Struts Code Pieces – BeanValidatorForm

This tutorial explains the usage of the BeanValidatorForm using a small working example.

Generals

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

Date:
February, 8th 2005

Development Tools
Eclipse 3.x

Dependencies
Struts 1.1
Jboss oder Tomcat
PDF download: http://www.laliluna.de/download/struts-beanvalidator-form-tutorial-en.pdf
Source download: http://www.laliluna.de/download/struts-beanvalidator-form-source.zip

BeanValidatorForm class

The BeanValidatorForm is based on a simple Java object (POJO). You are not extending any classes. The BeanValidatorForm requires the Struts version 1.2.4.

Example of a simple java class.

```
public class ExamplePOJO {}
```

Define a name for your POJO class in the Struts Config.

Example:

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

The Form Bean can be used in an Action. Below you can see an example ActionMapping.

Example:

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

Validating properties

The class BeanValidatorForm 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 form -->
        <form name="exampleForm">
            <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 class used as form bean is a simple Java class. So you can use the constructor to initialize the properties.

Example:

```
public class ExamplePOJO {

       //Konstruktor zum Initialisieren
       //von Eigenschaften
       public ExamplePOJO(){
             name = "Adam Weisshaupt";
             age = 23;
       }
}
```

Working example of a BeanValidatorForm Bean

Using a small working example we will show you the use of the BeanValidatorForm.

Create a POJO class

Create a new class *ExamplePOJO* in the package *de.laliluna.tutorials.validatorform.pojo* or where ever you like.

Add two properties name and age.
Create getter and setter methods for your properties.

The class looks like the following:

```
public class ExamplePOJO {

      //Properties of the class
      private String name;
      private int age;

      //Getter and Setter Methods
      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;
      }
}
```

Create an action class

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

The class extends the *Action class*.

Implement the method *execute(..)*.

Output the name and the age to the log.

The complete source code is shown below.

```
public class ExampleAction extends Action {

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

		//BeanValidatorForm zuweisen
		BeanValidatorForm exampleForm = (BeanValidatorForm) form;

		//Zugriff auf Eigenschaften der BeanValidatorForm
		//Klasse innerhalb der Action Klasse
		System.out.println(exampleForm.get("name"));
		System.out.println(exampleForm.get("age"));

		return mapping.findForward("showExample");
	}

}
```

Create a JSP file

Create a JSP example.jsp in the directory *../WebRoot/form/* .

Below you can see the source code of the JSP file.

```
<%@ 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="/example">
			<html:errors />
			Name: <html:text property="name" /> <br>
			Age: <html:text property="age" /> <br>
			<html:submit value="Send"/>
		</html:form>
	</body>
</html>
```

Configure a FormBean (struts-config.xml)

Open the struts-config.xml and add a form bean declaration, between the <form-beans> tags, The attribute name defines the name of the FormBean, by which it can be used in action mappings. The type defines the name and the complete package of the form bean class.

```
<form-beans >
```

```
      <form-bean name="exampleForm"
type="de.laliluna.tutorial.beanvalidatorform.pojo.ExamplePOJO" />
</form-beans>
```

Configure the Action (struts-config.xml)

Add a action mapping in the struts-config.xml. Add the form bean *exampleForm* to the action and create a forward to the *example.jsp*.
name specifys the action of the form bean.
Type is the path to our action class, *ExampleAction*.
<forward ...> is the forward to our *example.jsp*.

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

            <forward name="showExample" path="/form/example.jsp" />

      </action>
</action-mappings>
```

Initializing the properties of the POJO class

You can initialize form properties in the default contructor.

The example source code for *ExamplePOJO*.

```
public ExamplePOJO(){
      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.

Create the following validations for the form bean property:

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

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.beanvalidatorform.ApplicationResources" />
```

Test your example

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

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

(We expect a standard installation of JBOSS or Tomcat)