SOAP WebService Using CXF

Creating a Bottom-Up Web Service

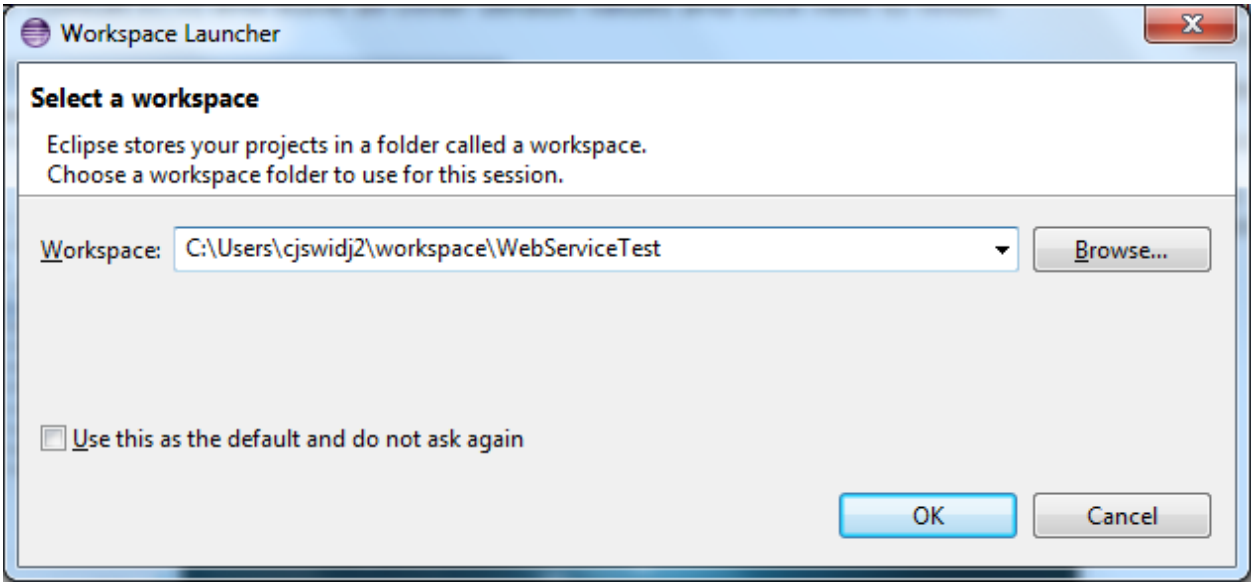
Web services can be created in two ways: Top-Down or Bottom-Up.

A top-down web service starts with the creation of a WSDL (Web Service Definition Language) file, followed by the generation of Java code conforming to the specifications set by the WSDL.

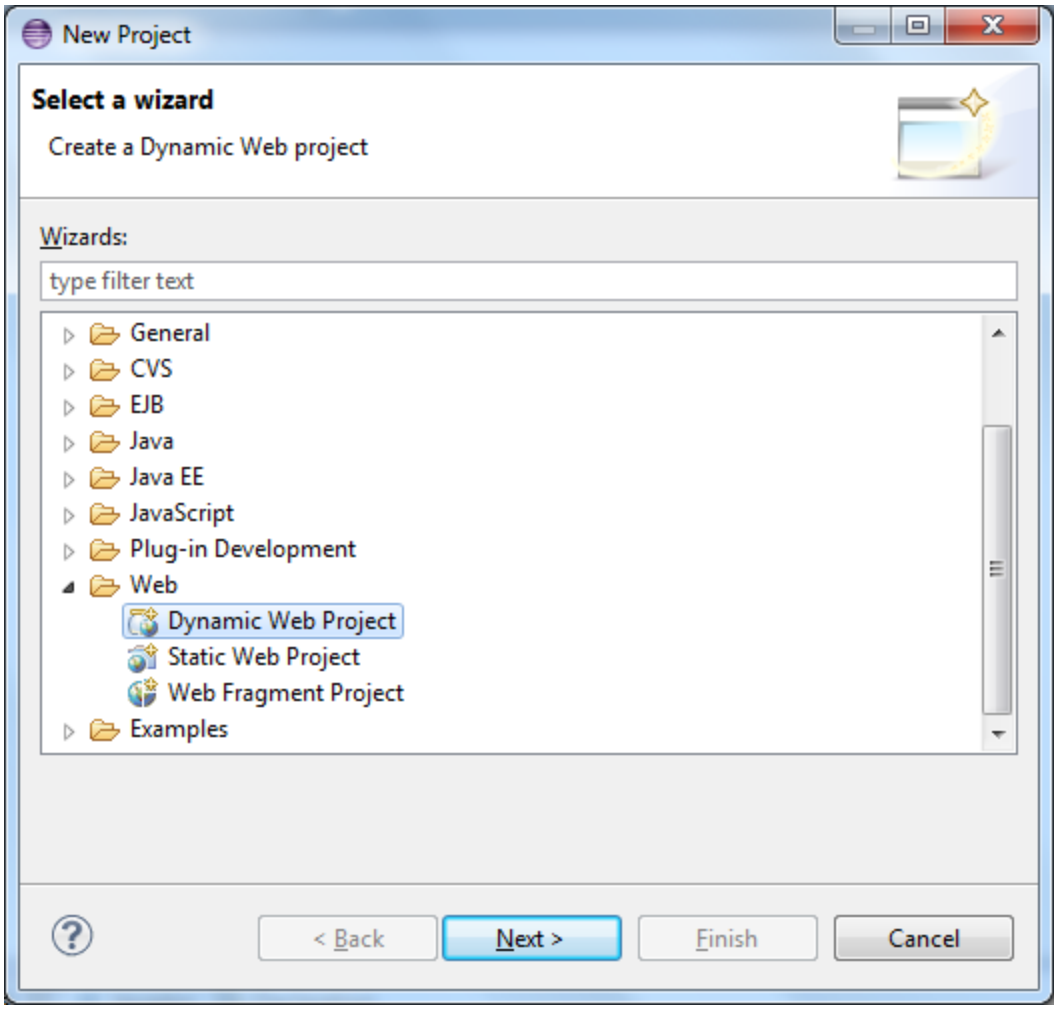
A bottom-up web service starts with the creation of Java code, followed by the generation of a WSDL file describing the operations and messages necessary to interact with the Java code.

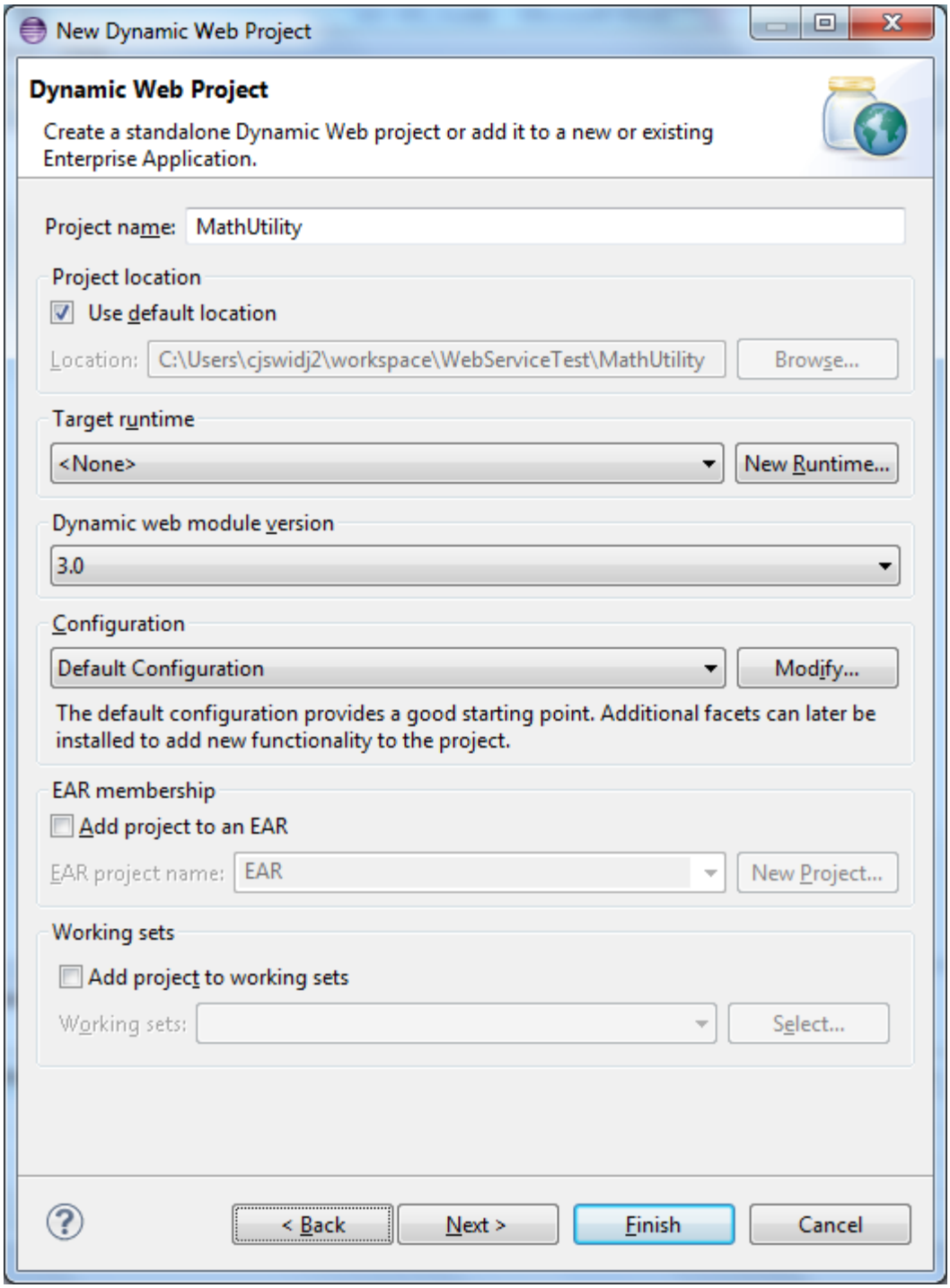
Create an Eclipse Project

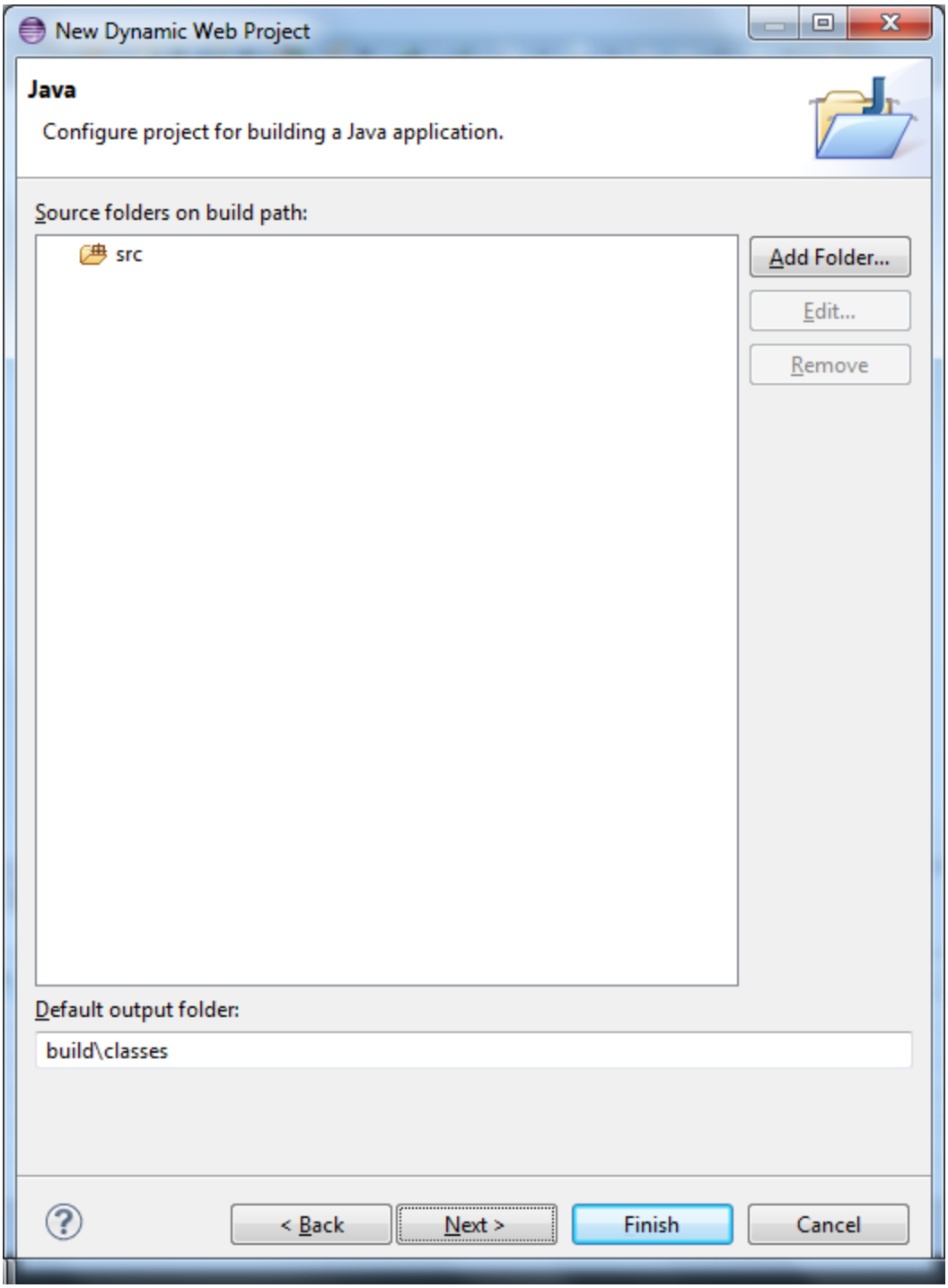
Create a project workspace

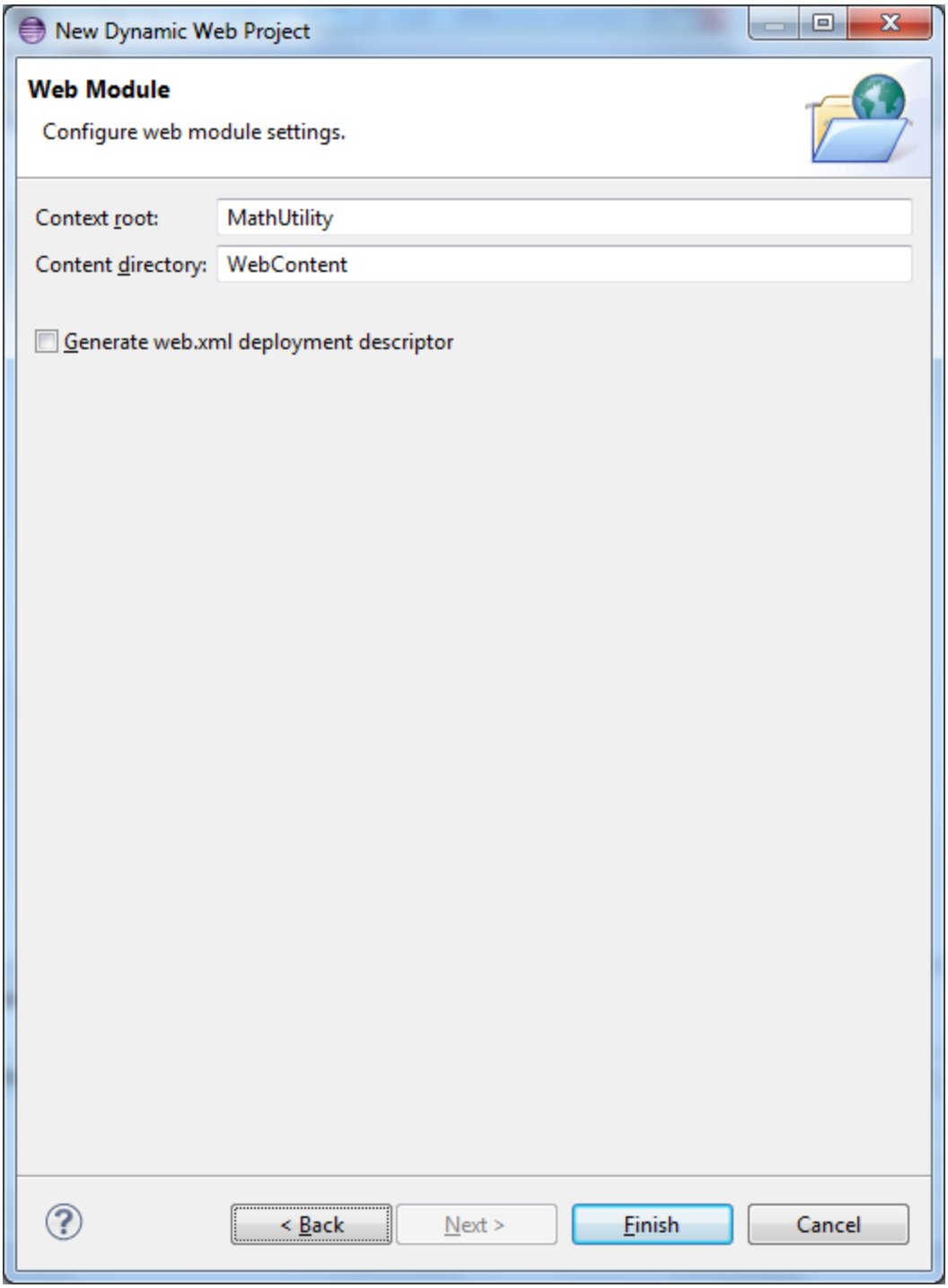


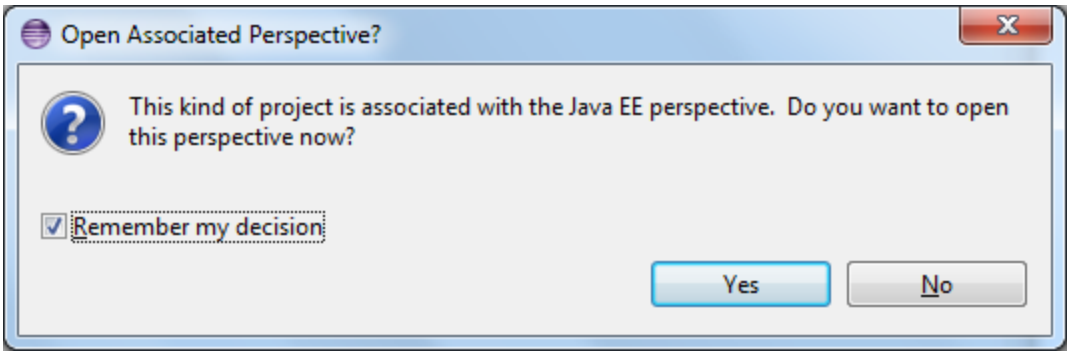
Create a Dynamic Web Project

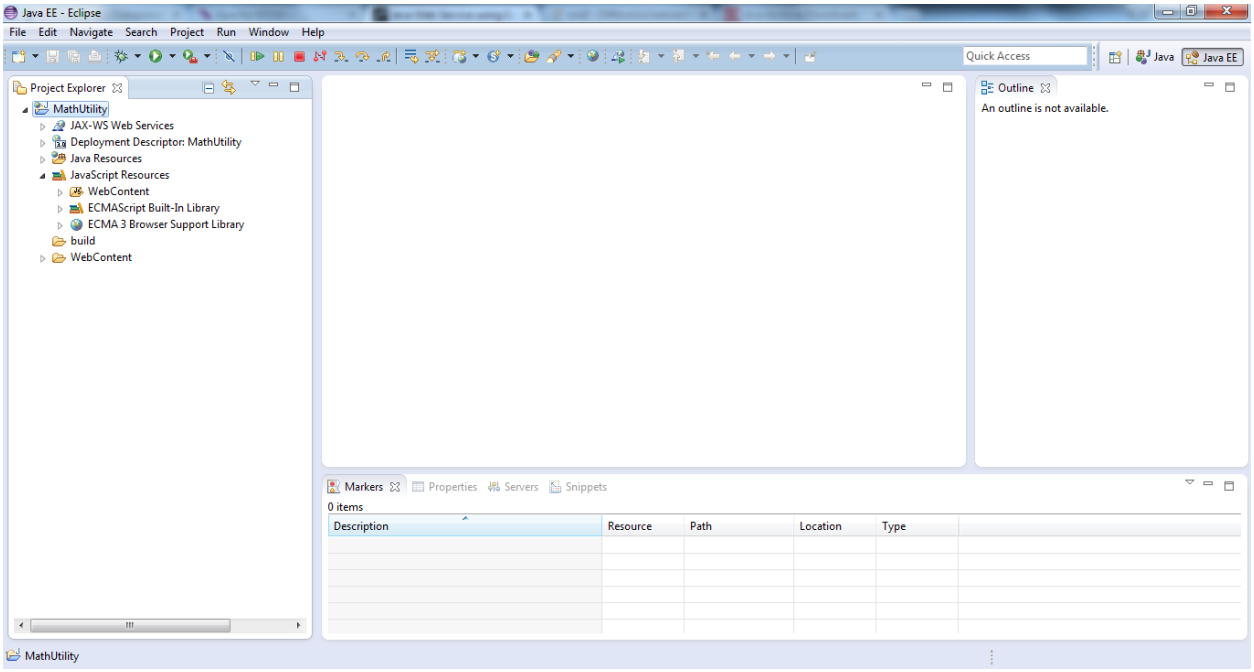

New Project
Select a wizard
Create a Dynamic Web project
Wizards:
type filter text
General
CVS
EJB
Java
Java EE
JavaScript
Plug-in Development
Web
Dynamic Web Project
Static Web Project
Web Fragment Project
Examples
< Back
Next >
Finish
Cancel


New Dynamic Web Project
Dynamic Web Project
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.
Project name: MathUtility
Project location
Use default location
Location: C:\Users\cjswidj2\workspace\WebServiceTest\MathUtility
Browse...
Target runtime
<None>
New Runtime...
Dynamic web module version
3.0
Configuration
Default Configuration
Modify...
The default configuration provides a good starting point. Additional facets can later be installed to add new functionality to the project.
EAR membership
Add project to an EAR
EAR project name: EAR
New Project...
Working sets
Add project to working sets
Working sets:
Select...
< Back
Next >
Finish
Cancel


New Dynamic Web Project
Java
Configure project for building a Java application.
Source folders on build path:
src
Add Folder...
Edit...
Remove
Default output folder:
build\classes
< Back
Next >
Finish
Cancel


New Dynamic Web Project
Web Module
Configure web module settings.
Context root:
MathUtility
Content directory:
WebContent
Generate web.xml deployment descriptor
< Back
Next >
Finish
Cancel

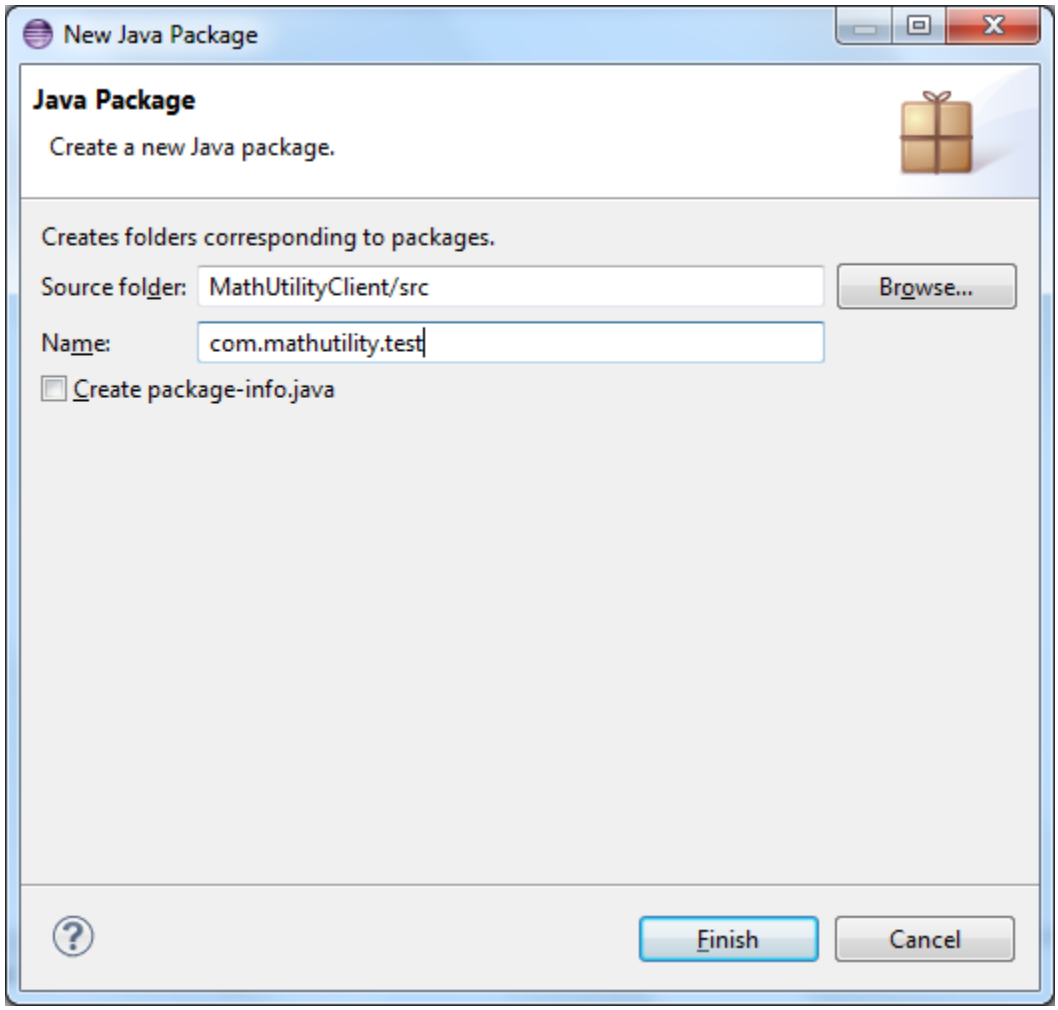




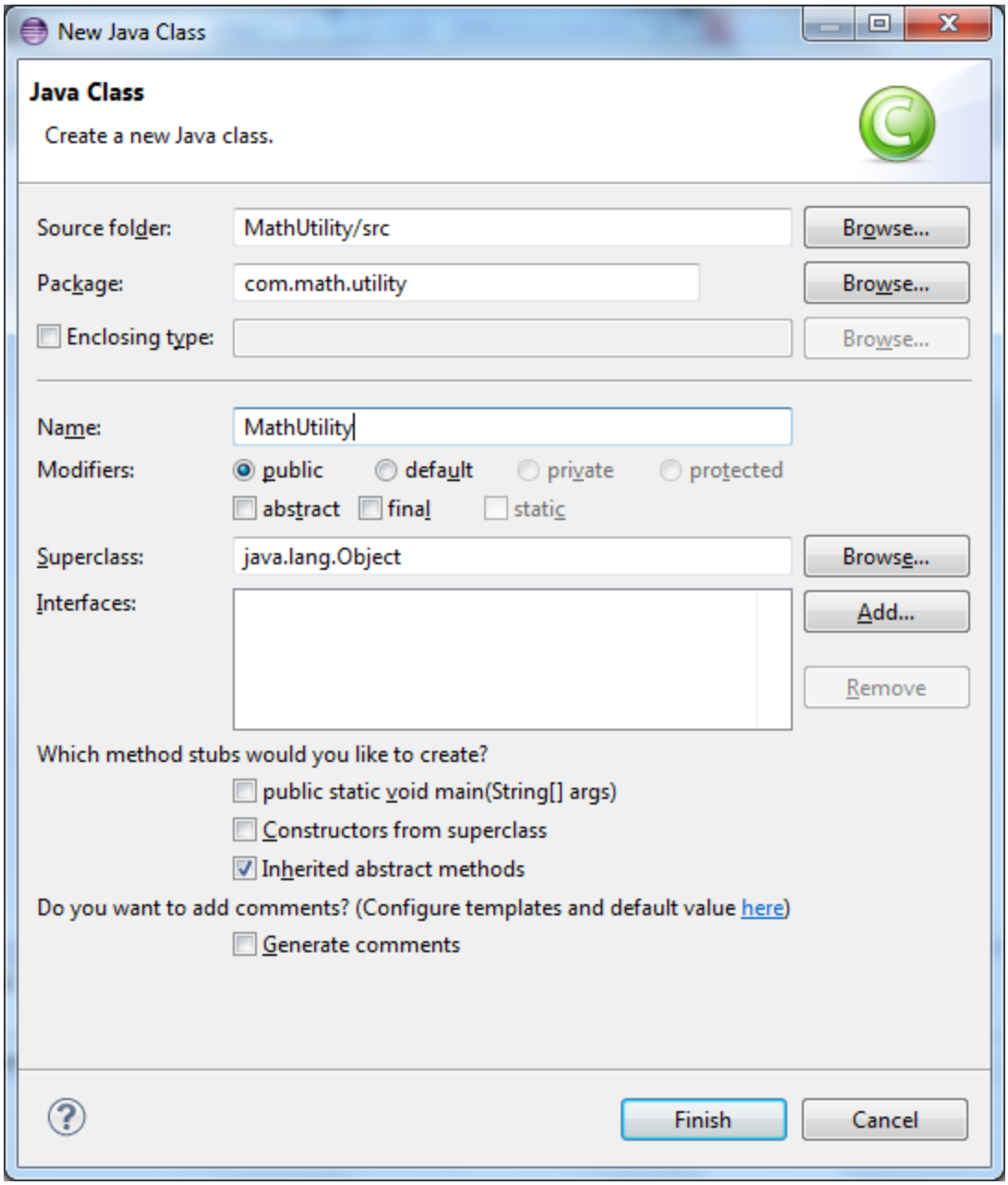
Create the Logic to Expose as a Web Service

For this example, we'll be creating a MathUtility class which exposes two methods for manipulating numbers. There is nothing inherently special or web service-related about this logic; it could easily be used, on its own, outside of the context of SOAP web services.

Create a new package under the Java Resources / src directory



Create a new class under the package. This class contains the logic that will be exposed as web service operations.



Create the logic

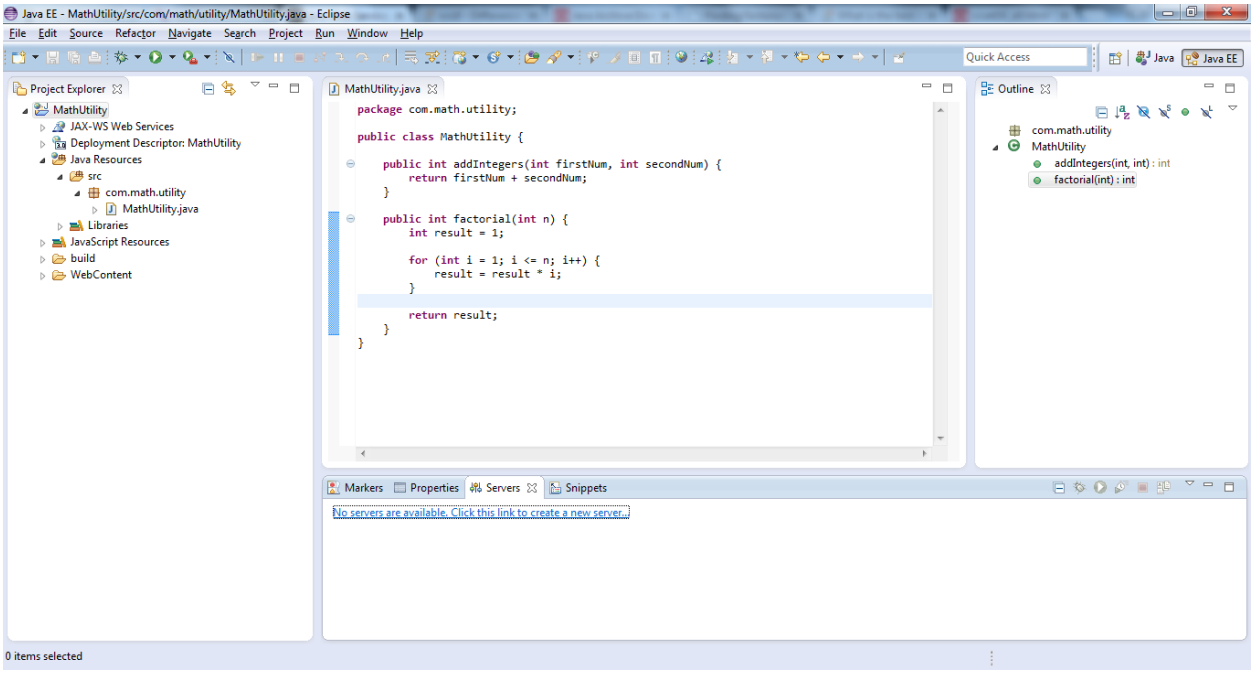
```
package com.math.utility;

public class MathUtility {

    public int addIntegers(int firstNum, int secondNum) {
        return firstNum + secondNum;
    }
```

```
    public int factorial(int n) {
        int result = 1;
        for (int i = 1; i <= n; i++) {
            result = result * i;
        }
        return result;
    }
}
```

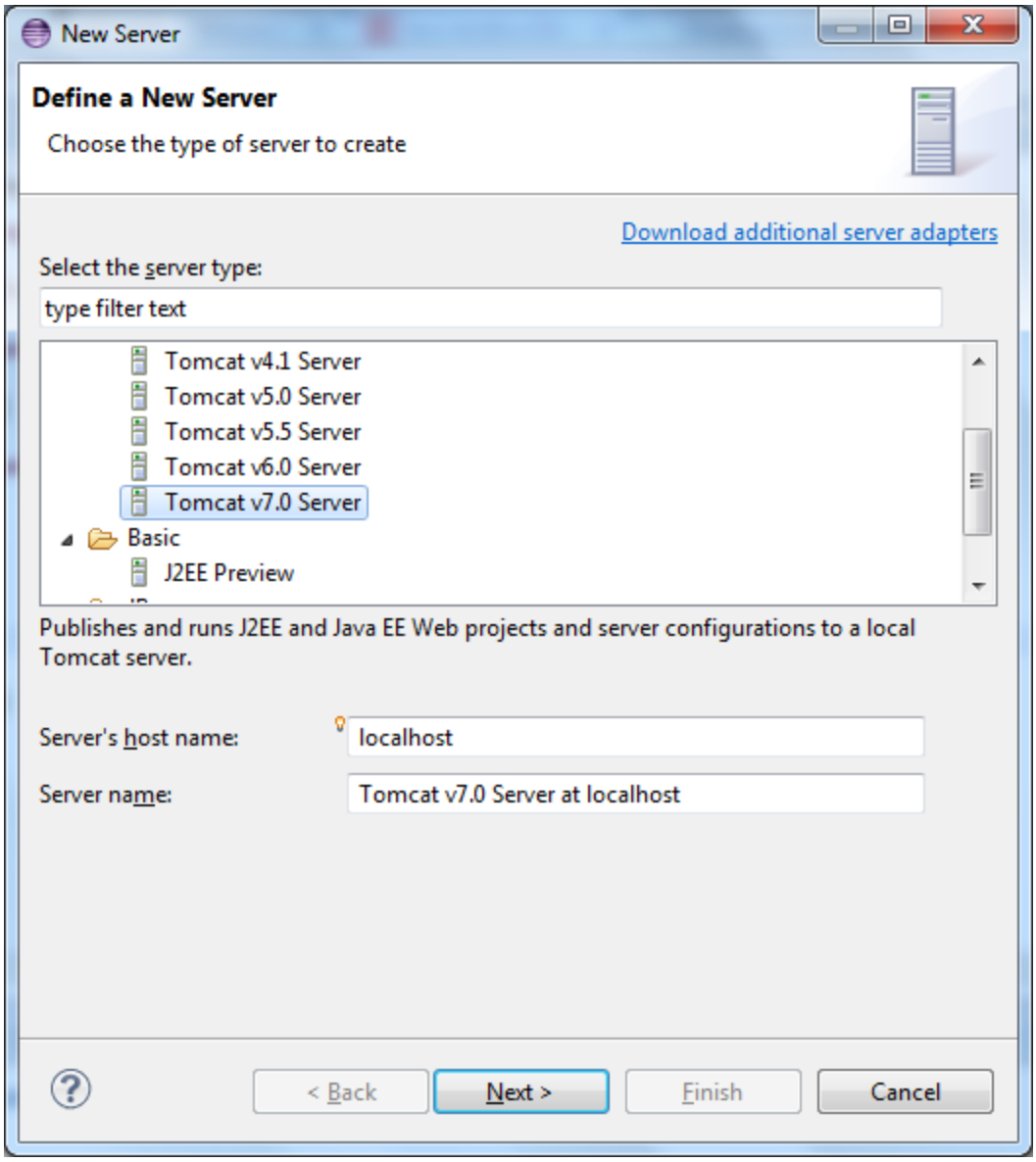
Web Server Configuration

The Apache Tomcat server, which will host the web service, needs to be configured in the Eclipse IDE.

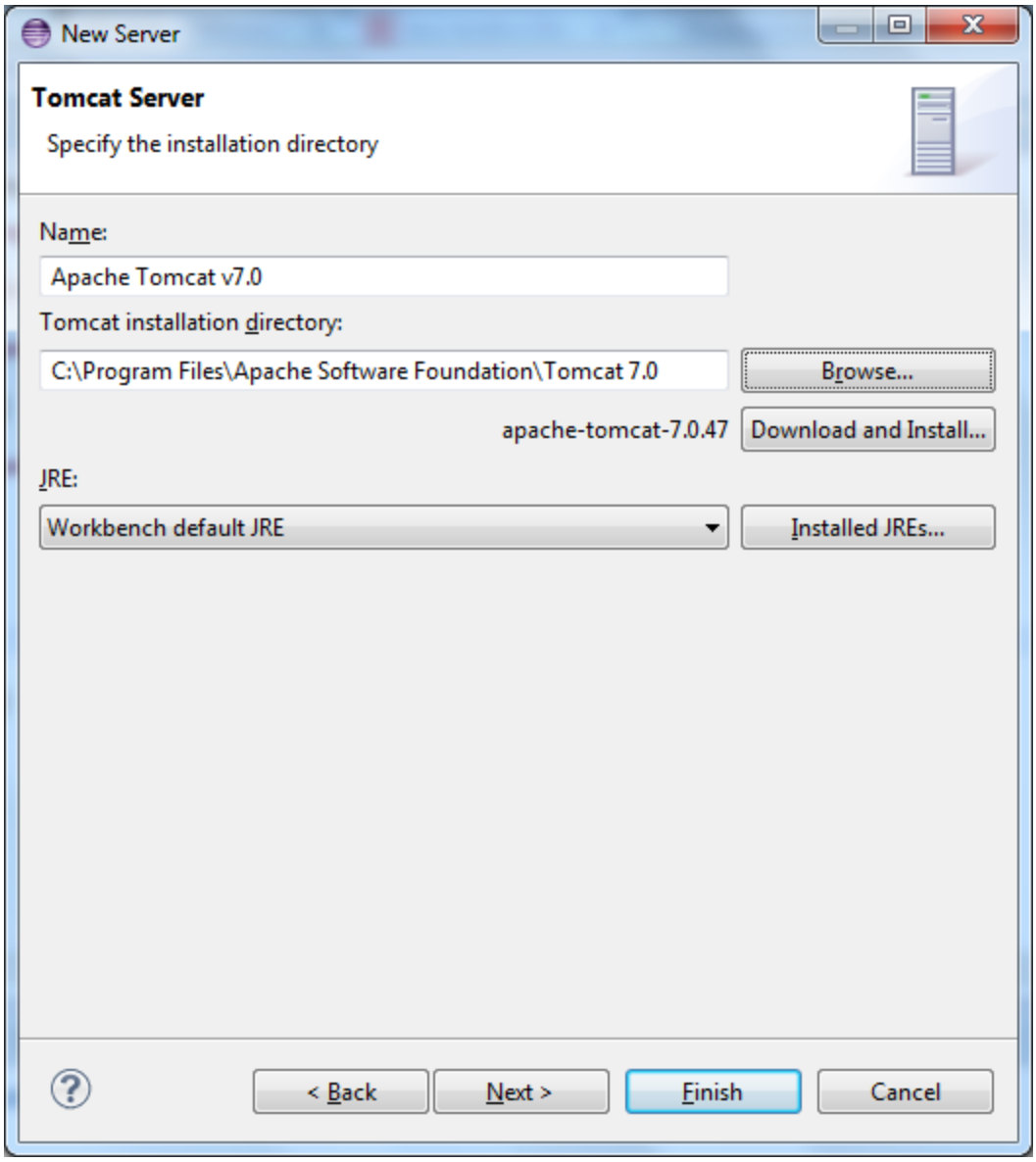
Click on the "Server" tab at the bottom of the UI



Select "Create a New Server" and then Select "Apache" – "Tomcat 7"



Select the Tomcat Installation Directory



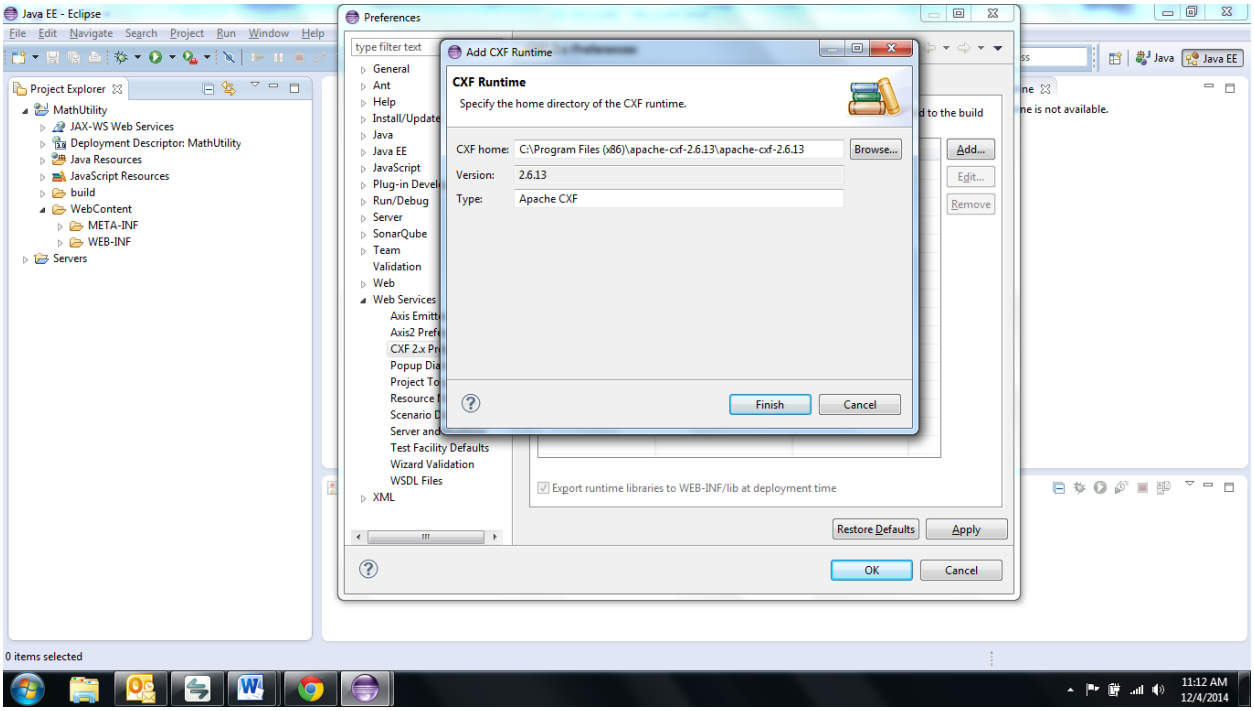
Select "Finish". Tomcat is now added to the Servers listing

Apache CXF 2.x Configuration

The Eclipse IDE needs to be configured to use the Apache CXF web service framework for creating and communicating with web services.

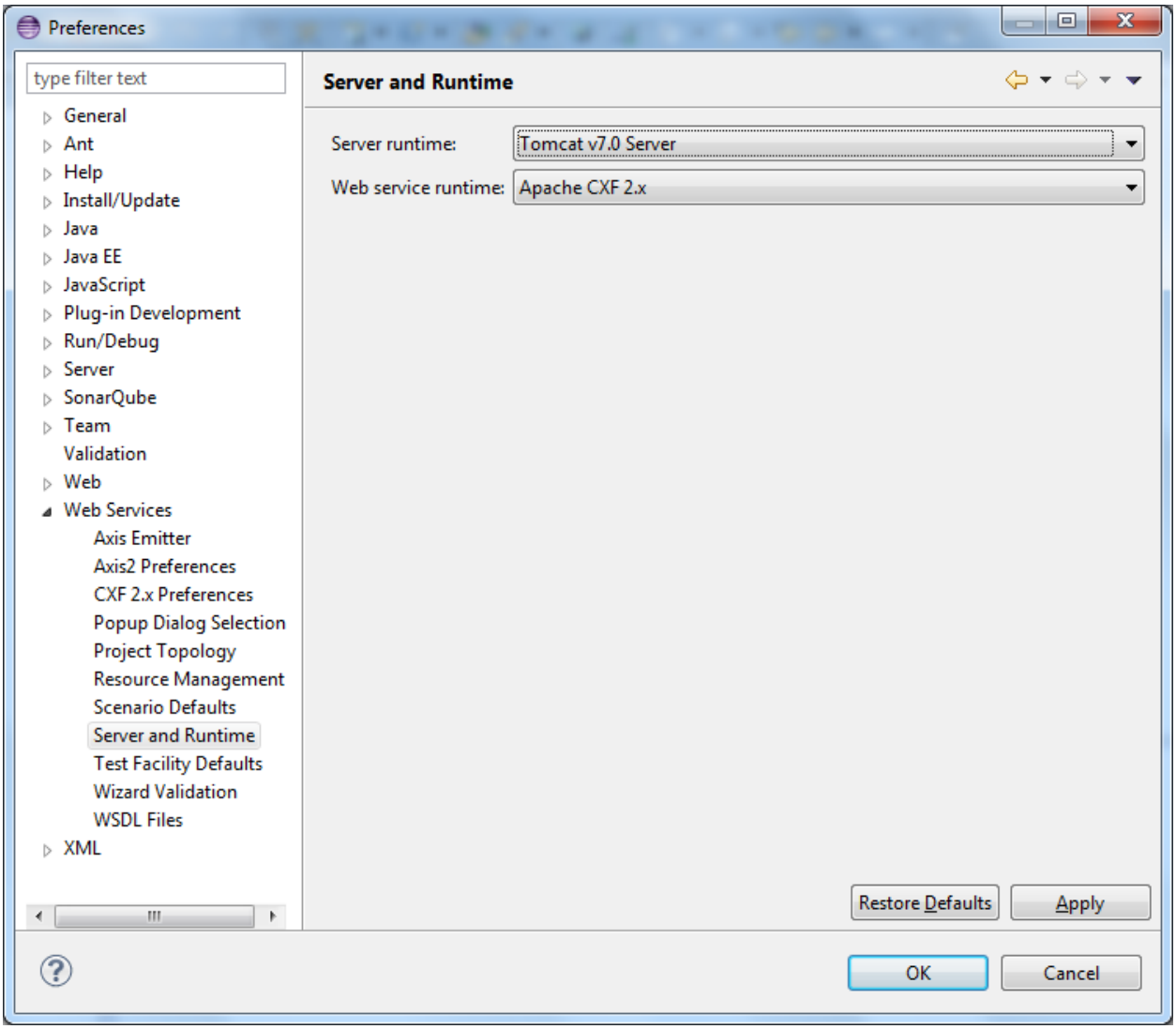
Select "Window" – "Preferences" – "Web Services" – "CXF 2.x Preferences".

Under "CXF Runtime" select "Add" and find the location in which you unpacked the CXF libraries



Select "Apply" and "Ok" to complete this step.

Finally, select "Server and Runtime" underneath "Web Services" and set the Server Runtime to Tomcat 7. The Web Service Runtime should be set to Apache CXF 2.x.

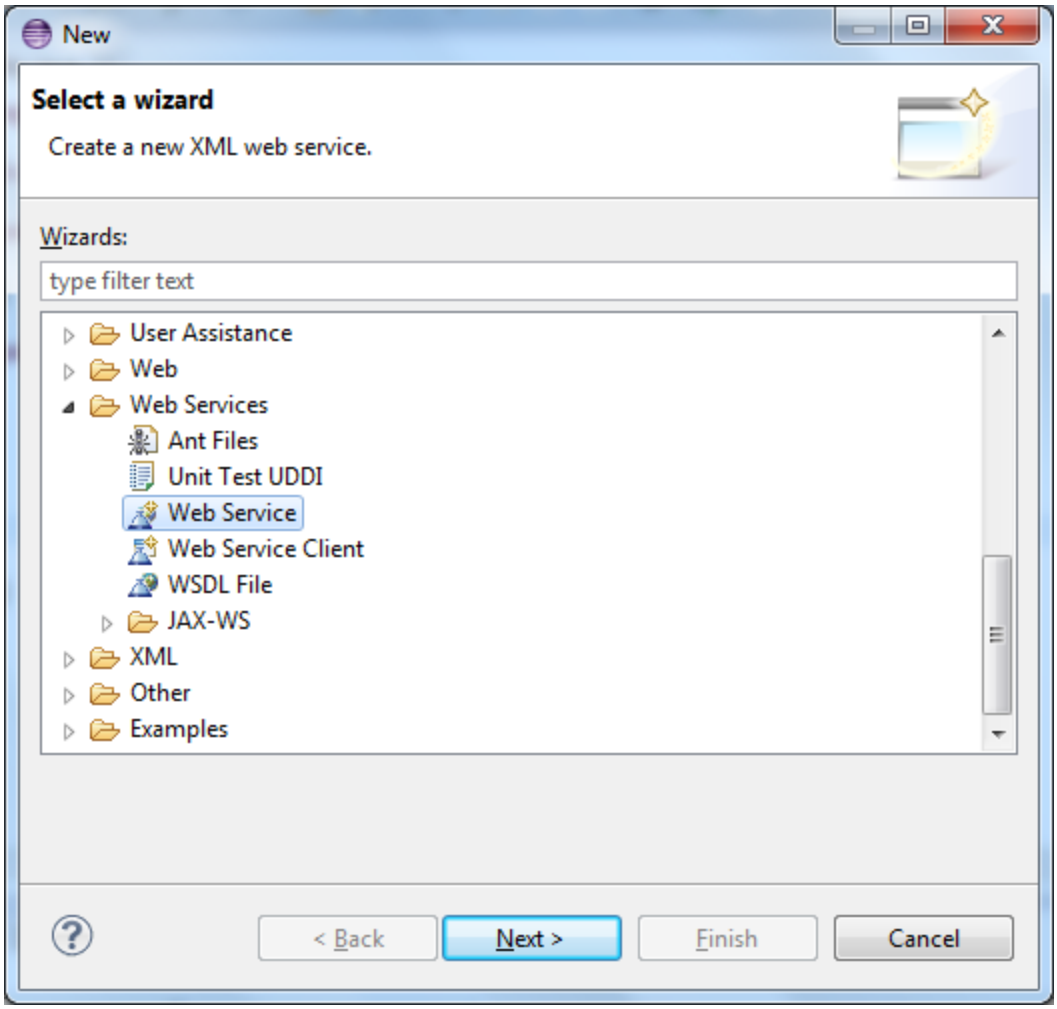


Select "Apply", then "OK" to finish.

Create the Web Service

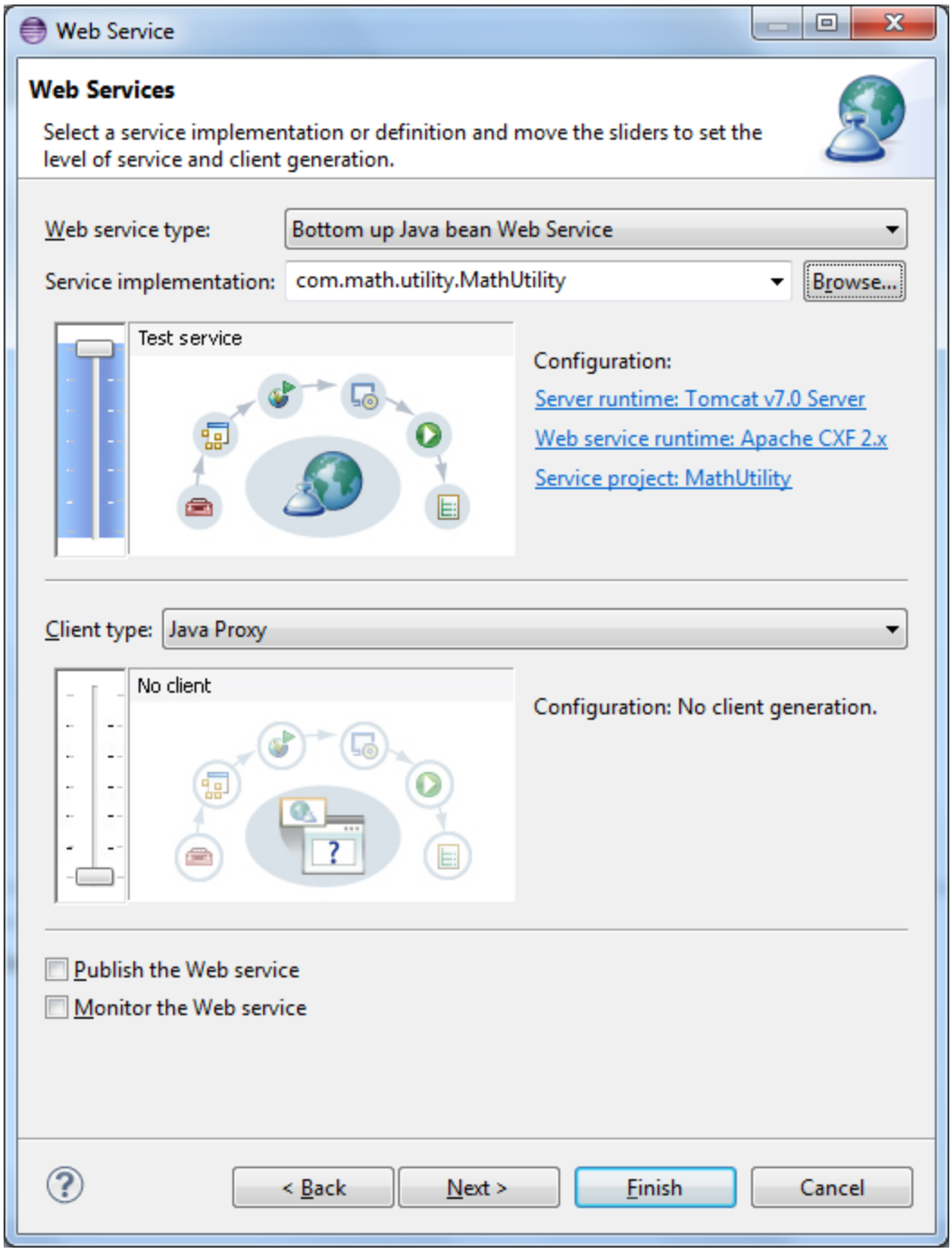
After creating the logic that we'd like to expose as a service, along with configuring the Eclipse workspace to use Tomcat and CXF, we are able to build the Web Service and run it.

Right Click Java Resources – New – Web Service



"Browse" for the Service Implementation to use for the Web Service. In our case, this will be "MathUtility".

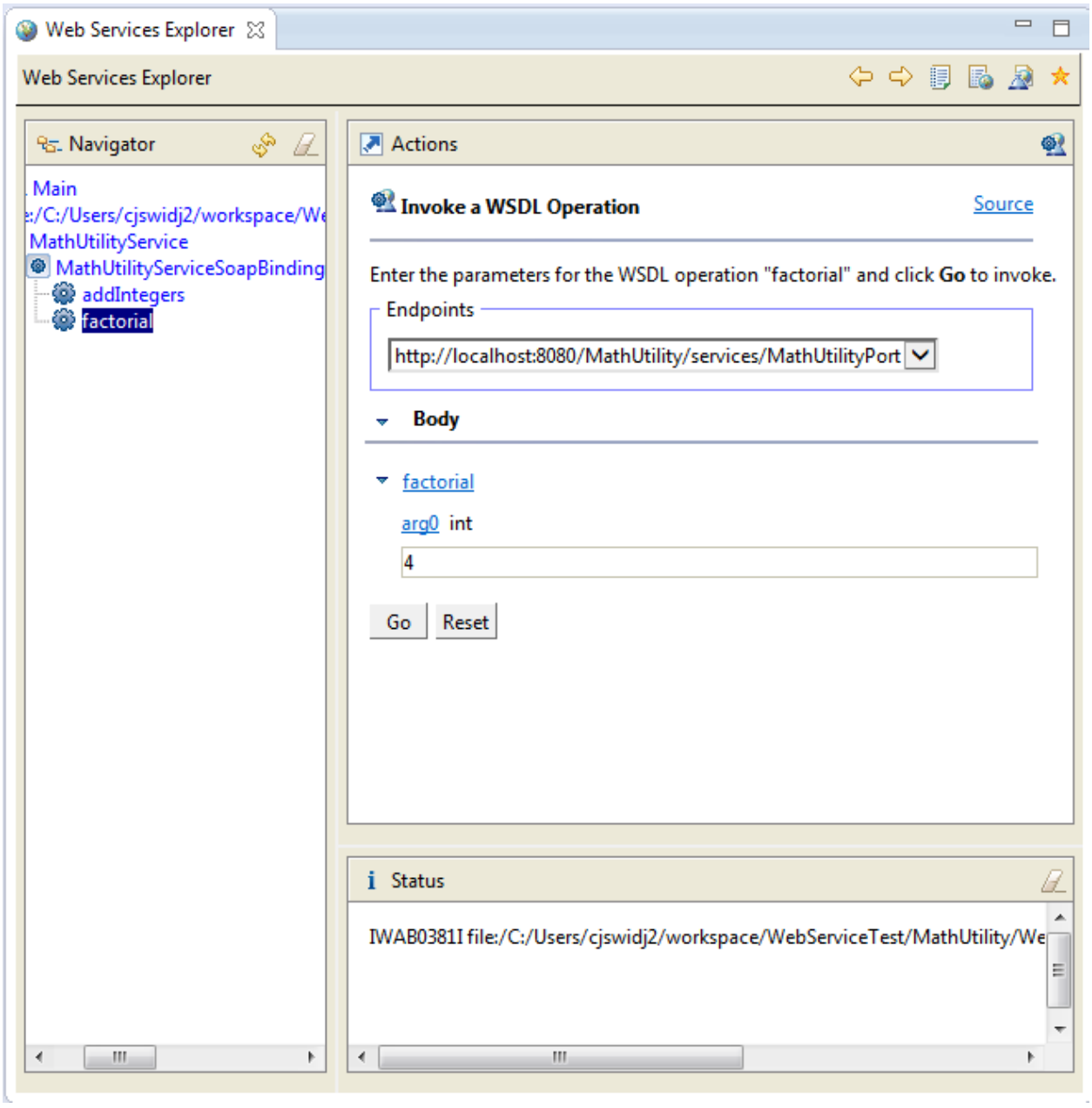
Move the Slider Bar to "Test Web Service" and select "Finish".



At this time, the Eclipse IDE uses Apache CXF to generate a WSDL file and associated schema to create a web service based on the MathUtility logic. After creating the necessary files, the tool adds these files to the "wsdl" directory within "WebContent".

After the web service creation is completed, the web service is started, along with a utility allowing us to verify that the logic is exposed as a web service and works correctly.

Once the web server has started, a Web Services Explorer window will appear with service details.



If you select an Operation, you can test that Service. In this sample, select "factorial". Then enter a number in the textbox.

Select "Go" and the operation is executed. The "Status" window shows the Response of the web service call.

The logic within "MathUtility" has now been exposed as a SOAP web service, deployed to an Apache Tomcat web server, and tested using the Web Services Explorer feature built-in to the Eclipse IDE.