

SPRING MVC HELLO WORLD EXAMPLE

http://www.tutorialspoint.com/spring/spring_mvc_hello_world_example.htm

Copyright © tutorialspoint.com

The following example show how to write a simple web based Hello World application using Spring MVC framework. To start with it, let us have working Eclipse IDE in place and follow the following steps to develop a Dynamic Web Application using Spring Web Framework:

Step	Description
1	Create a <i>Dynamic Web Project</i> with a name <i>HelloWeb</i> and create a package <i>com.tutorialspoint</i> under the <i>src</i> folder in the created project.
2	Drag and drop below mentioned Spring and other libraries into the folder <i>WebContent/WEB-INF/lib</i> .
3	Create a Java class <i>HelloController</i> under the <i>com.tutorialspoint</i> package.
4	Create Spring configuration files <i>Web.xml</i> and <i>HelloWeb-servlet.xml</i> under the <i>WebContent/WEB-INF</i> folder.
5	Create a sub-folder with a name <i>jsp</i> under the <i>WebContent/WEB-INF</i> folder. Create a view file <i>hello.jsp</i> under this sub-folder.
6	The final step is to create the content of all the source and configuration files and export the application as explained below.

Here is the content of **HelloController.java** file:

```
package com.tutorialspoint;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.ui.ModelMap;

@Controller
@RequestMapping("/hello")
public class HelloController{

    @RequestMapping(method = RequestMethod.GET)
    public String printHello(ModelMap model) {
        model.addAttribute("message", "Hello Spring MVC Framework!");

        return "hello";
    }
}
```

Following is the content of Spring Web configuration file **web.xml**

```
<web-app
    xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

    <display-name>Spring MVC Application</display-name>

    <servlet>
        <servlet-name>HelloWeb</servlet-name>
        <servlet-class>
            org.springframework.web.servlet.DispatcherServlet
```

```

        </servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>HelloWeb</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

</web-app>

```

Following is the content of another Spring Web configuration file **HelloWeb-servlet.xml**

```

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    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.tutorialspoint" />

    <bean >
        <property name="prefix" value="/WEB-INF/jsp/" />
        <property name="suffix" value=".jsp" />
    </bean>

</beans>

```

Following is the content of Spring view file **hello.jsp**

```

<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head>
<title>Hello World</title>
</head>
<body>
    <h2>${message}</h2>
</body>
</html>

```

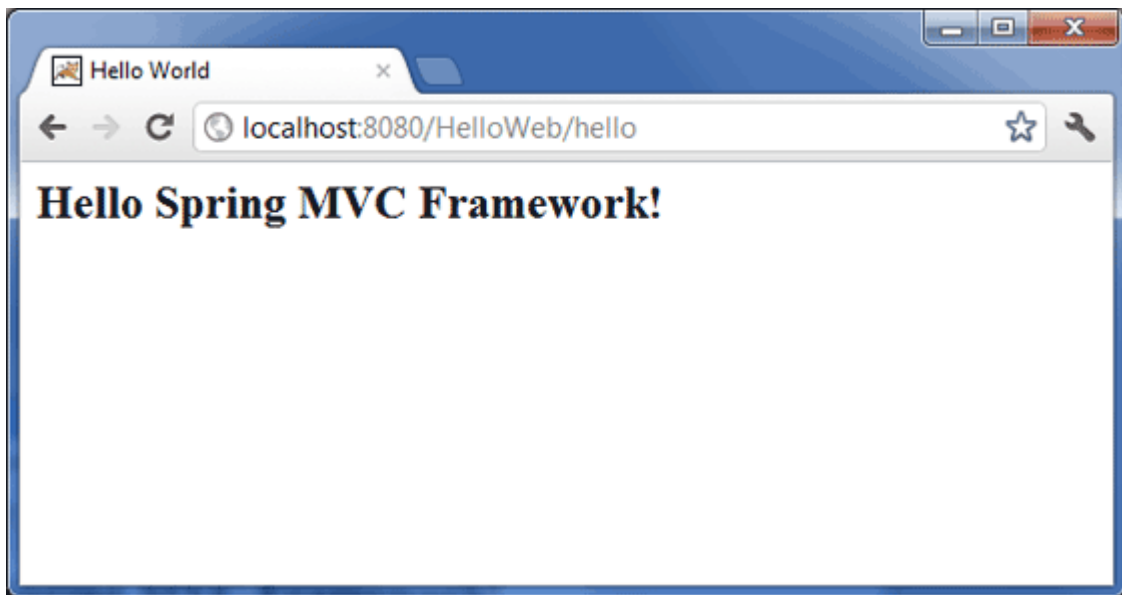
Finally, following is the list of Spring and other libraries to be included in your web application. You simply drag these files and drop them in **WebContent/WEB-INF/lib** folder.

- commons-logging-x.y.z.jar
- org.springframework.asm-x.y.z.jar
- org.springframework.beans-x.y.z.jar
- org.springframework.context-x.y.z.jar
- org.springframework.core-x.y.z.jar
- org.springframework.expression-x.y.z.jar
- org.springframework.web.servlet-x.y.z.jar
- org.springframework.web-x.y.z.jar
- spring-web.jar

Once you are done with creating source and configuration files, export your application. Right click on your application and use **Export > WAR File** option and save your **HelloWeb.war** file in Tomcat's *webapps* folder.

Now start your Tomcat server and make sure you are able to access other web pages from webapps folder using a standard browser. Now try to access the URL

http://localhost:8080/HelloWeb/hello and if everything is fine with your Spring Web Application, you should see the following result:



You should note that in the given URL, **HelloWeb** is the application name and **hello** is the virtual subfolder which we have mentioned in our controller using `@RequestMapping "/hello"`. You can use direct root while mapping your URL using `@RequestMapping "/"`, in this case you can access the same page using short URL **http://localhost:8080/HelloWeb/** but it is advised to have different functionalities under different folders.

Loading [Mathjax]/jax/output/HTML-CSS/jax.js