Spring @RequestMapping Annotation with Example

Last Updated : 07 Aug, 2025

The @RequestMapping annotation in Spring MVC maps incoming HTTP requests to controller methods. It works with the DispatcherServlet, which routes requests to the correct handler. You can use @RequestMapping at the class level to define a base URL path and at the method level to handle specific actions. This helps organize and route requests cleanly in your application.

@RequestMapping Annotation at Method Level

> ***Note**: We are going to use Spring Tool Suite 4 IDE for this project. Please refer to this article to install STS on your local machine.*

Step 1: Create a Dynamic Web Project in STS

- Open Spring Tool Suite (STS) and navigate to File > New > Dynamic Web Project.
- Enter the Project Name (e.g., myfirst-mvc-project) and select the Target Runtime as Apache Tomcat.
- Click Finish to create the project.

Step 2: Add Spring JAR Files

- Download the Spring framework JARs from the Spring Repository.
- Copy the JAR files into the src/main/webapp/WEB-INF/lib folder.

Step 3: Configure Apache Tomcat Server

- In STS, right-click on the project and go to Properties > Targeted Runtimes.
- Select Apache Tomcat and click Apply and Close.
- Right-click on the project, select Run As > Run on Server.

Now, the project setup is complete.

The DispatcherServlet is the front controller in Spring MVC that routes incoming HTTP requests to the appropriate handler methods. Let’s configure it in the web.xml file.

Step 4: Configure web.xml

Go to the **src/main/webapp/WEB-INF/web.xml file** and add the following configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="https://www.w3.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>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>

  <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>
```

- The DispatcherServlet is mapped to /student.com/*, meaning all requests starting with /student.com/ will be handled by Spring MVC controllers.
- Spring automatically looks for a configuration file named frontcontroller-dispatcher-servlet.xml.

Step 5: Configure frontcontroller-dispatcher-servlet.xml

Go to the **src/main/webapp/WEB-INF folder** and create an XML file named **frontcontroller-dispatcher-servlet.xml**. Add the following configuration:

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

The component-scan ensures that Spring scans the com.student.controllers package for annotated classes.

Step 6: Create DemoController.java

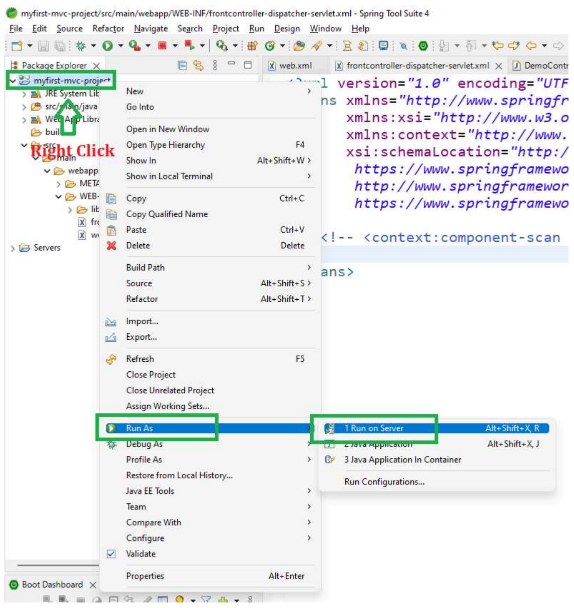
Go to the **src/main/java** folder and create a package named **com.student.controllers**. Inside this package, create a Java class named **DemoController.** Mark the class with the **@Controller annotation** to tell Spring that this is a controller class.

```java
package com.student.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class DemoController {

    @ResponseBody
    @RequestMapping("/hello")
    public String helloWorld() {
        return "Hello World!";
    }
}
```

- The @RequestMapping("/hello") annotation maps the /hello URL to the helloWorld() method.
- The @ResponseBody annotation indicates that the return value of the method will be the response body.

Step 7: Run the Application

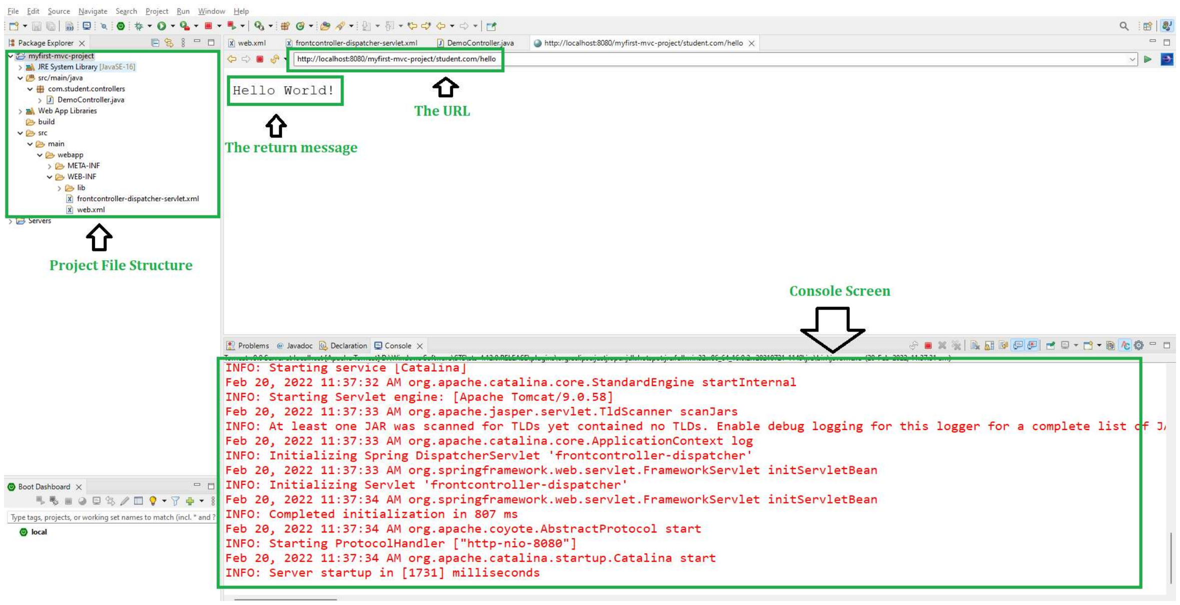
Right-click on your project and select **Run As > Run on Server.**



Use the following URL to access the controller:

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

Output:



```
INFO: Starting service [Catalina]
Feb 20, 2022 11:37:32 AM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet engine: [Apache Tomcat/9.0.58]
Feb 20, 2022 11:37:33 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 complete list of J
Feb 20, 2022 11:37:33 AM org.apache.catalina.core.ApplicationContext log
INFO: Initializing Spring DispatcherServlet 'frontcontroller-dispatcher'
Feb 20, 2022 11:37:33 AM org.springframework.web.servlet.FrameworkServlet initServletBean
INFO: Initializing Servlet 'frontcontroller-dispatcher'
Feb 20, 2022 11:37:34 AM org.springframework.web.servlet.FrameworkServlet initServletBean
INFO: Completed initialization in 807 ms
Feb 20, 2022 11:37:34 AM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-nio-8080"]
Feb 20, 2022 11:37:34 AM org.apache.catalina.startup.Catalina start
INFO: Server startup in [1731] milliseconds
```

The class-level @RequestMapping annotation maps a specific request path or pattern onto a controller. You can then apply additional method-level annotations to make mappings more specific to handler methods.

Multi-Action Controller

So in this example, we are going to create **Multi-Action Controller**. It's a Controller implementation that allows multiple request types to be handled by the same class. That means inside one controller class we can have many handler methods something like this.

Modify DemoController.java

Update the DemoController class to include a **class-level @RequestMapping annotation**:

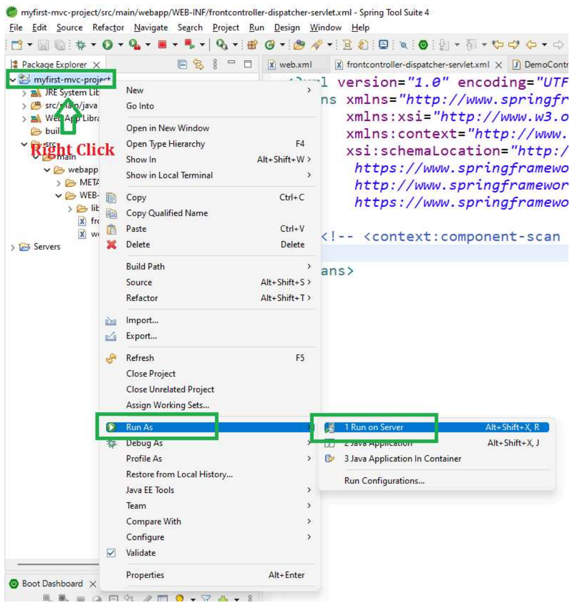
```java
package com.student.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
@RequestMapping("/boys")
```

```
public class DemoController {

    @ResponseBody
    @RequestMapping("/hello")
    public String helloWorld() {
        return "Hello World!";
    }

    @ResponseBody
    @RequestMapping("/geeksforgeeks")
    public String welcomeGfgMessage() {
        return "Welcome to GeeksforGeeks";
    }
}
```

- The class-level @RequestMapping("/boys") annotation applies a prefix to all method-level mappings.
- The helloWorld() method is now mapped to /boys/hello.
- The welcomeGfgMessage() method is mapped to /boys/geeksforgeeks.

Run the Updated Controller

Right-click on your project and select **Run As > Run on Server.**



And now, if you use this "http://localhost:8080/myfirst-mvc-project/student.com/hello" URL to run your controller then you are going to get the following warning and there will be no response

WARNING: No mapping for GET /myfirst-mvc-project/student.com/hello

In order to run your controller, you have to hit the following URL

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

Similarly, for the welcomeGfgMessage() handler method, you have to hit the following URL

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