

Building and Using Web Services with JDeveloper

Web Services provide client neutral access to data and other services. JDeveloper allows you to create different types of Web Services quickly and easily....

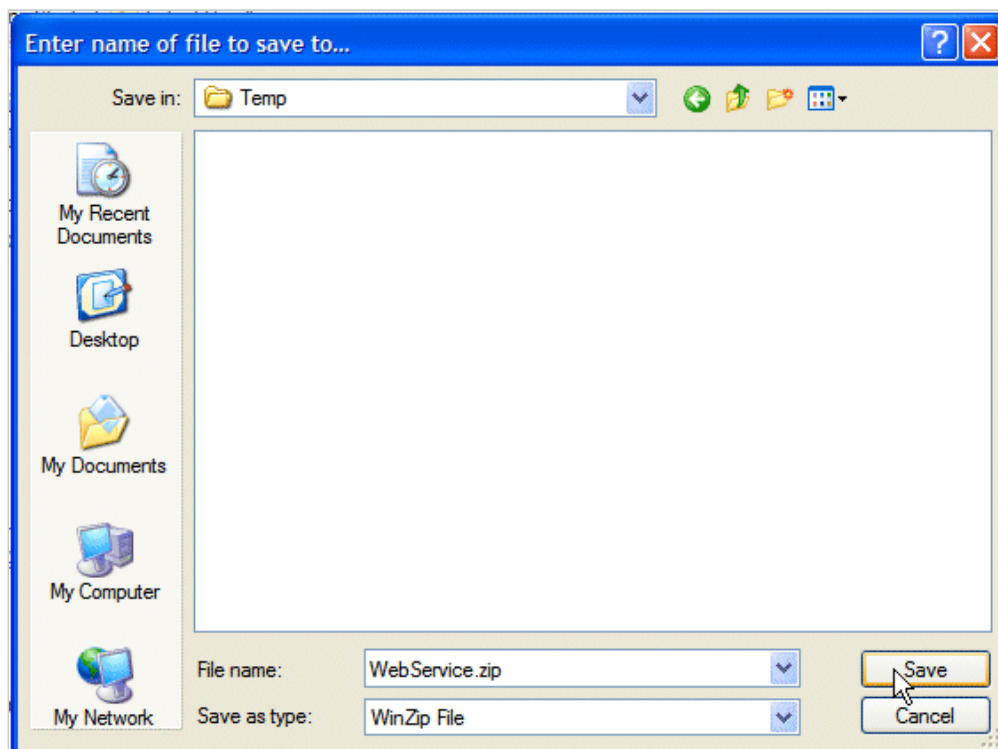
Purpose	Duration	Application
<p>This tutorial shows you how to build and consume Web Services. The tutorial shows several end-to-end scenarios for creating web services. After you develop several web services, you create a client application that uses those services.</p> <p>To see the complete application you will create, click the Download button to download a zip of the completed application, and then unzip it in a temporary folder of your choice.</p>	4 hours	Download

Part 1: Building a POJO Annotation-Driven Service

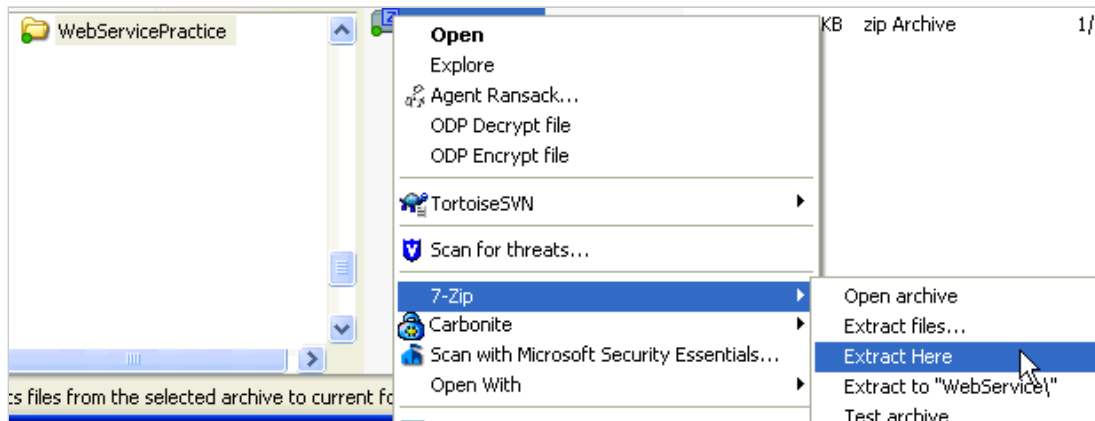
In this first part of the tutorial, you install the required lab files, start JDeveloper, and open the startup application and project.

Step 1: Getting Ready

1. Download the lab files and save the WebService.zip file in the directory you will use for this tutorial (such as d:\Temp.)



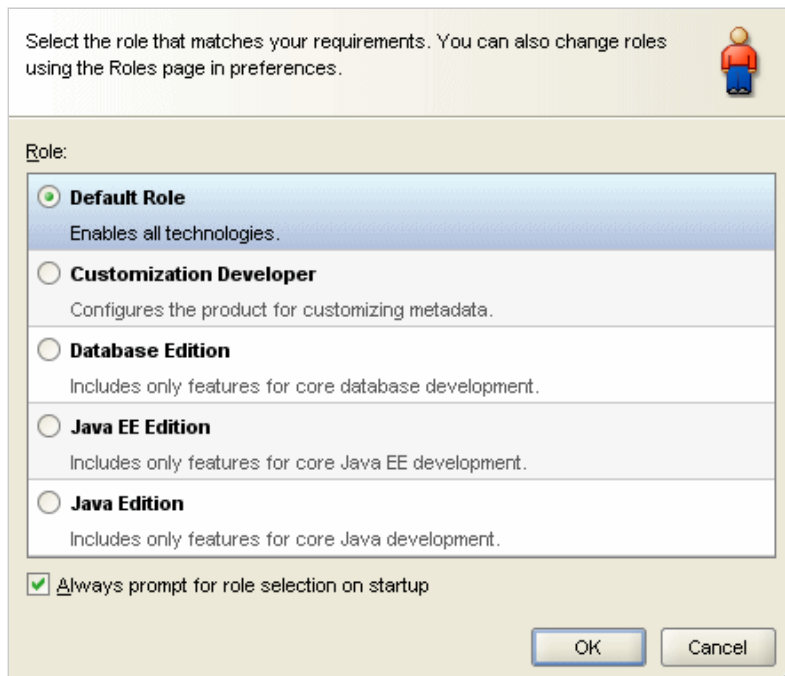
2. Open the directory where you saved the file, right-click the WebService.zip and extract the file. Use WinZip or whatever zip utility you have.



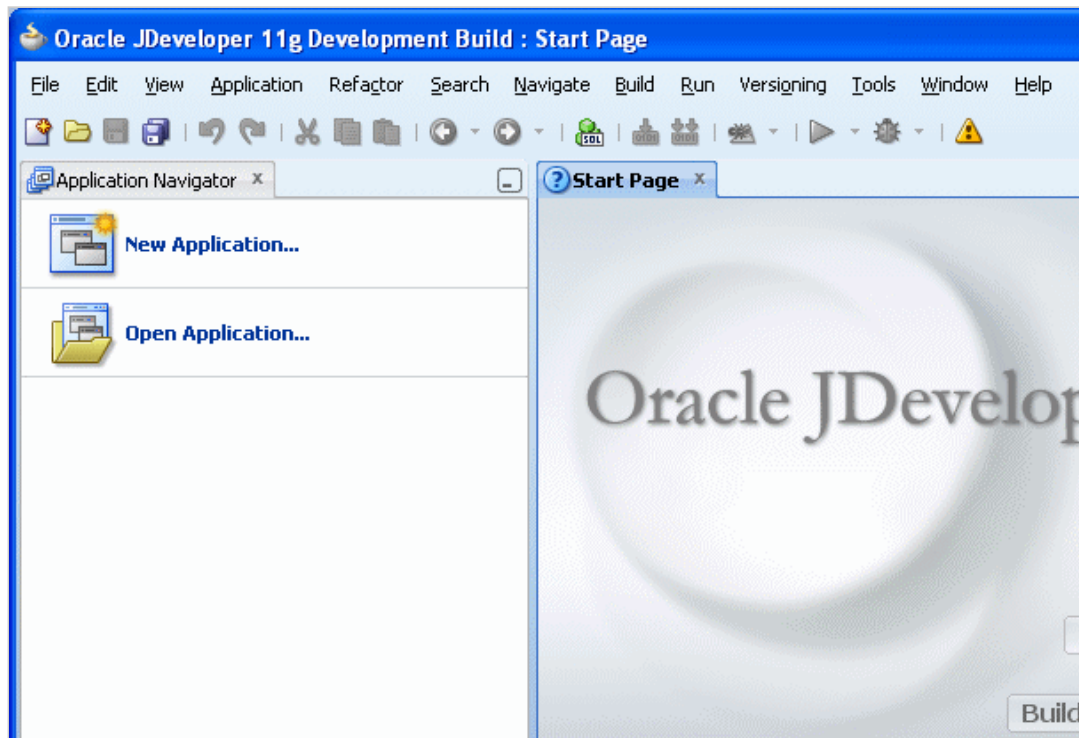
3. Start JDeveloper by selecting **Start > Programs > Oracle Fusion Middleware 11.1.2.0.0 > JDeveloper Studio 11.1.2.0.0**



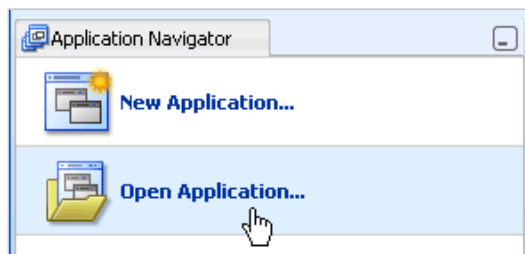
4. If the Migrate User Settings dialog opens, click **No**.
5. If prompted for a User Role, choose **Default Role**.



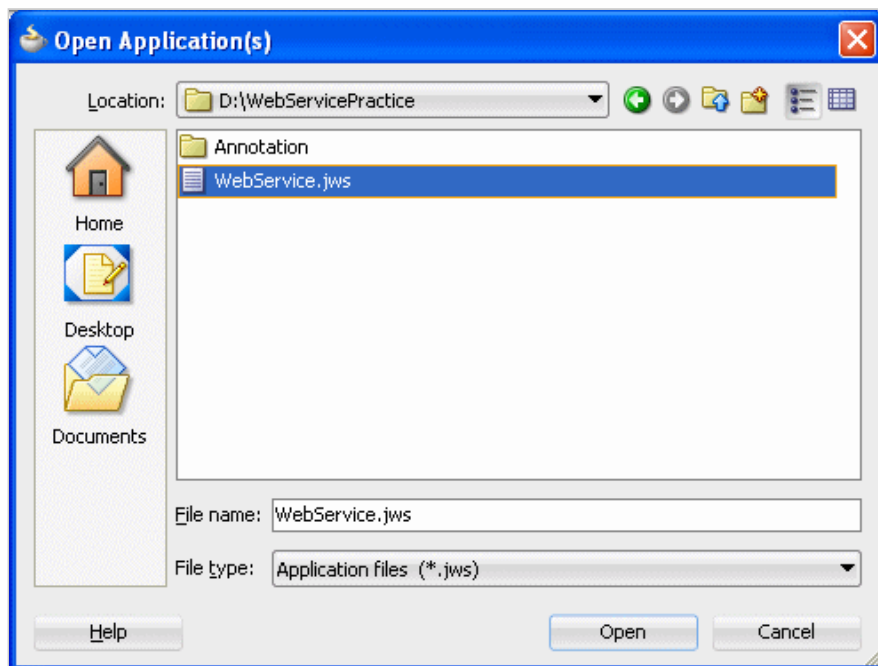
6. If the Tip of the Day window opens, click **Close**.
7. You should now see the JDeveloper IDE. Close the **Start page** by hovering your mouse over the tab and clicking the **X** on the tab.



8. Select the **Application Navigator** tab and click **Open Application** (alternatively, you can select **File | Open**)

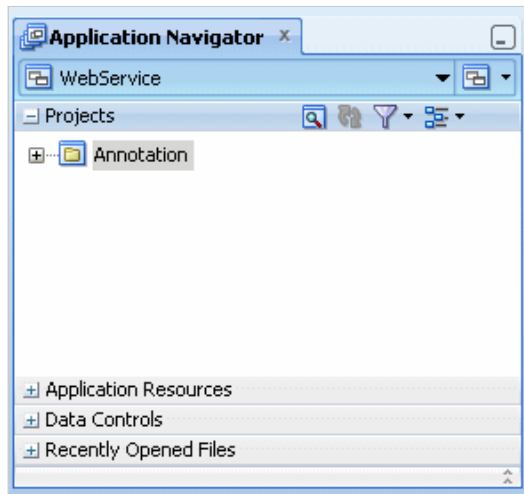


9. In the Open Application dialog box, locate the Web Service directory created when unzipping the WebService.zip file and select **WebService.jws**.



10. Click **Open**

The Application Navigator should look like this:



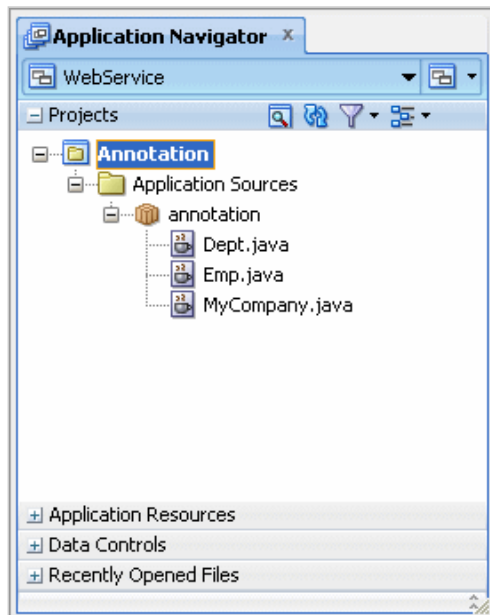
Step 2: Adding a Plain Old Java Object (POJO) to contain a Web Service Method

In this section you start with a project that contains plain old Java classes and add an annotated method that you publish as a web service.

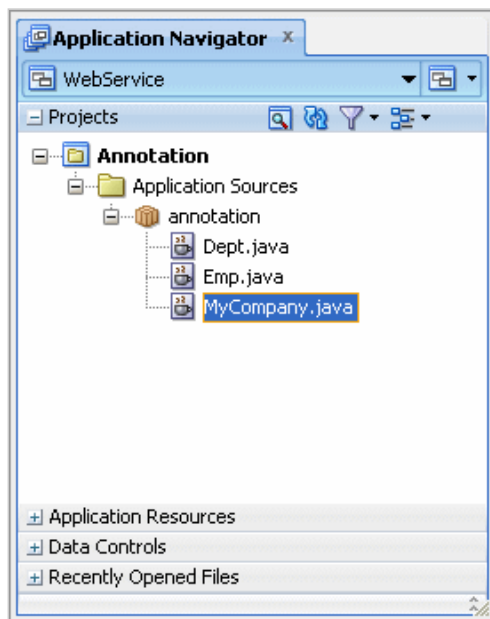
Web service annotation is a feature of J2EE which takes complexity out of creating and deploying Web Services.

1. In the Application Navigator, expand the Annotation project nodes to show the POJO classes:

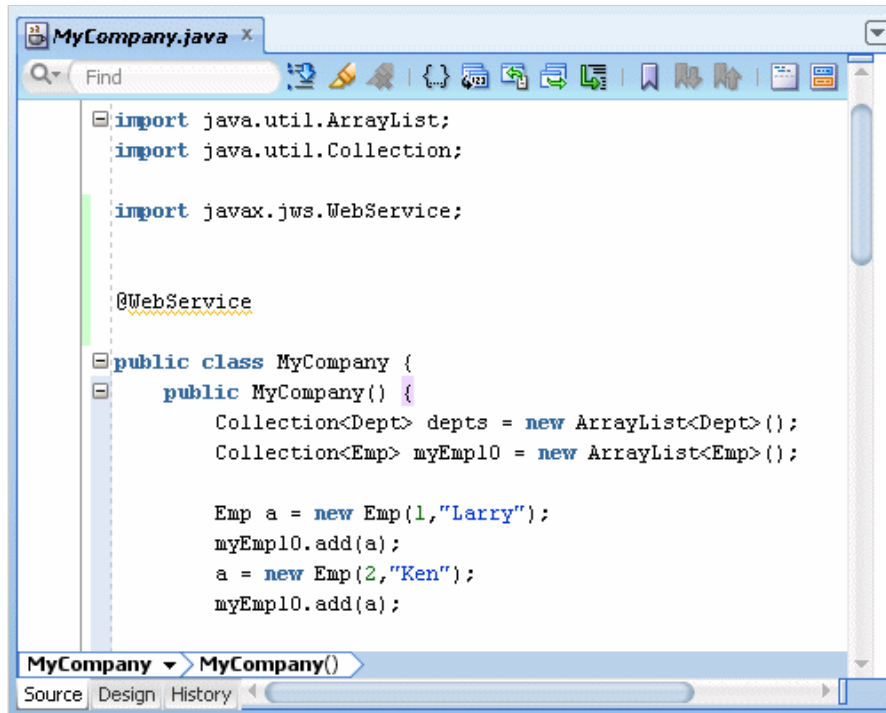
- Dept.java describes the department structure
- Emp.java describes the employee structure
- MyCompany.java populates information about departments and employees



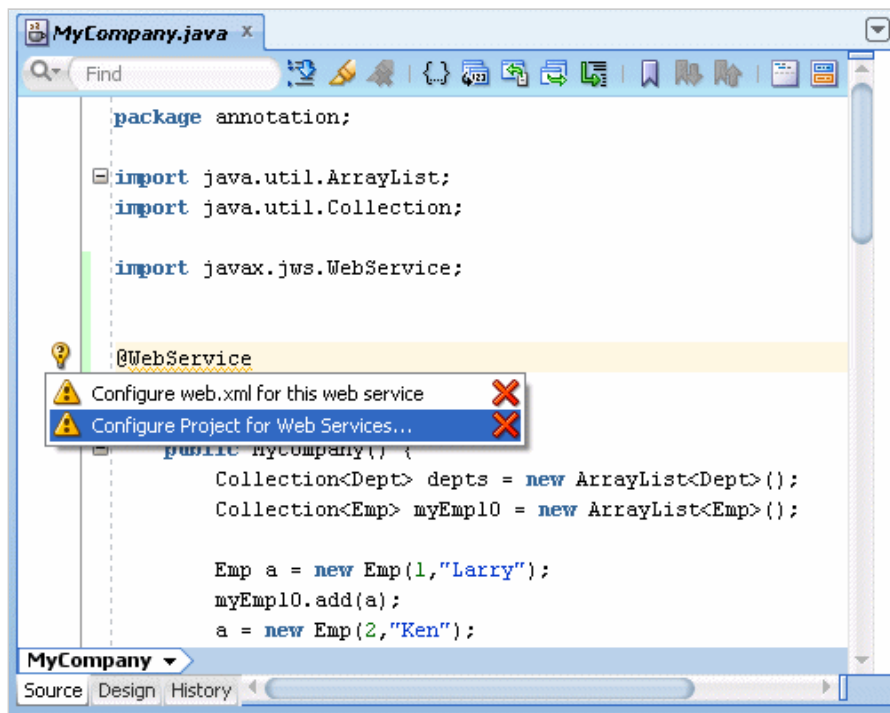
2. In the Application Navigator, double-click **MyCompany.java** to edit it.



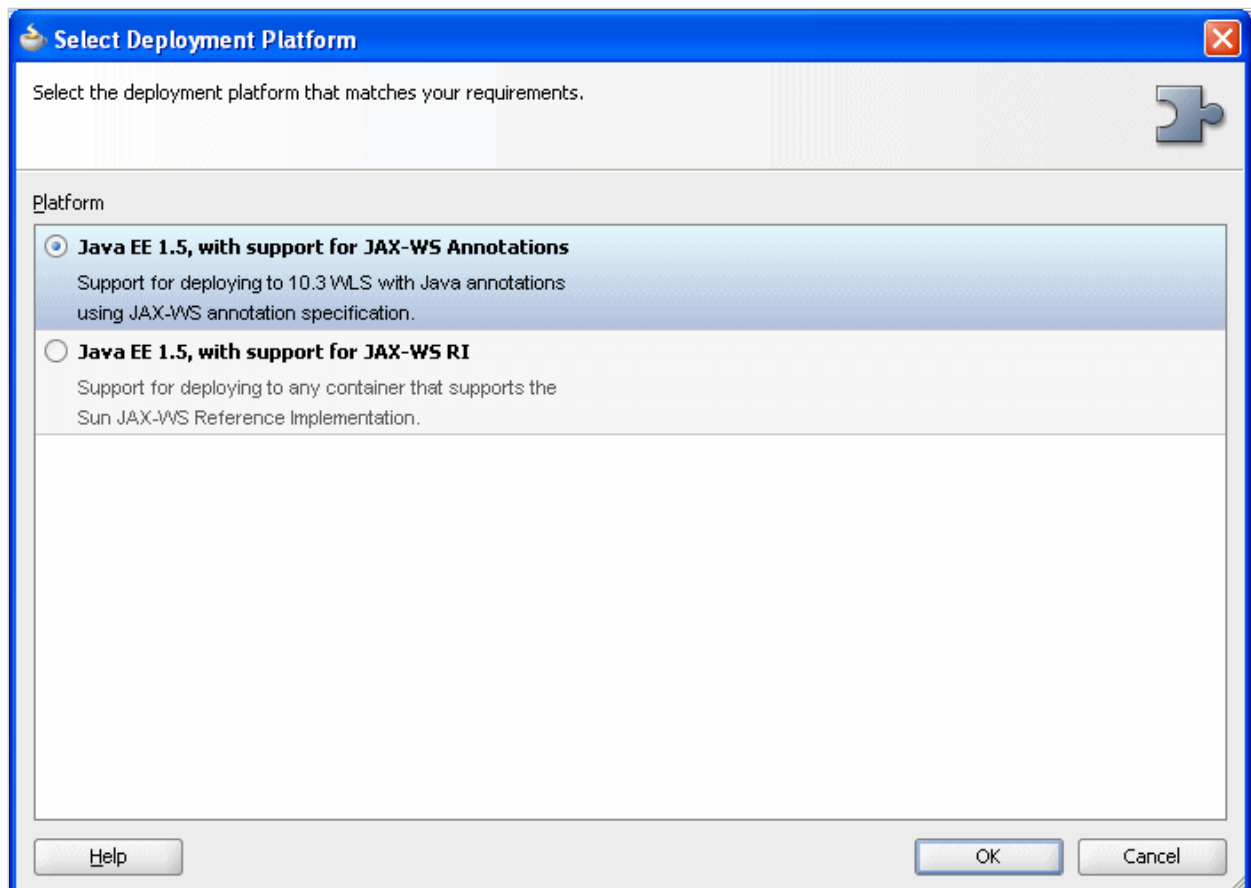
3. Add an **@WebService** annotation after the import statements. The IDE will add the import for the WebService class. This annotation denotes that the class contains a method to be used by a web service.



4. In the margin of the editor, click **Quick Hint** (light bulb icon) and select the **Configure project for web services** option.



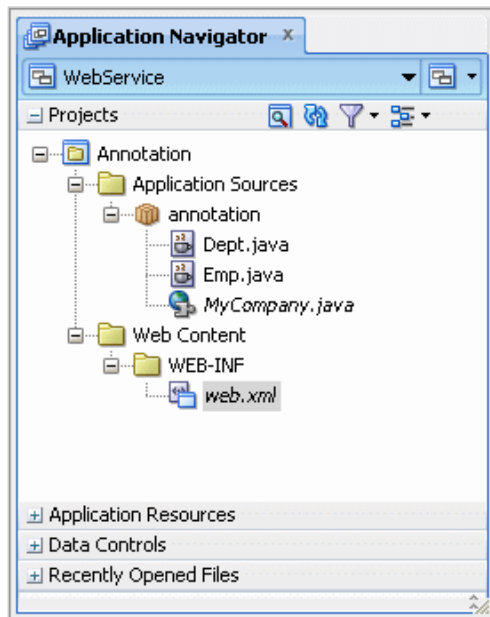
5. In the Select Deployment Platform dialog box, ensure that **Java EE 1.5, with support for JAX-WS Annotations** is selected.



6. Click **OK**. This step adds the javax.jws.WebService import statement to the Java class if it is not already there and creates a web.xml file.

The Application Navigator should look like the following:

Notice that the icon for **MyCompany.java** class is changed to represent a WebService class, and the web.xml file has been added to your project.

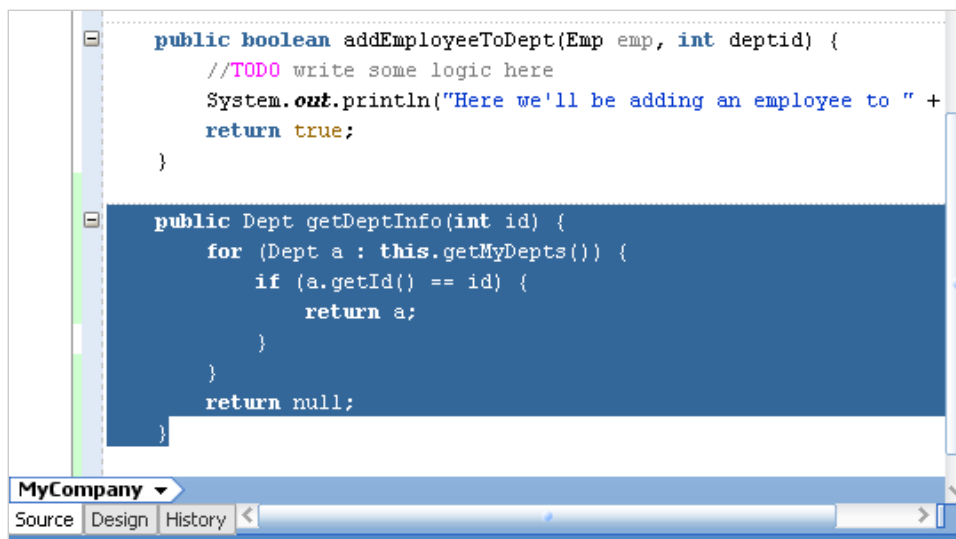


7. Click **Save All**  to save your work.

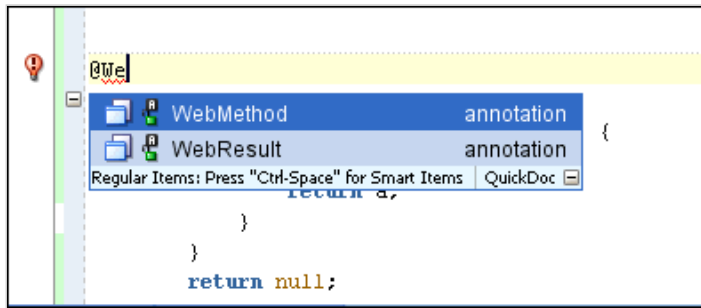
8. In the Code Editor, scroll to the bottom of the class and add the following code statements:

```
public Dept getDeptInfo (int id) {
    for (Dept a: this. getMyDepts() ) {
        if (a.getId() == id) {
            return a;
        }
    }
    return null;
}
```

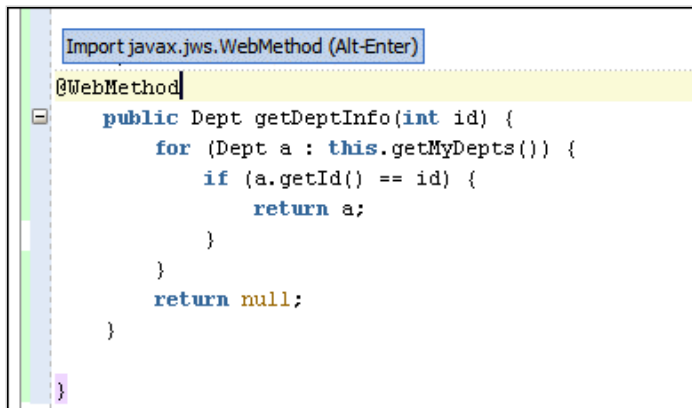
This loop returns information about all employees working in a specific department.
The code in the editor window should look like:



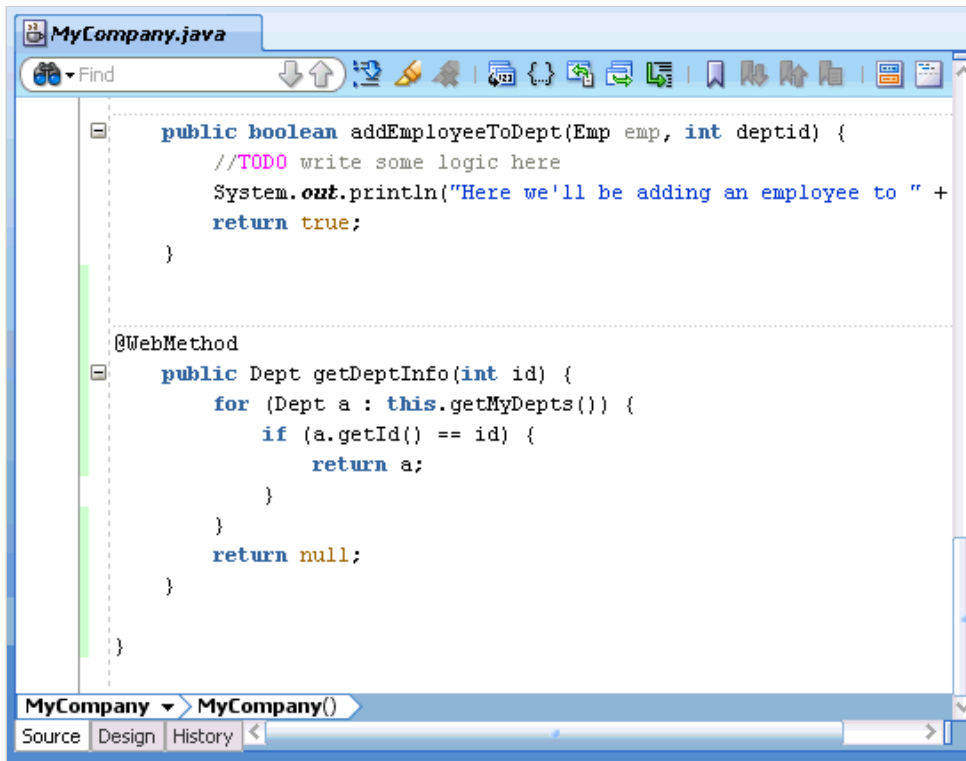
9. Create a second annotation before the **getDeptInfo()** method. The annotation signifies this is the method to be exposed from the web service. Add a blank line above the **getDeptInfo()** method, and start typing **@WebMethod**. Code insight pops up a list of available syntaxes. Select **WebMethod** from the list.




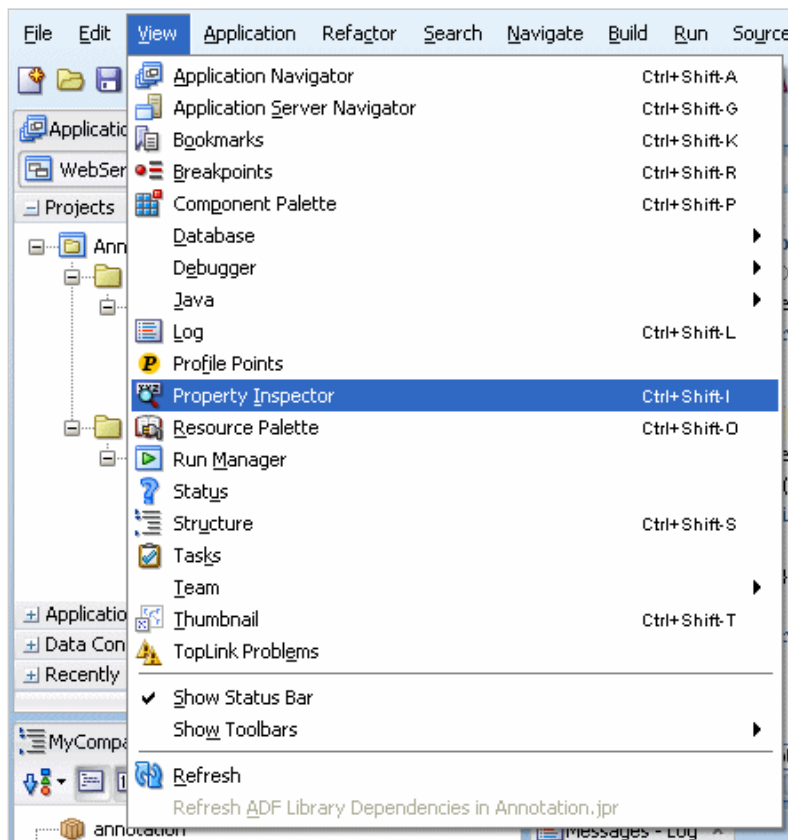
10. If suggested, press **[Alt]+[Enter]** to add the **import javax.ws.WebMethod;** statement (although this statement may be added automatically.)



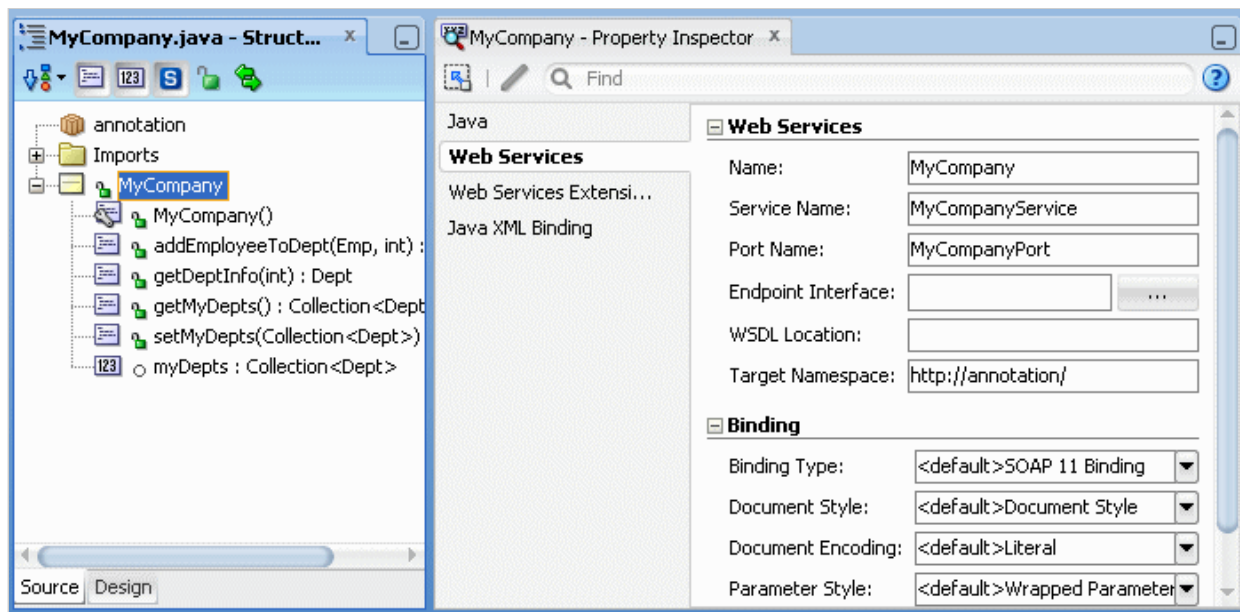
The class should now look like the following:



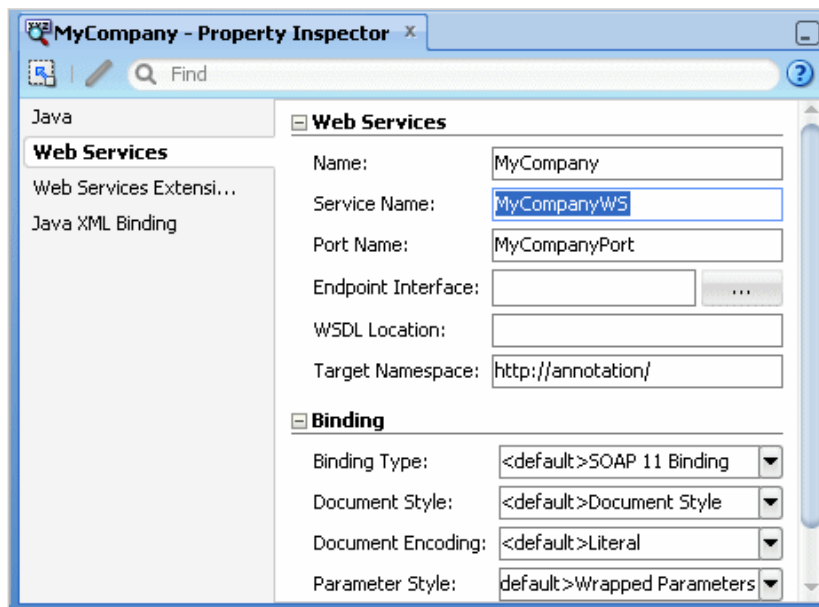
11. Click **Save All**  to save your work.
12. You can use the Property Inspector to modify the characteristics of the class. In the menu bar, select **View | Property Inspector** and it will open as a tab in the bottom portion of the IDE.
Note: If the Property Inspector opens in a different part of the IDE, you can drag its tab and drop it on the bottom panel if you would rather work with it there.



- To display the properties of the MyCompany class in the Property Inspector, select the Source tab at the bottom of the Structure window, then select the top level MyCompany class name.



- The Property Inspector displays a few finger tabs on the left side of the window. Select the Web Services tab and notice that the Service Name has the word 'Service' appended to the class name.
- Change the Service Name to **MyCompanyWS**. Notice that the class reflects the name change.



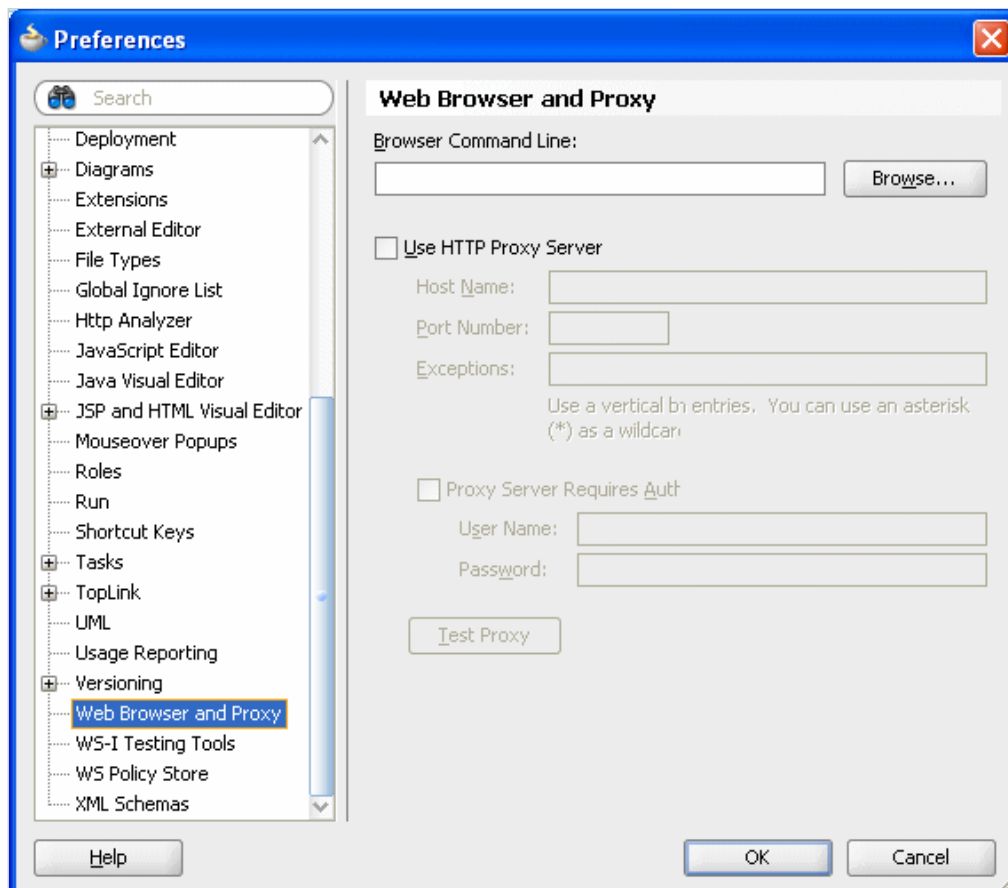
16. Click **Save All**  to save your work.

You have now created a POJO Web Service. In this next section, you will test you Web Service.

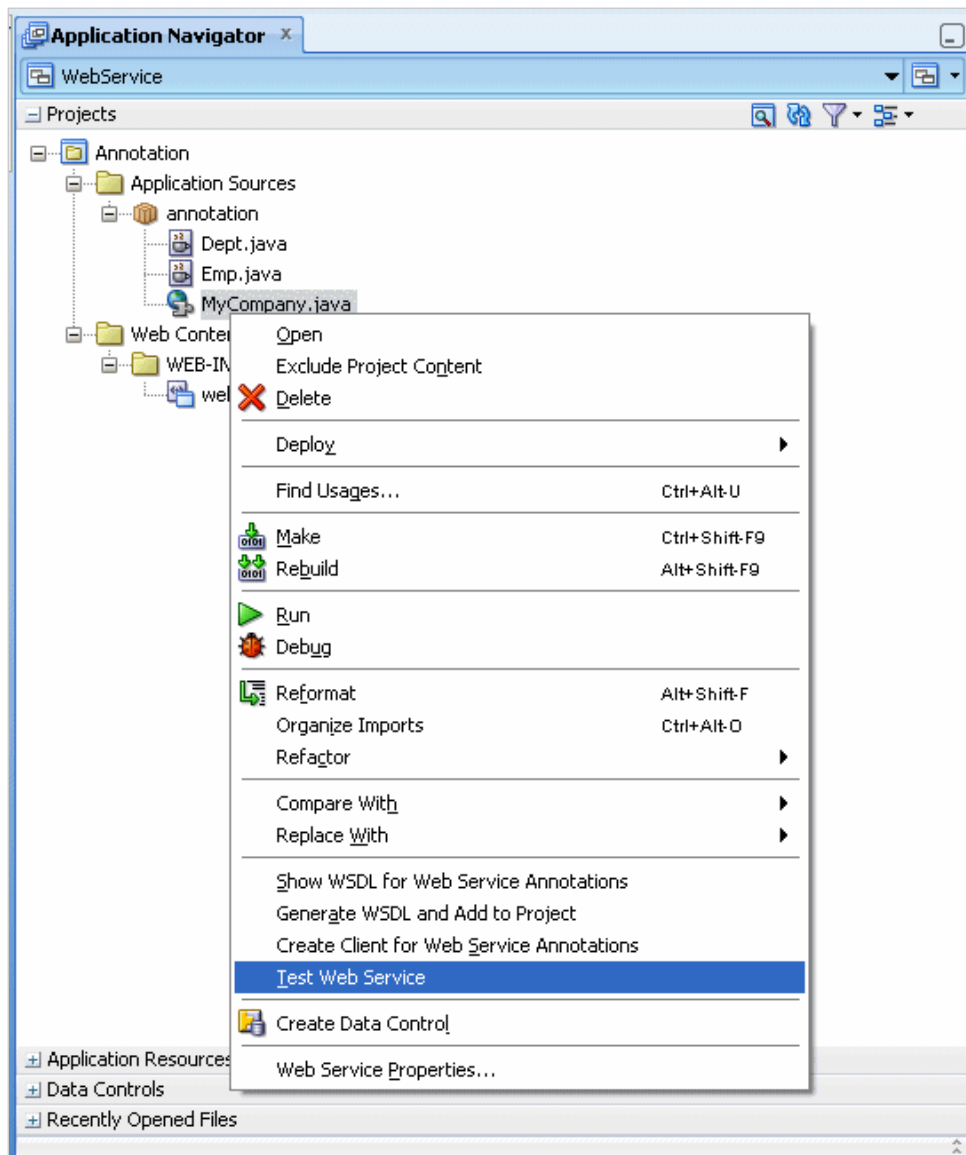
Step 3: Testing a Web Service

In this section you compile, deploy and test the web service using the HTTP Analyzer.

1. Before testing the web service, check that your web browser settings are correct. Choose **Tools > Preferences** and then scroll down the list on the left to select the **Web Browser and Proxy** page. Ensure that the **Use HTTP Proxy Server** check box is not selected, then click **OK**.

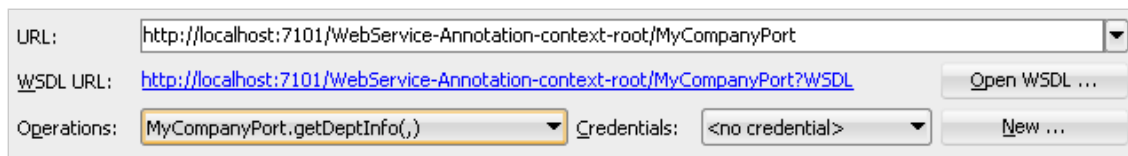


2. In the Application Navigator, **right-click** the **MyCompany.java** node and in the context menu, select **Test Web Service**.

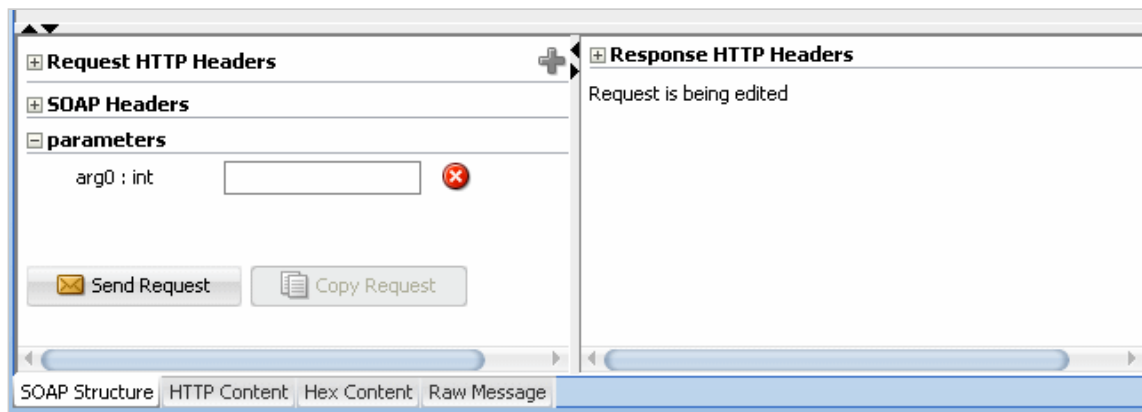


This option invokes the integrated WebLogic Server, deploys the service, and then starts the analyzer. It may take a few seconds to start WebLogic Server if you are running it for the first time. If this is the first time you test a service, Windows may ask you about blocking content. Allow the content to be displayed.

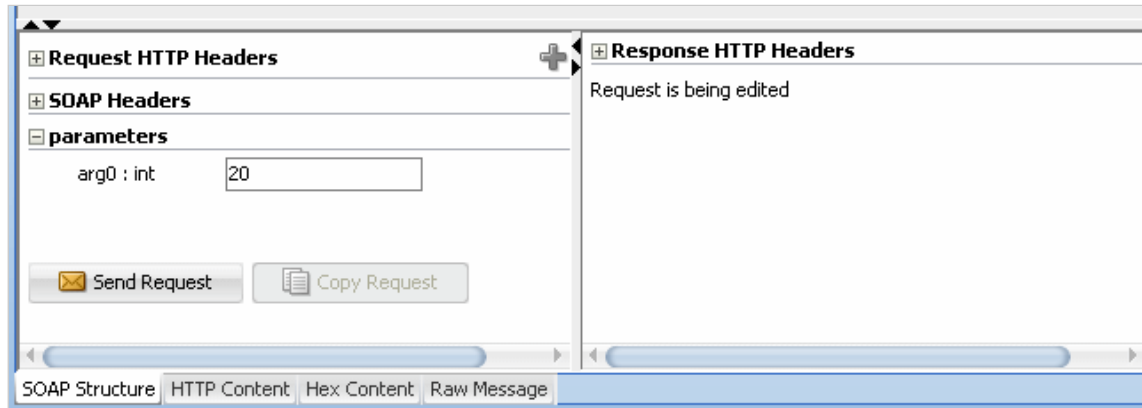
3. The top portion of the HTTP Analyzer editor window displays the URL for the web service, the WSDL URL, and the exposed Operations. Select the **MyCompanyPort.getDeptInfo(,)** operation from the list.




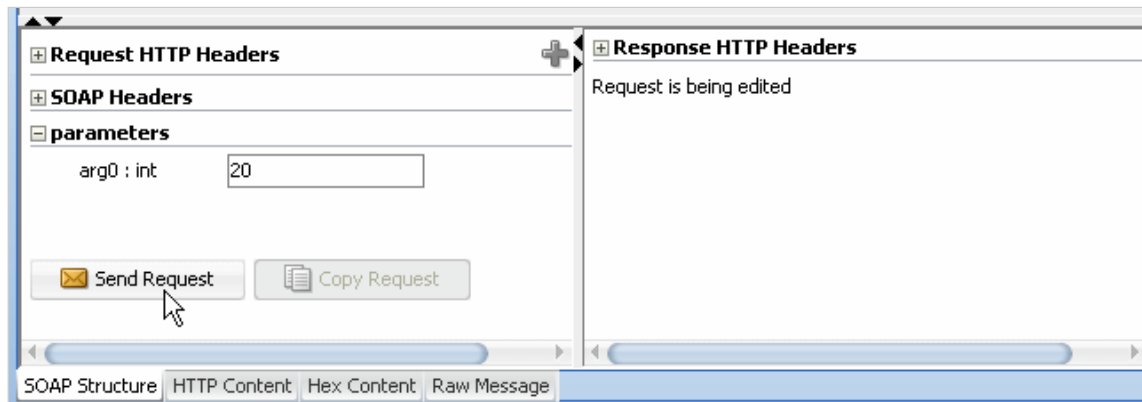
The bottom portion of the analyzer is split into two areas: Request and Response. The request area shows all the arguments from the exposed method (in this case, only one argument.) When the web service is executed, the Response area shows the results.



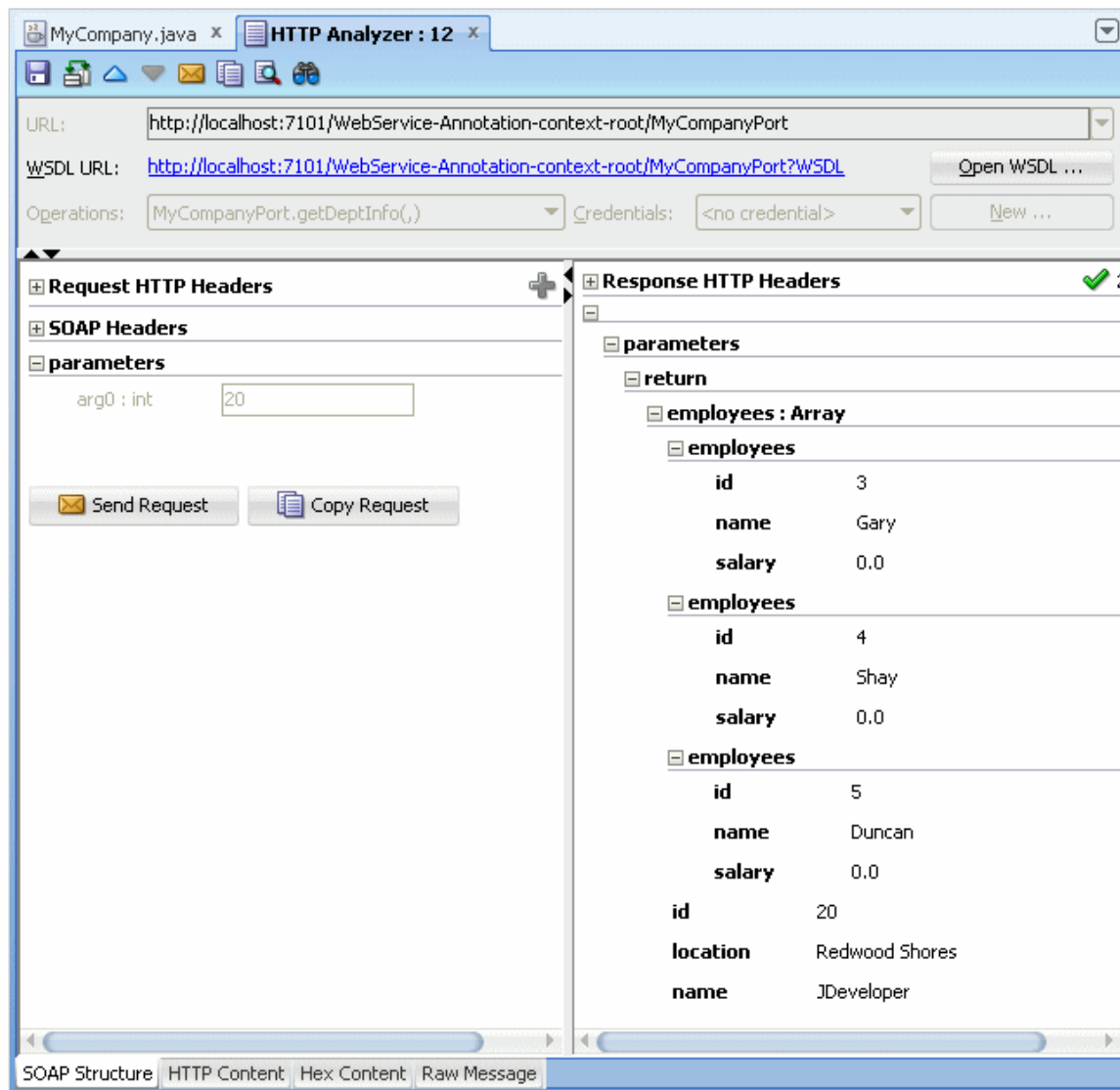
4. In the Request area, enter a department number value (10, 20 or 30) in the **arg0** field.



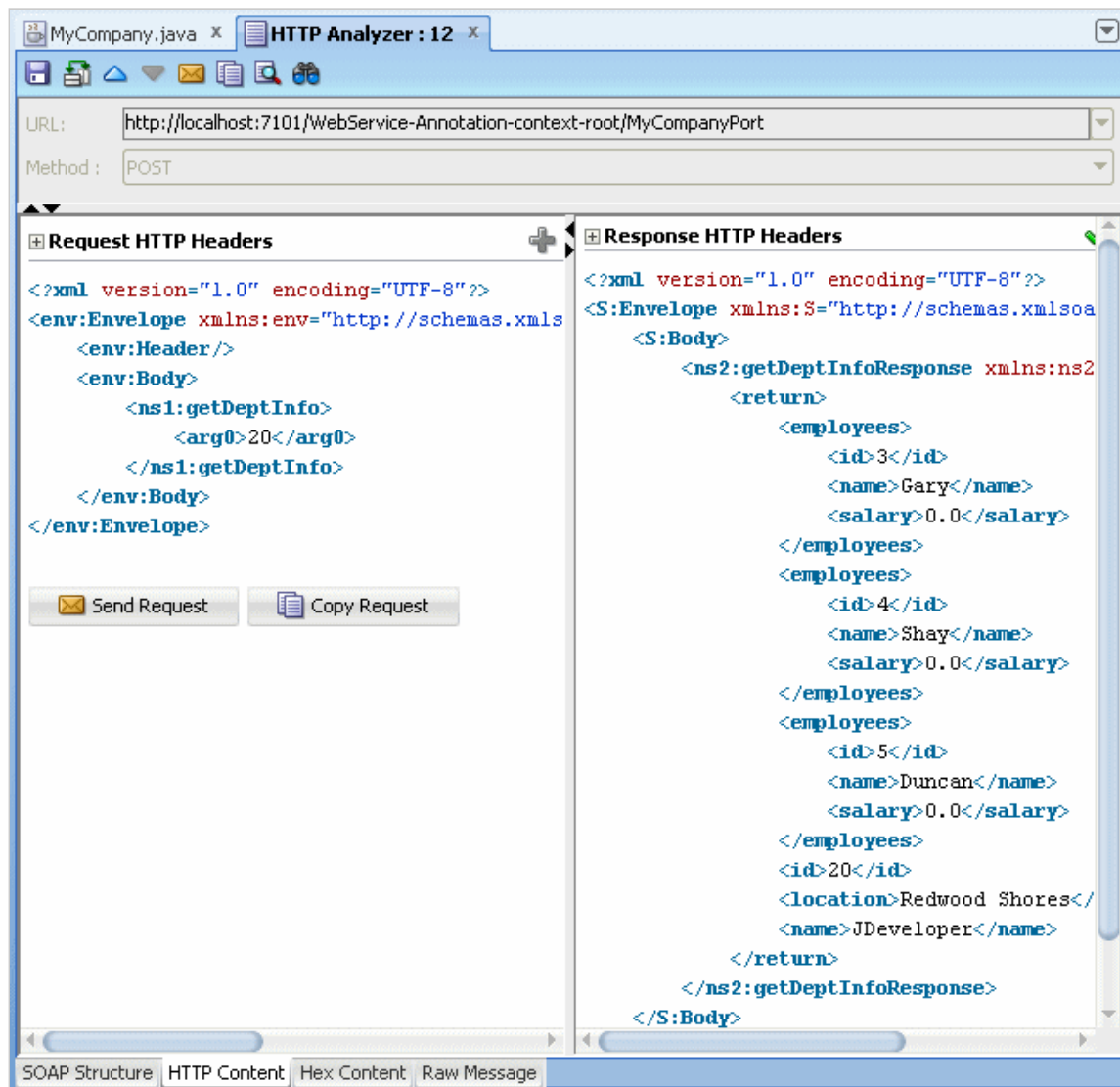
5. In the toolbar area of the analyzer, click Send Request, or click the Send Request button  Send Request below the argument.



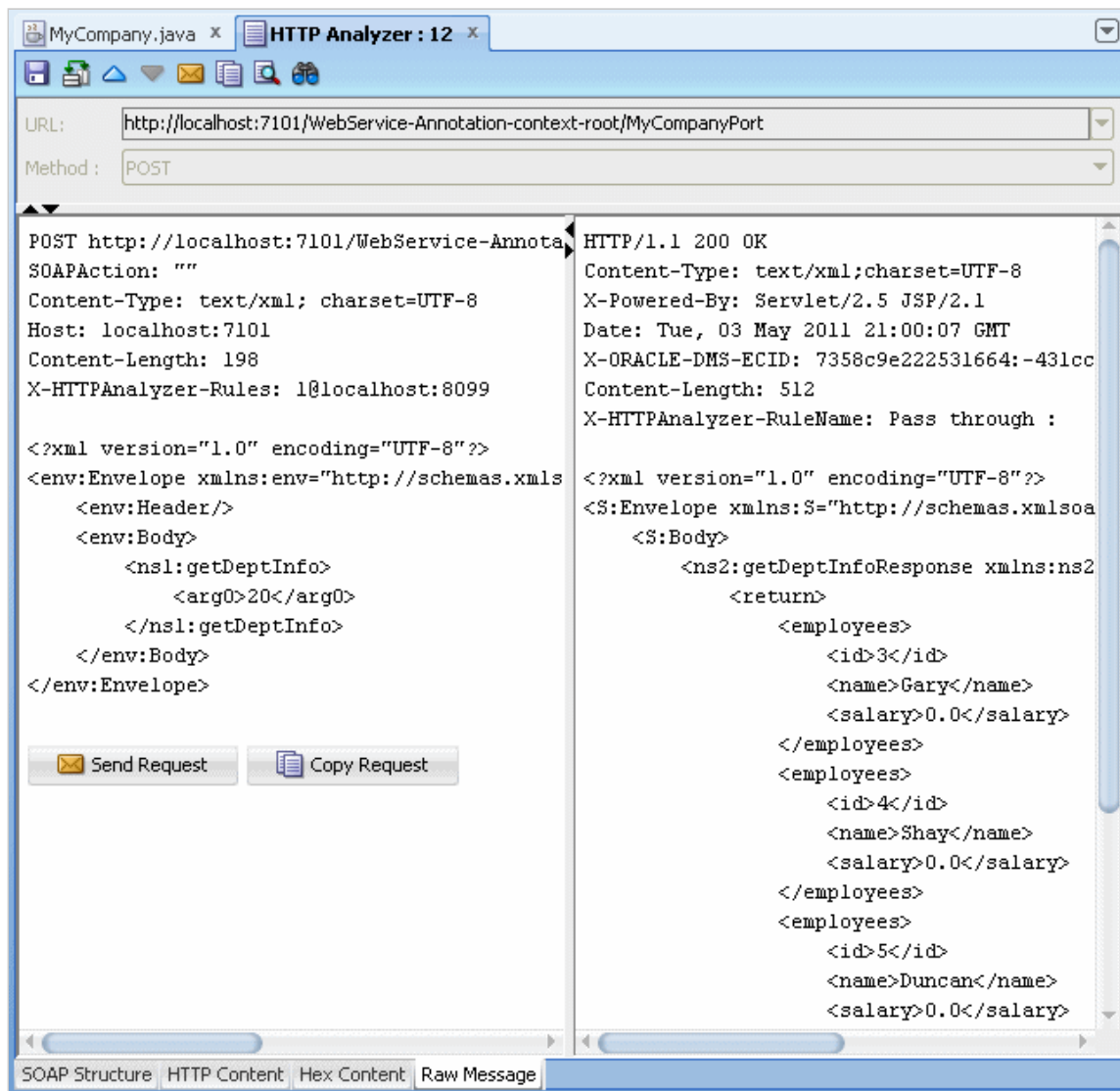
6. The analyzer sends the request to the service, returning after a few seconds the information about employees working in the specified department.



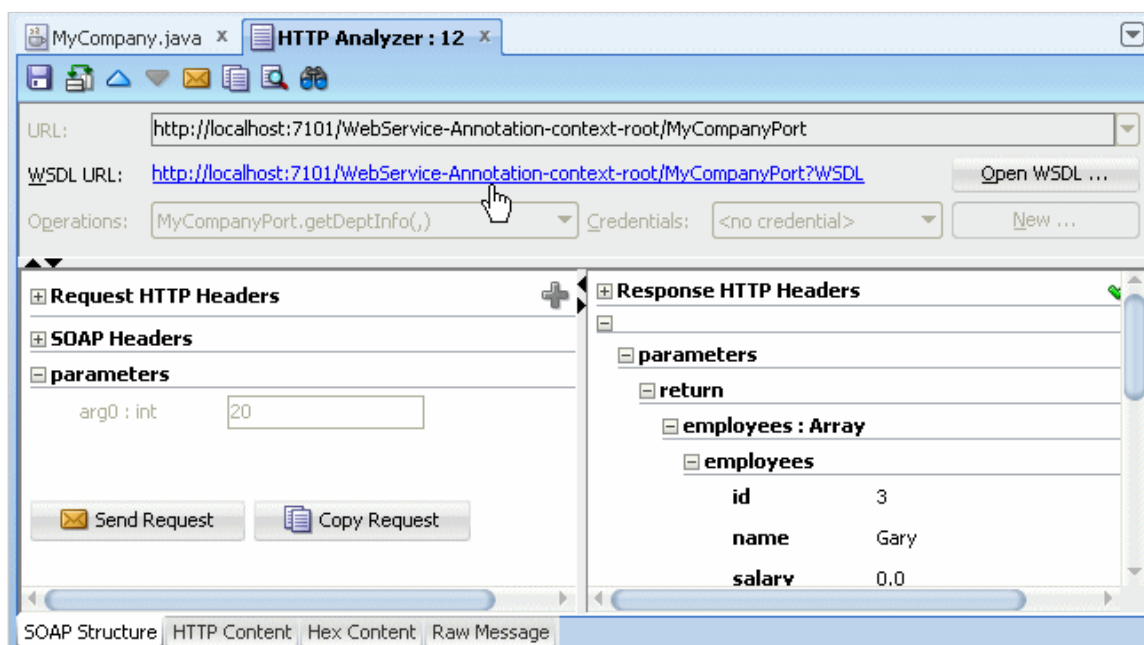
7. Click the **HTTP Content** tab at the bottom of the editor to look at the xml code.



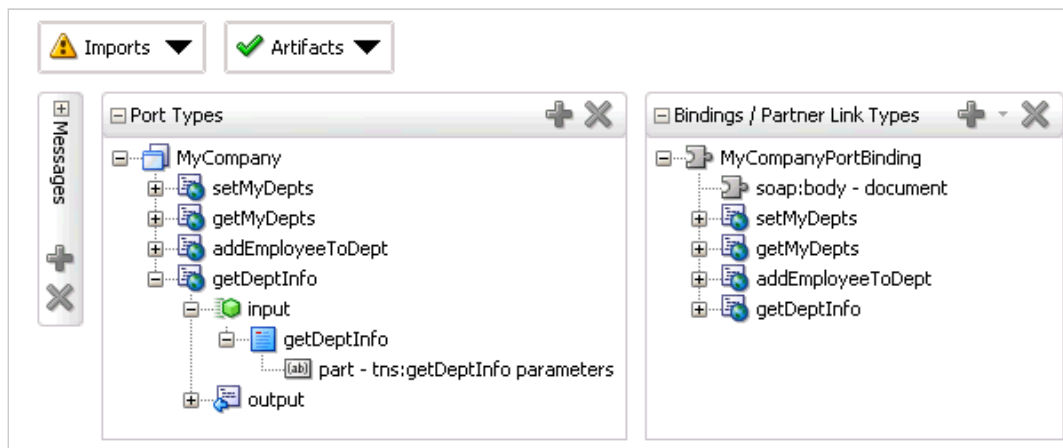
8. Click the **Raw Message** tab at the bottom of the editor for another presentation of the code.



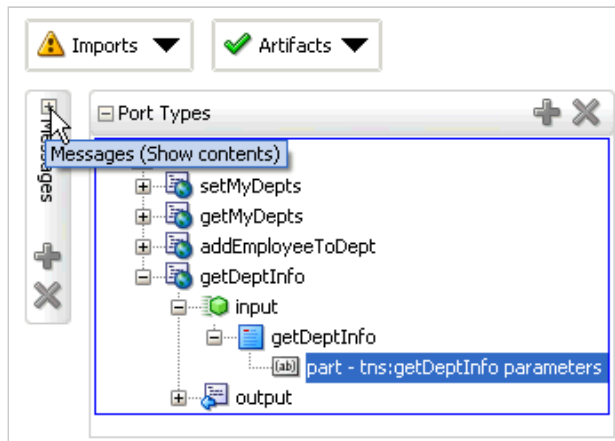
9. Click the **SOAP Structure** tab at the bottom of the editor, and then in the top part of the HTTP Analyzer, click the **WSDL URL** link.



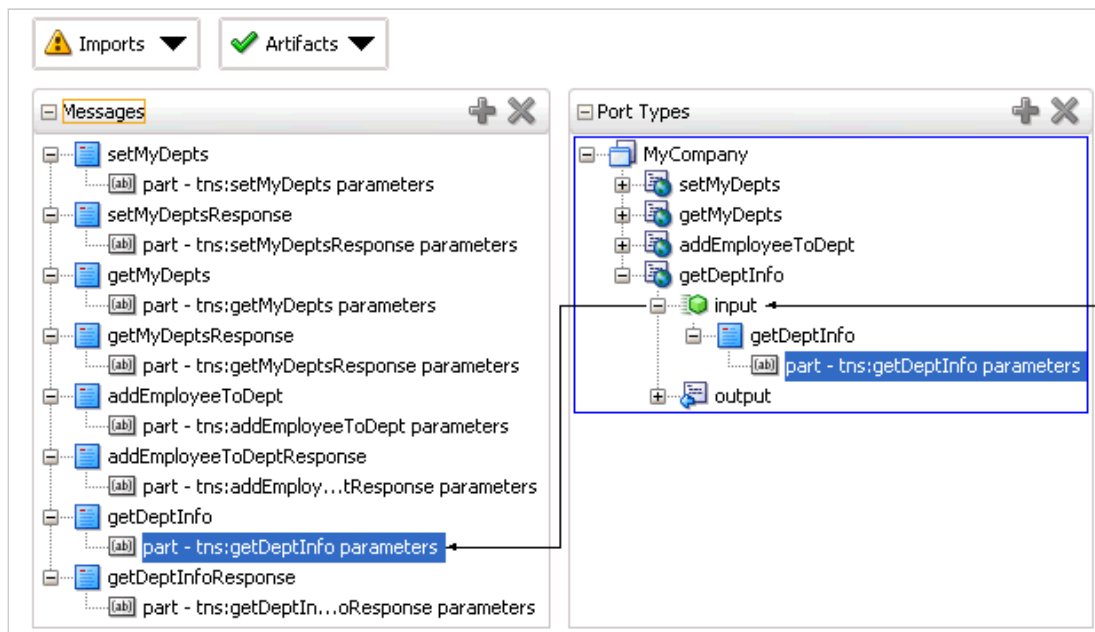
10. This opens the visual editor for the web service. In the Port Types panel, expand the **getDeptInfo | input | getDeptInfo** nodes.



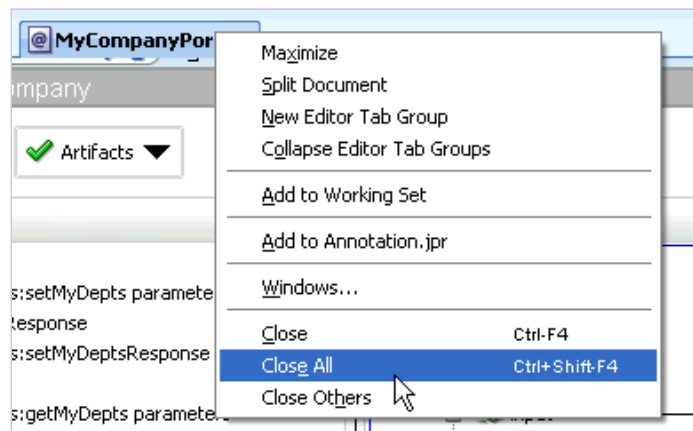
11. To the left of the Port Types panel, click the small **Plus** sign at the top of **Messages** to show message contents.



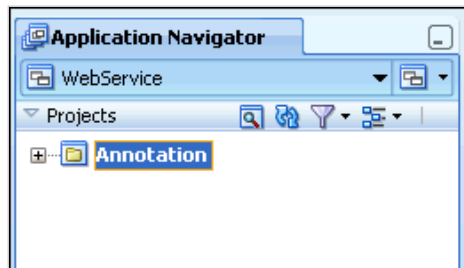
A new graphical representation shows the flow for any message you select.



12. Right-click any tab in the editor window and select the **Close All** option.



13. Collapse the **Annotation** project node in the Application Navigator.



Previous [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [Next](#)