How to Create Your First Model in Spring MVC?

Last Updated : 25 Jul, 2025

Spring MVC is a powerful Web MVC framework for building web applications. It is designed around the Model-View-Controller (MVC) pattern, which separates the application into three main components:

- **Model:** Represents the data of the application. It can be a single object or a collection of objects.
- **View:** Responsible for displaying the data to the user in a specific format. Spring supports various view technologies like JSP, Thymeleaf, Freemarker, and Velocity.
- **Controller:** This handles the logical part of the application. It processes user requests, interacts with the model, and returns the appropriate view.

In this article, we will focus on creating and using the Model component in a Spring MVC application. We will also cover how to create a controller, configure the Dispatcher Servlet, and render a view using Spring Tool Suite (STS) IDE.

Note: Models are nothing; they are just data.

> ***Pre-requisites:***
>
> *Prior to it, certain requirements are needed that are as follows:*
>
> - ***Eclipse*** *(EE version)/STS IDE*
> - ***Spring JAR Files***
> - ***Tomcat*** *Apache's latest version*

Note: We are going to use Spring Tool Suite 4 IDE for this project. Please do go through how to download and install Spring Tool Suite for Eclipse IDE

Step-by-Step Implementation

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 Dynamic Web Project in Spring Tool Suite?

Step 2: Download the spring JARs file and go to the **src > main > webapp > WEB-INF > lib** folder and past these JAR files.

Step 3: Configure Apache Tomcat Server and configure the **Tomcat Server** with the application. Now we are ready to go.

Configuring Dispatcher Servlet

> ***Note:*** *One should be well aware of what is Dispatcher Servlet in Spring as it is a crucial concept to understand prior adhering ahead.*

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 version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="https://www.geeksforgeeks.org/2001/XMLSchema-instance/"

xmlns="http://www.oracle.com/webfolder/technetwork/jsc/xml/ns/javaee/index.html"

xsi:schemaLocation="http://www.oracle.com/webfolder/technetwork/jsc/xml/ns/javaee/index.html

http://www.oracle.com/webfolder/technetwork/jsc/xml/ns/javaee/index.html/web-app_4_0.xsd"
    id="WebApp_ID" version="4.0">
    <display-name>springmvc-view-resolver</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>viewresolver-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>viewresolver-dispatcher</servlet-name>
        <!-- Provide a url pattern -->
        <url-pattern>/demo.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 the 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

viewresolver-dispatcher-servlet.xml

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

File: viewresolver-dispatcher-servlet.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans/"
       xmlns:xsi="https://www.geeksforgeeks.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">

  <!-- This Line is used for scanning all the packages that have controller
classes -->
  <context:component-scan base-package="com.demo.controllers">
</context:component-scan>

</beans>
```

Creating 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.demo.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.

```
@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 us create a simple method inside the Controller class and use the @RequestMapping annotation before the method something like this.

```
// Annotation
@RequestMapping("/hello")
// Method
public String helloWorld()
{
   return "";
}
```

Now in the return statement, we have to return some views (web pages), so whenever the endpoint '/hello' is invoked we can see our result on the web page. So let's create our first View.

Creating View

Go to the **src > main > webapp > WEB-INF > right-click > New > Folder** and name the folder as **views.** Then **views > right-click > New > JSP File** and name your first view. Here we have named it as **demo.jsp** file. Below is the code for the **demo.jsp** file. We have created a simple web page inside that file.

File: demo.jsp:

```
<!DOCTYPE html>
<html>
<body>
    <h1 align="center">GeeksforGeeks Welcome Page!</h1>
</body>
</html>
```

Now go to the DemoController class and inside the *helloWorld() method* we have to return a value something like this.

```
return "demo";
```

File: DemoController.java:

```
// Java Program to Illustrate DemoController Class

package com.demo.controllers;

// Importing required classes
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

// Class
@Controller
public class DemoController {

    // Method
    @RequestMapping("/hello") public String helloWorld()
    {
        // Just return the page name
```

```
        // No Path, no extension
        return "demo";
    }
}
```

Creating First Model

Now we are going to create our first Model inside our spring MVC application. We are going to send data to the view (jsp page) from the controller handler methods. So now let's create a String value using String literal something like this

String myName = "Amiya Rout";

And here we are going to send this "myName" data to the jsp page. And we can do it using the Model interface which is present inside the "org.springframework.ui.Model" package in Spring. We can write the code something like this

```
public String helloWorld(Model model) {
   // Sending data to view (jsp page)
   String myName = "Amiya Rout";
   model.addAttribute("myNameValue", myName);
}
```

The complete code for **DemoController.java** is given below.

File: DemoController.java:

```
// Java Program to Illustrate DemoController Class

package com.demo.controllers;

// Importing required classes
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

// Class
@Controller
public class DemoController {
```

```
    // Method
    @RequestMapping("/hello")
    public String helloWorld(Model model) {

        // Sending data to view (jsp page)
        String myName = "Amiya Rout";
        model.addAttribute("myNameValue", myName);

        // Just return the page name
        // No Path, no extension
        return "demo";
    }

}
```

And to display the data inside our jsp page we have to modify the **demo.jsp** page. And we can do it by the following line of code

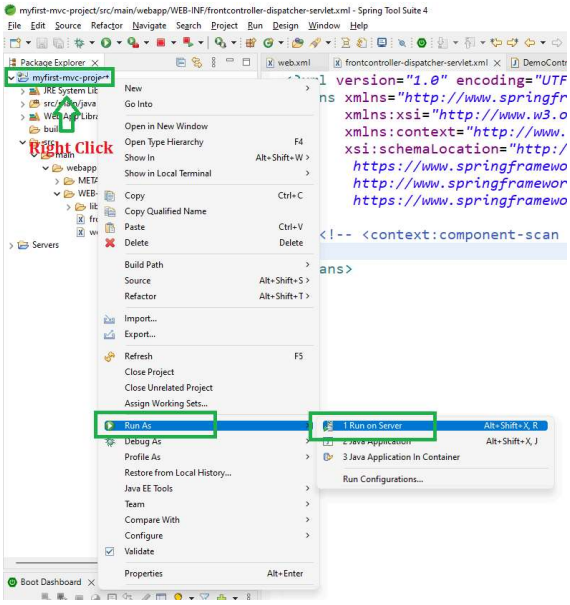
${myNameValue}

Now the complete code for **Demo.jsp** is given below and you are done. Now, let us run the application.

```
<!DOCTYPE html>
<html>
<body>
    <h1 align="center">GeeksforGeeks Welcome Page!</h1>
    <hr>
    My name is: ${myNameValue}
</body>
</html>
```

Run Spring MVC Application

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

http://localhost:8080/springmvc-view-resolver/demo.com/hello

Output: