Data Validation

Table of contents

1 Introduction....2
2 Validation Annotations....2
  2.1 ...on the Form Bean Getter Methods....2
  2.2 ...on the Action Method....3
  2.3 ...on the Controller Class....3
3 Handling Validation Errors....4
  3.1 Default Error Messages....4
  3.2 Other Message Resources....5
  3.3 Internationalization of Validation Rules....6
  3.4 Required Fields....6
  3.5 Post-Error Navigation....7

1. Introduction

NetUI offers a declarative validation model. This means that validation tasks are configured using metadata annotations within Page Flow controller classes. Annotations for common validation tasks are already provided: e.g., @Jpf.ValidateCreditCard, @Jpf.ValidateDate, etc.

2. Validation Annotations

Validation annotations can appear in three different places:

- Form Bean Getter Methods
- Action Methods
- Controller Class

Note:

If you are using Struts 1.1 in your project, then you need to add an additional build step in order for these annotations to take effect. All generated Validator XML config files must be copied out of `/WEB-INF/classes/_pageflow` and into `/_pageflow`:

```
    <delete dir="${web.build.dir}/_pageflow"/>
    <mkdir dir="${web.build.dir}/_pageflow"/>
    <copy todir="${web.build.dir}/_pageflow">
        <fileset dir="${web.build.dir}/WEB-INF/classes/_pageflow"
includes="*-validation*.xml"/>
    </copy>
```

Additionally, it is a good idea to make these files inaccessible from a web browser by adding the following entries in web.xml:

```
    <filter>
        <filter-name>PageFlowForbiddenFilter</filter-name>
        <filter-class>org.apache.beehive.netui.pageflow.PageFlowForbiddenFilter</filter-class>
        <init-param>
            <param-name>response-code</param-name>
            <param-value>404</param-value>
        </init-param>
    </filter>
    ...
    <filter-mapping>
        <filter-name>PageFlowForbiddenFilter</filter-name>
        <url-pattern>/_pageflow/*</url-pattern>
        <dispatcher>REQUEST</dispatcher>
    </filter-mapping>
```

By default, NetUI projects are configured with Struts 1.2. This workaround is **only** necessary when using Struts 1.1.

2.1. ...on the Form Bean Getter Methods

Form bean getter methods are the simplest place to put validation annotations:

```
    public static class MyForm implements Serializable
    {
        private String _date;
        @Jpf.ValidatableProperty(
            displayName="This field",
            validateRequired=@Jpf.ValidateRequired(),
            validateDate=@Jpf.ValidateDate(pattern="M-d-y")
        )
        public String getDate()
        {
            return _date;
        }

        public void setDate(String str)
        {
            _date = str;
        }
    }
```

2.2. ...on the Action Method

When the validation annotations are applied to the Action Method, you must specify the Form Bean field to which validation will apply. (You don't need to specify the Form Bean class, because it is already specified by the Bean parameter of the Action Method.)

`@Jpf.ValidatableProperty(propertyName)` specifies the Form Bean field.

```
    @Jpf.Action(
        forwards={
            @Jpf.Forward(name="success", path="success.jsp")
        },
        validatableProperties={
            @Jpf.ValidatableProperty(
                propertyName="date",
                displayName="This field",
                validateRequired=@Jpf.ValidateRequired(),
                validateDate=@Jpf.ValidateDate(pattern="M-d-y")
            )
        },
        validationErrorForward=@Jpf.Forward(name="fail", path="input.jsp")
    )
    public Forward submitForm(MyForm form)
    {
        return new Forward("success");
    }
```

2.3. ...on the Controller Class

The following class-level annotation says that anytime that an instance of MyForm is submitted, then its `date` field will be validated (against the pattern M-d-y).

When the validation annotations decorate the Controller class, you must specify the Form

Bean class and the Form Bean field to which validation will apply.

`@Jpf.ValidateableBean(type)` specifies the Form Bean class.

`@Jpf.ValidatableProperty(propertyName)` specifies the Form Bean field.

Note that the `propertyName` must match the setter/getter field name in the Bean. In the example below, the setter/getter field name is **date** (because the setter/getter methods are setDate(...)/getDate()).

```
@Jpf.Controller(
    validatableBeans={
        @Jpf.ValidatableBean(
            type=Controller.MyForm.class,
            validatableProperties={
                @Jpf.ValidatableProperty(
                    propertyName="date",
                    displayName="This field",
                    validateDate=@Jpf.ValidateDate(pattern="M-d-y")
                )
            }
        )
    }
)
public class Controller extends PageFlowController
{
    ...

    public static class MyForm implements Serializable
    {
        private String _date;

        public String getDate()
        {
            return _date;
        }

        public void setDate(String str)
        {
            _date = str;
        }
    }
}
```

3. Handling Validation Errors

3.1. Default Error Messages

If validation fails, then an ActionMessage is written describing the error under the key `propertyNameValue`.

```
                @Jpf.ValidatableProperty(
                    propertyName="propertyNameValue",
                                          ...
                )
```

This error is retreivable by the <netui:error> tag, using the matching key value.

```
        <netui:form action="handleSubmit">
            Enter the date (MM-dd-YYYY):
            <netui:textBox dataSource="actionForm.date"/>
            <netui:error key="propertyNameValue"/>
            <br/><netui:button value="Submit"/>
        </netui:form>
```

The `propertyName` prepends a string to the error message produced.

```
                @Jpf.ValidatableProperty(
                    propertyName="propertyNameValue",
                    displayName="This field",
                )
```

For example, if validation fails for a date field, the error message produced would be "`This field`" + "`is not a date.`"

3.2. Other Message Resources

Other message resources can be specified through the `message/messageKey/messageArgs` attributes.

```
                @Jpf.ValidatableProperty(
                    propertyName = "item3",
                    validateMinLength =
                        @Jpf.ValidateMinLength(
                            chars = 6,
                            message = "{0} {1} for {2} is {3} characters.",
                            messageArgs = {
                                @Jpf.MessageArg(
                                    arg = "The minimum",
                                    position = 0
                                ),
                                @Jpf.MessageArg(
                                    arg = "length"
                                ),
                                @Jpf.MessageArg(
                                    arg = "this field"
                                ),
                                @Jpf.MessageArg(
                                    arg = "six"
                                )
                            }
                        )
```

If the `message/messageKey` attributes are not present, then the default message key will be used ("`errors.date`" for `@Jpf.ValidateRequired`, "`errors.minLength`" for `@Jpf.ValidateMinLength`, etc.)

If the `arg0/arg0Key, etc...` attributes are present, it is used as the first argument to the message; otherwise, the `displayName/displayNameKey` attribute on the `@Jpf.ValidatableProperty` is resolved for the first argument.

Message resources can be associated with a Form Bean class:

```
@Jpf.FormBean(defaultMessageBundle="myFormBean.errors")
public static class MyFormBean
```

The <netui:error> and <netui:errors> tags will resolve validation messages in the following order:

1. The default message bundle associated with the Controller class
   ```
   @Jpf.Controller(
       messageBundles = {
           @Jpf.MessageBundle(bundlePath = "validation.messages.messages")
       },
   ```
2. The message bundle associated with the current Form Bean
3. The fallback message bundle in beehive-netui-pageflow.jar. This contains values for default validation messages (e.g., "This field is not a date.")

3.3. Internationalization of Validation Rules

In some cases the parameters to rules (and even the rules themselves) may change depending on the locale. Rules internationalization is supported through the `language, country, and variant` attributes on `@Jpf.ValidationRules`:

```
    @Jpf.ValidatableProperty(
        localeRules={
            @Jpf.ValidationLocaleRules(
                language="fr",
                validateDate=@Jpf.ValidateDate(pattern="DD/MM/YYYY")
            ),
            @Jpf.ValidationLocaleRules(
                language="fr",
                country="CA",
                variant="Quebec",
                validateDate=@Jpf.ValidateDate(pattern="MM/DD/YYYY")
            )
        },
        validateRequired=@Jpf.ValidateRequired()  // this one applies to
all locales
    )
    public String getDate()
    {
        ...
```

3.4. Required Fields

Required fields can be marked using the `validateRequired` attribute.

```
                @Jpf.ValidatableProperty(
                    propertyName="propertyNameValue",
                    displayName="This field",
                    validateRequired=@Jpf.ValidateRequired(),
                                        ...
                )
```

3.5. Post-Error Navigation

The action method handling the submit, specifies the JSP to go to, if the validation fails.

Typically, the user is returned to the same submission page, where the error message can prompt the user to correct the invalid submission.

```
    @Jpf.Action(
        forwards={
            @Jpf.Forward(name="success", path="success.jsp")
        },
        validationErrorForward=@Jpf.Forward(name="fail", path="input.jsp")
    )
    public Forward handleSubmit(MyForm form)
```