Cocoon Forms: XSLT (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. Intro

This page contains information on the default XSLT's provided by CForms, and the fi:styling directives they support. Be sure to read first about the template transformer.

There are basically 3 XSLTs:

- forms-samples-styling.xsl: this stylesheet includes two stylesheets: one for widget styling, one for page styling. You can choose between the basic types of it or advanced stylings. Usually, you will make a clone of this stylesheet for your own project, and use the other stylesheets as is.
- forms-field-styling.xsl: contains templates that style individual widgets, i.e. templates that translate fi:field, fi:booleanfield fi:action, etc. to HTML.
- forms-page-styling.xsl: contains templates for building high-level page layout effects, such as tabbed panes.

Additionally there are 2 XSLTs for advanced widget styling:

- forms-advanced-field-styling.xsl: contains templates that provide advanced styling of fields, e.g. the "double-listbox" for a multivaluefield. It's indeed an extension of the above basic forms-field-styling.xsl. Furthermore it includes the next stylesheet.
- forms-calendar-styling.xsl: contains the styling of a field with type "date" and provides a visual calendar for easy selection of date. So the calendar is an advanced styling too, but because it has much specific stuff we separated it out of forms-advanced-styling.xsl.

From the sitemap you only need to reference the first one, for example as follows:

```
<map:transform src="context://samples/forms/resources/forms-samples-styling.xsl"/>
```

2. fi:styling options

2.1. fi:field (without selection list)

By default all attributes on the fi:styling element will be copied onto the resulting HTML <input> element. Thus you can put e.g. type, class, style and size attributes on it. For example:

```
<ft:widget id="myfield"> <fi:styling size="50" class="kinky"/> </ft:widget>
```

Some values for the type attribute will cause special effects:

- **type="textarea"**: creates a <textarea> instead of an <input> element
- **type="htmlarea"**: creates a <textarea> instead of an <input> element, adding support for WYSIWYG editing of HTML using HTMLArea (works with IE 5.5+ under Windows and Mozilla 1.3+, any platform)
- **type="output"**: outputs the field's value as uneditable text

Adding an attribute **submit-on-change="true"** on fi:styling will cause an automatic form submit when the field's value changes.

2.2. fi:field (with selection list)

By default, a field with a selection list is rendered as a dropdown box.

Adding an attribute **list-type="radio"** on fi:styling produces a vertical list of radio buttons instead. Add list-orientation="horizontal" to have a horizontal list of radio buttons.

Adding an attribute **list-type="listbox"** on fi:styling produces a selection list, use the attribute list-size to specify its size (default 5).

2.3. fi:action

By default, creates an <input> of type submit, thus showing a standard button. To have an image button instead, try this:

<fi:styling type="image" src="blah.gif">

2.4. Other widgets

Not yet documented here, but don't be afraid to look at the source of the XSLT's.

3. High-level styling with fi:group

No documentation yet, checkout the samples and the source of forms-page-styling.xsl.

For storing the state of a tab or choice selection server-side just add a field to the form definition that shall hold this value:

<fd:field id="state"> <fd:datatype base="integer"/> </fd:field>

Bind this value to whatever you want. In the form template you need then following code:

<fi:group> <fi:styling type="choice"/> <fi:state> <ft:widget id="state"/> <!-- refering to the above defined field --> </fi:state> <fi:items> ... </fi:items> </fi:group>

4. Miscellaneous

4.1. fi:validation-errors

The fi:validation-errors tag is used to display all validation errors of all widgets in a form at one location, i.e. a the top of the form.

The fi:validation-errors tag must be a child of a ft:form-template element.

You can customise a message to be shown before and after the errors by adding a child header and/or footer element:

<fi:validation-errors> <header> Correct these errors please: </header> <footer> And then resubmit the form. </footer> </fi:validation-errors>

1. Comments

add your comments