

Struts Code Peaces – <html:checkbox> element

We explain the struts <html:checkbox> 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

Development Tools

Eclipse 3.x

The <html:checkbox> element

The <html:checkbox> element is used with an associated property of typ boolean. It renders a HTML <input> element of typ checkbox.

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

```
<html:checkbox property="checked">Label</html:checkbox>
```

The following HTML source code is rendered at runtime.

```
<input type="checkbox" name="checked" value="on">Label
```

Attributes of the <html:checkbox> element

Now the most important attributes will be explained. 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.html#checkbox>

Name	Beschreibung
disabled	Disable the element (true / false)
indexed	Only valid inside a logic:iterate tag. If the value is true, the name of the HTML element is rendered as „id[3].property“. id = scope of the property, property = property of the scope. The number inside the brackets is the current index of the iteration.
name	Name of the bean which contains the properties.
property	Name of the property or request parameter, which is associated with this element.
value	The value which is submitted to the server. If not specified the default value „on“ will be set as the value. If specified the value have to match one of these strings („on“, „true“, „yes“)

Usage of the <html:checkbox> element

Create a new struts project to illustrate the usage of the <html:checkbox> element.

Create a object class

Create a new java class customer in the package *de.laliluna.tutorial.checkbox.object*. This class represents a customer.

Add two properties, name of type string and checked of type boolean and generate a getter and setter method for each of them.

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

The object class looks like the following:

```
public class Customer {

    private String name;
    private boolean checked;

    public Customer() {}

    public Customer(String name, boolean checked) {
        this.name = name;
        this.checked = checked;
    }

    public boolean isChecked() {
        return checked;
    }

    public void setChecked(boolean checked) {
        this.checked = checked;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

}
```

Create a new action class

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

```
public class ExampleAction extends Action {

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

        return mapping.findForward("success");
    }

}
```

Create a new form bean

Provide a new action form class *ExampleForm* in the package *de.laliluna.tutorial.checkbox.form*.

Define a property *checked* of type boolean. This property holds the value of the associated `<html:checkbox>` element.

Define a second property *customers* of type Collection. The collection holds a list of customers to show you the usage of a `<html:checkbox>` element inside a list of data.

Implement the method *reset()* of the action form class and initialize the property *checked* to false. Within the method *reset()* we provide some dummy data for the property *customers* of the form-bean.

The following source code shows the action form class.

```
public class ExampleForm extends ActionForm {

    private boolean checked;
    private Collection customers;

    public boolean isChecked() {
```

```

        return checked;
    }
    public void setChecked(boolean checked) {
        this.checked = checked;
    }
    public Collection getCustomers() {
        return customers;
    }
    public void setCustomers(Collection customers) {
        this.customers = customers;
    }
}

/**
 * reset method
 * @param mapping
 * @param request
 */
public void reset(ActionMapping mapping,
                  HttpServletRequest request) {

    //reset properties
    this.checked = false;

    //initial the customers collection
    customers = new ArrayList();
    customers.add(new Customer("Marie", false));
    customers.add(new Customer("Klaus", false));
    customers.add(new Customer("Peter", false));

}
}

```

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.checkbox.form.ExampleForm" />
  </form-beans>

  <action-mappings>
    <action
      name="exampleForm"
      path="/example"
      scope="request"
      type="de.laliluna.tutorial.checkbox.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

The first example shows a checkbox element, which status can be checked and unchecked. The attribute *property* specifies the associated property *checked* of the form bean. If the user mark the checkbox and submits the form, the submitted value of the checkbox („on“, „true“, „yes“) will be converted into a value of type boolean and set in the property *checked* of the form bean.

```

<h4>Simple use of <html:checkbox> Tag</h4>
<html:checkbox property="checked">Label</html:checkbox>
<br /><br />
<html:submit property="btnApply"/>

```

Example 2

The second example shows the usage of a `<html:checkbox>` element within a `<logic:iterate>` tag. The attribute *name* of the `<logic:iterate>` element specifies the bean which holds the collection, in our case the form bean *exampleForm*. The attribute *property* specifies the name of the collection within the form bean. The attribute *id* specifies the name of the variable that will contain the current element of the collection on each iteration.

Note: In order to refer a property of an element within the collection to a `<html:checkbox>` element the attribute *id* of the `<logic:iterate>` tag must have the same name like the collection of the form bean. Furthermore the attribute *indexed* of the `<html:checkbox>` have to be set to „true“.

The value of the attribute *name* within the `<html:checkbox>` element refers to the variable which is specified by the attribute *id* of the `<logic:iterate>` tag.

The attribute *property* sets the associated property *checked* of the object class. The *indexed* attribute of the `<html:checkbox>` element must be set to „true“, because the element is inside a `<logic:iterate>` tag.

```

<h4>Use of <html:checkbox> Tag inside a <logic:iterate></h4>
<logic:iterate name="exampleForm" property="customers" id="customers">
    <html:checkbox name="customers" property="checked" indexed="true">
        <bean:write name="customers" property="name" />
    <br />
</html:checkbox>
</logic:iterate>
<br />
<html:submit property="btnApply"/>

```

Now you can test the project. Deploy the project and test it with the following link.

<http://localhost:8080/CheckboxTag/example.do>