TransformAble 2.0 Roadmap

This document provides a roadmap for the integration of parts of the TransformAble suite of tools into the Fluid community. It was written by Colin Clark in June 2008.

Introduction

TransformAble is a suite of Web services developed in 2006 as part of the CulturAll project. Building on the approach pioneered in our BarrierFree and TILE projects, TransformAble's goal was to offer a suite of services that could be integrated into any Web application to provide personalized UI and content transformations. Each service makes extensive use of the ISO AccessForAll standards for content metadata and user preferences. The services include:
• PreferAble, a preferences editor
• StyleAble, a service for transforming a Web application's UI
• SenseAble, a service for transforming content stored in a repository

In the end, only portions of TransformAble were fully completed. They were, however, successfully integrated into Sakai 2.3 and are available to the community as optional "contrib" tools. More information on TransformAble is available at http://transformable.atrc.utoronto.ca/. A live demo of TransformAble in Sakai is available at http://transformable-sakai.atrc.utoronto.ca/portal.

Today, only portions of the services are still working, and the source code is scattered across several SVN repositories. The code quality of TransformAble is mixed; there are a number of very well-written modules, while others were hastily implemented, poorly designed, and don't provide a workable foundation for future work in this area.

This roadmap, a strawman only, attempts to outline a sustainable plan for the TransformAble services as part of the Fluid community. Within, I make a couple of (hopefully) modest assumptions: 1) that we want to preserve working software at all times, and 2) that the code we produce should be well-made enough to serve as a foundation for future projects. This roadmap is the product of consultation with Anastasia, Jutta, and other members of the Fluid team.

Roadmap

Step 1: Consolidate the source code for the legacy version of TransformAble

With the help of Ian Boston, PreferAble is working again in Sakai and is compatible with version 2.5. The source code is temporarily available in CARET's CamTools source code repository:

https://source.caret.cam.ac.uk/camtools/trunk/camtools/transformable/

Next Steps

- Consolidate all of the TransformAble code in the Fluid SVN repository.

Step 2: Implement PreferAble 2.0 as a client-side Fluid component

Summary

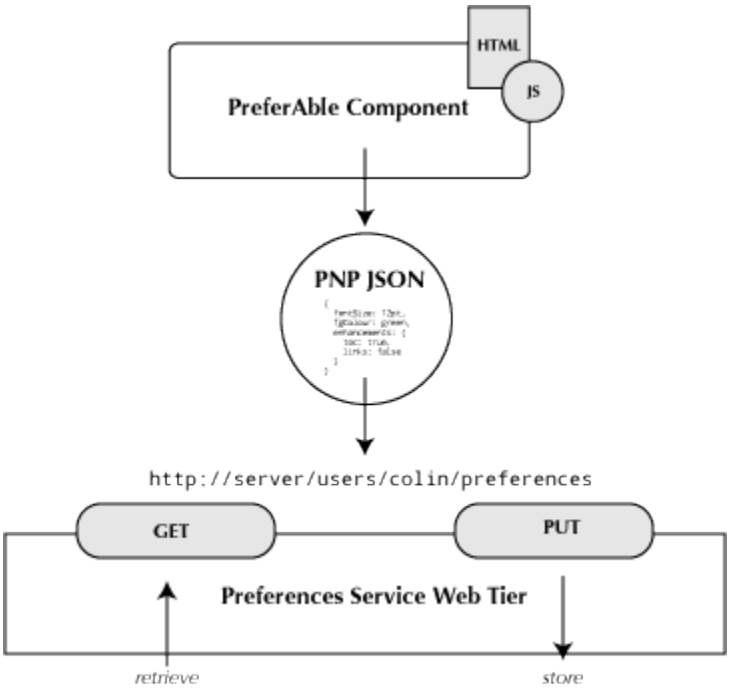
• Implement PreferAble 2.0 in JavaScript
• Create a lightweight PNP binding for JSON
• Establish RESTful conventions for storing and retrieving preferences on the server

Details

A refreshed version of PreferAble was listed in the original Fluid proposal as a primary deliverable. Our current plan lists it as being part of the Infusion 0.6 release in September 2008, though this is probably an ambitious deadline. It will undoubtedly need to be reworked in the Fluid way, but it should, in some form, play a role in the accessibility features offered by Fluid.

To get PreferAble to the point where it can easily be embedded in a wide variety of Web applications, we need to bring it in line with the Fluid component framework. PreferAble 2.0 will be built as an entirely client-side Fluid component using our new template rendering engine. Preferences will be conveyed as simple, JSON-based objects representing the AccessForAll PNP (Personal Needs and Preferences). This new binding will address the substantial weaknesses found in many of the current PNP implementations by providing a straightforward API with a flat hierarchy and a loose type system. A PNP validation library will be paired with this data model to ensure integrity and spec compliance, but without requiring the onerous data typing and nesting of our current Java-based implementation.

We'll also define a set of RESTful conventions for storing and retrieving user preferences in AccessForAll format. To integrate it with Sakai and uPortal, we would likely build an RSF-based to tool to manage the URL space and service calls. Other applications, such as ATutor or CollectionSpace, could easily build their own services with a minimum of effort based on this approach.



Risks

PreferAble will need to be redesigned and updated with input from our design team. While I can point to some existing models for inspiration, such as the BBC's new portal site [1], we'll need to identify an experienced designer who can represent the needs and interests of people with disabilities, along with the ATRC's personal optimization research goals.

Step 3: Merge StyleAble into Fluid's acccessible infrastructure

Summary

• Offer StyleAble's appearance customizations as a pure JavaScript service available to all Fluid components
• Refactor the content navigation functionality into Fluid components

Details

The StyleAble service currently has two main capabilities: 1) generating custom CSS style sheets from PreferAble's user preferences, and 2) injecting navigation enhancements such as a table of contents and list of links into a page. StyleAble was originally implemented as a blend of server-side Java and JavaScript.

Several aspects of StyleAble's functionality were challenging to implement due to a lack of clear semantics that would allow us to adjust the styling and navigation accurately. In Fluid, where we have an additional layer of stable semantics, specifically ARIA and our Infusion CSS theme, we have the potential to alter the appearance and styling of components more successfully than in our previous efforts. With this in mind, a pure JavaScript-based StyleAble service will start help us illustrate how Fluid's markup-driven architecture can provide advanced styling flexibility.

StyleAble's navigation enhancements may serve as useful Fluid components. In addition to providing quick ways to navigate through a page for keyboard-only users, they also provide useful functionality for content authors. As Fluid components, the Table of Contents and List of Links would provide a simple way to embed page summaries and navigation within any page, similar to Confluence's `toc` macro. Given that these are already written in pure JavaScript, they should be relatively straightforward to port to Fluid.

Risks

Many of StyleAble's appearance customizations overlap closely with the styling preferences in modern browsers. Firefox 3's new Zoom capability is astonishingly good, and we may choose to throw away much of our own code and recommend that our users use Firefox instead. More user-centred design thinking needs to be invested to come up with a forward-looking feature set for StyleAble. Similar concerns apply as with PreferAble; we need to ensure that we have skilled contributions from designers who are committed to personalization and accessibility. Since the navigational components aren't directly on the component matrix, we may get push back on adding these to the Fluid roadmap without some use cases to back us up.

Step 4: Put SenseAble to bed

The landscape of content management systems and repositories has changed significantly since we embarked on the TILE and TransformAble projects. There are now several mature repository solutions available in the form of JCR/Jackrabbit, Fedora, D-Space, and others. Additionally, the emergence of collaborative, content-driven sites such as YouTube and Flickr has changed the architectural requirements and user expectations for a Web service like SenseAble.

In its current form, SenseAble is a collection of half-finished Java classes and unit tests that haven't been assembled into a coherent product. You can't run or try out SenseAble--it's not an application, it doesn't have a user interface, and it doesn't actually do anything. It's just a sketch in code. SenseAble is built on top of the Java DRD data model, which suffers from acute over-engineering and an overly complex API. In short, its usefulness as a foundation for future projects is limited.

On the other hand, the conceptual model for SenseAble is both solid and innovative. Future projects in this area will undoubtedly explore participatory and collaborative environments for video, multimedia, and shared learning resources. This will require new technical designs to integrate with content repositories and authoring tools. We'll also want to explore the role of metadata inference [2] and content mashups [3] in StyleAble. These are all approaches that demand significantly different architectures from those explored in TILE or SenseAble.

With this in mind, we should assume that future projects related to content transformation will need to start from scratch. Given the changes in the Web 2.0 space, along with the poor quality of code in SenseAble and TILE, we'll need the time and resources to re-envision the functional and technical requirements of SenseAble. Given this, I'd suggest that no further development of SenseAble or the Java-based DRD bindings should occur until we have the opportunity to start fresh and explore the problem space anew.

Notes

[1] BBC's Display Options feature: http://www.bbc.co.uk/displayoptions/index.shtml?url=www.bbc.co.uk/

[2] Think about the iTunes experience. When you go to rip a CD, you don't have to painstakingly enter all the metadata for your tracks. iTunes automatically goes out to several services on the Internet and grabs as much information as possible. With user-contributed plugins, cover art can be automatically pulled from a variety of sources including Amazon's web services API and dedicated fan sites. An authoring tool for SenseAble should be dedicated to figuring out as much as possible about a particular piece of content using other Web services before asking the user to intervene.

[3] The concepts of content "aggregation" and "substitution" as we first conceived of them in TILE have found a more natural model in the kinds of Web 2.0 service mashups we've seen in for Google Maps, Yahoo Pipes, and so on. Applied to the open education space, imagine the potential of coupling SenseAble's transformation capabilities with solid authoring tools that can assemble new content from a variety of feeds and services.