

## Lesson 03 Demo 02

### Working with Controller and View Resolver

**Objective:** To demonstrate the implementation of Controllers and View Resolvers in a Spring MVC project

**Tool required:** Eclipse IDE

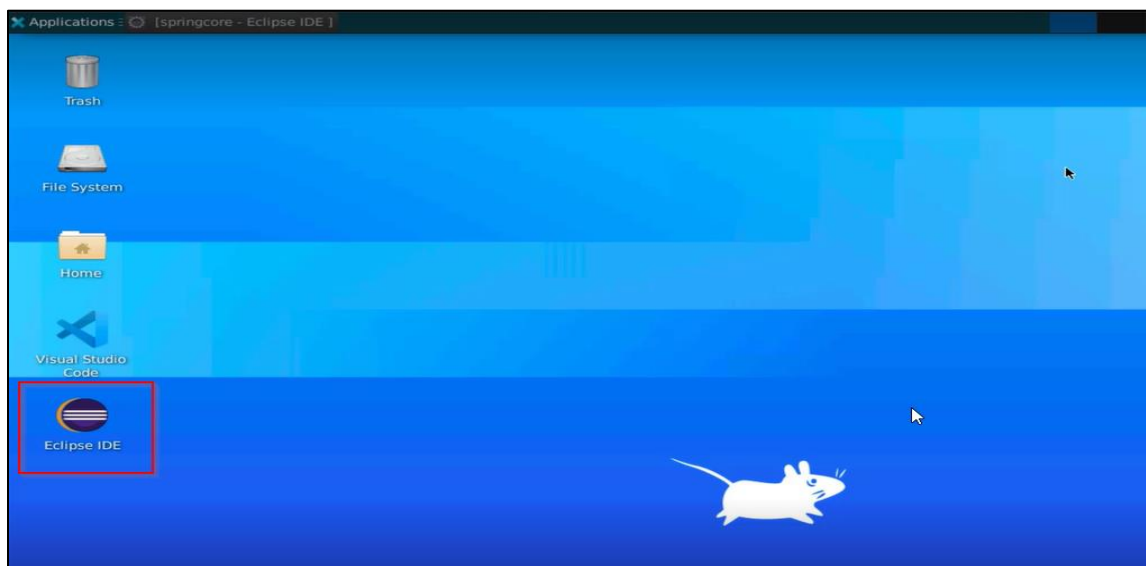
**Prerequisites:** None

#### Steps to be followed:

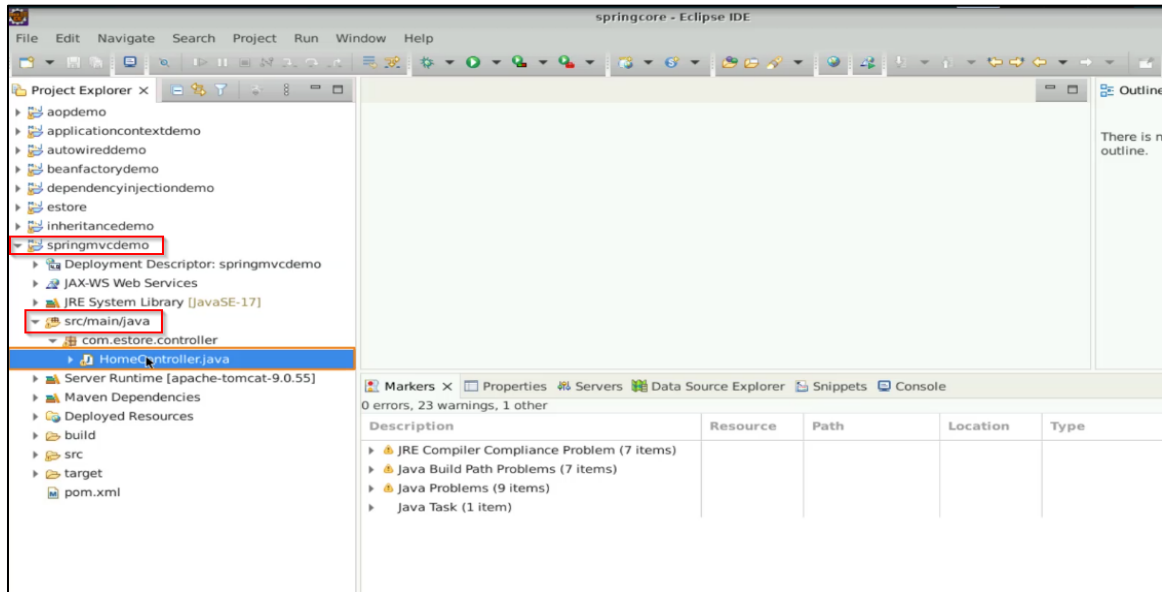
1. Adding Request mapping in HomeController
2. Creating a Bean in my-spring-web MVC
3. Creating a base-package
4. Creating a new JSP file in views

#### Step 1: Adding Request mapping in HomeController

##### 1.1 Open Eclipse IDE

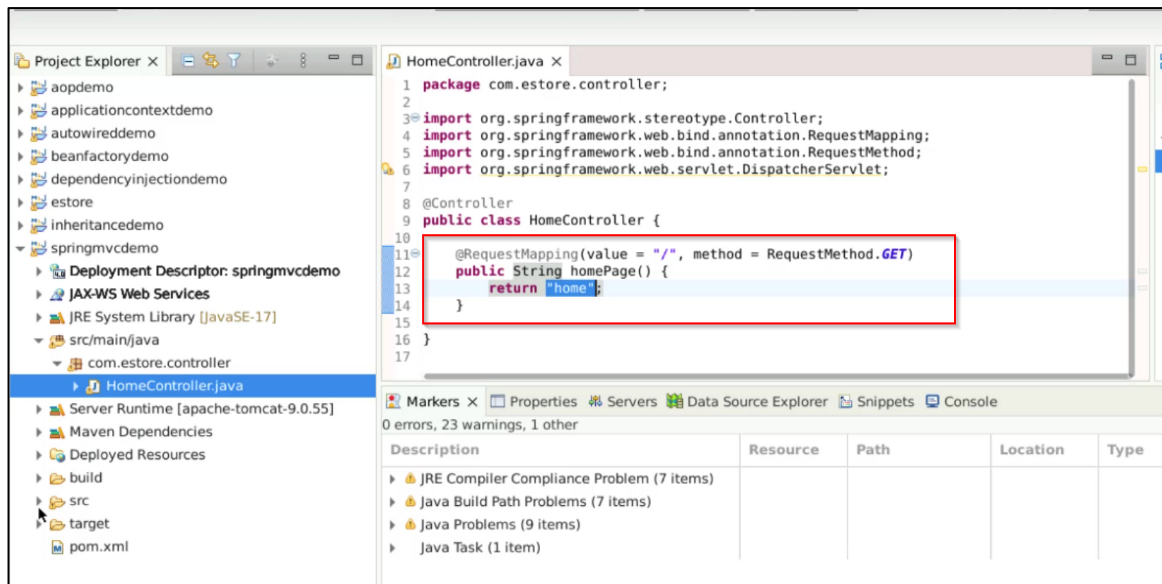


1.2 In the **springmvcdemo** project, navigate to the class **HomeController.java** under the **src/main/java** package



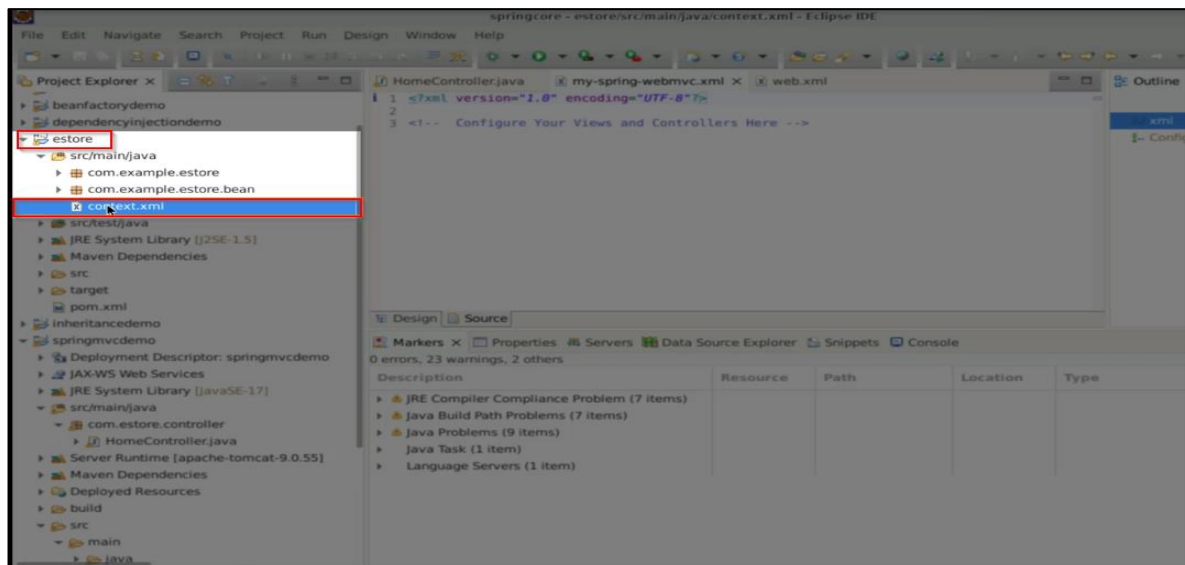
**Note:** Please refer to the previous demo on how to create a Spring MVC project

- 1.3 Create a method named **homePage** that returns a string **Home** and add a **@RequestMapping** annotation to define the endpoint as **/** with the request method set as **GET**



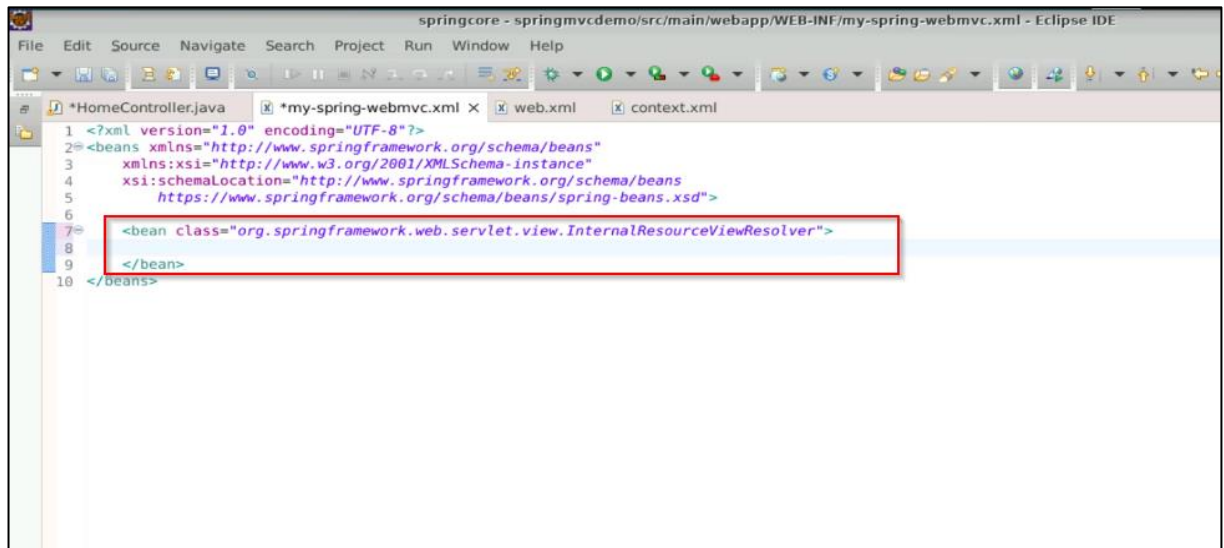
## Step 2: Creating a bean in my-spring-web MVC

- 2.1 Copy the **context.xml** file from the **estore** project to the **springmvcdemo** project

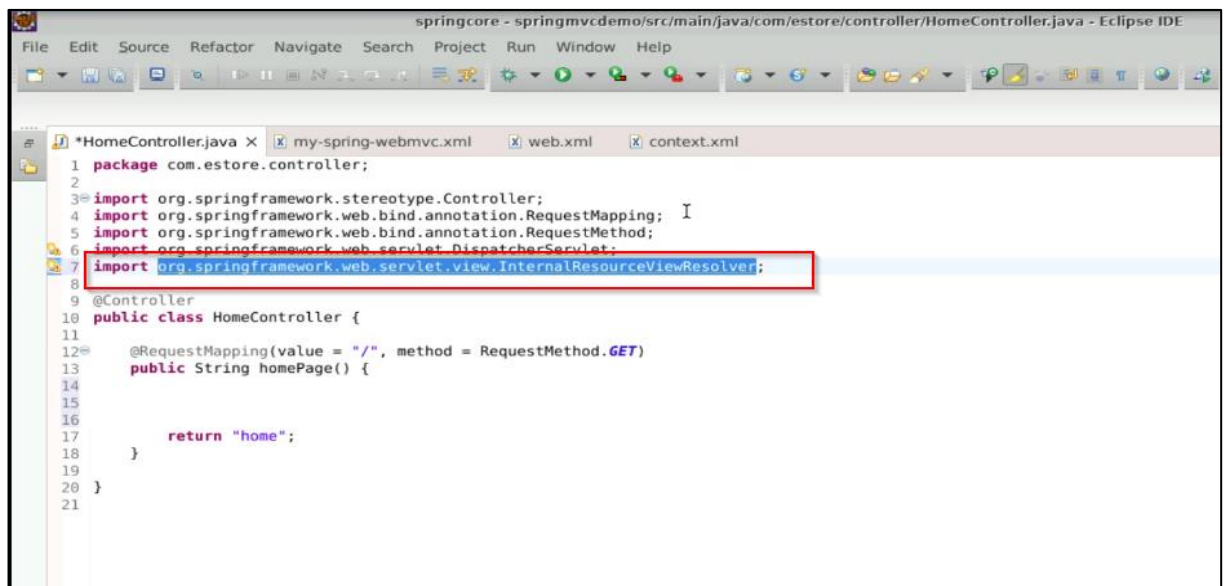


**Note:** Please refer to the previous demos on how to create an **estore** project

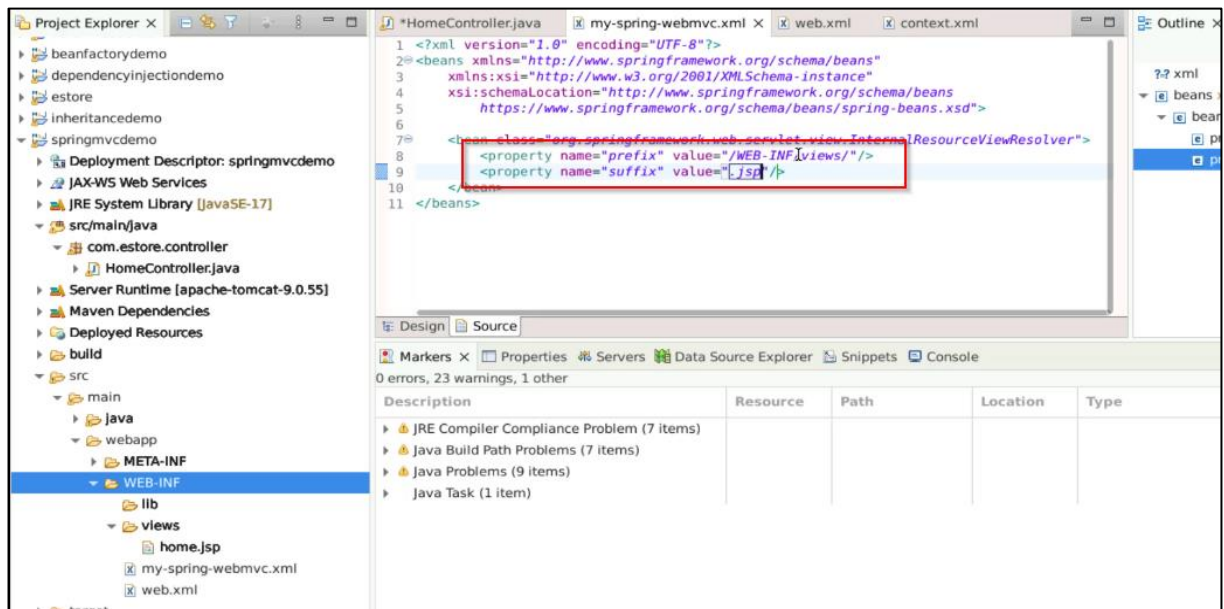
## 2.2 Remove all the existing beans and add a new bean called **InternalResourceViewResolver** in **my-spring-webmvc.xml**



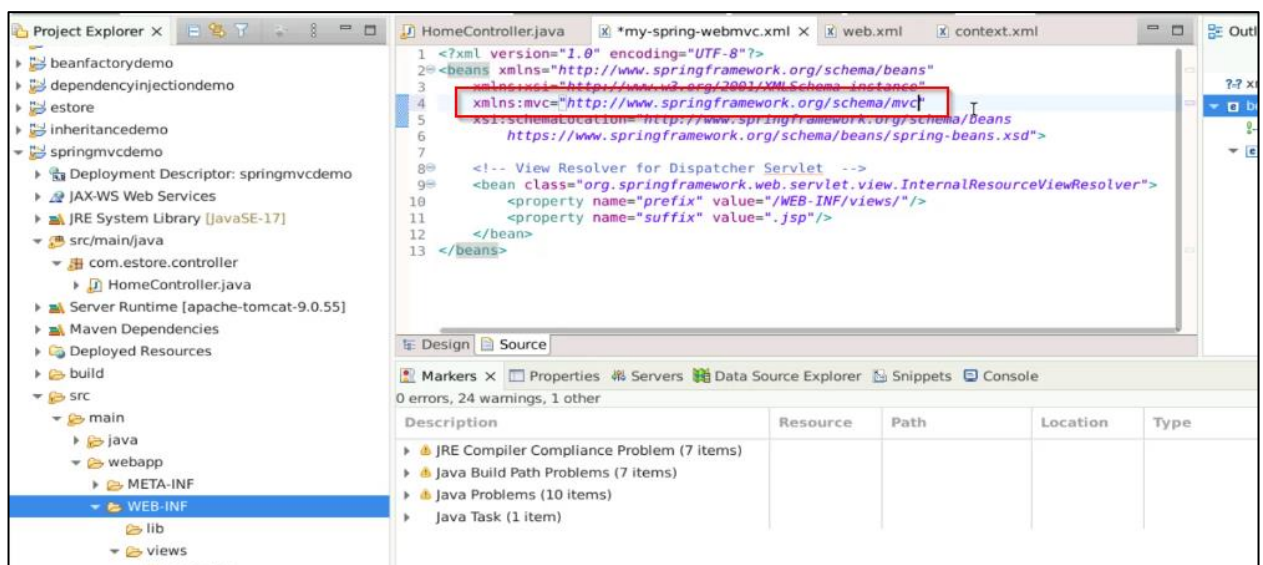
## 2.3 Navigate to **HomeController.java** and add an import statement for **InternalResourceViewResolver**

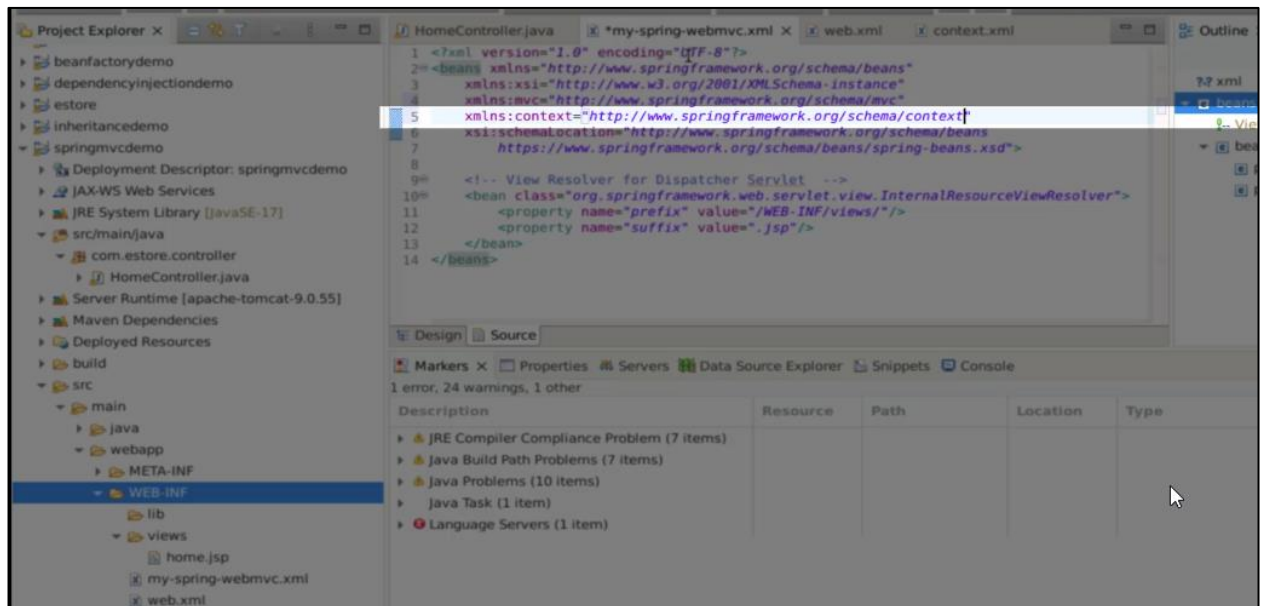


2.4 Set the **prefix** property as **/WEB-INF/views/** and the **suffix** property as **.jsp** in the **InternalResourceViewResolver** bean. This sets up the View Resolver for the Dispatcher Servlet



2.5 Add the **xmlns:mvc** and **xmlns:context** namespaces in **my-spring-webmvc.xml** to enable the web application to search for the context





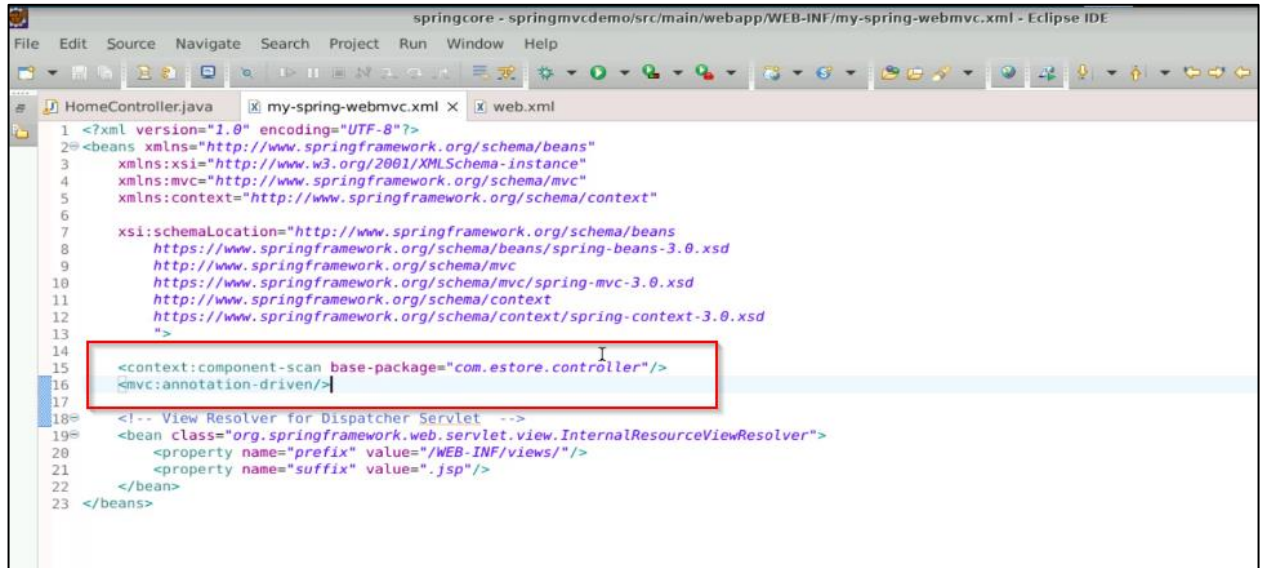
## 2.6 Add the `xsi:schemaLocation` attribute for the Spring Framework modifications and `schema locations`





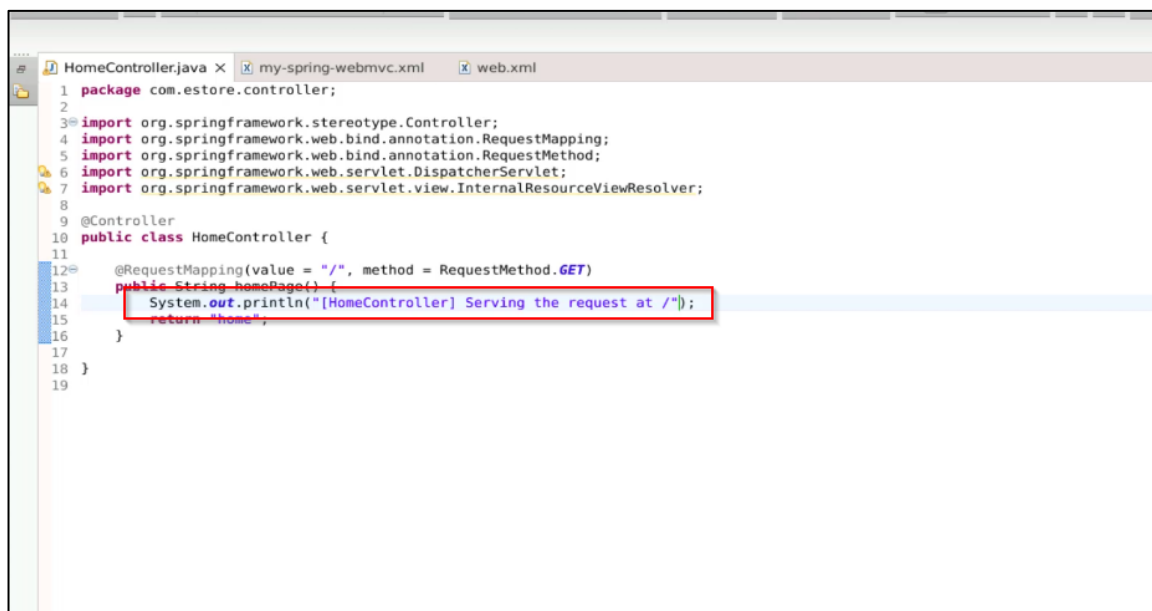
## Step 3: Creating a base-package

- 3.1 Create a `<context:component-scan>` tag and specify the **base-package** as **com.estore.controller** for the **HomeController.java** class. Additionally, include the `<mvc:annotation-driven/>` tag



```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
                           http://www.springframework.org/schema/mvc
                           http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd
                           http://www.springframework.org/schema/context
                           http://www.springframework.org/schema/context/spring-context-3.0.xsd"
>
    <context:component-scan base-package="com.estore.controller"/>
    <mvc:annotation-driven/>
    <!-- View Resolver for Dispatcher Servlet -->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/views/" />
        <property name="suffix" value=".jsp" />
    </bean>
</beans>
```

- 3.2 In **HomeController.java**, add a request mapping and a **System.out.println** statement to handle the request at the root/endpoint



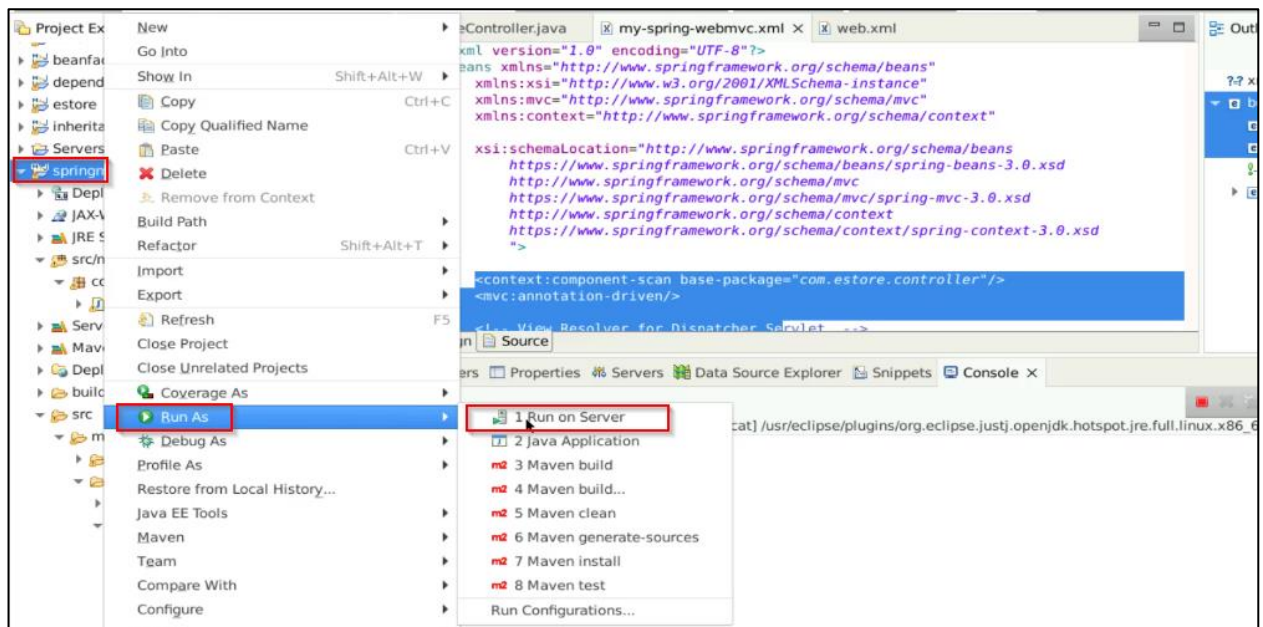
```
package com.estore.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.DispatcherServlet;
import org.springframework.web.servlet.view.InternalResourceViewResolver;

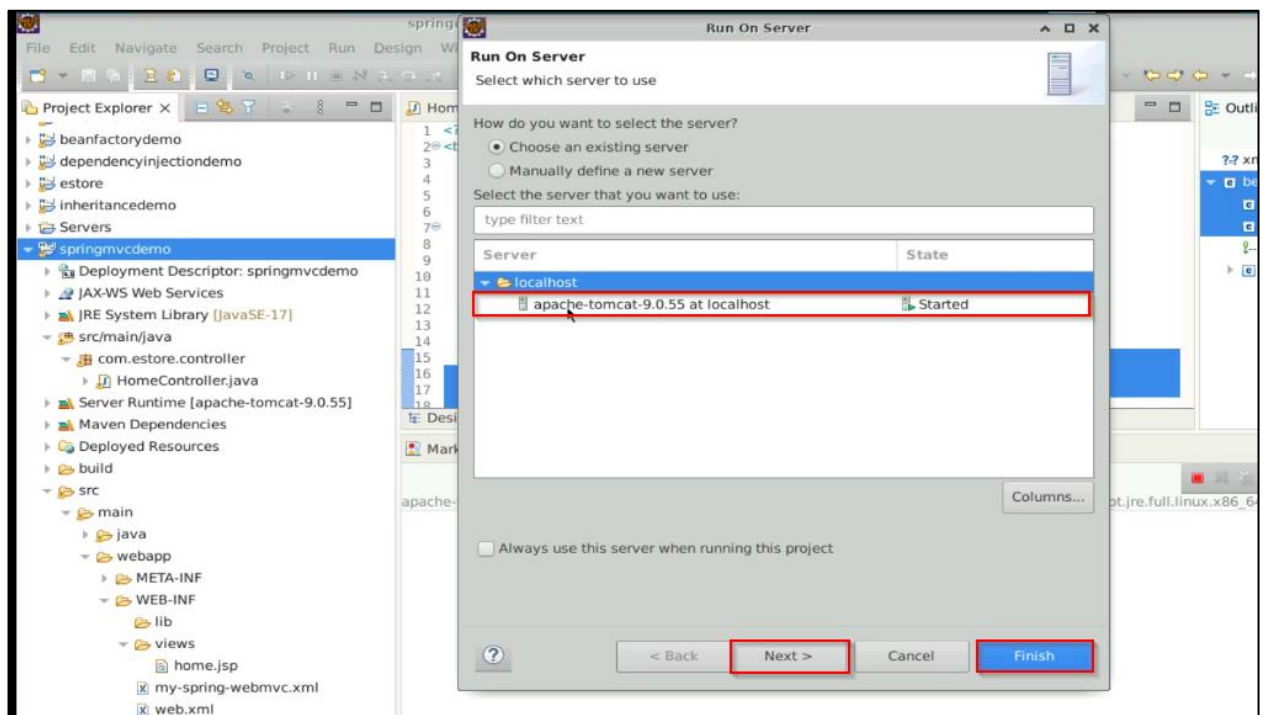
@Controller
public class HomeController {

    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String homePage() {
        System.out.println("[HomeController] Serving the request at /");
        return "home";
    }
}
```

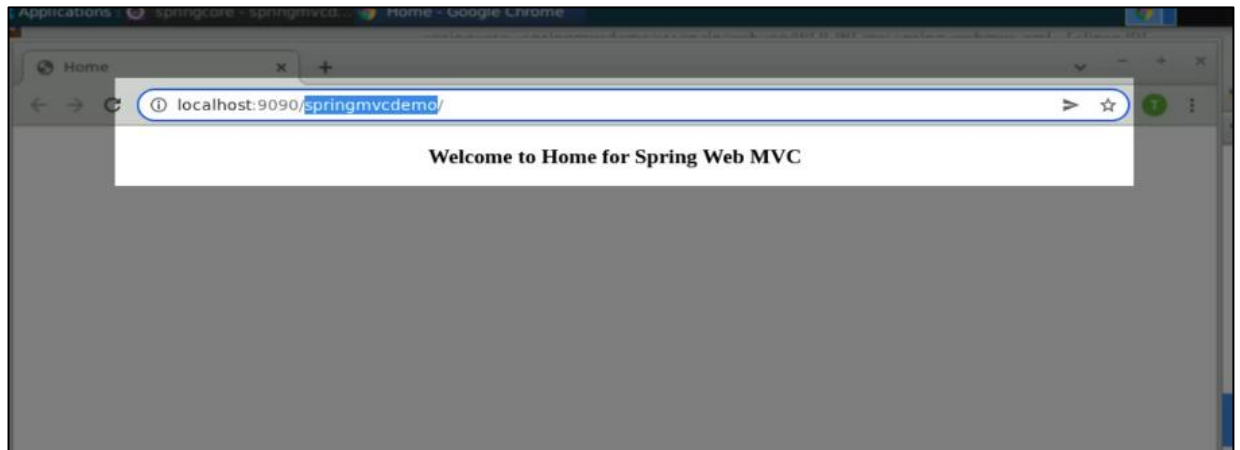
### 3.3 Select **Run As > Run on Server** to run the application on a server



### 3.4 Choose the Apache Tomcat server, click **Next**, and **Finish** to run the application on the browser

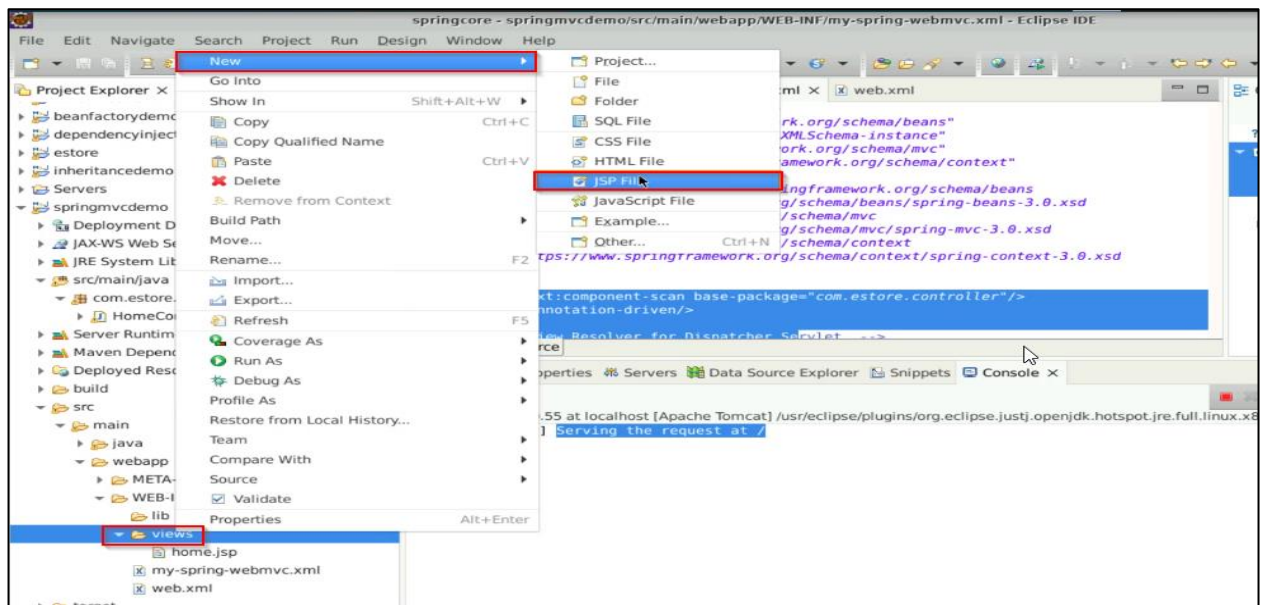






## Step 4: Creating a new JSP file in views

4.1 Right-click on **views** folder and select **New > JSP File** to create a new JSP file



[illegible]

4.4 Create a new method called **welcomePage()** in **HomeController.java** and add a **System.out.println** statement

```

1 package com.estore.controller;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.bind.annotation.RequestMethod;
6 import org.springframework.web.servlet.DispatcherServlet;
7 import org.springframework.web.servlet.view.InternalResourceViewResolver;
8
9 @Controller
10 public class HomeController {
11
12     @RequestMapping(value = "/", method = RequestMethod.GET)
13     public String homePage() {
14         System.out.println("[HomeController] Serving the request at /");
15         return "home";
16     }
17
18     @RequestMapping(value = "/welcome", method = RequestMethod.GET)
19     public String welcomePage() {
20         System.out.println("[HomeController] Serving the request at /welcome");
21     }
22 }
  
```

4.5 Use the **@RequestMapping** annotation and the date function to process the request and retrieve the date format

```

1 package com.estore.controller;
2
3 import java.text.SimpleDateFormat;
4 import java.util.Date;
5
6 import org.springframework.stereotype.Controller;
7 import org.springframework.ui.Model;
8 import org.springframework.web.bind.annotation.RequestMapping;
9 import org.springframework.web.bind.annotation.RequestMethod;
10 import org.springframework.web.servlet.DispatcherServlet;
11 import org.springframework.web.servlet.view.InternalResourceViewResolver;
12
13 @Controller
14 public class HomeController {
15
16     @RequestMapping(value = "/", method = RequestMethod.GET)
17     public String homePage() {
18         System.out.println("[HomeController] Serving the request at /");
19         return "home";
20     }
21
22     @RequestMapping(value = "/welcome", method = RequestMethod.GET)
23     public String welcomePage(Model model) {
24         System.out.println("[HomeController] Serving the request at /welcome");
25
26         Date date = new Date();
27         SimpleDateFormat formatter = new SimpleDateFormat("dd-MM-yyyy hh:mm:ss");
28         String dateTimeStamp = formatter.format(date);
29
30         model.addAttribute("welcomeDateTimeStamp", dateTimeStamp);
31
32         return "welcome";
33     }
34 }
  
```

4.6 Use the `<p>` tag in the **Welcome.jsp** file to display the date format obtained from the `addAttribute` method

```

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 <center>
11   <h3>Welcome to Spring Web MVC</h3>
12   <p>Its: {welcomeDateTimeStamp}</p>
13 </center>
14 </body>
15 </html>

```

4.7 Add an anchor tag in the **Welcome.jsp** file to link the home page

The screenshot shows the Eclipse IDE with the **Welcome.jsp** file open. The file content is as follows:

```

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>Home</title>
8 </head>
9 <body>
10 <center>
11   <h3>Welcome to Home for Spring Web MVC</h3>
12   <a href="welcome">Click to Enter</a>
13 </center>
14 </body>
15 </html>

```

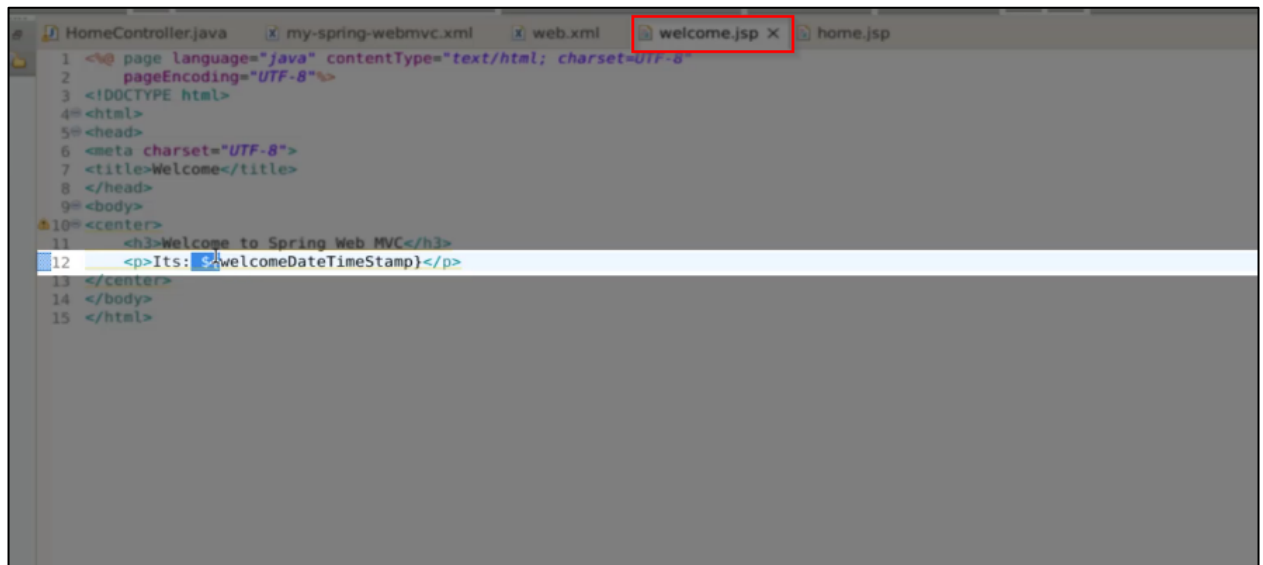
The **Project Explorer** on the left shows the project structure, with **home.jsp** selected under the **views** folder. The **Console** at the bottom displays the following log messages:

```

apache-tomcat-9.0.55 at localhost [Apache Tomcat] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full.linux.x86_64.17.0.1.v
INFO: Reloading Context with name [/springmvcdemo] is completed
Feb 10, 2022 10:39:42 AM org.apache.catalina.core.StandardContext reload
INFO: Reloading Context with name [/springmvcdemo] has started
Feb 10, 2022 10:39:42 AM org.apache.catalina.core.ApplicationContext log
INFO: Destroying Spring FrameworkServlet 'FrontControllerServlet'
Feb 10, 2022 10:39:43 AM org.apache.jasper.servlet.TldScanner scanJars
INFO: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for this logger for a
Feb 10, 2022 10:39:43 AM org.apache.catalina.core.ApplicationContext log
INFO: No Spring WebApplicationInitializer types detected on classpath
Feb 10, 2022 10:39:43 AM org.apache.catalina.core.ApplicationContext log
INFO: Initializing Spring DispatcherServlet 'FrontControllerServlet'
Feb 10, 2022 10:39:43 AM org.springframework.web.servlet.FrameworkServlet initServletBean
INFO: Initializing Servlet 'FrontControllerServlet'
Feb 10, 2022 10:39:44 AM org.springframework.web.servlet.FrameworkServlet initServletBean

```

#### 4.8 Use the \$ sign in the **Welcome.jsp** file to fetch the date function

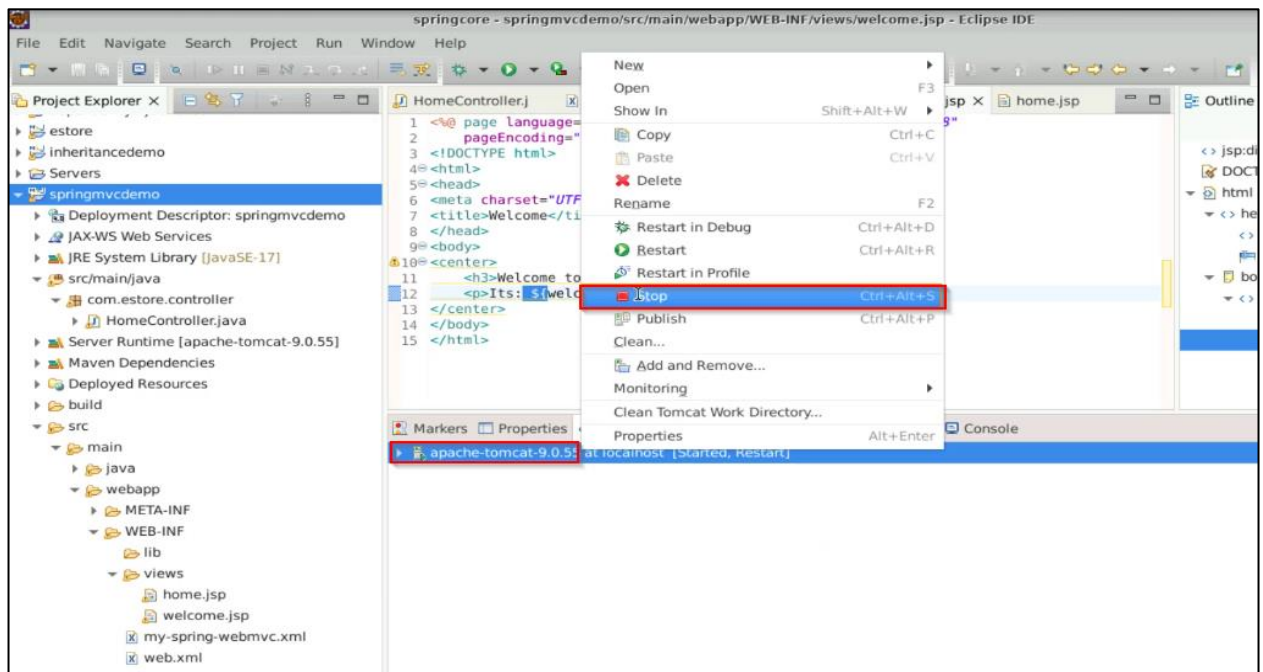


```

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 <center>
11 <h3>Welcome to Spring Web MVC</h3>
12 <p>Its: ${welcomeDateTimeStamp}</p>
13 </center>
14 </body>
15 </html>

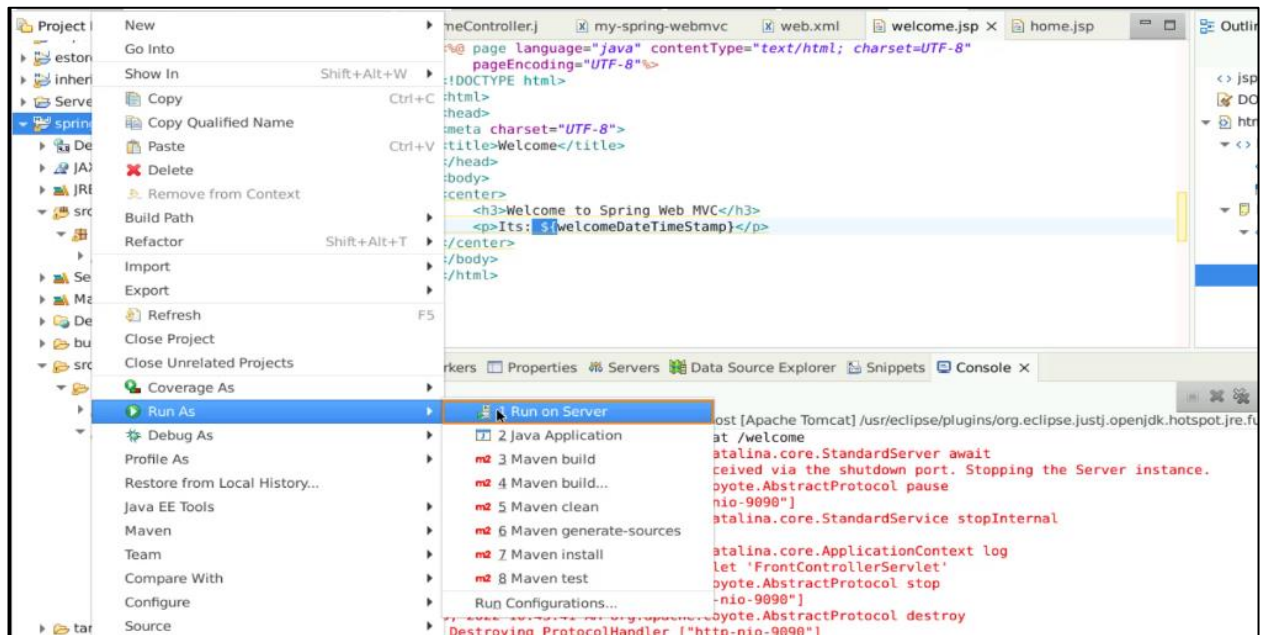
```

#### 4.9 Stop the Apache Tomcat server by right-clicking on the server's name and selecting the **Stop** option





#### 4.10 Rerun the project



#### 4.11 Once the server is up, go to the browser and click on the **Click to Enter** link

