Configuring Sakai/OSP: How to develop an Sakai/OSPortfolio XSL template

This is a two-step process. First you'll need to get your hands on the XML that you'll be creating your XSL for (you do this by creating a pass-through template so you'll have the resulting XML to work with) Once you have the XML, you can develop the XSL and your XPATH expressions on your computer with minimal fuss.

That is, you do NOT have wrestle with OSP to upload your trial XSL, create a temporary template, create a temporary portfolio, find the bugs, fix your original XSL, try to revise the XSL resource only to find out that you can't because it's in use by the template, then try to revise the template to find out you can't only because it's in use by the portfolio, destroy the portfolio, destroy the template, revise the uploaded XSL... that's madness! There's an easier way:

Part ONE:

- Install "passthrough.xsl" (attached) to your worksite as a resource (you could instead use "tree-view.xsl" (also attached) for a human-friendly version but it won't help with development turnaround time much)
- Define a template based on it: "Basic Template Outline" will be "passthrough.xml"
    - If you intend to use "Outline Options" (from "step 2 of 4" when creating your template) include that part here for "passthrough.xml" as well

- You can use anything at all as your "List Content" (when installing the template in Sakai 2.2.1 this is called "step 3 of 4") including matrices, forms, or files -- whatever you intend to use in your final template
    - Note that the "name" used in this step will become a node name in the XPATH expression for your XSL (for example, naming it "matrix" will give you /ospiPresentation/matrix/artifact to work with)
    - The "title" here will be a note to the user, something like "Select a matrix" or "Which artifact do you want here" would be appropriate
    - Make sure you click "add to list" for each type of artifact to include in the result output

- Now create a presentation based on the "passthrough" template and include as many different types of artifact as possible so you'll have lots to work with. Forms, PDFs, video clips, matrices, anything goes.
- The results will look awful in a web browser, but that's expected -- you've got structored XML there instead of HTML. View-source to see the XML, and save it to your local storage.
- If you included the "allow comments" option you'll need to ditch the trailing HTML that handles the comment: your XML stops at ‹/ospiPresentation› and anything that follows it must die!
-

  Insert an ?xml-stylesheet? directive into the XML you just downloaded:

  ```
  ‹!DOCTYPE ospiPresentation PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  ```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<?xml-stylesheet type="text/xsl" href="your-template-here.xsl"?>
<ospiPresentation>
...
```
Your web browser will now use the referred-to template to TRANSFORM the XML into something else (typically, browser-friendly HTML).

That's right -- use Firefox or Internet Explorer to open the XML file directly, and as long as the ?xml-stylesheet? directive is properly set, your XSL will transform it on-the-fly!

Part TWO:

- Now you iteratively develop XSL directives in "your-template-here.xsl" to produce the HTML you're after. As you do, **merely refresh your browser**'s view of the XML you downloaded from "passthrough.xsl". Once your template is working the way you want it to, you're ready to roll...
- Upload your XSL as a resource into your OSP worksite
- REVISE your existing portfolio using the new template XSL in place of "passthrough.xsl" -- that way it'll have the same steps and options required by your XSL
- Create a portfolio based on the new template, and fill it up with all kinds of stuff. Voila!

Note that you might also benefit from trying tree-view.xsl (also attached) for a more human-readable version of the structure of your XML results.

Drawbacks/Caveats:
Okay, there's one small snag we've discovered so far: disabling output escaping: if you expect to use <xsl:value-of select="something" disable-output-escaping="yes"/> in your final XSL, don't panic when your markup is escaped anyway -- passthrough.xsl doesn't disable the escaping, so it passes it through escaped already, and your XSL isn't able to undo that.

Sakai is a community-source or open-source LMS (learning management system) or CLE (collaborative learning environment); OSP is an electronic portfolio toolkit within Sakai.

Unique solution ID: #1017
Author: will trillich
Last update: 2008-01-27 12:02