

Exadel Studio

Struts Validation Examples

Validation of input is an important part of any Web application. All Apache Jakarta frameworks, including Struts, can use a common Jakarta Validation Framework for streamlining this aspect of Web application development. The Validation Framework allows the developer to define validation rules and then apply these rules on the client-side or the server-side.

Exadel Studio makes using the Validation Framework in Struts even easier through a specialized editor for the XML files that control validation in a project. In this document, we'll show you how this all works by creating some simple client-side validation and server-side validation examples.

Starting Point

The example assumes that you have already created our sample **StrutsHello** application from the *Getting Started Guide for Creating a Struts Application*. You should have the Exadel Studio perspective open on this **StrutsHello** project.

Defining the Validation Rule

In these steps you will set up the validation that can be used for either client-side or server-side validation. You need to enable validation as part of the project, define an error message, and tie it into the appropriate part of the application.

1. Right-click the **plug-ins** node under the **StrutsHello/Configuration/default/struts-config.xml** node in the Web Projects view and select **Create Special Plug-in/Validators** from the context menu.
2. Further down in the Web Projects view, right-click on the **StrutsHello/Resource Bundles** node and select **New/Properties File...** from the context menu.
3. In the dialog box, click on the **Browse...** button next to the **Folder** field, expand the **JavaSource** folder in this next dialog box, select the **sample** subfolder, and click on the **OK** button.
4. Back in the first dialog box, type in `app1Resources` for the **Name** field and click on the **Finish** button.
5. Right-click the newly created file and select **Add/Default Error Messages** from the context menu.

Exadel Studio: Struts Validation Examples

6. Drag up the **sample.appResources** icon until you can drop it on the **resources** folder under **struts-config.xml**.
7. Select **File/Save All** from the menu bar.
8. Select **validation.xml** under the **StrutsHello/Validation** node and double-click it to open it with the Exadel Studio Validation Editor.
9. Expand the **form-beans** node under the **StrutsHello/Configuration/default/struts-config.xml** node. Then, drag the form bean **GetNameForm** and drop it onto **formset (default)** in the Validation Editor.
10. In the Validation Editor, expand the **formset** node, right-click **GetNameForm**, and select **Add Field...** from the context menu.
11. Enter name for **Property** in the dialog box.
12. In the properties for the **name** field to the right of the “tree” for the validation.xml file, click on the **Change...** button next to the **Depends** entry field.
13. In the displayed double list, select **required** from the left list and then click **Add->** .
14. Click **Ok**.
15. Right-click **name** and select **Add Arg...** from the context menu.
16. In the **Add Arg** dialog box, click on the **Change...** button next to the **Key** field.
17. In the **Key** dialog box that appears now, click on the **Add** button.
18. Enter name **.required** in the **Name** field, and enter **A person's name** in the **Value** field.
19. Click **Finish**, then **Ok**, and then **Ok** again.
20. Select **File/Save All** from the menu bar.

Client-Side Validation

Client-side validation uses a scripting language (like JavaScript) running in the client browser to actually do the validation. In a Struts application using the Validation Framework, however, you don't actually have to do any of the script coding. The Validation Framework handles this. To see how this works in our application, you'll just need to make a couple of modifications to one of the JSP files.

21. Double-click **inputname.jsp** under **StrutsHello/WEB-ROOT (WebContent)/pages** to open it for editing.
22. Find the **</title>** tag near the top and hit Return to make a new line under it.
23. In the Exadel Palette view to the right, open the **HTML** folder and click on the **javascript** tag.

Exadel Studio: Struts Validation Examples

24. Back in the editor, just in front of the closing slash for this inserted tag, hit Ctrl-space and select **formName** from the prompting menu.
25. Over in the Web Projects view, select **GetNameForm** under the **StrutsHello/Configuration/default/struts-config.xml/form-beans** node, drag it, and drop it between the quotes in the editor.
26. Modify the `<html:form>` tag by inserting this attribute:

```
onsubmit="return validateGetNameForm(this)"
```

The file should now look like this:

```
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<html:html>
<head>
    <title>Input name</title>
    <html:javascript formName="GetNameForm" />
</head>
<body>
    <html:form action="/greeting.do" onsubmit="return
validateGetNameForm(this)">
        <table border="0" cellspacing="0" cellpadding="0">
            <tr>
                <td><b>Input name:</b></td>
            </tr>
            <tr>
                <td>
                    <html:text property="name" />
                    <html:submit value="    Say Hello!    " />
                </td>
            </tr>
        </table>
    </html:form>
</body>
</html:html>
```

27. Select **File/Save** from the menu bar.
28. Start Tomcat by clicking on its icon (a right-pointing arrow) in the toolbar.
29. Click on the **Run** icon in the toolbar.



30. In the browser window, click on the **Say Hello!** button without having entered any name in the form.

A JavaScript error message should be displayed in an alert box.

Server-Side Validation

Server-side validation does the validation inside the application on the server. In a Struts application using the Validation Framework, you still don't have to do any of the actual vali-

Exadel Studio: Struts Validation Examples

ation coding. The Validation Framework handles this. You will, though, have to make a few changes to the JSP file you modified for client-side validation along with a change to an action and a few changes to the form bean class.

Editing the JSP File

31. Reopen **inputname.jsp** for editing.
32. Delete the `onsubmit` attribute in the `<html:form>` element that you put in for client-side validation.
33. Add an `<html:errors/>` tag after the `</html:form>` tag.

The JSP file should now look like this:

```
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<html:html>
<head>
<title>Input name</title>
<html:javascript formName="GetNameForm"/>
</head>
<body>
  <html:form action="/greeting.do">
    <table border="0" cellspacing="0" cellpadding="0">
      <tr>
        <td><b>Input name:</b></td>
      </tr>
      <tr>
        <td>
          <html:text property="name" />
          <html:submit value=" Say Hello! " />
        </td>
      </tr>
    </table>
  </html:form>
  <html:errors />
</body>
</html:html>
```

Editing the Action

34. In the Web Projects view, expand the node under the **StrutsHello/Configuration/default/struts-config.xml/action-mappings** node, right-click the **/greeting** action, and then select **Properties...** from the context menu.
35. In the Edit Properties window, insert the cursor into the **value** column for the **input** property and click on the ... button.
36. In the dialog box, make sure the **Pages** tab is selected, select **StrutsHello/WEB-ROOT(WebContent)/pages/ inputname.jsp**, click the **Ok** button, and then click on the **Close** button.

Editing the Form Bean

37. Right-click the **/greeting** action again and select **Open Form-bean Source** to open the **GetNameForm.java** file for editing.

38. Change the class that it extends to from:

```
org.apache.struts.action.ActionForm
```

to:

```
org.apache.struts.validator.ValidatorForm
```

39. Comment out out the validate method.

The file should now look like this:

```
package sample;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionMapping;

public class GetNameForm extends
org.apache.struts.validator.ValidatorForm {
    private String name = "";

    /**
     * @return Returns the name.
     */
    public String getName() {
        return name;
    }
    /**
     * @param name The name to set.
     */
    public void setName(String name) {
        this.name = name;
    }
    public GetNameForm () {
    }

    public void reset(ActionMapping actionMapping, HttpServletRequest
request) {
        this.name = "";
    }

    //    public ActionErrors validate(ActionMapping actionMapping,
HttpServletRequest request) {
    //        ActionErrors errors = new ActionErrors();
    //        return errors;
    //    }

}
```

Exadel Studio: Struts Validation Examples

Select **File/Save All** from the menu bar.

40. Reload the application into Tomcat by clicking on the Change Time Stamp icon (a finger pointing with a little star) in the toolbar.
41. Run the application.
42. In the browser window, click on the **Say Hello!** button without having entered any name in the form.

The error message should appear in a refreshed version of the form.