Practical 08 - Build, deploy and test a Calculator WebService and Web interface

Objectives:

- To build a functioning Calculator WebService in Eclipse
- To build a functioning WebSite front end to that WebService
- To publish the WebService and WebSite as an application on TomCat
- To view the WebService WSDL
- Functions:
- The overall system should should provide the following functions:
- Add
- Subtract
- Divide
- Multiply
- Each function should take in two "numbers", perform the operation, and return a single result
- NB: The simplest way to implement this is to have 4 operations, each with two inputs and one output, but if you want get fancier, you could also implement as on operation with 3 inputs (number 1, number 2, function to perform) and one output.

Requirements for tonight's exercises:

- Oracle Java JDK v1.7 (aka Java 7) - www.java.com
- Eclipse JEE Kepler - www.eclipse.org
- Apache Tomcat - www.apache.org
- SOAPUI (free version, not Pro)- www.soapui.org
- ...all installed as per Practical 01.

Instructions:

1. Launch Eclipse
2. Tell Eclipse about the Tomcat server (as already done in Practical 06)
3. Create a new project to hold our application
   3.1. Back at the Eclipse desktop, right click in a blank area of the "Project Explorer" window.
   3.2. Select "New", then "Project"
   3.3. In the "New Project" dialog:
      3.3.1.Locate and open the "Web" folder.

3.3.2. In the "Web" folder, locate and select the "Dynamic Web Project" item.
3.3.3. Click <Next>

3.4. In the "New Dynamic Web Project" dialog:
3.4.1. Enter a project name like "Calculator"
3.4.2. Check that the "Target Runtime" is set to "Apache Tomcat v7.0"
3.4.3. Click <Next>

3.5. At the "Java" dialog, click <Next>
3.5.1. At the "Web Module" dialog:

3.6. Write down the value in "Context Root" here: - NB: Case Sensitive! : _____

3.6.1.1. Ensure that "Generate web.xml deployment descriptor" is selected.
3.6.1.2. Click <Finish>
3.6.1.3. Wait for the Dynamic Web Project to be created

3.7. Back in the "Project Explorer" window:
3.7.1. Locate the dynamic web project we just created, and open it.
3.7.2. Note the folders and other items created to support the project.

4. Create a new Java Class containing our code
4.1. Back at the Eclipse desktop, locate the new Dynamic Web Project in the "Project Explorer" window.
4.2. Open the Dynamic Web Project
4.3. Within the Dynamic Web Project, locate and open the "Java Resources" folder.
4.4. Within the "Java Resources" folder, locate the "src" folder.
4.5. Right click the "src" folder, then select "New" then "Class"
4.6. In the "New Java Class" dialog:
4.6.1. Set the "Package" name ("ie.ipa.soaandws.<username>" is appropriate)
4.6.2. NB: replace <username> above with your username, e.g. jsmith
4.6.3. Set the "Class" name (e.g. "Calculator") write it down here: - NB: Case Sensitive! _____
4.6.4. Leave all the other defaults unchanged.
4.6.5. Click <Finish>
4.7. Back at "src" folder, note that the package and .java file have been created

5. Add code to the .java file:
5.1. In the Package Explorer window, locate the ".java" file.
5.2. Double-click on the .java file to it for editing in the Editor Window.
5.3. The .java file currently contains:

```
package <package name, as set>
public class <class name as set> {
}
```

5.4. Add code so it looks like the following:

```
package <package name, as set>
public class <class name as set> {
public double calculatorAdd( double firstNumber , double secondNumber ){
return firstNumber + secondNumber ;
}
/** work the remaining operations out for yourselves!! */
}
```

5.5. Save the changed code, and validate it.
5.6. Ensure that the validation completes with no errors or warnings before proceeding
5.7. Close the editor window for "Calculator.java"

6. Create a new web service based on this code.
   6.1. In the Package Explorer window, locate the "Calculator.java" file.
   6.2. Right click the ".java" file, then select "New" then "Other"
   6.3. In the "New" dialog:
      6.3.1. Locate and open the "Web Services" folder
      6.3.2. Within the "Web Services" folder, locate then select "Web Service" item.
      6.3.3. Click <Next>
   6.4. Within the "Web Service" dialog:
      6.4.1. The top section refers to the WebService we are creating:
      6.4.2. Note that the "Web service type" is set to "Bottom up Java bean Web Service"
      6.4.3. Check that "Web service type" is set to "Bottom up Java bean Web Service"
      6.4.4. Check that "Service Implementation" is set to the package name and class name you set earlier.
      6.4.5. Check that the top "slider" is set to the value "Start Service"
      6.4.6. The lower section refers to the "client"; we do want a client this time, so set the lower slider to the highest value, i.e. "Test Client"
      6.4.7. Ensure that "Publish the Web Service" is selected.
      6.4.8. Ensure that "Monitor the Web Service" is not selected.
      6.4.9. Click <Next>
   6.5. Within the "Web Service Java Bean Identity" dialog.
      6.5.1. Ensure that all of your operations are present and all are selected
      6.5.2. Note the name of the WSDL file.
      6.5.3. Leave all other settings at default.
      6.5.4. Click <Next>
   6.6. At the "Server Startup" dialog (if TomCat is currently started, the “Server Startup” dialogue will not appear):
      6.6.1. If the server is currently stopped, click <Start server>
      6.6.2. Wait for the server (the Tomcat instance we specified) to start.
      6.6.3. Click <Next>

6.7. At the "Test Web Service" dialogue:
  6.7.1. Note that the "Test Facility" is set to WebServices Explorer"
  6.7.2. Click <Launch> and wait for the WebServices Explorer to Launch
  6.7.3. Use the WebServices Explorer to test the various operations in your WebService.
  6.7.4. When complete, you can close the browser window or tab.
  6.7.5. Back at the "Test Web Service" dialogue:
  6.7.6. Click <Next>
6.8. At the "Web Service Proxy Page" dialogue, click <Next>
6.9. At the "Web Service Client Test", click <Next>
6.10. At the "Web Service Publication" dialogue:
  6.10.1. De-select both options (we're not using UDDI today)
  6.10.2. Click <Finish>
6.11. Note that the "Web Services Test Client" has opened in a Tab in Eclipse
  6.11.1. Copy the URL of this page.
  6.11.2. Open your browser, and paste this URL into a new Tab
  6.11.3. Test the "Web Services Test client"

7. This is a fully functional Web Application, which can be exported and run on another Java Server.
8. Leave Eclipse running - don't close the application!
9. Use your browser to examine the WebService and WSDL
  9.1. Launch a browser.
  9.2. Enter the address of http://localhost:8080/<context root>/services/<class name>
  9.3. Note that there is a dummy "holding" page there.
  9.4. Enter the address of http://localhost:8080/<context root>/services/<class name>?WSDL
  9.5. Note the returned WSDL is based on the code in the .class file
  9.6. In particular, note the type, messages, operations and ports - make sure you can see how these relate to the code.
10. Highlight and copy this URL - also, write it down here: - NB: Case Sensitive!
  10.1. WEB_SERVICE_URL = _____
11. Now, use SOAPUI to write a series of tests to confirm that your application is running correctly, and quickly
12. Use SOAPUI to perform a load test run based on the tests you have just created.