How to Configure Dispatcher Servlet in web.xml File?

Last Updated : 23 Jul, 2025

In a Spring-based web application, the DispatcherServlet acts as the Front Controller. The Front Controller is responsible for handling all incoming requests and also figuring out which part of the application should handle it.

What is a Front Controller?

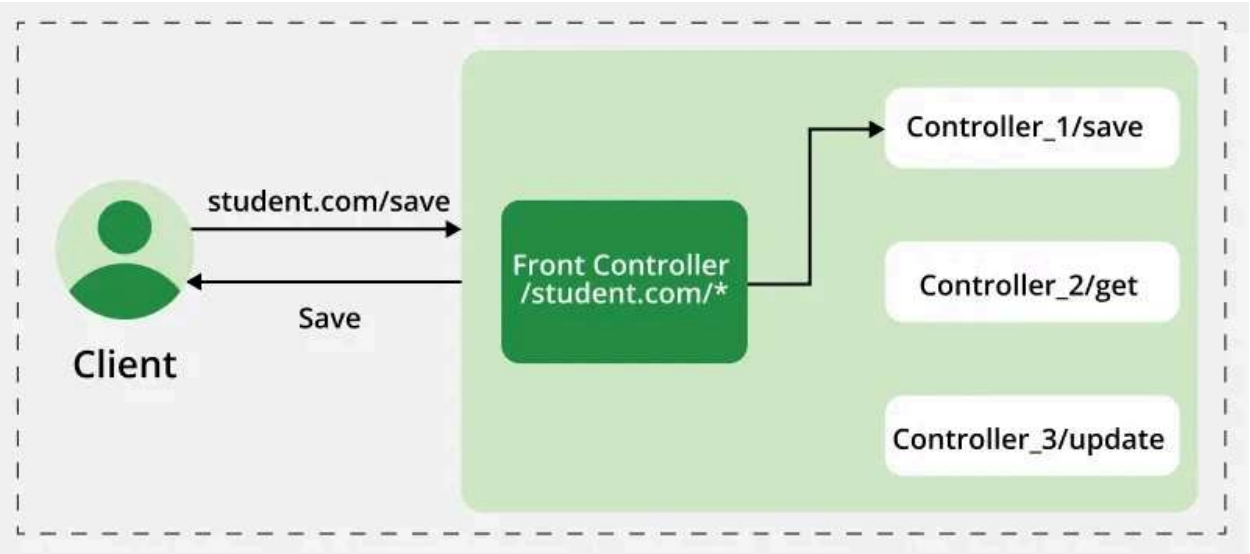
A Front Controller is a design pattern, which means it acts as a single entry point for all requests. It makes things like security, logging, and routing easier to manage.

- A client sends an HTTP request to the server.
- The DispatcherServlet receives the request and determines the appropriate controller to handle it.
- The controller processes the request and returns a response.
- The DispatcherServlet sends the response back to the client.

For example, ifa client requests to **student.com/save**, the Front Controller (DispatcherServlet) will receive the request and transfer it to the appropriate controller that handles the /save operation.

How to Create a Front Controller in a Spring MVC Application?

The Front Controller is already created by the Spring Framework, and it's called DispatcherServlet. we don't need to create a Front Controller from scratch we can use the dispatcherServlet provided by the Spring Framework.

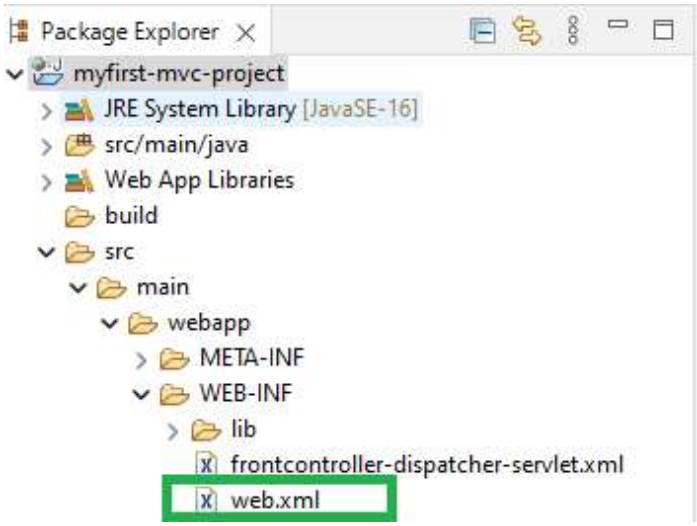


What is web.xml File?

In Spring, the **web.xml** file is the **Web Application Deployment Descriptor.** It defines the configuration of your web application, including servlets, filters, and other components.

Step 1: Configure DispatcherServlet in web.xml

Go to the **src/main/webapp/WEB-INF/ wen.xml** file.



Configure the **DispatcherServlet** inside the <servlet> tag as shown below

```
<servlet>
    <!-- Provide a Servlet Name -->
    <servlet-name>frontcontroller-dispatcher</servlet-name>
```

```
    <!-- Provide the fully qualified path to the DispatcherServlet class -->
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

    <!-- Load the DispatcherServlet on server startup -->
    <load-on-startup>1</load-on-startup>
</servlet>
```

Map the servlet to handle all requests coming to our website. For example, our website is gfg.com, we can configure it as follows:

```
<servlet-mapping>
    <!-- Provide the Servlet Name that you want to map -->
    <servlet-name>frontcontroller-dispatcher</servlet-name>

    <!-- Provide the URL pattern -->
    <url-pattern>/gfg.com/*</url-pattern>
</servlet-mapping>
```

This configuration means that the servlet frontcontroller-dispatcher will handle all requests starting with gfg.com/, such as gfg.com/save or gfg.com/get.

> ***Note:*** *The* ***<load-on-startup>1</load-on-startup>*** *tag ensures that the DispatcherServlet is initialized when the server starts, rather than waiting for the first request.*

Complete web.xml file

Here's the complete web.xml file with the DispatcherServlet 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.ht
ml"

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

http://www.oracle.com/webfolder/technetwork/jsc/xml/ns/javaee/index.html/web-
```

```
app_4_0.xsd"
         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>
        <servlet-name>frontcontroller-dispatcher</servlet-name>
        <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>frontcontroller-dispatcher</servlet-name>
        <url-pattern>/gfg.com/*</url-pattern>
    </servlet-mapping>

</web-app>
```

Step 2: Testing the Application

To verify that the DispatcherServlet has been successfully initialized, follow these steps which are listed below

- Go to **src > main > webapp > WEB-INF** and create a**n XML** file. This file acts as the Spring configuration file similar to beans.xml
- The name of the file must follow the following format

  YourServletName-servlet.xml

- For this project the file should named as **frontcontroller-dispatcher-servlet.xml**

Add the following line of code to the file:

```
<?xml version="1.0" encoding="UTF-8"?>
```

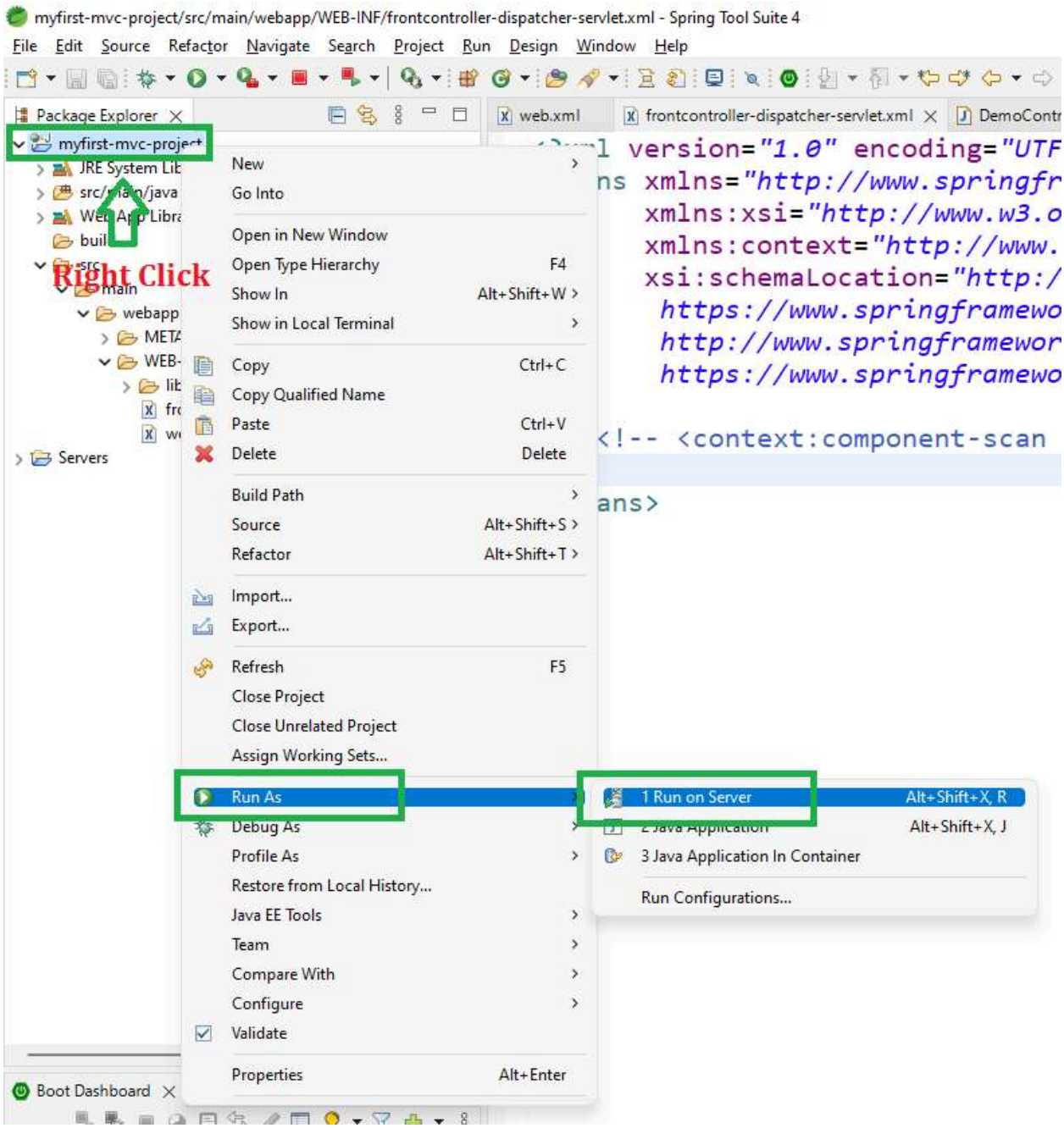
```
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans/"
       xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context/"
       xsi:schemaLocation="http://www.springframework.org/schema/beans/
       https://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context/
       https://www.springframework.org/schema/context/spring-context.xsd">

</beans>
```

Deploy the application and check the console logs.

> ***Note:*** *If the DispatcherServlet initializes without any exception or errors that means our configuration is successful.*

Step 3: Run the application

In order to run the application refer to the below image.



Now in the Console, you can see our DispatcherServlet has successfully initialized and also initialization completed without any exceptions or errors.