

STRUTS 2 & SPRING INTEGRATION

http://www.tutorialspoint.com/struts_2/struts_spring.htm

Copyright © tutorialspoint.com

Spring is a popular web framework that provides easy integration with lots of common web tasks. So the question is, why do we need Spring when we have Struts2? Well, Spring is more than a MVC framework - it offers many other goodies which are not available in Struts. For example: dependency injection that can be useful to any framework. In this chapter, we will go through a simple example to see how to integrate Spring and Struts2 together.

First of all, you need to add the following files to the project's build path from Spring installation. You can download and install latest version of Spring Framework from <http://www.springsource.org/download>

- org.springframework.asm-x.y.z.Ma.jar
- org.springframework.beans-x.y.z.Ma.jar
- org.springframework.context-x.y.z.Ma.jar
- org.springframework.core-x.y.z.Ma.jar
- org.springframework.expression-x.y.z.Ma.jar
- org.springframework.web-x.y.z.Ma.jar
- org.springframework.web.servlet-x.y.z.Ma.jar

Finally add **struts2-spring-plugin-x.y.z.jar** in your **WEB-INF/lib** from your struts lib directory. If you are using Eclipse then you may face an exception *java.lang.ClassNotFoundException: org.springframework.web.context.ContextLoaderListener*. To fix this problem, you should have to go in **Marker** tab and right click on the class dependencies one by one and do Quick fix to publish/export all the dependencies. Finally make sure there is no dependency conflict available under the marker tab.

Now let us setup the **web.xml** for the Struts-Spring integration as follows:

```
<?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_3_0.xsd"
  >

  <display-name>Struts 2</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

  <listener>
    <listener-class>
      org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>

  <filter>
    <filter-name>struts2</filter-name>
    <filter-class>
      org.apache.struts2.dispatcher.FilterDispatcher
    </filter-class>
  </filter>

  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
```

```
</filter-mapping>

</web-app>
```

The important thing to note here is the listener that we have configured. The **ContextLoaderListener** is required to load the spring context file. Spring's configuration file is called **applicationContext.xml** file and it must be placed at the same level as the **web.xml** file

Let us create a simple action class called **User.java** with two properties - firstName and lastName.

```
package com.tutorialspoint.struts2;

public class User {
    private String firstName;
    private String lastName;

    public String execute()
    {
        return "success";
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
}
```

Now let us create the **applicationContext.xml** spring configuration file and instantiate the **User.java** class. As mentioned earlier, this file should be under the WEB-INF folder:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
<beans>
    <bean >
        <property name="firstName" value="Michael" />
        <property name="lastName" value="Jackson" />
    </bean>
</beans>
```

As seen above, we have configured the user bean and we have injected the values **Michael** and **Jackson** into the bean. We have also given this bean a name "userClass", so that we can reuse this elsewhere. Next let us create the **User.jsp** in the WebContent folder:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
<title>Hello World</title>
</head>
<body>
```

```

<h1>Hello World From Struts2 - Spring integration</h1>

<s:form>
  <s:textfield name="firstName" label="First Name"/><br/>
  <s:textfield name="lastName" label="Last Name"/><br/>
</s:form>

</body>
</html>

```

The **User.jsp** file is pretty straight forward. It serves only one purpose - to display the values of the firstname and lastname of the user object. Finally, let us put all entities together using the **struts.xml** file.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
  <constant name="struts.devMode" value="true" />
  <package name="helloworld" extends="struts-default">
    <action name="user"
      method="execute">
      <result name="success">/User.jsp</result>
    </action>
  </package>
</struts>

```

The important thing to note is that we are using the id **userClass** to refer to the class. This means that we are using spring to do the dependency injection for the User class.

Now right click on the project name and click **Export > WAR File** to create a War file. Then deploy this WAR in the Tomcat's webapps directory. Finally, start Tomcat server and try to access URL <http://localhost:8080/HelloWorldStruts2/User.jsp>. This will give you following screen:

We have now seen how to bring two great frameworks together. This concludes the Struts - Spring integration chapter.

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