Cocoon Forms: Validation (2.1 legacy document)

Table of contents

Table of contents

Warning:
This document was copied as is from the Cocoon 2.1 documentation, but has not yet been fully reviewed or moved to its new home.

1. Concept

For each widget you can define a number of widget validators. A widget validator can perform some checks on the widget and if these fail, set a validation error on the widget.

Implementation-wise, a widget validator is an object implementing the WidgetValidator interface (javadoc).

WidgetValidators can either be defined as part of the form definition (inside the fd:validation element) or added to a widget instance at runtime. The former is useful for validators that only act on the data in the form. The latter is useful if the validation logic needs access to other objects you have available in your controller.

The validation logic runs over the widget tree. For each widget first the child widgets are validated and then the widget itself (recursively). The validators on a widget are executed in the order in which they were defined or added. First the ones defined in the form definition are executed, then the ones added on the widget instance. The validation of a widget stops at the first validator that fails (but continues to execute for the other widgets).

For widgets having a datatype and hence a convertor (field and multivaluefield), the convertor could be considered to be the first validator, i.e. it is executed before the other validators (because those operate on the converted value). If the conversion fails a validation error is set on the widget.

Validation errors can only be set on widgets implementing the interface ValidationErrorAware, which currently is not implemented by all widgets. For example, a repeater widget does not implement ValidationErrorAware. However, a validator attached to a repeater could perform inter-row checks on the fields in the different rows of the repeater, and set validation errors on these fields (instead of on the repeater itself).

CForms supplies a number of default widget validators, mostly for performing checks on the value of field widgets. Additionally you can write your own ones in Java or in Javascript.

2. Table of widgets supporting ValidationErrorAware

These are the widgets on which you can call setValidationError (and getValidationError). This is relevant if you are writing your own validation logic.

Widget	Supports ValidationErrorAware
field	yes
multivaluefield	yes
booleanfield	
repeater	
output	
submit	
action	

repeater-action	
row-action	
aggregatefield	yes
upload	yes
messages	

3. Reference

3.1. General remarks

For most widget validators, the failmessage (i.e. the message displayed to the user in case the validation failed) can be overridden by specifying a child **fd:failmessage** element inside the validator element. The failmessage can contain mixed content. Example:

```
<fd:field id="yourmail"> <fd:datatype base="string"/> <fd:validation> <fd:email>
<fd:failmessage>Not a valid email address!</fd:failmessage> </fd:email> </fd:validation>
</fd:field>
```

To provide locale-dependent messages, use i18n tags in combination with the I18nTransformer.

Often the values that validators will check are specified as expressions. CForms uses for this the xReporter expression interpreter.

Note that you cannot use each validator with each widget. Most validators only work with certain types of widgets, in case of field widgets often expecting a specific datatype. The below table shows the supported combinations for the default validators.

Validator	Allowed datatypes
fd:assert	all datatypes
fd:email	string
fd:length	string
fd:mod10	string
fd:range	integer, long, decimal
fd:regexp	string
fd:value-count	all array types (use this with multivaluefield)

3.2. fd:assert

Evaluates the expression specified in the "test" attribute. This expression should have a boolean result, it should evaluate to either true or false. Example: Suppose there are 2 fields widgets password and confirmPassword. We can use assert inside confirmPassword to check if is equals to password widget:

```
<fd:assert test="password = confirmPassword"> <fd:failmessage>The two passwords are not
equal.</fd:failmessage> </fd:assert>
```

3.3. fd:email

Checks that a value is a valid email address. Example:

<fd:email/>

Currently this checks the email does not contain any spaces, contains exactly one @ symbol with at least one character before it and at least one dot after it.

3.4. fd:length

Checks the length of strings. This validator can take 3 attributes: min, max and exact. You can use either of these three separately or min and max together. The values of these attributes are expressions. Example:

<fd:length min="2" max="4"/> Another example: <fd:length exact="2*2"> <fd:failmessage>Must be 4 characters long!</fd:failmessage> </fd:length>

3.5. fd:mod10

Uses the "mod10" algorithm used to check the validity of credit card numbers such as VISA. This validator does not require any additional attributes. Example:

<fd:mod10> <fd:failmessage>Invalid credit card number.</fd:failmessage> </fd:mod10>

3.6. fd:range

Checks the numeric range. This validator can take 3 attributes: min, max and exact. You can use either of these three separately or min and max together. The values of these attributes are expressions. Example:

<fd:range min="2" max="4"/> Another example: <fd:range exact="2*2"/>

3.7. fd:regexp

Checks that a string matches a regular expression. It requires a "pattern" attribute specifying the regexp. The regular expression library used is Jakarta ORO, see here for some information. Example:

<fd:regexp pattern="[a-z]{3,5}"> <fd:failmessage>Invalid code!</fd:failmessage> </fd:regexp>

3.8. fd:value-count

Checks the number of items selected in a multivaluefield. Again works with min, max and exact attributes. Example:

<fd:value-count min="2" max="4"/> Another example: <fd:value-count exact="2"/>

3.9. fd:javascript

Allows to write a validator using Javascript, embedded directly in the form definition. The widget in question is available in the Javascript snippet as a variable called widget.

Checkout the samples of Cocoon for an example.

1. Comments

add your comments