Michigan Scaffolded Portfolio Authoring Work

University of Michigan - Portfolio Authoring

Michigan has done extensive work to improve the experience of authoring portfolios. We have worked closely with students to understand their needs and frustrations and have developed a number of enhancements. These concentrate on a concept we have informally dubbed the "scaffolded portfolio", which provides suggestive guidance, but attempts to avoid any hard requirements.

A scaffolded portfolio is implemented with an OSP Portfolio Template and generic Forms, using one Form per page of the final presentation. The intensive work with learners has concentrated on showcasing previous accomplishments in leadership and research, so the pages fall in two categories: Basic Page and Example of Work. This augmented editing mode for Metaobj Forms is locally referred to as the "Page Composer".

Note that this work depends on functionality that is included in Sakai 2.5.0, but can be included in 2.4.x. In fact, U-M currently uses it in production 2.4.x. See the Page Composer 2.4.x Compatibility Notes for more details.

You may download a comprehensive package for trying out the Scaffolded Portfolio / Page Composer. Included in page-composer.zip are all of the materials and instructions to set up a Scaffolded Portfolio we call the Leadership Portfolio with minimal effort. It can be ready for use in about 15 minutes on the standard Sakai 2.5.0 release, without requiring any restarts or configuration changes. See the README.txt file for a quick-start and the docs/ directory for a detailed walkthrough.

Download page-composer.zip to try the Scaffolded Portfolio and Page Composer
View a Camtasia video of the Page Composer

Issues being addressed:

- Lack of context and styling
- Ease of losing work
- Expressiveness
- Sustainability across projects

Strategy:

- WYSIWYG editing in context
- Autosaving / navigation blocking
- "One Big Editor"
- Separate portfolio assembly from Matrix

Techniques employed:

- Standardized, structural HTML/XSL
- Pure CSS themes
- Generic Metaobj Forms
- Custom JavaScript inclusion through XSL

Issues

Lack of context and styling

When using the standard configurations for editing portfolio content, the result is quite unsatisfactory. The stock settings for the FCKeditor result in an editing area that is small and does not apply the final styling of the content. It also does not provide any of the surrounding target context, such as the banner or footer. This causes many delays by forcing learners to preview the content repeatedly and attempt to compensate for styling or layout without predictable results.

Ease of losing work

When editing content that is personal and intended to have specific styling, it is very easy to spend a great deal of time in one editing session. Given that the content is meaningful and takes extended effort to create, it is very frustrating to lose this work. Any number of small mistakes or glitches - such as hitting backspace without a text field selected or a network connection dropping - can cause confounding loss of work.

Expressiveness

When working with portfolio content, expressive ability is important to the learner. This means, primarily, that rigid structures assuming too much control over the final output or difficulty in completing a familiar task like aligning an image are very discouraging. These impediments actually hamper creativity and limit both the content and formatting of finished portfolios.

Sustainability across projects

The technique that has emerged in the OSP community as standard practice for working with Portfolio Templates involves extensive XSL work, specific to the kinds of Forms and Matrices used for a given project. Minor variations to those structures or the output are difficult and time-consuming, and present significant sustainability concerns when working with multiple projects.

Strategy

WYSIWYG editing in context

The most pressing demand from users is to not be surprised by how their content is presented in relation to what they can view while editing. By placing the FCKeditor within the visual context of its destination and applying the target CSS rules to the editor, a nearly WYSIWYG experience is delivered. This removes the slow and frustrating steps of saving and previewing with each adjustment.

Autosaving / navigation blocking

To avoid the loss of work, two overlapping strategies can be applied. These are autosaving of the content and prompting for confirmation when navigating away from an editing page. The combination strategy covers most common data loss scenarios redundantly for high confidence.

"One Big Editor"

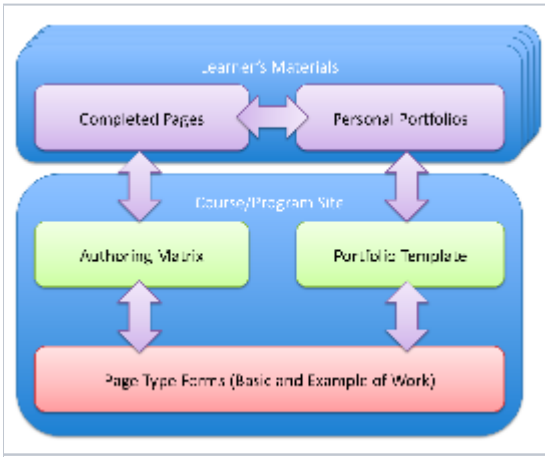
To address the expressiveness concerns, it is reasonable to consider each "page" of a Portfolio as a single, large, editable area, surrounded by some structure and navigation. This avoids the need to create and modify Forms to collect data, and then subsequently display the structured data. It also allows the learners control of more of each final page, which is almost universally requested.

Separating portfolio assembly from Matrix

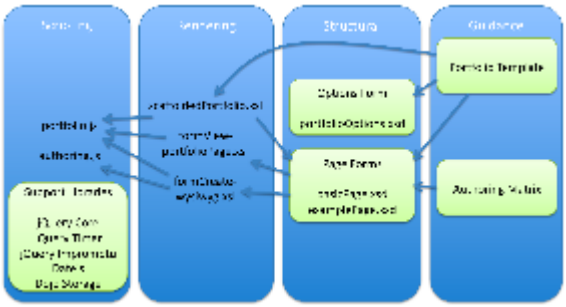
A common strategy is to provide learners with guidance on a suggested portfolio structure. Within OSP, this can be implemented by developing Portfolio Templates that reference a Matrix structure and specialized Forms. Doing this provides some standardization, but at a high development cost due to complexity in the XSL, and a cost of user expressiveness. An alternative strategy lets learners select the individual artifacts for inclusion, which reduces complexity and allows remix capability for different target audiences.

Techniques

To deploy the strategy outlined above, the Scaffolded Portfolio and Page Composer have been developed. Here are overview and more technical diagrams, respectively:



Overview



Technical

Standardized, structural HTML

Rather than using HTML specialized for a given layout, we have created a generic structure that has a limited number of semantic components. There are header and footer sections, primary and secondary navigation, and a container for an unbounded number of "pages", to be shown or hidden with script. This shift has proven to have extreme cost advantages by reducing the XSL maintenance to near zero.

Pure CSS themes

With structural HTML in place, everything for layout and styling can be applied with pure CSS. This simplifies skinning dramatically and allows design talent to be applied without requiring intricate XSL modifications.

Generic Metaobj Forms

Another major component of the cost of maintaining the XSL transforms for presentations is the reliance on specific Form elements. By simplifying the portfolio model into using generic "page" Forms, the XSL relies only on minimal structure, which can be used by convention. Combined with the generic HTML, this allows very cost-effective variations of a Portfolio Template with almost no modification to the complex underpinnings.

Custom JavaScript inclusion through XSL

In order to deliver the experience of the WYSIWYG editing with on-the-fly theme selection, autosaving, auto-sizing editors, and the switching between embedded pages in the final presentations, script support is needed. The custom Form creation, Form viewing, and Portfolio presentation XSL files provide a place to reference script files without modifying the Sakai build. This flexibility has proven essential to evolve the experience in real-time. Hosting the files within Resources is an elegant solution that requires no extra infrastructure.