Experiment-5

1.1 Objective:

Developing Servlets based applications.

Task-1: Sever Configuration - Configuration of Tomcat Server using Eclipse for EE Developer.

Task-2: **Understanding the role of XML file** - Write a HTML file index.html to show the welcome message. Understand the importance of server and XML file. Check and verify the working of tomcat server using the appropriate port number display the Catalina page to show the connectivity.

Task-3: Demonstration of Servlet Life Cycle - Write a servlet program to demonstrate the working of Servlet life cycle. Understand and observe the working of XML file.

Task-4: Demonstration of Cookies - Create persistent and non-persistent cookies on client side using a servlet. Create a servlet file which will display the first name and last name when the request will come. Create an html file Hello.html. In the html file write the appropriate code to create caption First name and Last name. Design text boxes to accept the first name and last name create a button submit. When a user will hit submit button then XML file should render the detail from the servlet file HelloForm.java created in Task-2.

Task-5: Preparing Write Up - Finally, understand the working of each task one by one prepare write up as per the template provided.

1.2 Learning Outcomes:

At the end learners will be able to:

- Describe the working of Servlets.
- Explain and demonstrate the architectural aspects involved in Servlets application development.
- They will be able to justify the Tasks performed by servlets as a middle layer.
- Describe and understand the servlet life-cycle.

1.3 Resources required

- Software: JDK, Eclipse for EE Developer, Tomcat server, web browser.

- Hardware: you need to specify as appropriate.

1.4 Faculty/Instructor Role

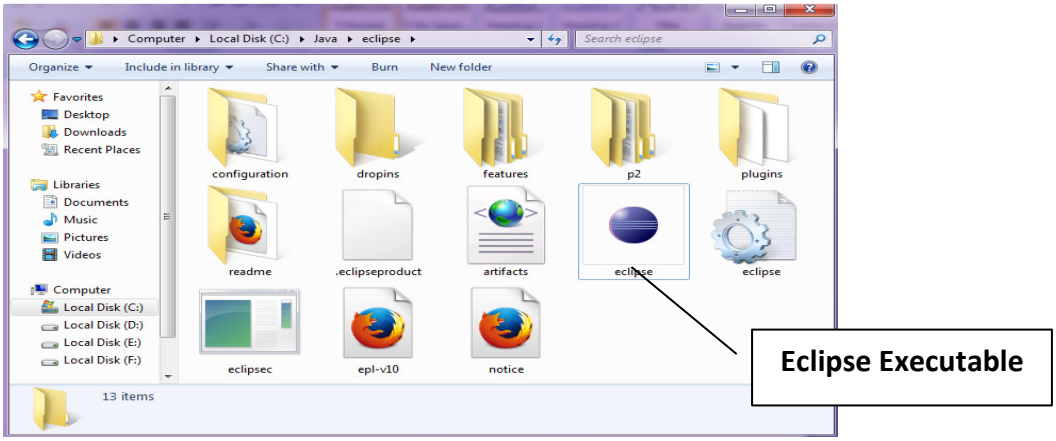
- Explain each concept to the learners.
- Make sure that every understood the concept in enough depth.
- Ensure the leaner's understandability by asking questions or by any other mechanism.
- Work as a mentor and supervisor when leaner implement the tasks.

1.5 Students/Learners Roles

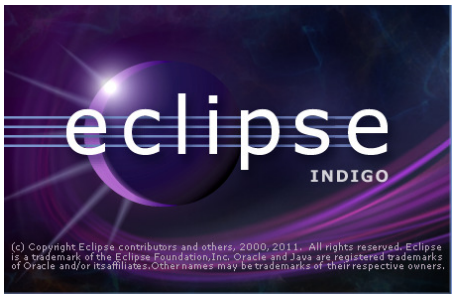
- Understand each and every concept with their implementation in enough details.
- Ask questions, if anything not clear.
- Implement each task and show the working.
- Explain the observations during the implementation and after implementation.
- Prepare write up according to the template given.
- Ready for answer the questions asked during the discussion session.

1.6 Steps Involved

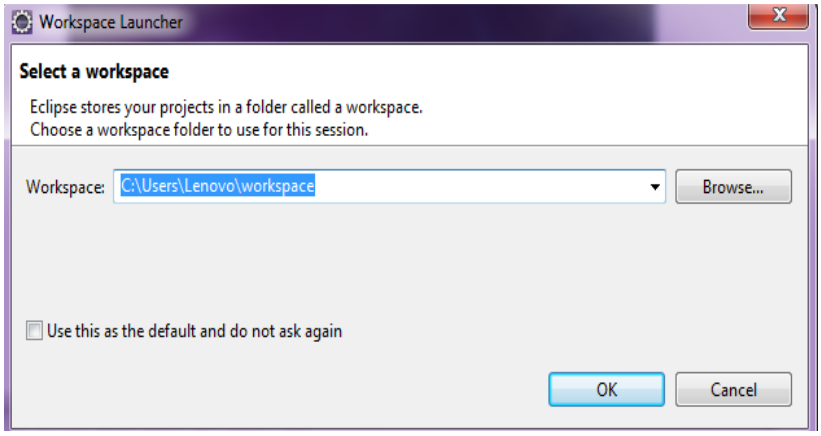
Step-1: Download Eclipse EE Developer and install in your machine by unzipping.



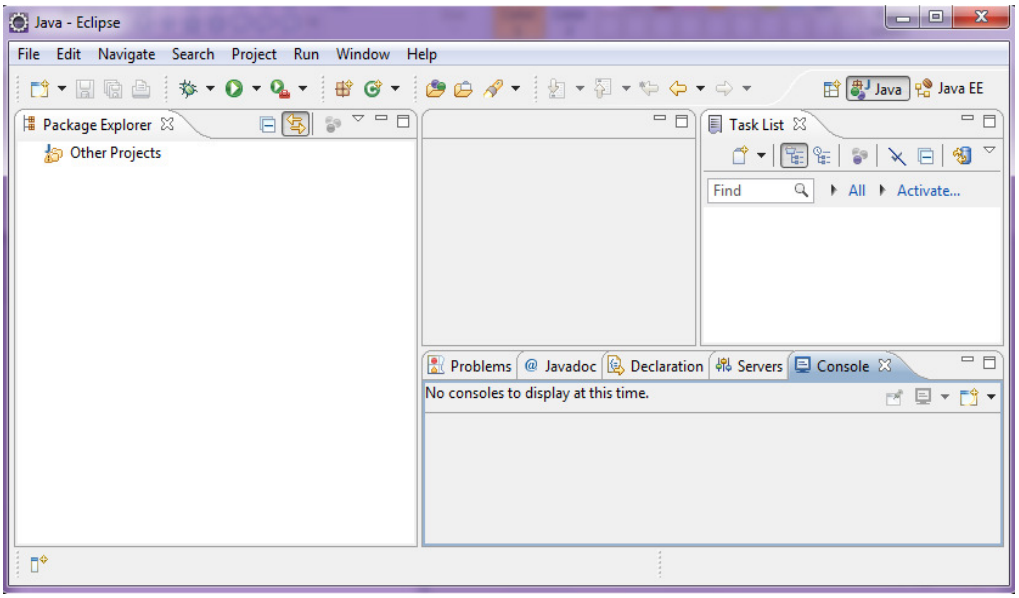
Step-2: open Eclipse executable



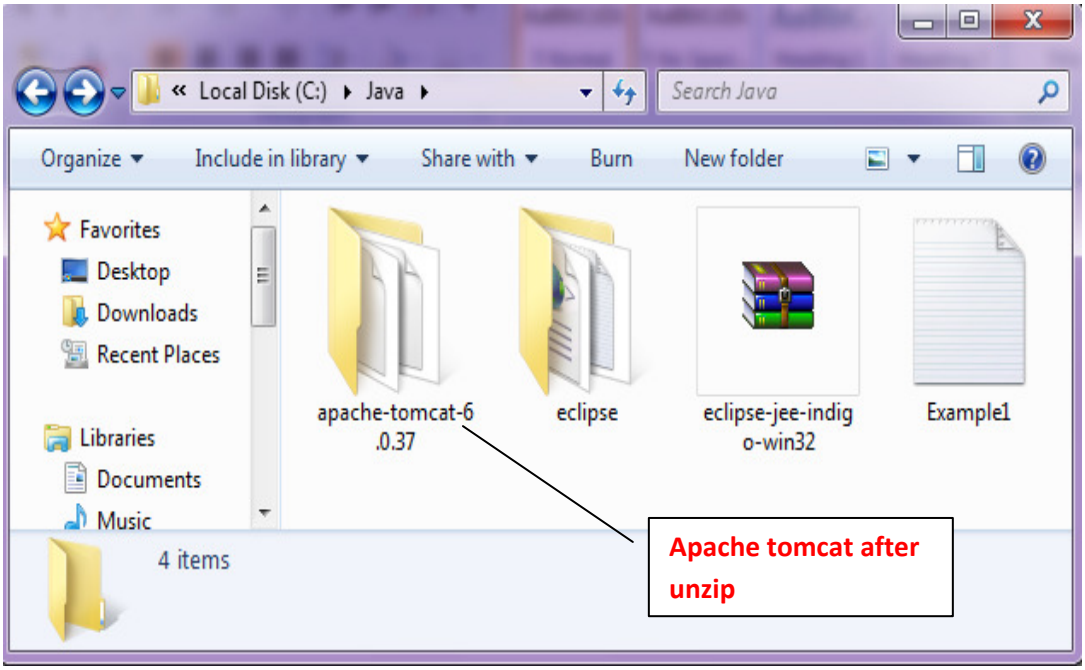
Step3: Select the workspace



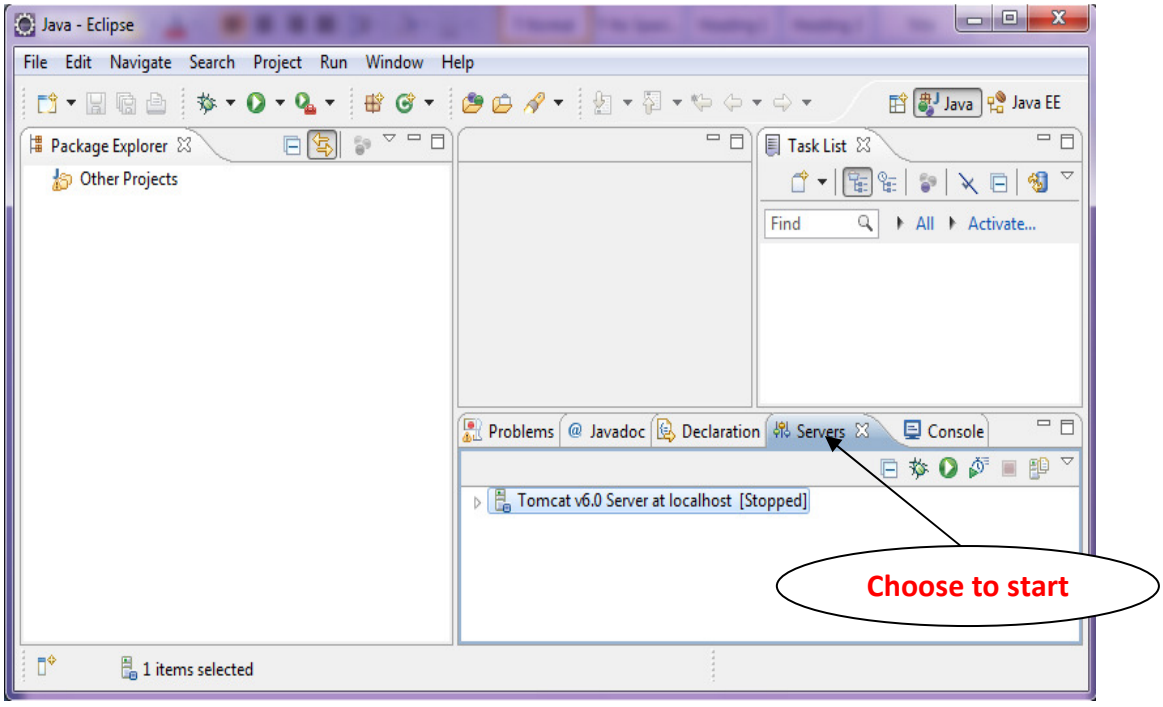
Step4: once you will choose the workspace you will see the IDE as shown below:



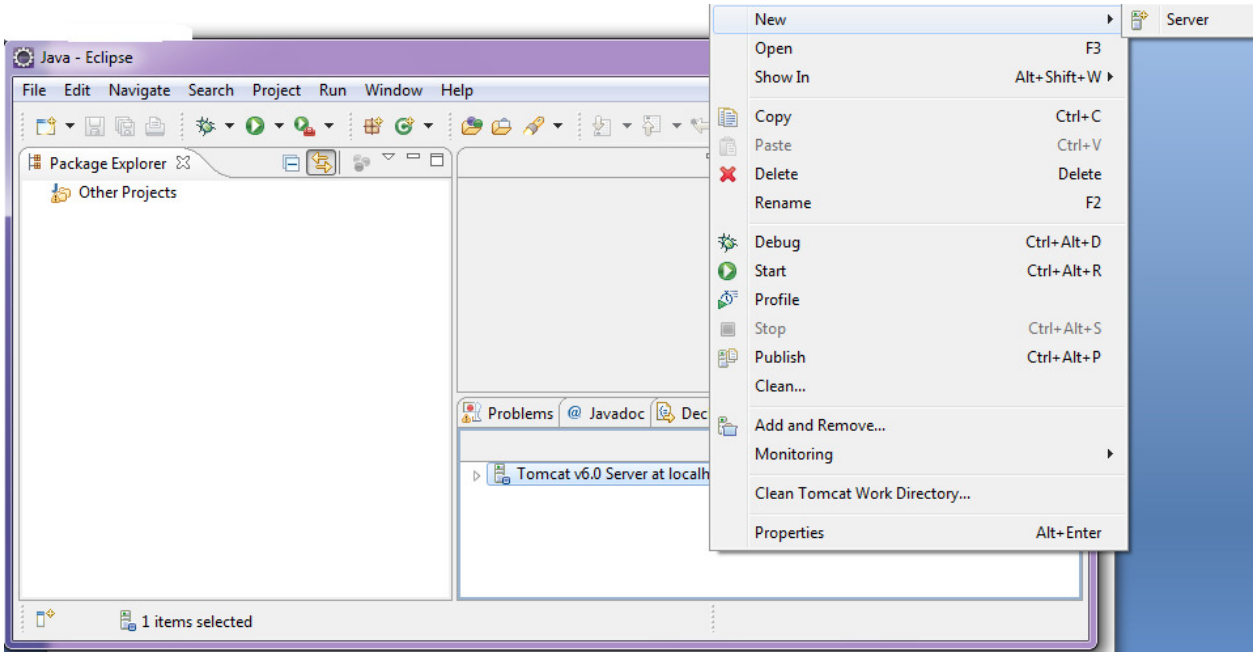
Step5: Download Tomcat and unzip.



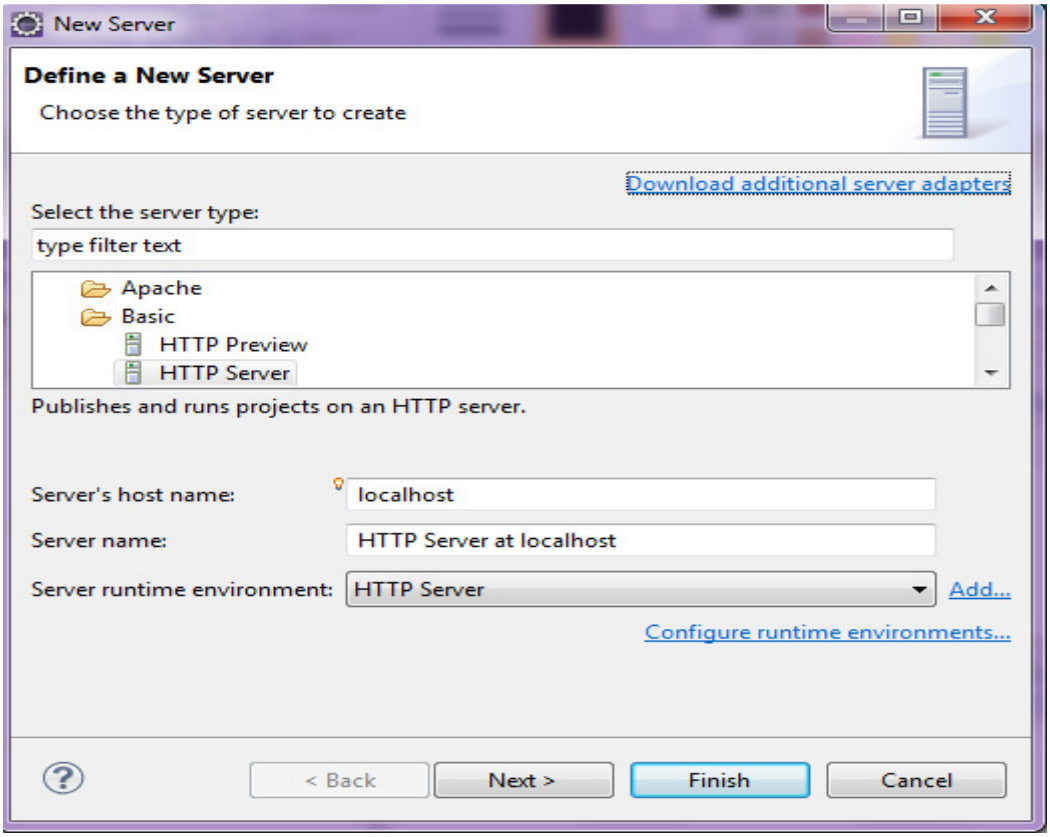
Step6: Starting the server using Eclipse.



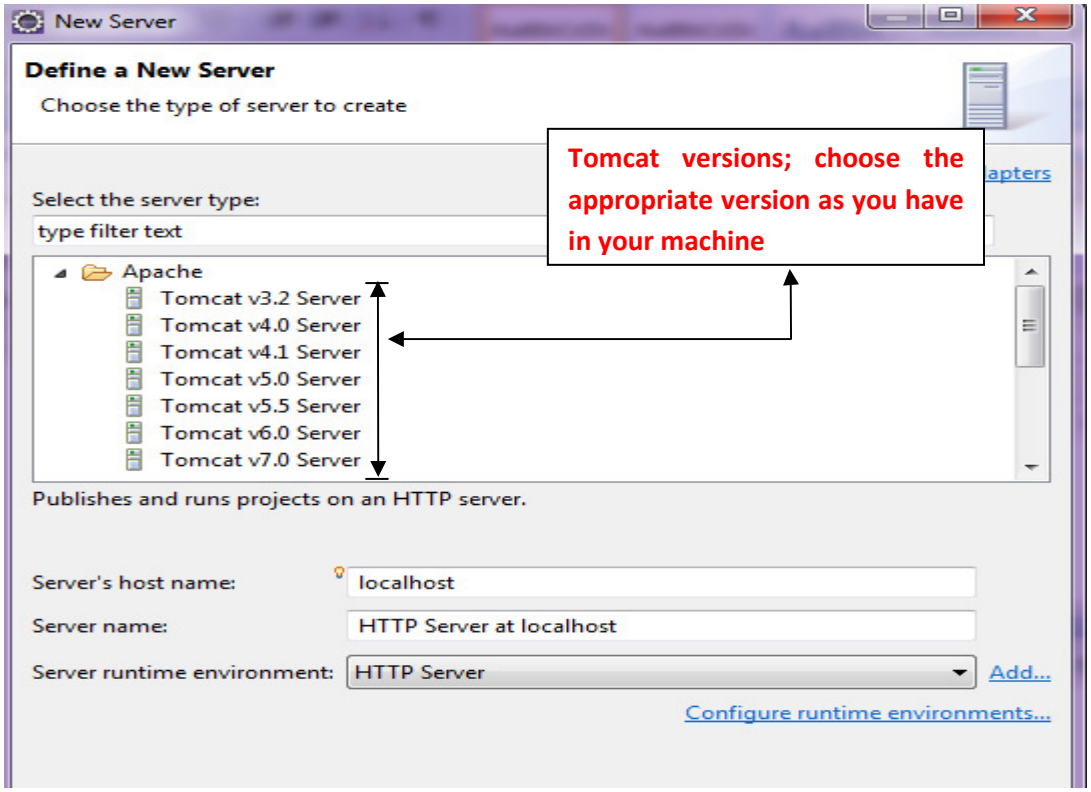
Step7: To configure the new server follow the steps as shown below:

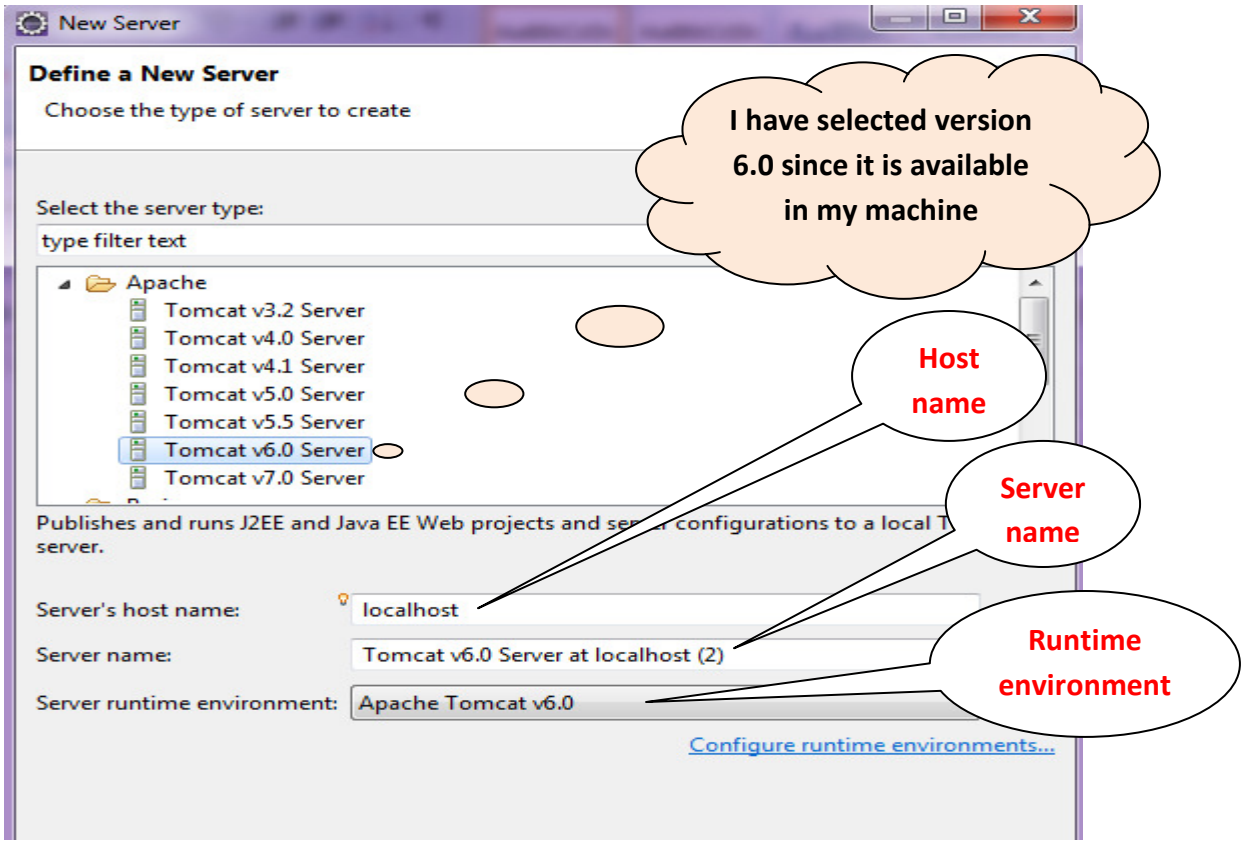


Step8: when you will click on server option you will see:

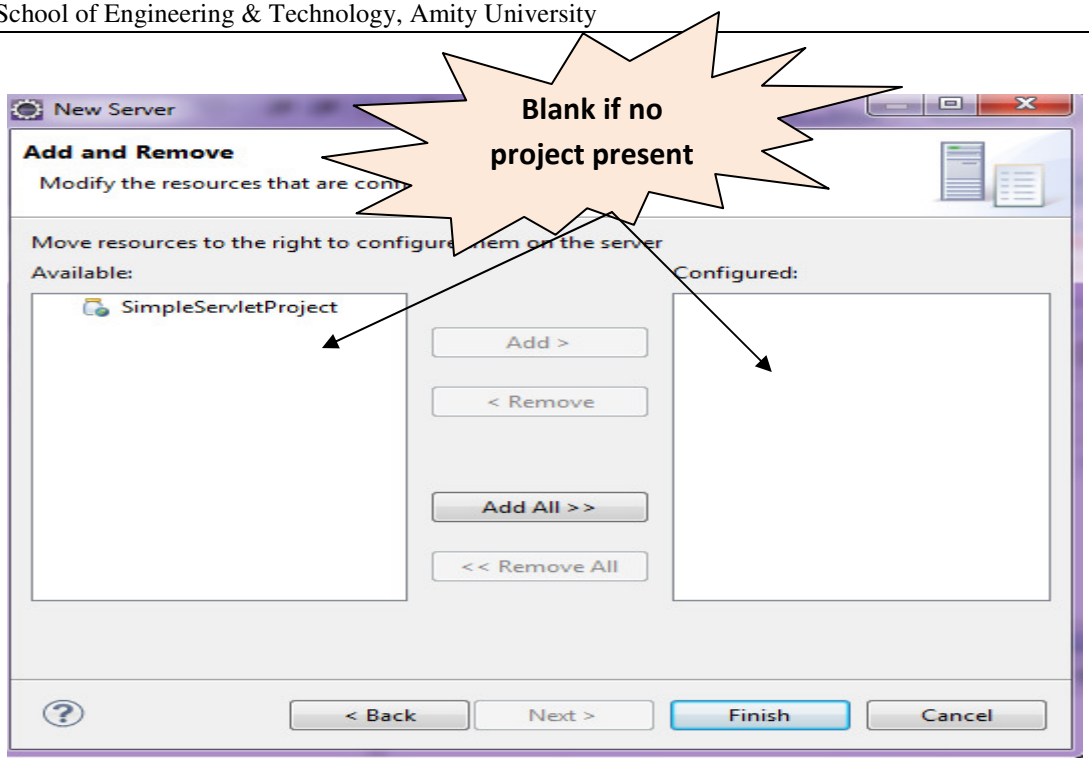


Step9: At this stage you need to choose the appropriate server you want. As we want to use Apache therefore we will chose the Tomcat serve.

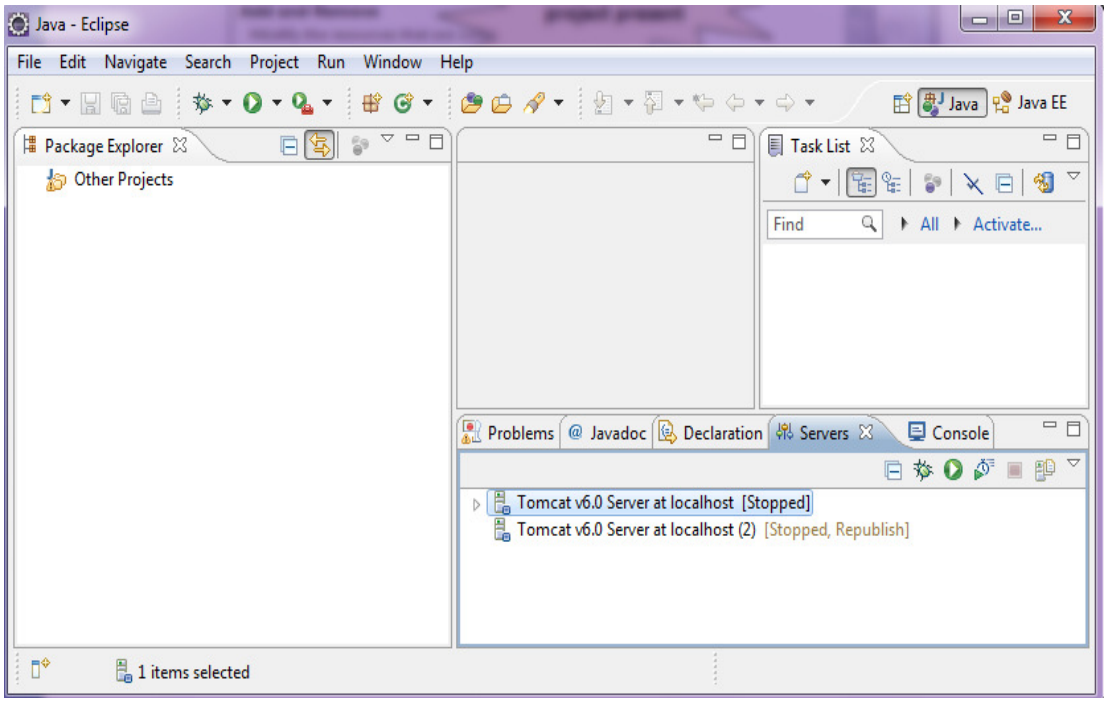




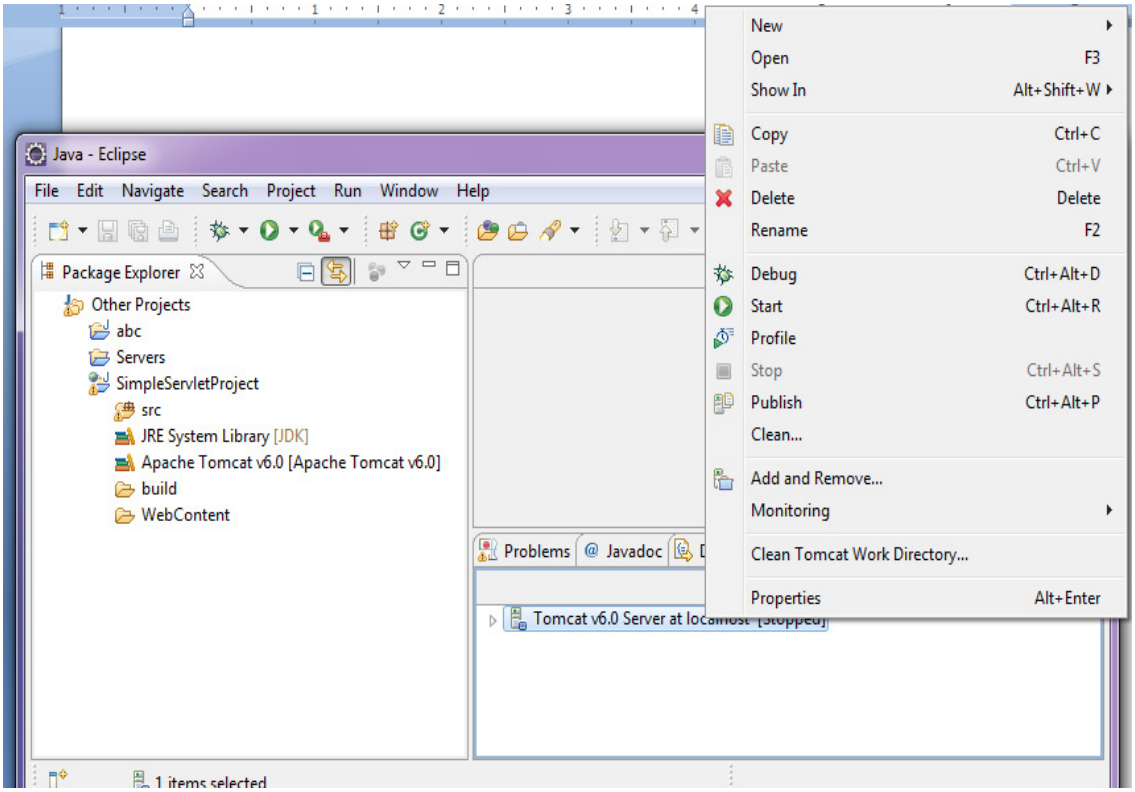
Now, click on next button.



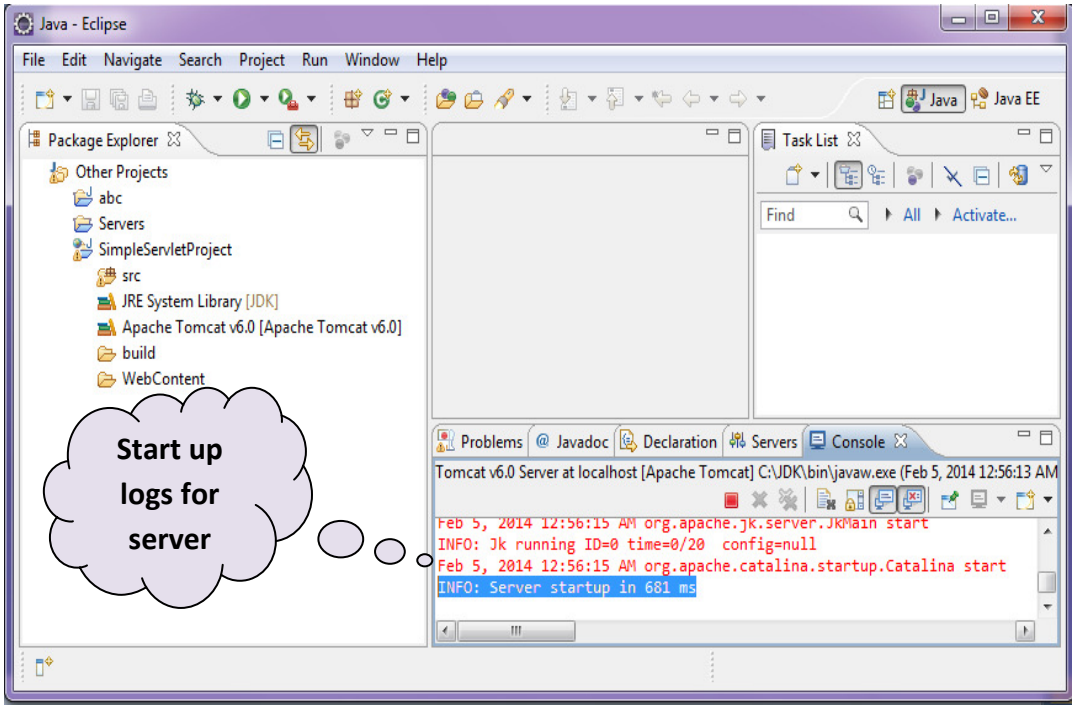
Then click on finish. Now, you can see the servers as shown in the below diagram.



Step10: click of start to start the server.



Step 11: Once, you will choose the start option, server will start and you can see the logs using console option.

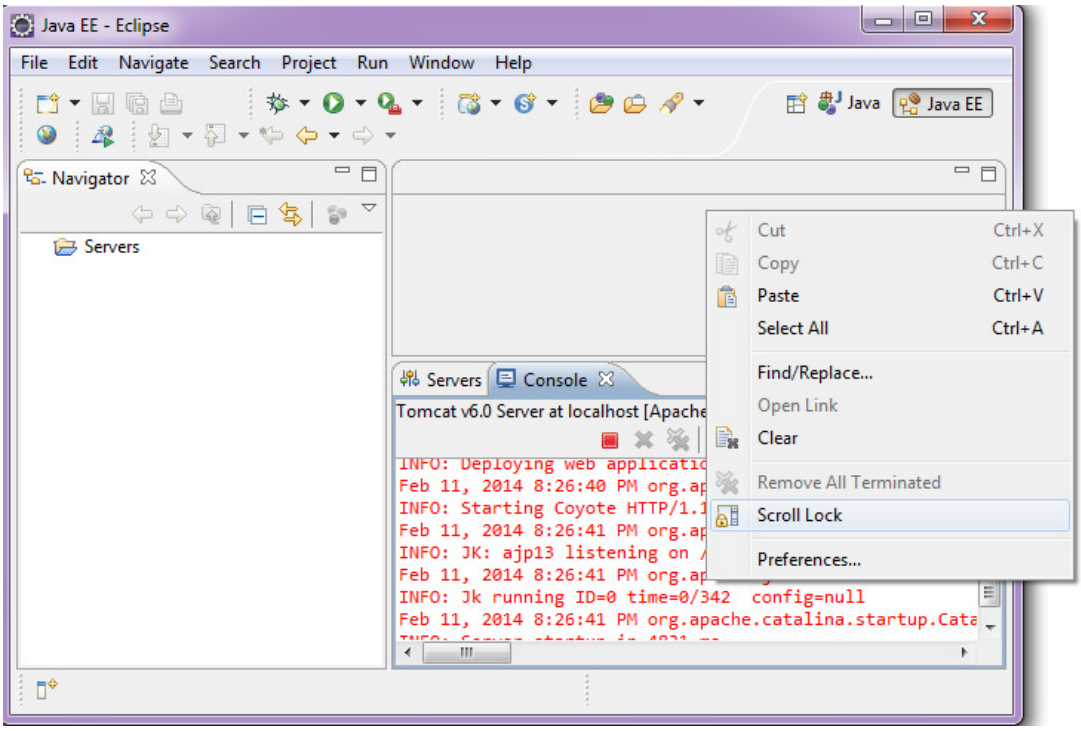


Now, at this stage your server is ready to use.

Creating a servlet application:

Now first stop the tomcat.

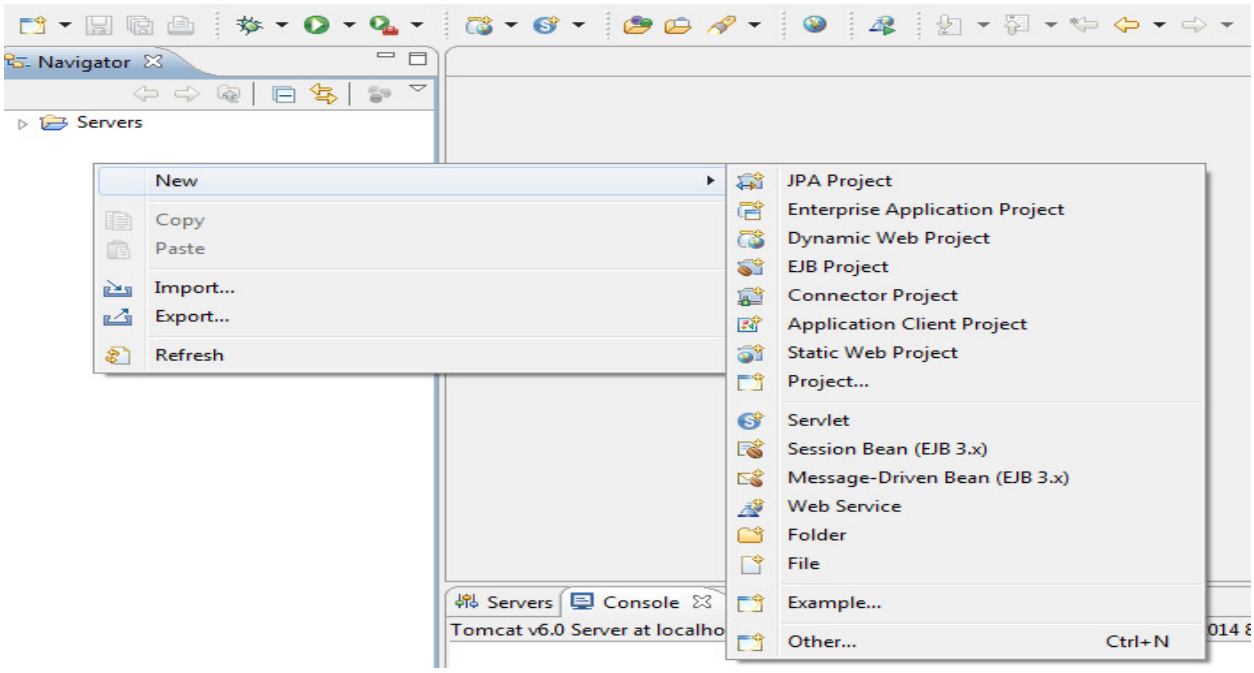
- **Right click and then choose clear.**



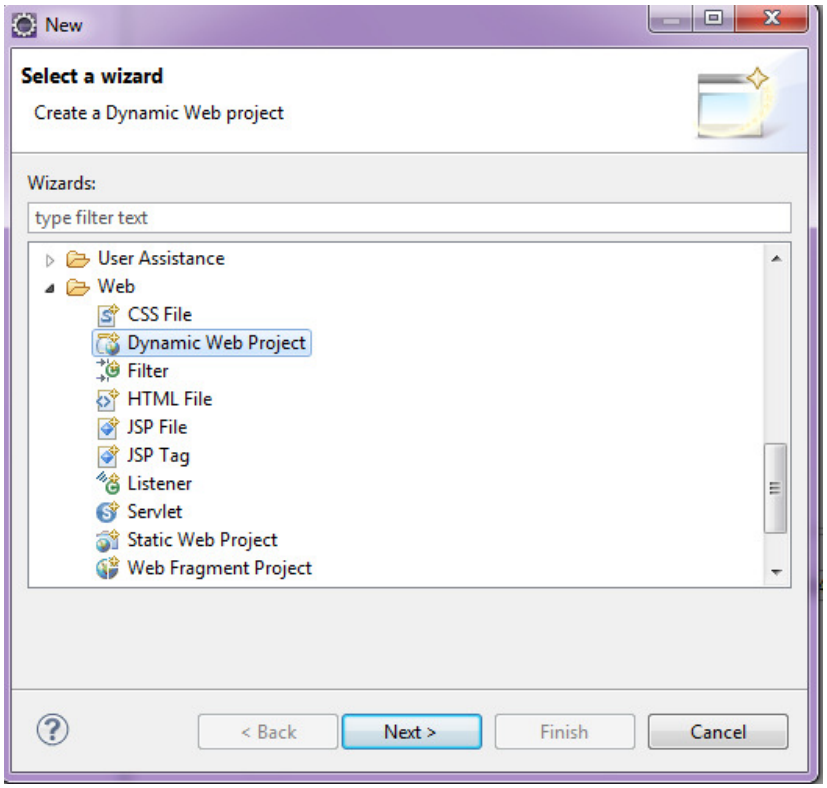
Now create a project just like any other java applications. In this case we will choose Java web application project.

- **Go to the package explorer tab right click, select new go to other.**

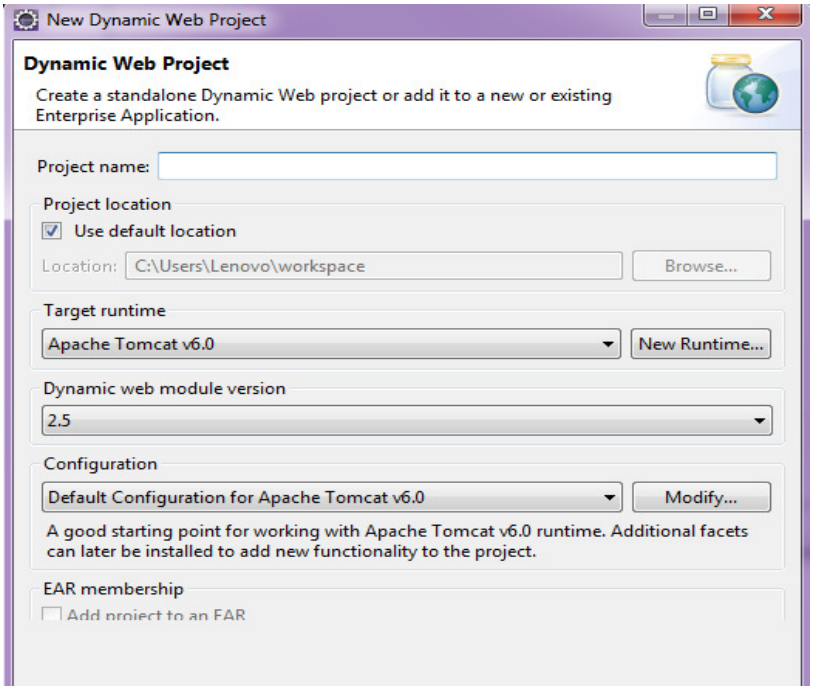
You will see this only if you have installed Eclipse for Java EE Developers. When you will go to the Eclipse URL you will see many packages such as Eclipse for Java Developer, Eclipse for EE Developer and lot more. It is recommended that you should download Eclipse for Java EE Developer.



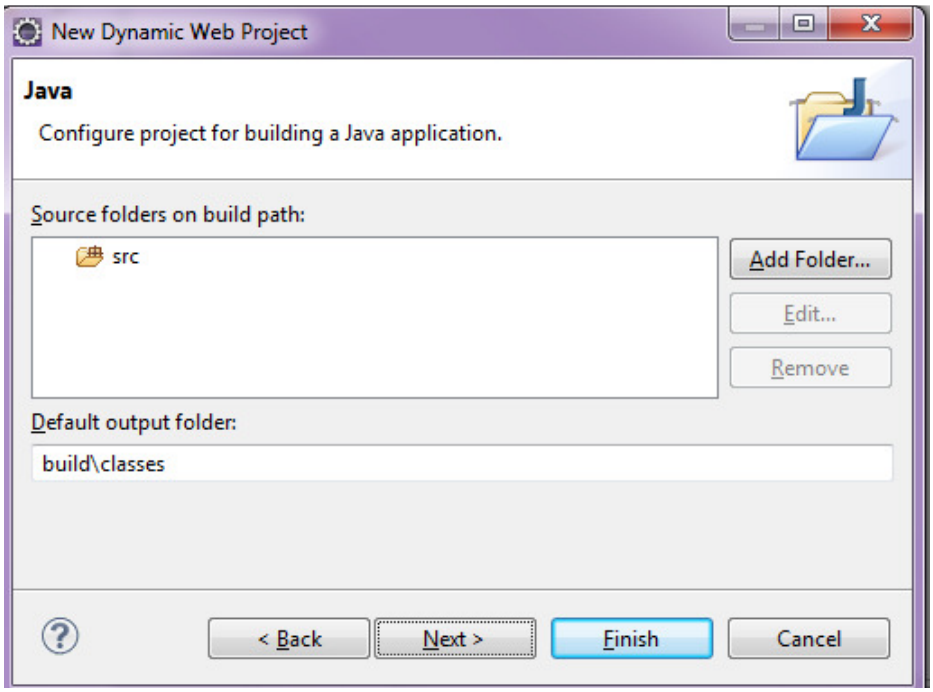
What we are interested in this is web folder. This has Dynamic Web Project; this is what we will select here and then choose next option.



Now, it will ask for a project name.

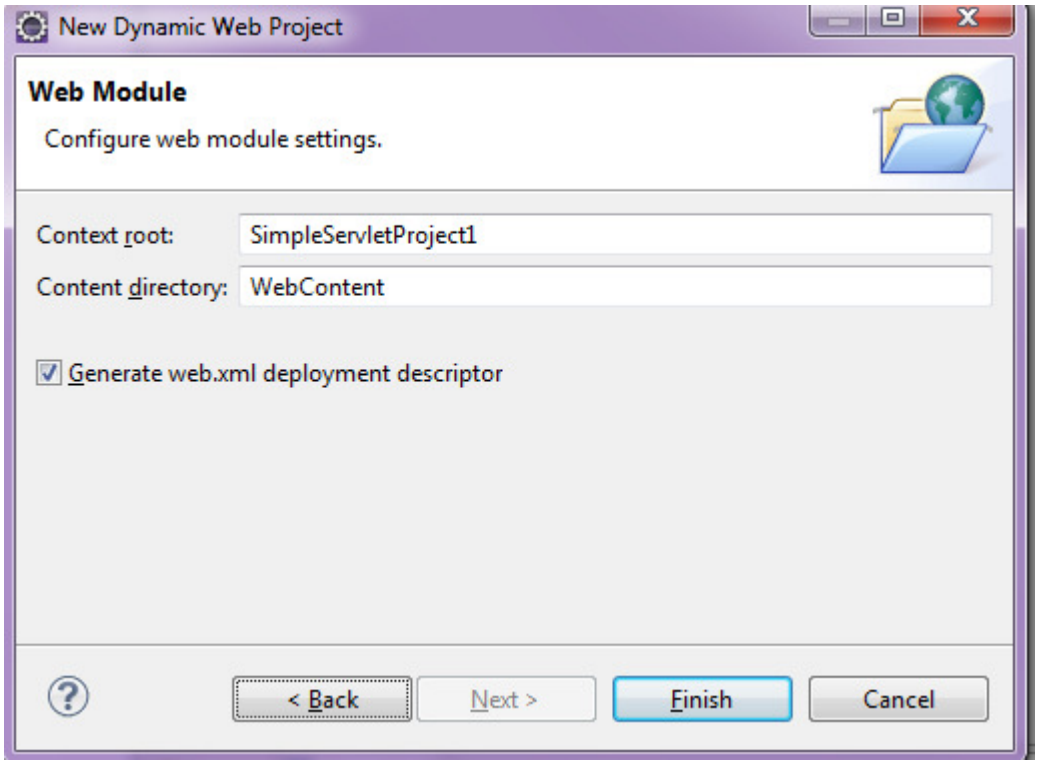


Here give the suitable name for the project. It will ask for default location, if you want any location for your choice you can set here. It will also ask for Target runtime, you are happy with the default runtime set it as default otherwise choose as appropriate. You can also set the Dynamic we module version at this stage. The last option which we will see here is configuration which will be the tomcat chosen by you. Now hit next.

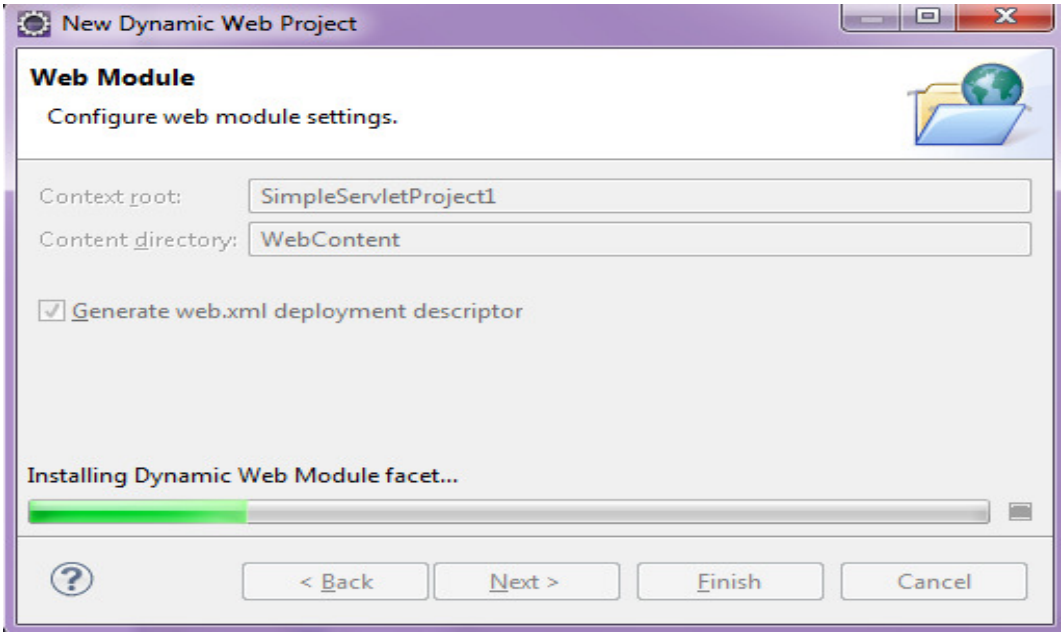


Now, it will ask for source folder where the source files will get saved. I will choose src. At this stage I am telling Eclipse that I want to save all my source code in src folder.

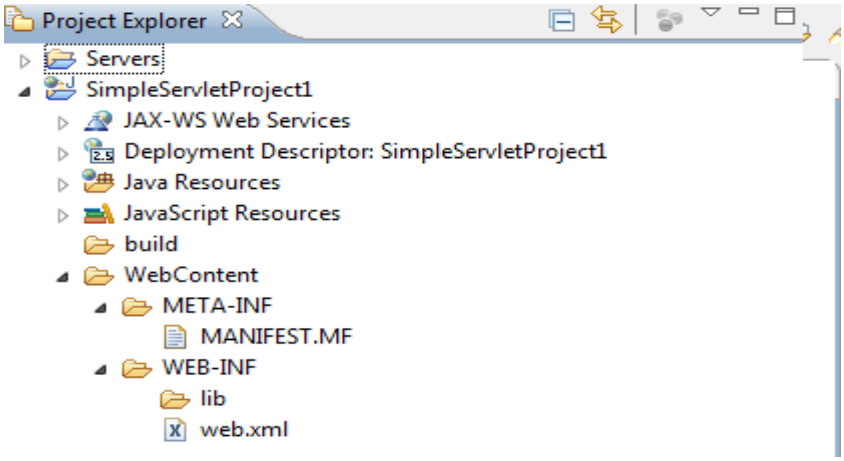
The Default output folder is build\classes again I am leaving it as it is. Now press next.



Now, it will ask me for context root and content directory. Here we will check on Generate web.xml deployment descriptor. And finally click “Finish”.



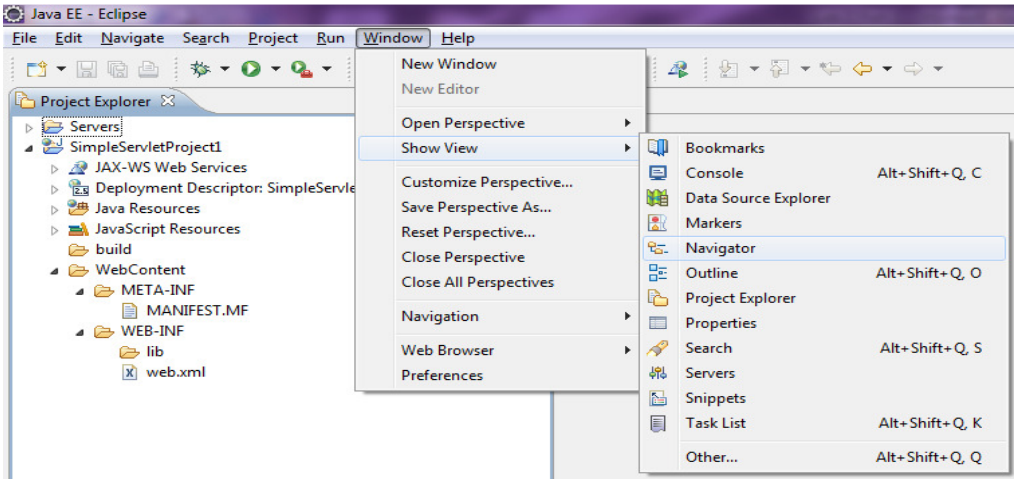
So finally you can see the project you created in the project explorer

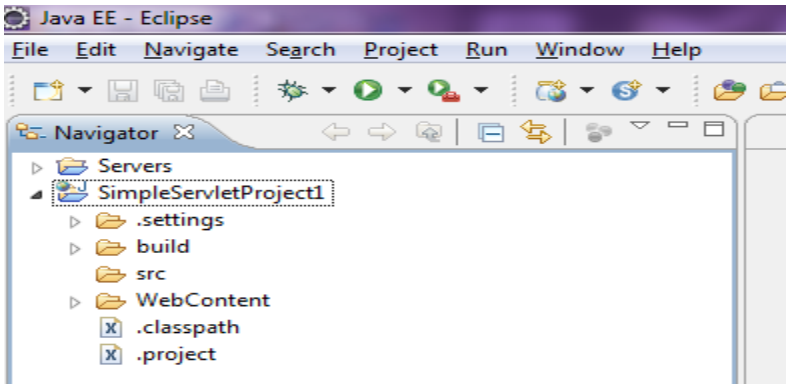


In the beginning we are not interested in all the nodes given but we are interested in Deployment Descriptor, this is actually an XML file.

Now let's open the navigator.

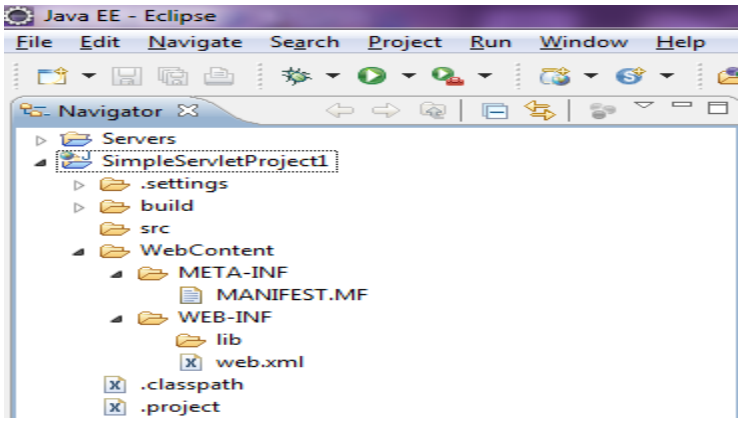
- Go to window→ Show view→ Navigator



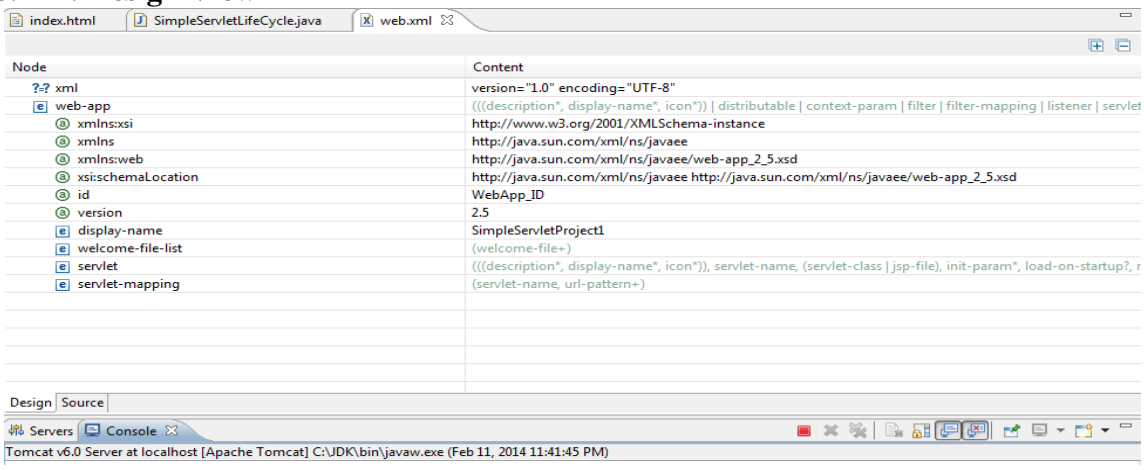


Here, .classpath and .project are Meta data for eclipse.

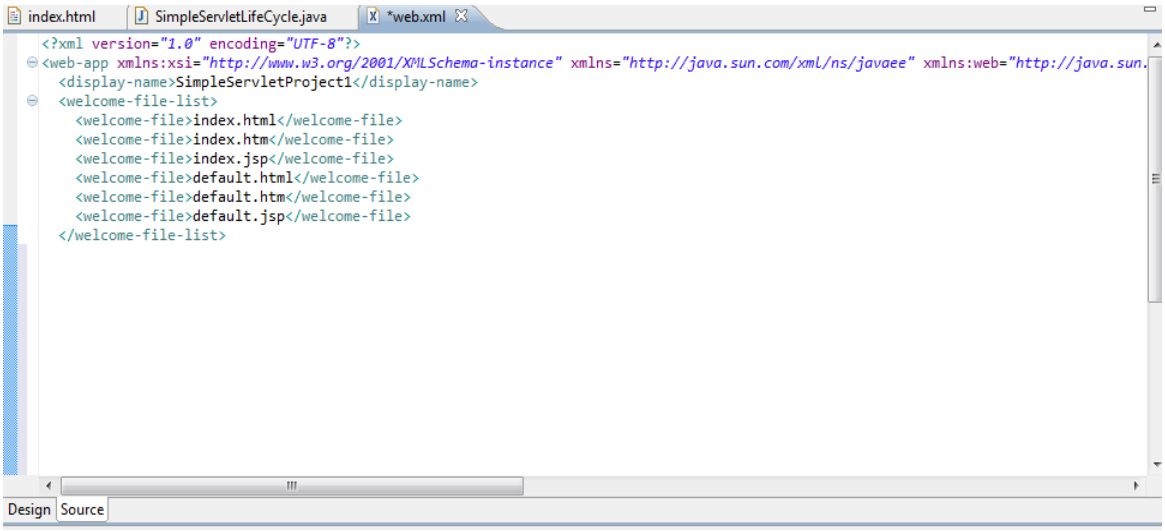
- **Go to the WebContent folder and expand it and see the WEB-INF file.**



Web.xml: Design View



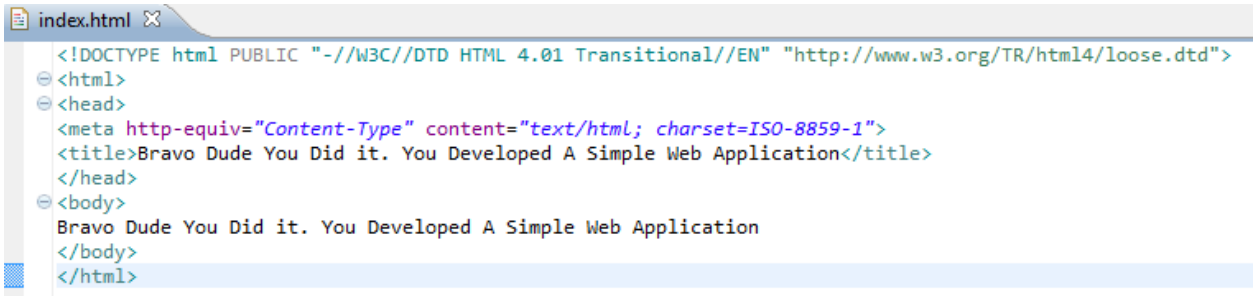
Web.xml: Source View: shows welcome file details



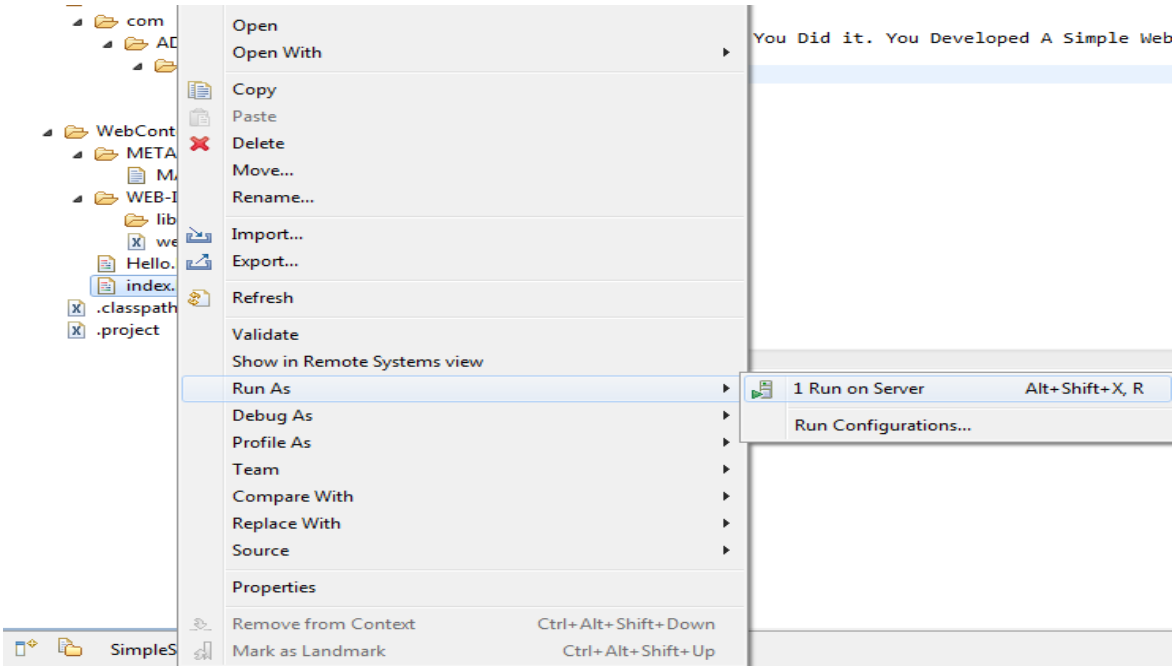
➢ Writing new HTML file named: index.html

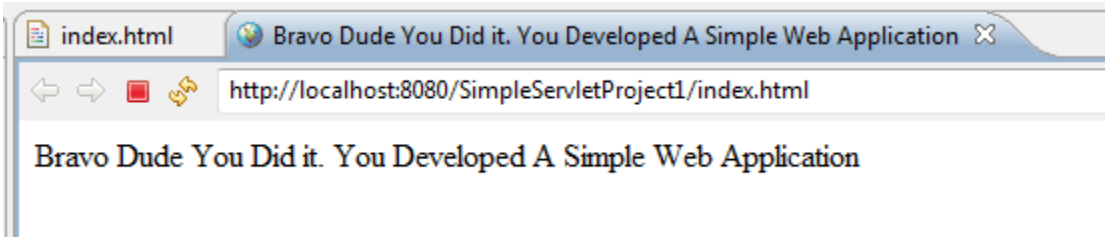
If you are interested in showing your own welcome message then you may create/write your own html file named index.html as shown below:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Bravo Dude You Did it. You Developed A Simple Web Application</title>
</head>
<body>
Bravo Dude You Did it. You Developed A Simple Web Application
</body>
</html>
```

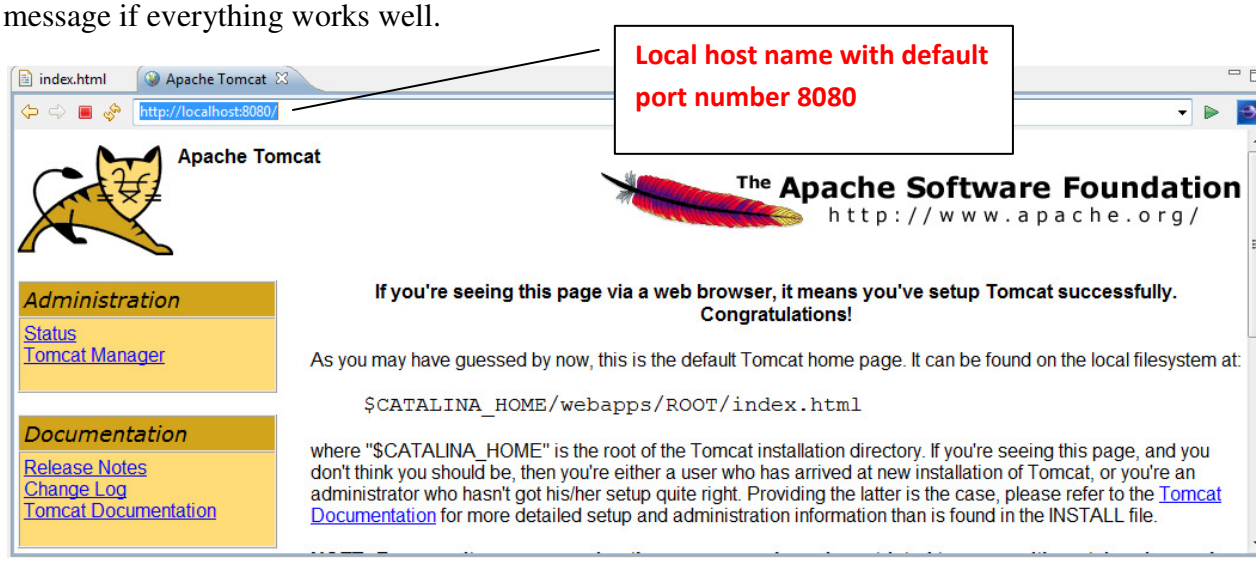


In order to run this HTML file we will use tomcat server (deploy with tomcat server) as shown below:

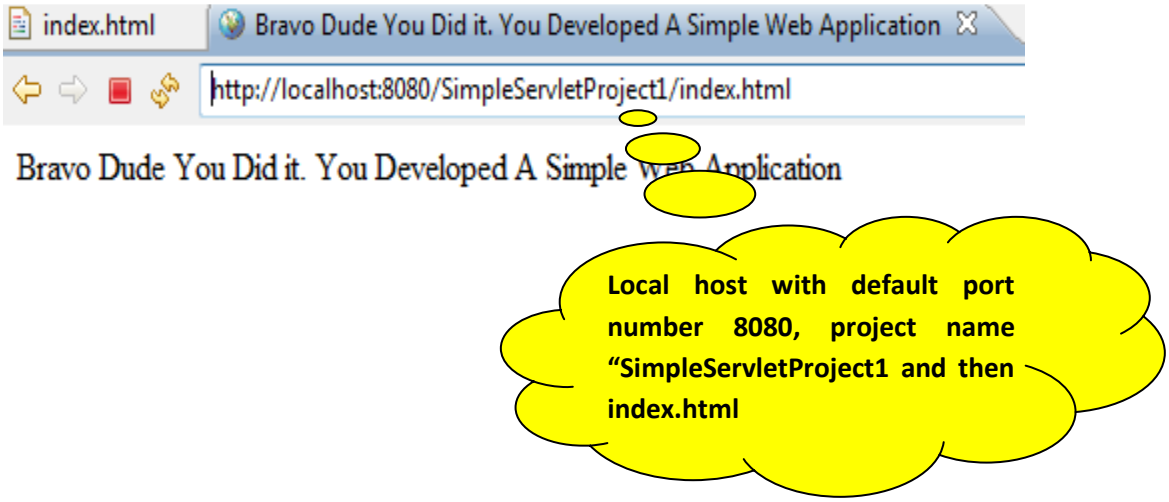




It is running using the localhost: 8080 port which is nothing but the default port for tomcat server. If we will use localhost: 8080 then we can see the cataline property i.e. the congratulation message if everything works well.

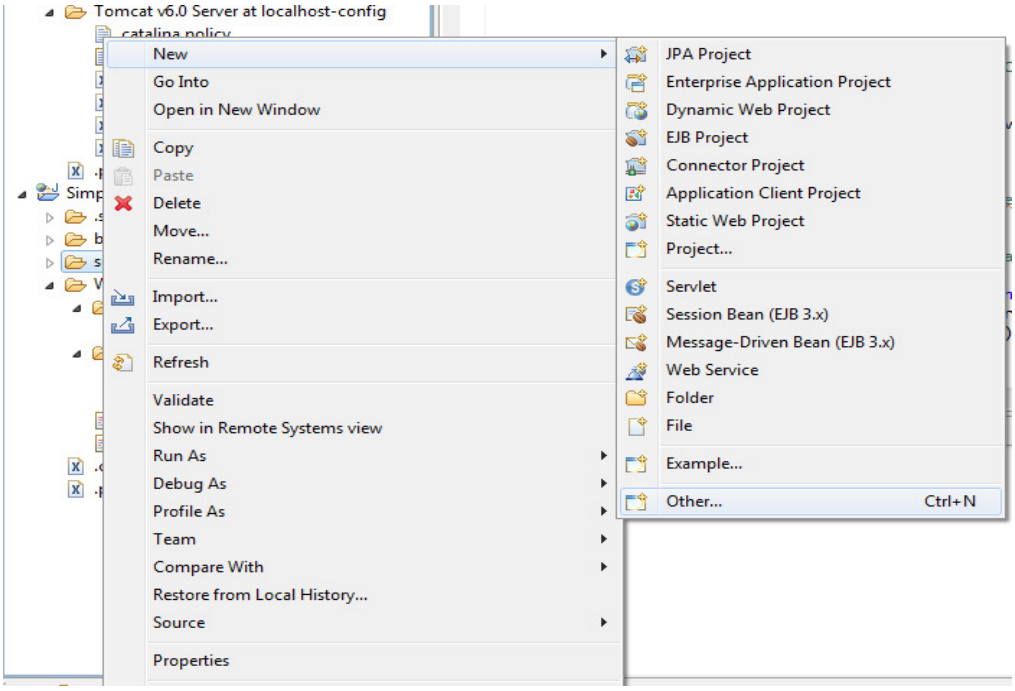


Now if you want to see the result of the html file then simply type the project name and then html file name and then hit the enter key.

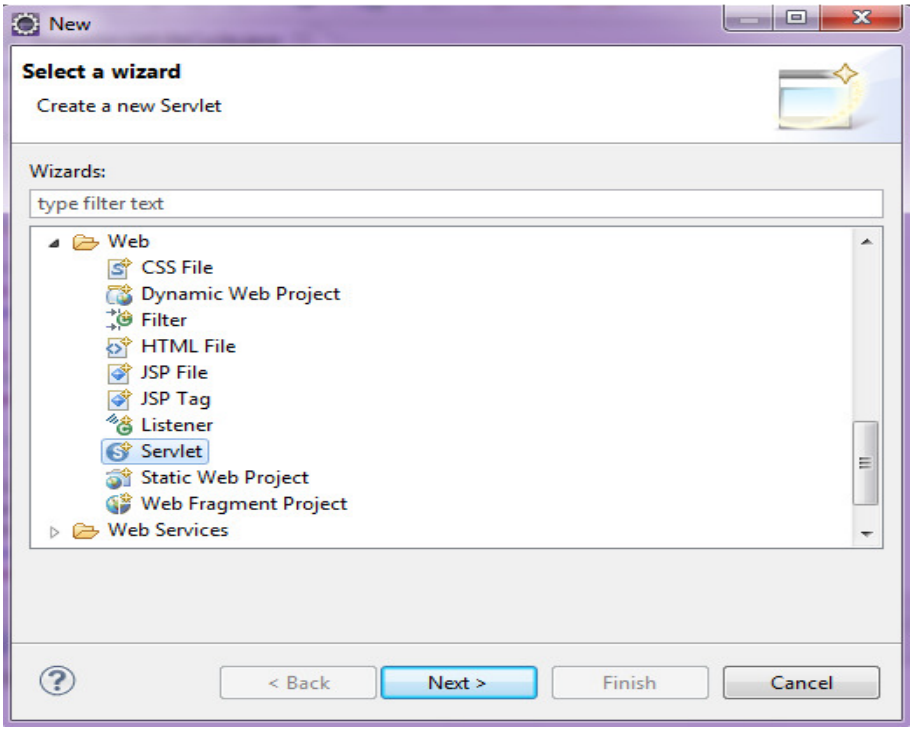


Now at this stage I would recommend you to develop a servlet program to see the "Servlet life cycle".

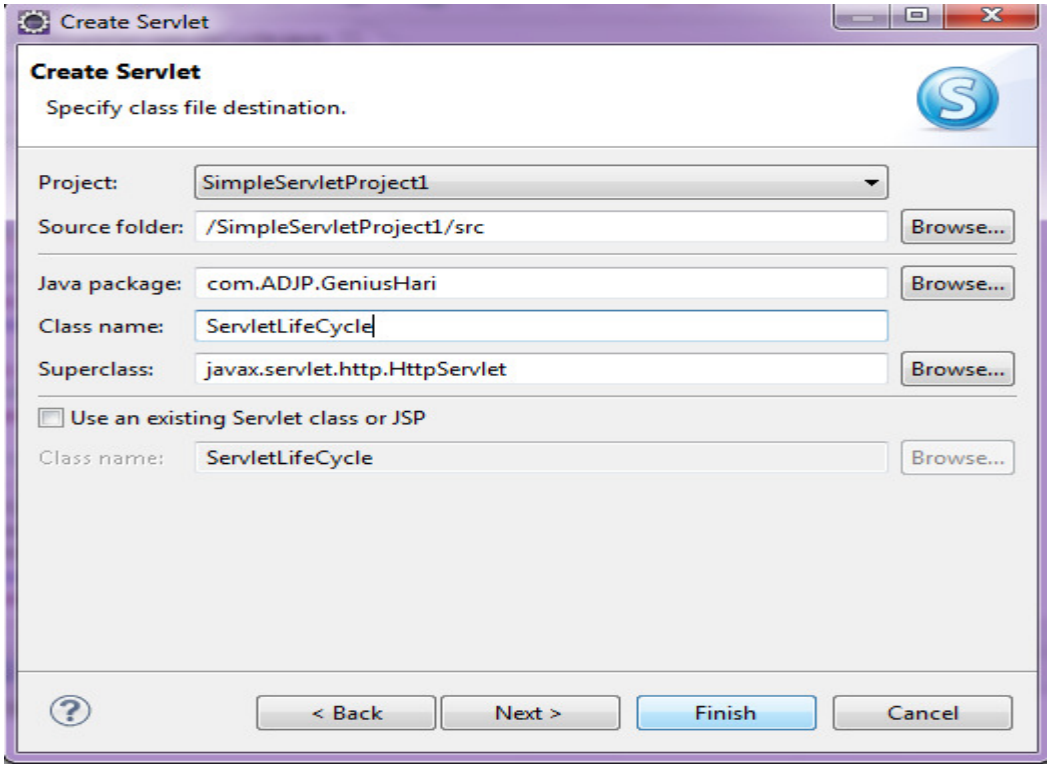
- **Go to src folder in the project explorer right click on it select new and then other.**



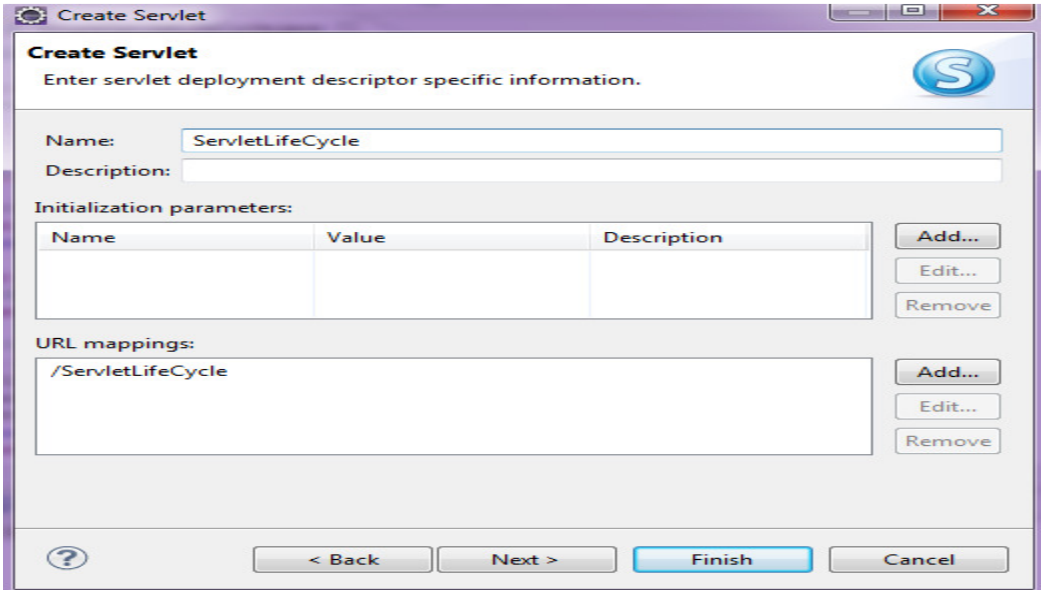
You will see:



- **Go to web folder, choose Servlet and hit on next button. Now you will see:**



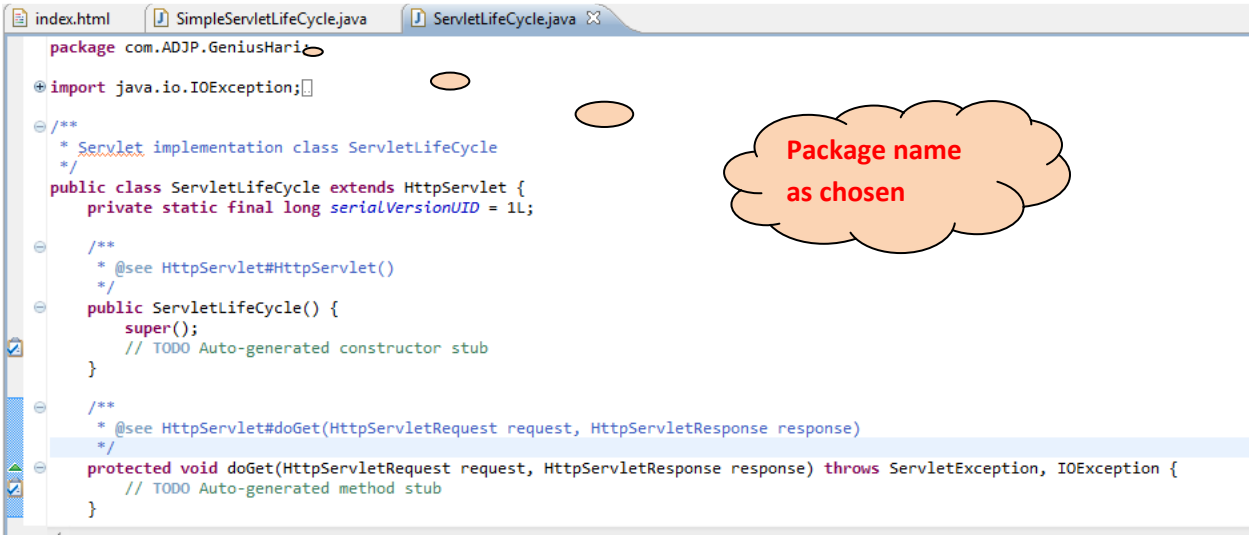
Here, write package name for example I have given "com.ADJP.GeniusHari" and then choose the appropriate class name of your choice. At last you have to choose the Superclass option, if you want to go with default then no need to change. And then hit on Next



Here, you can initialize the parameters if you are interested otherwise you can do the same within the code. In addition, you can set the URL mapping option, the purpose of this step is to make web.XML to understand what to render because during the implementation there is

possibility that many applications are using the server and therefore this mapping option will tell the server to run the particular application which user/program/developer want to execute at any particular time.

Finally hit the Finish button. At this stage you will be able to see the following:



This is nothing but the servlet file with doGet and doPost methods. Here, you can write the code as you want and execute.

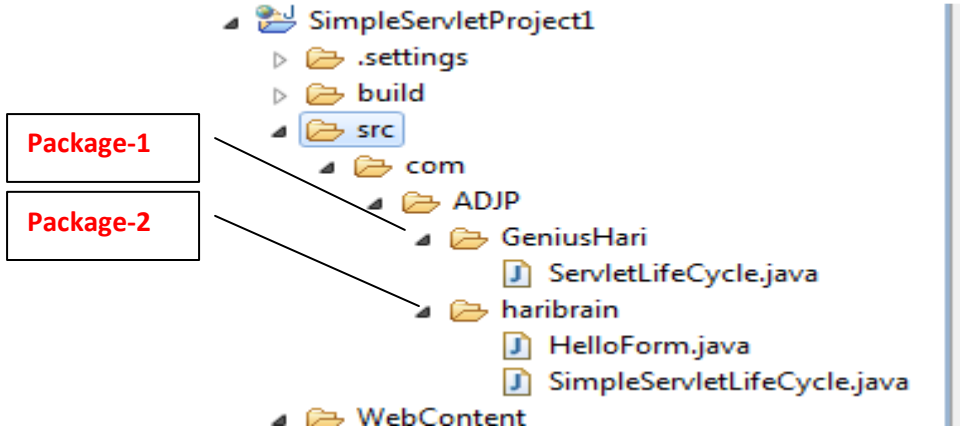


Figure: packages named **GeniusHari** and **haribrain** lies in **ADJP** which is in **com** and stored in **src** folder.

➢ Servlet Life Cycle

```
package com.ADJP.haribrain;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.*;
import javax.servlet.*;
//import javax.servlet.http.*;


//**
 //* Servlet implementation class SimpleServletLifeCycle
 //*/
@SuppressWarnings("serial")
public class SimpleServletLifeCycle extends HttpServlet {
        int count;

         public void init(ServletConfig config) throws ServletException {
           // Always call super.init(config) first  (servlet mantra #1)
           super.init(config);

           // Try to load the initial count from our saved persistent state
           try {
             FileReader fileReader = new FileReader("InitDestroyCounter.initial");
             BufferedReader bufferedReader = new BufferedReader(fileReader);
             String initial = bufferedReader.readLine();
             count = Integer.parseInt(initial);
             return;
           }
           catch (FileNotFoundException ignored) { }  // no saved state
           catch (IOException ignored) { }            // problem during read
           catch (NumberFormatException ignored) { }  // corrupt saved state

           // No luck with the saved state, check for an init parameter
           String initial = getInitParameter("initial");
           try {
             count = Integer.parseInt(initial);
             return;
           }
           catch (NumberFormatException ignored) { }  // null or non-integer value

           // Default to an initial count of "0"
           count = 0;
         }

         public void doGet(HttpServletRequest req, HttpServletResponse res)
                                     throws ServletException, IOException {
           res.setContentType("text/plain");
           PrintWriter out = res.getWriter();
           count++;
           out.println("Since the beginning, this servlet has been accessed " +
                       count + " times.");
         }

         public void destroy() {
           saveState();
         }
```

```
        public void saveState() {
          // Try to save the accumulated count
          try {
            FileWriter fileWriter = new FileWriter("InitDestroyCounter.initial");
            String initial = Integer.toString(count);
            fileWriter.write(initial, 0, initial.length());
            fileWriter.close();
            return;
          }
          catch (IOException e) {  // problem during write
            // Log the exception. See.
          }
        }
      }
```

Output:

Writing cookies

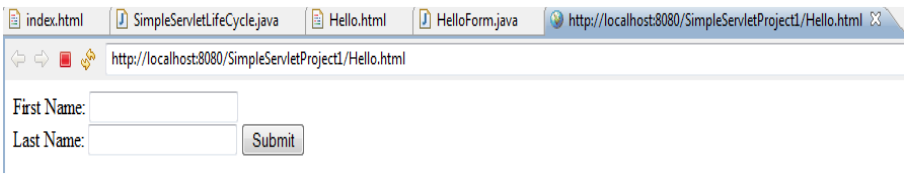
```
package com.ADJP.haribrain;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.*;
//import javax.servlet.*;
import javax.servlet.http.*;

/**
 * Servlet implementation class HelloForm
 */
@SuppressWarnings("serial")
public class HelloForm extends HttpServlet {
        public void doGet(HttpServletRequest request,
                        HttpServletResponse response)
                        throws ServletException, IOException
                        {

                    // Create cookies for first and last names.
                    Cookie firstName = new Cookie("first_name",
request.getParameter("first_name"));
                        Cookie lastName = new Cookie("last_name",
request.getParameter("last_name"));

                        // Set expiry date after 24 Hrs for both the cookies.
                        firstName.setMaxAge(60*60*24);
```

```
lastName.setMaxAge(60*60*24);

// Add both the cookies in the response header.
response.addCookie( firstName );
response.addCookie( lastName );

// Set response content type
response.setContentType("text/html");
PrintWriter out = response.getWriter();
String title = "Setting Cookies Example";
String docType =
"<!doctype html public \"-//w3c//dtd html 4.0 " +
"transitional//en\">\n";
out.println(docType +
"<html>\n" +
"<head><title>" + title + "</title></head>\n" +
"<body bgcolor=\"#f0f0f0\">\n" +
"<h1 align=\"center\">" + title + "</h1>\n" +
"<ul>\n" +
" <li><b>First Name</b>: "
+ request.getParameter("first_name") + "\n" +
" <li><b>Last Name</b>: "
+ request.getParameter("last_name") + "\n" +
"</ul>\n" +
"</body></html>");
}
}
```

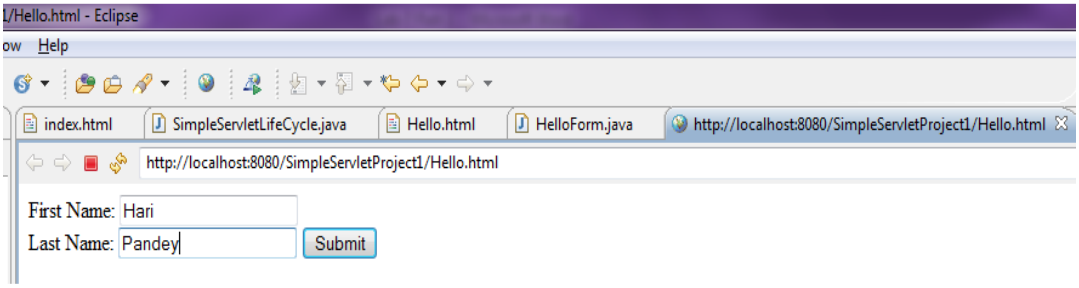
The html file

```
<html>
<body>
<form action="HelloForm" method="GET">
First Name: <input type="text" name="first_name">
<br />
Last Name: <input type="text" name="last_name" />
<input type="submit" value="Submit" />
</form>
</body>
</html>
```

Output of html file

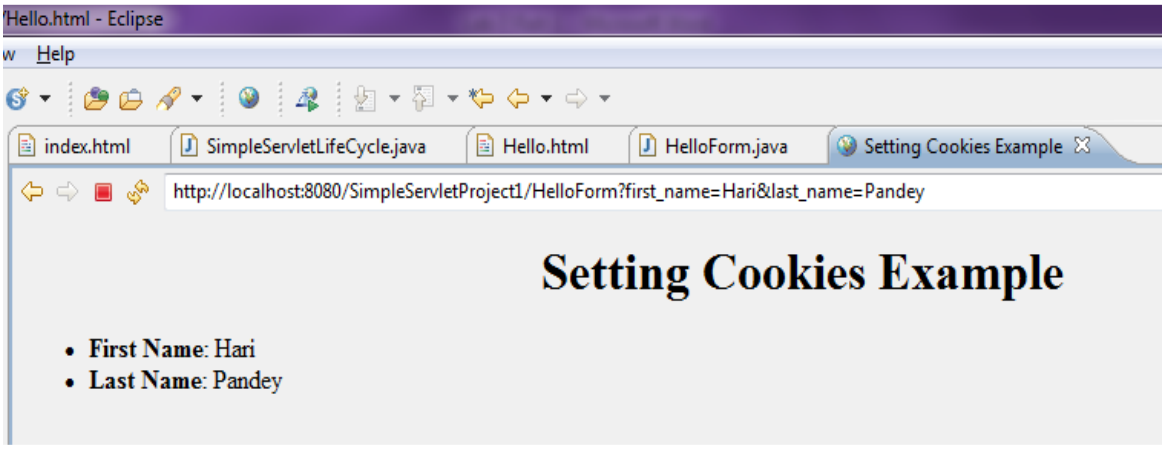


Enter the relevant data



Now hit on submit.

Once you will hit on submit the following screen should come.



Finally verify using web browser and understand the working of doGet, doPost, Cookies and other related concepts.