Spring MVC Tutorial

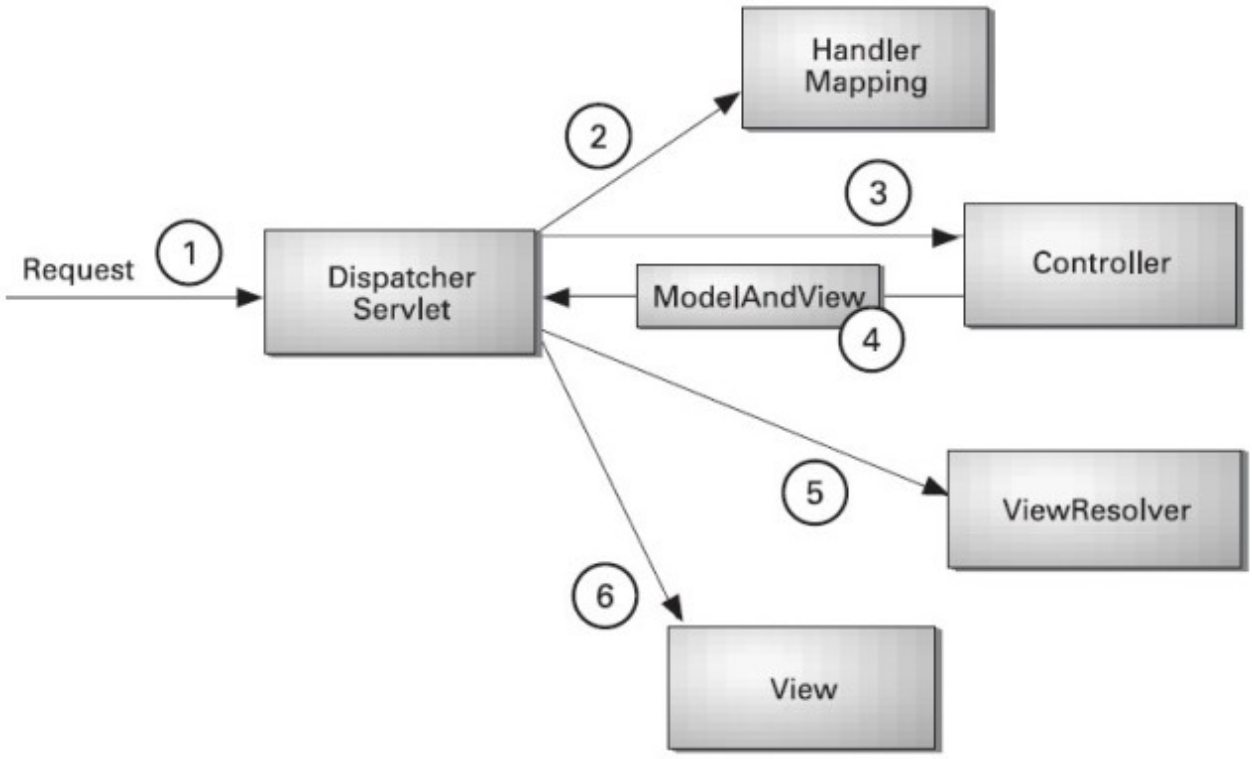
Spring MVC tutorial provides an elegant solution to use MVC in spring framework by the help of DispatcherServlet.

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



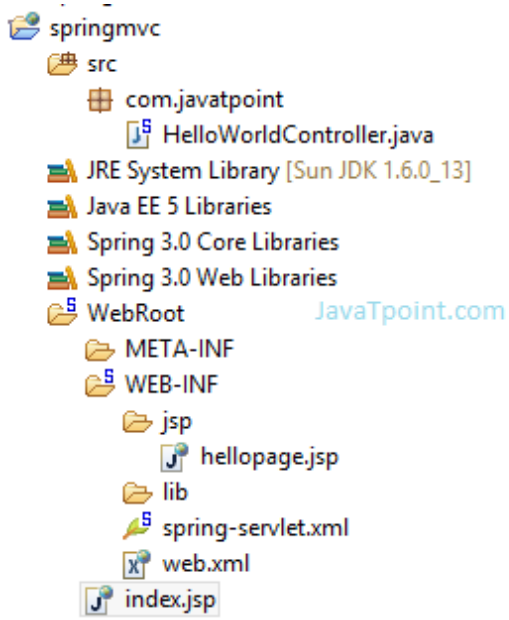
As displayed in the figure, all the incoming request is intercepted by the DispatcherServlet that works as the front controller. The DispatcherServlet gets entry of handler mapping from the xml file and forwards the request to the controller. The controller returns an object of ModelAndView. The DispatcherServlet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

Spring Web MVC Framework Example

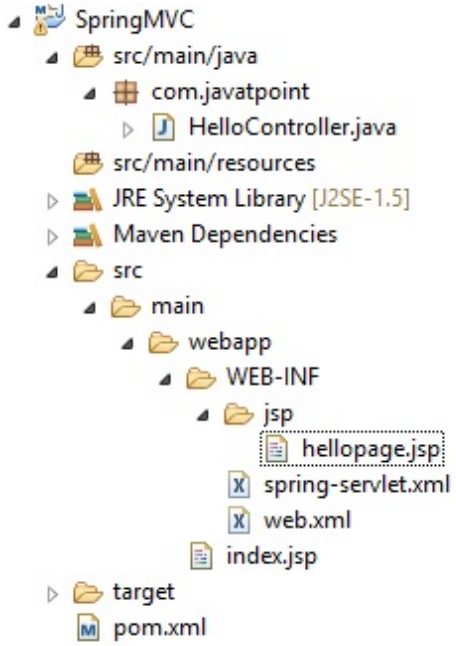
Let's see the simple example of spring web MVC framework. There are given 7 steps for creating the spring MVC application. The steps are as follows:

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



Directory Structure of Spring MVC using Maven



Required Jar files or Maven Dependency

To run this example, you need to load:

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

Download Link: Download all the jar files for spring including core, web, aop, mvc, j2ee, remoting, oxm, jdbc, orm etc.

If you are using Maven, you don't need to add jar files. Now, you need to add maven dependency in pom.xml file.

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.javatpoint</groupId>
  <artifactId>SpringMVC</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>SpringMVC Maven Webapp</name>
  <url>http://maven.apache.org</url>
  <dependencies>

<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
</dependency>
<dependency>
   <groupId>javax.servlet</groupId>
   <artifactId>servlet-api</artifactId>
   <version>3.0-alpha-1</version>
</dependency>
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-core</artifactId>
   <version>3.1.2.RELEASE</version>
</dependency>
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-context</artifactId>
   <version>3.1.2.RELEASE</version>
</dependency>
<dependency>
   <groupId>org.springframework</groupId>
```

```
    <artifactId>spring-beans</artifactId>
    <version>3.1.2.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>3.1.2.RELEASE</version>
</dependency>

</dependencies>
  <build>
    <finalName>SpringMVC</finalName>
  </build>
</project>
```

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

```
<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

```
package com.javatpoint;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;
@Controller
public class HelloWorldController {
```

```
    @RequestMapping("/hello")
    public ModelAndView helloWorld() {
        String message = "HELLO SPRING MVC HOW R U";
        return new ModelAndView("hellopage", "message", message);
    }
}
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
 <servlet>
    <servlet-name>spring</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>spring</servlet-name>
    <url-pattern>*.html</url-pattern>
</servlet-mapping>
</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

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
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">
    <context:component-scan  base-package="com.javatpoint" />
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/jsp/" />
        <property name="suffix" value=".jsp" />
    </bean>
</beans>
```

5) Display the message in the JSP page

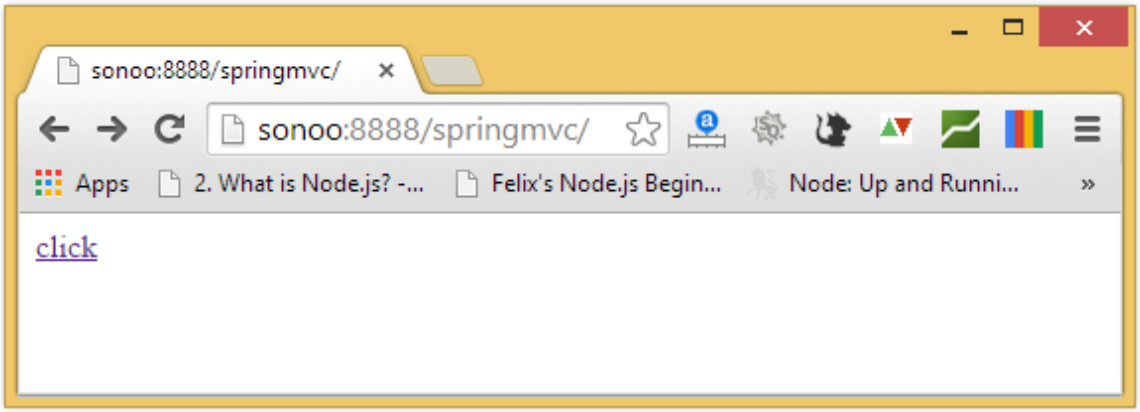
This is the simple JSP page, displaying the message returned by the Controller.

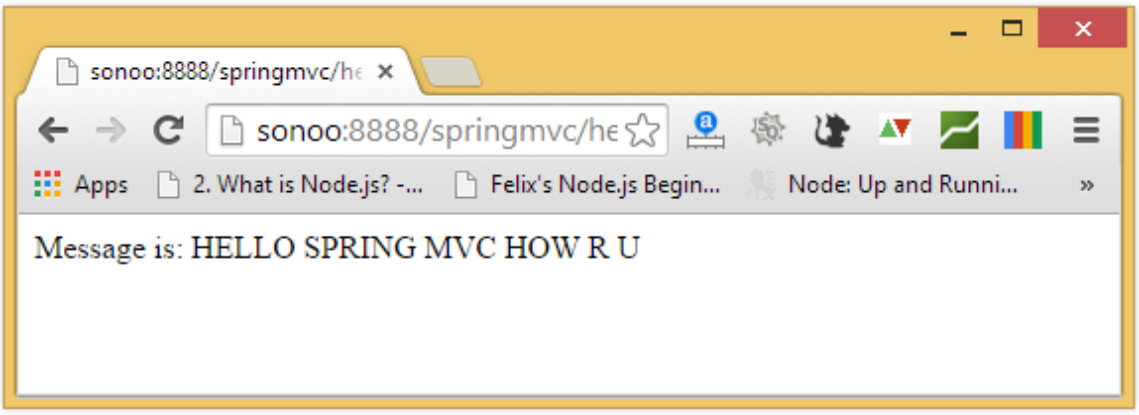
It must be located inside the WEB-INF/jsp directory for this example only.

hellopage.jsp

```
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)
Download this example (developed using Eclipse with Maven)
download this example (Developed using MyEclipse)

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

```
package com.javatpoint;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;
```

```
@Controller
public class HelloWorldController {

    @RequestMapping("/hello")
    public ModelAndView helloWorld() {

        String message = "HELLO SPRING MVC";
        return new ModelAndView("hellopage", "message", message);
    }

}
```

WelcomeWorldController.java

```
package com.javatpoint;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class WelcomeWorldController {

    @RequestMapping("/welcome")
    public ModelAndView helloWorld() {

        String message = "WELCOME SPRING MVC";
        return new ModelAndView("welcomepage", "message", message);
    }

}
```

2) View components

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

hellopage.jsp

```
Message is: ${message}
```

welcomepage.jsp

```
Message is: ${message}
```

3) Index page

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

index.jsp

```
<a href="hello.html">click</a>|
<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)

Spring MVC Request Response 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

```
package com.javatpoint;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class HelloWorldController {

    @RequestMapping("/hello")
    public ModelAndView helloWorld(HttpServletRequest request,HttpServletResponse res) {
        String name=request.getParameter("name");
        String password=request.getParameter("password");

        if(password.equals("admin")){
```

```
        String message = "HELLO "+name;
        return new ModelAndView("hellopage", "message", message);
        }
        else{
            return new ModelAndView("errorpage", "message","Sorry, username or password error");
        }
    }

}
```

2) View components

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

hellopage.jsp

```
Message is: ${message}
```

errorpage.jsp

```
${message}
<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

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

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

Spring MVC Form Handling Example

By the help of form handling, we will be able to create CRUD application using Spring MVC.

Click here for more details about spring mvc form handling example.

Spring MVC CRUD Example

By the help of form handling and JdbcTemplate, we will be able to create CRUD application using Spring MVC.

Click here for more details about spring mvc crud example.

Spring MVC Pagination Example

Let's see a simple example of pagination using Spring MVC.

Click here for more details about spring mvc pagination example.

Spring MVC File Upload Example

Let's see a file upload application using Spring MVC.

Click here for more details about spring mvc file upload example.

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.

← prev next →