Struts – Validation and error handling

Struts provides capabilities to validate form properties in two major modes, one is Java and the other is XML related. Another feature of Struts is error handling. In this tutorial we want to explain both of them and show you a small example application using these features.

General

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

Date
Updated: December 2007
First edition: February 2005

Software:
Struts Framework 1.3

PDF download: http://www.laliluna.de/download/struts-validation-error-handling-en.pdf
Source download: http://www.laliluna.de/download/struts-validation-error-handling.zip

Requirements

We require the basics of Struts to understand this tutorial. If you are new to Struts, read the 'first steps in struts' tutorial, you can find at http://www.laliluna.de/first-steps-using-struts-tutorial.html.

Create a struts project

Let's start, create a new struts project and a package named *de.laliluna.tutorial.validation*.

Action form class (java related validation)

Create a new class *ExampleForm* in the package *de.laliluna.tutorial.validation.form*, which extends the class *ActionForm*.

Add two properties, *name* of type String and *age* of type String.

Add a getter and setter method for each property.

Initialize the properties in the *reset()* method.

The following source code shows the content of the class ExampleForm.

```
public class ExampleForm extends ActionForm {

      private String name;
      private Integer age;

      public String getName() {
            return name;
      }

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

      public Integer getAge() {
            return age;
      }

      public void setAge(Integer age) {
```

```
            this.age = age;
      }
}
```

The Validate Method

The action form class provides a method *validate()* which is called before the action class is executed. So you can validate your properties within this method. The return-value of the method is a collection (ActionErrors) of error messages (ActionMessage).

You can validate your properties and add error messages to this collection for each wrong validation. In the JSP file you can display these messages to the user. The error messages are message keys of the message resource bundle. (You will find more infos about message resource bundle at http://www.laliluna.de/struts-message-resources-tutorial.html.)

Implement the *validate()* method of the action form class.

Validate each property and add an error message for each wrong validation.

These are the criteria for the validation:

- *name* must have more than three characters
- *age* must be not null and greater than 18

The following source code shows the validate() method.

```
public ActionErrors validate(ActionMapping mapping,
                  HttpServletRequest request) {

            // create a new instance of actionerrors
            ActionErrors actionErrors = new ActionErrors();

            // valdiate name
            if (name.length() < 3) {
                  actionErrors.add("name", new ActionMessage("error.name"));
            }

            // validate age
            if (age == null || age < 18) {
                  actionErrors.add("age", new ActionMessage("error.age"));
            }

            // return collection of action messages
            return actionErrors;
      }

      public void reset(ActionMapping mapping, HttpServletRequest request) {

            // reset properties
            name = "";
            age = 0;

      }
```

Validator form class (XML related validation)

Create a new class *ExampleXMLForm* in the package *de.laliluna.tutorial.validation.form*, which extends the class ValidatorForm.

Add the same properties in this class, *name* of type String and *age* of type String.

Add a getter and setter method for each property.

Initial the properties in the *reset()* method.

The following source code shows the content of the class ExampleXMLForm.

```
public class ExampleXMLForm extends ValidatorForm {

	private String name;
	private Integer age;

	public String getName() {
		return name;
	}

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

	public Integer getAge() {
		return age;
	}

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

	public void reset(ActionMapping mapping, HttpServletRequest request) {
		// reset properties
		name = "";
		age = 0;
	}
}
```

The XML related validation (validation.xml)

The ValidatorForm class provides validation based on an XML file. Within this file you define your rules and error message keys. To use this feature you have to configure the validator plugin in your *struts-config.xml*.

Open your *struts-conig.xml* and add the following lines:

```
	<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 provides a XML file *validator-rules.xml* which contains standard validation methods. These validation methods use standard message keys to display error messages to the user.

The following default error message keys exists for the struts validator:

```
# Struts Validator Error Messages
errors.required={0} is required.
errors.minlength={0} can not be less than {1} characters.
errors.maxlength={0} can not be greater than {1} characters.
errors.invalid={0} is invalid.

errors.byte={0} must be a byte.
errors.short={0} must be a short.
errors.integer={0} must be an integer.
errors.long={0} must be a long.
errors.float={0} must be a float.
errors.double={0} must be a double.

errors.date={0} is not a date.
errors.range={0} is not in the range {1} through {2}.
errors.creditcard={0} is an invalid credit card number.
errors.email={0} is an invalid e-mail address.
```

Create a new XML file *validation.xml* in the folder */WebRoot/WEB-INF/*. This file contains the validation mapping for the form properties.

These are the criteria for the validation:

- *name* must have more than three characters
- *age* must be a number and greater than 18

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

```
<form-validation>
 <formset>
        <!-- validation mappings -->
        <form name="exampleXMLForm">
            <field
                  property="name"
                depends="required, minlength">
                  <arg key="exampleXMLForm.name" />
                  <arg key="${var:minlength}" resource="false" />
                          <var>
                            <var-name>minlength</var-name>
                            <var-value>3</var-value>
                          </var>
            </field>
            <field
                property="age"
                depends="required, integer, validwhen">
                    <arg key="exampleXMLForm.age"/>
                    <arg name="validwhen" key="${var:min}" resource="false" />
                    <var>
                            <var-name>test</var-name>
                            <var-value>(*this* > 18)</var-value>
                          </var>
            </field>
        </form>
    </formset>
</form-validation>
```

The action classes

We need two action classes for each validation option (java related, XML related).

Create two classes, *ExampleAction* and *ExampleXMLAction* in the package *de.laliluna.tutorials.validation.action* which extends the class Action.

The following source codes shows the classes:

Action class *ExampleAction*:

```
public class ExampleAction extends Action {

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

            return mapping.findForward("example");
      }

}
```

Action class *ExampleXMLAction*:

```
public class ExampleXMLAction extends Action {

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

            return mapping.findForward("example");
      }

}
```

Message resource bundle

To display errors we need message keys which contains the error messages. Create a new text file named *ApplicationResources.properties* in the package *de.laliluna.tutorials.validation*.

Open the file and add the struts validator error message keys first:

```
# Struts Validator Error Messages
errors.required={0} is required.
errors.minlength={0} can not be less than {1} characters.
errors.maxlength={0} can not be greater than {1} characters.
errors.invalid={0} is invalid.

errors.byte={0} must be a byte.
errors.short={0} must be a short.
errors.integer={0} must be an integer.
errors.long={0} must be a long.
errors.float={0} must be a float.
errors.double={0} must be a double.

errors.date={0} is not a date.
errors.range={0} is not in the range {1} through {2}.
errors.creditcard={0} is an invalid credit card number.
errors.email={0} is an invalid e-mail address.
```

For the error messages of the XML related validation, we need two message keys which hold the labels for the form propeties, because the {0} in the struts validator error messsages will be

replaced with them. You will find these keys in the *validation.xml*.

```
# property labels
exampleXMLForm.name=Name
exampleXMLForm.age=Age
```

Now add the message keys we used in the *validate()* method of the action form class.

```
# Custom Error Messages
error.name=Name must have 3 Characters
error.age=Age is not over 18
error.number=Age is not a number
```

The JSP files

We need two JSP files for each action class we created before. The difference between these two files is the action in the form tag.

Create a JSP file named *example.jsp* and one named *exampleXML.jsp* in the folder */WebRoot/form*

Add a *html:form* and two *html:text* elements for each property.

Use *html:messsages* tags to display errors instead of html:errors. If you want to display an error associated with a form property use the attribute *property* of the html:messages tag. The value of the attribute specifies the property which error message should displayed.

The following source code shows the first jsp file *example.jsp*:

```
<%@ page language="java"%>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html>
<head>
      <html:base />
      <title>example.jsp</title>
</head>

<body>
      <html:form action="/example">
            <p>
                  <i><b>Display all error messages:</b>
                  </i>
                  <br />
                  <html:messages id="errors">
                        <bean:write name="errors" />
                        <br />
                  </html:messages>
            </p>
            <p>
                  <%-- input field for property name --%>
                  Name:
                  <html:text property="name" />
                  <br />

                  <i><b>Only error messages for property name:</b>
                  </i>
                  <br />
                  <html:messages id="err_name" property="name">
                        <bean:write name="err_name" />
                        <br />
                        <br />
                  </html:messages>
            </p>
```

```
            <p>
                  <%-- input field for property age --%>
                  Age:
                  <html:text property="age" />
                  <br />

                  <i><b>Only error messages for property age:</b>
                  </i>
                  <br />
                  <html:messages id="err_age" property="age">
                        <bean:write name="err_age" />
                        <br />
                  </html:messages>
            </p>
            <html:submit />
      </html:form>
</body>
</html:html>
```

The following source code shows the second jsp file *exampleXML.jsp*.

```
<%@ page language="java"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html>
<head>
      <html:base />
      <title>exampleXML.jsp</title>
</head>

<body>
      <html:form action="/exampleXML">
            <p>
                  <i><b>Display all error messages:</b>
                  </i>
                  <br />
                  <html:messages id="errors">
                        <bean:write name="errors" />
                        <br />
                  </html:messages>
            </p>
            <p>
                  <%-- input field for property name --%>
                  Name:
                  <html:text property="name" />
                  <br />

                  <i><b>Only error messages for property name:</b>
                  </i>
                  <br />
                  <html:messages id="err_name" property="name">
                        <bean:write name="err_name" />
                        <br />
                        <br />
                  </html:messages>
            </p>
            <p>
                  <%-- input field for property age --%>
                  Age:
                  <html:text property="age" />
                  <br />
```

```
                  <i><b>Only error messages for property age:</b>
                  </i>
                  <br />
                  <html:messages id="err_age" property="age">
                        <bean:write name="err_age" />
                        <br />
                  </html:messages>
            </p>
            <html:submit />
      </html:form>
</body>
</html:html>
```

Configure the struts-config.xml

Now open your *struts-config.xml* and add the form bean, the action and message resource mappings. And don't forget to configure the validator plugin ;-)

With the attribute *input* of the *<action>* tag you can specify a jsp file, Struts automatically forwards to it, if an error occurs. In our example we do not use a different jsp file to display error messages.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts
Configuration 1.3//EN" "http://struts.apache.org/dtds/struts-config_1_3.dtd">

<struts-config>
   <form-beans>
            <form-bean name="exampleForm"
                  type="de.laliluna.tutorial.validation.form.ExampleForm" />
            <form-bean name="exampleXMLForm"
                  type="de.laliluna.tutorial.validation.form.ExampleXMLForm" />
      </form-beans>

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

            <action attribute="exampleXMLForm" input="/form/exampleXML.jsp"
                  name="exampleXMLForm" path="/exampleXML" scope="request"

type="de.laliluna.tutorial.validation.action.ExampleXMLAction">
                  <forward name="example" path="/form/exampleXML.jsp" />
            </action>
      </action-mappings>

      <message-resources
            parameter="de.laliluna.tutorial.validation.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>
```

Run the example application

Now you can test the example. We recommend an installation of Jboss, Jetty or Tomcat to run this example. Call the example by using the following links:

http://localhost:8080/Validation/