Struts – Multi Page Form

This tutorial shows how to create a multi page forms with struts. It explains two options to create a multi page form and shows how you can execute different validations for each step of the multi page form.

General

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

Date
Updated: December 2007
First edition: March 2004

Software:
Struts Framework 1.3

PDF download: http://www.laliluna.de/download/struts-multipageform-en.pdf
Source download: http://www.laliluna.de/download/struts-multipageform.zip

A Multi Page Form using an ActionForm Bean

The Workflow

The example application is having three pages. On the first page the user can input his name and age. After submitting the first page the name and the age will be validated. If the validation is correct the second pages will be loaded and the user can input his city and phone number. If the form is submitted only the city and phone number will be validated. If no errors occur the last page will be loaded and shows all informations.

New Struts Project

Create a new struts project named *MultipageFormExample*.

Add a package named *de.laliluna.tutorial.multipageform*.

ActionForm class ExampleForm

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

Add four properties, *name*, *age*, *city* and *phone* of type String.

Provide a getter and setter method for each property.

Implement the method *validate()* to validate your properties.

We use the same action form class for all pages, but we want to validate different properties for each page of our application. In the first page we want to validate the *name* and the *age* of the user, in the second page the *city* and the *phone* number.

To do this, we need a parameter *step* which identifies the page we are currently working on. This parameter will be saved in the request after submitting the page and holds a value which identifies the current page. Within the *validate()* method we get the parameter from the request and check the value of the parameter.

Note: You will find more informations about validating form properties and error handling in our tutorial Struts Validation and and Error Handling

The following source code shows the action form class *ExampleForm*:

```
public class ExampleForm extends ActionForm {
	// properties
	String name;
	Integer age;
	String city;
	String phone;

	public Integer getAge() {
		return age;
	}

	public void setAge(Integer age) {
		this.age = age;
	}

	public String getCity() {
		return city;
	}

	public void setCity(String city) {
		this.city = city;
	}

	public String getName() {
		return name;
	}

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

	public String getPhone() {
		return phone;
	}

	public void setPhone(String phone) {
		this.phone = phone;
	}

	public ActionErrors validate(ActionMapping mapping,
			HttpServletRequest request) {

		ActionErrors actionErrors = new ActionErrors();

		// get the hidden field to identify which
		// validation will be execute
		String step = request.getParameter("step");

		if (step != null) {
			// validation for step 1
			if (step.equals("1")) {

				// name must have at least 3 characters
				if (this.name.length() < 3) {
					actionErrors.add("name", new
ActionMessage("error.name"));
				}

				// age must be between 18 - 90
				if (age == null || age < 18 || age > 90) {
					actionErrors.add("age", new
ActionMessage("error.age"));
				}
			}
```

```
                  // vaidation for step 2
                  if (step.equals("2")) {

                        // city must have at least 2 characters
                        if (this.city.length() < 2) {
                              actionErrors.add("city", new
ActionMessage("error.city"));
                        }

                        // phone must have at least 5 characters
                        if (this.phone.length() < 5) {
                              actionErrors.add("phone", new
ActionMessage("error.phone"));
                        }
                  }
            }

            return actionErrors;
      }
}
```

Message Resource Bundle

Create a new text file *ApplicationResources.properties* in the package *de.laliluna.tutorial.multipageform*, which contains the error message keys.

Add the error keys we have assigned to the error messages in the method *validate()* of the action form class.

The following text shows the content of the file:

```
error.name=Name must have at least 3 characters
error.number=Age must be a number
error.age=Age must be between 18 and 90
error.city=City must have at least 2 characters
error.phone=Phone must have at least 5 characters
```

Action classes

Now create three java classes, *Step1*, *Step2* and *Finish* in the package *de.laliluna.tutorial.multipageform.action* Every class should extend the class Action.

The first source code shows the class *Step1*:

Within the method *execute(..)* we check a parameter *btnStep1* of the request. This parameter will be set by the submit button in the JSP file. If the submit button is pressed and the form is submitted the parameter *btnStep1* is not null and we want to forward to the *Step2* action. If the parameter *btnStep1* is not null change the *action forwards to step2*.

```
public class Step1 extends Action {

      public ActionForward execute(ActionMapping mapping,
            ActionForm form,
            HttpServletRequest request,
            HttpServletResponse response)
            throws Exception {

            String forward = "step1";
            if(request.getParameter("btnStep1") != null){
                  forward = "step2";
            }

            return mapping.findForward(forward);
```

```
	}
}
```

The source code below shows the class Step2:

Within the method *execute(..)* do the same like in the class Step1, but the parameter here is *btnStep2. If the* value is set, the action forwards to “finish” else it stays in step2.

```
public class Step2 extends Action {

	public ActionForward execute(ActionMapping mapping,
		ActionForm form,
		HttpServletRequest request,
		HttpServletResponse response)
		throws Exception {

		String forward = "step2";
		if(request.getParameter("btnStep2") != null){
			forward = "finish";
		}

		return mapping.findForward(forward);
	}
}
```

The last source code shows the class Finish:

The *methode(..)* only returns the forward *finish.*

```
public class Finish extends Action {

	public ActionForward execute(ActionMapping mapping,
		ActionForm form,
		HttpServletRequest request,
		HttpServletResponse response)
		throws Exception {

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

Configure the Struts config file

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

All action mappings use the same form bean.

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

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

   <action-mappings >
      <action
         attribute="exampleForm"
         input="/form/step1.jsp"
         name="exampleForm"
         path="/step1"
         scope="request"
         type="de.laliluna.tutorial.multipageform.action.Step1" >
         <forward name="step1" path="/form/step1.jsp" />
         <forward name="step2" path="/step2.do" />
      </action>
      <action
         attribute="exampleForm"
         input="/form/step2.jsp"
         name="exampleForm"
         path="/step2"
         scope="request"
         type="de.laliluna.tutorial.multipageform.action.Step2" >
         <forward name="step2" path="/form/step2.jsp" />
         <forward name="finish" path="/finish.do" />
      </action>

      <action
         attribute="exampleForm"
         input="/form/finish.jsp"
         name="exampleForm"
         path="/finish"
         scope="request"
         type="de.laliluna.tutorial.multipageform.action.Finish" >
         <forward name="finish" path="/form/finish.jsp" />
      </action>
   </action-mappings>

   <message-resources
parameter="de.laliluna.tutorial.multipageform.ApplicationResources" />
</struts-config>
```

Create the JSP files

We need three JSP files for the multi page form named *step1.jsp*, *step2.jsp* and *finish.jsp*.

Create the JSP files in the folder */WebRoot/form* of your project.

step1.jsp

Define two *html:text* elements associated with the properties *name* and *age*.

Add a *html:hidden* element *step* which identifies the current page and set the *value* to 1.

Define a *html:submit* element with the value *btnStep1* of the attribute *property*.

Add a *html:messages* element to output the error messages.

The following source code shows the JSP file *step1.jsp*

```
<%@ page language="java" pageEncoding="UTF-8"%>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<html>
      <head>
            <title>Step 1</title>
      </head>
      <body>
            <html:form action="/step1">

                  <%-- ouput errors --%>
                  <html:messages id="error" message="false">
                        <bean:write name="error" /> <br />
                  </html:messages>

                  <%-- input field for properties --%>
                  Name: <html:text property="name" /> <br />
                  Age: <html:text property="age" /> <br />

                  <%-- hidden field which specify the page --%>
                  <html:hidden property="step" value="1" />

                  <html:submit property="btnStep1"/>
            </html:form>
      </body>
</html>
```

step2.jsp

Define two *html:text* elements associated with the properties *city* and *phone*.

Define two html:hidden elements which associated with the properties *name* and *age*.

Note: If your form bean is saved in the session not in the request, you do not need to add the *html:hidden* elements associated with the properties *name* and *age*.

Add a *html:hidden* element *step* which identifies the current page and set the *value* to 2.

Define a *html:submit* element with the value *btnStep2* of the attribute *property*.

Add a *html:messages* element to output the error messages.

The following source code shows the JSP file *step2.jsp*

```
<%@ page language="java" pageEncoding="UTF-8"%>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<html>
      <head>
            <title>Step 2</title>
      </head>
      <body>
            <html:form action="/step2">

                  <%-- ouput errors --%>
                  <html:messages id="error" message="false">
                        <bean:write name="error" /> <br />
                  </html:messages>

                  <%-- input field for properties --%>
                  City: <html:text property="city" /> <br />
                  Phone: <html:text property="phone" /> <br />

                  <%-- hidden field which specify the page --%>
                  <html:hidden property="step" value="2" />

                  <%-- hidden fields for properties of step1 --%>
                  <html:hidden property="name" />
                  <html:hidden property="age" />

                  <html:submit property="btnStep2"/>
            </html:form>
      </body>
</html>
```

finish.jsp

Within the finish.jsp we want to output the form data.

Add four *bean:write* elements associated with the four properties of the form bean.

The following source code shows the JSP file *finish.jsp*:

```
<%@ page language="java" pageEncoding="UTF-8"%>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<html>
      <head>
            <title>Finish</title>
      </head>
      <body>
            <b>The values are:</b> <br /><br />
            Name = <bean:write name="exampleForm" property="name" /><br />
            Age = <bean:write name="exampleForm" property="age" /><br />
            City = <bean:write name="exampleForm" property="city" /><br />
            Phone = <bean:write name="exampleForm" property="phone" /><br />
      </body>
</html>
```

Now you can test the first way of creating a multi page form.

Call the project with the link below:

http://localhost:8080/MultipageFormExample/step1.do

A Multi Page Form using the ValidatorActionForm

Note: The ValidatorActionForm requires the version 1.2 of struts, to run successfully.

The next example shows the usage of the ValidatorActionForm class to create a multi page form with an validation based on XML files. The application does the same like the example application before, but it is a more elegant solution.

The Workflow

The example application has three pages. On the first page the user can input his name and age. After submitting the first page the name and the age will be validated. If the validation is correct the second pages will be loaded and the user can input his city and phone number. If the form is submitted only the city and phone number will be validated. If no errors occurred the last page is loaded and displays all informations.

New Struts Project

Create a new struts project named *MultipageFormValidatorExample*.

Add a package named *de.laliluna.tutorial.multipageform*.

ValidatorActionForm class ExampleForm

Create a new java class *ExampleForm* in the package *de.laliluna.tutorial.multipageform.form* which extends from the superclass *ValidatorActionForm*.

Add four properties, name, age, city and phone of type String.

Provide a getter and setter method for each property.

The following source code shows the class ExampleForm:

```
public class ExampleForm extends ValidatorActionForm{

      // properties
      String name;
      Integer age;
      String city;
      String phone;

      // getter and setter methods
      public String getCity() {
            return city;
      }
      public void setCity(String city) {
            this.city = city;
      }
      public String getName() {
            return name;
      }
      public void setName(String name) {
            this.name = name;
      }
      public String getPhone() {
            return phone;
      }
      public void setPhone(String phone) {
            this.phone = phone;
      }
      public Integer getAge() {
            return age;
      }
      public void setAge(Integer age) {
            this.age = age;
      }
}
```

The validation.xml

The ValidatorActionForm class *ExampleForm* provides a validation based on an XML file.

Create a new XML fle *validation.xml* in the folder *WEB-INF* of your project.

The validations are associated with the called action mapping, for example */step1* or */step2*.

The following source code shows the content of the *validation.xml*:

```
<form-validation>

    <formset>
        <!-- validation mapping for action /step1 -->
        <form name="/step1">
            <field
                  property="name"
                depends="required, minlength">
                  <arg key="exampleForm.name" />
                  <arg key="${var:minlength}" resource="false" />
                          <var>
                            <var-name>minlength</var-name>
                            <var-value>3</var-value>
                          </var>
            </field>
            <field
                property="age"
                depends="required, intRange, integer">
                    <arg key="exampleForm.age"/>
                    <arg name="intRange" key="${var:min}" resource="false" />
                        <arg name="intRange" key="${var:max}"
resource="false" />
                    <var>
                            <var-name>min</var-name>
                            <var-value>18</var-value>
                          </var>
                           <var>
                            <var-name>max</var-name>
                            <var-value>90</var-value>
                          </var>
            </field>
        </form>

        <!-- validation mapping for action /step2 -->
        <form name="/step2">
            <field
                  property="city"
                depends="required, minlength">
                  <arg key="exampleForm.city" />
                  <arg key="${var:minlength}" resource="false" />
                          <var>
                            <var-name>minlength</var-name>
                            <var-value>2</var-value>
                          </var>
            </field>
            <field
                  property="phone"
                depends="required, minlength">
                  <arg key="exampleForm.phone" />
                  <arg key="${var:minlength}" resource="false" />
                          <var>
                            <var-name>minlength</var-name>
                            <var-value>5</var-value>
                          </var>
            </field>
        </form>
    </formset>
</form-validation>
```

Message Resource Bunlde

Create a new text file *ApplicationResources.properties* in the package *de.laliluna.tutorial.multipageform*, which contains the error message keys.

Add the default error keys of the struts validator and four message keys which hold the labels for the properties, to display them in the error messages.

The following text shows the content of the file:

```
# default error messages for struts validator
errors.required='{0}' is required.
errors.minlength='{0}' can not be less than {1} characters.
errors.range='{0}' is not in the range {1} through {2}.
errors.integer={0} must be an integer.

# form labels
exampleForm.name=Name
exampleForm.age=Age
exampleForm.city=City
exampleForm.phone=Phone
```

Action classes

Create three classes, *Step1*, *Step2* and *Finish* in the package *de.laliluna.tutorial.multipageform.action* which extends the class Action.

The classes are the same as in the first example. You can copy them or copy and paste the content of the classes.

The following source code shows the class Step1:

```
public class Step1 extends Action {

      public ActionForward execute(ActionMapping mapping,
                                                   ActionForm form,
                                                   HttpServletRequest request,
                                                   HttpServletResponse response)
                                                  throws Exception {

            String forward = "step1";
            if(request.getParameter("btnStep1") != null){
                  forward = "step2";
            }

            return mapping.findForward(forward);
      }
}
```

The following source code shows the class Step2:

```
public class Step2 extends Action {

      public ActionForward execute(ActionMapping mapping,
                                                   ActionForm form,
                                                   HttpServletRequest request,
                                                   HttpServletResponse response)
                                                  throws Exception {

            String forward = "step2";
            if(request.getParameter("btnStep2") != null){
                  forward = "finish";
            }

            return mapping.findForward(forward);
      }
}
```

The following source code shows the class Finish:

```
public class Finish extends Action {

      public ActionForward execute(ActionMapping mapping,
                                                   ActionForm form,
                                                   HttpServletRequest request,
                                                   HttpServletResponse response)
                                                  throws Exception {

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

The JSP files

The JSP files are the same like the JSP files in the first example.

Copy them to the folder */WebRoot/form* or copy and paste the source code.

The following source code shows the *step1.jsp*:

```
<%@ page language="java" pageEncoding="UTF-8"%>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
```

```
<html>
      <head>
            <title>Step 1</title>
      </head>
      <body>
            <html:form action="/step1">

                  <%-- ouput errors --%>
                  <html:messages id="error" message="false">
                        <bean:write name="error" /> <br />
                  </html:messages>

                  <%-- input field for properties --%>
                  Name: <html:text property="name" /> <br />
                  Age: <html:text property="age" /> <br />

                  <%-- hidden field which specify the page --%>
                  <html:hidden property="step" value="1" />

                  <html:submit property="btnStep1"/>
            </html:form>
      </body>
</html>
```

The following source code shows the *step2.jsp*:

```
<%@ page language="java" pageEncoding="UTF-8"%>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<html>
      <head>
            <title>Step 2</title>
      </head>
      <body>
            <html:form action="/step2">

                  <%-- ouput errors --%>
                  <html:messages id="error" message="false">
                        <bean:write name="error" /> <br />
                  </html:messages>

                  <%-- input field for properties --%>
                  City: <html:text property="city" /> <br />
                  Phone: <html:text property="phone" /> <br />

                  <%-- hidden field which specify the page --%>
                  <html:hidden property="step" value="2" />

                  <%-- hidden fields for properties of step1 --%>
                  <html:hidden property="name" />
                  <html:hidden property="age" />

                  <html:submit property="btnStep2"/>
            </html:form>
      </body>
</html>
```

The following source code shows the *finish.jsp*:

```
<%@ page language="java" pageEncoding="UTF-8"%>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<html>
      <head>
```

```
            <title>Finish</title>
      </head>
      <body>
            <b>The values are:</b> <br /><br />
            Name = <bean:write name="exampleForm" property="name" /><br />
            Age = <bean:write name="exampleForm" property="age" /><br />
            City = <bean:write name="exampleForm" property="city" /><br />
            Phone = <bean:write name="exampleForm" property="phone" /><br />
      </body>
</html>
```

Configure the Struts Config

The struts-config requires the configuration of the ValidatorPlugin.

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

```
<struts-config>
   <form-beans>
      <form-bean name="exampleForm"
type="de.laliluna.tutorial.multipageform.form.ExampleForm" />
   </form-beans>
   <action-mappings>
        <action
         attribute="exampleForm"
         input="/form/step1.jsp"
         name="exampleForm"
         path="/step1"
         scope="request"
         type="de.laliluna.tutorial.multipageform.action.Step1" >
         <forward name="step1" path="/form/step1.jsp" />
         <forward name="step2" path="/step2.do" />
      </action>

      <action
         attribute="exampleForm"
         input="/form/step2.jsp"
         name="exampleForm"
         path="/step2"
         scope="request"
         type="de.laliluna.tutorial.multipageform.action.Step2" >
         <forward name="step2" path="/form/step2.jsp" />
         <forward name="finish" path="/finish.do" />
      </action>

      <action
         attribute="exampleForm"
         input="/form/finish.jsp"
         name="exampleForm"
         path="/finish"
         scope="request"
         type="de.laliluna.tutorial.multipageform.action.Finish" >
         <forward name="finish" path="/form/finish.jsp" />
      </action>
   </action-mappings>

   <message-resources
parameter="de.laliluna.tutorial.multipageform.ApplicationResources" />

   <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
    <set-property property="pathnames"
       value="/org/apache/struts/validator/validator-rules.xml,/WEB-
INF/validation.xml"/>
   </plug-in>
```

```
</struts-config>
```

Now the second example application has finished and can test now. Call the project with the link below:

http://localhost:8080/MultipageFormValidatorExample/