

Lesson 01 Demo 01

Create a JSP Page

Objective: To demonstrate creation of a JSP (Java Server Pages) page using Eclipse IDE

Tools Required: Eclipse IDE

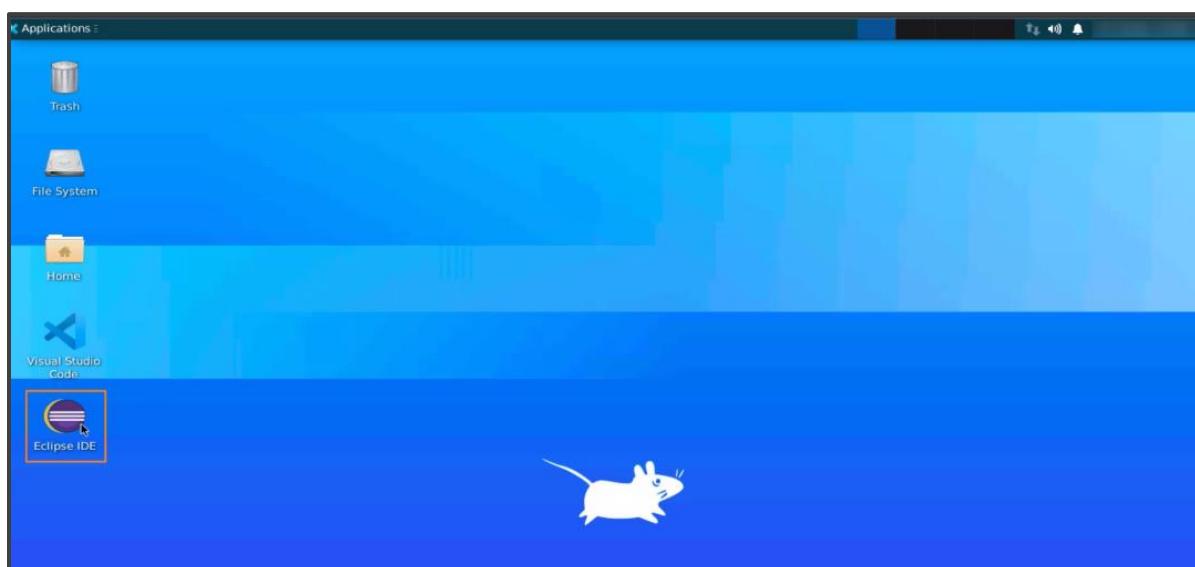
Prerequisites: None

Steps to be followed:

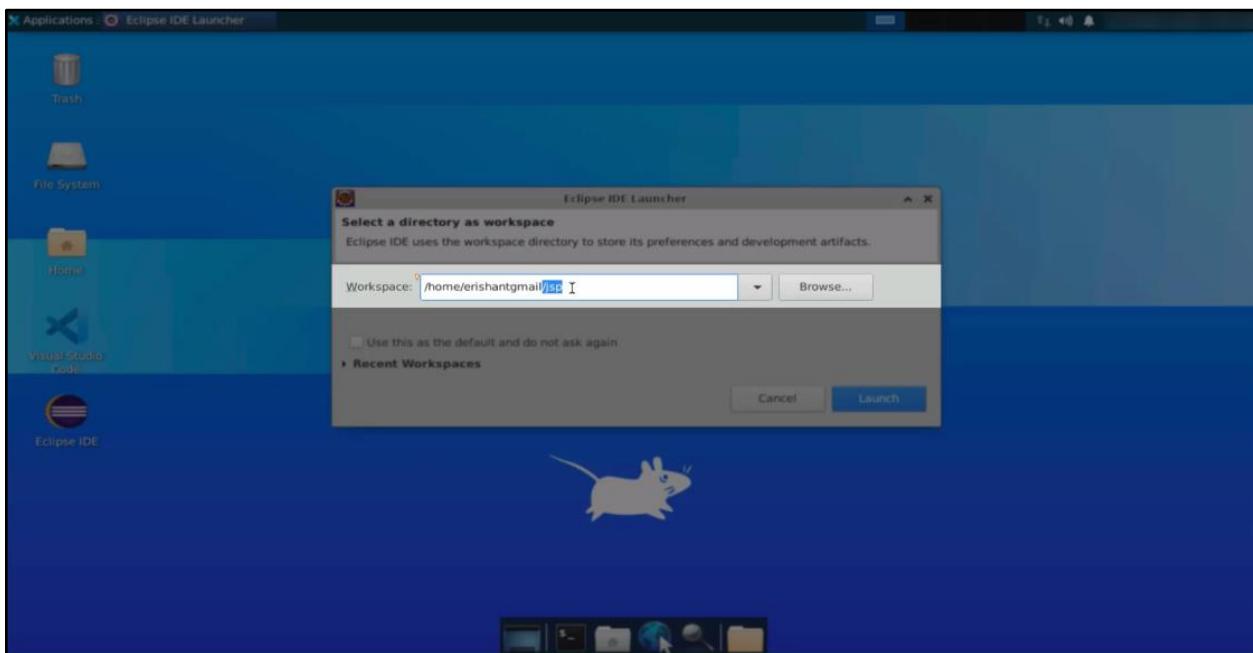
1. Create a new dynamic web project
2. Create a new JSP file
3. Incorporate changes in the previous operation
4. Create a new Maven project
5. Add the dependencies
6. Create a scriptlet tag

Step 1. Create a new dynamic web project

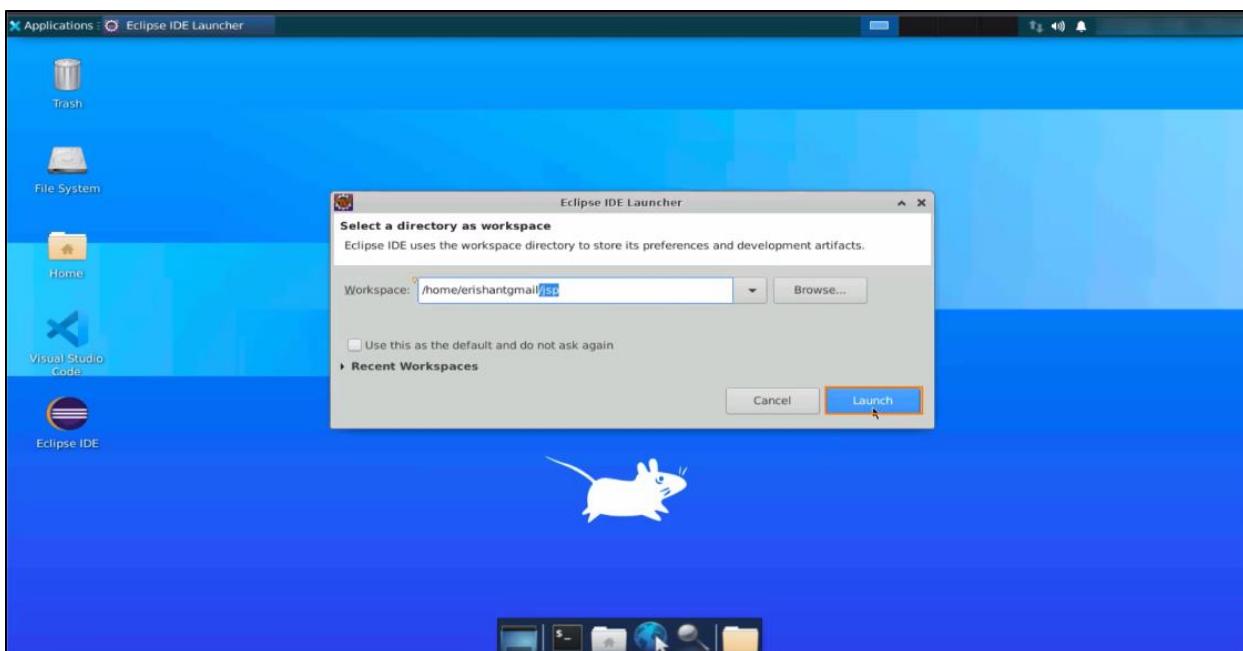
- 1.1 Open the **Eclipse IDE** in your lab



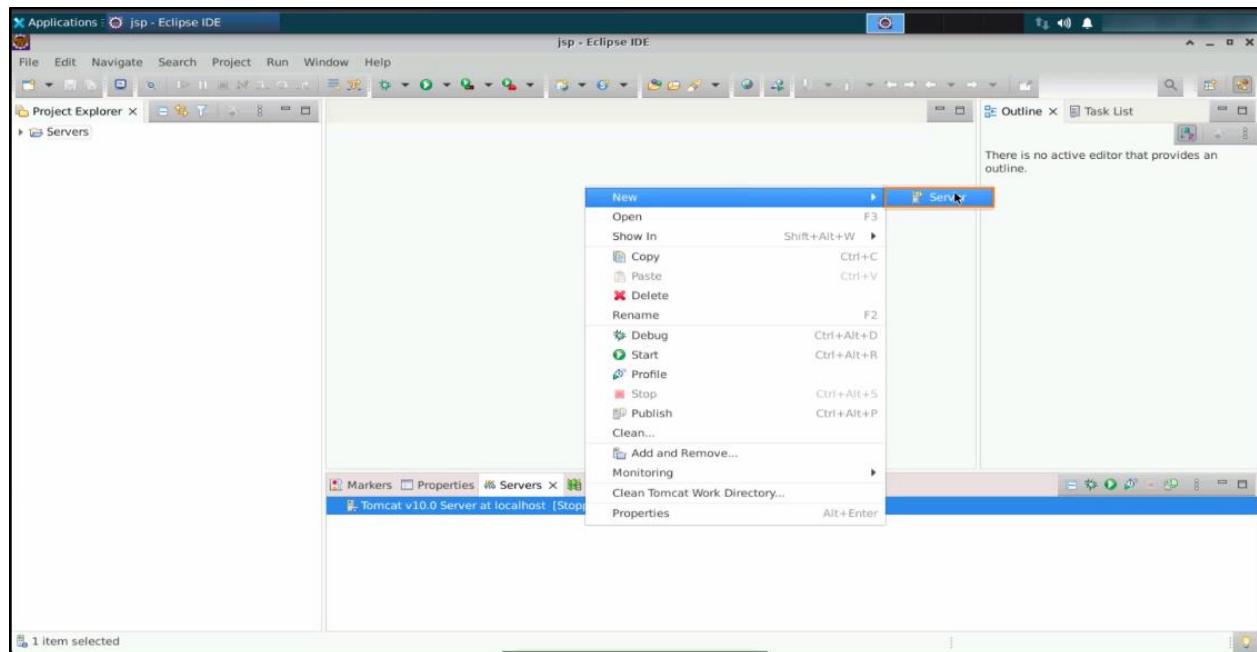
1.2 Set the workspace name as **jsp**



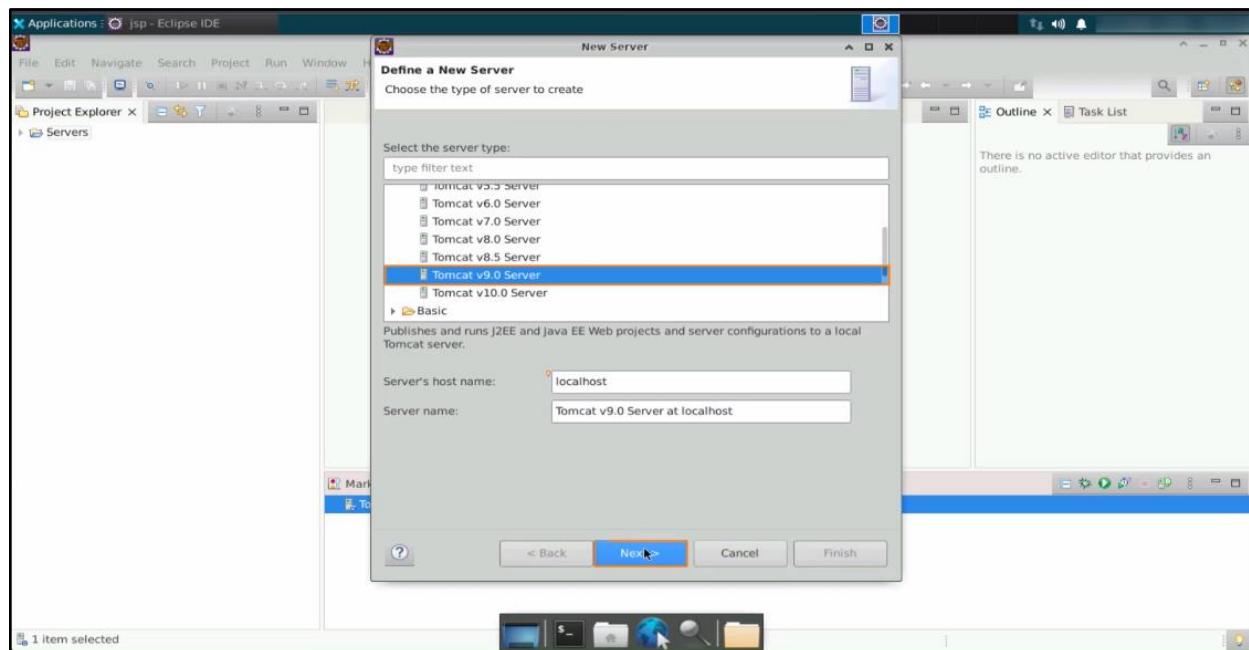
1.3 Click on **Launch** to open the IDE



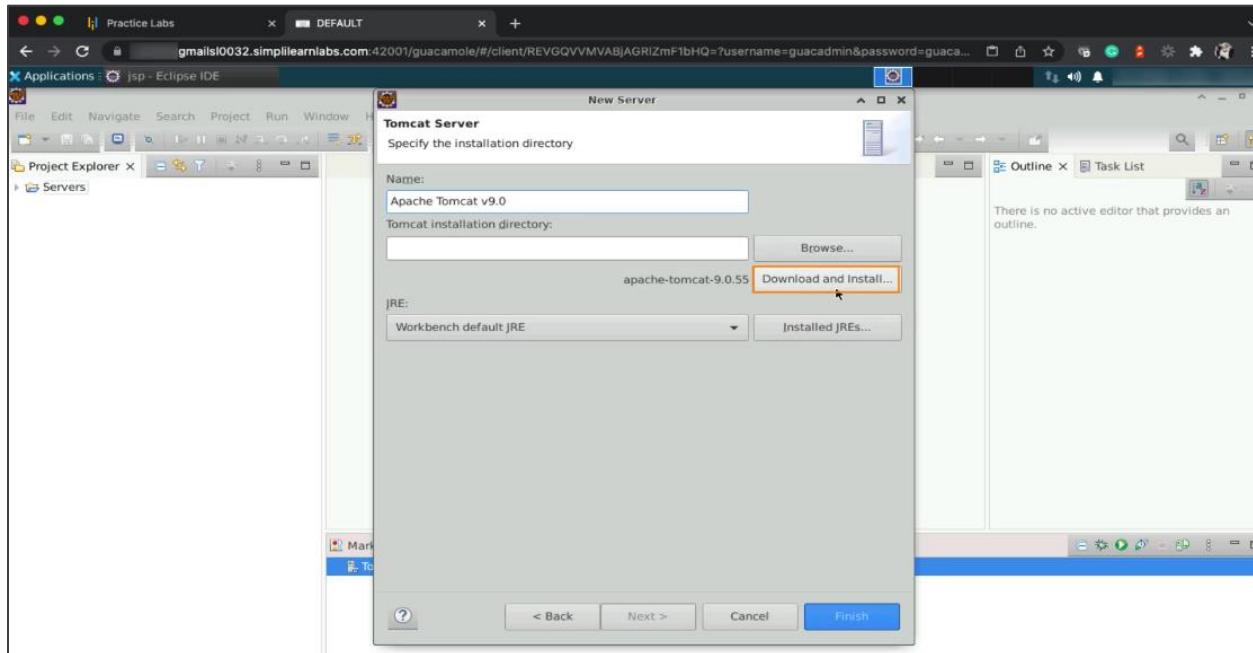
1.4 Right-click in the Project Explorer area and select **New**, then choose **Server** from the context menu



1.5 In the New Server dialog box, select **Tomcat v9.0 Server** and click **Next**

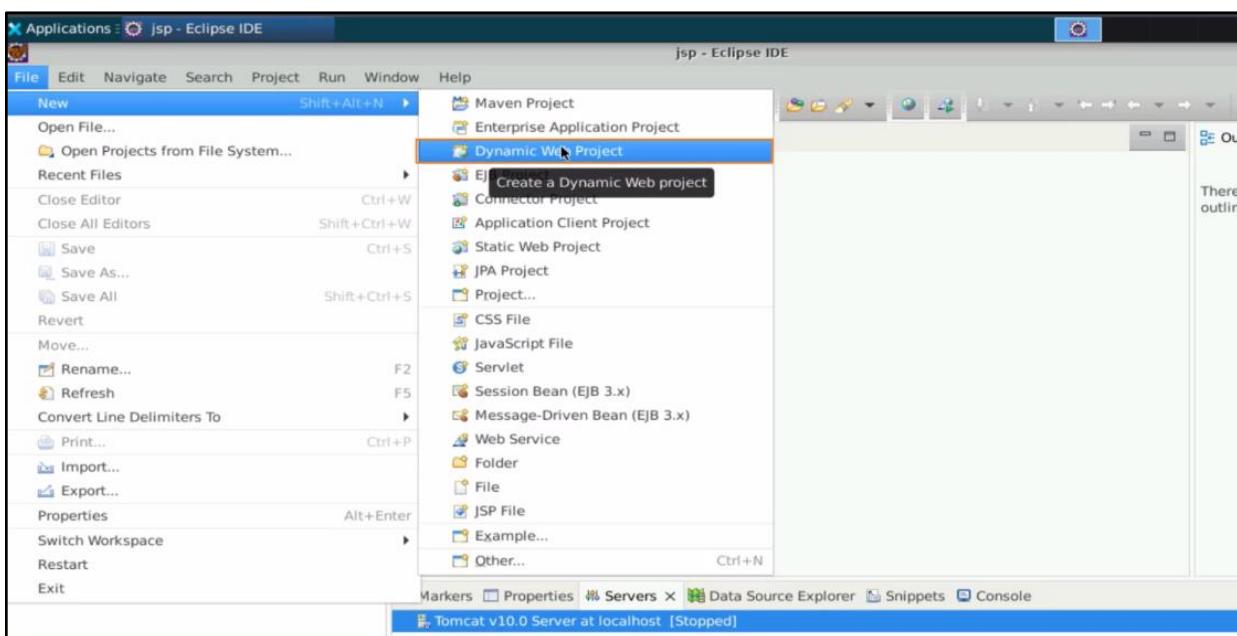


1.6 Choose the existing installation of Apache Tomcat or click on **Download and Install** to install it

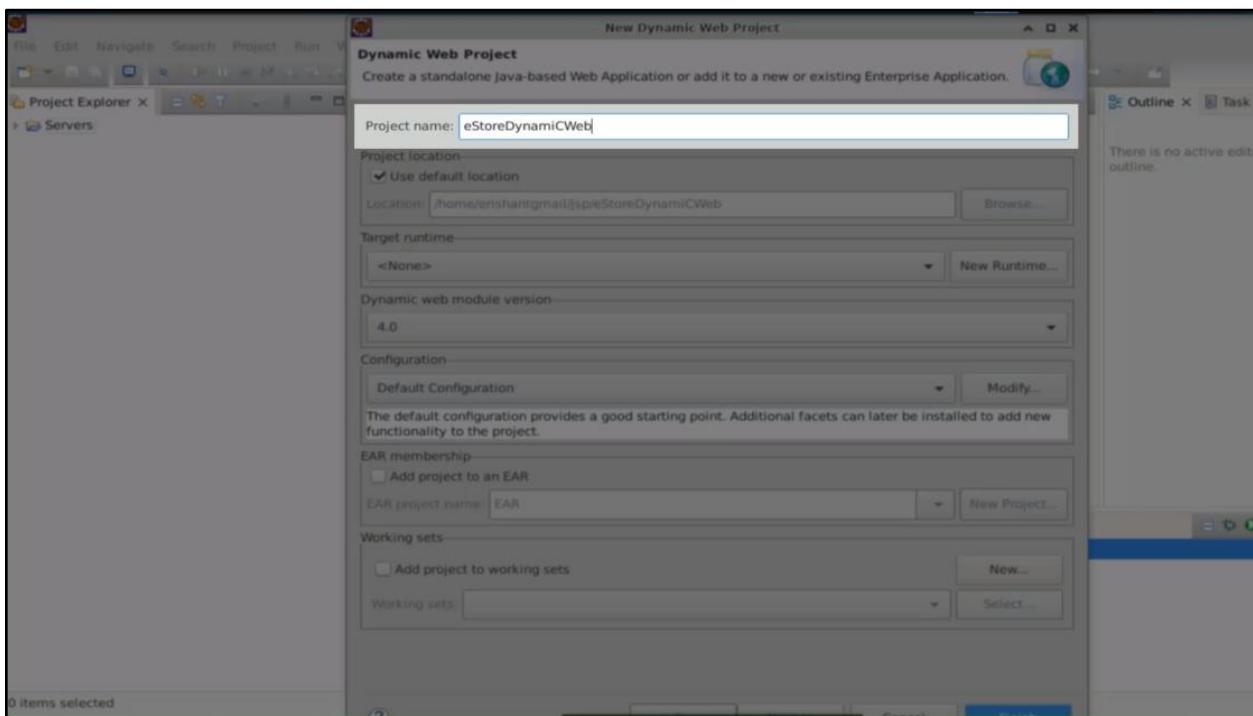


1.7 Create a new Dynamic Web Project by following the below path:

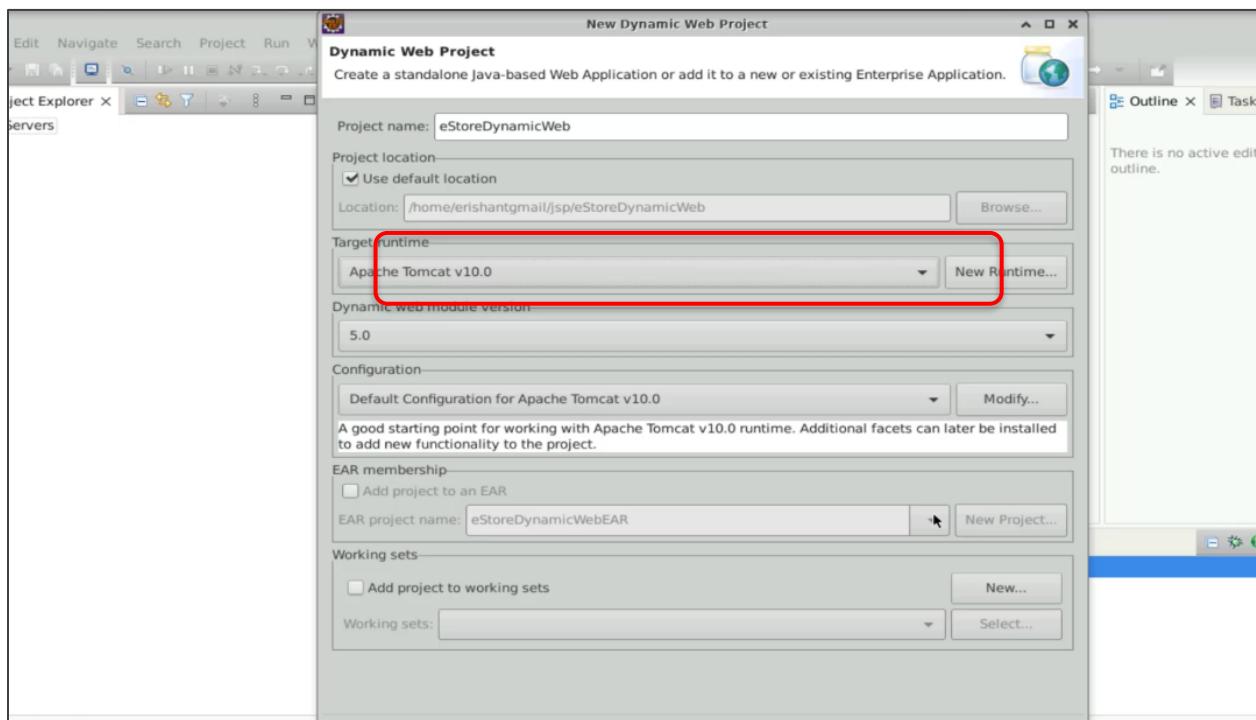
File->New->Dynamic Web Project



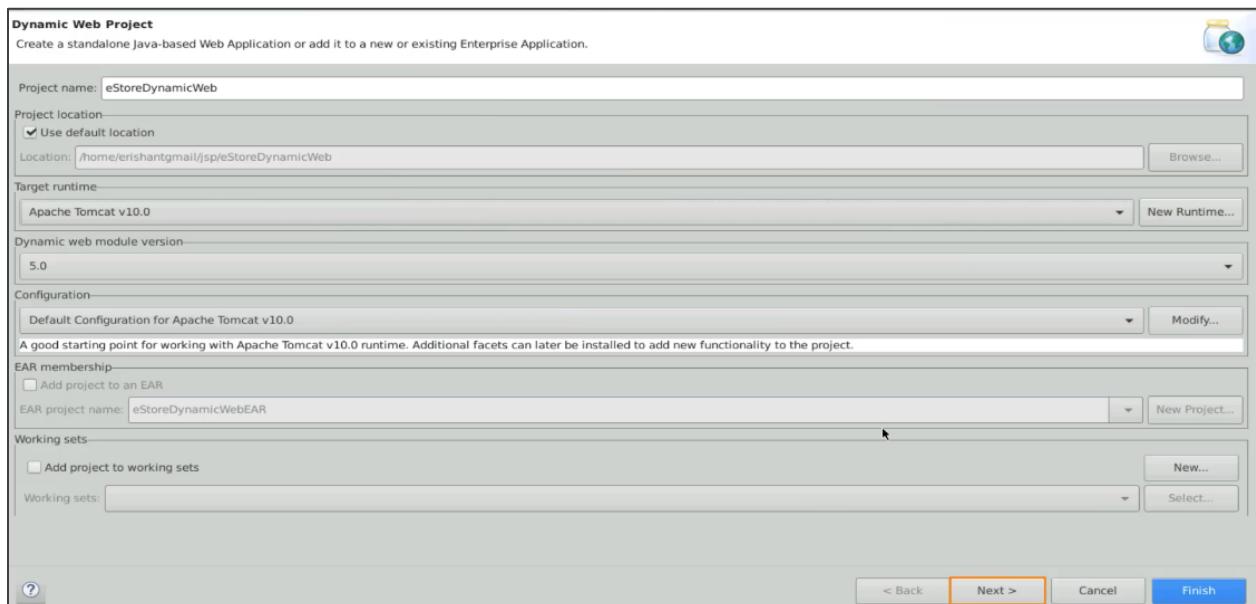
1.8 Name the project as eStoreDynamicWeb



1.9 Keep the default configuration and set the Target runtime as Apache Tomcat v10.0



1.10 Click on Next

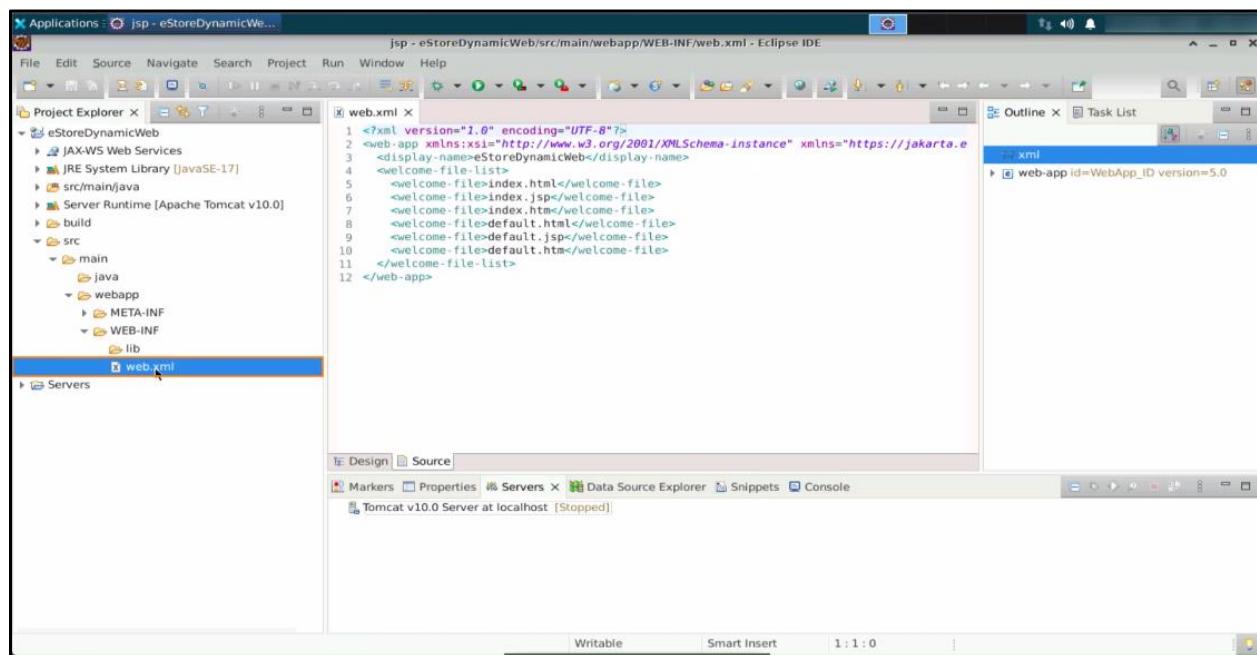


1.11 Check the option **Generate web.xml deployment descriptor** and click **Finish** to create the project

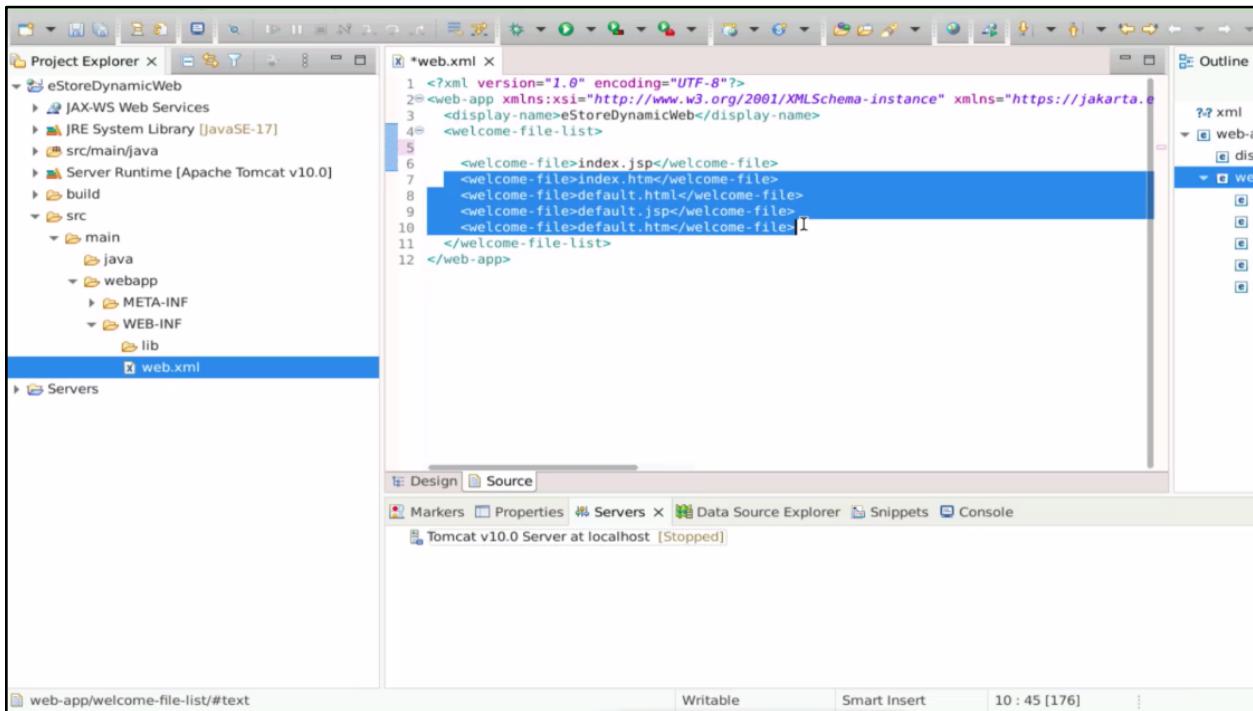


Step 2: Creating a new JSP file

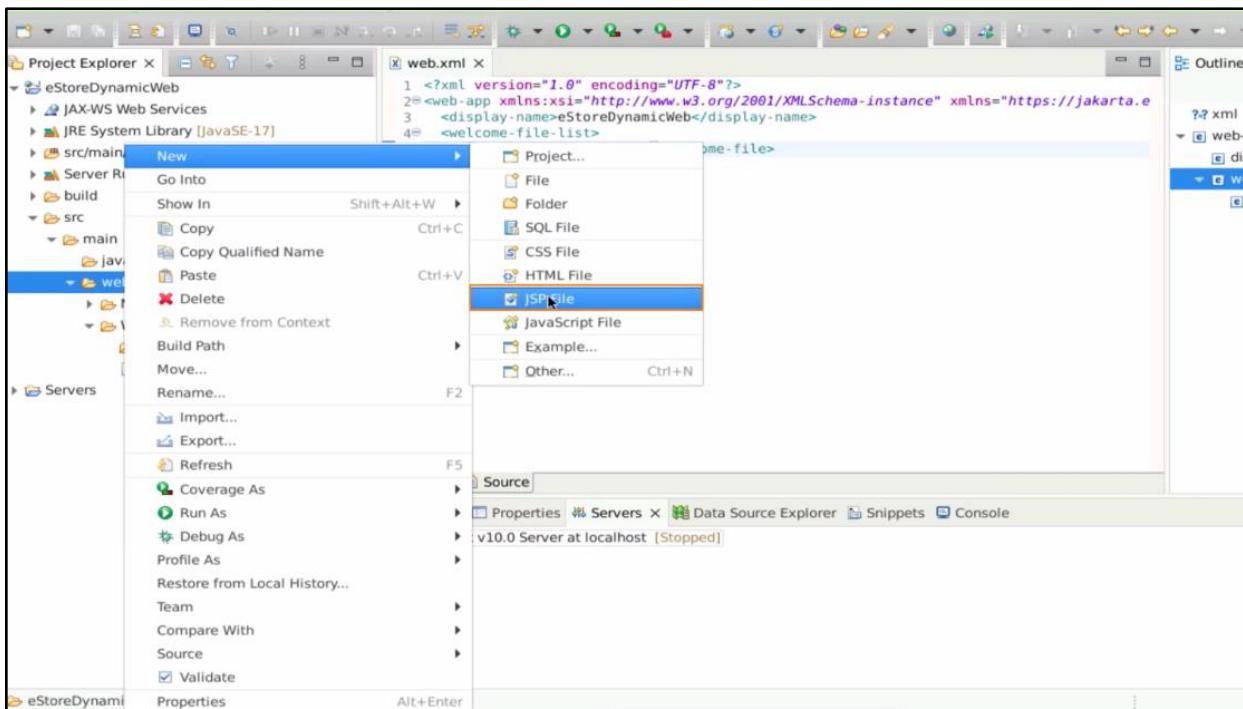
2.1 Open the **web.xml** file located in the **WEB-INF** directory of the project



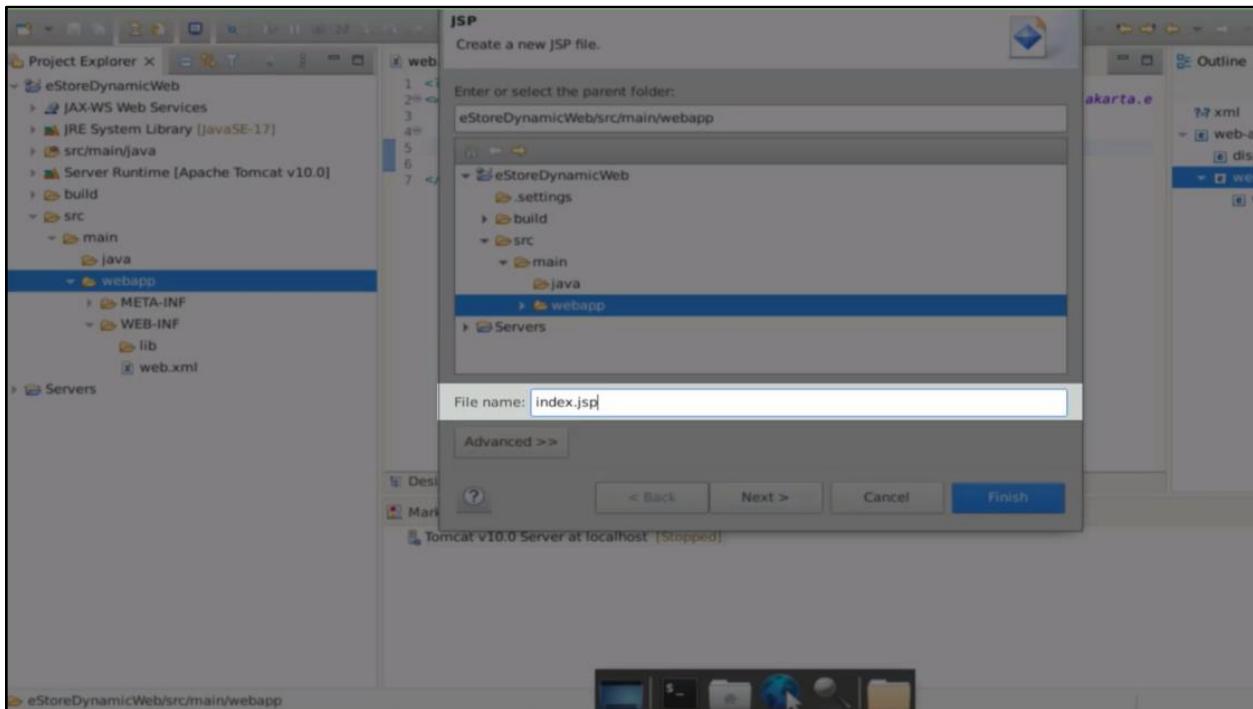
2.2 Remove all the files listed under the **welcome-file-list** section except the JSP file



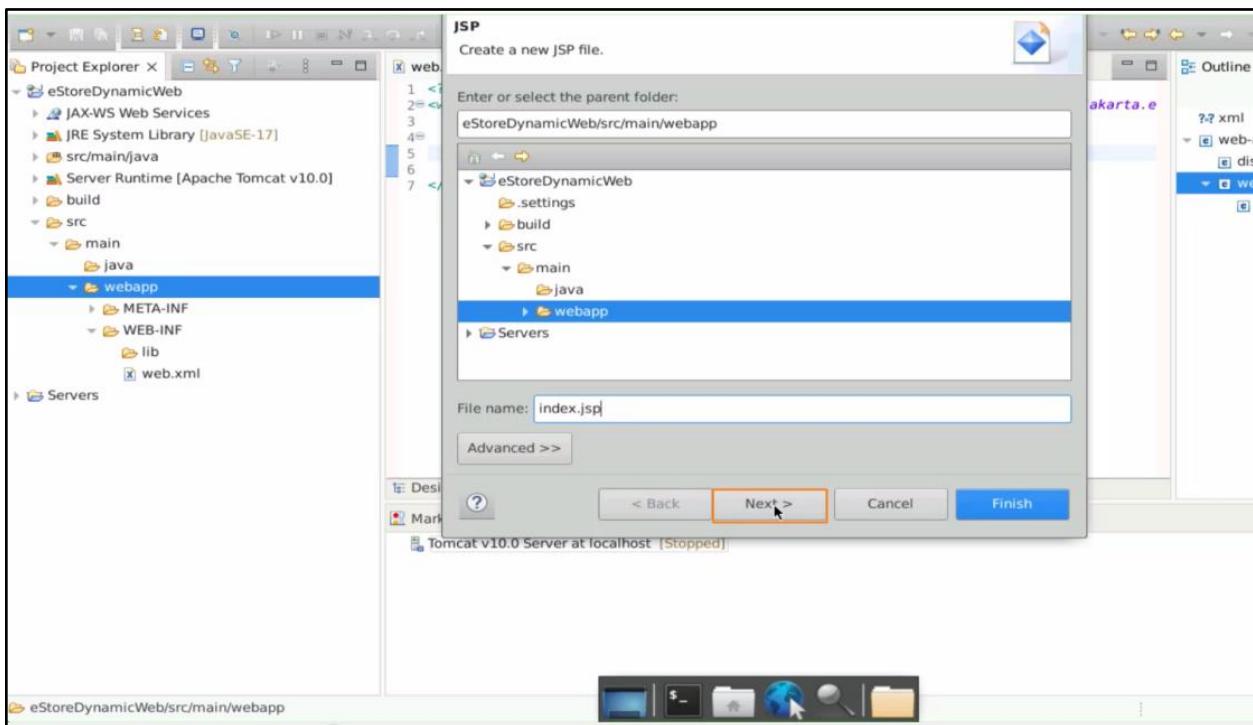
2.3 Right-click the webapp directory then select **New**, and choose **JSP File**

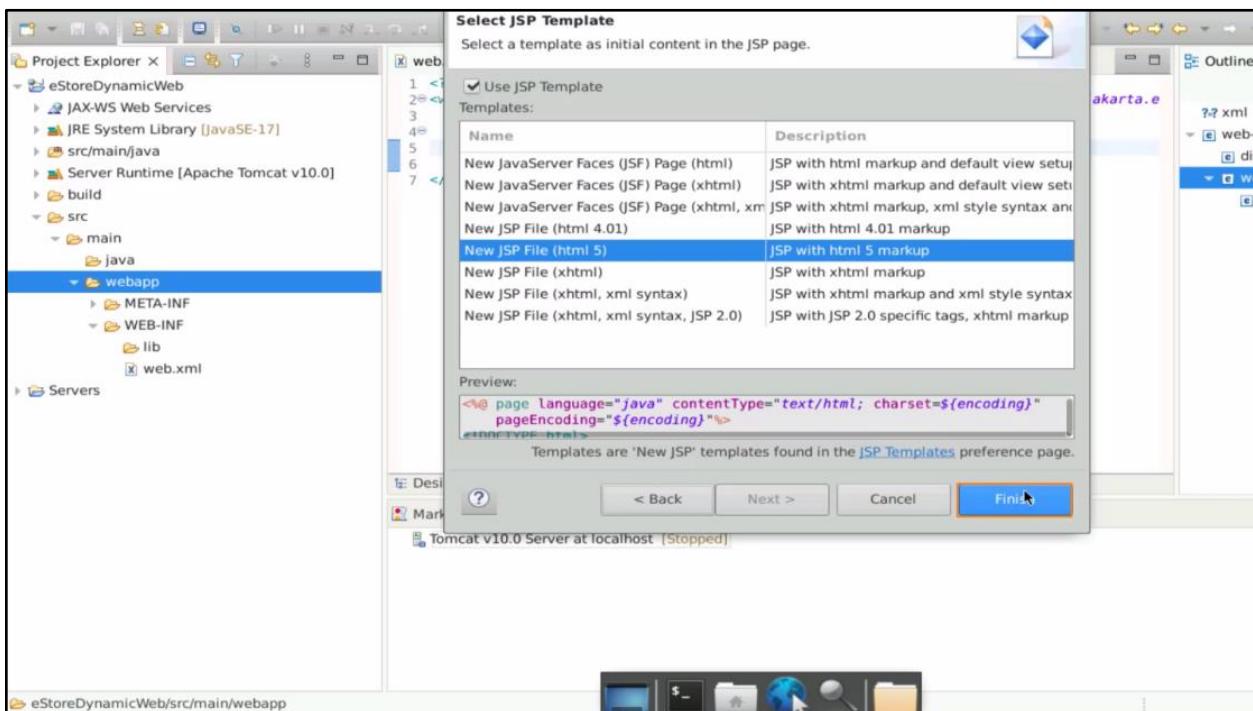


2.4 Name the file as **index.jsp**

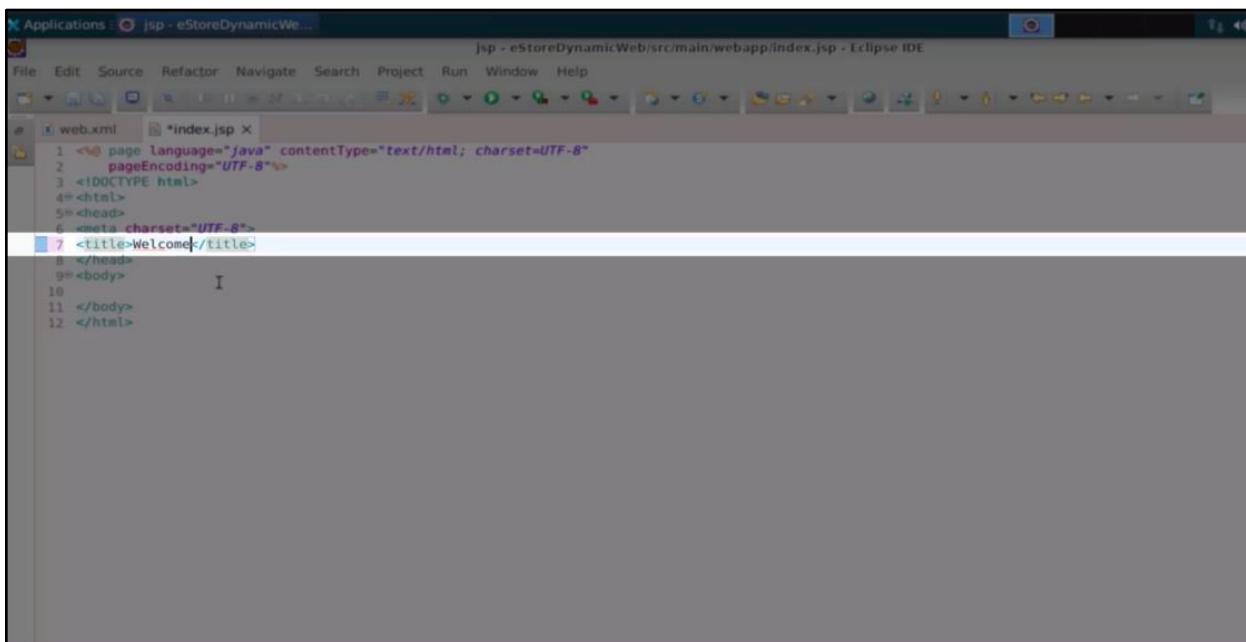


2.5 Click **Next** and then **Finish**



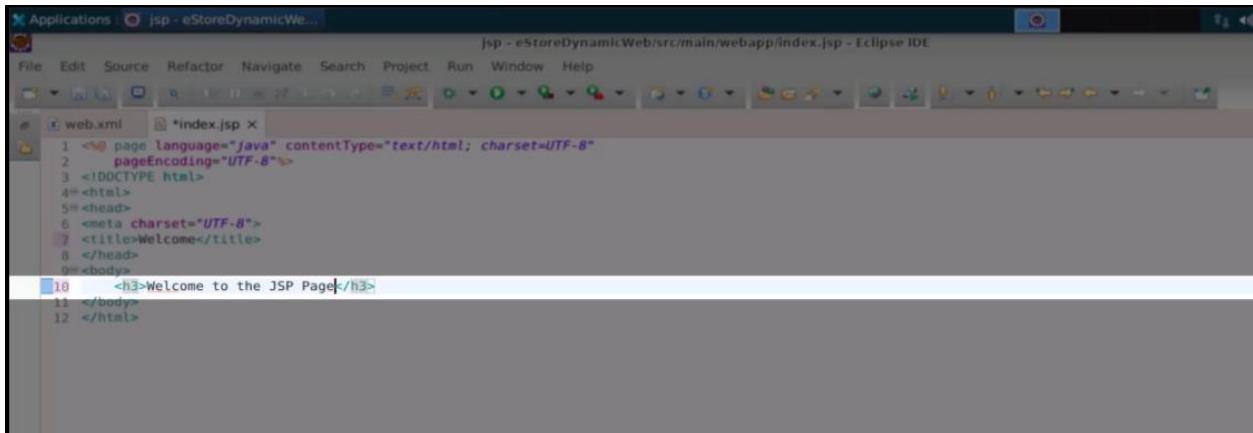


2.6 Open the index.jsp file and modify the title to Welcome



2.7 Add an h3 tag:

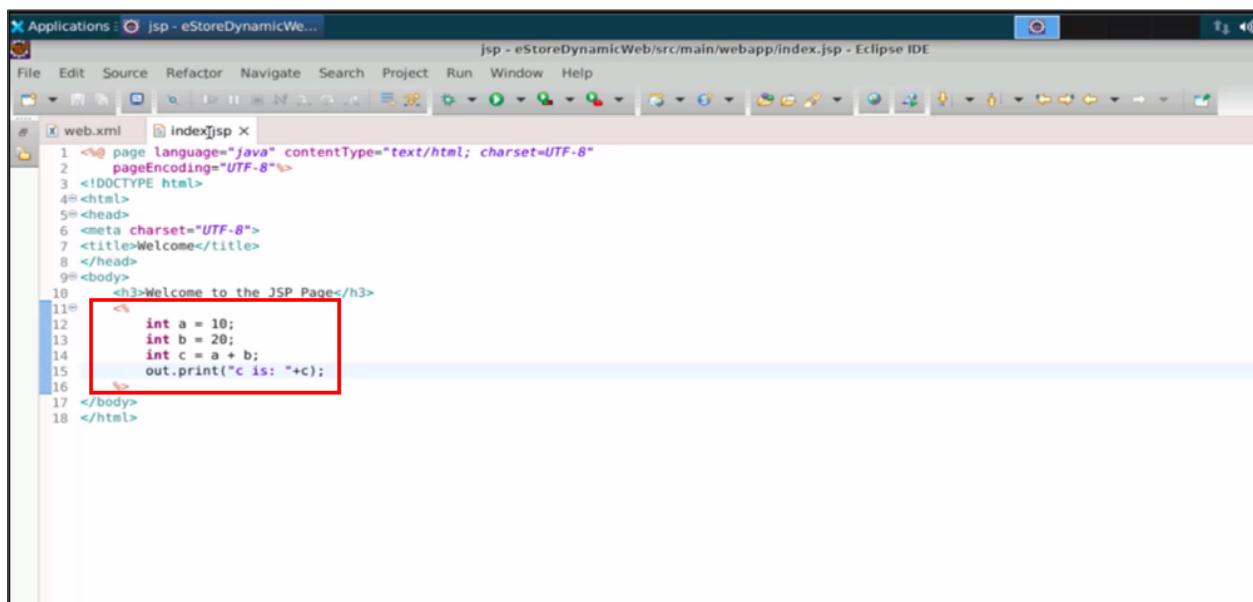
```
<h3>Welcome to the JSP Page</h3>
```



The screenshot shows the Eclipse IDE interface with the title bar "Applications jsp - eStoreDynamicWeb...". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations. The left sidebar shows a tree view with "web.xml" and "index.jsp". The main editor window displays the following code:

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2 pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="UTF-8">
7 <title>Welcome</title>
8 </head>
9 <body>
10 <h3>Welcome to the JSP Page</h3>
11 </body>
12 </html>
```

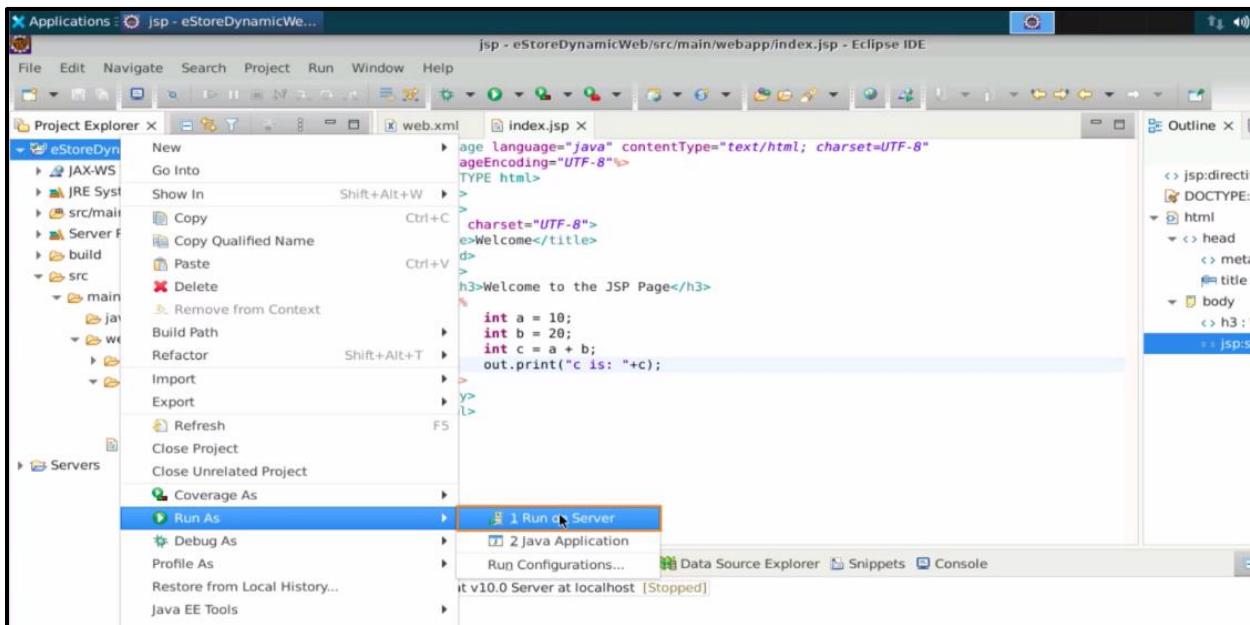
2.8 Create a scriptlet tag <% %> and add the desired logic or code within it



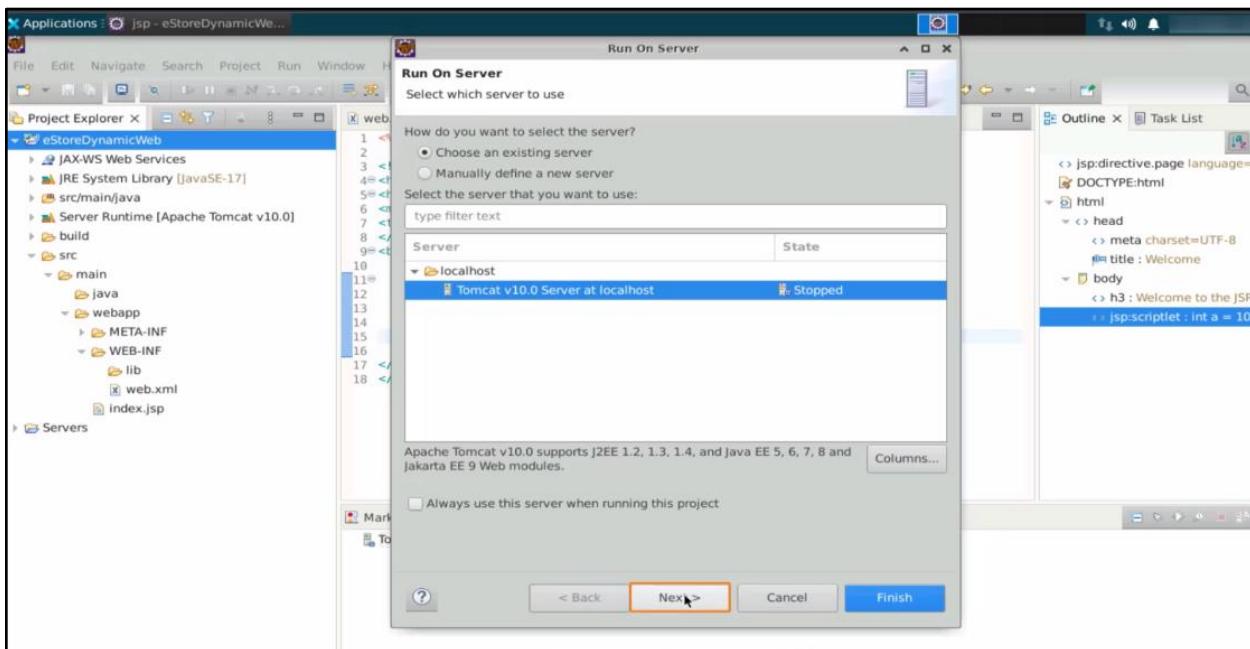
The screenshot shows the Eclipse IDE interface with the title bar "Applications jsp - eStoreDynamicWeb...". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations. The left sidebar shows a tree view with "web.xml" and "index.jsp". The main editor window displays the following code, with lines 11-15 highlighted by a red box:

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2 pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="UTF-8">
7 <title>Welcome</title>
8 </head>
9 <body>
10 <h3>Welcome to the JSP Page</h3>
11<% int a = 10;
12 int b = 20;
13 int c = a + b;
14 out.print("c is: "+c);
15%>
16 </body>
17 </html>
```

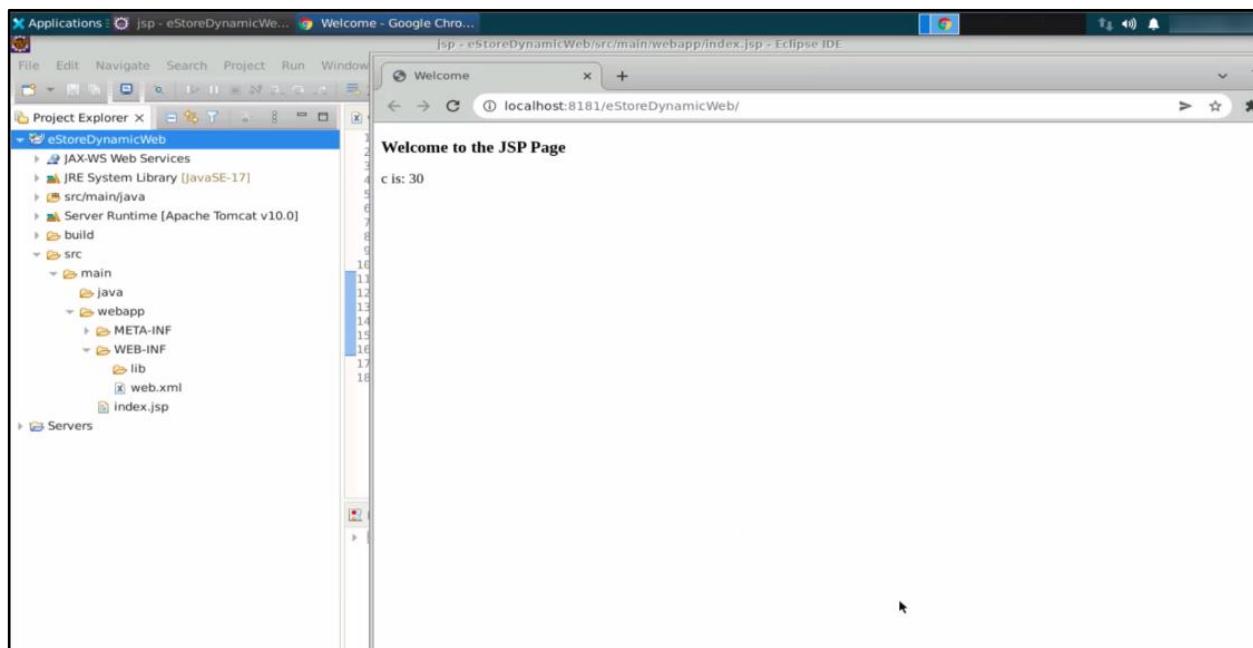
2.9 Right-click on the project, select Run As, and click on Run on Server



2.10 Choose the configured Tomcat server and click **Next**



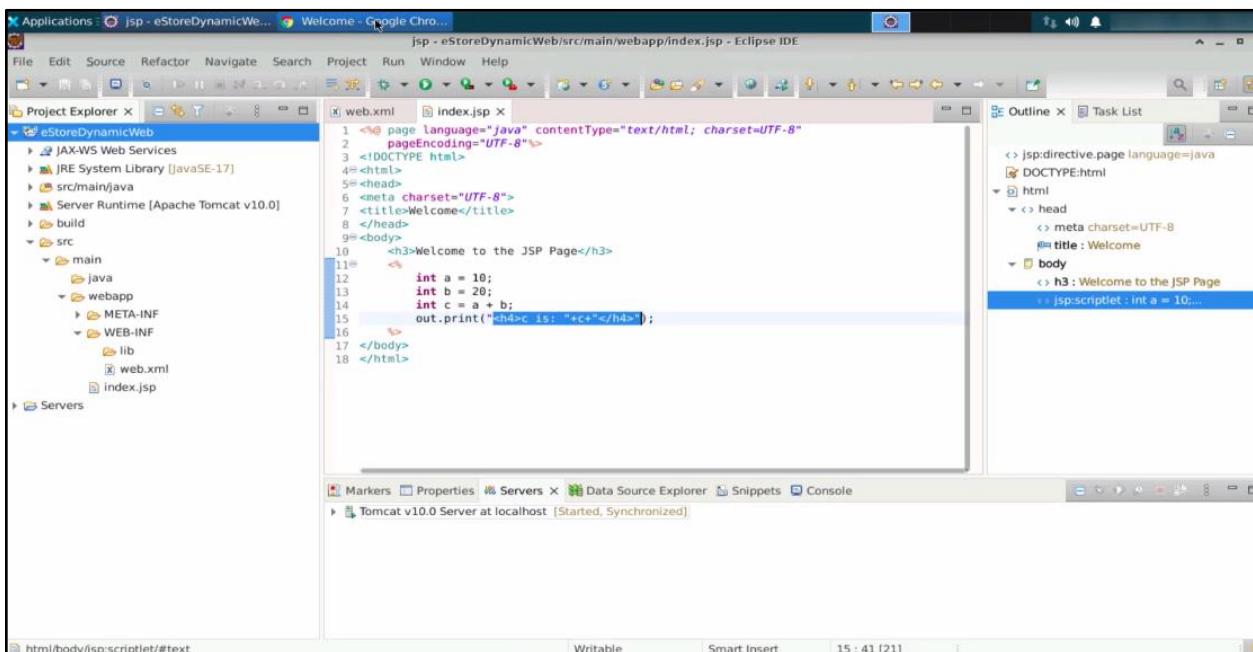
2.11 Review the server configuration and click **Finish** to run the project on the server



The JSP page will be displayed in the web browser.

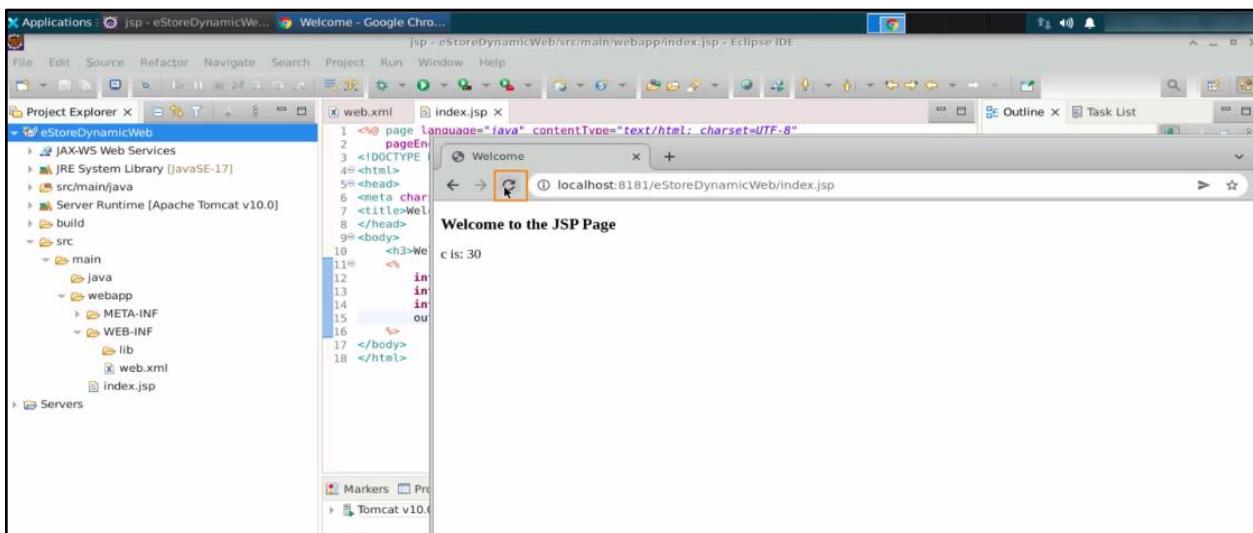
Step 3: Incorporate changes in the previous operation

3.1 If any changes are required, modify the necessary files or configurations based on the previous steps

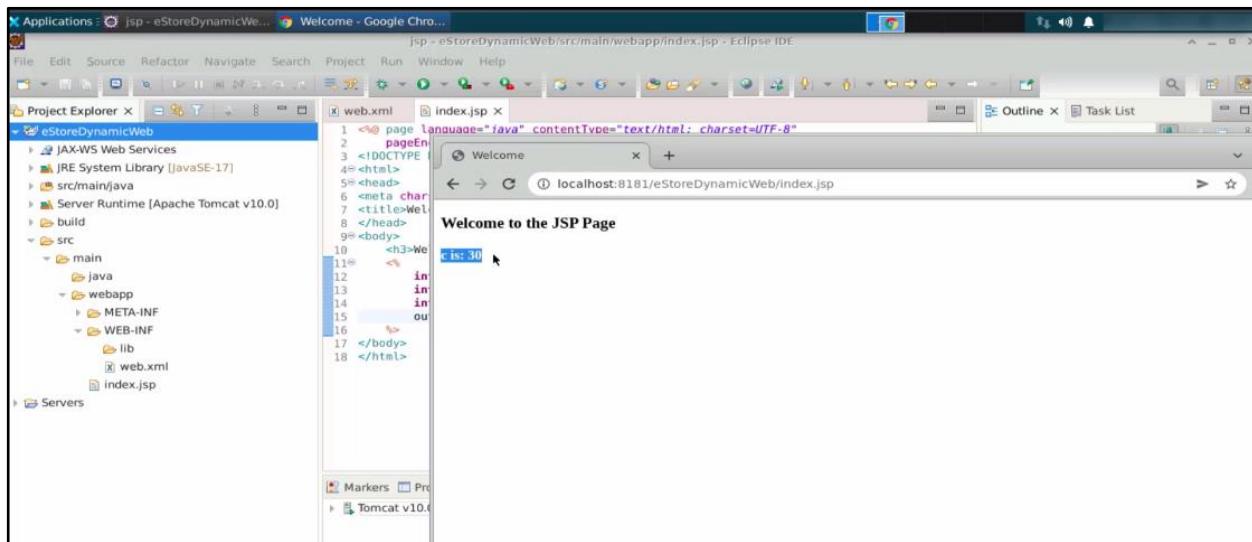


```
1 <%@ page language="java" contentType="text/html; charset=UTF-8" %>
2 <!DOCTYPE html>
3 <html>
4   <head>
5     <meta charset="UTF-8">
6     <title>Welcome</title>
7   </head>
8   <body>
9     <h3>Welcome to the JSP Page</h3>
10    <%
11      int a = 10;
12      int b = 20;
13      int c = a + b;
14      out.print("The value of c is: " + c + "</h4>") ;
15    %
16  </body>
17 </html>
18
```

3.2 Navigate back to the web browser where the JSP page is displayed and refresh the page



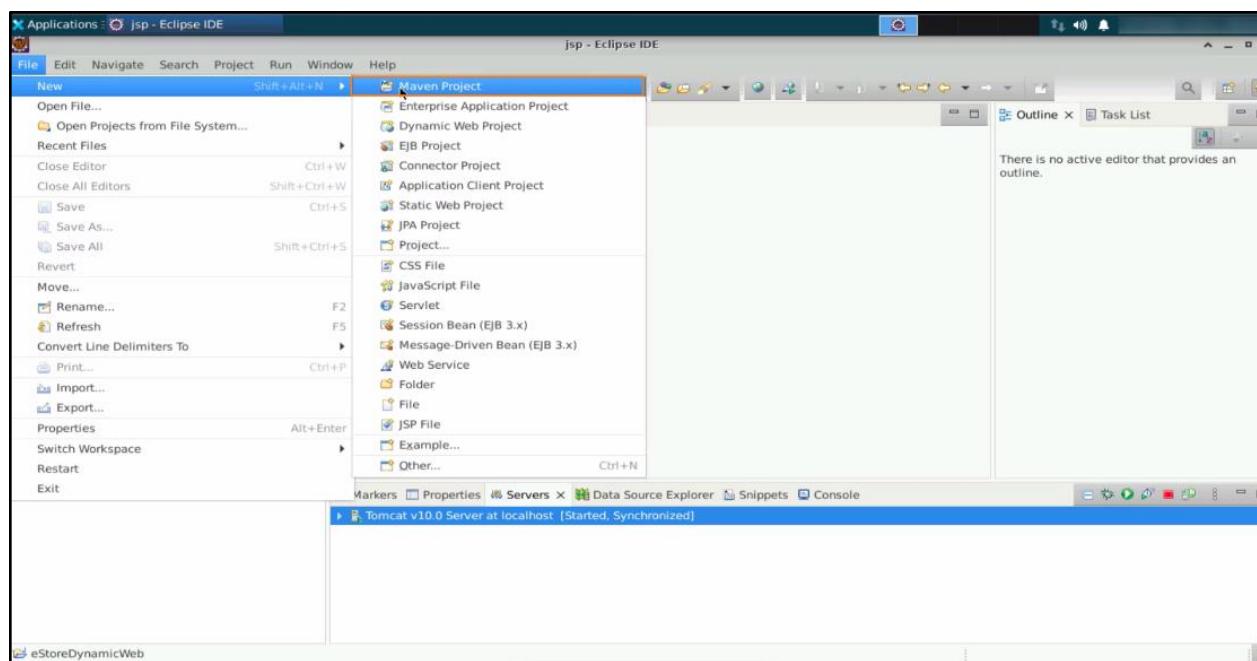
```
1 <%@ page language="java" contentType="text/html; charset=UTF-8" %>
2 <!DOCTYPE html>
3 <html>
4   <head>
5     <meta charset="UTF-8">
6     <title>Welcome</title>
7   </head>
8   <body>
9     <h3>Welcome to the JSP Page</h3>
10    <%
11      int a = 10;
12      int b = 20;
13      int c = a + b;
14      out.print("The value of c is: " + c + "</h4>") ;
15    %
16  </body>
17 </html>
18
```



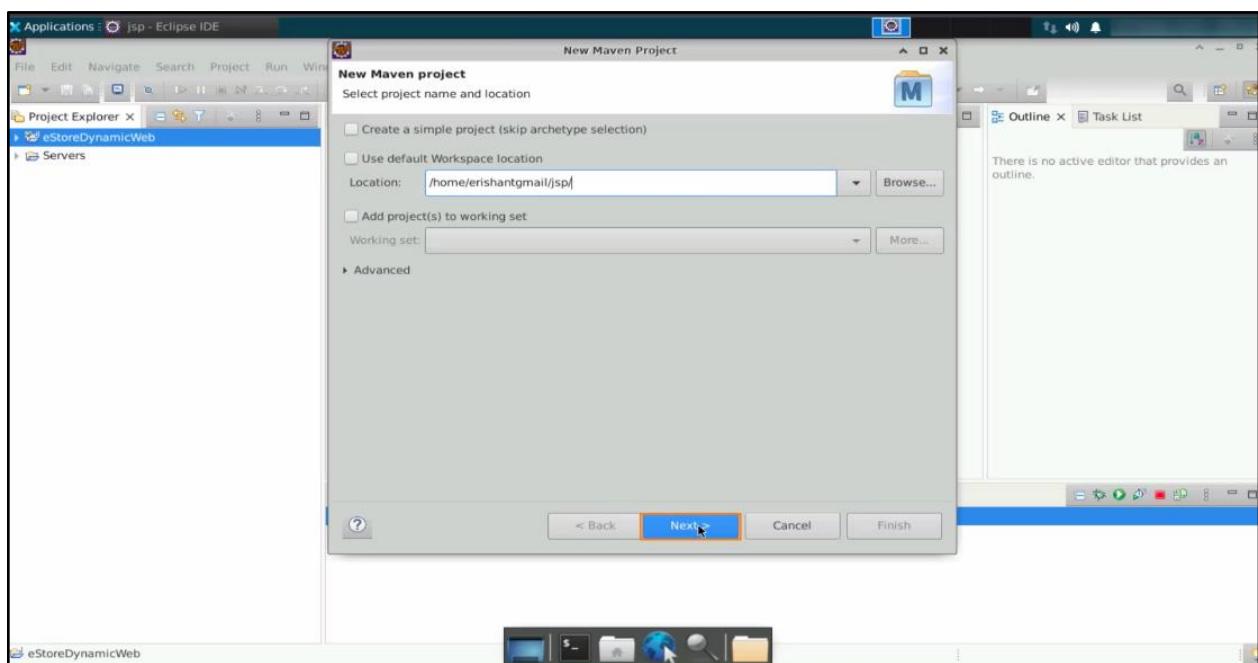
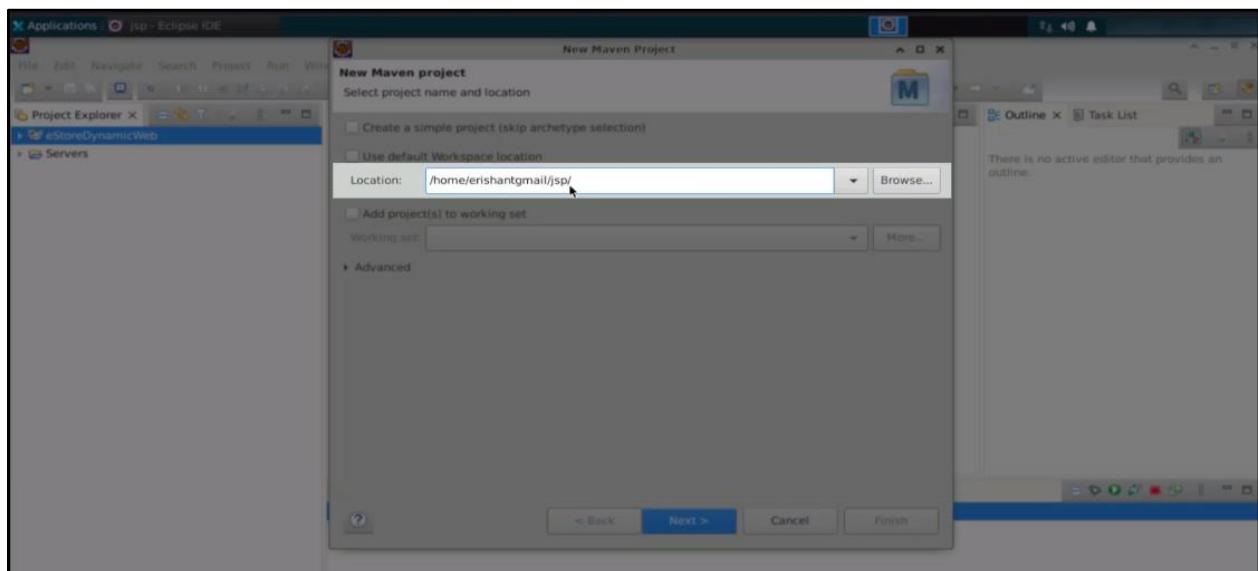
Observe the modifications made to the JSP page.

Step 4: Create a new Maven project

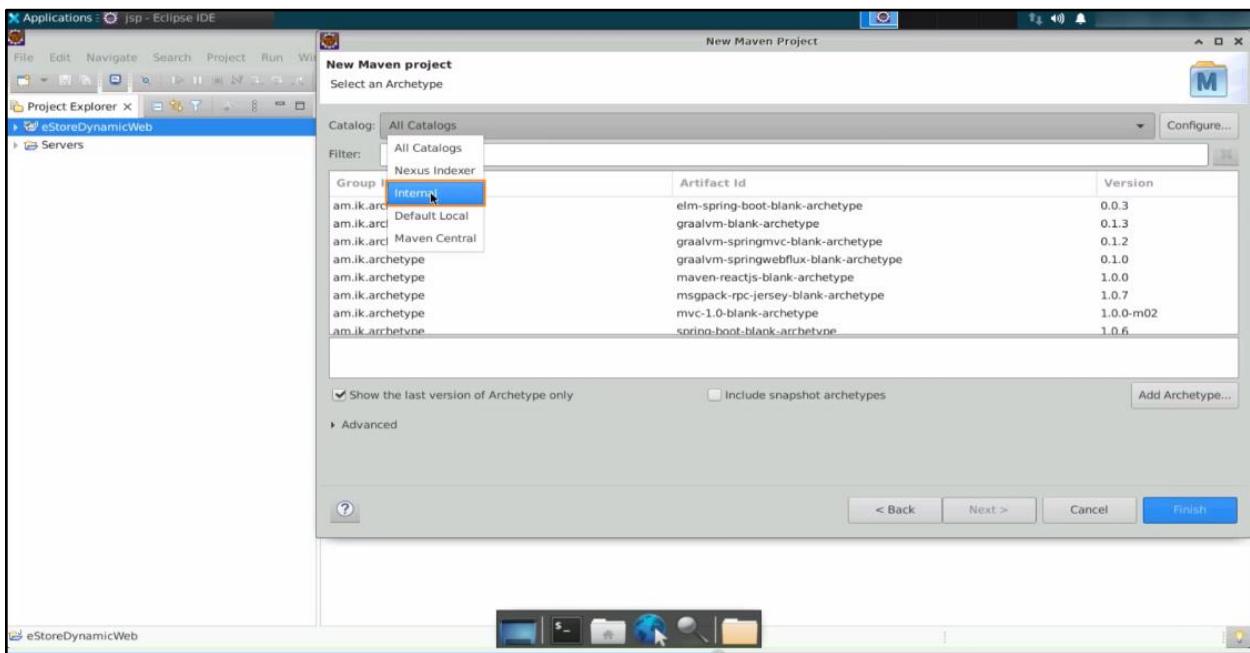
4.1 Create a new Maven project by following the path **File->New->Maven Project** in Eclipse



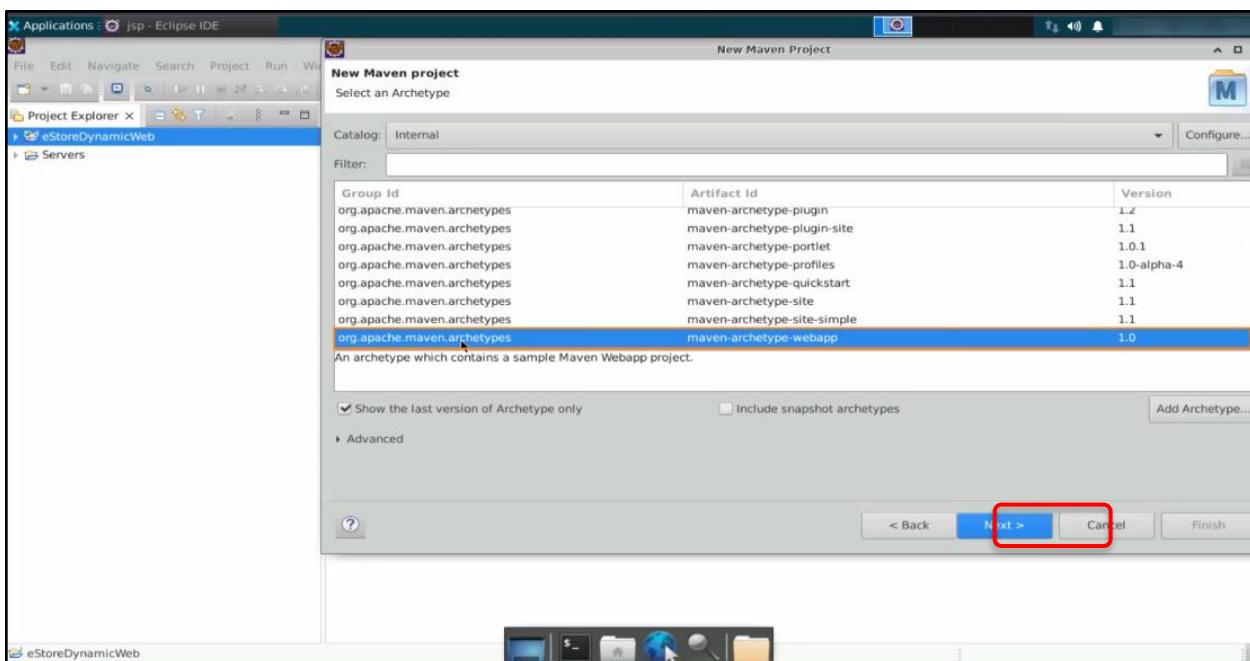
4.2 Choose a location for the project and click **Next**



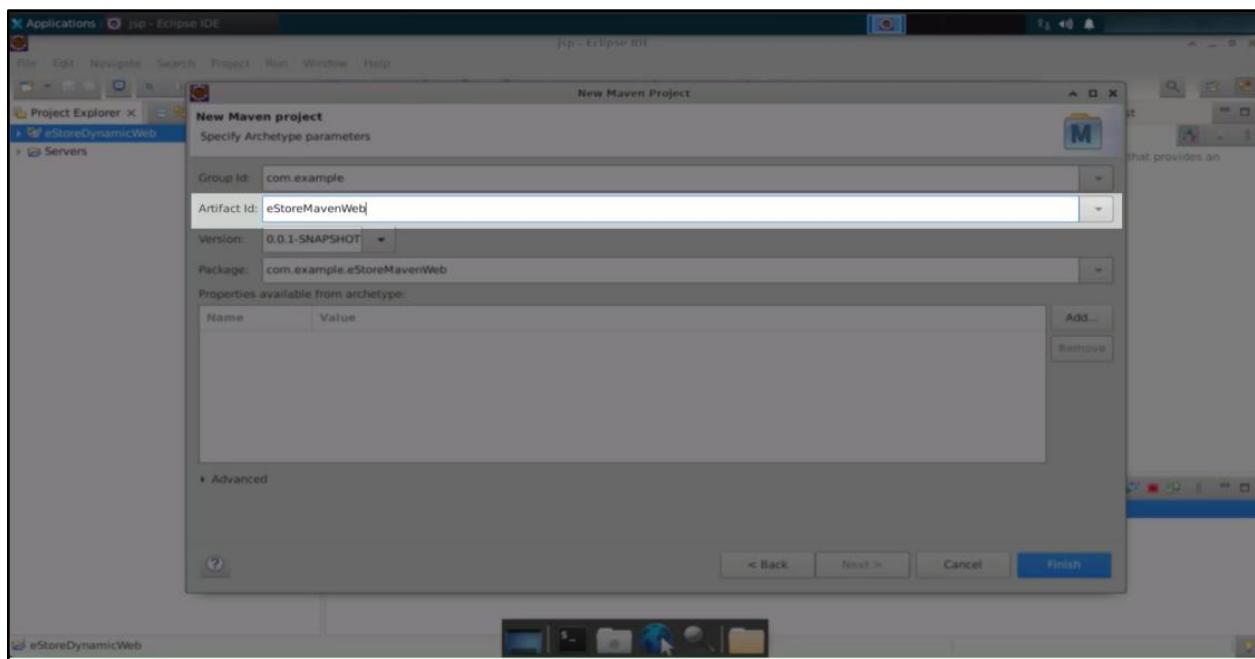
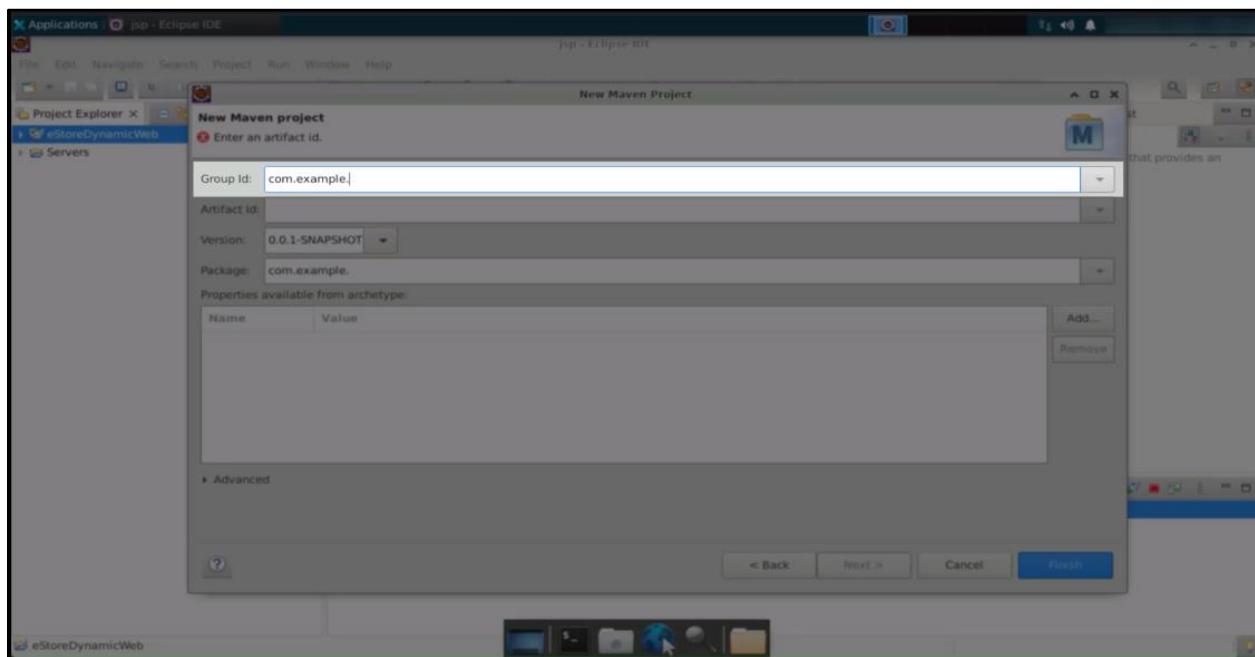
4.3 Select Internal as the Catalog and search for maven-archetype-webapp



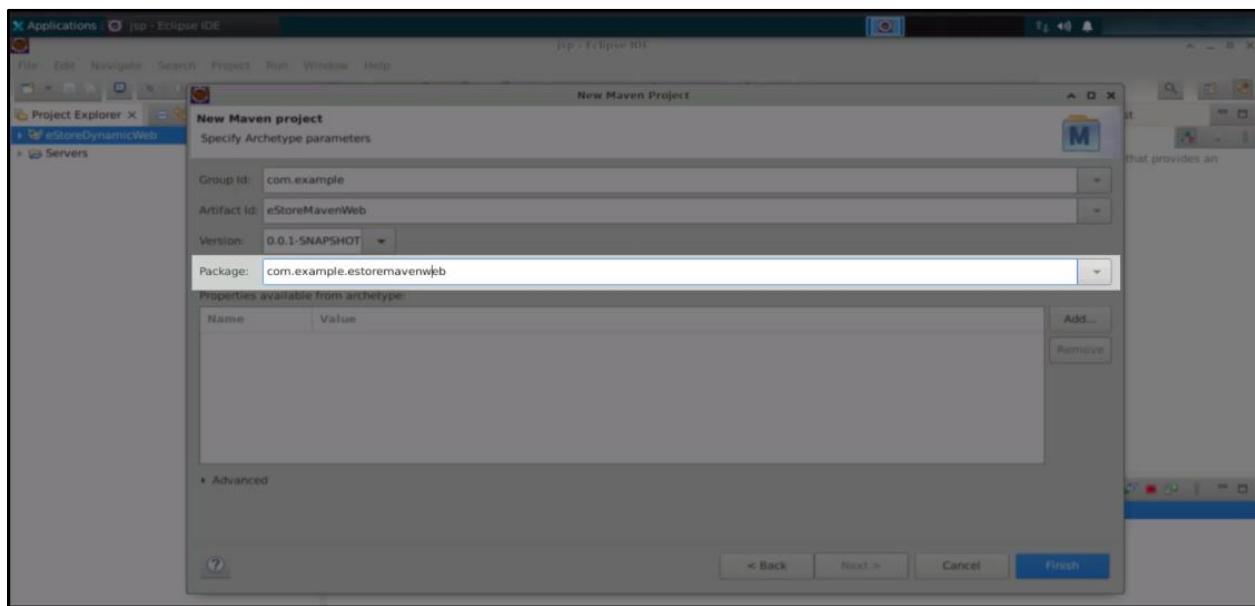
4.4 Select maven-archetype-webapp and click Next



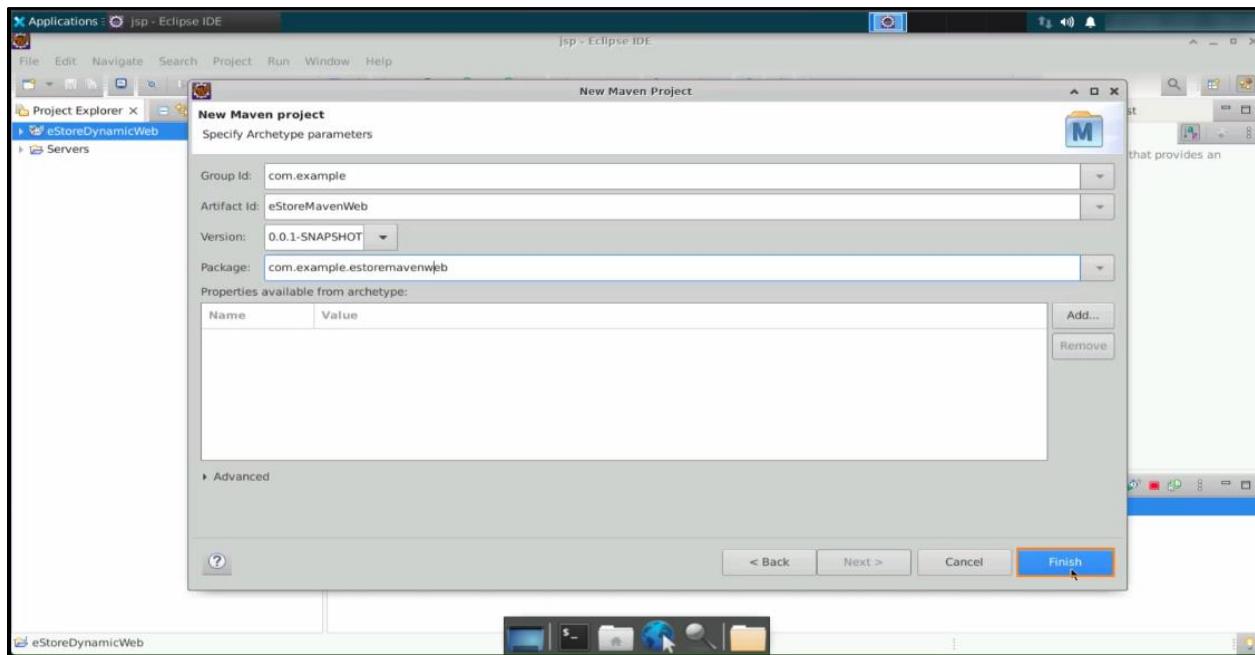
4.5 Set the Group Id as **com.example** and specify the Artifact Id as **eStoreMavenWeb**



The package name will be automatically generated.

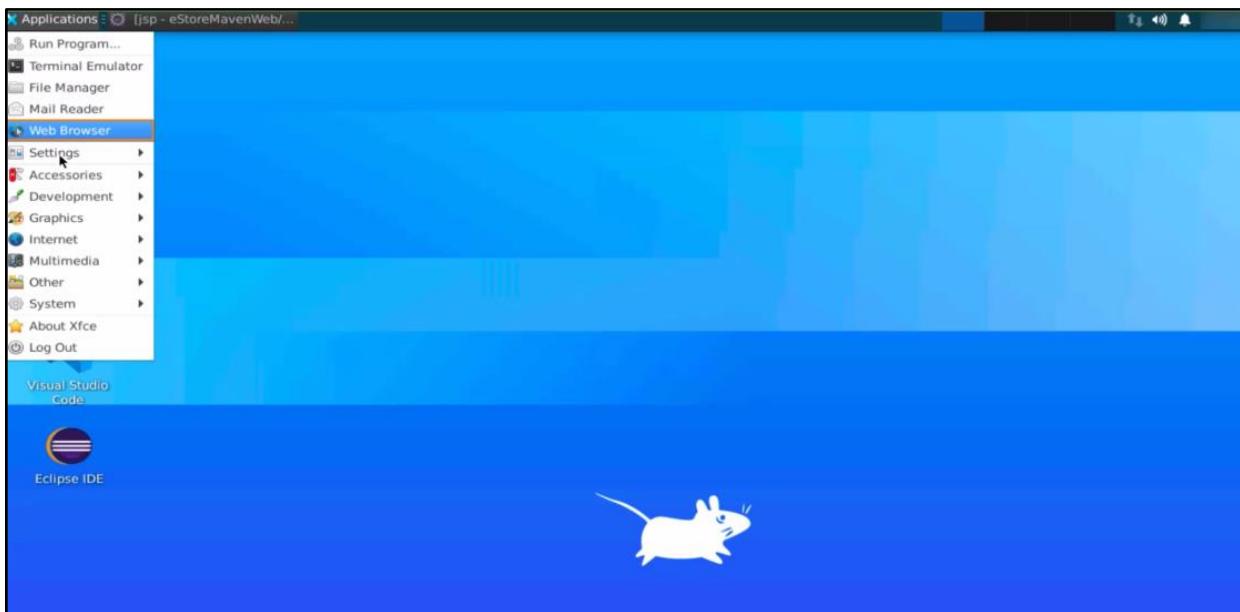


4.6 Click **Finish** to create the Maven project



Step 5: Add the dependencies

5.1 Launch a Web Browser



5.2 Search for mvnrepository and click on the search result for mvnrepository.com

A screenshot of a Google search results page. The search query 'mvnrepository' is entered in the search bar. The top result is 'https://mvnrepository.com' with the title 'Maven Repository: Search/Browse/Explore'. This result is highlighted with a yellow box. Below it, there are other search results for 'Central', 'Selenium', 'Junit', 'Mysql', and 'Indexed Artifacts (25.3M)'. At the bottom of the page, there is a 'People also ask' section.

5.3 Search for servlet in the search bar on the website

The screenshot shows the Maven Repository search results for the term 'servlet'. The search bar at the top contains 'servlet'. Below it, the results are listed under the heading 'What's New in Maven'. There are five entries:

- Tools**: io.github.shashizhuze - tools - 1.9.7 (12 usages, Apache)
- Microcks Model**: io.github.microcks - microcks-model - 1.5.0 (4 usages)
- Microcks Model**: io.github.microcks - microcks-model - 1.5.0-RC1 (4 usages)
- Microcks Model**: io.github.microcks - microcks-model - 1.5.0-RC2 (4 usages)
- JDBC**: io.github.shashizhuze - jtds - 2.5 (3 usages, Apache)

On the left sidebar, there are sections for 'Indexed Artifacts (25.3M)', 'Popular Categories' (Aspect Oriented, Actor Frameworks, Application Metrics, Build Tools, Bytecode Libraries, Command Line Parsers, Cache Implementations, Cloud Computing, Code Analyzers, Collections, Configuration Libraries, Core Utilities, Date and Time Utilities, Dependency Injection, Embedded SQL Databases, HTML Parsers, HTTP Clients, IO Utilities), and 'Indexed Repositories (1252)' (Central, Sonatype, Atlassian, Hortonworks, Spring Plugins, Spring Lib M, JCenter, Atlassian Public, JBossEA, BeDataDriven). On the right sidebar, there is a 'Popular Tags' section.

5.4 Open the Java Servlet API page from the search results

The screenshot shows the Maven Repository search results for the term 'servlet'. The search bar at the top contains 'servlet'. Below it, the results are listed under the heading 'Found 2617 results'. The results are sorted by relevance. The first result is highlighted:

- Java Servlet API**: javax.servlet - javax.servlet-api (15,528 usages, GPL | CDDL)

Below the highlighted result, there are four more results:

- JavaServer(TM) Specification**: javax.servlet - servlet-api (11,424 usages, GPL | CDDL)
- Jetty :: Servlet Handling**: org.eclipse.jetty - jetty-servlet (2,195 usages, EPL | Apache)
- Java Servlet 3.1 API**: org.jboss.spec.javaee - jboss-servlet-api_3_1_spec (494 usages, GPL)
- GWT Servlet**: com.google.gwt - gwt-servlet (679 usages)

On the left sidebar, there are sections for 'Repository' (Central, Sonatype, Spring Plugins, Spring Lib M, JCenter, Atlassian Public, JBossEA, BeDataDriven) and 'Category' (Web App, Maven Archetype, Java Spec, Maven Plugins, JSP Tag Library). On the right sidebar, there is a 'Popular Tags' section.

5.5 Select version 4.0.1 of the Java Servlet API

| Version | Vulnerabilities | Repository | Usages | Date |
|-----------|-----------------|------------|--------|-----------|
| 4.0.1 | 3 | Central | 3,213 | Apr, 2018 |
| 4.0.0 | 410 | Central | 410 | Aug, 2017 |
| 4.0.0-b07 | 22 | Central | 22 | Jun, 2017 |
| 4.0.0-b06 | 0 | Central | 0 | May, 2017 |
| 4.0.0-b05 | 4 | Central | 4 | Mar, 2017 |
| 4.0.0-b04 | 0 | Central | 0 | Mar, 2017 |
| 4.0.0-b03 | 2 | Central | 2 | Mar, 2017 |
| 4.0.0-b02 | 3 | Central | 3 | Feb, 2017 |
| 4.0.0-b01 | 26 | Central | 26 | Oct, 2015 |
| 3.1.0 | 8,855 | Central | 8,855 | Apr, 2013 |
| 3.1-b09 | 1 | Central | 1 | Apr, 2013 |
| 3.1-b08 | 3 | Central | 3 | Apr, 2013 |

5.6 Copy the Maven dependency information provided on the page

```

<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>4.0.1</version>
    <scope>provided</scope>
</dependency>

```

Copied to clipboard!

Return to the Eclipse IDE and open the **pom.xml** file in the Maven project

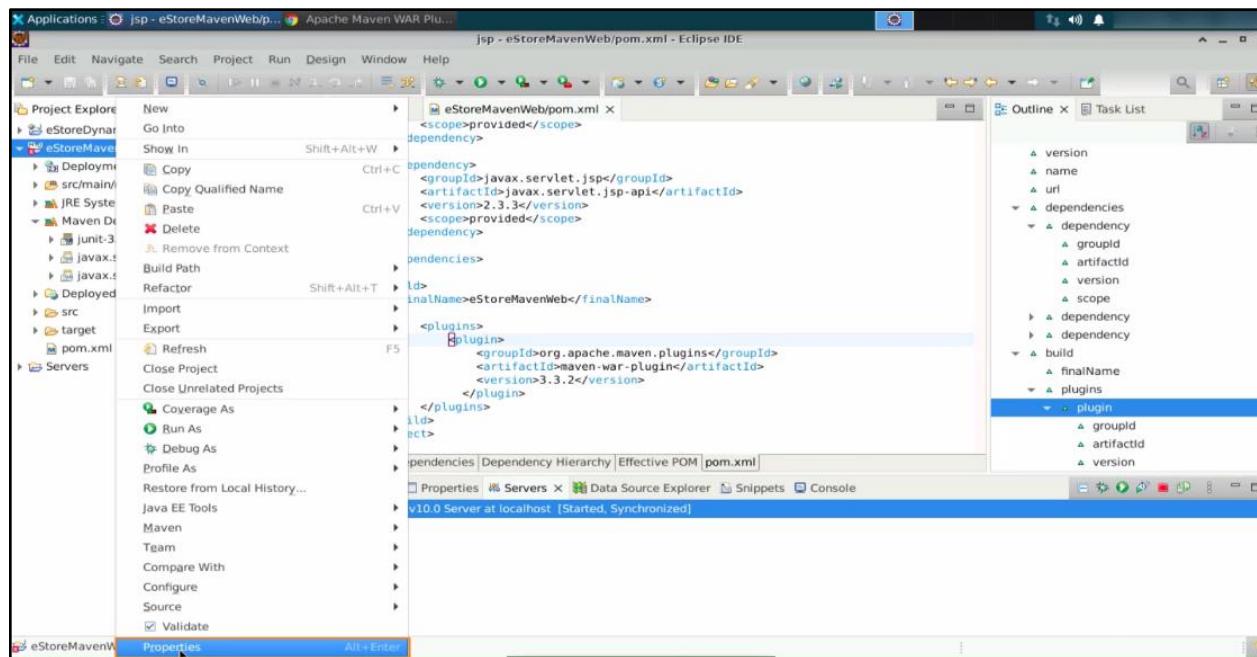
5.7 Paste the copied dependency information into the **pom.xml** file to include the Java Servlet API dependency

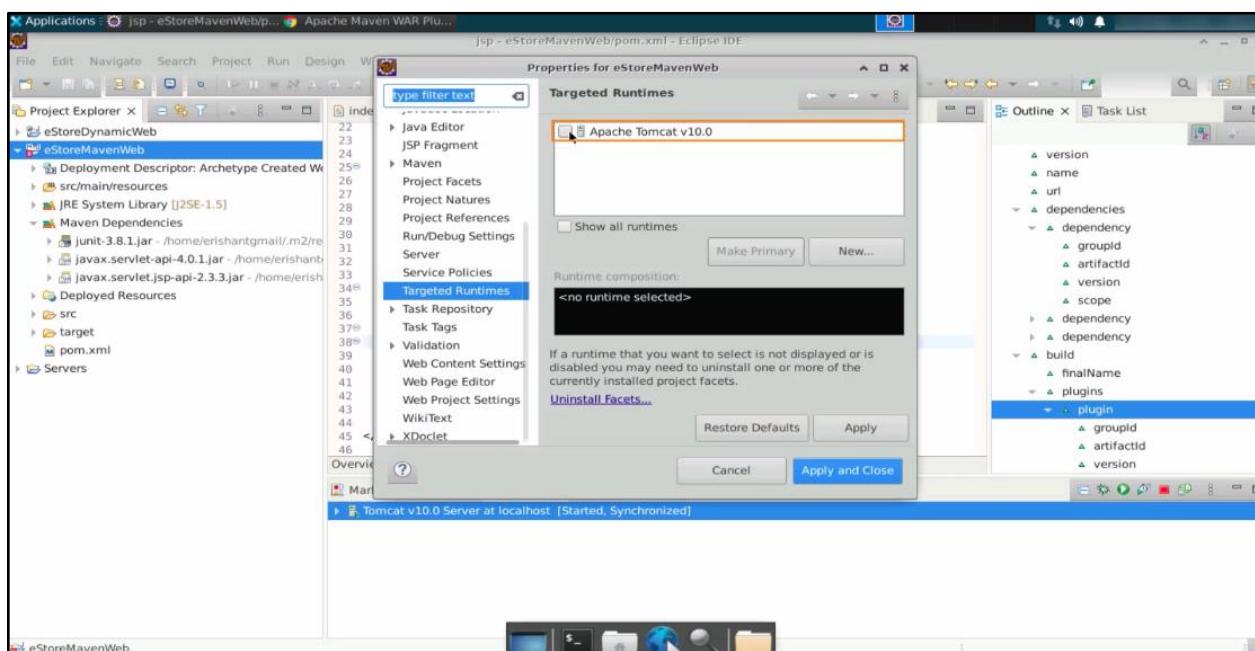
```

<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>4.0.1</version>
    <scope>provided</scope>
</dependency>

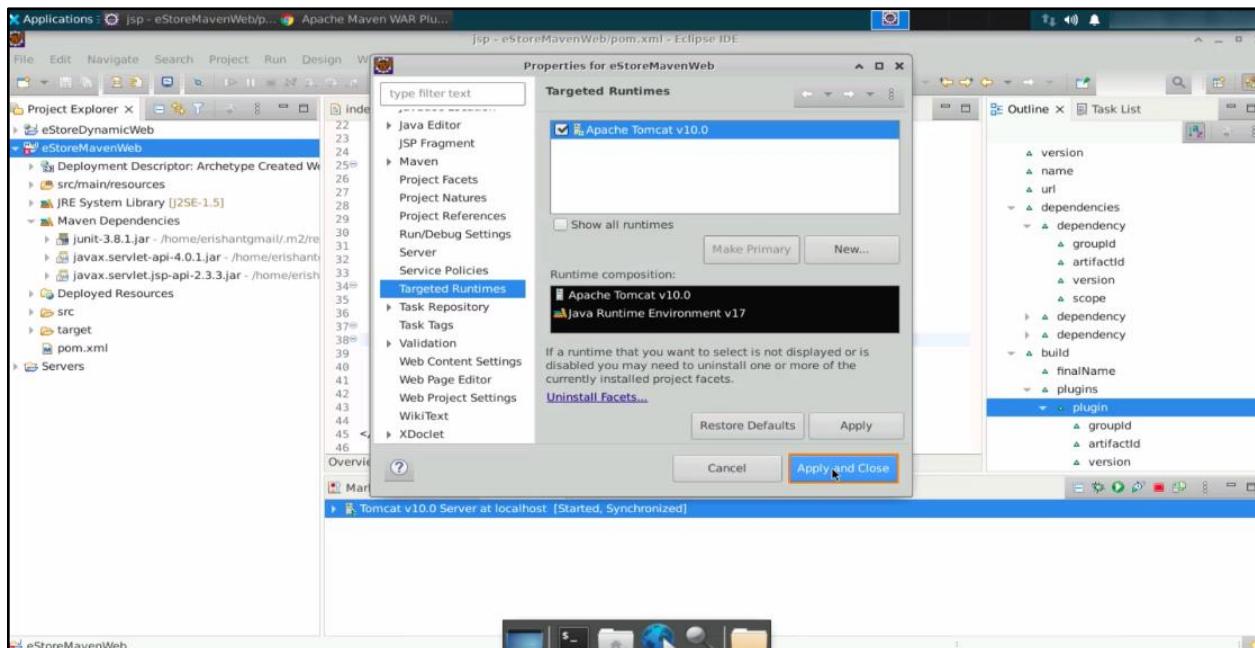
```

5.8 Go to the project properties by right-clicking on the project, selecting **Properties**, and choosing **Targeted Runtimes**



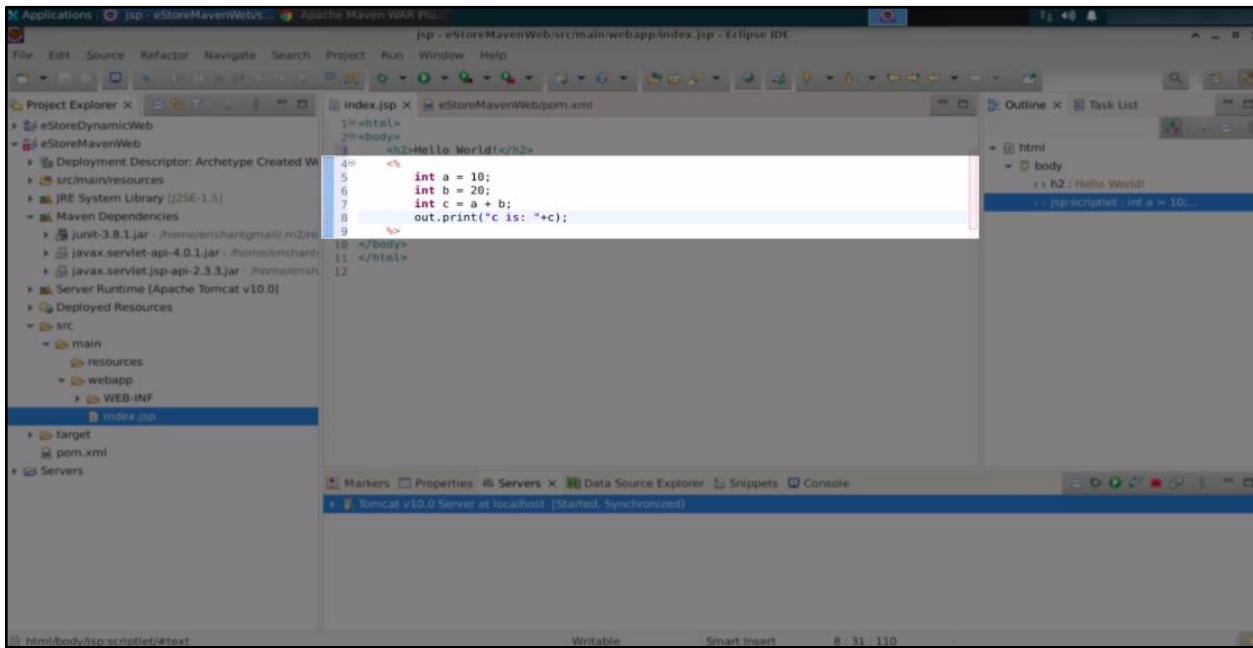


5.9 Check the box for the Apache Tomcat runtime and click **Apply and Close** to configure the targeted runtime



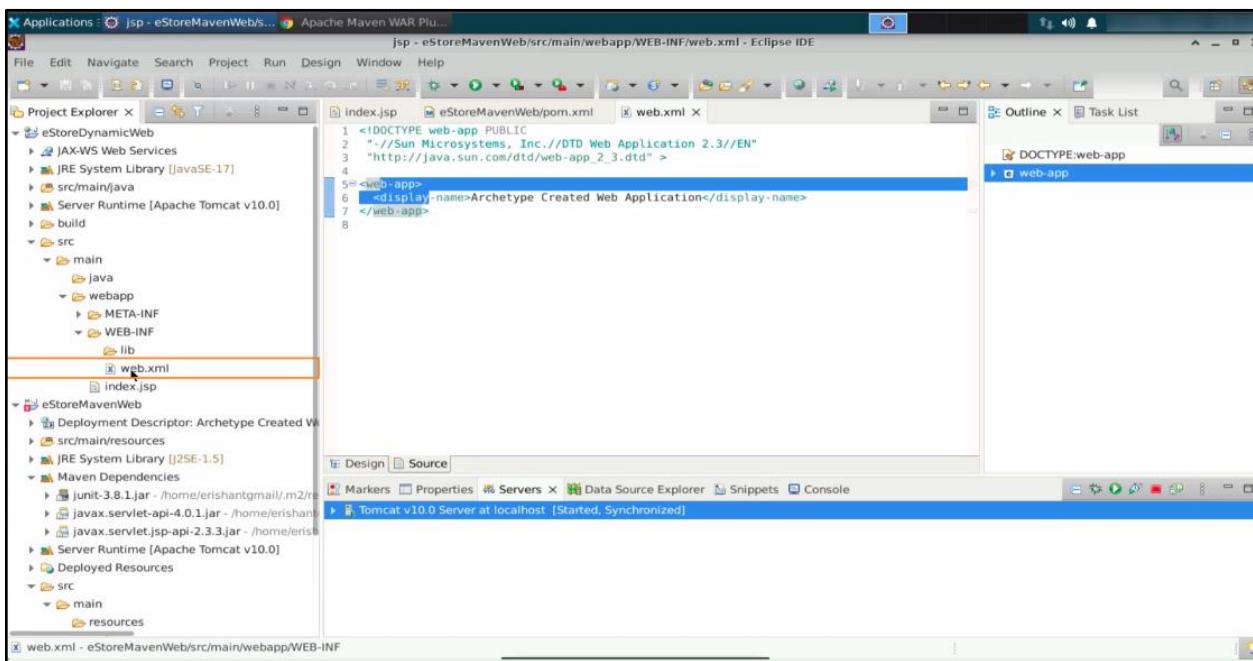
Step 6: Create a scriptlet tag

6.1 Go to the **index.jsp** page and create a scriptlet tag `<% %>` to add server-side logic or code

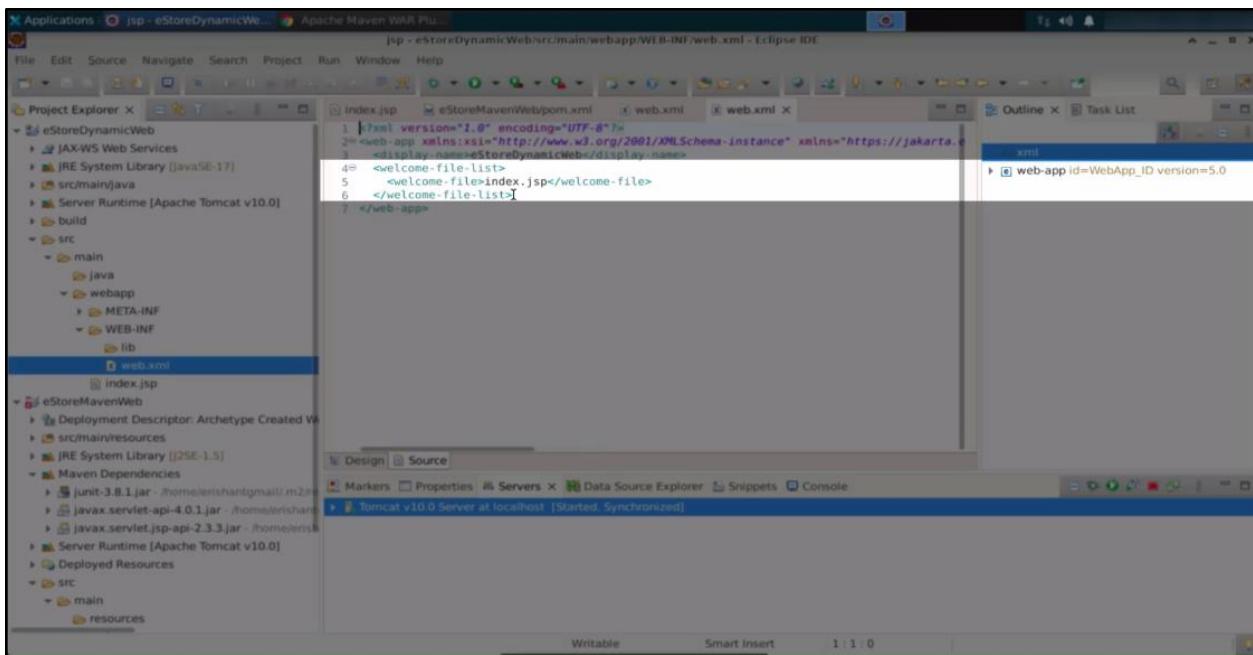


```
1<html>
2<body>
3<h2>Hello World!</h2>
4<% int a = 10;
5    int b = 20;
6    int c = a + b;
7    out.print("c is: "+c);
8%>
9</body>
10</html>
11
12
```

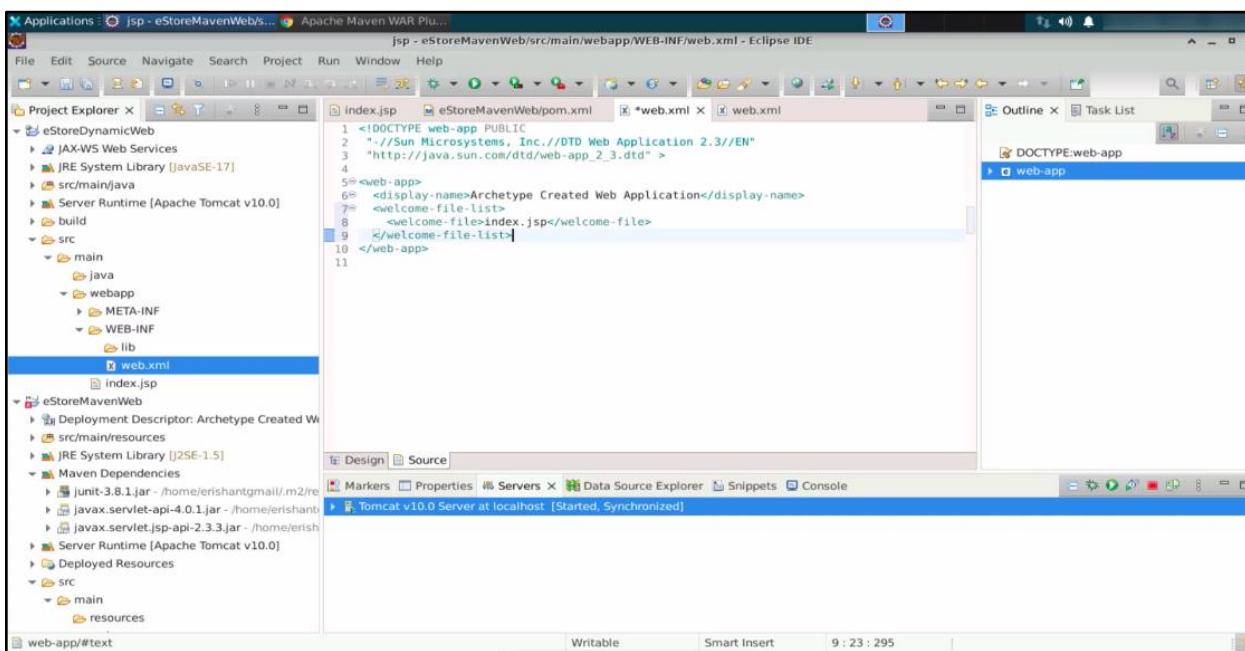
6.2 Open the **web.xml** file located in the **WEB-INF** directory



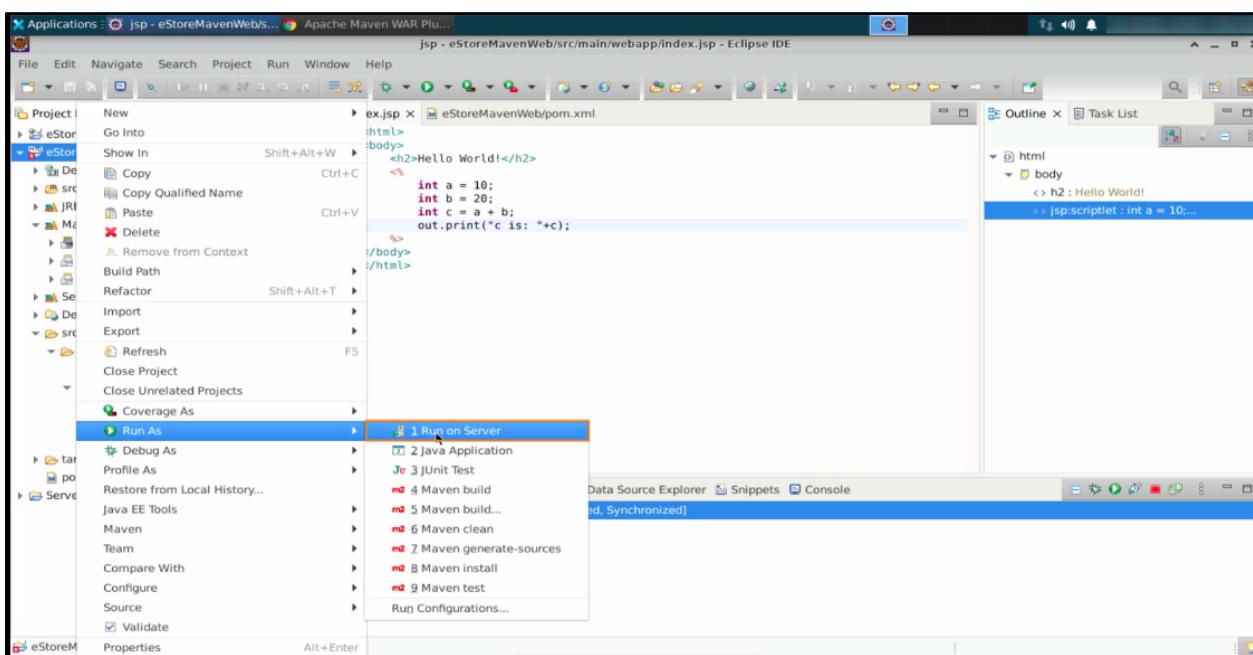
6.3 Copy the necessary code or configuration from the documentation or example



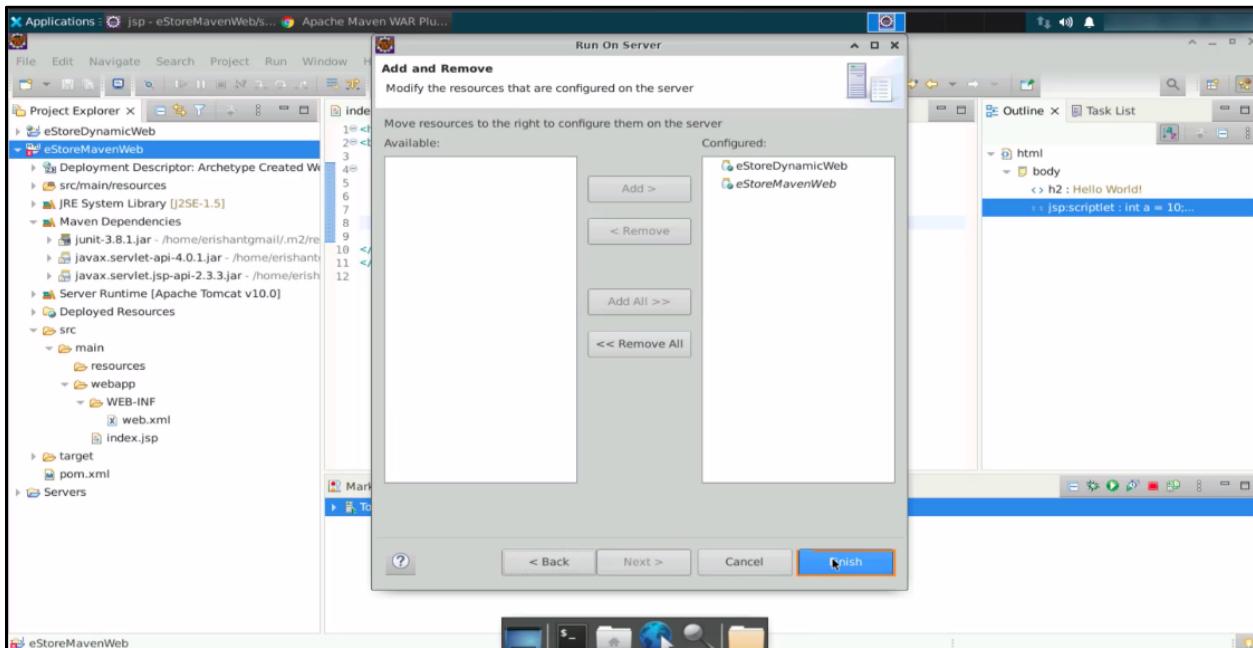
6.4 Paste the copied code or configuration into the web.xml file and save it



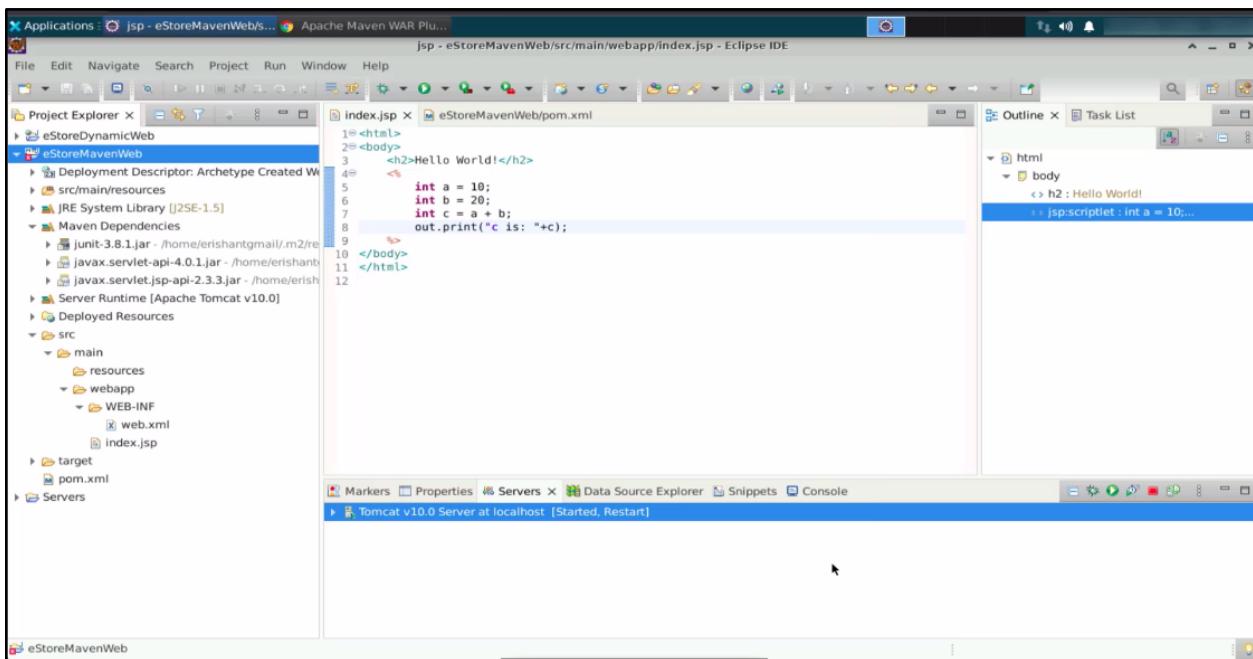
6.5 Right-click on the project, select Run As, and click Run on Server to run the project again



6.6 Choose the configured Tomcat server and click **Finish** to run the project on the server



Observe the changes made and displayed in the web browser.



Following these steps, you have successfully created a JSP (Java Server Pages) page using Eclipse IDE.