Spring MVC Tutorial

1. [Spring MVC Tutorial](http://www.javatpoint.com/spring-3-mvc-tutorial)
2. [Example of Spring 3 Web MVC](http://www.javatpoint.com/spring-3-mvc-tutorial#ex)
3. [Spring 3 MVC Multiple Controller Example](http://www.javatpoint.com/spring-3-mvc-tutorial#multiple)
4. [Spring 3 MVC Login Example](http://www.javatpoint.com/spring-3-mvc-tutorial#login)

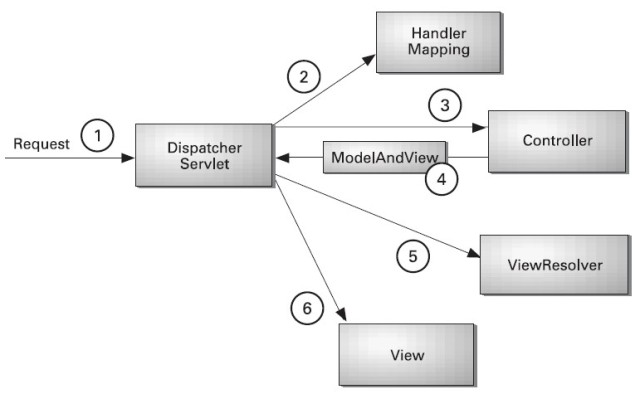
**Spring MVC** tutorial provides an elegant solution to use MVC in spring framework by the help of DispatcherServlet.

In Spring Web MVC, **DispatcherServlet** class works as the front controller. It is responsible to manage the flow of the spring mvc application.

The **@Controller** annotation is used to mark the class as the controller in Spring 3.

The **@RequestMapping** annotation is used to map the request url. It is applied on the method.

Understanding the flow of Spring Web MVC



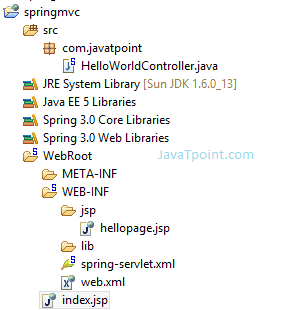
As displayed in the figure, all the incoming request is intercepted by the DispatcherServlet that works as the front controller. The DispatcherServlet gets entry of handler mapping from the xml file and forwards the request to the controller. The controller returns an object of ModelAndView. The DispatcherServlet checks the entry of view resolver in the xml file and invokes the specified view component.

Spring MVC Example

Let's see the simple example of spring 3 web MVC. There are given 7 steps for creating the spring MVC application. The steps are as follows:

1. **Create the request page (optional)**
2. **Create the controller class**
3. **Provide the entry of controller in the web.xml file**
4. **Define the bean in the xml file**
5. **Display the message in the JSP page**
6. **Load the spring core and mvc jar files**
7. **Start server and deploy the project**

Directory Structure



Required Jar files

To run this example, you need to load:

* **Spring Core jar files**
* **Spring Web jar files**

[download all the jar files for spring including core, web, aop, mvc, j2ee, remoting, oxm, jdbc, orm etc.](http://www.javatpoint.com/src/sp/springjars.zip)

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;
5. @Controller
6. **public** **class** HelloWorldController {
7. @RequestMapping("/hello")
8. **public** ModelAndView helloWorld() {
9. String message = "HELLO SPRING MVC HOW R U";
10. **return** **new** ModelAndView("hellopage", "message", message);
11. }
12. }

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.

**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">
7. <servlet>
8. <servlet-name>spring</servlet-name>
9. <servlet-**class**>org.springframework.web.servlet.DispatcherServlet</servlet-**class**>
10. <load-on-startup>1</load-on-startup>
11. </servlet>
12. <servlet-mapping>
13. <servlet-name>spring</servlet-name>
14. <url-pattern>\*.html</url-pattern>
15. </servlet-mapping>
16. </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.

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">
10. <context:component-scan  base-**package**="com.javatpoint" />
11. <bean **class**="org.springframework.web.servlet.view.InternalResourceViewResolver">
12. <property name="prefix" value="/WEB-INF/jsp/" />
13. <property name="suffix" value=".jsp" />
14. </bean>
15. </beans>

5) Display the message in the JSP page

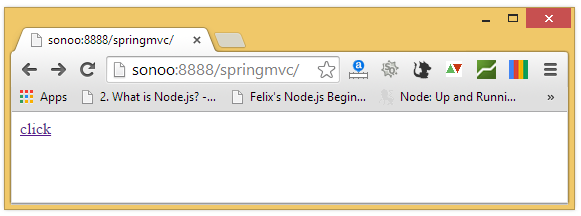
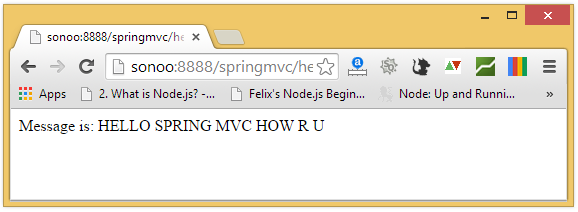
This is the simple JSP page, displaying the message returned by the Controller.

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

**hellopage.jsp**

1. Message is: ${message}

Output

Download spring MVC example

We have created this application in MyEclipse IDE which already provides the jar files. If you use eclipse or other IDE's, you need to load the jar file for spring MVC.

[download this example (developed using Eclipse)](http://www.javatpoint.com/sppages/download/SpringMVCBasic.zip)  
[download this example (developed using MyEclipse)](http://www.javatpoint.com/src/sp/mvc1.zip)

Spring 3 MVC Multiple Controller Example

We can have a lot of controller classes in Spring Framework. In this example, we are creating two Controller classes HelloWorldController and WelcomeWorldController.

1) Controller Classes

**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";
13. **return** **new** ModelAndView("hellopage", "message", message);
14. }
16. }

**WelcomeWorldController.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** WelcomeWorldController {
9. @RequestMapping("/welcome")
10. **public** ModelAndView helloWorld() {
12. String message = "WELCOME SPRING MVC";
13. **return** **new** ModelAndView("welcomepage", "message", message);
14. }
16. }

2) View components

To run this example, It must be located inside the WEB-INF/jsp directory.

**hellopage.jsp**

1. Message is: ${message}

**welcomepage.jsp**

1. Message is: ${message}

3) Index page

It is the optional welcome page, that provide the links to invoke both controller.

**index.jsp**

1. <a href="hello.html">click</a>|
2. <a href="welcome.html">click</a>

***Other pages are same e.g. spring-servlet.xml and web.xml.***

[download this example (developed using myeclipse IDE)](http://www.javatpoint.com/src/sp/mvc2.zip)

Spring MVC Login Example

We can simply create login application by following the Spring MVC. We need to pass HttpServletRequest and HttpServletResponse objects in the request processing method of the Controller class. Let's see the example:

1) Controller Class

**HelloWorldController.java**

1. **package** com.javatpoint;
2. **import** javax.servlet.http.HttpServletRequest;
3. **import** javax.servlet.http.HttpServletResponse;
4. **import** org.springframework.stereotype.Controller;
5. **import** org.springframework.web.bind.annotation.RequestMapping;
6. **import** org.springframework.web.servlet.ModelAndView;
8. @Controller
9. **public** **class** HelloWorldController {
11. @RequestMapping("/hello")
12. **public** ModelAndView helloWorld(HttpServletRequest request,HttpServletResponse res) {
13. String name=request.getParameter("name");
14. String password=request.getParameter("password");
16. **if**(password.equals("admin")){
17. String message = "HELLO "+name;
18. **return** **new** ModelAndView("hellopage", "message", message);
19. }
20. **else**{
21. **return** **new** ModelAndView("errorpage", "message","Sorry, username or password error");
22. }
23. }
25. }

2) View components

To run this example, It must be located inside the WEB-INF/jsp directory.

**hellopage.jsp**

1. Message is: ${message}

**errorpage.jsp**

1. ${message}
2. <jsp:include page="/index.jsp"></jsp:include>

3) Index page

It is the login page, that recieve name and password from the user.

**index.jsp**

1. <form action="hello.html" method="post">
2. Name:<input type="text" name="name"/><br/>
3. Password:<input type="password" name="password"/><br/>
4. <input type="submit" value="login"/>
5. </form>

***Other pages are same e.g. spring-servlet.xml and web.xml.***

Spring MVC Tiles Example

By the help of Tiles framework, we can manage the **layout** of the spring mvc web application.

Click here for more details about [spring mvc example with tiles](http://www.javatpoint.com/spring-mvc-tiles-example).