Struts 2 – Tag Library

Simple UI Tags

```
<%@ 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>
<s:head/>
<title>Hello World</title>
</head>
<body>
  <s:div>Email Form</s:div>
  <s:text name="Please fill in the form below:" />
  <s:form action="hello" method="post" enctype="multipart/form-data">
  <s:hidden name="secret" value="abracadabra"/>
  <s:textfield key="email.from" name="from" />
  <s:password key="email.password" name="password" />
  <s:textfield key="email.to" name="to" />
  <s:textfield key="email.subject" name="subject" />
  <s:textarea key="email.body" name="email.body" />
  <s:label for="attachment" value="Attachment"/>
  <s:file name="attachment" accept="text/html,text/plain" />
  <s:token />
  <s:submit key="submit" />
  </s:form>
</body>
</html>
```

If you are aware of HTML then all the tags used are very common HTML tags with an additional prefix **s:** alongwith each tag and different attributes. When we execute the above program, we get the following user interface provided you have setup proper mapping for all the keys used.

s:head generates the javascript and stylesheet elements required for the Struts2 application.

The s:div is used to render a HTML Div element. This is useful for people who do not like to mix HTML and Struts tags together.

The s:text as shown is used to render a text on the screen.

The s:form tag has an action attribute that determines where to submit the form. Because we have a file upload element in the form, we have to set the enctype to multipart. Otherwise, we can leave this blank.

The s:submit tag - This is used to submit the form. When the form is submitted, all the form values are submitted to the the action specified in the s:form tag

Inside the s:form, we have a hidden attribute called secret. This renders a hidden element in the HTML. In our case, the "secret" element has the value "abracadabra". This element is not visible to the end user and is used to carry the state from one view to another.

Next we have the s:label, s:textfield, s:password and s:textarea tags. These are used to render the label, input field, password and the text area respectively. The important thing to note here is the use of "key" attribute. The "key" attribute is used to fetch the label for these controls from the property file.

Then, we have the s:file tag which renders a input file upload component. This component allows the user to upload files. In this example, we have used the "accept" paramter of the s:file tag to specify which file types are allowed to be uploaded.

Finally we have the s:token tag. The token tag generates an unique token which is used to find out whether a form has been double submitted.

Group UI Tags

```
<%@ page contentType="text/html; charset=UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<html>
<head>
<title>Hello World</title>
<s:head />
</head>
<body>
  <s:form action="hello.action">
  <s:radio label="Gender" name="gender" list="{'male','female'}" />
  <s:checkboxlist label="Hobbies" name="hobbies"
  list="{'sports','tv','shopping'}" />
  </s:form>
</body>
</html>
```

Let us look at the example now. In the first example, we are creating a simple radiobutton with the label "Gender". The name attribute is mandatory for the radiobutton tag, so we specify a name which is "gender". We then supply a list to the gender. The list is populated with the values "male" and "female". Therefore, in the output we get a radiobutton with two values in it.

In the second example, we are creating a checkbox list. This is to gather the user's hobbies. The user can have more than one hobby and therefore we are using the checkbox instead of the radiobutton. The checkbox is populated with the list "sports", "Tv" and "Shopping". This presents the hobbies as a checkbox list.

Select UI Tags

Let us explore the different variations of the Select Tag offered by Struts. Let us look a simple view page **HelloWorld**.jsp with select tags:

```
<%@ page contentType="text/html; charset=UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<html>
```

```
<head>
<title>Hello World</title>
<s:head />
</head>
<body>
  <s:form action="login.action">
    <s:select name="username" label="Username"
      list="{'Mike','John','Smith'}" />

    <s:select label="Company Office" name="mySelection"
      value="%{'America'}"
      list="%{#{'America':'America'}}">
    <s:optgroup label="Asia"
      list="%{#{'India':'India','China':'China'}}" />
    <s:optgroup label="Europe"
      list="%{#{'UK':'UK','Sweden':'Sweden','Italy':'Italy'}}" />
    </s:select>

    <s:combobox label="My Sign" name="mySign"
      list="#{'aries':'aries','capricorn':'capricorn'}"
      headerKey="-1"
      headerValue="--- Please Select ---" emptyOption="true"
      value="capricorn" />
    <s:doubleselect label="Occupation" name="occupation"
      list="{'Technical','Other'}" doubleName="occupations2"
      doubleList="top == 'Technical' ?
      {'I.T', 'Hardware'} : {'Accounting', 'H.R'}" />

  </s:form>
</body>
</html>
```

The if and else tags

These tags perform basic condition flow found in every language. 'If' tag could be used by itself or with 'Else If' Tag and/or single/multiple 'Else' Tag as shown below:

```
<s:if test="%{false}">
   <div>Will Not Be Executed</div>
</s:if>
<s:elseif test="%{true}">
   <div>Will Be Executed</div>
</s:elseif>
<s:else>
   <div>Will Not Be Executed</div>
</s:else>
```

The iterator tags

These iterator will iterate over a value. An iterable value can be any of: java.util.Collection, java.util.Iterator. While iterating over an iterator, you can use Sort tag to sort the result orSubSet tag to to get a sub set of the list or array.

The following example retrieves the value of the getDays() method of the current object on the value stack and uses it to iterate over. The <s:property/> tag prints out the current value of the iterator.

```
<s:iterator value="days">
  <p>day is: <s:property/></p>
</s:iterator>
```