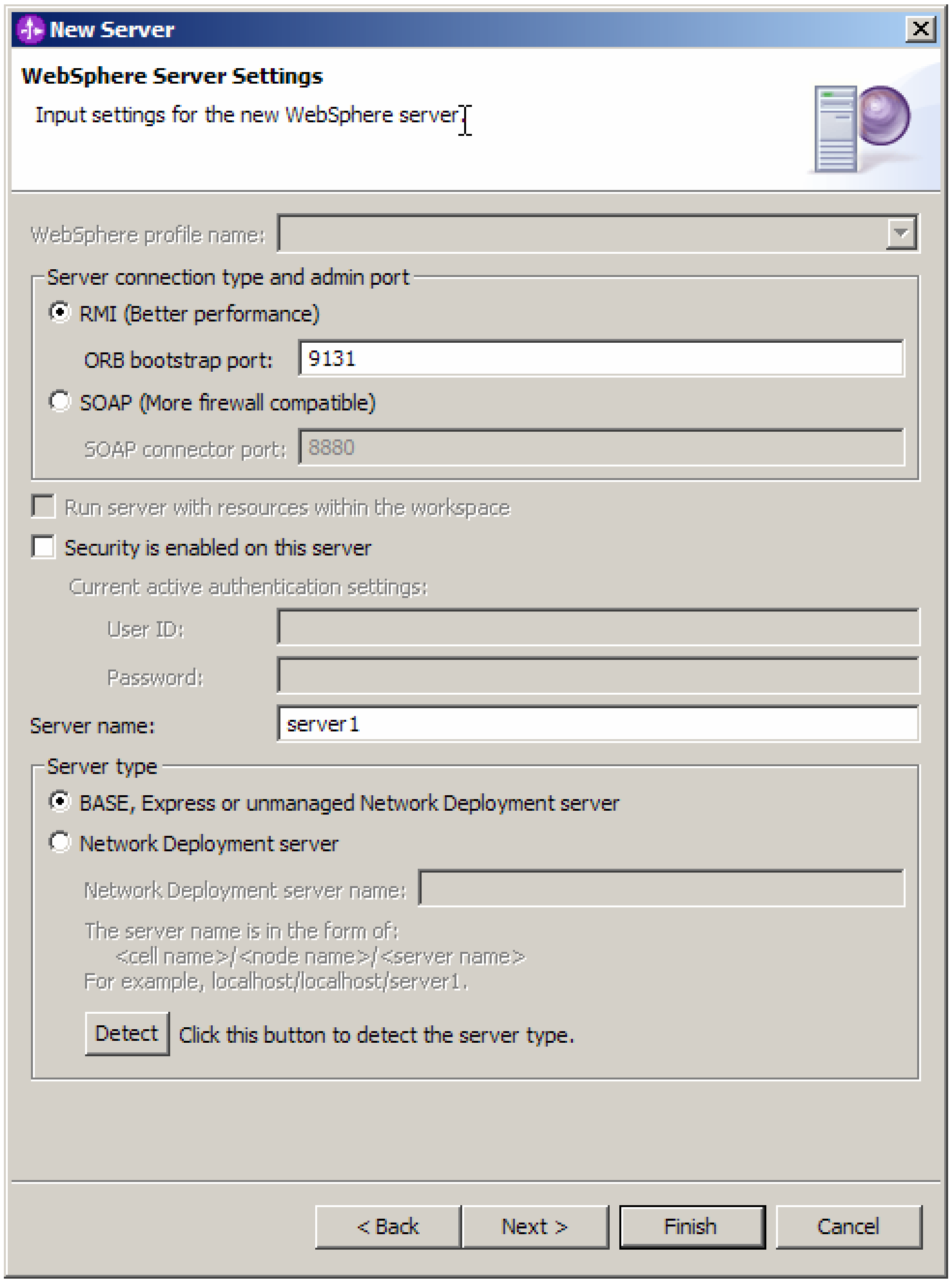
- IBM
  - WebSphere ESB Server v6.0
  - WebSphere Express v5.0 Server
  - WebSphere Express v5.1 Server
  - WebSphere Process v6.0 Server
  - WebSphere v5 Server Attach

View By: Vendor

Description: WebSphere Process v6.0 Server

__ e. Click **Next.**

__ f. On the WebSphere Server Settings page, select the radio button for **RMI** and change the ORB bootstrap port to the correct setting (**<BOOTSTRAP_PORT>**).



WebSphere Server Settings

Input settings for the new WebSphere server

WebSphere profile name:

Server connection type and admin port

(●) RMI (Better performance)

ORB bootstrap port: 9131

( ) SOAP (More firewall compatible)

SOAP connector port: 8880

☐ Run server with resources within the workspace

☐ Security is enabled on this server

Current active authentication settings:

User ID:

Password:

Server name: server1

Server type

(●) BASE, Express or unmanaged Network Deployment server

( ) Network Deployment server

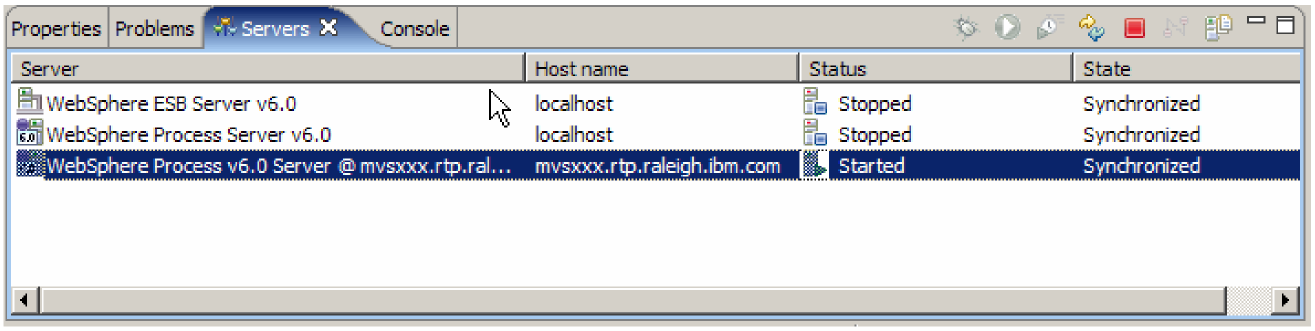
Network Deployment server name:

The server name is in the form of:
<cell name>/<node name>/<server name>
For example, localhost/localhost/server1.

Click this button to detect the server type.

__ g. Click **Finish**.

__ h. The new server should be seen in the Server view.



Server	Host name	Status	State
WebSphere ESB Server v6.0	localhost	Stopped	Synchronized
WebSphere Process Server v6.0	localhost	Stopped	Synchronized
WebSphere Process v6.0 Server @ mvsxxx.rtp.ral...	mvsxxx.rtp.raleigh.ibm.com	Started	Synchronized

____ 2. Start the remote server if it is not already started. WebSphere Integration Developer does not support starting remote servers from the Server View.

__ a. From a command prompt, telnet to the remote system if needed:

‘**telnet <HOSTNAME> <TELNET_PORT>**’

userid : **<USERID>**

pw : **<PASSWORD>**

__ b. Navigate to the bin directory for the profile being used:

cd <WAS_HOME>/profiles/<PROFILE_NAME>/bin

__ c. Run the command file to start the server: **./startServer.sh <SERVER_NAME>**

__ d. Wait for status message indicating server has started:

```
ADMU3200I: Server launched. Waiting for initialization status.

ADMU3000I: Server cl1sr01 open for e-business; process id is 0000012000000002
```