Spring MVC Multiple submit buttons in a single form

April 29, 2015 by **javainterviewpoint** **4 Comments**

Recently i came across a sitution where in which i needed to have two submit buttons in a Single **Spring MVC form** and have seperate action mapped to each of them. Previously we had been using the below approach in my application.

```
 <form action="dosomething">
     <input type='submit' name='action' value='action1' />
    <input type='submit' name='action' value='action2' />
 </form>
```

The controller part look like

```
public class Controller
{
        public String doSomething()
        {
                String action= request.getParameter("action");
                if(action == "action1")
                {
                        dosomething
                }
                else if(action == "action2")
                {
                        do something else
                }
        }
}
```

Now here comes the question how to achieve the same in Spring MVC. We will have to use an attribute called as **params** in the **@RequestMapping** to get the value of the button

Spring MVC Form

```
<input type = "submit" name = "action1" />
<input type = "submit" name = "action2" />
```

Spring Controller

```
@RequestMapping(params = "action1")
public ModelAndView action1(...)

@RequestMapping(params = "action2")
public ModelAndView action2(....)
```

Lets look into the complete example

Folder Structure :

1. Create a *Dynamic Web Project* **"SpringMVCMultipleSubmitButton"** and create a package for our src files **"*com.javainterviewpoint*"**
2. Place the Spring 3 jar files under *WEB-INF/Lib*

commons-logging-1.1.1.jar
log4j-1.2.16.jar
slf4j-api-1.7.5.jar
slf4j-log4j12-1.7.5.jar
spring-aspects-3.2.4.RELEASE.jar
spring-beans-3.2.4.RELEASE.jar
spring-context-3.2.4.RELEASE.jar
spring-core-3.2.4.RELEASE.jar
spring-expression-3.2.4.RELEASE.jar
spring-web-3.2.4.RELEASE.jar
spring-webmvc-3.2.4.RELEASE.jar

3. Create the Java classes ***SpringMVCController.java*** under *com.javainterviewpoint* folder.
4. Place the ***SpringConfig-servlet.xml*** *and* ***web.xml*** under the *WEB-INF directory*
5. View files ***welcome.jsp*** are put under the sub directory under *WEB-INF/Jsp*

welcome.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
 pageEncoding="ISO-8859-1"%>
 <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
   <form:form action="processForm" method="post">
     <input type = "submit" name = "action1" value="Action1"/>
     <input type = "submit" name = "action2" value="Action2"/>
   </form:form>
</body>
</html>
```

our welcome page has nothing but two submit buttons with names assigned as **"action1"**and **"action2"**

SpringMVCController.java

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

@Controller
public class SpringMVCController
```

```
{
    @RequestMapping("/showForm")
    public String showForm()
    {
        return "welcome";
    }

@RequestMapping(value="/processForm",params="action1",method=RequestMethod.
POST)
    public void action1()
    {
        System.out.println("Action1 block called");
    }

@RequestMapping(value="/processForm",params="action2",method=RequestMethod.
POST)
    public void action2()
    {
        System.out.println("Action2 block called");
    }
}
```

Our controller class has the **RequestMapping** corresponding to the action from the form **"processForm"** and **param** attribute which corresponds to the name of the each button. so when 1st button is clicked **action1()** method will be called and when 2nd button is clicked **action2()** method will be called.

web.xml

The web.xml has everything about the application that a server needs to know, which is placed under the WEB-INF directory. *<servlet-name>* contains the name of the SpringConfiguration , when the *DispatcherServlet* is initialized the framework will try to load a configuration file "*[servlet-name]-servlet.xml"* under the WEB-INF directory.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID"
version="2.5">
   <display-name>SpringMVCFormHandling</display-name>
   <welcome-file-list>
       <welcome-file>index.html</welcome-file>
       <welcome-file>index.htm</welcome-file>
       <welcome-file>index.jsp</welcome-file>
       <welcome-file>default.html</welcome-file>
       <welcome-file>default.htm</welcome-file>
       <welcome-file>default.jsp</welcome-file>
   </welcome-file-list>
   <servlet>
       <servlet-name>SpringConfig</servlet-name>
       <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>SpringConfig</servlet-name>
      <url-pattern>/</url-pattern>
```

```
   </servlet-mapping>
</web-app>
```

SpringConfig-servlet.xml

The **SpringConfig-servlet.xml** is also placed under the WEB-INF directory.

- **<context:component-scan>** will let the Spring Container to **search** for all the annotation under the package “com.javainteriviewpoint”.
- **<mvc:annotation-driven/>** annotation will activate the **@Controller, @RequestMapping, @Valid** etc annotations.
- The view is resolved through “org.springframework.web.servlet.view.InternalResourceViewResolver” which searches for the jsp files under the /WEB-INF/Jsp/ directory.

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:mvc="http://www.springframework.org/schema/mvc"
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
 http://www.springframework.org/schema/mvc
 http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">

    <mvc:annotation-driven/>

    <context:component-scan base-
package="com.javainterviewpoint"></context:component-scan>

    <bean id="viewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/Jsp/"></property>
        <property name="suffix" value=".jsp"></property>
     </bean>
 </beans>
```

Output

URL : *http://localhost:8080/SpringMVCMultipleSubmitButton/processForm*

when 1st button is clicked,**action1()** method is called

```
Action1 block called
```

when 2nd button is clicked,**action2()** method is called

```
Action2 block called
```

Related posts:

1. **Spring MVC Form Validation Tutorial (With Annotations and ResourceBundle)**
2. **Spring MVC Form Validation Tutorial (With Annotations)**

3. **Spring 4 – Spring MVC Hello World Example**
4. **Spring MVC Form Handling Example**