Struts Nested Tags

Since the version 1.1 of Struts, the tag library “nested” is included in Struts. In this tutorial we want to explain what are the features of the new nested tag library and show some little examples how you can use it.

General

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

Date:
February, 29th 2004

Software:
Eclipse 3.x
MyEclipse 3.8.x

Source code:
Source code

The nested tag library

Nested tags make it easy to manage nested beans. For example a list like the following:

Department A
Customer A
Customer B
Customer C
Department B
Customer D

All nested tags inside a nested tag refers to the tag which surrounds them. For example your form bean holds an object *department* and you want the property *name* of the department. Normally you can use a dot notation to get the name of the department.

With a *nested:nest* tag you do not need the dot notation any more.

The following example shows a dot notation to output the name of a department bean:

```
<nested:write property="department.id" />
```

This example shows the usage of a *nested:nest* element to output the same property.

```
<nested:nest property="department">
      <nested:write property="id" />
</nested:nest>
```

The tags inside the *nested:nest* can refers directly to the properties of the department. When you have many properties this is quite an advantage.

The most functionality elements of the other tag libraries are rebuild as nested tags, like *bean:write* is *nested:write*, *logic:iterate* is *nested:iterate* etc. You will find a complete list of all supported tags by the nested tab library in the Apache Struts Nested Tag Library API.

Usage of the nested tags

Create a new struts 1.1 project to get more familiar with the nested tags.

Add a package *de.laliluna.tutorial.nested* to the *src* folder of the project.

Object class Customer

Create a new java class *Customer* in the package *de.laliluna.tutorial.nested.object*.

Add two properties, *id* of type int and *name* of type String.

Provide a getter and setter method for each property.

Create a constructor that allow you to set the properties on initialization.

The following source code shows the class *Customer* :

```
/**
 * Object Class Customer
 */
public class Customer {

	private int id;
	private String name;

	//constructors
	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;
	}
}
```

Object class Department

Create a second java class *Departments* in the same package *de.laliluna.tutorial.nested.object*.

Add two properties, *id* of type int and *name* of type String and one property *customers* of type Collection, which holds a list of customers.

Provide a getter and setter method for each property.

Create a constructor that allows you to set the properties on initialization.

The following source code shows the class *Department* :

```
/**
 * Object Class Department
 */
public class Department {

	private int id;
	private String name;
```

```
	//customers collection
	private Collection customers;

	//constructors
	public Department() {}
	public Department(int id, String name, Collection customers){
		this.id = id;
		this.name = name;
		this.customers = customers;
	}

	public Collection getCustomers() {
		return customers;
	}
	public void setCustomers(Collection customers) {
		this.customers = customers;
	}
	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;
	}
}
```

Action form class ExampleForm

Create a java class *ExampleForm* in the package *de.laliluna.tutorial.nested.form*, which extends the class *ActionForm* of struts.

Add a property *department* of type Department.

Provide a getter and setter method for the property *department*.

Implement the *reset()* method of the *ActionForm* class and provide some dummy data.

The following source code shows the class *ExampleForm*:

```
/**
 * Action Form Class ExampleForm
 */
public class ExampleForm extends ActionForm {

	Department department;

	public Department getDepartment() {
		return department;
	}
	public void setDepartment(Department department) {
		this.department = department;
	}

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

		//initial a dummy collection of customers
		Collection customers = new ArrayList();
```

```
            customers.add(new Customer(1, "Maria"));
            customers.add(new Customer(2, "Klaus"));
            customers.add(new Customer(3, "Peter"));

            //initial a dummy department
            department = new Department(1, "Department A", customers);

      }
}
```

Action class ExampleAction

Create a java class *ExampleAction* in the package *de.laliluna.tutorial.nested.action*, which extends the class *Action* of struts.

Return the forward *example*.

The following source code shows the class *ExampleAction*:

```
/**
 * Action Class ExampleAction
 */
public class ExampleAction extends Action {

      /**
       * Method execute
       * @param mapping
       * @param form
       * @param request
       * @param response
       * @return ActionForward
       */
      public ActionForward execute(
            ActionMapping mapping,
            ActionForm form,
            HttpServletRequest request,
            HttpServletResponse response) {
            ExampleForm exampleForm = (ExampleForm) form;

            return mapping.findForward("example");
      }

}
```

Configure the struts-config.xml

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

The following source code shows the content of the *struts-config.xml*.

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

   <action-mappings >
      <action
         attribute="exampleForm"
         input="/form/example.jsp"
         name="exampleForm"
         path="/example"
         scope="request"
         type="de.laliluna.tutorial.nested.action.ExampleAction"
         validate="false" >
         <forward name="example" path="/form/example.jsp" />
      </action>
```

```
   </action-mappings>
</struts-config>
```

The JSP file

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

Add the reference to the tag library nested at the top of the file.

The following source code shows the JSP file. The bold line at the top is the reference to the nested tag library.

```
<%@ 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-nested" prefix="nested"%>

<html>
      <head>
            <title>JSP for exampleForm form</title>
      </head>
      <body>
            <html:form action="/example">

                  ... example code ...

                  <br />
                  <br />
                  <html:submit/><html:cancel/>
            </html:form>
      </body>
</html>
```

Inside the *<html:form>* element you insert the following examples.

Example 1

In the first example we show the usage of dot notation to get the properties form the department object of the form bean.

```
<h3>Use of nested:text tag with dot notation</h3>

DEP. ID: <nested:text property="department.id" /> <br />
NAME: <nested:text property="department.name" /> <br />

<br />
```

Example 2

The second example shows the usage of the *nested:nest* tag to use the properties inside the *nested:nest* tag without dot notation. The attribute *property* of the nested:nest element refers to the property *department* of our form bean.

The *nested:text* elements inside the *nested:nest* element refers to the properties of the surrounds nested tag. The attribute property of the *nested:text* element refers to a property of the object class *Department*.

```
<h3>Use of nested:nest tag</h3>

<nested:nest property="department">
      DEP. ID: <nested:text property="id"/> <br />
      NAME: <nested:text property="name"/> <br />
</nested:nest>

<br />
```

Example 3

Example 3 shows the usage of an iteration inside a *nested:nest* tag. The *nested:iteration* element works like the *logic:iteration* element, but refers to a property of the parent nested tag.

```
<h3>Use of nested:iteration tag</h3>

<nested:nest property="department">
      DEP. ID: <nested:text property="id"/> <br />
      NAME: <nested:text property="name"/> <br /><br />
      <nested:iterate property="customers">
            <b>Customer info</b><br />
            CUST. ID: <nested:text property="id"/> <br />
            NAME: <nested:text property="name"/> <br />
      </nested:iterate>
</nested:nest>
```

You see that the usage of the nested tags is very comfortable and easy to understand. Now you can deploy your project to your favorite application server (we recommend jboss or tomcat) and call the project with the following link:

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