

Struts Code Peaces – <html:optionscollection> Element

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

Generals

Author:

Sascha Wolski

Sebastian Hennebrueder

<http://www.laliluna.de/tutorials.html> – Tutorials für Struts, EJB, xdoclet und eclipse.

Date:

February 22th 2005

The <html:optionscollection> element

The <html:optionscollection> element is only valid nested inside a <html:select> element. It renders a HTML <option> element. The <html:optionscollection> element is used to output lists (arrays, collections) for a HTML select field. This tag differs from the <html:options> tag in that it makes more consistent use of the *name* and *property* attributes, and allows the collection to be more easily obtained from the enclosing form bean.

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

```
<html:select property="selectedItem">
    <html:optionscollection="customers" value="id" label="name" />
</html:select>
```

The following HTML source code is rendered at runtime:

```
<select name="selectedItem">
    <option value="1">Marie</option>
    <option value="2">Klaus</option>
</select>
```

Attributes of the <html:optionscollection> element

We list the most important attributes below. You find a complete list of all available attributes for this tag in the API of the HTML tag library.

<http://struts.apache.org/userGuide/struts-html>

Name	Description
label	Specify the property within the collection, which holds the label for this element. If not specified struts look in the bean for a getter <code>getLabel()</code>
name	Name of the collection in the request or the name of the scope, which holds the collection.
property	Name of the associated collection of the form bean.
Value	Specify the property of the bean within the collection, which holds the value.

Usage of the <html:optionscollection> element

We will create a new struts project to illustrate the usage of the <html:optionscollection> element.

Create an object class

Create a new java class Customer in the package `de.laliluna.tutorial.optionscollection.object`.

Define two properties, *id* of type int and *name* of type String. Add a getter and setter method for each of them.

Define a constructor which allows you to set the properties when initializing the class.

If the attributes *value* and *label* of the *<html:optioncollection>* tag are not used, Struts tries to find a getter method for a value and a label attribute (*getValue()*, *getLabel()*). Create both getter methods. The method *getValue()* returns the property id and the method *getLabel()* returns the property name of the form bean.

The object class looks like the following:

```
public class Customer {  
  
    private int id;  
    private String name;  
  
    public Customer() {}  
  
    public Customer(int id, String name) {  
        this.id = id;  
        this.name = name;  
    }  
    public int getId() {  
        return id;  
    }  
    public void setId(int id) {  
        this.id = id;  
    }  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public String getLabel(){  
        return this.name;  
    }  
  
    public int getValue(){  
        return this.id;  
    }  
}
```

Create a new action form class

Create an action form class *ExampleForm* in the package *de.laliluna.tutorial.optionscollection.form*.

Define a property *selectedItem* of type String. This property will be associated with the *<html:select>* element.

The following source code shows the action form class

```
public class ExampleForm extends ActionForm {  
  
    private String selectedItem;  
  
    public String getSelectedItem() {  
        return selectedItem;  
    }  
    public void setSelectedItem(String selectedItem) {  
        this.selectedItem = selectedItem;  
    }  
}
```

Create a new action class

Create a action class *ExampleAction* in the package *de.laliluna.tutorial.optionscollection.action*.

We provide some dummy data as collection.

The following source code shows the action class:

```
public class ExampleAction extends Action {  
  
    public ActionForward execute(  
        ActionMapping mapping,  
        ActionForm form,  
        HttpServletRequest request,  
        HttpServletResponse response) {  
        ExampleForm exampleForm = (ExampleForm) form;  
  
        //define a dummy collection  
        Collection customers = new ArrayList();  
        customers.add(new Customer(1, "Marie"));  
        customers.add(new Customer(2, "Klaus"));  
        customers.add(new Customer(3, "Peter"));  
  
        //set the collection in the request  
        request.setAttribute("customers", customers);  
  
        return mapping.findForward("success");  
    }  
}
```

Create the struts-config.xml

Now open the struts-config.xml and specify the form bean and the action mapping.

```
<struts-config>  
    <form-beans>  
        <form-bean name="exampleForm"  
        type="de.laliluna.tutorial.optionscollection.form.ExampleForm" />  
    </form-beans>  
  
    <action-mappings>  
        <action  
            name="exampleForm"  
            path="/example"  
            scope="request"  
            type="de.laliluna.tutorial.optionscollection.action.ExampleAction">  
            <forward name="success" path="/form/example.jsp" />  
        </action>  
    </action-mappings>  
</struts-config>
```

Create the JSP file

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

Open the JSP file *example.jsp* and add the following 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>
```

Within the `<html:form>` element add the first example.

Example 1

In the first example we output the collection, which was saved in the request (inside the action class). The attribute `property` of the `<html:select>` element specifies the associated property `selectedItem` of the form bean. This property holds the value of the selected option after submitting the form. The attribute `name` of the `<html:optioncollection>` tag specifies the `name` of the collection. We do not use the attributes `value` and `label`, so Struts will search for the default getter methods (`getValue()` and `getLabel()`). If this methods do not exists an error occurs.

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

<html:select property="selectedItem">
    <html:optionsCollection name="customers" />
</html:select>

<html:submit/>
```

Example 2

The next example shows the usage of the attributes `value` and `label`. The default getter methods (`getValue` and `getLabel()`) do not have to be specified, when the two attributes are set in the `<html:optionscollection>` tag. Set the attribute `label` to `id` and the attribute `value` to `name`.

```
<h4>Use value and label attribute of the &lt;html:options&gt; Tag</h4>

<html:select property="selectedItem">
    <html:optionsCollection name="customers" label="id" value="name"/>
</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/OptionsCollectionTag/example.do>