FormValidationUsingCocoon

Form validation in Cocoon can be achieved using two components:

- The Form Validator Action – carries out the validation
- The Form Validator Logicsheet – provides access to result of processing within an XSP page.

As the form validator is a generic component it must be provided with parameters that describe the fields, and their validation routines for each form. The Action and the Logicsheet both process this description. This allows the action to apply the constraints, and the logicsheet to provide useful feedback to the user, i.e. what those constraints actually are.

Describing the Form

The XML file that describes form fields contains the following information:

- Global field definitions – each field has a name, a type, and some constraints
- Constraint Sets – combinations of fields that define a set of constraints that will be applied to a single form.

XML file structure:

```
<root>
  <!-- field definitions -->
  <parameter name="" type="" nullable="" .../>
  <!-- constraint sets -->
  <constraint-set name="">
    <validate name=""/>
  </constraint-set>
</root>
```

Field Definitions

- Each field should have a unique name
- Each field should have a type. Legal types are string, long, double.
- Optional constraints

Constraints

The legal constraints are:

- Nullable indicator – `nullable="yes|no"`
- Default value – `default="..."`
- Min and maximum values – `min="..." max="..."`
- Min and maximum lengths – `min-len="..." max-len="..."`
- Regular expression matching – POSIX regular expression match, `matches-regex="..."`

The constraints defined for individual parameters can be overridden within a particular constraint set.

Examples

```
<parameter name="username" type="string" nullable="no" min-len="5" max-len="10"/>
<parameter name="email" type="string"
           max-len="50"
           matches-regex="^[\d\w][\d\w\-_\.]*@([\d\w\-_]+\.)+(\w){2,4}$"/>
```

Additional Attributes

Additional attributes can be added to parameter definitions beyond the constraints specified above.

These will be ignored by the Form Validator Action, but are accessible using the Form Validator Logicsheet tags. This means that additional form-related parameters can be included in the descriptor and referenced from XSP pages, e.g. length of input fields, or other HTML form parameters.

Constraint Sets

A constraint set defines a named list of parameters that should be checked in one validation attempt. This avoids the need to explicitly list all parameters in the sitemap.

```
<constraint-set name="user">
  <validate name="username"/>
  <validate name="email"/>
</constraint-set>
```

Parameters defined in a constraint set can override constraints set globally in the main parameter definition. However all parameters must be declared separately before being used in a set.

There are also two additional attributes that can be specified on a `validate` element in a `constraint-set`.

- `equals-to` – ensures parameter is equal to a fixed string
- `equals-to-param` – ensures one parameter is equal to another, e.g. for checking passwords

Carrying out Validation

To carry out validation the FormValidatorAction requires two parameters:

- The location of the form descriptor file
- EITHER a list of named parameters that should be validated
- OR the name of a constraint set to apply

```
<map:act type="form-validator">
  <parameter name="descriptor" value="context://descriptor.xml">
  <parameter name="validate" value="username,email">
</map:act>
```

```
<map:act type="form-validator">
  <parameter name="descriptor" value="context://descriptor.xml">
  <parameter name="validate-set" value="user">
</map:act>
```

This second form will test all parameters against the available descriptions. Request parameters not described in the constraint set will be ignored.

The action will return null if the validation fails, however an additional request parameter is added which describes the outcome of the validation. This parameter can be interpreted using the Form Validation Logicsheet.

Note: if a redirect is carried out at the end of the validation, then the results will be lost, although the original request parameters are retained.

Sitemap Example

```
<map:match pattern="target-for-form-POST">
  <map:act type="form-validator">
   <map:parameter name="descriptor"
                  value="context://descriptor.xml"/>
   <map:parameter name="validate-set"
                  value="constraint-set"/>
  <!-- do successful processing -->
  </map:act>
  <!-- do unsuccessful processing -->
</map:match>
```

Using the Logicsheet

As explained above the Form Validator Action adds a new request parameter that contains the results of the validation. The easiest way to interpret the results of processing is using the Form Validator Logicsheet from within an XSP page.

See XSPFormValidator Logicsheet for more information.