**Step 1: Set Up Eclipse IDE for Java EE Developers**

Ensure that you are using **Eclipse IDE for Java EE Developers** or **Eclipse IDE for Java and Web Developers**, as it includes the necessary tools for creating web applications with Apache Tomcat.

**Step 2: Create a Dynamic Web Project**

1. Open **Eclipse IDE**.
2. Go to **File > New > Dynamic Web Project**.
3. Name your project (e.g., spring-mvc-app).
4. Select the **Runtime** as **Apache Tomcat 10** (if you haven't already added Tomcat to Eclipse, you can do so by going to **Window > Preferences > Server > Runtime Environments** and adding Tomcat 10).
5. Click **Finish**.

**Step 3: Add Spring MVC Dependencies**

You will need to add Spring MVC dependencies to your project. This example uses **Maven** for dependency management.

1. Right-click your project and select **Configure > Convert to Maven Project**.
2. Open the pom.xml file and add the following dependencies:

xml

<dependencies>

<!-- Spring Web MVC (Jakarta compatible) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>6.1.4</version>

</dependency>

<!-- Jakarta Servlet API -->

<dependency>

<groupId>jakarta.servlet</groupId>

<artifactId>jakarta.servlet-api</artifactId>

<version>6.0.0</version>

<scope>provided</scope>

</dependency>

<!-- JSTL (optional) -->

<dependency>

<groupId>jakarta.servlet.jsp.jstl</groupId>

<artifactId>jakarta.servlet.jsp.jstl-api</artifactId>

<version>3.0.0</version>

</dependency>

<dependency>

<groupId>org.glassfish.web</groupId>

<artifactId>jakarta.servlet.jsp.jstl</artifactId>

<version>3.0.1</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.13.0</version>

<configuration>

<release>21</release>

</configuration>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-war-plugin</artifactId>

<version>3.3.1</version>

</plugin>

</plugins>

</build>

**Step 4: Configure web.xml**

The web.xml file should define the Spring DispatcherServlet. Here's a sample web.xml for your Spring MVC app.

1. Navigate to WebContent/WEB-INF/web.xml and replace its content with the following:

xml

**<web-app version=*"3.0"* xmlns=*"http://java.sun.com/xml/ns/javaee"***

**xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"***

**xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"*>**

**<display-name>spring-mvc-app</display-name>**

**<!-- Spring MVC Dispatcher Servlet -->**

**<servlet>**

**<servlet-name>dispatcher</servlet-name>**

**<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>**

**<init-param>**

**<param-name>contextConfigLocation</param-name>**

**<param-value>/WEB-INF/dispatcher-servlet.xml</param-value>**

**</init-param>**

**<load-on-startup>1</load-on-startup>**

**</servlet>**

**<!-- Map all URLs to DispatcherServlet -->**

**<servlet-mapping>**

**<servlet-name>dispatcher</servlet-name>**

**<url-pattern>/</url-pattern>**

**</servlet-mapping>**

**</web-app>**

**Step 5: Create dispatcher-servlet.xml Configuration**

This XML file is where you configure Spring to scan for controllers and set up a view resolver.

1. Create a new file under WebContent/WEB-INF/dispatcher-servlet.xml and add the following configuration:

Xml

<?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:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context.xsd"*>

<!-- Enable component scanning for @Controller, @Service, etc. -->

<context:component-scan

base-package=*"com.example.controller"* />

<!-- Configure view resolver to resolve JSP pages -->

<bean

class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<property name=*"prefix"* value=*"/WEB-INF/"* />

<property name=*"suffix"* value=*".jsp"* />

</bean>

</beans>

**Step 6: Create a Controller**

1. Create a new Java class for the controller. Right-click the src folder and create a new package (e.g., com.example.controller).
2. Create a new class HomeController.java under that package and add the following code:

java

**package** com.example.controller;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.servlet.config.annotation.EnableWebMvc;

@Controller

@EnableWebMvc

**public** **class** HomeController {

@GetMapping("/")

**public** String showHomePage() {

**return** "home"; // This will map to /WEB-INF/home.jsp

}

@GetMapping("/test")

**public** String testPage() {

**return** "test"; // Resolves to /WEB-INF/test.jsp

}

}

**Step 7: Create the JSP View**

1. In the WebContent/WEB-INF/ directory, create a new file called home.jsp.
2. Add the following content to home.jsp:

jsp

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<%@ taglib uri=*"http://java.sun.com/jsp/jstl/core"* prefix=*"c"*%>

<%@ page session=*"false"*%>

<html>

<head>

<title>Home</title>

</head>

<body>

<h1>Hello world!</h1>

<P>The time on the server is ${serverTime}.</p>

<form action=*"user"* method=*"post"*>

<input type=*"text"* name=*"userName"*><br> <input

type=*"submit"* value=*"Login"*>

</form>

</body>

</html>

Test.jsp

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<body>

test

</body>

</html>

**Step 8: Configure Apache Tomcat in Eclipse**

1. Go to **Window > Preferences**.
2. Under **Server > Runtime Environments**, click **Add External Server** and select **Apache Tomcat 10**.
3. Ensure your Tomcat 10 installation is set correctly.

**Step 9: Deploy the Application**

1. Right-click the project and select **Run As > Run on Server**.
2. Choose your Apache Tomcat 10 server and click **Finish**.
3. Eclipse will build and deploy the project to the Tomcat server.

**Step 10: Test the Application**

1. Open your browser and go to the following URL:

bash

<http://localhost:9095/spring-mvc-app/>

You should see **Hello world!**

The time on the server is .

message from your home.jsp page.

**Troubleshooting Tips**

* If you see a **404 error**, ensure that your controller’s request mapping (/home) and the JSP path (/WEB-INF/home.jsp) are correct.
* Make sure Tomcat is running, and check the Tomcat logs for any specific issues related to the deployment.