Lab 1 – Creating my first Java Web Application

Developing a Web Application using Eclipse for Java EE

A *web application* runs over the Internet on HTTP protocol with Browser as the client accessing to a HTTP server. In this practical, students will experience the use Eclipse together with Apache-Tomcat as the Web Server (Web container for your servlet) to create their first Java Servlet. Java Servlet is the simplest model to build a complete Java EE Web Application. To date, it is still a very important Java EE component that cannot be ignored.

Objectives:

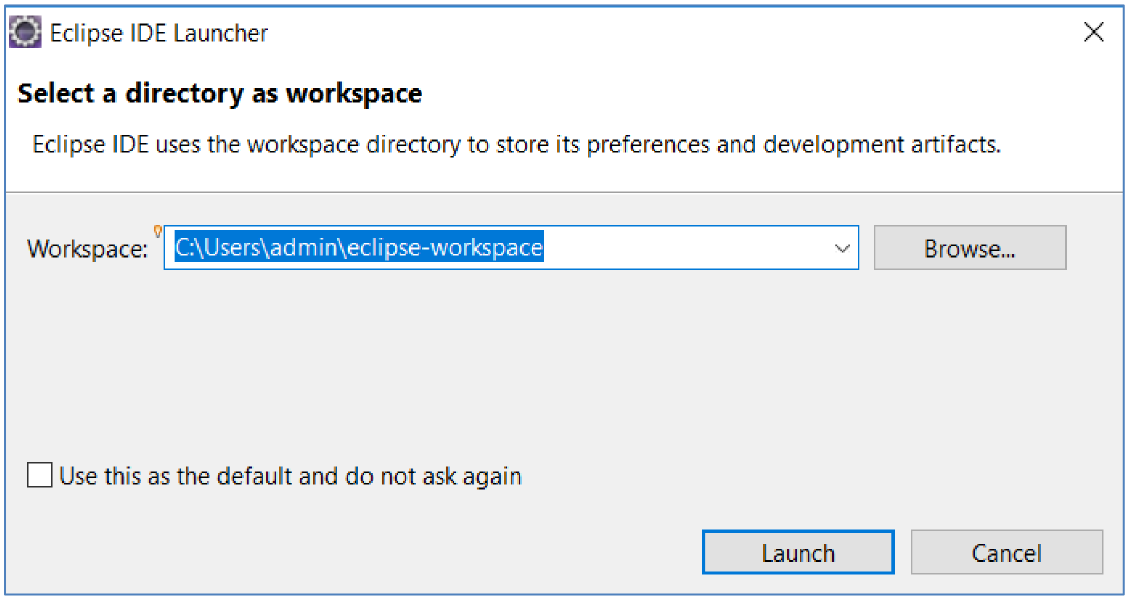
- ❑ Learn to use Eclipse for Java EE for web application Development
    - Familiarize with the Development Environment
    - Learn how to import/export the web application project
    - Learn how to use Eclipse for Java EE for development and testing

- ❑ Developing your first web application (Servlet) using Eclipse
    - Understand Web Application Structure
    - Creating a servlet
    - Retrieving user parameter using servlet
    - Perform user request and generate response

In this practical exercise, we will demonstrate how a JSP file **index.jsp** sends a HTTP request (GET/POST Method) to a Servlet name **LoginServlet** that checks for a matching credential. Index.jsp will be used to obtain the user id and password of the users. For this case, textboxes should be adequate as user id and password. The servlet contains programming (business logic) codes to read the HTTP parameter values and perform user id and password verification. Lastly, the servlet will perform a HTTP response (POST) back to the client as some HTML output.

1. Running Eclipse

- Locate your Eclipse and run it.
- Ensure that you have a valid JRE or JDK 1.8 before starting eclipse.


Eclipse IDE Launcher

Select a directory as workspace

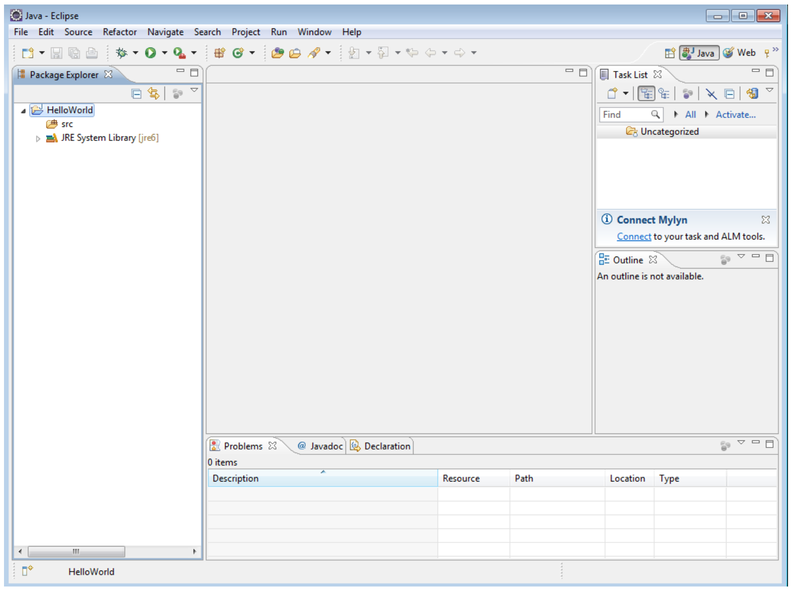
Eclipse IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Users\admin\eclipse-workspace

☐ Use this as the default and do not ask again

Perspectives, Views and Editors

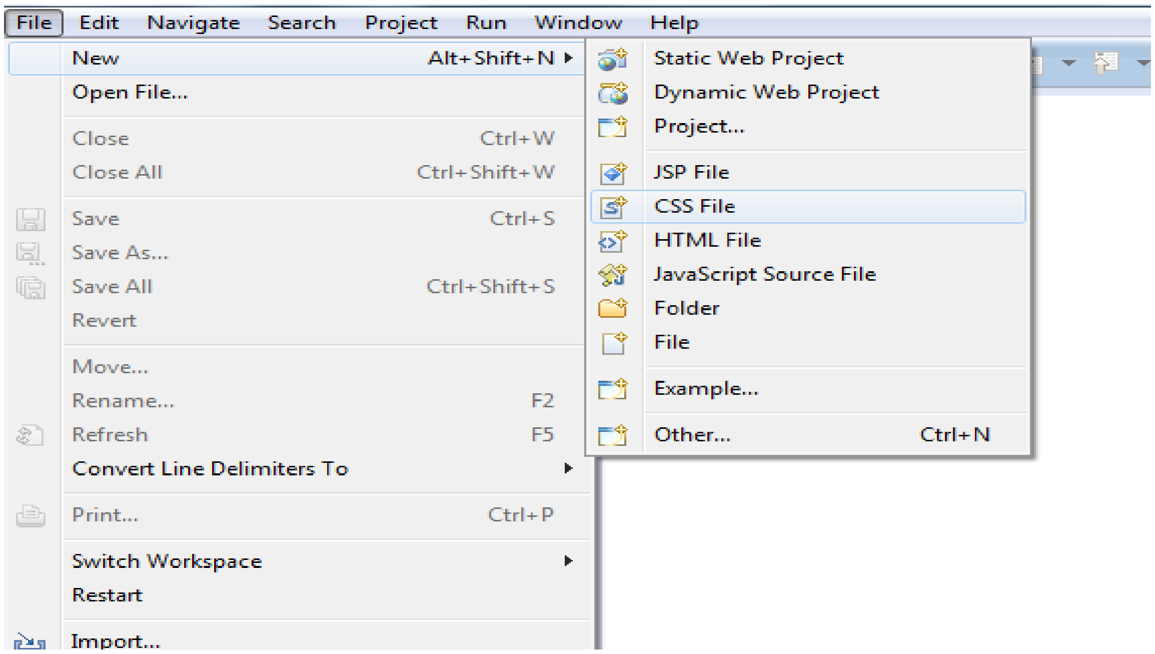
Perspectives are made up of views and editors. Each view has its own function and displays information about your project. For example, the *Package Explorer* view provides a tree allowing you to view all the files of your project nicely categorized.

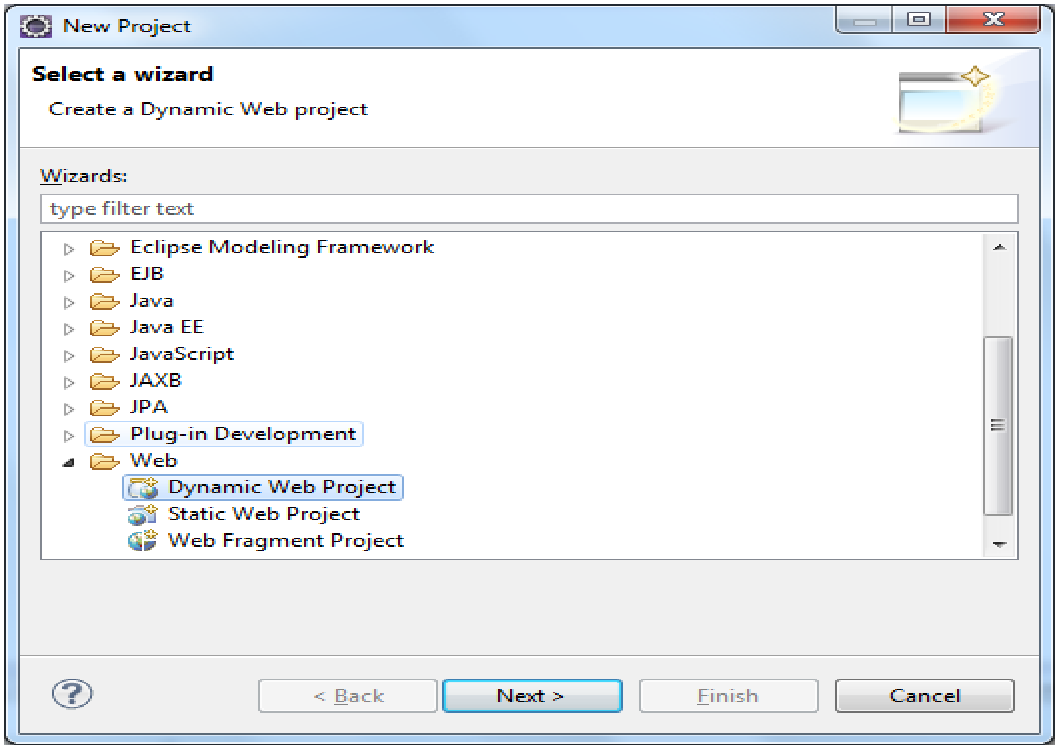


Depending on what you are doing, you use different perspective. For example, when we are developing Java projects, it is most convenient to work in the **Java perspective**, where views related to Java classes and class hierarchy are presented. For debugging, it would be the **Debug perspective** and so on. By default the IDE will use the **Java EE perspective**.

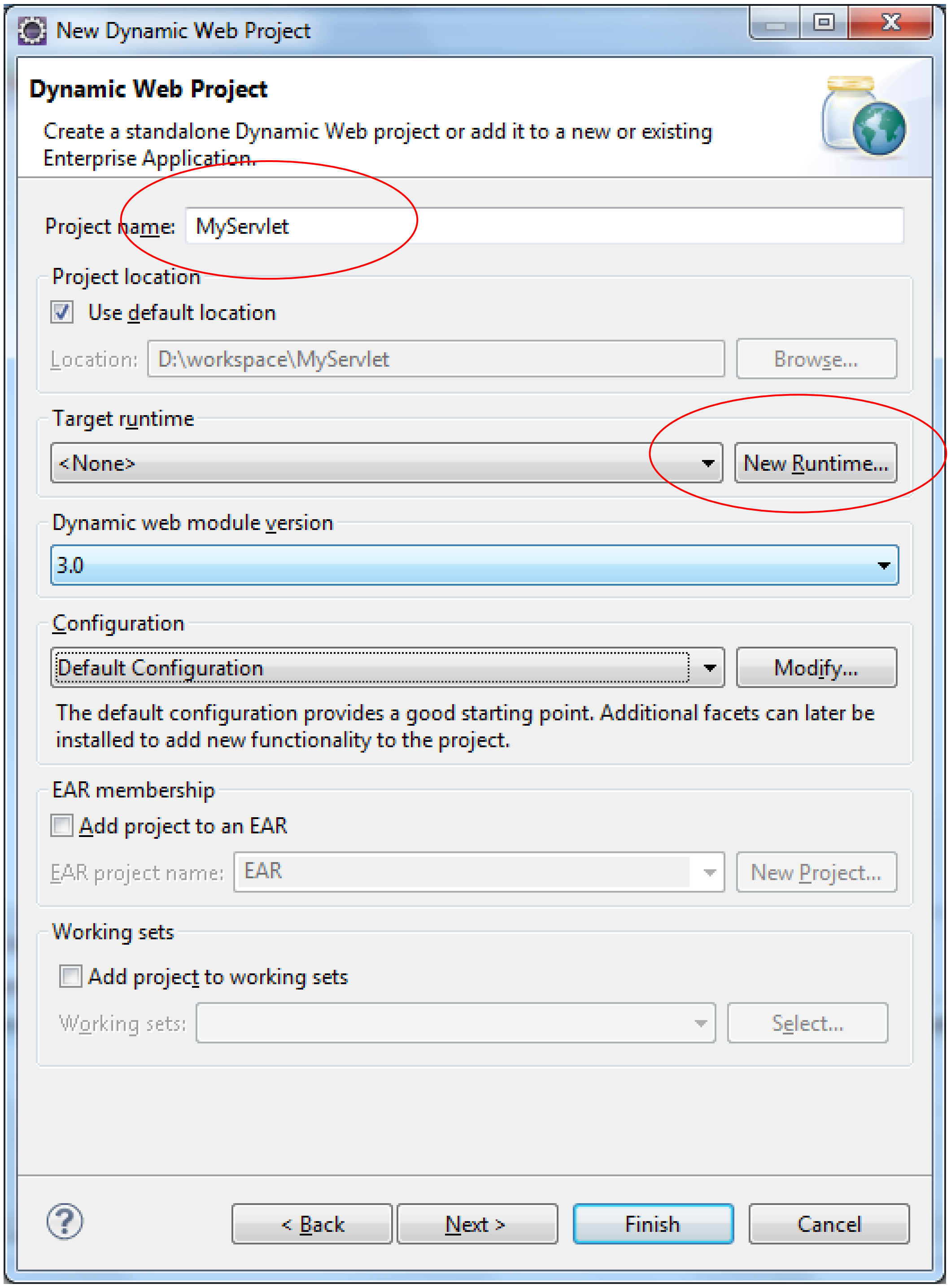
2. Create Web Application Project in Eclipse

- Create a new Web project by selecting "New" → Project. Select "Dynamic Web Project"





- Give a name to the new project e.g "MyServlet".



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: MyServlet

Project location

☑ Use default location

Location: D:\workspace\MyServlet

Target runtime

<None>

Dynamic web module version

3.0

Configuration

Default Configuration

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

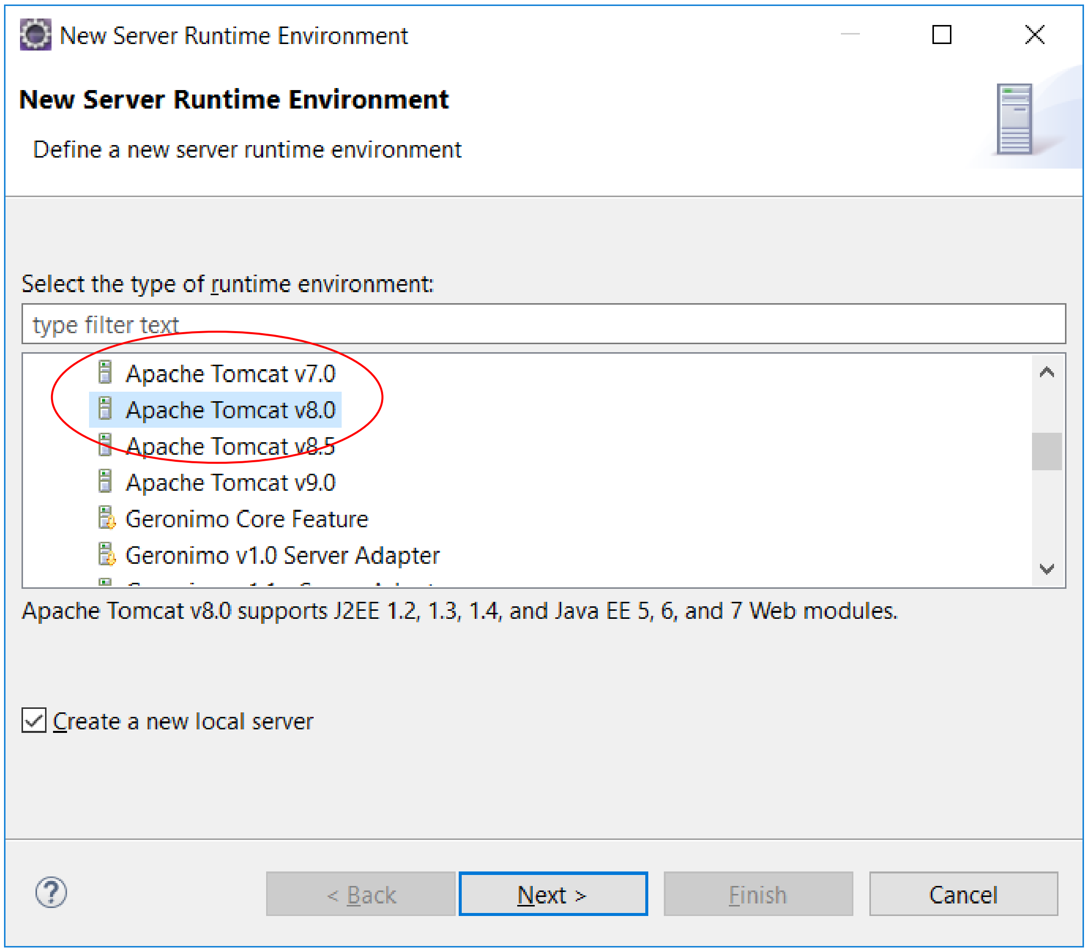
EAR project name: EAR

Working sets

☐ Add project to working sets

Working sets:

- Create a new Runtime (Runtime is required when you test run your web application from within Eclipse environment. You will need to configure a Web Server e.g Tomcat to run your web application.)


New Server Runtime Environment

New Server Runtime Environment

Define a new server runtime environment

Select the type of runtime environment:

type filter text

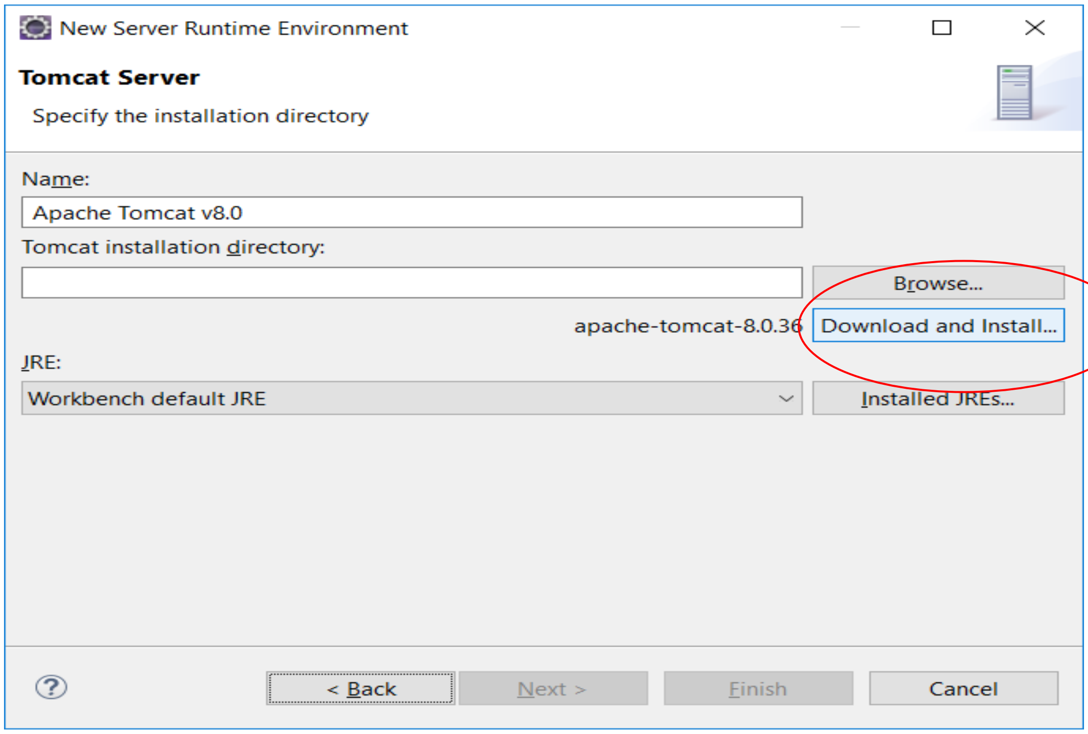
- Apache Tomcat v7.0
- Apache Tomcat v8.0
- Apache Tomcat v8.5
- Apache Tomcat v9.0
- Geronimo Core Feature
- Geronimo v1.0 Server Adapter

Apache Tomcat v8.0 supports J2EE 1.2, 1.3, 1.4, and Java EE 5, 6, and 7 Web modules.

☑ Create a new local server

- Select "Apache Tomcat v8.0" as the runtime. Click "Next".
- You will need to provide the source destination of your Web Container (Apache Tomcat). If a Web Container is not present, download and install one from here (Apache-Tomcat 8.0).

- Choose to download apache-tomcat from the web. Select “Download and Install”



Tomcat Server

Specify the installation directory

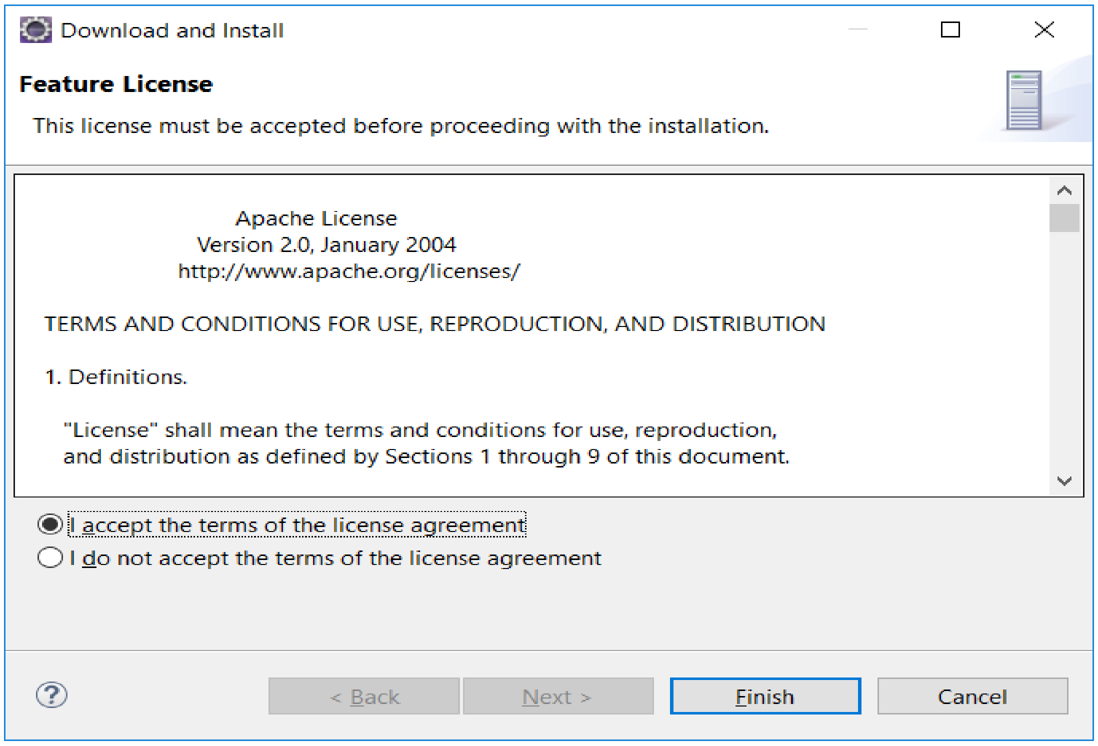
Name: Apache Tomcat v8.0

Tomcat installation directory:

apache-tomcat-8.0.36

JRE: Workbench default JRE

- Accept the license agreement.



Feature License

This license must be accepted before proceeding with the installation.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

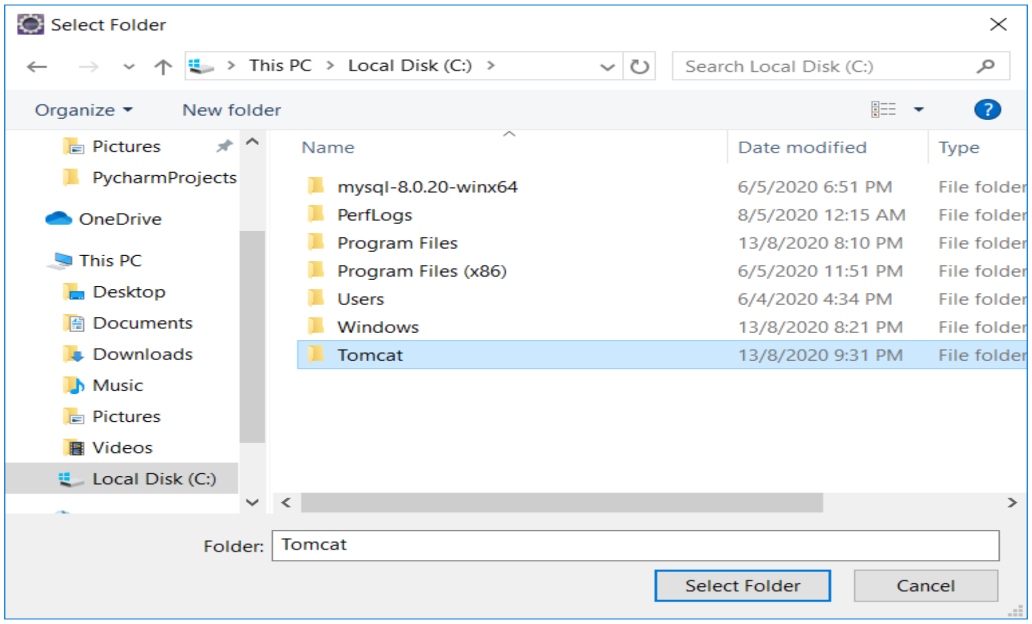
TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

[✓ I accept the terms of the license agreement]
I do not accept the terms of the license agreement

- Create a folder "Tomcat" and select that folder for the download.



Select Folder

This PC > Local Disk (C:) >

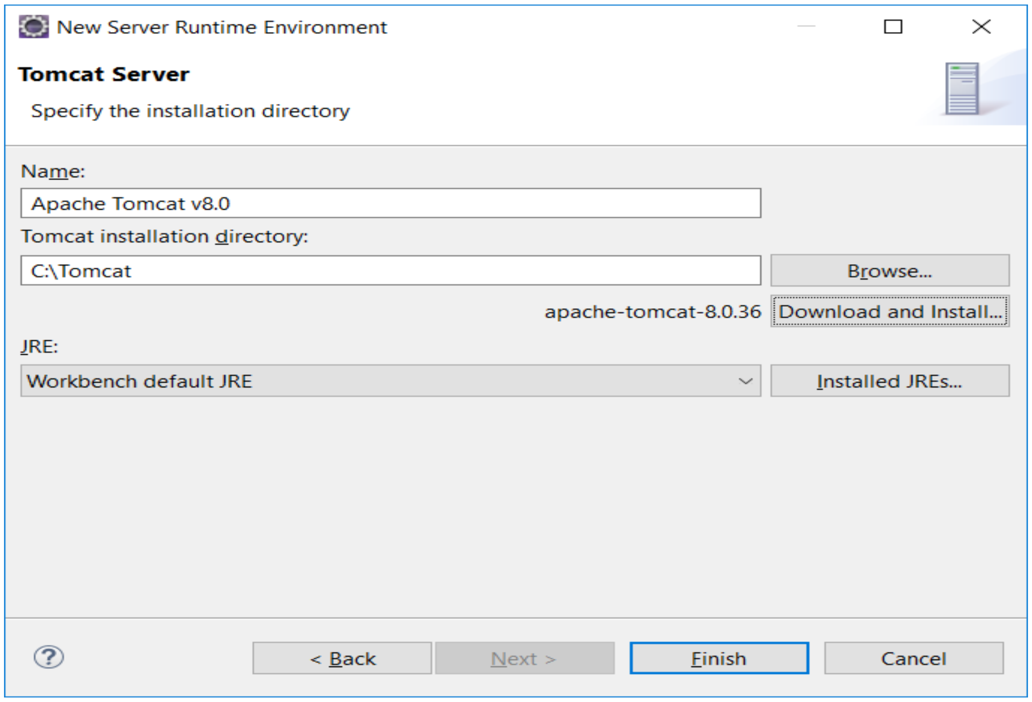
Search Local Disk (C:)

Organize ▾ New folder

Name	Date modified	Type
mysql-8.0.20-winx64	6/5/2020 6:51 PM	File folder
PerfLogs	8/5/2020 12:15 AM	File folder
Program Files	13/8/2020 8:10 PM	File folder
Program Files (x86)	6/5/2020 11:51 PM	File folder
Users	6/4/2020 4:34 PM	File folder
Windows	13/8/2020 8:21 PM	File folder
Tomcat	13/8/2020 9:31 PM	File folder

Folder: Tomcat

Click "Finish" to complete the setup.



New Server Runtime Environment

Tomcat Server

Specify the installation directory

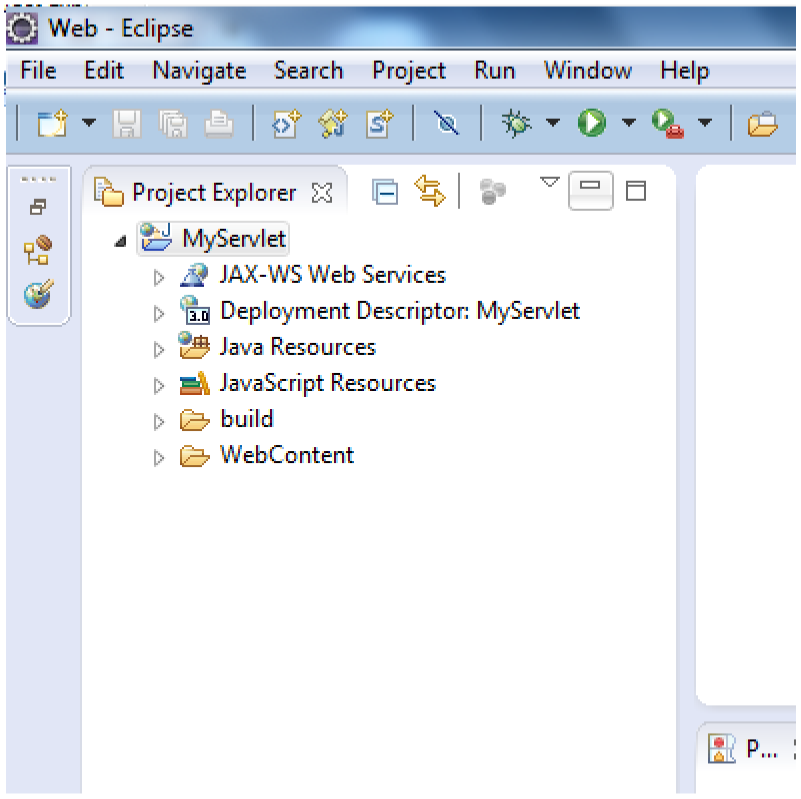
Name: Apache Tomcat v8.0

Tomcat installation directory: C:\Tomcat

apache-tomcat-8.0.36

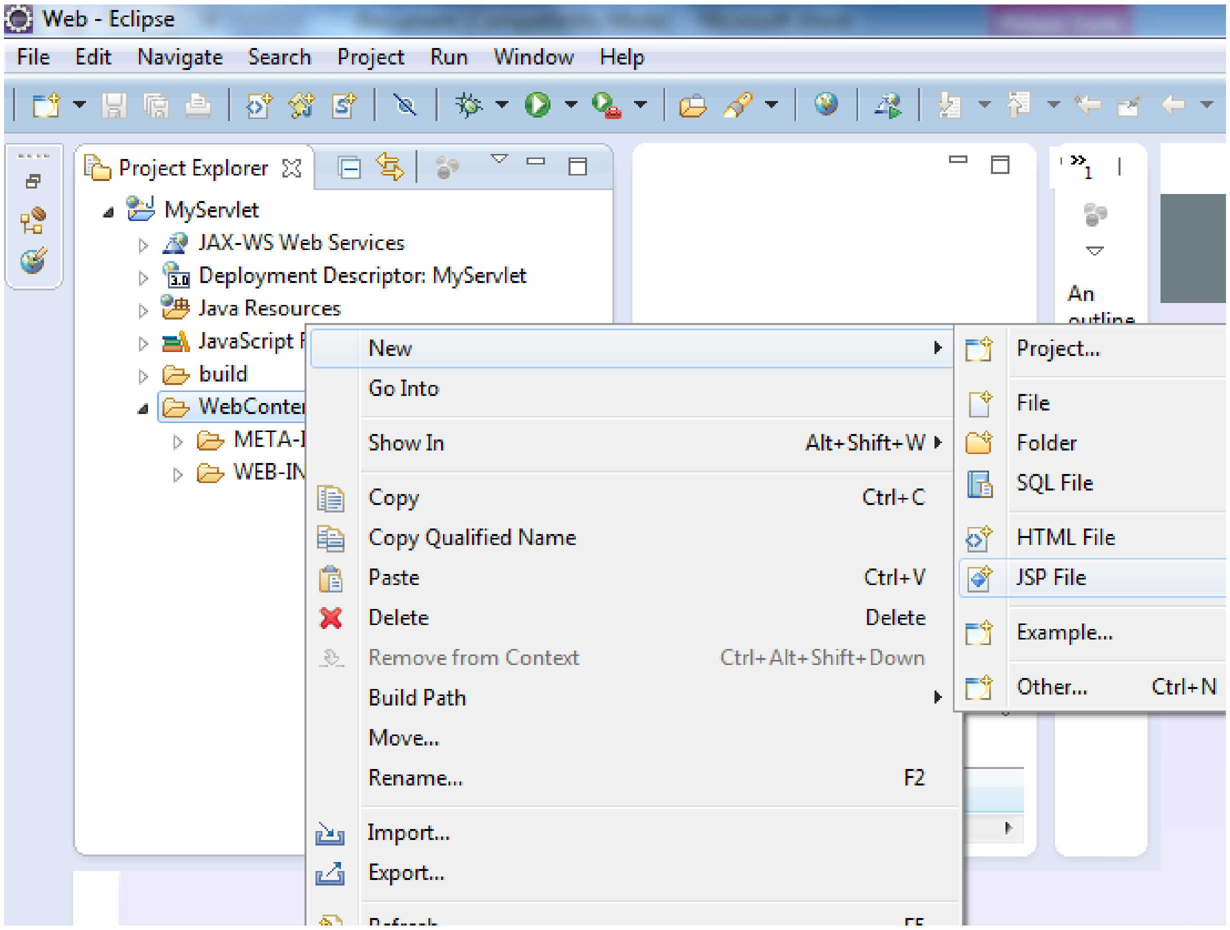
JRE: Workbench default JRE

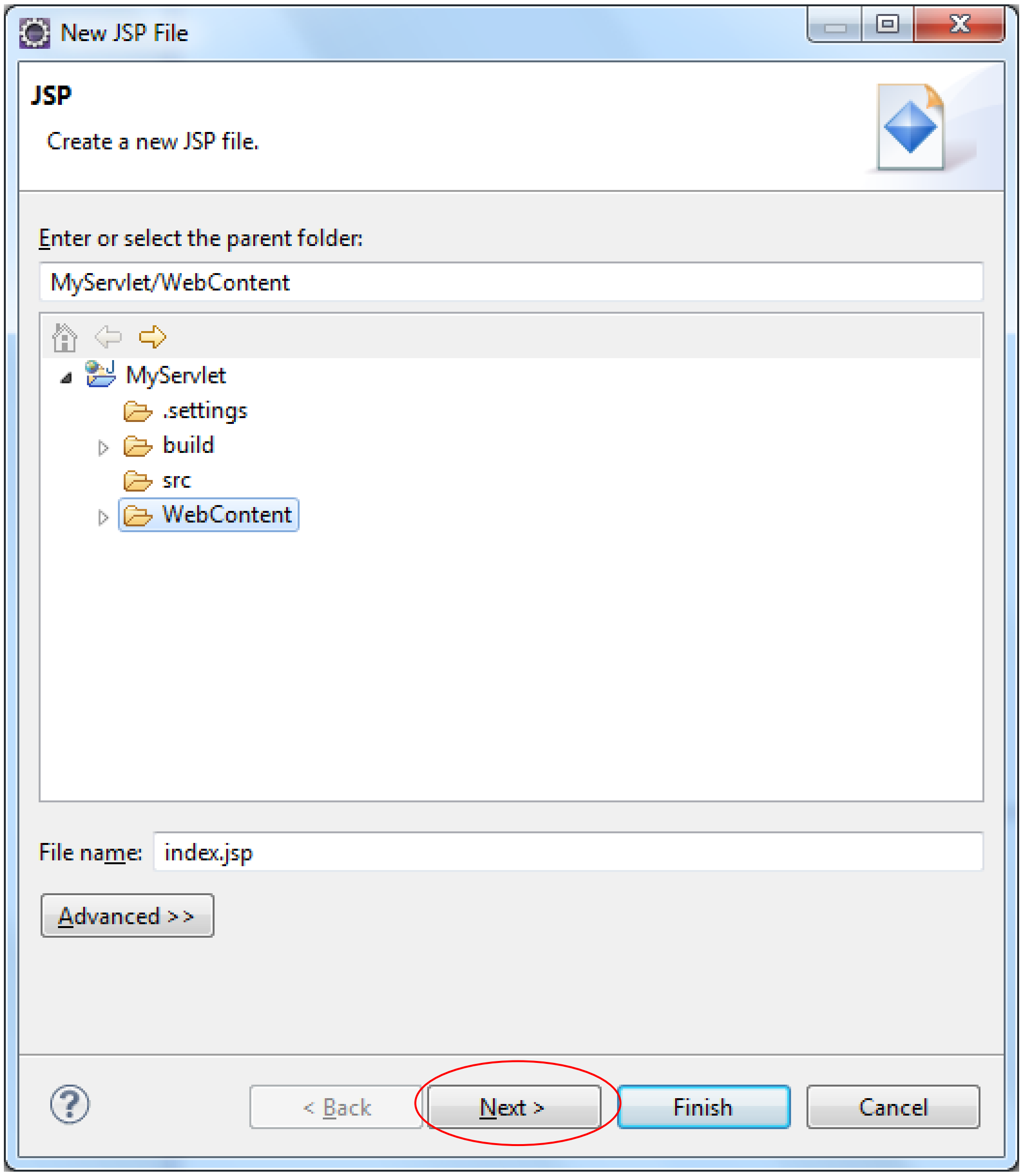
Eclipse will create a file structure in the Project Explorer similar to the one shown below.



3. Creating a JSP (Client-Side) page for sending HTTP Request to Server

- Create a JSP page and name the file “index.jsp”. Right-click “WebContent” and select “New” → JSP File





New JSP File

JSP

Create a new JSP file.

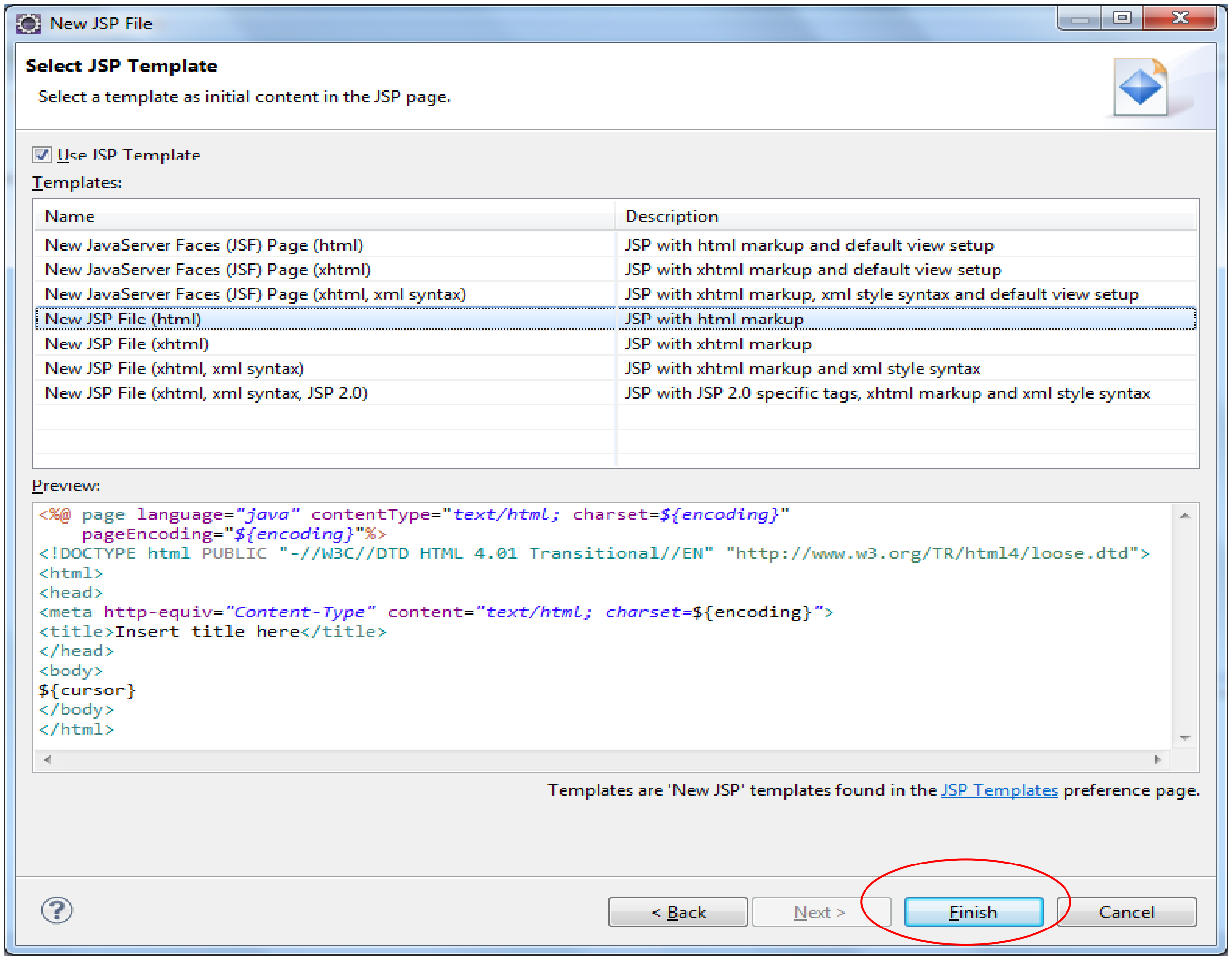
Enter or select the parent folder:

MyServlet/WebContent

File name: index.jsp

- Choose “WebContent” as the folder to save your index.jsp. D. Click Next.

- Accept the default template from Eclipse. Press Finish button and you should have index.jsp under your WebContent folder.



Select JSP Template

Select a template as initial content in the JSP page.

[✓] Use JSP Template

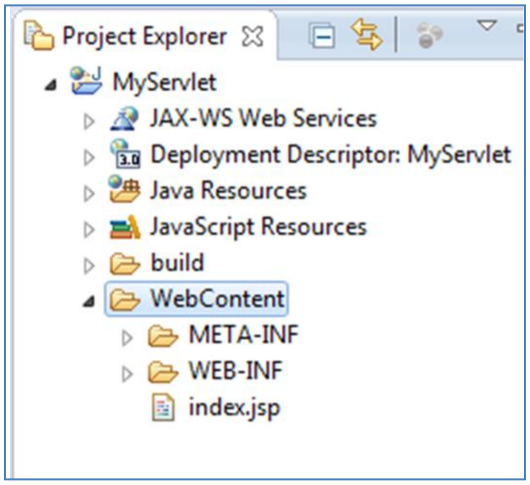
Templates:

Name	Description
New JavaServer Faces (JSF) Page (html)	JSP with html markup and default view setup
New JavaServer Faces (JSF) Page (xhtml)	JSP with xhtml markup and default view setup
New JavaServer Faces (JSF) Page (xhtml, xml syntax)	JSP with xhtml markup, xml style syntax and default view setup
New JSP File (html)	JSP with html markup
New JSP File (xhtml)	JSP with xhtml markup
New JSP File (xhtml, xml syntax)	JSP with xhtml markup and xml style syntax
New JSP File (xhtml, xml syntax, JSP 2.0)	JSP with JSP 2.0 specific tags, xhtml markup and xml style syntax

Preview:

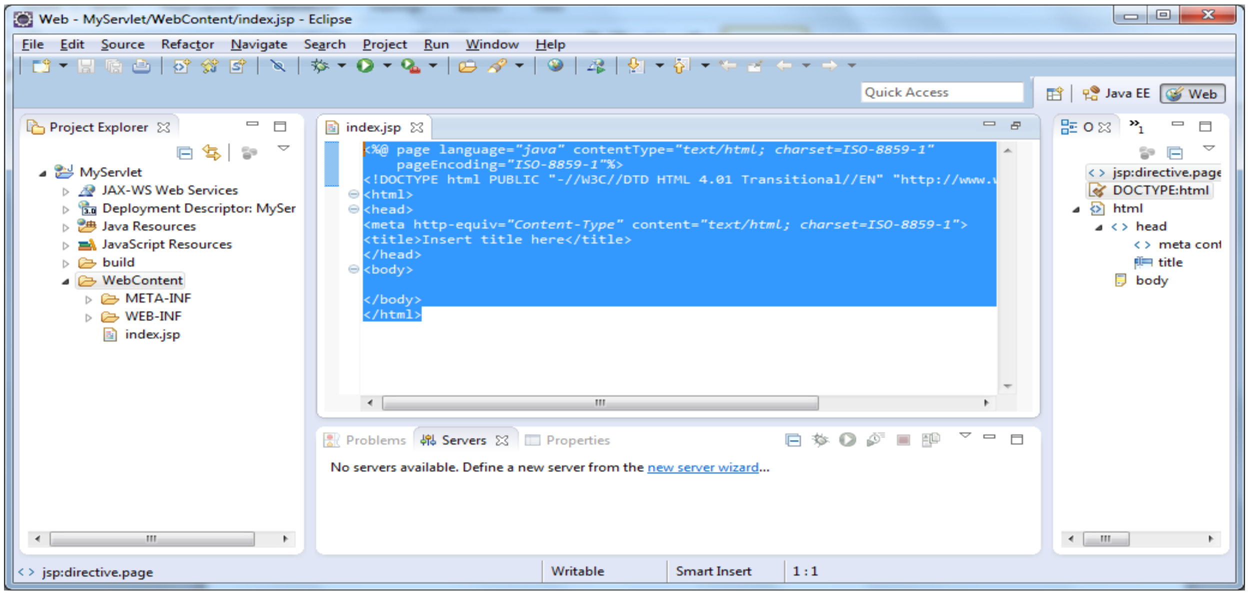
```
<%@ page language="java" contentType="text/html; charset=${encoding}"
    pageEncoding="${encoding}"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=${encoding}">
<title>Insert title here</title>
</head>
<body>
${cursor}
</body>
</html>
```

Templates are 'New JSP' templates found in the JSP Templates preference page.



- Double-click on “index.jsp” file and it should bring you to the code editor.

- We need to include a HTML Form in between the “<body> …</body>” section.

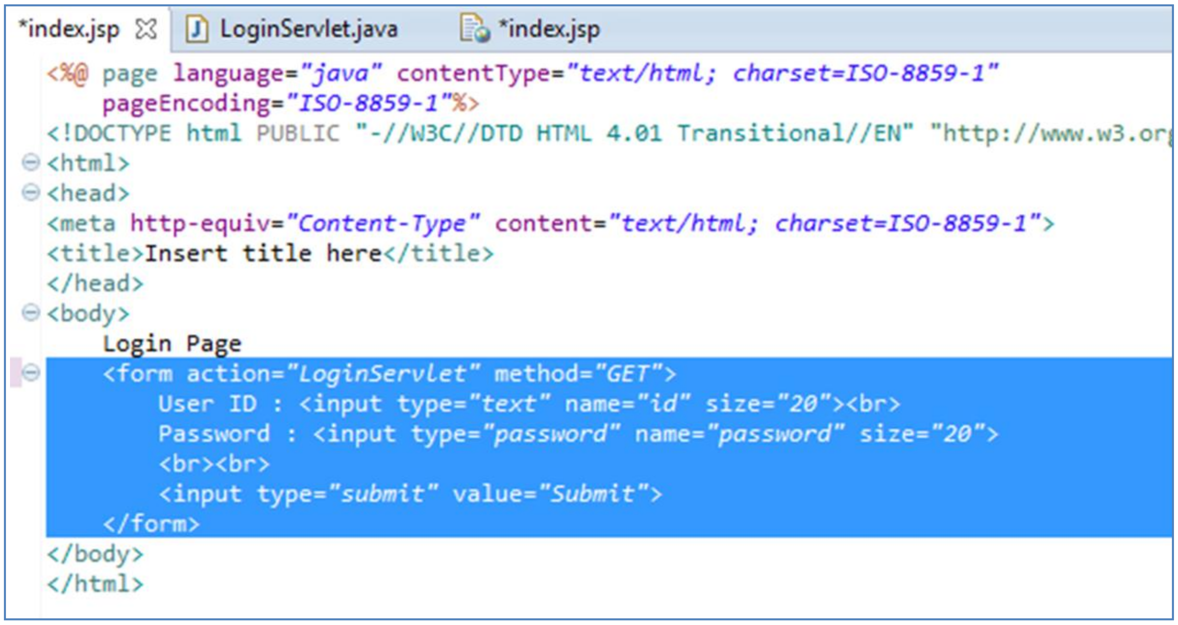


```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>

</body>
</html>
```

No servers available. Define a new server from the new server wizard...

- Add the following HTML codes : 2 textboxes

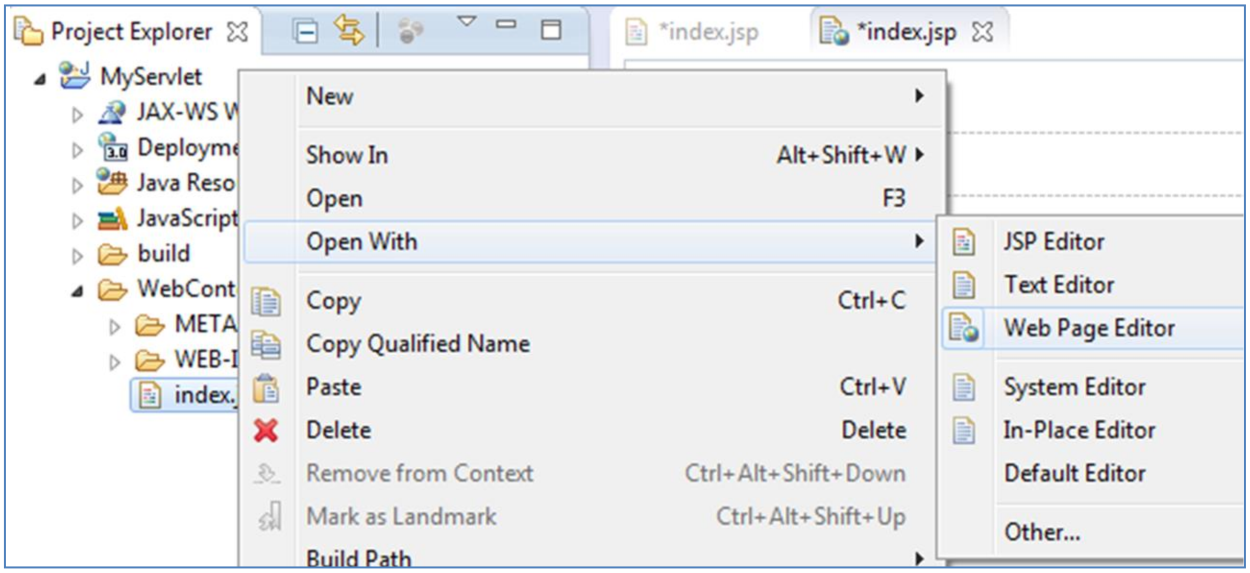
```
    <form action="LoginServlet" method="GET">
        User ID : <input type="text" name="id" size="20"><br>
        Password : <input type="password" name="password" size="20">
        <br><br>
        <input type="submit" value="Submit">
    </form>
```



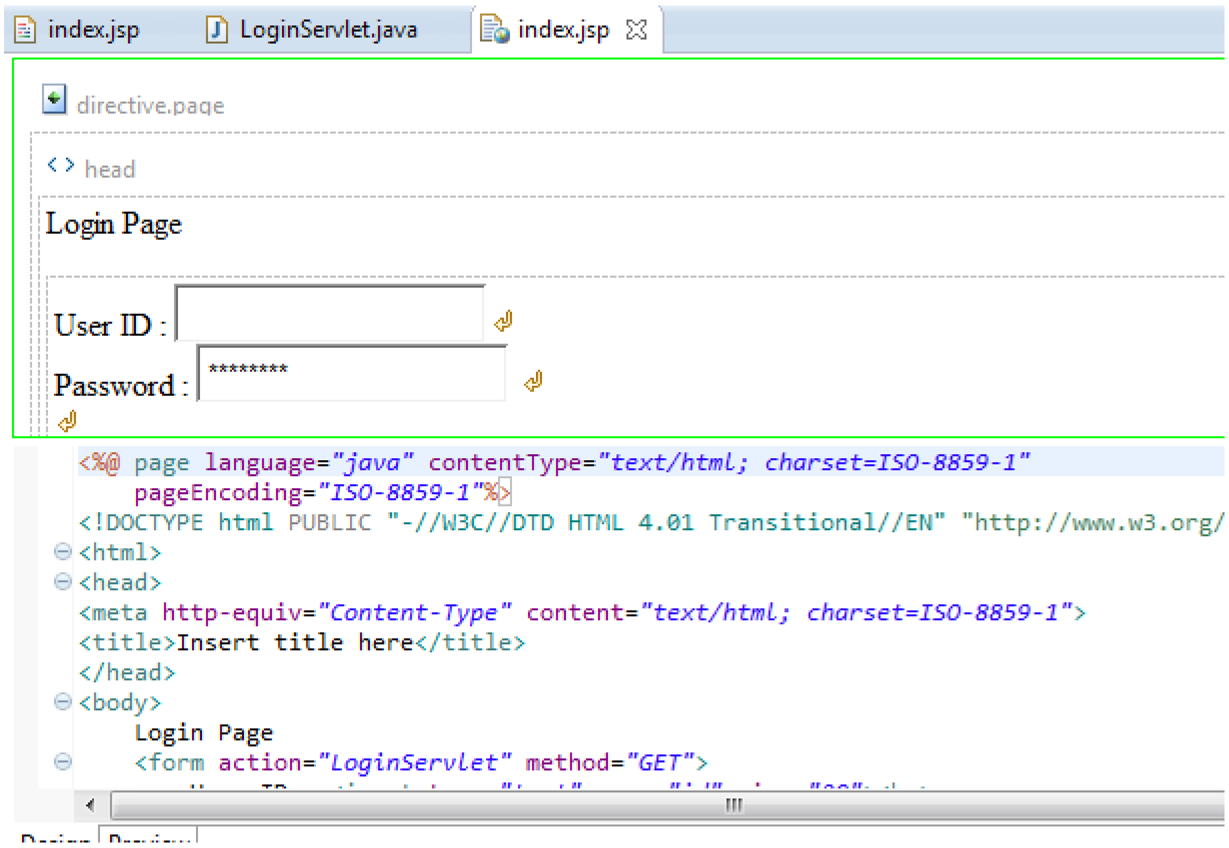
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.or
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    Login Page
    <form action="LoginServlet" method="GET">
        User ID : <input type="text" name="id" size="20"><br>
        Password : <input type="password" name="password" size="20">
        <br><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

- Save your work.

- Preview your JSP file. Right-click on index.jsp select “Open With” → “Web Page Editor”.



- Eclipse allow users to preview the output page of HTML and JSP files.



Login Page

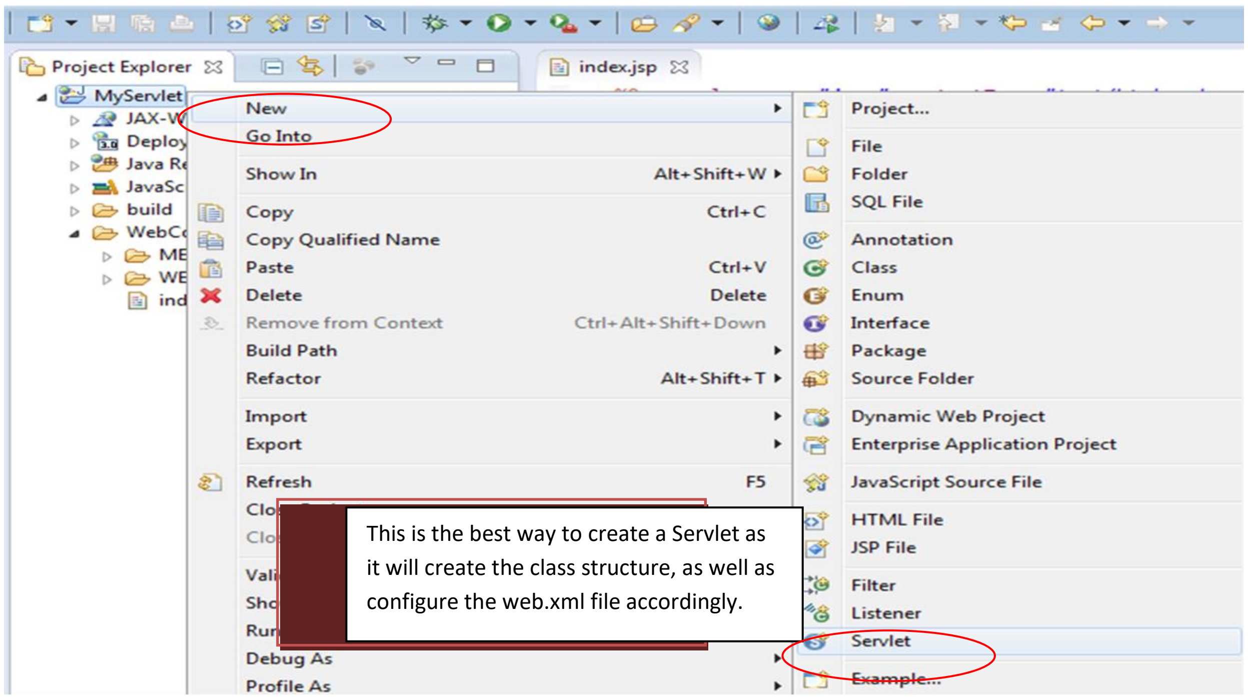
User ID :

Password : ********

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    Login Page
    <form action="LoginServlet" method="GET">
```

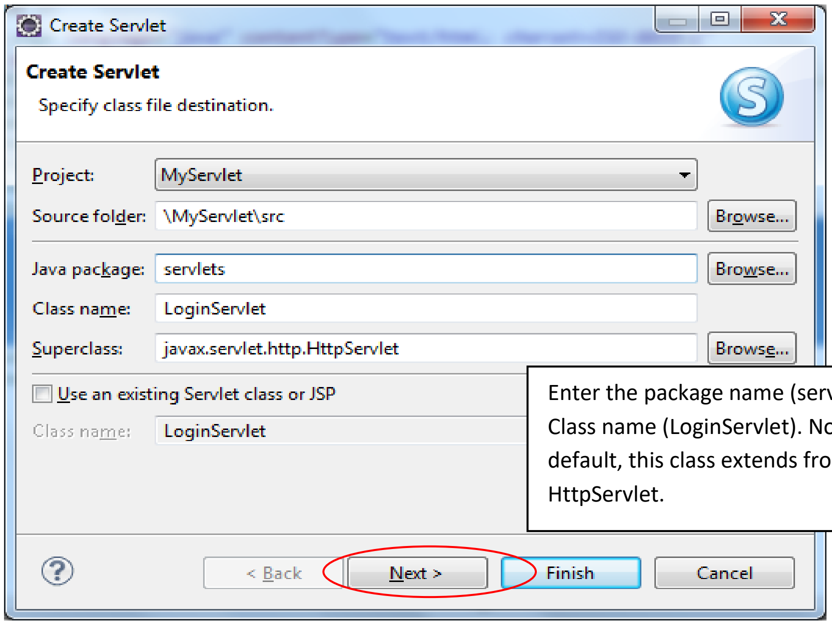
4. Creating a Servlet (LoginServlet) to service the HTTP Request

- Right-click project name ("MyServlet") and select "New" → Servlet.



This is the best way to create a Servlet as it will create the class structure, as well as configure the web.xml file accordingly.

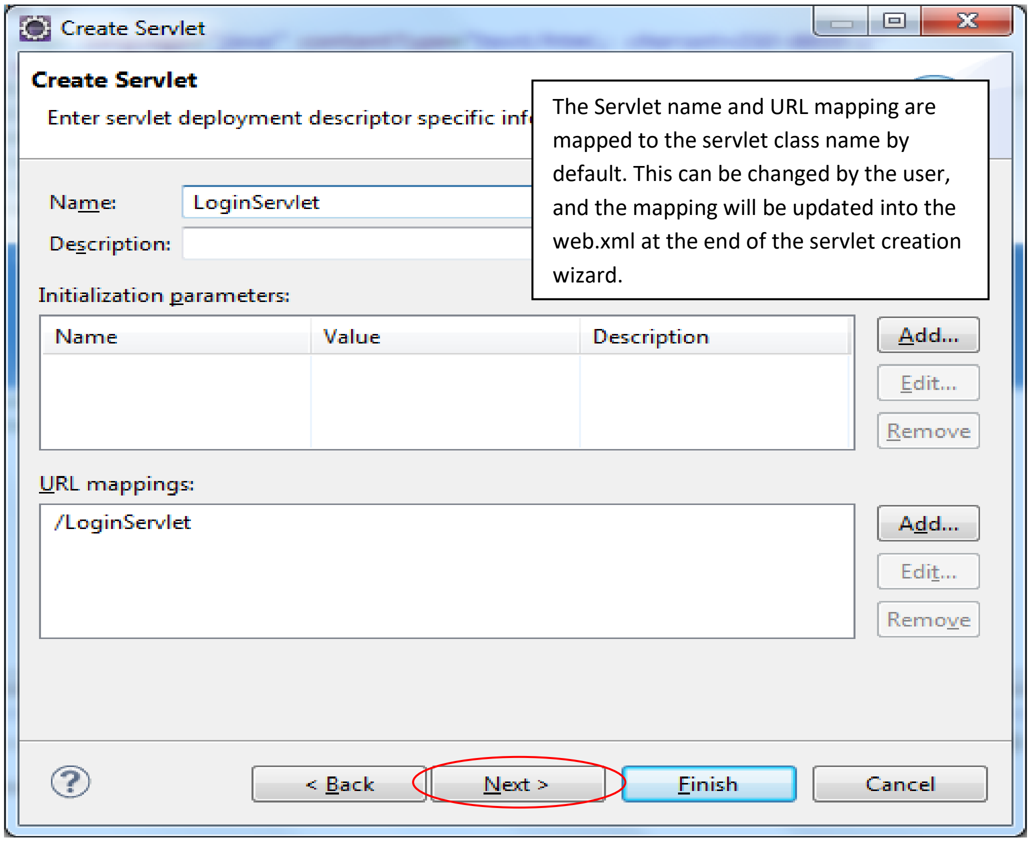
- Set Class name as "LoginServlet" and Java Package as "servlets". Notice that the
- servlet extends ***javax.servlet.http.HttpServlet.***



Project: MyServlet
Source folder: \MyServlet\src
Java package: servlets
Class name: LoginServlet
Superclass: javax.servlet.http.HttpServlet
Use an existing Servlet class or JSP
Class name: LoginServlet

Enter the package name (servlets) and Class name (LoginServlet). Notice that by default, this class extends from HttpServlet.

- By default, the Servlet Name is mapped to the Servlet class name, and the URL mapping is mapped to the Servlet name.   The name and mapping can be changed accordingly, but we will stick with the default values.
- Click Next.
- Select Next. Note /LoginServlet is being set as the URL mappings.The URL mapping is used to identify on how the Java Servlet should be called. This is the URL that is called in your <form></form> tag in JSP file. If you set the URL Mapping into /go/LoginServlet, you need to call it as /go/GreetingServlet from your <form></form> tag in your JSP file.



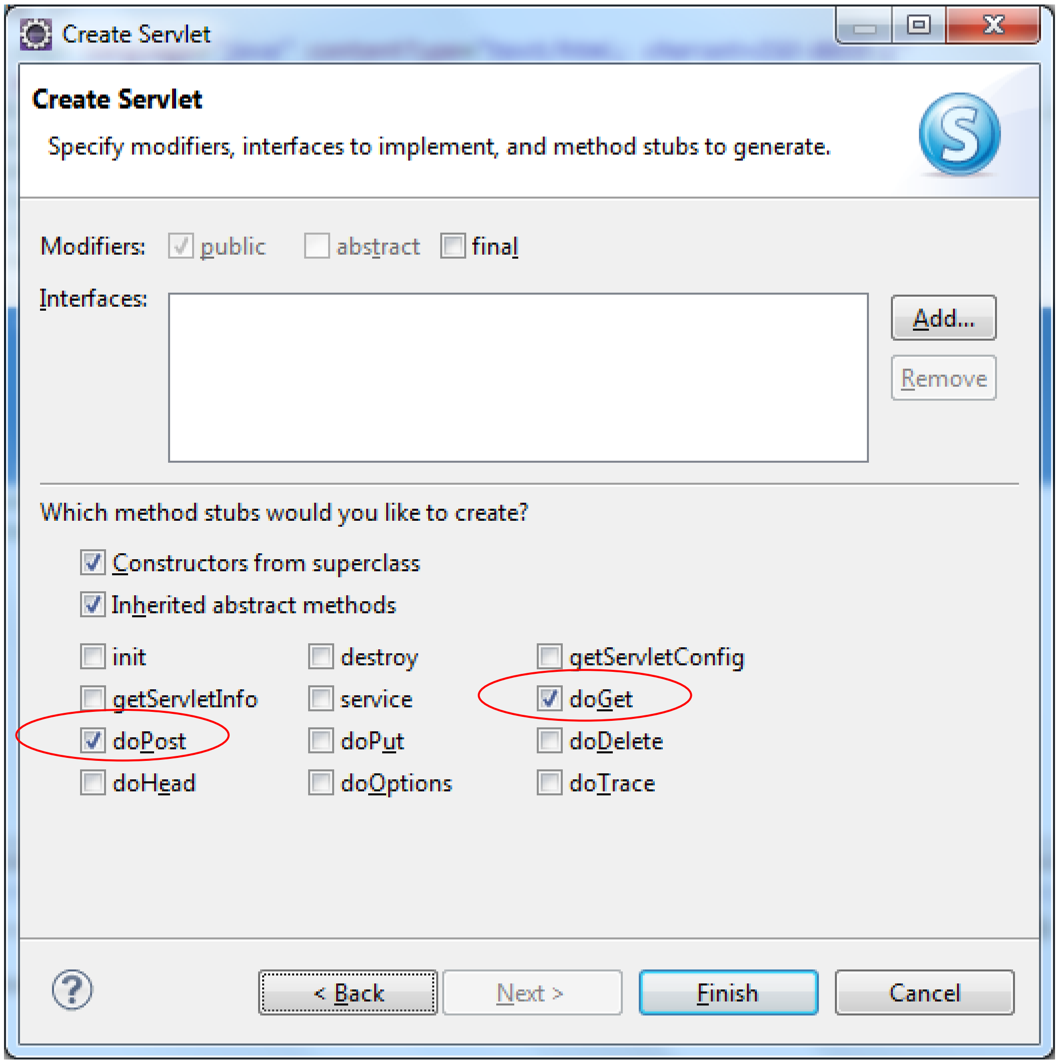
Name: LoginServlet

Description:

Initialization parameters:

URL mappings: /LoginServlet

The Servlet name and URL mapping are mapped to the servlet class name by default. This can be changed by the user, and the mapping will be updated into the web.xml at the end of the servlet creation wizard.



Create Servlet

Create Servlet

Specify modifiers, interfaces to implement, and method stubs to generate.

Modifiers: [✓ public] abstract final

Interfaces:

Which method stubs would you like to create?

[✓ Constructors from superclass]

[✓ Inherited abstract methods]

init destroy getServletConfig

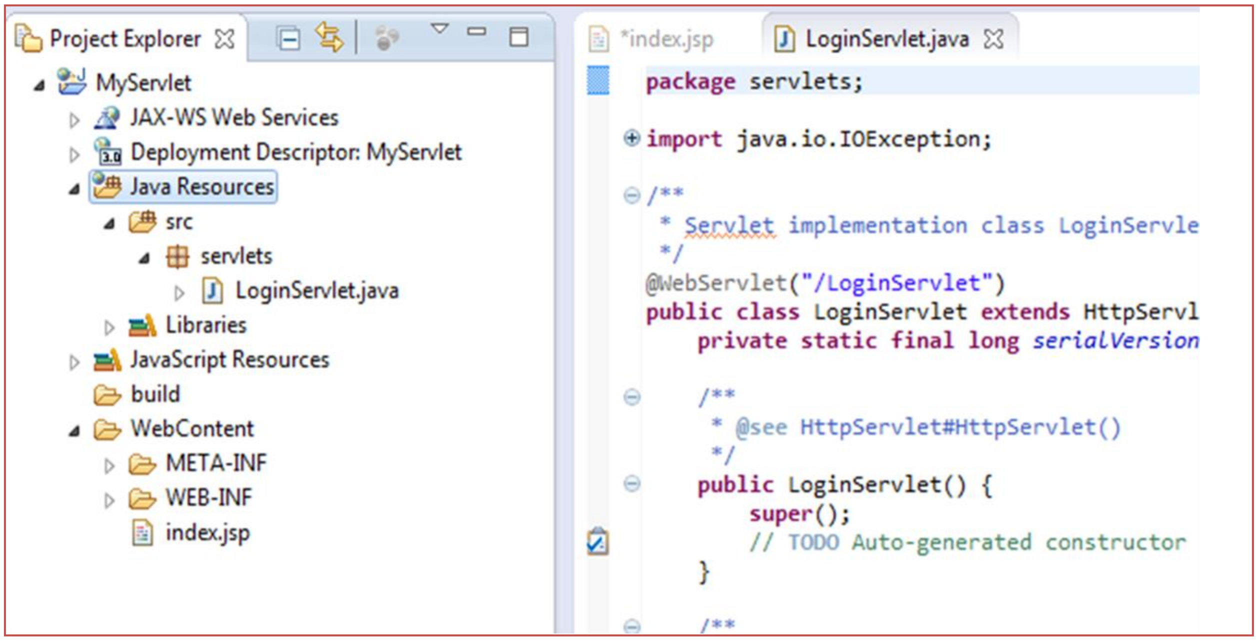
getServletInfo service [✓ doGet]

[✓ doPost] doPut doDelete

doHead doOptions doTrace

- In this screen, you may choose the methods you wish to override. By default, doGet and doPost will always be selected. In general we can use the default value.
- Click finish to complete the servlet creation wizard.

- LoginServlet.java is being created in the file structure as shown in the file explorer.
- It is created in the “servlets” java package.



```
package servlets;

import java.io.IOException;

/**
 * Servlet implementation class LoginServle
 */
@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServl
    private static final long serialVersion

    /**
     * @see HttpServlet#HttpServlet()
     */
    public LoginServlet() {
        super();
        // TODO Auto-generated constructor
    }

    /**
```

- Double-click on LoginServlet.java.
- The LoginServlet code editor, overrides the doGet() method and verify that the userid is the same as the password. If the userid and password is the same, servlet should returns a HTML stream with a “Hello World” otherwise display a “Login failure” in the System.out.

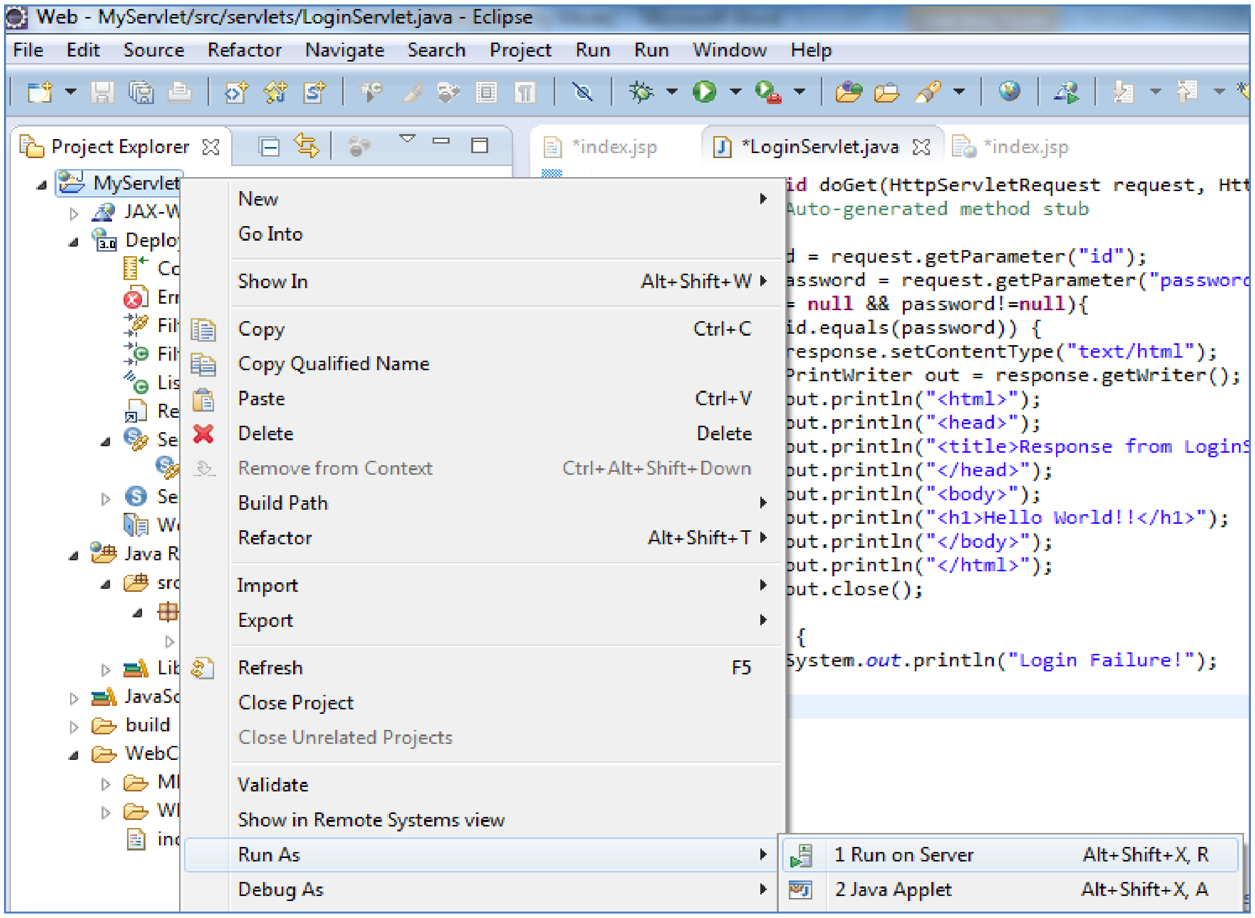
```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
            // TODO Auto-generated method stub

      String id = request.getParameter("id");
      String password = request.getParameter("password");

      if (id != "" && password != "" ){

            if (id.equals(password)){

                  response.setContentType("text/html");
                  PrintWriter out = response.getWriter();

                  out.println ("<html>");
                  out.println ("<head>");
                  out.println ("<title>Response from LoginServlet</title>");
                  out.println ("</head>");
                  out.println ("<body>");
                  out.println ("<h1>Hello World!</h1>");
                  out.println ("</body>");
                  out.println ("</html>");
                  out.close();
            }
            else { System.out.println("login failure");

            }

      }

}
```

- Import java.io.* ; - required for PrintWriter.

5. Test running your Web Application project

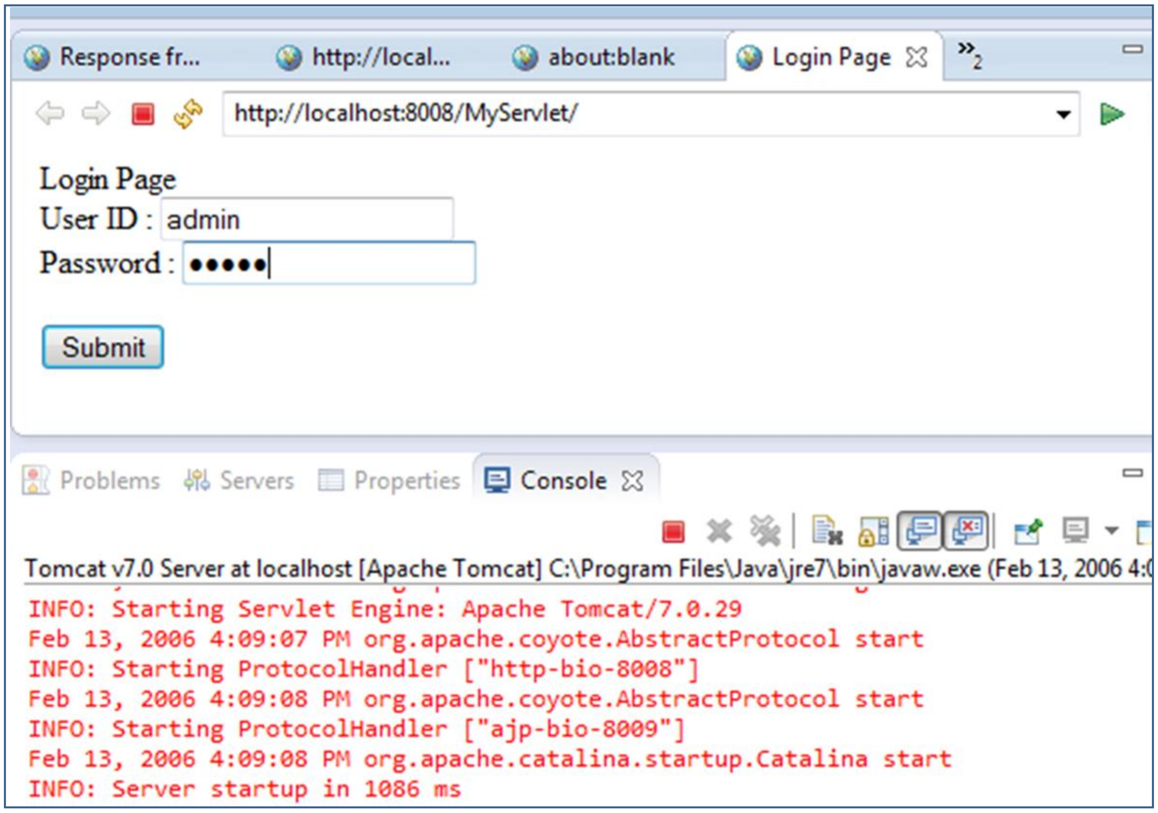
There are 2 ways to test run the project

- Get Apache-Tomcat server to run within Eclipse and run the test inside Eclipse.
- Eclipse will start Apache-tomcat. Preview with internal browser.
- Right-click project name → choose “Run As” → “Run on Server”



```
id doGet(HttpServletRequest request, Htt
Auto-generated method stub

d = request.getParameter("id");
assword = request.getParameter("passwor
= null && password!=null){
id.equals(password)) {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head>");
out.println("<title>Response from LoginS
out.println("</head>");
out.println("<body>");
out.println("<h1>Hello World!!</h1>");
out.println("</body>");
out.println("</html>");
out.close();

 {
System.out.println("Login Failure!");
```

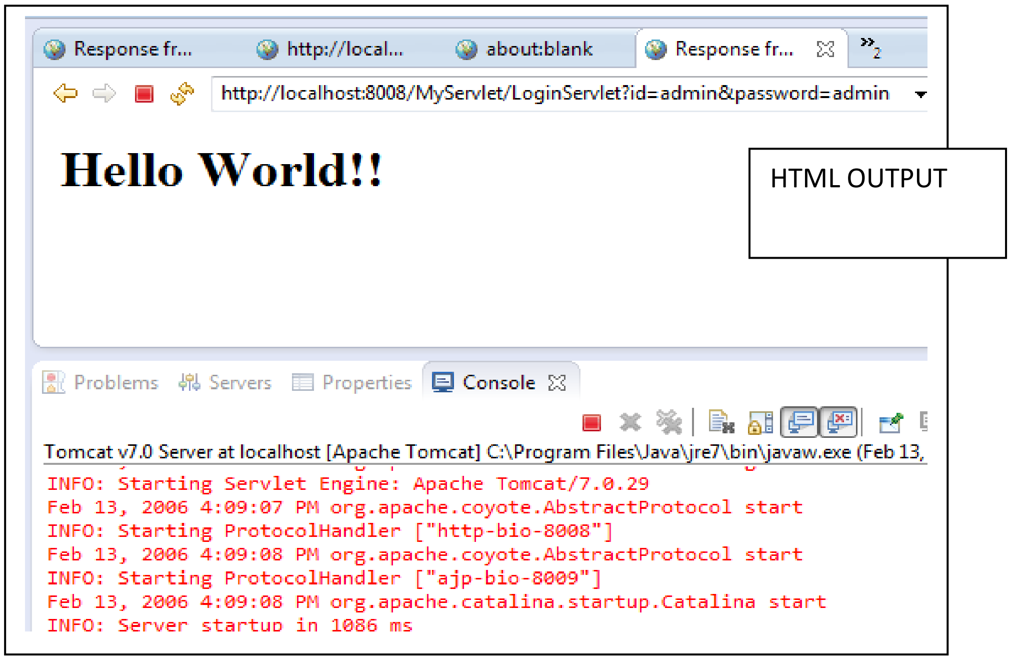
- Eclipse will attempt to start Tomcat server Instance. The cue is “Server startup in … ms”.
- Notice the URL **http://localhost:8008/MySerlvet/** is use to access the web application.



```
Tomcat v7.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre7\bin\javaw.exe (Feb 13, 2006 4:
INFO: Starting Servlet Engine: Apache Tomcat/7.0.29
Feb 13, 2006 4:09:07 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-bio-8008"]
Feb 13, 2006 4:09:08 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-bio-8009"]
Feb 13, 2006 4:09:08 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 1086 ms
```

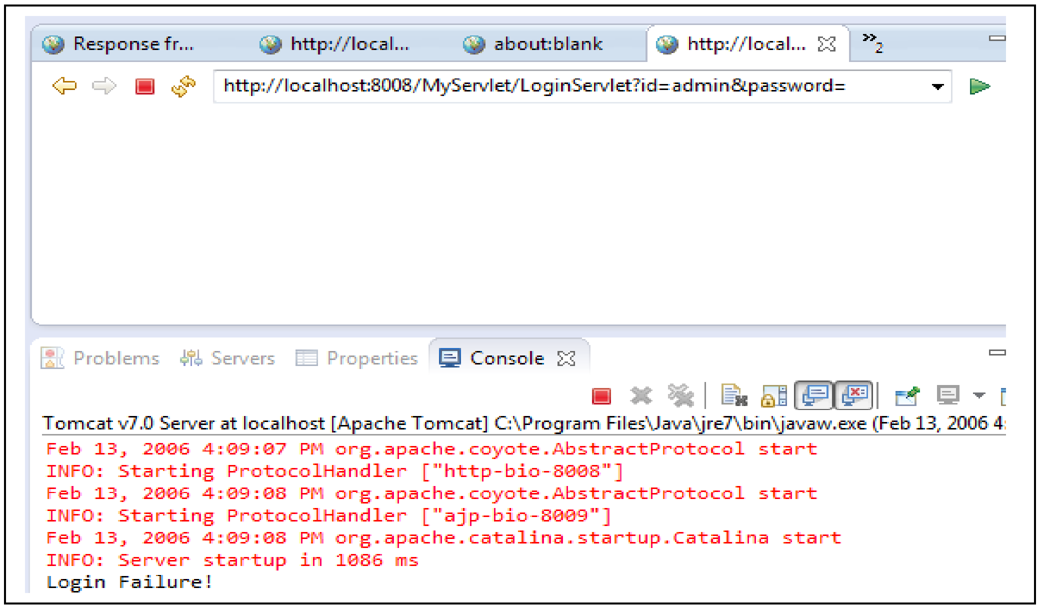
- F. Type similar value for userid and password in the textboxes. Click Submit.

- G. The server should return a screen similar to the one shown below.



```
Tomcat v7.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre7\bin\javaw.exe (Feb 13,
INFO: Starting Servlet Engine: Apache Tomcat/7.0.29
Feb 13, 2006 4:09:07 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-bio-8008"]
Feb 13, 2006 4:09:08 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-bio-8009"]
Feb 13, 2006 4:09:08 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 1086 ms
```

- H. If the userid and password value is different, system should display a message in
- the System.out screen.



```
Tomcat v7.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre7\bin\javaw.exe (Feb 13, 2006 4
Feb 13, 2006 4:09:07 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-bio-8008"]
Feb 13, 2006 4:09:08 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-bio-8009"]
Feb 13, 2006 4:09:08 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 1086 ms
Login Failure!
```

End