

MIS 768: Advanced Software Concepts

Spring 2024

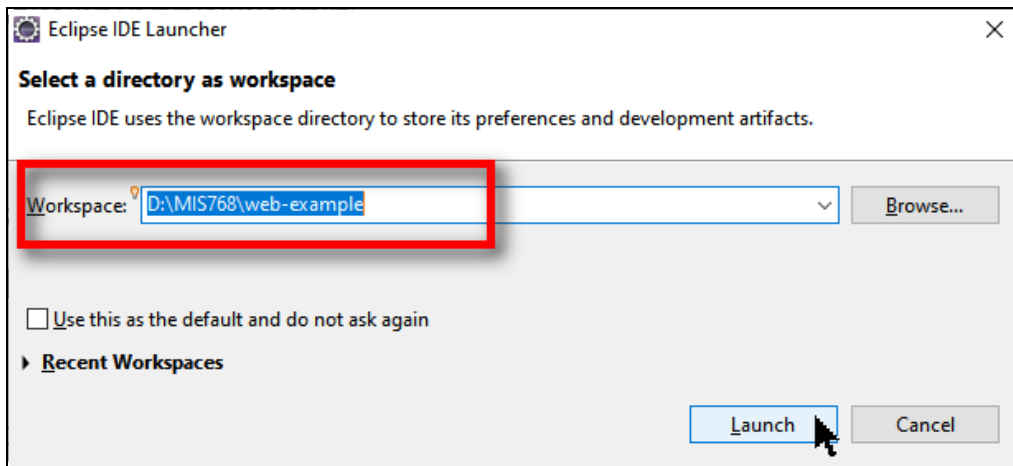
Web Application (1)

Purpose

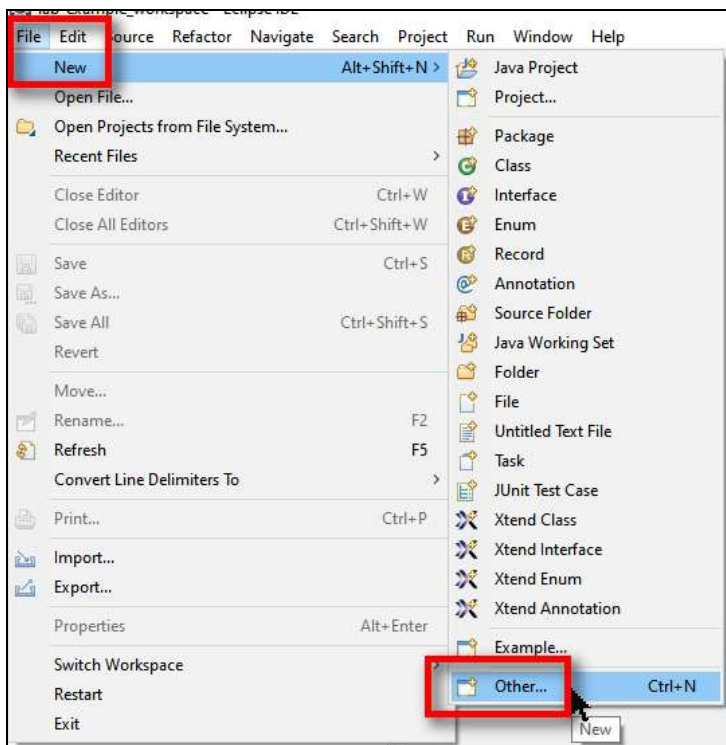
- Set up projects for Web Applications
- Create the first servlet and JSP

1. Preparation: Create a Web Application Project

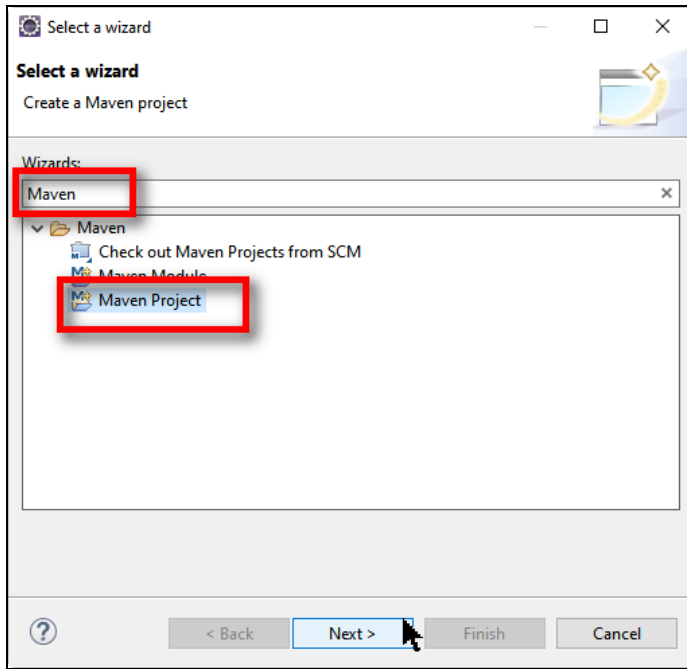
- (1) When you launch Eclipse, please **set up a new workspace**.



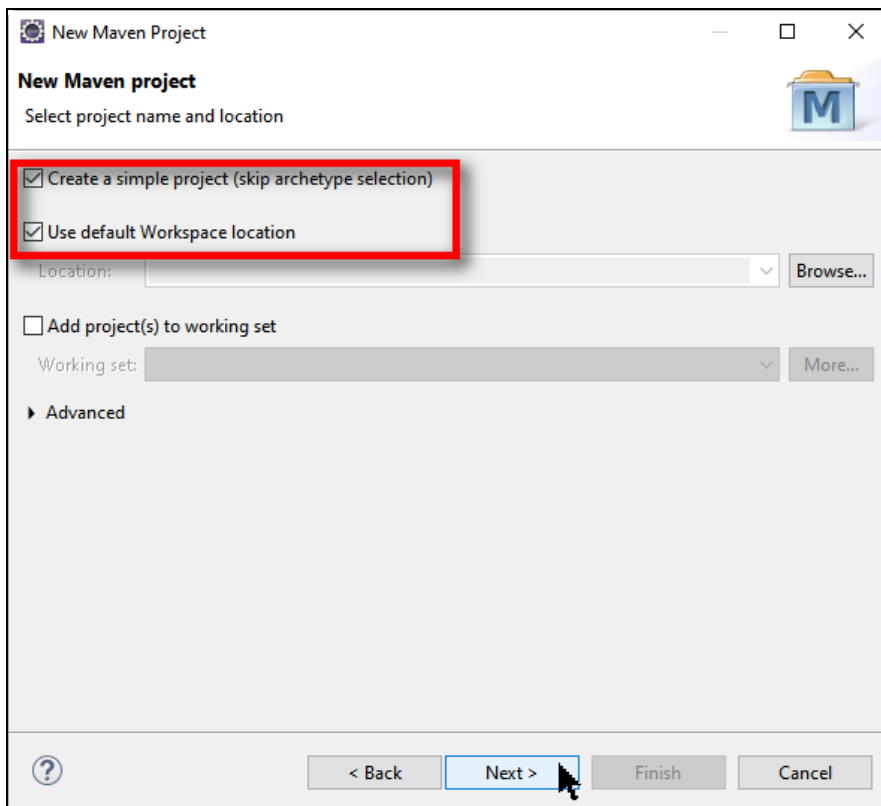
- (2) Go to the menu bar and select **File \ New \ Other ...**



- (3) In the Wizard selection widow, search and select **Maven Project**

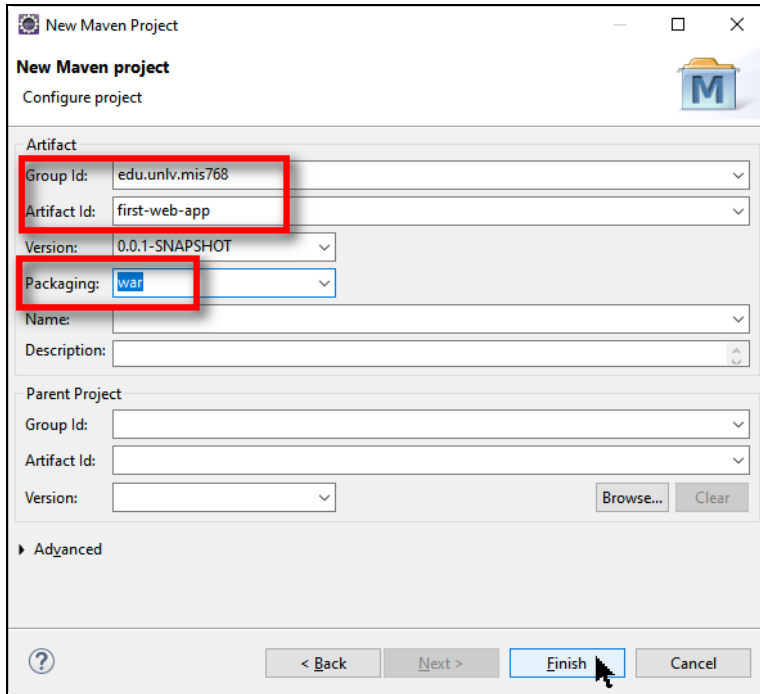


- (4) In the New Maven Project Wizard, check the option of “**Create a simple project**” and “**Use default Workspace location**”



- (5) In the next screen, enter the **group Id** and the **Artifact Id**.

Select the **Packaging** as **war**.



New Maven Project

Configure project

Artifact

Group Id: edu.unlv.mis768

Artifact Id: first-web-app

Version: 0.0.1-SNAPSHOT

Packaging: war

Name:

Description:

Parent Project

Group Id:

Artifact Id:

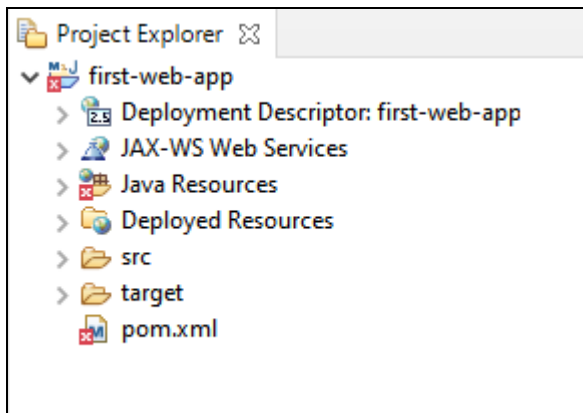
Version:

Browse... Clear

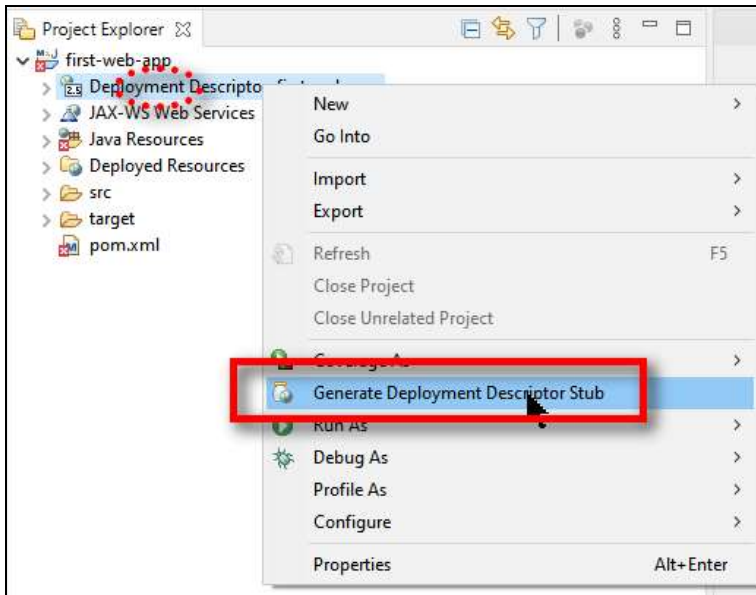
Advanced

< Back Next > Finish Cancel

- (6) A Maven project will be created with a few folders. You will see an error at pom.xml.

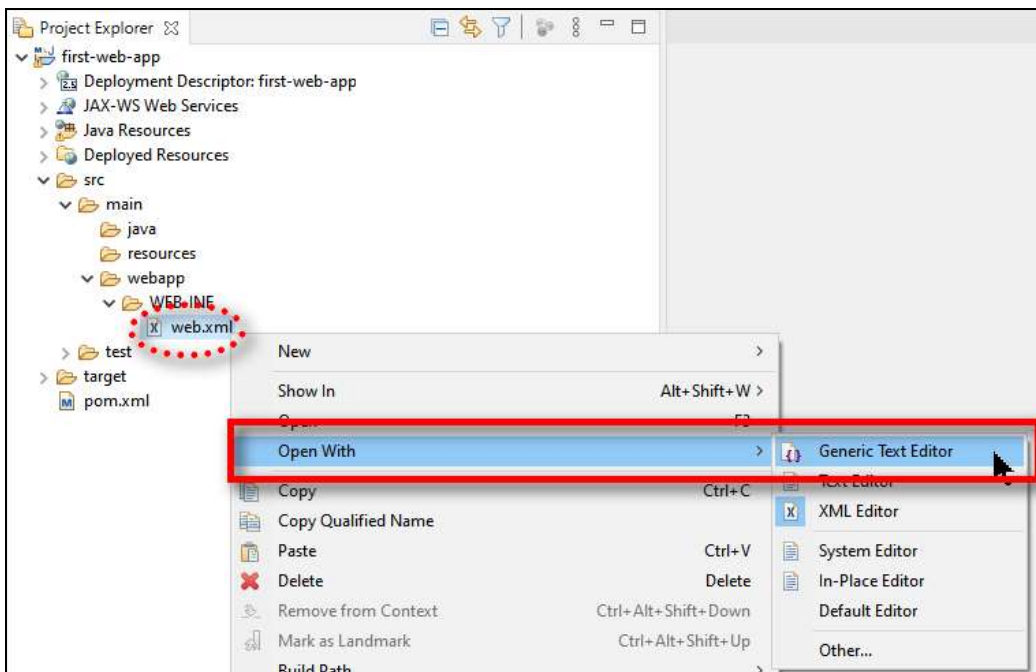


- (7) Right click on **Deployment Descriptor**, then select **Generate Deployment Descriptor Stub**.

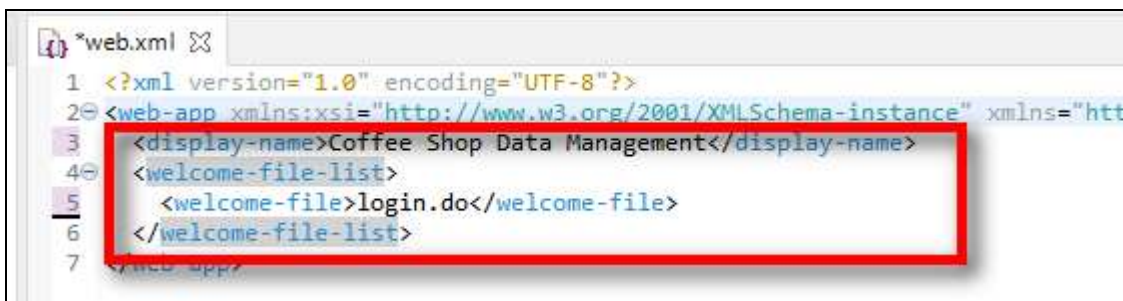


- (8) It will generate the **WEB-INF** folder in **src/main/webapp** and a **web.xml** in it.

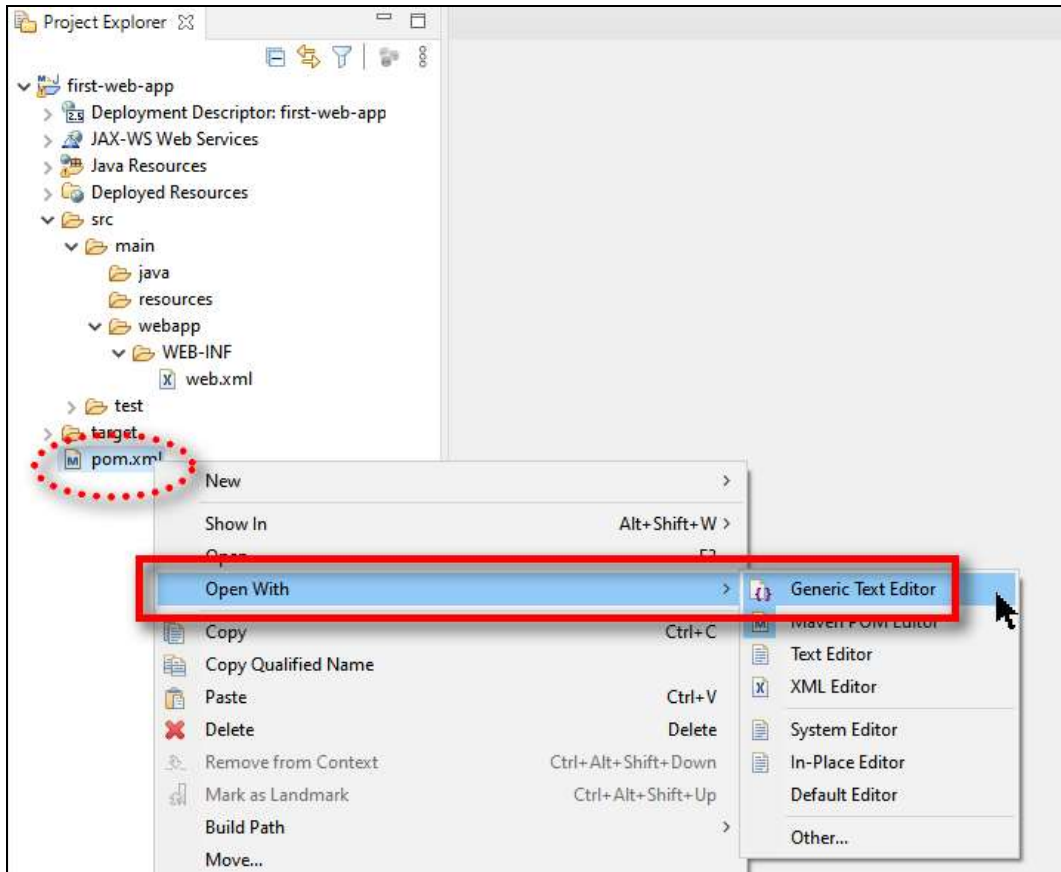
Please open **web.xml** with **Generic Text Editor**



- (9) Revise **web.xml** as following to set the display name and the entry page of the web application. Save and close the file.



(10) Please open pom.xml with **Generic Text Editor**.



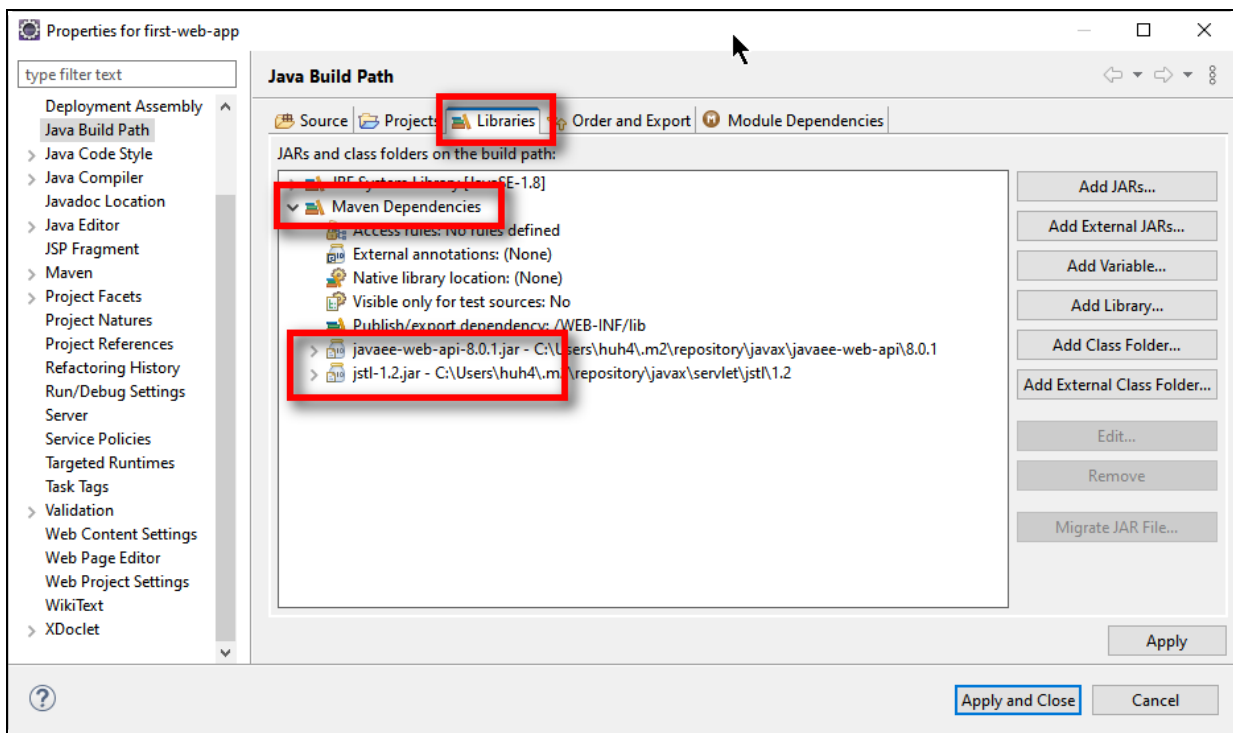
(11) Please download **pom.xml** from WebCampus, copy the content of it, and paste it into your **pom.xml**.

We need to specify the needed packages we would need to include in creating a Web Application

```
*pom.xml
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>edu.unlv.mis7684</groupId>
4   <artifactId>first-web-app</artifactId>
5   <version>0.0.1-SNAPSHOT</version>
6   <packaging>war</packaging>
7
8   <!-- add the dependencies; i.e., files need to import -->
9   <dependencies>
10    <dependency>
11      <groupId>javax</groupId>
12      <artifactId>javaee-web-api</artifactId>
13      <version>8.0.1</version>
14      <scope>provided</scope>
15    </dependency>
16
17    <dependency>
18      <groupId>javax.servlet</groupId>
19      <artifactId>jstl</artifactId>
20      <version>1.2</version>
21    </dependency>
22  </dependencies>
23
24  <!-- specify the tool used in the development environment -->
25  <build>
26    <pluginManagement>
27      <plugins>
28        <plugin>
29          <groupId>org.apache.maven.plugins</groupId>
30          <artifactId>maven-compiler-plugin</artifactId>
31          <version>3.8.1</version>
32          <configuration>
33            <verbose>true</verbose>
34            <source>1.8</source>
35            <target>1.8</target>
36            <showWarnings>true</showWarnings>
37          </configuration>
38        </plugin>
39        <plugin>
40          <groupId>org.apache.tomcat.maven</groupId>
41          <artifactId>tomcat7-maven-plugin</artifactId>
42          <version>2.2</version>
43          <configuration>
44            <path>/${path}</path>
45            <contextReloadable>true</contextReloadable>
46          </configuration>
47        </plugin>
48      </plugins>
49    </pluginManagement>
50  </build>
51 </project>
```

(12) Once we specify the files needed in **pom.xml**, the libraries will be automatically downloaded to the project if you are connected to the Internet. Wait for a few minutes for Eclipse to complete the download.

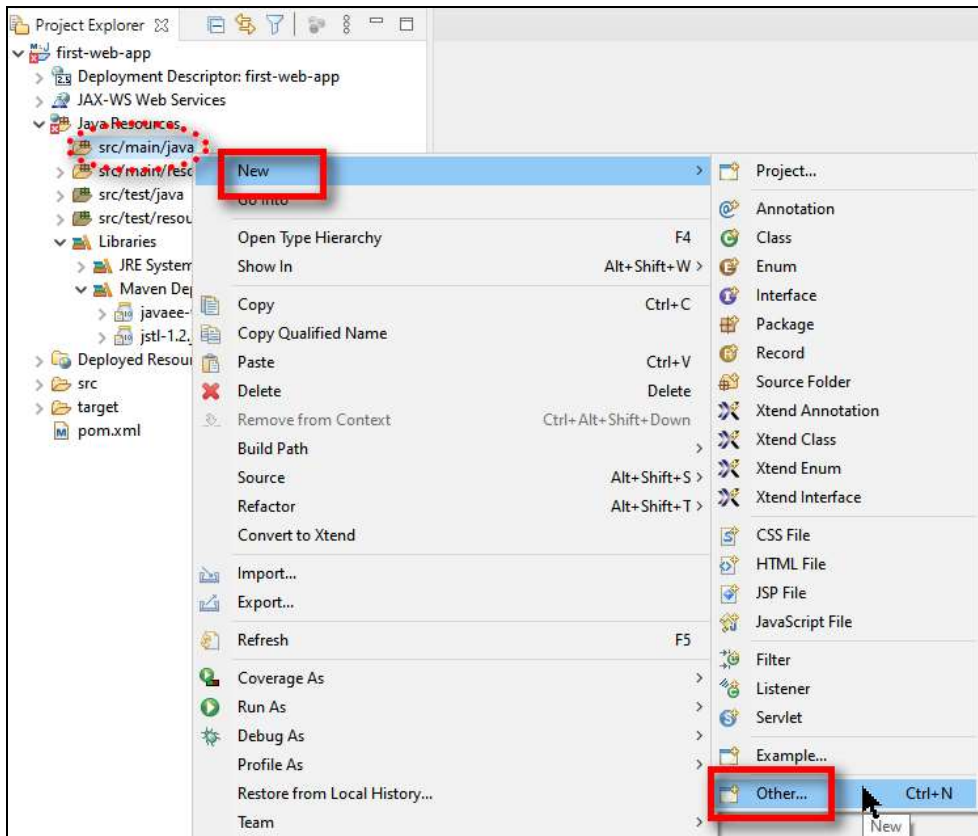
Right click on project, select Build Path, and then you can find them under **Maven Dependencies**.



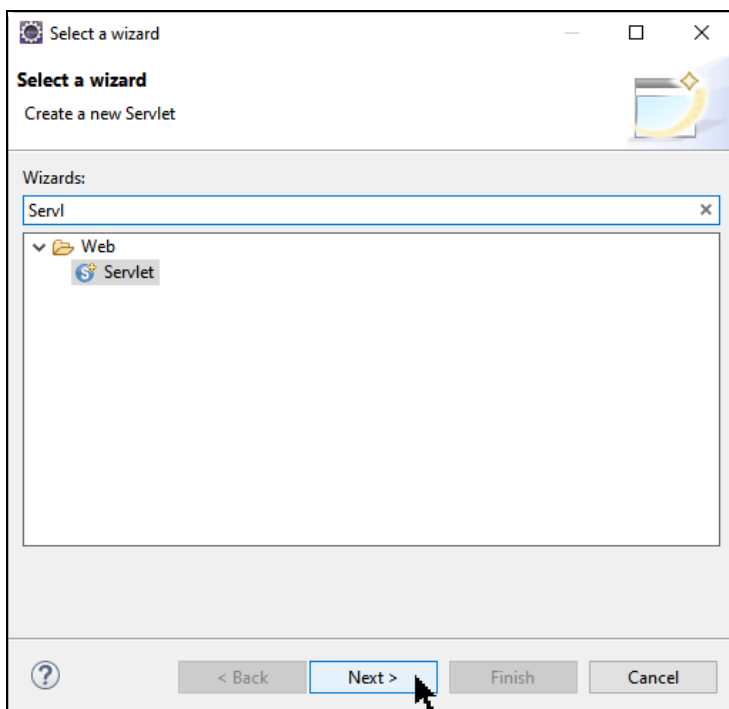
(13) Now we are ready to implement a Java web application.

2. Create the first Servlet: LoginServlet

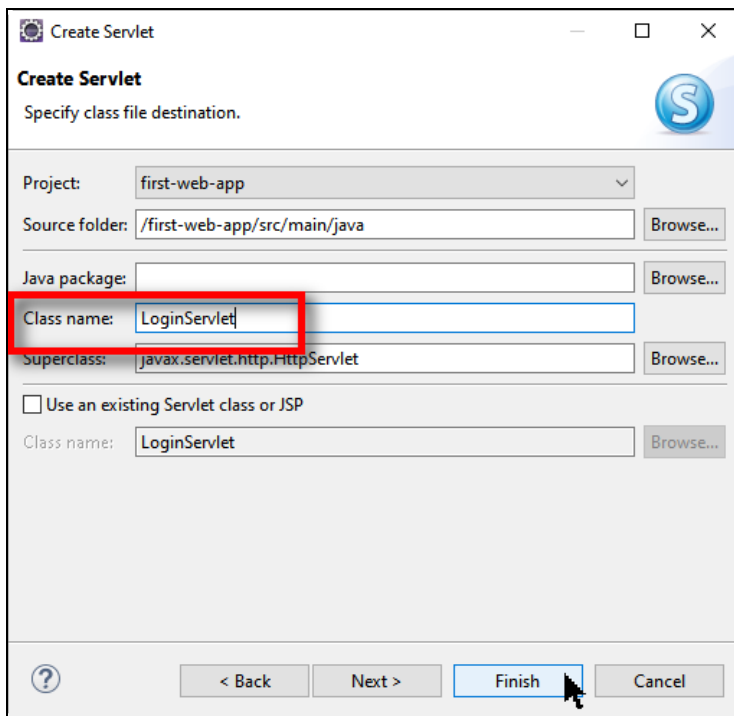
(14) Under **src/main/java**, please right click, select **New \ Other ...**



(15) In the **Wizard**, search for Servlet, and select **Web \ Servlet**



(16) Please name it as **LoginServlet**



The 'Create Servlet' dialog box is shown. It has a title bar with a question mark icon and standard window controls. The main title is 'Create Servlet' with a sub-label 'Specify class file destination.' and a blue 'S' icon. The fields are: 'Project' (first-web-app), 'Source folder' (/first-web-app/src/main/java), 'Java package' (empty), 'Class name' (LoginServlet, highlighted with a red box), and 'Superclass' (javax.servlet.http.HttpServlet). There are 'Browse...' buttons for the last three fields. Below these is a checkbox 'Use an existing Servlet class or JSP' which is unchecked, with a 'Class name' field (LoginServlet) and a 'Browse...' button. At the bottom are buttons for '?', '< Back', 'Next >', 'Finish' (highlighted with a mouse cursor), and 'Cancel'.

(17) A default template for Servlet will be created. We can clean it up a bit to keep only the **doGet()** method:



```
1
2
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.http.HttpServlet;
6 import javax.servlet.http.HttpServletRequest;
7 import javax.servlet.http.HttpServletResponse;
8
9 /**
10  * Servlet implementation class LoginServlet
11  */
12 public class LoginServlet extends HttpServlet {
13
14     /**
15      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
16      */
17     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
18
19     }
20
21
22
23 }
```

(18) Enter the following code.

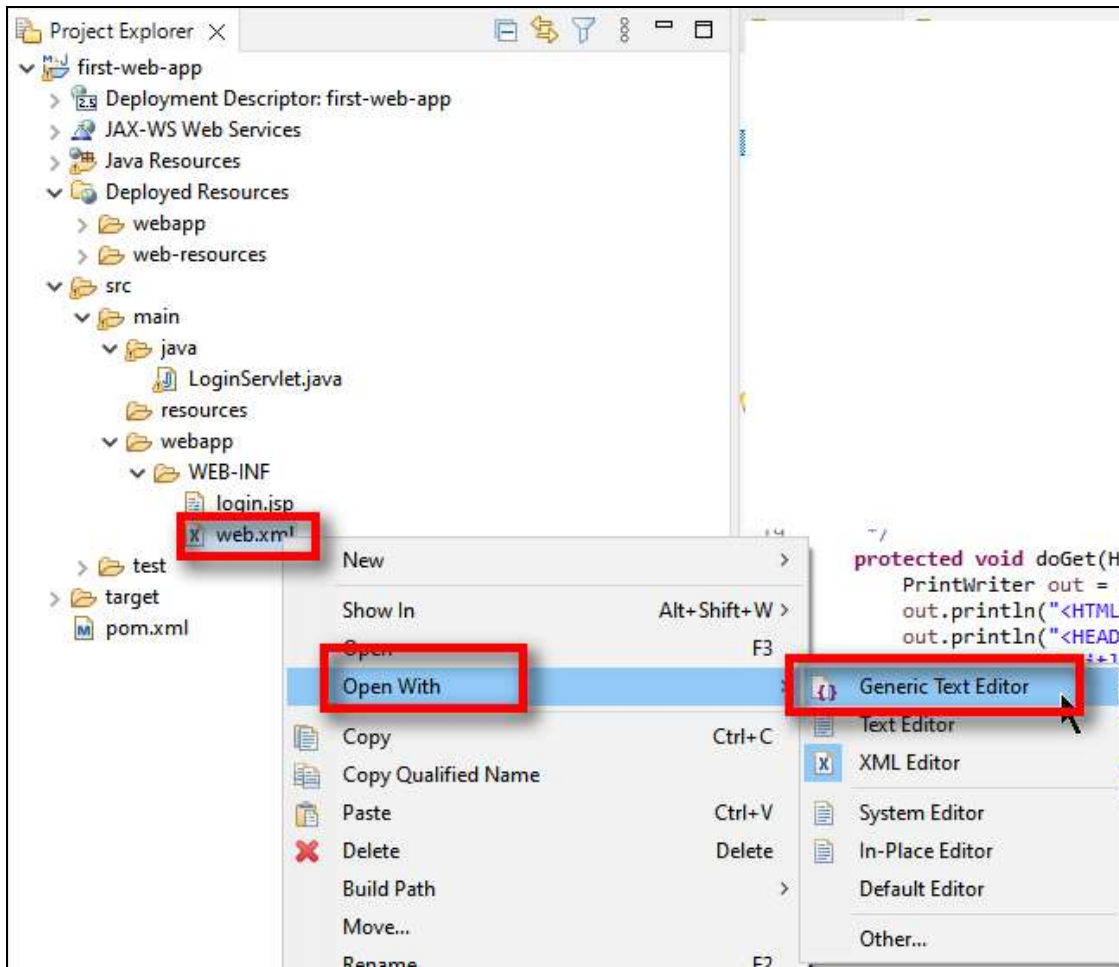
It specifies the web URL receiving the request. Please import the needed package to fix the errors.

```
*LoginServlet.java
1
2
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.annotation.WebServlet;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9
10 @WebServlet (urlPatterns = "/login.do")
11
12 * Servlet implementation class LoginServlet
13 */
14 public class LoginServlet extends HttpServlet {
15
16     /**
17     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
18     */
19     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException
20     {
21
22
23
24
25 }
26
```

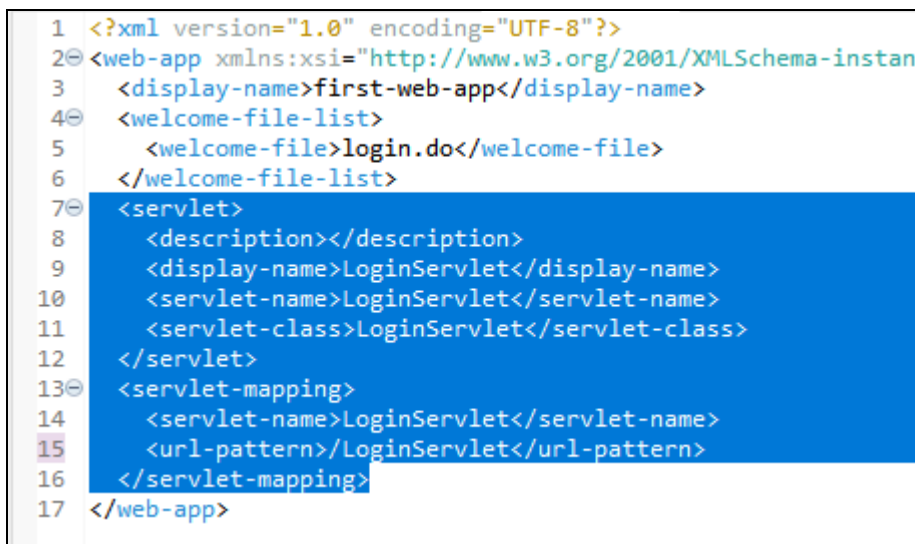
(19) We can write the output to the webpage by using a **PrintWriter** object and then use the **println()** method to output the strings. Save the file and we are ready to test this servlet.

```
3 import java.io.IOException;
4 import java.io.PrintWriter;
5
6 import javax.servlet.ServletException;
7 import javax.servlet.annotation.WebServlet;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 @WebServlet (urlPatterns = "/login.do")
13 /**
14 * Servlet implementation class LoginServlet
15 */
16 public class LoginServlet extends HttpServlet {
17
18     /**
19     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
20     */
21     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException
22     {
23         PrintWriter out = response.getWriter();
24
25         out.println("<HTML>");
26         out.println("<head>");
27         out.println("<title>Coffee Shop XYZ</title>");
28         out.println("</head>");
29         out.println("<body>");
30         out.println("My first servlet");
31         out.println("</body>");
32         out.println("</HTML>");
33     }
34 }
```

- (20) Upon creation of the servlet, the **web.xml** was updated with the new servlet information. We need to edit the file to remove it. Please right click on **web.xml**, select **Open With \ Generic Text Editor**.

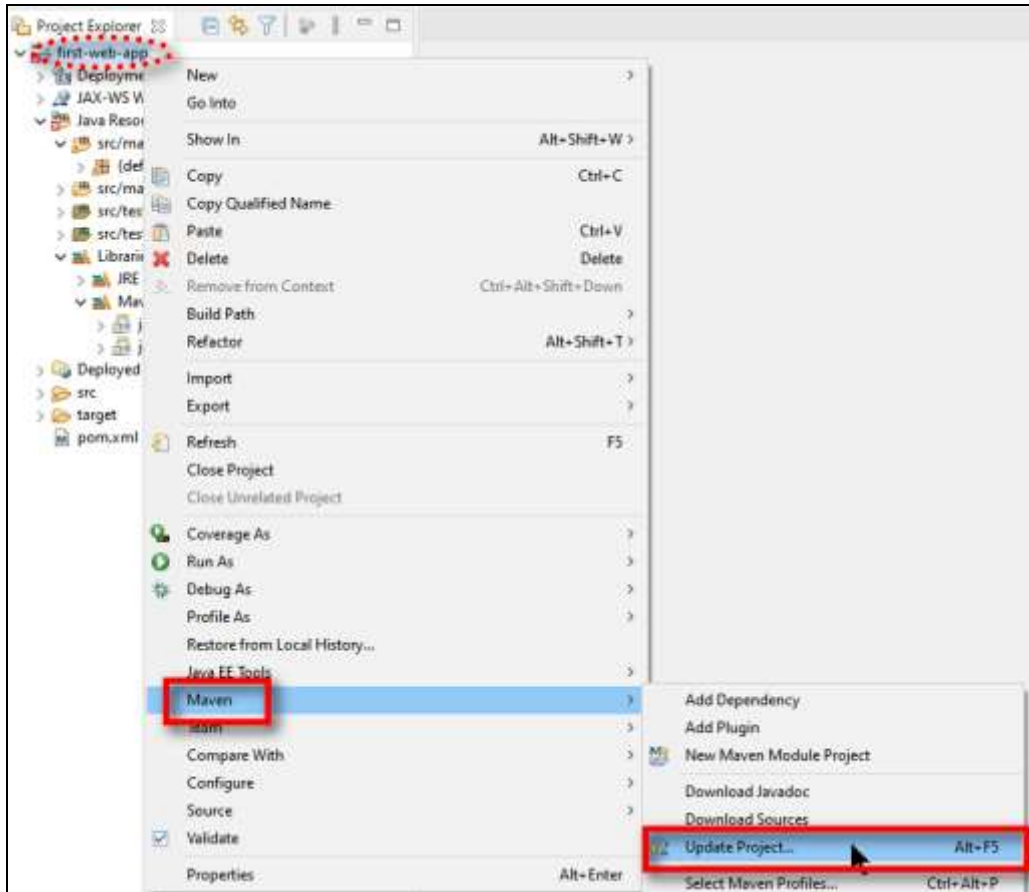


- (21) Remove lines 7-16. Save and close the file.



(22) Since we updated the configurations of the project, we need to refresh the project.

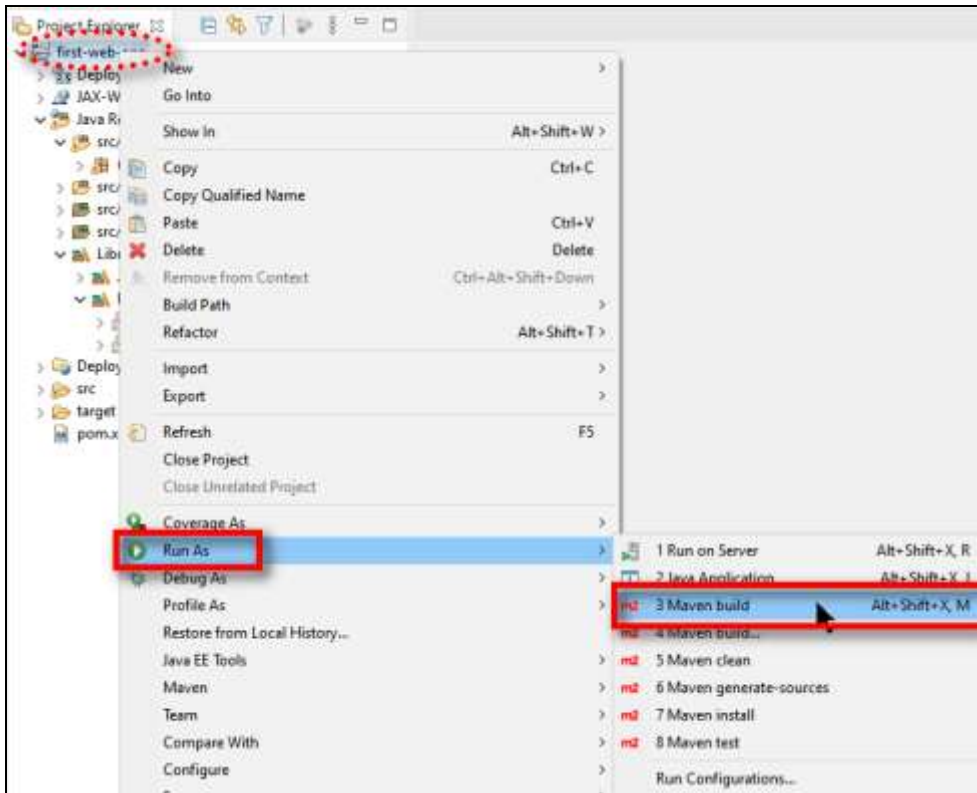
Right click on the project, select **Maven \ Update Project**



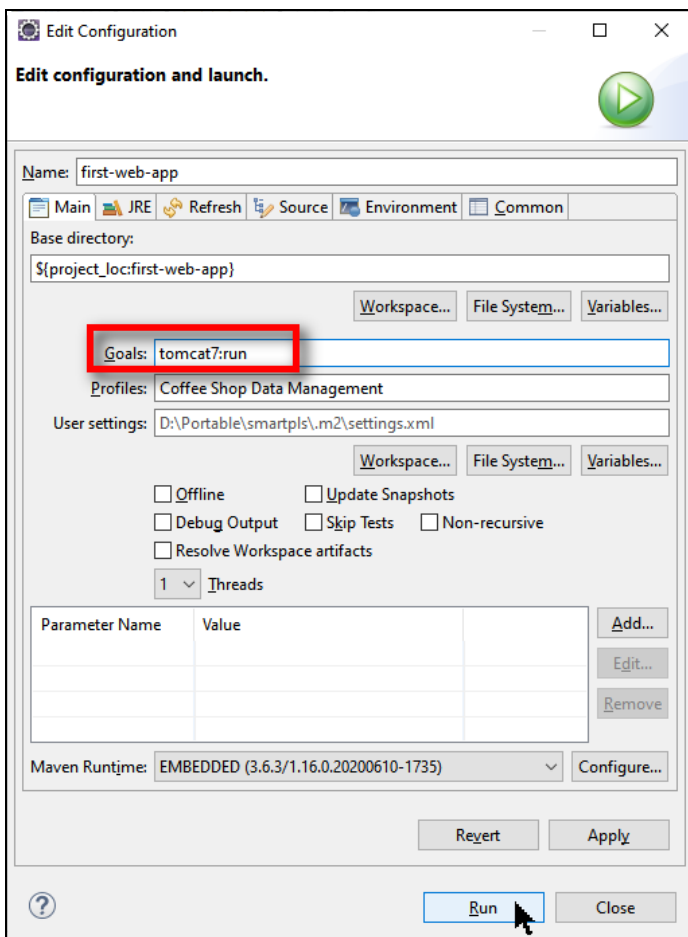
(23) Click **OK** button.



(24) To run the servlet on a built-in Web server, right click on the project, select **Run As \ Maven build**



(25) As **Goals**, enter **tomcat7:run**. Click the **Run** button.



- (26) For the first time of running this project, it will take a minute to download relevant files.

```
first-web-app [Maven Build] C:\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_15.0.1.v20201027-0507\jre\bin\javaw.exe (
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/C:/eclipse/plugins/org.eclipse.m2e.maven.runtime.slf4j.simple_1.16.0.20200
SLF4J: Found binding in [file:/C:/eclipse/configuration/org.eclipse.osgi/5/0/.cp/org.slf4j/impl/StaticLoggerF
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.SimpleLoggerFactory]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/C:/eclipse/plugins/org.eclipse.m2e.maven.runtime.slf4j.simple_1.16.0.20200
SLF4J: Found binding in [file:/C:/eclipse/configuration/org.eclipse.osgi/5/0/.cp/org.slf4j/impl/StaticLoggerF
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.SimpleLoggerFactory]
[INFO] Scanning for projects...
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-install-plugin/
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-install-plugin/2
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-deploy-plugin/2
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-deploy-plugin/2.3
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-site-plugin/3.3/
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-site-plugin/3.3/
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-antrun-plugin/1
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-antrun-plugin/1.1
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/12/mave
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-plugins/12/mave
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-antrun-plugin/1
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-antrun-plugin/1.1
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-assembly-plugi
```

- (27) If you see a Security Alert in Windows, please click **Allow Access**.



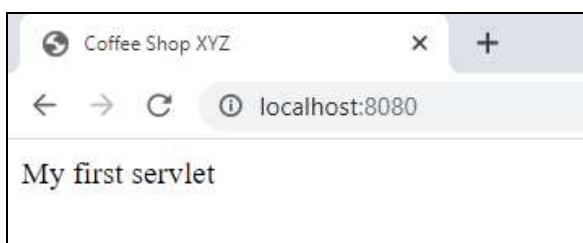
- (28) Once you see the **Running war on http://localhost:8080**, the server started successfully.

You are ready to test the servlet.

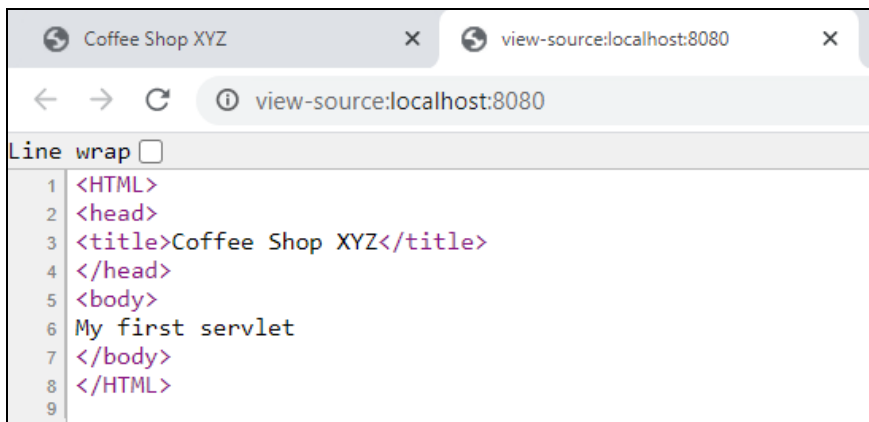
```
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/maven
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/slf4j/slf4j
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/maven
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/maven
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/slf4j/jcl-ov
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/maven
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/slf4j/slf4j-
[INFO] Running war on http://localhost:8080/
[INFO] Creating Tomcat server configuration at: \MIS768\web-example\first-web
[INFO] create webapp with contextPath:
```

- (29) Open a Web browser, and type in **localhost:8080** at the URL.

It sends a request to the servlet and get the response as a webpage. You can see the title is shown as specified.



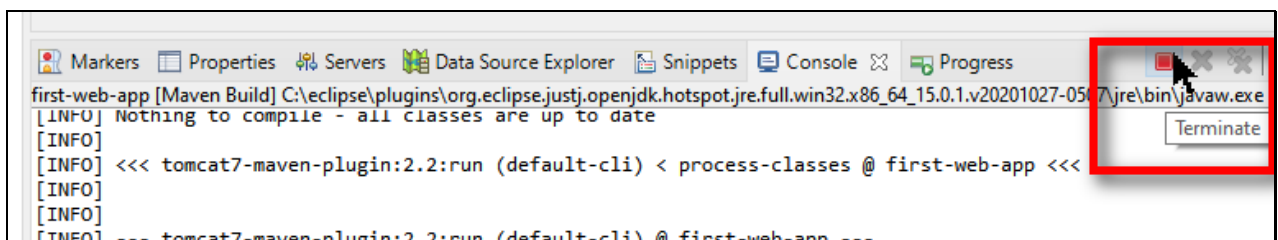
- (30) The page source shows exactly what the servlet write to the output using the PrintWrite object.



The screenshot shows a web browser window with the address bar displaying 'view-source:localhost:8080'. The page content is the source code of an HTML document. The code is as follows:

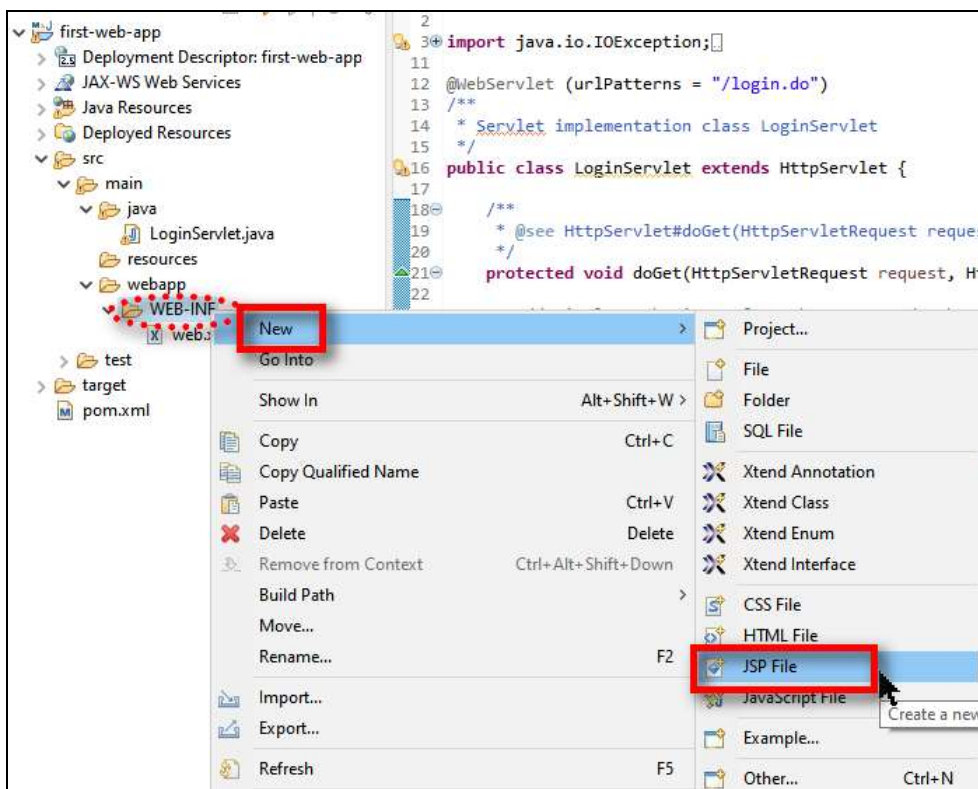
```
1 <HTML>
2 <head>
3 <title>Coffee Shop XYZ</title>
4 </head>
5 <body>
6 My first servlet
7 </body>
8 </HTML>
9
```

- (31) Please note that a Web server is up and running for your testing. To stop the server (i.e., killing the process), please click on the terminate icon at the console.

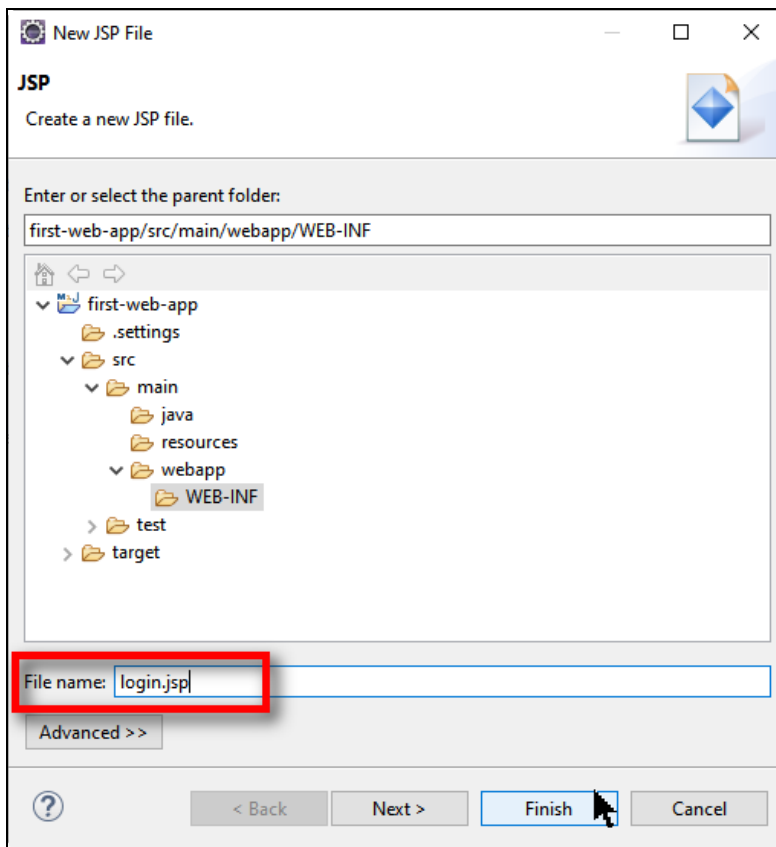


3. Create a Java Server Page: login.jsp

- (32) Since Servlet is not efficient in writing static output, it is recommended to redirect / dispatch the request to a Java Server Page (JSP).
- (33) Under **src/ main/ webapp/ WEB-INF**, create a JSP file



(34) Name it as **login.jsp**.



(35) Open **login.jsp** and enter the following text message between the **<body></body>** tag.
We will send the value of **username** from the Servlet.
Save and close the file.



(36) Please switch back to **LoginServlet.java** and revise the code as following:

```
2
3+ import java.io.IOException;
11
12 @WebServlet (urlPatterns = "/login.do")
13 /**
14  * Servlet implementation class LoginServlet
15  */
16 public class LoginServlet extends HttpServlet {
17
18     /**
19      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
20      */
21     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException {
22
23         // Simulate the input from the user or database query results
24         String str = "Jo";
25
26         // set the value for username
27         request.getSession().setAttribute("username", str);
28
29         // dispatch the request to jsp
30         request.getRequestDispatcher("/WEB-INF/login.jsp").forward(request, response);
31
32
33
34
35
36
```

(37) If your web server is still up and running, open a Web browser, and type in **localhost:8080** at the URL.
(If not, run the project again like what you did at step (22)).

It sends a request to the servlet. The servlet then writes the username string to the request object and then forward the request to the jsp.

