Spring MVC framework enables separation of modules namely Model, View, and Controller, and seamlessly handles the application integration. This enables the developer to create complex applications also using plain java classes. The model object can be passed between view and controller using maps. In this article, we will see how to set up a Spring MVC application in the Eclipse IDE and understand how to make applications. The Spring MVC framework is comprised of the following components:

* **Model**: A model can be an object or collection of objects which basically contains the data of the application.
* **View**: A view is used for displaying the information to the user in a specific format. Spring supports various technologies like freemarker, velocity, and thymeleaf.
* **Controller**: It contains the logical part of the application. @Controller annotation is used to mark that class as controller.
* **Front Controller**: It remains responsible for managing the flow of the web application. DispatcherServlet acts as a front controller in Spring MVC.

Here are going to create our first spring MVC controller in Spring Tool Suite IDE.

***Requirements:***

* *Eclipse (EE version)/STS IDE*
* *Spring JAR Files*
* *Tomcat Apache latest version*

### Implementation:

It is illustrated below step by step as follows:

***Note****: We are going to use Spring Tool Suite 4 IDE for this project. Please refer to this article to install STS in your local machine [How to Download and Install Spring Tool Suite (Spring Tools 4 for Eclipse) IDE?](https://www.geeksforgeeks.org/how-to-download-and-install-spring-tool-suite-spring-tools-4-for-eclipse-ide/)*

**Step 1:**Create a Dynamic Web Project in your STS IDE. You may refer to this article to create a Dynamic Web Project in STS: [How to Create a Dyna](https://www.geeksforgeeks.org/how-to-create-a-dynamic-web-project-in-eclipse-spring-tool-suite/)

[mic Web Project in Spring Tool Suite?](https://www.geeksforgeeks.org/how-to-create-a-dynamic-web-project-in-eclipse-spring-tool-suite/)

**Step 2:**Download the spring JARs file from [this link](https://repo.spring.io/ui/native/libs-release/org/springframework/spring/5.3.9/spring-5.3.9-dist.zip" \t "https://www.geeksforgeeks.org/create-and-run-your-first-spring-mvc-controller-in-eclipse-spring-tool-suite/_blank) and go to the **src > main > webapp > WEB-INF > lib** folder and past these JAR files.

**Step 3:**[Configure Apache Tomcat Server](https://www.geeksforgeeks.org/configuration-of-apache-tomcat-server-with-eclipse-ide/) and configure the **Tomcat Server** with the application. Now we are ready to go.

### Configuring Dispatcher Servlet

Please refer to this article [What is Dispatcher Servlet in Spring?](https://www.geeksforgeeks.org/what-is-dispatcher-servlet-in-spring/) and read more about Dispatcher Servlet which is a very very important concept to understand. Now we are going to configure Dispatcher Servlet with our Spring MVC application.

**Step 4:** Go to the **src > main > webapp > WEB-INF > web.xml** file

### -File: web.xml

* XML

|  |
| --- |
| <?**xml** version="1.0" encoding="UTF-8"?>  <**web-app** xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"           xmlns="<http://xmlns.jcp.org/xml/ns/javaee>"           xsi:schemaLocation="<http://xmlns.jcp.org/xml/ns/javaee>  <http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd>" id="WebApp\_ID" version="4.0">      <**display-name**>myfirst-mvc-project</**display-name**>    <**welcome-file-list**>      <**welcome-file**>index.html</**welcome-file**>      <**welcome-file**>index.jsp</**welcome-file**>      <**welcome-file**>index.htm</**welcome-file**>      <**welcome-file**>default.html</**welcome-file**>      <**welcome-file**>default.jsp</**welcome-file**>      <**welcome-file**>default.htm</**welcome-file**>    </**welcome-file-list**>      <**absolute-ordering**/>      <**servlet**>        <!-- Provide a Servlet Name -->      <**servlet-name**>frontcontroller-dispatcher</**servlet-name**>      <!-- Provide a fully qualified path to the DispatcherServlet class -->      <**servlet-class**>org.springframework.web.servlet.DispatcherServlet</**servlet-class**>      <**load-on-startup**>1</**load-on-startup**>    </**servlet**>      <**servlet-mapping**>        <!-- Provide a Servlet Name that you want to map -->      <**servlet-name**>frontcontroller-dispatcher</**servlet-name**>      <!-- Provide a url pattern -->      <**url-pattern**>/student.com/\*</**url-pattern**>    </**servlet-mapping**>    </**web-app**> |

**Step 5:**Now go to the  **src > main > webapp > WEB-INF**and create an XML file. Actually, this is a Spring Configuration file like beans.xml file. And the name of the file must be in this format

YourServletName-servlet.xml

For example, for this project, the name of the file must be

frontcontroller-dispatcher-servlet.xml

So either you can create a Spring Configuration File or you can just create a simple XML file add the below lines of code inside that file.

### File: frontcontroller-dispatcher-servlet.xml

* 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>  <https://www.springframework.org/schema/beans/spring-beans.xsd>  <http://www.springframework.org/schema/context>  <https://www.springframework.org/schema/context/spring-context.xsd>">    </**beans**> |

### Creating First Spring MVC Controller

**Step 6:**Now, let’s create some controllers. Go to the src/main/java and create a new controllers package (For ex. com.student.controllers) as per your choice. And inside that create a Java class and name the class as DemoController. Now how to tell the Spring that this is our controller class. So the way we are going to tell the Spring is by marking it with a [@Controller annotation](https://www.geeksforgeeks.org/spring-controller-annotation-with-example/).

@Controller

public class DemoController {}

***Note:****Spring will automatically initialize the class having a @Controller annotation and register that class with the spring container.*

Now let’s create a simple method inside the Controller class and use @RequestMapping and @ResponseBody annotation before the method something like this.

// Annotation

@ResponseBody

@RequestMapping("/hello")

// Method

public String helloWorld()

{

return "Hello World!";

}

Now let’s understand both annotations.

**@RequestMapping(“/hello”),**so what does it mean? This line means, in the URL if somebody hits **student.com/hello**then this particular method will be executed and it is going to perform the operation that is written inside that particular method. For example, for this project, we are just returning a message “Hello World!” and we are expecting this is going to display in the client browser. But this thing will not happen. And to make it happen we need to use the @ResponseBody annotation. So @ResponseBody annotation is basically gonna write this particular message, here “Hello World!”, in your HTTP response.

### File: DemoController.java

* Java

|  |
| --- |
| // Java Program to Illustrate DemoController Class    **package** com.student.controllers;    // Importing required classes  **import** org.springframework.stereotype.Controller;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.ResponseBody;    // Annotation  @Controller  // Class  **public** **class** DemoController {        // Annotation      @ResponseBody      @RequestMapping("/hello")        // Method  **public** String helloWorld()      {    **return** "Hello World!";      }  } |

**S**

**tep 7:** Add the below line inside the **frontcontroller-dispatcher-servlet.xml** file

<context:component-scan base-package="com.student.controllers"></context:component-scan>

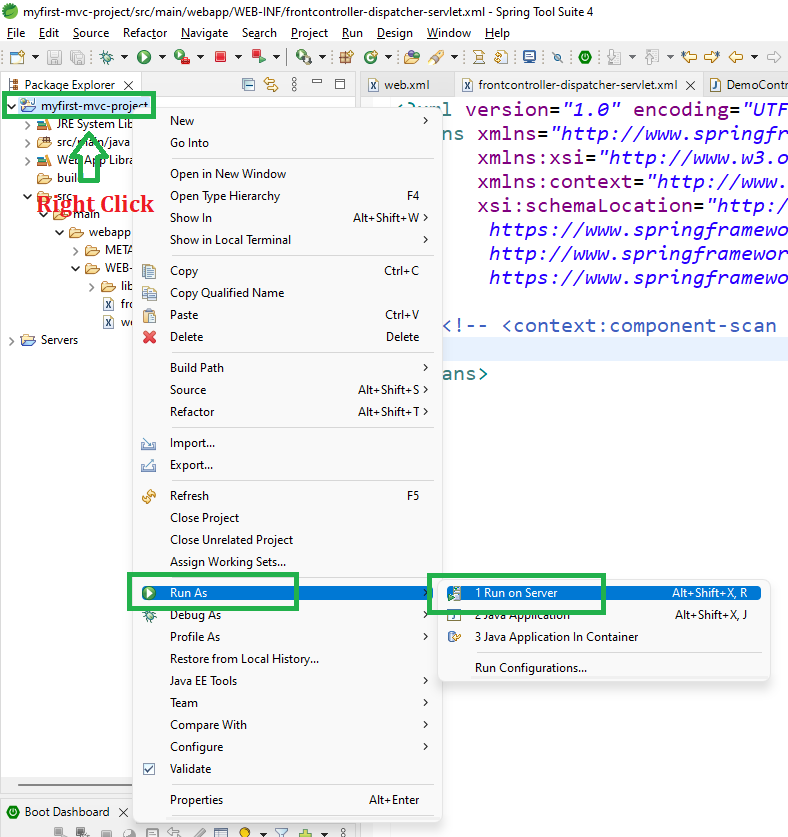
### File: frontcontroller-dispatcher-servlet.xml

* 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>  <https://www.springframework.org/schema/beans/spring-beans.xsd>  <http://www.springframework.org/schema/context>  <https://www.springframework.org/schema/context/spring-context.xsd>">        <**context:component-scan** base-package="com.student.controllers"></**context:component-scan**>    </**beans**> |

### Run Your First Spring MVC Controller

**Step 8:**To run your Spring MVC Application **right-click on your project > Run As > Run on Server**and run your application as shown in the below image.



After that use the following URL to run your controller as shown in the below image. All other details can be perceived through below image as follows:

http://localhost:8080/myfirst-mvc-project/student.com/hello

