1. Creating a Web Service Project

Begin by launching the Web Service Project wizard. This wizard consists of three pages. Page-1 collects Web Project configuration details. Page-2 collects the XFire configuration details. And page-3 configures the XFire libraries on the new project's build path.

1. Launching the Web Services Project Wizard.
2.
    1. Create a simple Web Service Project by selecting Web Service Project from the new toolbar menu:

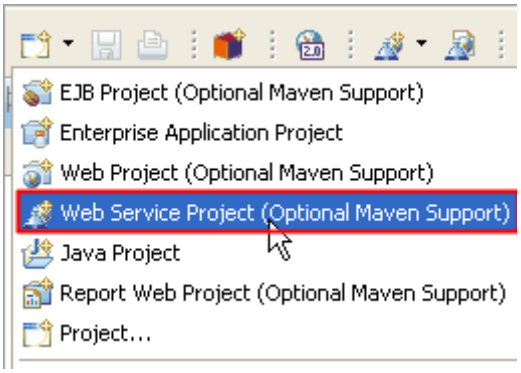

Figure-1: New Project Wizard Launcher

1. Complete the Web Project details on page-1 of the wizard and select **Next** to proceed

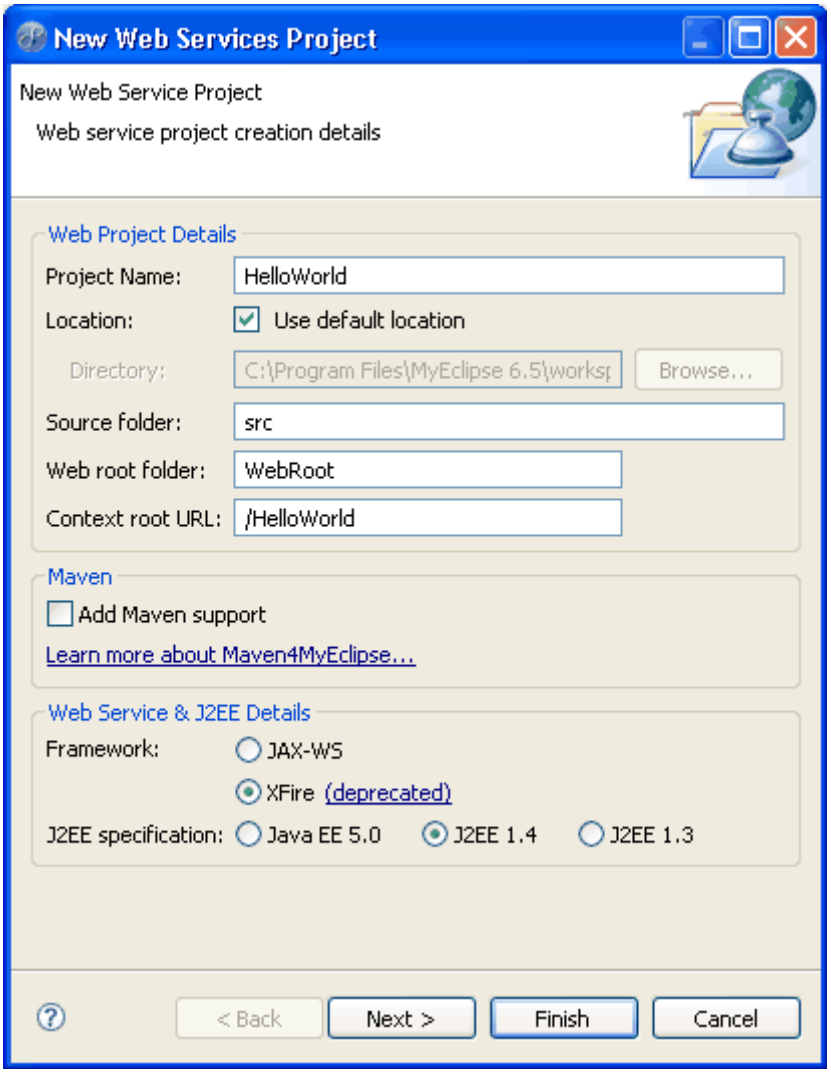

Figure-2: Page-2, Collecting web configuration details

1. On page-2 of the wizard enter the XFire servlet and service.xml configuration details. The default values are provided to allow the page to be completed without additional user input.

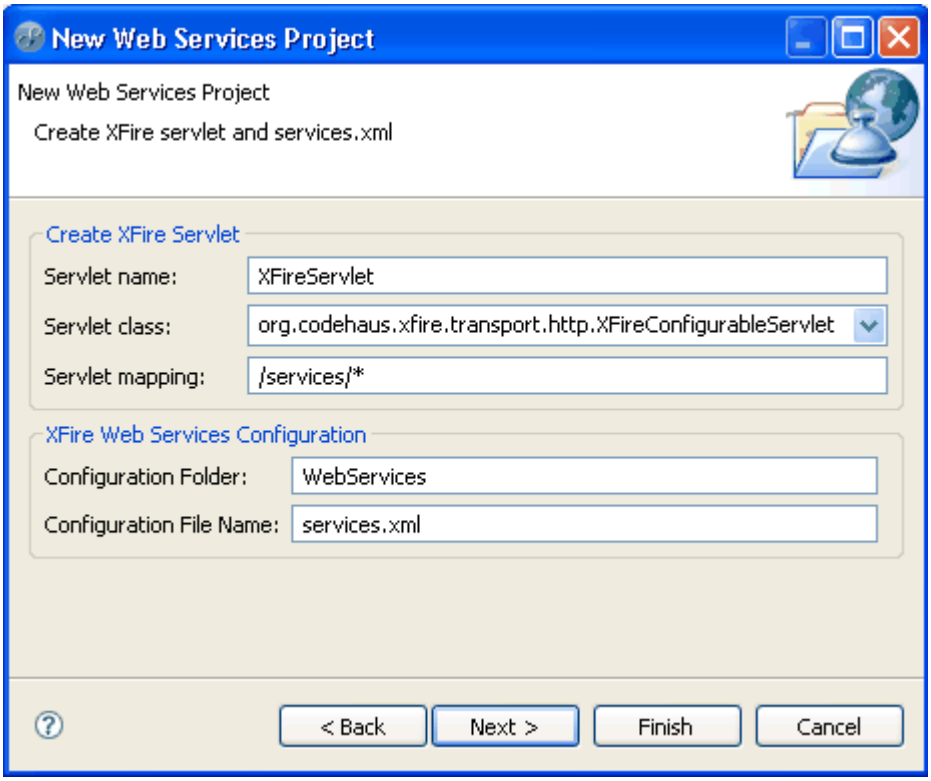

Figure-3: XFire servlet and services.xml configuration

1. On page-3 of the wizard select the libraries to add to the project's build path and deployment profile. The XFire Core Library is required and should always be checked unless you are manually managing the XFire libraries within the project. If you plan to develop a client application in the project include the XFire HTTP Client Libraries as well.

Upon completion of the wizard the Libraries will be added to project's build path. No Jar files are copied to the project folder.

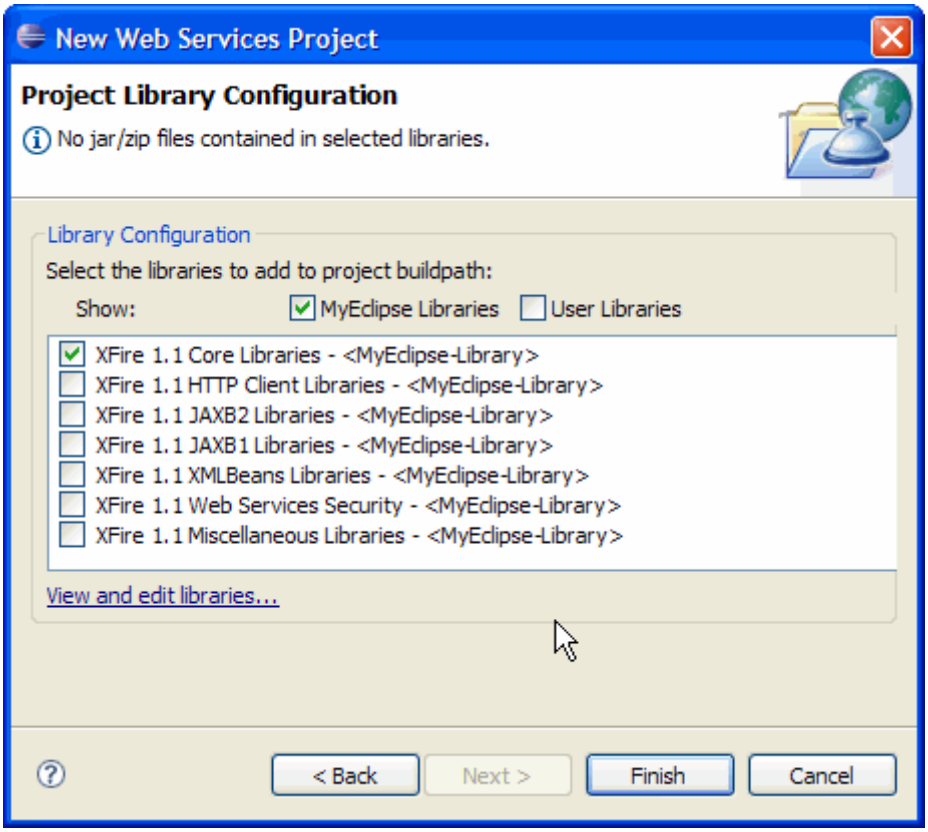

Figure-4: Selecting XFire libraries to add to new web service project buildpath

1. Select **Finish** to complete the wizard and initiate the web service creation process.

Figure-5, illustrates the structure of the newly created HelloWorld web service project. Note the similarity of a web service project to a standard MyEclipse web project. The additional XFire web service configuration elements are shown in red.

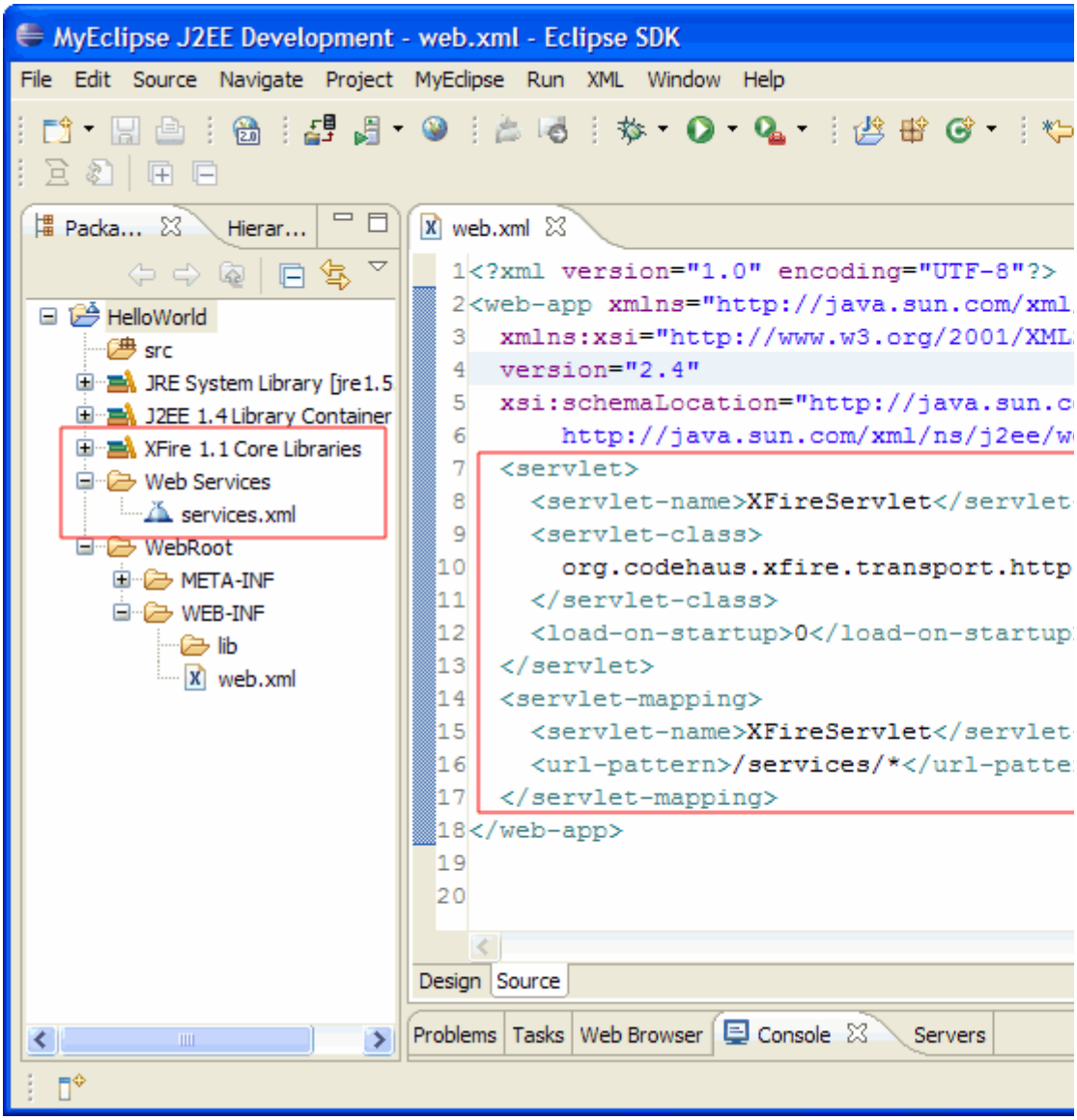

Figure-5: Web services artifacts of a new web service project

2. Creating a Web Service - Code-first Strategy

1. Launch the Web Service Wizard. There are two methods for launching the MyEclipse Web Service Wizard:

Method-1: Launch wizard from MyEclipse perspective toolbar
Choose the New Web Service button on the workbench to open the Web Service Wizard.

Method-2: Launch wizard from workbench menubar

1. From the workbench menubar select **File>New>Other>MyEclipse>Web Service**.

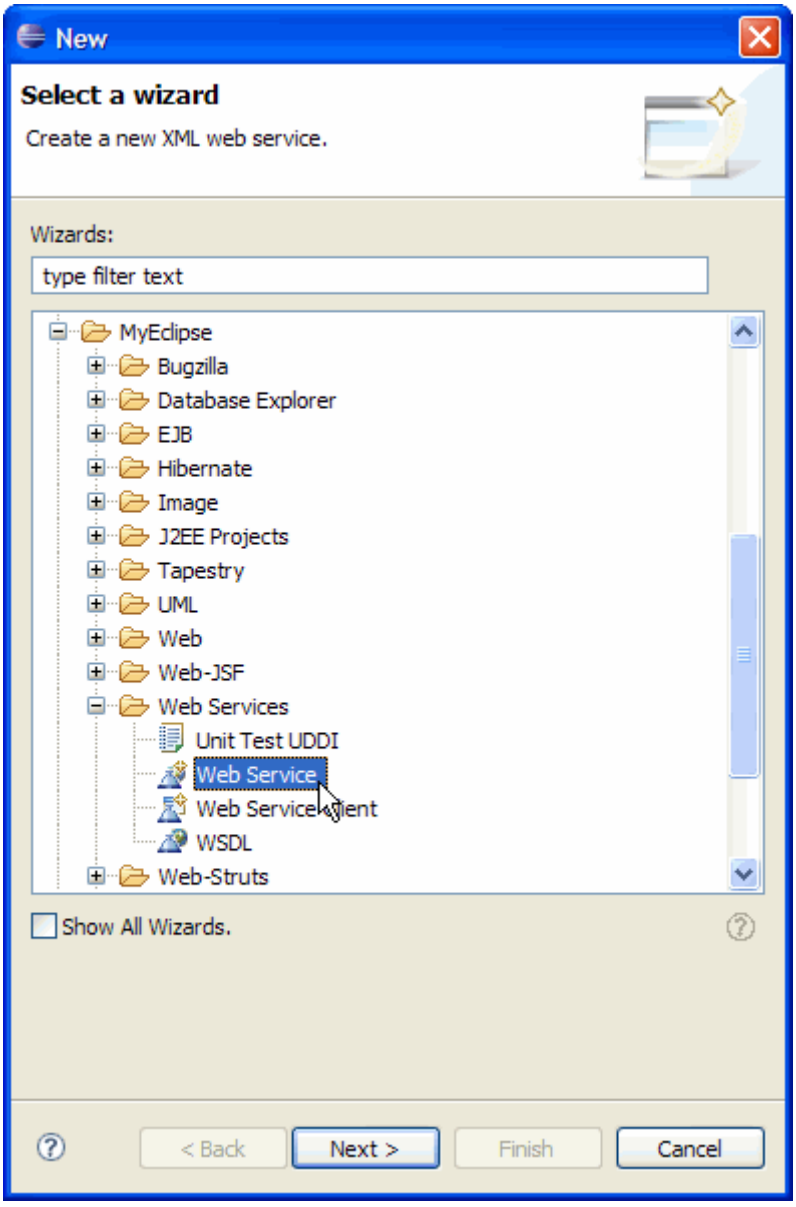

Figure-6: Launching Web Service Wizard

With the Web Service Wizard launched proceed as follows:

1. On page-1 of the wizard, select the *HelloWorld* web service project and

choose the *Create web service from Java bean* creation scenario as shown in Figure X.

2. Select **Next** to advance to page 2.

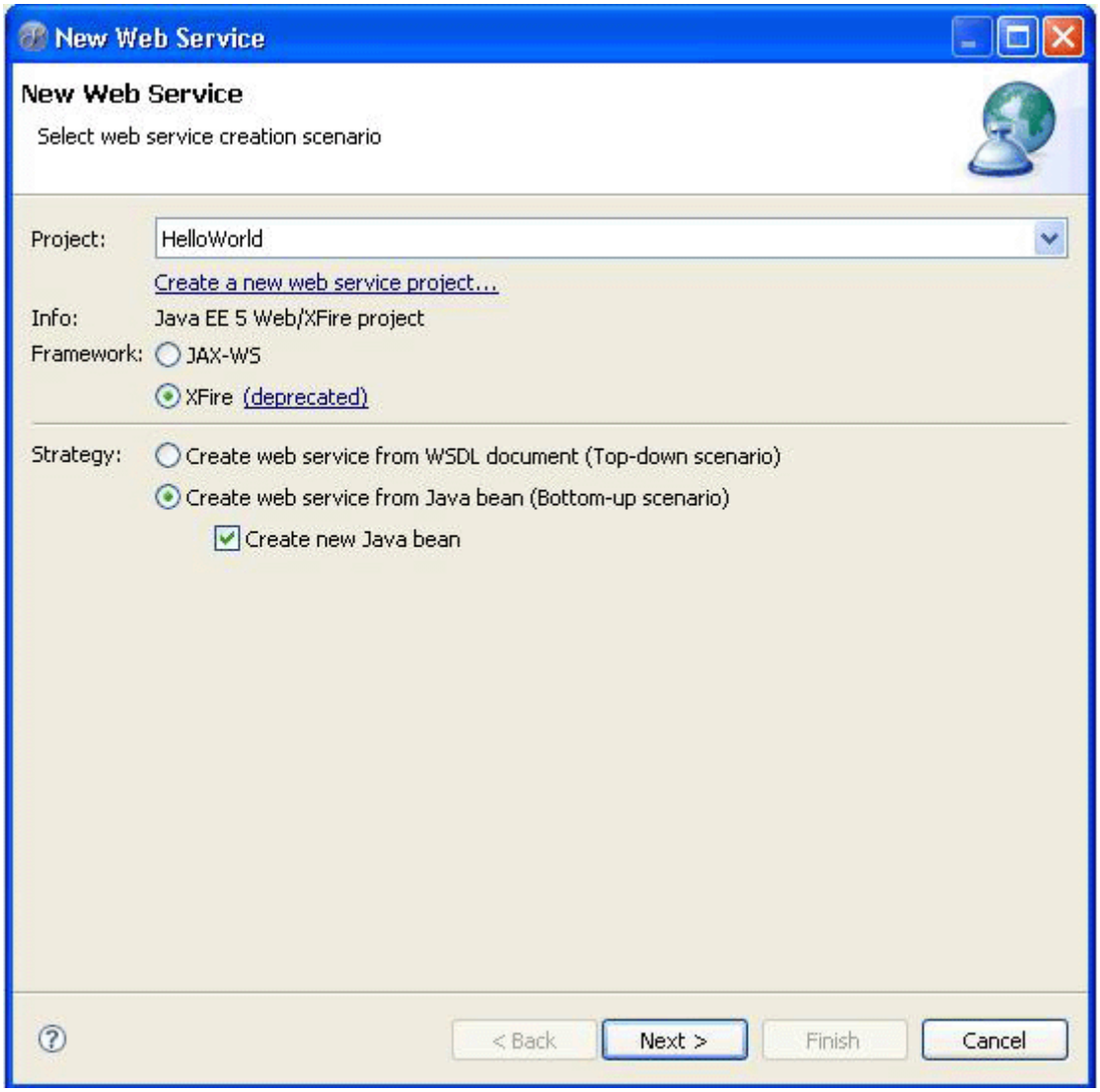

Figure-7: Page-1 of new web service wizard

1. Enter *HelloWorldService* for the web service name.
2. Select the Java source folder or Select the New button to create a new source folder
3. In the Java package field, enter an existing Java package name (code-assist is available) or choose the **Browse** button to select from a list of existing packages. Choose the **New** button to create a new Java package.

Note that default values for the Service interface and the Service implementation class are generated based on the name entered in the web service name field. Figure X illustrates the default details

1. Select **Next** to initiate the web service creation process.

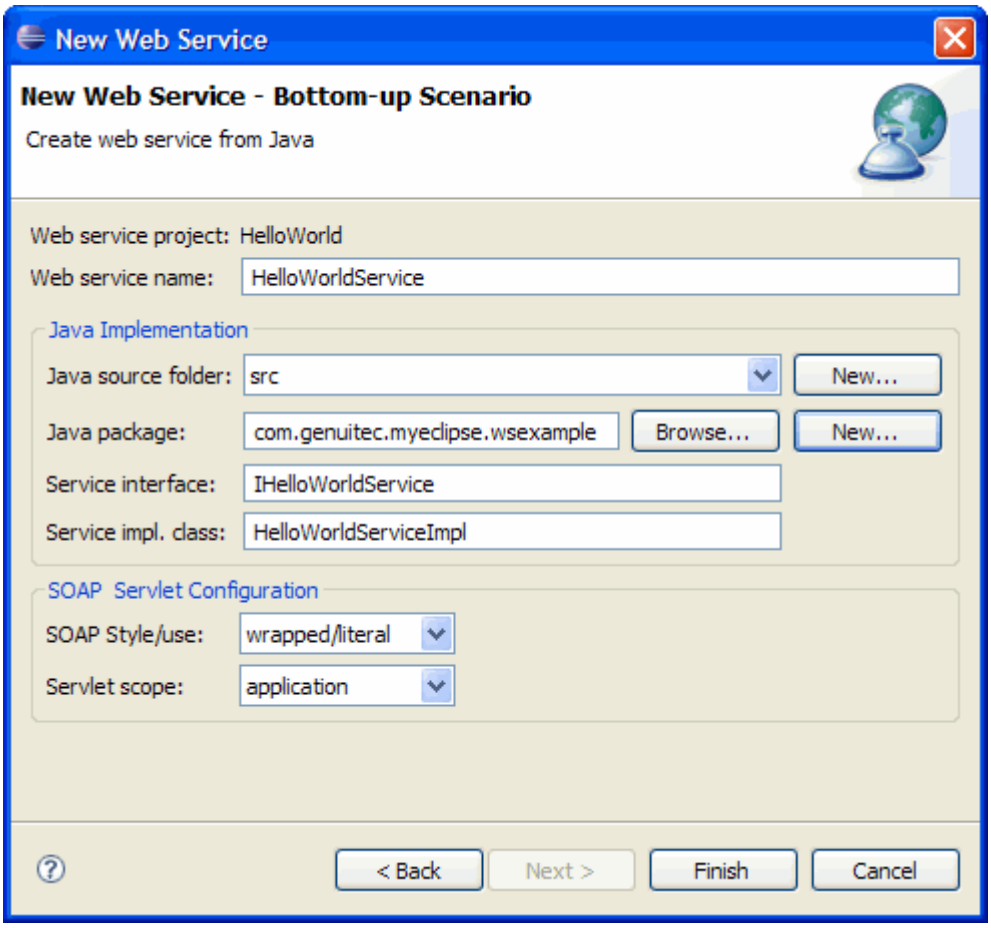

Figure-8: Page-2 of new web service wizard.

The wizard generates the IHelloWorldService Java interface and the HelloWorldServiceImpl Java class, and creates a `<service>` entry in the project's services.xml configuration file, (see Figure-9 below). Note that the *example(String message)* method was generated in the interface and class to serve as a simple testing operation for when the web service is deployed. This method is not essential to the operation of the web service and should be removed.

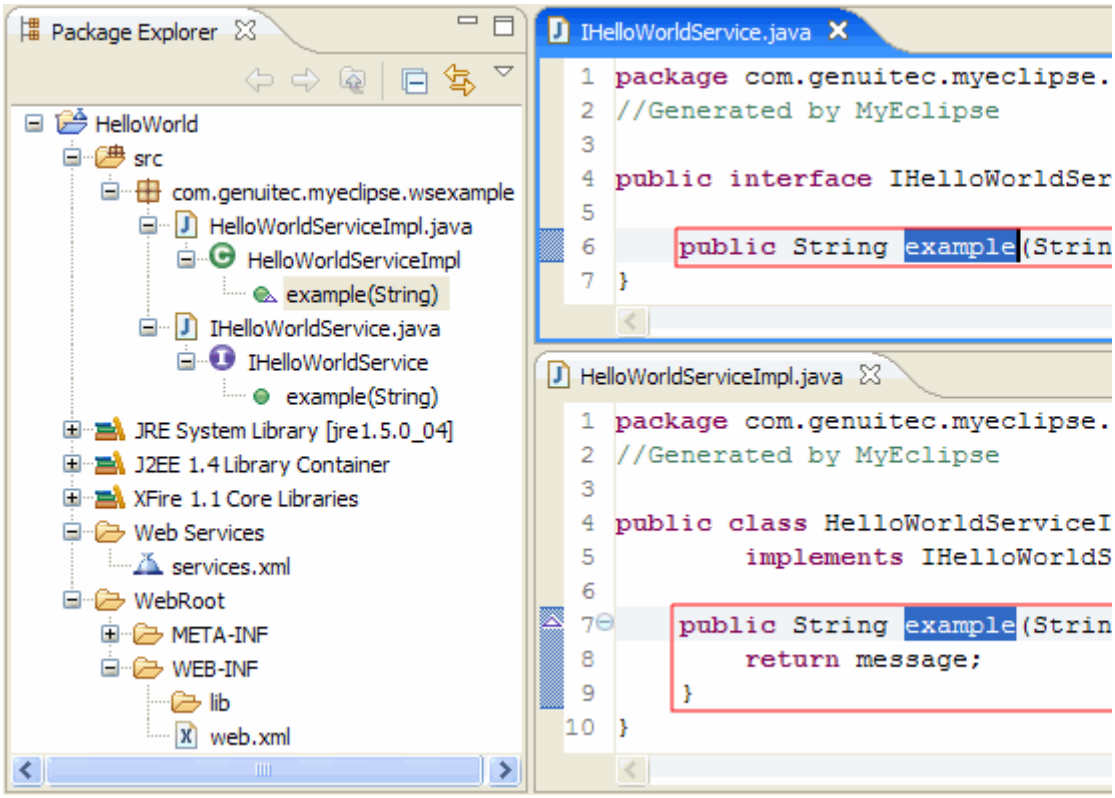

Figure-9: Newly created HelloWorld interface and implementation class

top

3. Deploying a Web Service Project

Figure-10 below provides a visual outline of the 3 steps required to deploy the HelloWorld application using the MyEclipse J2EE Application Deployer.

1. From the Server Manager select the Deployer button (step-1)
2. In the Server Deployments dialog select Add to create a new deployment (step-2)
3. In the New Deployment dialog select the HelloWorld project and the Exploded Archive option.
4. Select Finish in the New Deployment dialog to create and stage the HelloWorld web project as an exploded WAR to the Tomcat 5 server's automatic deployment location
5. Select **OK** in the Server Deployments dialog

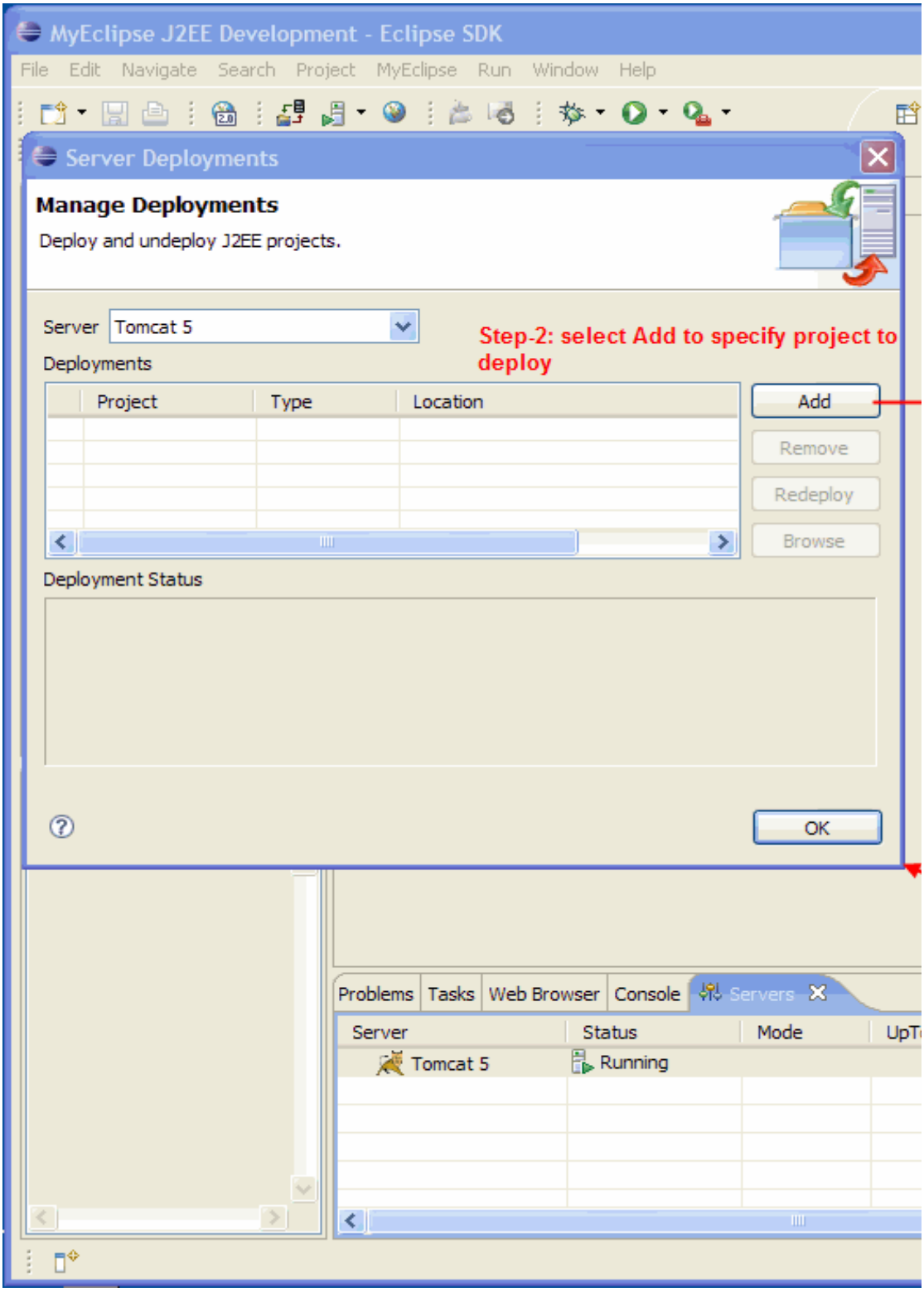

Figure-10: Three-step deployment process

The new HelloWorld WAR deployment will appear under the Tomcat 5 node in the Server Manager view (see Figure 11).

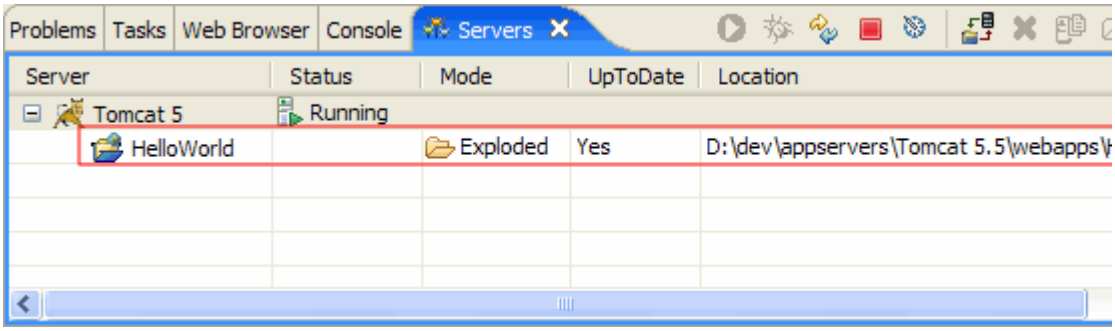

Figure-11: Servers Manager View depicting deployed HelloWorld web service project

4. Starting the Tomcat Server

With the HelloWorld project deployed to Tomcat, we need to launch Tomcat if it is not already running. If the server is running the HelloWorld webapp will be automatically loaded and made accessible. To launch Tomcat choose either the debug launch-mode button or the run launch-mode button (see highlighted buttons in Figure-12).

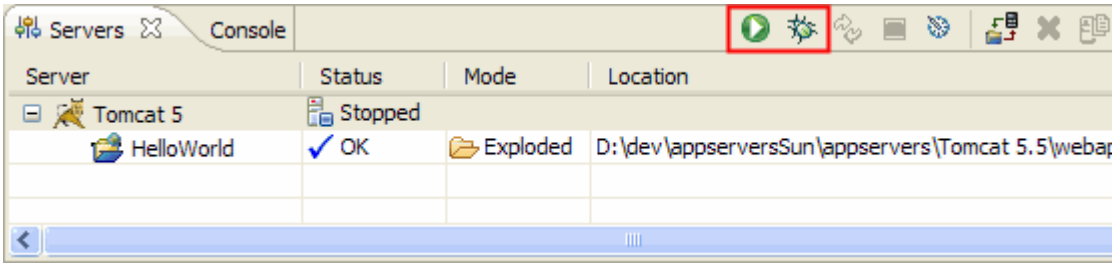

Figure-12: Server Manager before Tomcat 5 launch

A Tomcat starts up can monitor its start up progress in the Eclipse console view.

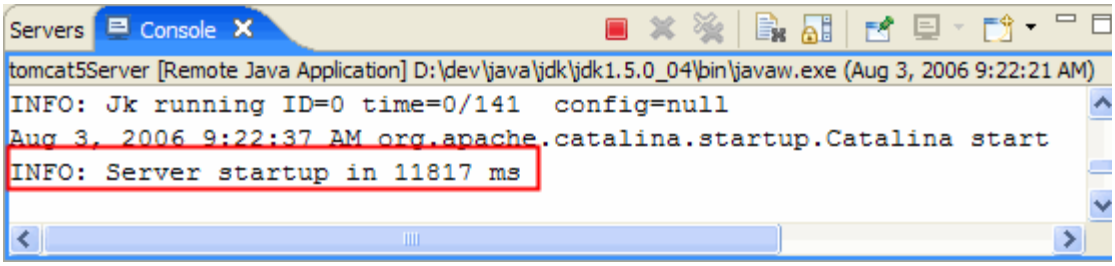

Figure-13: Eclipse console view depicting real-time Tomcat 5 status info

Once Tomcat is launched and fully initialized the Server Manager will update to reflect the server's execution status and mode.

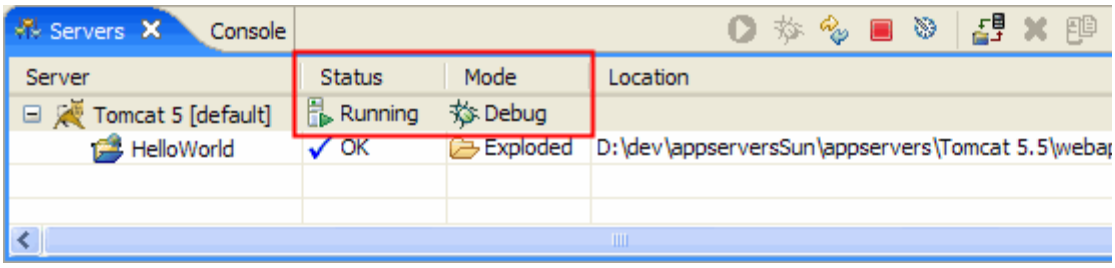

Figure-14: Server Manager showing active status of Tomcat 5

4.Test Xfire In Client（IOS）

Take the codes for example straight.

```
NSString *soapMessage =
[NSString stringWithFormat: @"<?xml version=\"1.0\" encoding=\"utf-8\"?>\n"
        "<soap:Envelope       xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
        xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"
        xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\"
        soap:encodingStyle=\"http://www.w3.org/2001/12/soap-encoding\">\n"
        "<soap:Body>\n"
        "<exmaple xmlns=\"http://wsexample.genuite.myclipse.wsexample.com\">\n"
          //"< message >Mcgratty"
          //"</ message >"
          "</ exmaple >\n"
          "</soap:Body>\n"
            "</soap:Envelope>\n"];

    //Target URL.
NSURL *url =
[NSURL
URLWithString:@"http://192.168.50.22:8080/HelloWorld/services/HelloWorldService"];

NSMutableURLRequest *theRequest =
```

```
[NSMutableURLRequest requestWithURL:url];
NSString *msgLength = [NSString stringWithFormat:@"%d", [soapMessage length]];

    //Set the appropriate request header.
    [theRequest
    addValue: @"text/xml; charset=utf-8" forHTTPHeaderField:@"Content-Type"];
    [theRequest
    addValue:@http://wsexample.genuite.myclipse.wsexample.com/example
    forHTTPHeaderField:@"SOAPAction"];
    [theRequest addValue: msgLength forHTTPHeaderField:@"Content-Length"];
    [theRequest setHTTPMethod:@"POST"];
    [theRequest
    setHTTPBody: [soapMessage dataUsingEncoding:NSUTF8StringEncoding]];

    //Send the connection and register the callback function to receive the response data from
the webService.
NSURLConnection *theConnection
= [[NSURLConnection alloc] initWithRequest:theRequest delegate:self];

    //process the data sent from WebService.
    -(void)parser:(NSXMLParser *)parser foundCharacters:(NSString *)string
    {
        NSLog(@"5---recordResults:%@", string);
    }
```

Test Result:

If this method print the string of “Mcgratty”, it means that the method in IOS is successful to call the WebService.