Spring 3 MVC Tutorial

In Spring Web MVC, **DispatcherServlet** class works as the front controller.

[](http://www.java4s.com/wp-content/uploads/2013/07/Spring-MVC-execution-flow.png)

## Spring MVC 3.2 Execution Flow

Step **1**: First request will be received by DispatcherServlet (in web.xml)

Step **2**: DispatcherServlet will take the help of HandlerMapping(in [servlet-class]-servlet.xml) and get to know the Controller class name associated with the given request

Step **3**: So request transfer to the Controller, and then controller will process the request by executing appropriate methods and returns ModeAndView object (contains Model data and View name) back to the DispatcherServlet

Step 4: Now DispatcherServlet(in web.xml) send the model object to the ViewResolver to get the actual view page

Step **5**: Finally DispatcherServlet will pass the Model object to the View page to display the result

## Execution Flow

* Run the application, then index.jsp file will be executed > click on the link given (I have given <a href=”java4s.html”>Click here to check the output :-)</a>)
* Once you click on that link, container will check the URL pattern at web.xml and passes the request to the DispatcherServlet
* DispatcherServlet then passes that request to our controller class
* Actually we are passing java4s.html from index.jsp right ? so DispatcherServlet verifies this ‘java4s’ name with the string in @RequestMapping(“-”) in our controller class if same it will executes the following method, which gives ModelAndView object as return type

|  |  |
| --- | --- |
| 1 | return new ModelAndView("welcomePage", "welcomeMessage", message); |

Means first argument is ‘View’ page name [ Where we are sending our result ], second, third arguments are key,values

* So DispatcherServlet search for the name welcomePage in /jsp folder with extension .jsp [ you can change the 'view page' folder name/location and its extension in welcome-servlet.xml at line numbers 14,15],  once the file was opened you can access the data by using the key welcomeMessage [2nd parameter in ModelAndView object]
* Check welcomePage.jsp > i am printing the result by calling the key${welcomeMessage}

## Note

* In web.xml we have given servlet name as **welcome**, so spring configuration file name must be welcome-servlet.xml [ {servletName-in-web.xml}-servlet.xml ]

### 1) Create the request page (optional)

This is the simple jsp page containing a link. It is optional page. You may direct invoke the action class instead.

**index.jsp**

1. <a href="hello.html">click</a>

### 2) Create the controller class

To create the controller class, we are using two annotations @Controller and @RequestMapping.

The **@Controller** annotation marks this class as Controller.

The **@Requestmapping** annotation is used to map the class with the specified name.

This class returns the instance of ModelAndView controller with the mapped name, message name and message value. The message value will be displayed in the jsp page.

**HelloWorldController.java**

1. **package** com.javatpoint;
2. **import** org.springframework.stereotype.Controller;
3. **import** org.springframework.web.bind.annotation.RequestMapping;
4. **import** org.springframework.web.servlet.ModelAndView;
6. @Controller
7. **public** **class** HelloWorldController {
9. @RequestMapping("/hello")
10. **public** ModelAndView helloWorld() {
12. String message = "HELLO SPRING MVC HOW R U";
13. **return** **new** ModelAndView("hellopage", "message", message);
14. }
16. }

### 3) Provide the entry of controller in the web.xml file

In this xml file, we are specifying the servlet class DispatcherServlet that acts as the front controller in Spring Web MVC. All the incoming request for the html file will be forwarded to the DispatcherServlet.

The web.xml file is the file that defines everything about an application that a server needs to know. It is placed in /WEB-INF/ directory of the application. The <servlet> element declares the DispatcherServlet. When the DispatcherServlet is initialized, the framework will try to load the application context from a file named [servlet-name]-servlet.xml located in /WEB-INF/ directory. So, we have created the spring-servlet.xml file, that will be explained below. The <servlet-mapping> element of web.xml file specifies what URLs will be handled by the DispatcherServlet.

ThewelcomePage.jsp will be invoked first when we execute the Spring web application. This is the only jspfile outside the WEB-INF directory and it is here to provide a redirect to the DispatcherServlet. All the other views should be stored under the WEB-INF directory so that they can be invoked only through the controller process.

**web.xml**

1. <?xml version="1.0" encoding="UTF-8"?>
2. <web-app version="2.5"
3. xmlns="http://java.sun.com/xml/ns/javaee"
4. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5. xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
6. http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd">
8. <servlet>
9. <servlet-name>spring</servlet-name>
10. <servlet-**class**>org.springframework.web.servlet.DispatcherServlet</servlet-**class**>
11. <load-on-startup>1</load-on-startup>
12. </servlet>
14. <servlet-mapping>
15. <servlet-name>spring</servlet-name>
16. <url-pattern>\*.html</url-pattern>
17. </servlet-mapping>

       <welcome-file-list>

<welcome-file>welcomePage.jsp</welcome-file>

1. </welcome-file-list>
2. </web-app>

### 4) Define the bean in the xml file

This is the important configuration file where we need to specify the ViewResolver and View components, HandlerMapping

The **context:component-scan** element defines the base-package where DispatcherServlet will search the controller class.

Here, the **InternalResourceViewResolver** class is used for the ViewResolver.

The **prefix+string returned by controller+suffix** page will be invoked for the view component.

This xml file should be located inside the WEB-INF directory.

**spring-servlet.xml**

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:p="http://www.springframework.org/schema/p"
5. xmlns:context="http://www.springframework.org/schema/context"
6. xsi:schemaLocation="http://www.springframework.org/schema/beans
7. http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
8. http://www.springframework.org/schema/context
9. http://www.springframework.org/schema/context/spring-context-3.0.xsd">
11. <context:component-scan  base-**package**="com.javatpoint" />
13. <bean **class**="org.springframework.web.servlet.view.InternalResourceViewResolver">
15. <property name="prefix" value="/WEB-INF/jsp/" />
16. <property name="suffix" value=".jsp" />
17. </bean>
19. </beans>

### 5) Display the message in the JSP page

This is the simple JSP page, displaying the message returned by the Controller.

It must be located inside the WEB-INF/jsp directory for this example only.

**hellopage.jsp**

1. Message is: ${message}

**Spring MVC provides different choice to map URL requests to the controller**

## BeanNameUrlHandlerMapping

The BeanNameUrlHandlerMapping class maps URL requests to beans names. It is the default handler mapping class, so it is the one created by the DispatcherServlet when Spring cannot find any handler mapping class declared. An example of using theBeanNameUrlHandlerMapping class is shown below. There are two beans declared, the first one’s name is helloWorld.htm and its class is the HelloWorldController. So the BeanNameUrlHandlerMapping will map any helloWorld URL request to this Controller. The second bean’s name is the hello\*.htm and its class is also the HelloWorldController. So, in this case, theBeanNameUrlHandlerMapping will map any URL request that starts with hello (such as helloWorld, helloAll) to theHelloWorldController.

mvc-dispatcher-servlet.xml

|  |  |
| --- | --- |
| 01 | <beans xmlns="<http://www.springframework.org/schema/beans>" |
| 02 | xmlns:context="<http://www.springframework.org/schema/context>" | |

|  |  |  |  |
| --- | --- | --- | --- |
| 03 | xmlns:mvc="<http://www.springframework.org/schema/mvc>"xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>" | | |
| 04 | | xsi:schemaLocation=" [http://www.springframework.org/schema/beans](http://www.springframework.org/schema/beans%C2%A0%C2%A0%C2%A0%C2%A0) |

|  |  |  |
| --- | --- | --- |
| 05 | <http://www.springframework.org/schema/beans/spring-beans-3.0.xsd><http://www.springframework.org/schema/context> <http://www.springframework.org/schema/context/spring-context-3.0.xsd> | |
| 06 | | <http://www.springframework.org/schema/mvc> <http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd>"> |

|  |  |
| --- | --- |
| 07 |  |
| 08 | <bean | |

|  |  |  |  |
| --- | --- | --- | --- |
| 09 | class="org.springframework.web.servlet.view.InternalResourceViewResolver"> | | |
| 10 | | <property name="prefix"> |

|  |  |  |
| --- | --- | --- |
| 11 | <value>/WEB-INF/</value> | |
| 12 | </property> |

|  |  |  |
| --- | --- | --- |
| 13 | <property name="suffix"> | |
| 14 | <value>.jsp</value> |

|  |  |  |
| --- | --- | --- |
| 15 | </property> | |
| 16 | </bean> |

|  |  |
| --- | --- |
| 17 |  |
| 18 | <bean | |

|  |  |  |  |
| --- | --- | --- | --- |
| 19 | class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping"/> | | |
| 20 | |  |

|  |  |
| --- | --- |
| 21 | <bean name="/helloWorld.htm" |
| 22 | class="com.javacodegeeks.snippets.enterprise.HelloWorldController" /> | |

|  |  |
| --- | --- |
| 23 |  |
| 24 | <bean name="/hello\*.htm" | |

|  |  |  |
| --- | --- | --- |
| 25 | class="com.javacodegeeks.snippets.enterprise.HelloWorldController" /> | |
| 26 |  |

|  |  |
| --- | --- |
| 27 | </beans> |

So, check what happens when the calling the URL helloWorld.htm:  
[](http://a5e2fba00d8bcb729d89839f.javacodegeeks.netdna-cdn.com/wp-content/uploads/2014/03/BeanNameUrlHandlerMapping1.png)

And here is the case of helloGeeks.htm:  
[](http://a5e2fba00d8bcb729d89839f.javacodegeeks.netdna-cdn.com/wp-content/uploads/2014/03/BeanNameUrlHandlerMapping2.png)

## 5. ControllerClassNameHandlerMapping

The ControllerClassNameHandlerMapping class uses a convention to determine the mapping between request URLs and the Controller instances that are to handle those requests. In this case, there is no need to declare a bean name for the Controller. In the example below, the ControllerClassNameHandlerMapping will map to the HelloWorldController all URL requests that start with helloWorld, or helloWorld\*. In the ControllerClassNameHandlerMapping bean declaration there are two properties to configure, thecaseSensitive, which is set to true, and the pathPrefix, which is set to /javacodegeeks/. These properties allowControllerClassNameHandlerMapping to also map to the HelloWorldController all URL requests with uppercase characters, likehelloWorldJavaCodeGeeks, as also URL requests with path prefix like /javacodegeeks/helloWorld.

mvc-dispatcher-servlet.xml

|  |  |
| --- | --- |
| 01 | .... |
| 02 | <bean | |

|  |  |  |  |
| --- | --- | --- | --- |
| 03 | class="org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping" > | | |
| 04 | | <property name="caseSensitive" value="true" /> |

|  |  |  |
| --- | --- | --- |
| 05 | <property name="pathPrefix" value="/javacodegeeks" /> | |
| 06 | </bean> |

|  |  |
| --- | --- |
| 07 |  |
| 08 |  | |

|  |  |  |
| --- | --- | --- |
| 09 | <bean class="com.javacodegeeks.snippets.enterprise.HelloWorldController" /> | |
| 10 | ... |

The cases described above are shown in the screenshots below.  
Here is a case of uppercase characters:

[](http://a5e2fba00d8bcb729d89839f.javacodegeeks.netdna-cdn.com/wp-content/uploads/2014/03/ControllerClassNameHandlerMapping2.png)

And here is a case with a pathPrefix:

[](http://a5e2fba00d8bcb729d89839f.javacodegeeks.netdna-cdn.com/wp-content/uploads/2014/03/ControllerClassNameHandlerMapping1.png)

## 6. SimpleUrlHandlerMapping

The SimpleUrlHandlerMapping provides a property called mappings so as to be configured. This property is set in the bean declaration and consists of key value mapping pairs. It can be set in two ways, as shown below:

mvc-dispatcher-servlet.xml

|  |  |  |
| --- | --- | --- |
| 01 | | .... |
| 02 | <bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping"> | | |

|  |  |  |
| --- | --- | --- |
| 03 | <property name="mappings"> | |
| 04 | <props> |

|  |  |  |
| --- | --- | --- |
| 05 | <prop key="/helloWorld.htm">helloWorldController</prop> | |
| 06 | <prop key="/\*/hello.htm">helloWorldController</prop> |

|  |  |  |
| --- | --- | --- |
| 07 | <prop key="/hello\*.htm">helloWorldController</prop> | |
| 08 | </props> |

|  |  |  |
| --- | --- | --- |
| 09 | </property> | |
| 10 | </bean> |

|  |  |  |
| --- | --- | --- |
| 11 | |  |
| 12 | <bean id="helloWorldController"class="com.javacodegeeks.snippets.enterprise.HelloWorldController" /> | | |

|  |  |
| --- | --- |
| 13 | ... |

mvc-dispatcher-servlet.xml

|  |  |  |
| --- | --- | --- |
| 01 | | .... |
| 02 | <bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping"> | | |

|  |  |  |
| --- | --- | --- |
| 03 | <property name="mappings"> | |
| 04 | <value> |

|  |  |  |
| --- | --- | --- |
| 05 | /helloWorld.htm=helloWorldController | |
| 06 | /\*/hello.htm=helloWorldController |

|  |  |  |
| --- | --- | --- |
| 07 | /hello\*.htm=helloWorldController | |
| 08 | </value> |

|  |  |  |
| --- | --- | --- |
| 09 | </property> | |
| 10 | </bean> |

|  |  |  |
| --- | --- | --- |
| 11 | |  |
| 12 | <bean id="helloWorldController"class="com.javacodegeeks.snippets.enterprise.HelloWorldController" /> | | |

|  |  |
| --- | --- |
| 13 | .... |

Note that the Controller bean declaration uses an id property, which is used in the SimpleUrlHandlerMapping bean declaration for the mapping. Each one of the cases configured above, are shown in the screenshots below:

[](http://a5e2fba00d8bcb729d89839f.javacodegeeks.netdna-cdn.com/wp-content/uploads/2014/03/SimpleUrlHandlerMapping1.png)

[](http://a5e2fba00d8bcb729d89839f.javacodegeeks.netdna-cdn.com/wp-content/uploads/2014/03/SimpleUrlHandlerMapping2.png)

[](http://a5e2fba00d8bcb729d89839f.javacodegeeks.netdna-cdn.com/wp-content/uploads/2014/03/SimpleUrlHandlerMapping3.png)

## 7. Handler mapping priorities

The handler mapping implementations described can be mixed and used together. The only thing that needs to be configured is the priority of each mapping class, so that Spring MVC DispatcherServlet will know which handler mapping implementation to use with what priority. The priority can be set as a property in every mapping bean declaration, as shown below:

mvc-dispatcher-servlet.xml

|  |  |  |
| --- | --- | --- |
| 01 | | ... |
| 02 | <bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping"> | | |

|  |  |  |
| --- | --- | --- |
| 03 | <property name="mappings"> | |
| 04 | <value> |

|  |  |  |
| --- | --- | --- |
| 05 | /helloWorld.htm=helloWorldController | |
| 06 | /\*/hello.htm=helloWorldController |

|  |  |  |
| --- | --- | --- |
| 07 | /hello\*.htm=helloWorldController | |
| 08 | </value> |

|  |  |
| --- | --- |
| 09 | </property> |
| 10 | <property name="order" value="0" /> | |

|  |  |  |
| --- | --- | --- |
| 11 | </bean> | |
| 12 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 13 | <bean id="helloWorldController"class="com.javacodegeeks.snippets.enterprise.HelloWorldController" /> | | |
| 14 | |  |

|  |  |
| --- | --- |
| 15 |  |
| 16 | <bean | |

|  |  |  |  |
| --- | --- | --- | --- |
| 17 | class="org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping" > | | |
| 18 | | <property name="caseSensitive" value="true" /> |

|  |  |  |
| --- | --- | --- |
| 19 | <property name="pathPrefix" value="/javacodegeeks" /> | |
| 20 | <property name="order" value="1" /> |

|  |  |  |
| --- | --- | --- |
| 21 | </bean> | |
| 22 |  |

|  |  |
| --- | --- |
| 23 |  |
| 24 | <bean class="com.javacodegeeks.snippets.enterprise.HelloWorldController" /> | |

|  |  |
| --- | --- |
| 25 | ... |

In this case, both ControllerClassNameHandlerMapping and SimpleUrlHandlerMapping are used, but the first one to handle a URL request will be the SimpleUrlHandlerMapping.