Struts Code Peaces – <html:option> element

We explain the struts <html:option> element and illustrate the usage with some small examples.

Generals

Author:
Sascha Wolski
Sebastian Hennebrueder
http://www.laliluna.de/tutorials.html – Tutorials for Struts, EJB, xdoclet and eclipse.

Date:
February 15th 2005

The <html:option> element

The *<html:option>* element is a part of the *<html:select>* element and must be nested inside this element. It can be used multiple times inside the *<html:select>* element. The *<html:option>* renders a HTML <option> element at runtime.

The following example shows the source code of the JSP file:

```
<html:select property="selectedItem">
      <html:option value="Marie">Marie</html:option>
      <html:option value="Klaus">Klaus</html:option>
</html:select>
```

The following HTML source code will be rendered.

```
<select name="selectedItem">
      <option value="Marie">Marie</option>
      <option value="Klaus">Klaus</option>
</select>
```

Attributes of the <html:option> element

Name	Description
bundle	Defines the key for the MessageResources instance to use.
disabled	If value is „true“, it disable the element.
key	If specified, defines the message key to be looked up in the message resource bundle specified by the bundle for the text displayed to the user for this option. If not specified, the text to be displayed is taken from the body content of the tag.
locale	Defines the local to be used for the message keys, specified by the key attribute. If not specified, uses the standard local of the session.
style	CSS styles for the element
styleId	Identifier of the HTML element. (render a id attribute)
styleClass	CSS stylesheet Class for this element (render a class attribute)
value	Specified the value, which is submitted to the server if the user select this option

Usage of <html:option> element

We illustrate the usage of the *<html:option>* element with some examples. Create a new project with an action class, an action form class and a JSP file.

Create a new action class

Create a new class *ExampleAction* in the package *de.laliluna.tutorial.option.action.*
We create some dummy data for an array and a collection. This data will be used later in the JSP

file.

```
public class ExampleAction extends Action {

      public ActionForward execute(
            ActionMapping mapping,
            ActionForm form,
            HttpServletRequest request,
            HttpServletResponse response) {
            ExampleForm selectForm = (ExampleForm) form;

            //create an dummy array
            String[] array = new String[3];
            array[0] = "Maria";
            array[1] = "Klaus";
            array[2] = "Peter";

            //create a dummy collection
            Collection collection = new ArrayList();
            collection.add("Maria");
            collection.add("Klaus");
            collection.add("Peter");

            //set it in the request
            request.setAttribute("array", array);
            request.setAttribute("collection", collection);

            return mapping.findForward("success");
      }
}
```

Create a new FormBean

Create a new action form class *ExampleForm* in the package *de.laliluna.tutorial.option.form*.

Add a property *selectedItem* of type string, which refers to the <*html:select*>. This property contains the value of the selected option in our select box.

Create a getter and setter method for the property.

```
public class ExampleForm extends ActionForm {

      private String selectedItem;

      public String getSelectedItem() {
            return selectedItem;
      }
      public void setSelectedItem(String selectedItem) {
            this.selectedItem = selectedItem;
      }
}
```

Create the struts-config.xml

Open the struts-config.xml and define the form bean and the action mapping.

```
<struts-config>
   <form-beans>
      <form-bean name="exampleForm"
type="de.laliluna.tutorial.option.form.ExampleForm" />
   </form-beans>

   <action-mappings>
      <action
         name="exampleForm"
         path="/example"
         scope="request"
```

```
         type="de.laliluna.tutorial.option.action.ExampleAction">
         <forward name="success" path="/form/example.jsp" />
      </action>
   </action-mappings>
</struts-config>
```

Create a JSP file

Create a JSP file *example.jsp* in the folder */WebRoot/form/*

Open the JSP file and add the following HTML source code.

```
<%@ page language="java"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html" prefix="html"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic" prefix="logic"%>

<html>
      <head>
            <title>example.jsp</title>
      </head>
      <body>
            <html:form action="/example">

            .... sample code ...

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

Add the first example inside the *<html:form>* element.

Example 1

The attribute *property* of *<html:select>* element refers to the property *selectedItem* in our form bean, which holds the value after the form is submitted. The value to be submitted is specified by the value attribute of the *<html:option>* tag. If the user chooses an option, for example option two, the value „Klaus“ is set in the property selectedItem of the form bean. The text displayed to the user is specified in the body of the element.

```
<h4>Simple use &lt;html:option&gt; Tag</h4>

<html:select property="selectedItem">
      <html:option value="Marie">Marie</html:option>
      <html:option value="Klaus">Klaus</html:option>
      <html:option value="Peter">Peter</html:option>
</html:select>

<html:submit/>
```

Example 2

In the example we use an array, which is defined in the action class and saved in the request scope. It is possible to display dynamic conents in the HTML select box. With the <logic:iterate> you can iterate over the content of the array and fill the <html:option> element. The variable var holds the current value of the iteration, specified by the attribute id in the <logic:iterate> tag.

```
<h4>Use &lt;html:option&gt; Tag within an iteration over an array</h4>

<html:select property="selectedItem">
      <logic:iterate name="array" id="var" type="java.lang.String">
            <html:option value="<%= var %>">
                  <bean:write name="var" />
            </html:option>
```

```
      </logic:iterate>
</html:select>

<html:submit/>
```

Example 3

The last example shows you an iteration over a collection. The JSP source code is the same we used to iterate over an array.

```
<h4>Use &lt;html:option&gt; Tag within an iteration over a collection</h4>

<html:select property="selectedItem">
      <logic:iterate name="collection" id="var" type="java.lang.String">
            <html:option value="<%= var %>">
                  <bean:write name="var" />
            </html:option>
      </logic:iterate>
</html:select>

<html:submit/>
```

Now you can test the project. We use a JBOSS or Tomcat installation. Call the project with the following link.

http://localhost:8080/OptionTag/example.do