Field Validators

Validating Field Content

A field validator is a Java class which prevents from changing to the next panel if one or more input field content is not valid during the installation. If the chosen validator reports some field content to be false, a messagebox is popped up with a defined (translated or untranslated content) and the user will be unable to continue to the next panel.

Validators are bound to particular fields, each field can have zero, one or more validators assigned in the user panel definition.

The validation itself is triggered when the user presses the Next button to change to the nex panel. This change is prevented in case of a negative validation, the user is forced to fix the bad input field values he previously provided.

There are built-in validators available, and own, user-defined validator implementations can be used.

Using Validators

Validators are bound to user input fields using the nested <validator> element;

Example

```
<field type="text" variable="ora.db">
  <spec txt="SID:" id="text.oraclesettings.sid" size="3" set=""/>
  <validator class="com.izforge.izpack.panels.userinput.validator.NotEmptyValidator"
txt="Invalid oracle SID!" id="text.oraclesettings.error.sid" />
</field>
```

Built-in Validators

HostAddressValidator

IsPortValidator

NotEmptyValidator

The `NotEmptyValidator` simply checks that the user entered a non-null value into each subfield, and returns false otherwise.

Usage example

```
<field type="rule" variable="test.notempty">
  <description align="left" txt="A description for a rule input field."
id="description.rule.1"/>
  <spec txt="Please enter your phone number:" layout="( N:3:3 ) N:3:3 - N:4:4 x N:5:5"
resultFormat="specialSeparator" separator="."/>
  <validator class="com.izforge.izpack.panels.userinput.validator.NotEmptyValidator"
txt="The phone number is mandatory!" />
</field>
```

PasswordEqualityValidator

This validator uses a password field specification to compare the values in each field for equality. Normally, this would be to ensure a password was typed correctly before any other validation takes place.

```
<field type="password" align="left" variable="the.password">
  <spec>
    <pwd txt="The Password:" size="25" set=""/>
    <pwd txt="Retype Password:" size="25" set=""/>
  </spec>
  <validator class="com.izforge.izpack.panels.userinput.validator.PasswordEqualityValidator"
        txt="Both passwords must match." id="lang pack key for the error text"/>
</field>
```

PasswordKeystoreValidator

This validator uses the password field and parameters you send in from previous user input or predefined properties to open a keystore and optionally try to get a specified key.

You must specify the parameter 'keystoreFile', and optionally 'keystoreType' (defaults to JKS), 'keystoreAlias' (to check for existence of a key), and 'aliasPassword' (for trying to retrieve the key).

An additional parameter 'skipValidation' can be set to 'true' in a checkbox and allow the validator framework to run, but not actually do the validation.

Optionally checking the key password of multiple keys within a keystore requires the keystore password (if different from the key password) be set in the 'keystorePassword' parameter.

```
<field type="password" align="left" variable="keystore.password">
  <spec>
    <pwd txt="Keystore Password:" size="25" set=""/>
    <pwd txt="Retype Password:" size="25" set=""/>
  </spec>
  <validator class="com.izforge.izpack.panels.userinput.validator.PasswordEqualityValidator"
               txt="Both keystore passwords must match." id="key for the error text"/>
  <validator class="com.izforge.izpack.panels.userinput.validator.PasswordKeystoreValidator"
               txt="Could not validate keystore with password and alias provided." id="key for the
error text">
    <param name="keystoreFile" value="${ssl.keystore}"/>
    <param name="keystoreType" value="${ssl.keystore.type}"/>
    <param name="keystoreAlias" value="${keystore.key.alias}"/>
    <param name="skipValidation" value="${skip.keystore.validation}"/>
  </validator>
</field>
```

PortValidator

RegularExpressionValidator

The RegularExpressionValidator checks that the user entered a value which matches a specified regular expression, as accepted by the Jakarta Regexp library (http://jakarta.apache.org/regexp/). The syntax of this implementation is described in the javadoc of the RE class (http://jakarta.apache.org/regexp/apidocs/org/apache/regexp/RE.html).

You can specify the regular expression to be tested by passing a parameter with a name of pattern to the validator (via the param element), with the regular expression as the value attribute. For example, the following would validate an e-mail address:

The example of using Regexp validator in rule input field:

```
<field type="rule" variable="EMAILaddress">
  <spec
      txt="Your Email address:" layout="O:12:U @ O:8:40 .
      A:4:4"
      set="0: 1:domain 2:com" resultFormat="displayFormat"
  />
  <validator
  class="com.izforge.izpack.util.RegularExpressionValidator"
      txt="Invalid email address!">
    <param
       name="pattern"
       value="[a-zA-Z0-9._-]{3,}@[a-zA-Z0-9._-]+([.][a-zA-Z0
       -9_-]+)*[.][a-zA-Z0-9._-]{2,4}"
    />
  </validator>
</field>
```

The example of using Regexp validator in text input field:

```
<field type="text" variable="EMAILaddress">
  <spec
      txt="Your Email address:" set="you@domain.com" size="20" id=""
  />
  <validator
  class="com.izforge.izpack.panels.userinput.validator.RegularExpressionValidator"
      txt="Invalid email address!">
    <param
        name="pattern"
        value="[a-zA-Z0-9._-]{3,}@[a-zA-Z0-9._-]+([.][a-zA-Z0
        -9_-]+)*[.][a-zA-Z0-9._-]{2,4}"
    />
  </validator>
</field>
```

An example of using regexp validator in a password field (attribute text wrapped for readability):

```
<field type="password" align="left" variable="db.password">
  <spec>
    <pwd txt="DB Password:" size="25" set=""/>
    <pwd txt="Retype Password:" size="25" set=""/>
  </spec>
  <validator class="com.izforge.izpack.panels.userinput.validator.PasswordEqualityValidator"
               txt="Both DB passwords must match." id="key for the error text"/>
  <validator class="com.izforge.izpack.panels.userinput.validator.RegularExpressionValidator"
               txt="Service password must begin with a character and be 8-20
                                mixed-case characters, numbers, and special characters [#@!$_]."
                                id="key for the error text">
    <param name="pattern" value="^(?=[a-zA-Z])(?=.*[0-9])(?=.*[#@!$_])
               (?=.*[A-Z])(?=.*[a-z])(?!.*[^a-zA-Z0-9#@!$_])(?!.*\s).{8,20}$"/>
  </validator>
</field>
```

You can test your own regular expressions using the handy applet at http://jakarta.apache.org/regexp/applet.html.

User-defined Validators

You can implement your own custom Validator implementation simply by creating a new class which implements the com.izforge.izpack.panels.userinput.validator.Validator interface. This interface specifies a single method: validate(ProcessingClient) , which returns a boolean value. You can retrieve the value entered by the user by casting the input ProcessingClient for example in the RuleInputField and call the RuleInputField.getText() method. You can also retrieve any parameters to your custom Validator by calling theRuleInputField.getValidatorParams() which returns a java.util.Map object containing parameter names mapped to parameter values. For an example, take a look at com.izforge.izpack.util.RegularExpressionValidator.