Struts Code Pieces – ValidatorActionForm

This tutorial explains the usage of the Struts form bean ValidatorActionForm using a working example.

Generals

Autor:
Sascha Wolski
Sebastian Hennebrüder
http://www.laliluna.de/tutorials.html – Tutorials for Struts, EJB, xdoclet und eclipse.

Datum:
February, 16th 2005

Dependencies
Struts 1.1
Jboss, Tomcat, Jetty etc
PDF download: http://www.laliluna.de/download/struts-validatoraction-form-en.pdf
Source download: http://www.laliluna.de/download/struts-validatoraction-form-source.zip

ValidatorActionForm class

The form bean class ValidatorActionForm extends the ValidatorForm class.

It has the capacity to validate fields of a form using validation rules which are defined in an XML file.

There is one difference between these classes. The validation rules of a ValidatorActionForm are not assigned to the action form but to the action. (*/path-to-action/action.do*).

So it is possible to reuse a bean in multiple actions but to define different validations for each action.

For the FormBean class a name is defined in the Struts Config.

Example:

```
<form-beans >
      <form-bean name="exampleForm" type="my.package.ExampleForm" />
</form-beans>
```

The form bean can than be reused in an action definition. Below you see an example for an ActionMapping in the Struts Config.

Example:

```
<action attribute="exampleForm"
        input="/form/example.jsp"
        name="exampleForm"
        path="/example"
        scope="request"
        type="my.package.ExampleAction" />
```

Validation of properties

The form bean ValidatorActionForm uses the Struts validation capabilities using validation rules defined in XML files. Struts offers a wide choice of rules, you can all find in the file validator-rules.xml.

You configure the rules for each property of a FormBean. These validations have to be written in

the XML file (validation.xml)

Example validation file validation.xml:

```
<form-validation>
 <formset>
        <!-- validation mapping für example action -->
        <form name="/example">
            <field
                  property="name"
                depends="required, minlength">
                  <arg0 key="exampleForm.name" />
                  <arg1 key="${var:minlength}" resource="false" />
                          <var>
                            <var-name>minlength</var-name>
                            <var-value>3</var-value>
                          </var>
            </field>
       </form>
    </formset>
</form-validation>
```

Initializing the properties

The ValidatorActionForm class offers a reset method which is called automatically by Struts. In this method you can initialize properties.

Example:

```
public void reset(ActionMapping mapping,
                  HttpServletRequest request) {

       //Initialisieren der Eigenschaft text
       text = "Hello World";
}
```

Working example of a ValidatorActionForm Beans

We will show you the usage of an ValidatorActionForm Bean using a simple working example.

Creating a ValidatorActionForm class

Create a bean form class *ExampleForm* in a package.
(Example*: de.laliluna.tutorials.validatoractionform.form*).

The class extends the *ValidatorActionForm*.

Create two properties name of type String and age of type int.

Create getters and setters for your properties.

The class looks like the following.

```
public class ExampleForm extends ValidatorActionForm {

      //Eigenschaften der Klasse
      private String name;
      private int age;

      //Getter und Setter Methoden
      public int getAge() {
            return age;
      }
      public void setAge(int age) {
            this.age = age;
      }
```

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

Creating an Action class

Create a new class of type ExampleAction in the package *de.laliluna.tutorial.validatoractionform.action.*

The class extends the Action class of Struts.

Implement the *execute(..)* method and output the variables.

The following is the source code of the class *ExampleAction*

```
public class ExampleAction extends Action {

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

		//ValidatorActionForm zuweisen
		ExampleForm exampleForm = (ExampleForm) form;

		//Zugriff auf Eigenschaften der ValidatorActionForm
		//Klasse innerhalb der Action Klasse
		System.out.println(exampleForm.getName());
		System.out.println(exampleForm.getAge());

		return mapping.findForward("success");
	}

}
```

Create the JSP files

Create two new JSP files named *example1.jsp* and *example2.jsp* in the directory *../WebRoot/form/* .

Below you can see the source code. It is apart from a simple difference the same.

Change the value of *<html:form action="..">* in the first JSP to */example1* and in the second JSP to */example2*.

```
<%@ 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"%>

<html>
	<head>
		<title>JSP for exampleForm</title>
	</head>
	<body>
		<html:form action="/example1">
			<html:errors />
			Name: <html:text property="name" /> <br>
			Age: <html:text property="age" /> <br>
			<html:submit value="Send"/>
		</html:form>
	</body>
```

```
</html>
```

Configure the form bean (struts-config.xml)

Open the *struts-config.xml* and add your form bean definition between the *<form-bean>,*

The attribute *name* defines the name of the form bean. Other action can refer to the form bean using this name.

type defines the form bean class. In our case it is ValidatorActionForm.

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

Create two action mappings (struts-config.xml)

We will create two action mappings to show the usage of the ValidatorActionForm class. Both actions will use the same form bean.

As we are lazy we will also use the same action class *ExampleAction*.

Create the action mappings in between the <*action-mappings*> tags. Use the form bean *exampleForm* and forward to the JSP files.

The first action mapping has the path */example1,* the second one /*example2*.

name is the name of the form bean

input defines the JSP which is shown when an error occured during the validation. In our case this is the same JSP we used to show the form.

type defines the action class which is called for the action

<forward ...> defines the forwards of the action. Here the JSP file are *example1.jsp* and *example2.jsp*.

```
<action-mappings >
      <action
         attribute="exampleForm"
         input="/form/example1.jsp"
         name="exampleForm"
         path="/example1"
         scope="request"
         type="de.laliluna.tutorial.validatoractionform.action.ExampleAction">
         <forward name="showExample" path="/form/example1.jsp" />
      </action>

       <action
         attribute="exampleForm"
         input="/form/example2.jsp"
         name="exampleForm"
         path="/example2"
         scope="request"
         type="de.laliluna.tutorial.validatoractionform.action.ExampleAction">
         <forward name="showExample" path="/form/example2.jsp" />
      </action>
   </action-mappings>
```

Initializing the ValidatorActionForm class

Add a reset method in the ValidatorActionForm class *ExampleForm* . Initialize the form properties in this method.

```
public void reset(ActionMapping mapping,
      HttpServletRequest request) {

      //Initialisieren der Eigenschaften
      name = "Adam Weisshaupt";
      age = 23;
}
```

Validating properties with XML validation rules

To validate the user input, if a name's length is greater than 3 character or the age is between 0 and 150, you have to configure this validations in an XML file.

Create the XML file *validation.xml* in the directory */WebRoot/WEB-INF/*.

<form name=".."> defines the Form Bean to which the validations are applied.

<field property=".."> defines a property of a form bean. The attribute *depends* configures the used rule from the Struts rule set. (All rules are defined in the validator-rules.xml).

<arg0 key=".."> defines a parameter which is passed to the error message. In the error message for intRange, there is one parameter expected. (more informations at MessageResouce).

<var-name> sets the name of the variable used in the validation rule and *<var-value>* the value of the variable.

We will create validation rules for each mapping.

```
<form-validation>
    <formset>
        <!-- validation mapping for action /example1 -->
        <form name="/example1">
            <field
                  property="name"
                depends="required, minlength">
                  <arg0 key="exampleForm.name" />
                  <arg1 key="${var:minlength}" resource="false" />
                          <var>
                            <var-name>minlength</var-name>
                            <var-value>3</var-value>
                          </var>
            </field>
            <field
                property="age"
                depends="required, intRange, integer">
                    <arg0 key="exampleForm.age"/>
                    <arg1 name="intRange" key="${var:min}" resource="false" />
                    <arg2 name="intRange" key="${var:max}" resource="false" />
                    <var>
                        <var-name>min</var-name>
                        <var-value>1</var-value>
                    </var>
                    <var>
                        <var-name>max</var-name>
                        <var-value>150</var-value>
                    </var>
            </field>
        </form>

        <!-- validation mapping for action /example2 -->
        <form name="/example2">
            <field
                property="name"
                depends="required, minlength">
```

```
                    <arg0 key="exampleForm.name" />
                    <arg1 key="${var:minlength}" resource="false" />
                    <var>
                          <var-name>minlength</var-name>
                          <var-value>6</var-value>
                    </var>
               </field>
          </form>
     </formset>
</form-validation>
```

Configure the ValidatorPlugins in the Struts Config file

In order to use the Struts-Validator you must add the ValidatorPlugin in the Struts Config. Otherwise Struts does not know your validation files and will not use them.

Open the struts-config.xml and add the following properties to the end of the struts config file.into the tag *<struts-config>* .

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

Create a Message Resource file

The Message Resource file is needed for the output of the error messages, we used in the execute method.

Create a new file named *ApplicationResources.properties* in the package *de.laliluna.tutorial.validatoractionform.*

You can find more information about message resource files in our Message Resource tutorial. http://www.laliluna.de/struts-message-resources-tutorial.html

Add the following to the file:

```
errors.suffix=<br>
# -- 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}.
# -- field names
exampleForm.name=Name
exampleForm.age=Age
```

Open the *struts-config.xml* and add the following lines to configure your resource file.

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

Test your example

We have finished our example application. Test the example by calling

http://localhost:8080/ValidatorActionForm/example1.do

(Validation of name and age)

http://localhost:8080/ValidatorActionForm/example2.do

(validation of name)