WoodyXSLT

This page contains information on the default XSLT's provided by Woody, and the **wi:styling** directives they support. Be sure to first read about the Woody TemplateTransformer.

There are basically 3 XSLTs:

- `woody-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.
- `woody-field-styling.xsl`: contains templates that style individual widgets, i.e. templates that translate wi:field, wi:booleanfield, wi:action, etc. to HTML.
- `woody-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:

- `woody-advanced-field-styling.xsl`: contains templates that provide advanced styling of fields, e.g. the "double-listbox" for a multiplevaluefield. It's indeed an extension of the above basic `woody-field-styling.xsl`. Furthermore it includes the next stylesheet.
- `woody-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 `woody-advanced-styling.xsl`.

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

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

wi:styling options

wi:field (without selection list)

By default all attributes on the wi: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:

```
<wt:widget id="myfield">
  <wi:styling size="50" class="kinky"/>
</wt: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
- **type="date"**: will put a little icon next to the input field that, when clicked upon, shows a calendar popup. Using an attribute named format you can specify the date format (default is yyyy-MM-dd)

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

wi: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 wi: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 wi:styling produces a selection list, use the attribute list-size to specify its size (default 5).

wi:action

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

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

Other widgets

Not yet documented here, but don't be afraid to look at the source of the `woody-field-styling.xsl` XSLT.

High-level styling with wi:group

No documentation yet, checkout the samples and the source of `woody-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:

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

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

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

Miscellaneous

wi:validation-errors

The `wi: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 `wi:validation-errors` tag must be a child of a `wt: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:

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