Lab 2: Hello, World Web Service

In this lab, you will write the "Hello, world" of Web services to get you started in Web services development.

This lab will be much the same as the first lab, but you will create, run and test within the RAD development environment instead of at the command line.

Objectives:

- To write a simple Web service
- To become familiar with the Web service development process
- To write a simple standalone Web service client

Part 1: Creating a Web Service

Steps:

_1. Ask your instructor for the following information:

Lab Installation directory: _____

RAD Installation directory: _____

If your machine has the standard lab setup on Windows XP, the information will be something like:

Setting	Value
Lab Installation directory:	c:\websrvclass
RAD Installation directory	C:\Program Files\IBM\SDP

If your computer is running Windows Vista, Windows 7 or later, the standard directories are:

Setting	Value
Lab Installation directory:	c:\Users\username\websrvclass
RAD Installation directory	C:\Program Files\IBM\SDP

Please substitute your actual user name, e.g. *libtrain1* for the "username" above.

_2. Start RAD from by selecting *IBM Software Development Platform - Rational Application Developer Standard Edition* the Windows Start menu.

Note: If your computer has Windows Vista, Windows 7 or later, right-click on the RAD Start menu entry and choose *Run As Administrator*, then OK the access.

When prompted for the *Workspace Location*, enter the **{Lab Installation Directory}\workspace**, for example, on Windows XP:

```
c:\websrvclass\workspace
```


Copyright © Descriptor Systems 2014. All Rights Reserved. Materials may not be reproduced in any form without prior written consent of Joel Barnum.

On Windows Vista, Windows 7 or later:

```
c:\Users\username\websrvclass\workspace
```

Please substitute your actual user name, e.g. *libtrain1* for the "username" above.

You can close the *Rational Software Development Platform* welcome window if you like by clicking the 'X' in its title-bar tab.

_3. Next, you need to configure RAD for our labs. Follow these steps:

a. From the RAD menu, choose *Window - Preferences* to display the Preferences property sheet.

b. In the left-hand navigation tree, expand the *General* category and click on *Capabilities*.

c. At the bottom of the *Capabilities* list, click the *Enable All* button, then press the Apply button.

d. At this point, you can examine and modify other RAD preferences. For example, you can change the editor font by navigating to *General - Appearance - Colors and Fonts - Basic - Text Font*

Then press Apply and OK to save your changes and close the Preferences.

_4. RAD supports the notion of *perspectives*, which are sets of window panes (a.k. a *Views*) that let you work in a particular role, e.g. Java coder, Web developer, server administrator and so forth. For this lab, you will work in the *Web* perspective, which contains window panes that let you create, edit and preview JSPs and HTML pages as well as work with Web services.

Open the *Web perspective* by choosing **Window - Open perspective - Web**.

_5. Your first job is to create a *project* for the lab. Follow these steps:

a. Choose **File - New - Dynamic Web Project** to start the wizard.

b. On the first wizard page, enter **lab02Web** for the *Name*. For the *EAR Project Name*, CHANGE the suggested name to **lab02EAR**.

Press Finish.

You can close the Quick Starts window.

_6. Examine the *lab02Web* project in the *Enterprise Explorer* window pane by clicking on the "+" symbol to expand it. Note the following:

- The *Java Resources/src* folder is where you put your Java packages and code including servlets and JavaBeans.
- The *WebContent* folder is where you put HTML, JSPs and so forth.

_7. RAD makes it easy to expose Java methods as RPC Web services. Follow these steps to write a class containing a method that you will configure as an RPC:

a. In the Project Explorer, right-click on the *lab02Web* project and choose *New - Class*.

For the *package*, enter **service** and for the *Name*, enter **HelloWorld**, then press Finish.

b. Within the new class, write a method that accepts a String and returns a String:

```
public String hello(String name)
{
    return "Hello, " + name;
}
```


Copyright © Descriptor Systems 2014. All Rights Reserved. Materials may not be reproduced in any form without prior written consent of Joel Barnum.

c. Above the class definition, write an *annotation* to indicate that this class contains one or more Web service methods:

```
@WebService
public class HelloWorld ....
```

d. From the RAD menu, choose *Source - Organize Imports* to import the type for the annotation.

e. Press Ctrl+S or choose *File - Save* to save and compile your class. Look in the *Problems* tab at the bottom of the RAD window to ensure that you have no compile errors.

_8. Now you can deploy your Web service: In the RAD *Servers* tab at the bottom of the window, right-click on the WebSphere Application server and choose *Add and Remove* and move *lab02EAR* to the Configured list.

a. In the RAD *Servers* tab at the bottom of the RAD window, right-click on the WebSphere server and choose *Start*, then wait for the server to start completely.

b. Right-click on the WebSphere Application Server again and choose *Add and Remove* and move *lab02EAR* to the Configured list.

_9. Next, test the new Web service. Follow these steps:

a. In the Enterprise Explorer, expand the *lab02Web* project and then the *Services* folder.

b. Right-click on the *{http:service/}HelloWorldService* entry and choose *Test with Web Services Explorer*.

c. The Web Services Explorer (WSE) is a generic Web services client provided by RAD that lets you easily test your services. **Note:** We recommend that you double-click on the WSE titlebar tab to maximize the view. You can restore it by double-clicking on the tab again.

d. In the WSE right-side pane, there's a section labeled *Operations* that contains a clickable list of the methods exposed as Web services -- in this case, only the *hello* method.

Click on the *hello* link. Then press the *Add* link and enter your name to provide the argument to your Web service method.

Press the *Go* button to invoke the service -- you should see the returned string in the *Status* pane. Does it match your expectations?

e. The WSE also lets you examine the raw SOAP messages that make up an RPC. Double-click on the *Status* pane's titlebar to make it bigger, then press the *Source* link in the pane's upper-right corner to display the SOAP request and response messages.

Don't worry too much about syntax, but see if you can find the parameter string that you entered in the request message and returned string in the response message.

f. You can double-click on the Status pane titlebar again to restore it and press the *Form* link in the upper-right corner to see just the returned string.

You can close the Web Browser for the WSE when you are finished experimenting.

Part 2: Writing a Standalone Client

In this part, you will write a Java program that will act as a client to your Web service.

Steps:

_1. Create a new *Java Project* for the client. Follow these steps:

a. Choose **File - New - Project** to start the wizard.

Copyright © Descriptor Systems 2014. All Rights Reserved. Materials may not be reproduced in any form without prior written consent of Joel Barnum.

b. On the first wizard page, choose **Java Project**, then press Next.

c. Enter **lab02Client** for the *Project name*, then press Finish.

When RAD asks you if you want to change to the Java Perspective, press the No button.

_2. Next, generate a Java *proxy* for your Web service into the new project. A proxy or stub is a client-side Web services "stand-in" for the remote service. Follow these steps:

a. Back in the *lab02Web* project, right-click on *Services/{http:service/}HelloWorldService* and choose *Generate - Client* to start the wizard.

b. On the first wizard page, examine the defaults, noting that the *Client proxy type* is *Java proxy*.

In the *Configuration* section, click the link for *Client project: lab02Web* and change the *Client Project* to **lab02Client**, then press OK to go back to the wizard.

WARNING: Be sure to do this correctly! The Client Project MUST be **lab02Client**, NOT *lab02Web*.

Press Next.

c. On the next wizard page, note the defaults, then press Finish to complete the wizard.

d. In the Project Explorer, expand the *lab02Client* project and look in the *service* package -- these generated files comprise the Java *proxy* for your Web service.

_3. Next, create a class with a *main* method for the Java client. Follow these steps:

a. Right-click on the *lab02Client* project and choose **New - Class**.

b. In the wizard, for the *Package*, enter **main** and for the *Name*, enter **Client**. Ensure that *public static void main(String[] args)* checkbox is selected, then press Finish.

_4. Complete the *main* method to invoke the Web service via the generated proxy:

a. Instantiate the proxy factory and retrieve the proxy object itself:

```
HelloWorldService service = new HelloWorldService();
HelloWorld stub = service.getHelloWorldPort();
```

Use *Source - Organize imports* to get rid of compile errors about "XXXX cannot be resolved".

b. After the last line you added in main, add the following code to invoke the Web service and print the results to the console:

```
System.out.println(stub.hello("xxxx"));
```

Substitute your name for the xxxx.

c. Press Ctrl+S to save and compile. Look in the Problems tab at the bottom of the RAD window and fix any errors before continuing.

_5. To run your client, in the Project Explorer, right-click on on HelloWorldClient.java and choose *Run As - Java Application*. You should see the returned String from the Web service RAD's Console pane.

Part 3: Re-deploying

In this part, you will learn the procedure to modify and re-deploy a Web service.

Steps:

Copyright © Descriptor Systems 2014. All Rights Reserved. Materials may not be reproduced in any form without prior written consent of Joel Barnum.

_1. In the *lab02Web* project, edit HelloWorld.java and modify the *hello* method's return so it looks like:

```
return "Hello, " + name+ ", the time is: "
   + new java.util.Date();
```

Be sure to save the class before continuing.

_2. Near the bottom of the RAD window, find the *Servers* tab and click on it .

Right-click on the WebSphere Application Server and choose *Publish*. This ensures that the server is in synch with the changes you made to the Web service.

_3. Use the same techniques you learned in the first part of lab to test your Web service using the Web Services Explorer. You should see the updated return String that includes the current date and time.

Close the Web Services Explorer window when you are finished.

_4. Run your standalone Java client in the *lab02Client* project -- you should see the updated string that includes the current time and date in the RAD console.

_5. From the RAD main menu, choose *File* - *Close All* to close all of the editor windows.

Copyright © Descriptor Systems 2014. All Rights Reserved. Materials may not be reproduced in any form without prior written consent of Joel Barnum.

Copyright © Descriptor Systems 2014. All Rights Reserved. Materials may not be reproduced in any form without prior written consent of Joel Barnum.