Tagline

A group project which involved organizing and structuring information before publishing the information to multiple mediums.

Introduction

This was a group project I did in a class called Cross Media Publishing; the projects main goal was to improve the presentation and organization of a set of existing documents that had been published by our school’s Digital Publishing Center (DPC).

Information Structuring

Initially, we created user personas, use cases, and wireframes to focus the project scope and determine a structure for the different information types in in the documents. The information structure was then used to create an extensible markup language (XML) that was applied to the information from the original DPC documentation.

Document Generation and Hosting

Lastly, we generated a set of HTML documents and a set of printed documents, with both document sets originating from a single set of XML files. The HTML documents were hosted on a mobile friendly website that used AJAX to quickly switch between the documents.

Approach

User Personas and Use Cases

Our initial group task was to brainstorm possible users who would access the documents, and the different use cases they could have when using the documents. We came up with three personas: Bridgette, a fourth year graphic design student who was familiar with technology and wanted an easy to use mobile application; Ron, a college graduate who needed intuitive and accessible documents; and Samantha, a professor who was interested in technology, but was disorganized and had slow internet.

This process forced our group to consider all of the different users that could attempt to access the DPC documents, which was important to ensure our end product would be accessible to all of our potential users.

XML Schemas

With our example personas and use cases in mind we created our project’s XML schema. Our schema defined the different types of possible information that could be contained in the documents, the hierarchical relationships of the different information types, and the allowed number of occurrences of every information type.

For example, one information type we found in the documents was a title—our schema specified that every document should have one and only one title, and. Another possible information type was paragraphs—our schema stated that the number of paragraphs in a document should not be limited, but a paragraph should always exist in a section.

We found the documents had a title, description, publishing date, and often had sections, subsections, headers, and paragraphs. Each of these things became a different information type in our schema. A document should always have a single title, description, and publishing date; however, a document does not have a set number of sections or subsection, but if a subsection does exist it must always be contained within a section.

This is an abbreviated example of how we used a schema to help distinguish and organize the different information types and information relationships within the set of documents.

XML Document Markup

With all of the different information types defined in our schema, our groups next task was to create our own XML markup language that assigned a different ‘tag’ to each specific information type in the documents. We then split up the documents between the group members, and applied our markup language to the information within the documents.

After this step was completed our group was left with documents that were fully marked up with xml tags that matched our information structure that we had defined in our schema. With all of the information in the same markup language we were able to begin generating our final document outputs.

XSL

Our group used XSL, a family of languages that allows users to navigate and transform XML documents, to generate our final documents. We used XPATH to navigate through the different levels of our information hierarchy that we established in our schema and applied to our XML. Once we navigated to a node in our XML document we would use XSL to perform an appropriate transformation to that node depending on the type of our document output (print or html). XSL allowed us to find different elements in our XML documents and apply a transformation to them depending on the type of our document output (print or html). We used the software “Oxygen” a program called “Saxon” to run our XSL code on our XML documents.

Print Documents

The group’s print document team used XSL to transform the team’s XML documents in to XSL:FO format. XSL:FO is a specific XML markup that is used for page layout of XML documents. XSL:FO specifies the different page elements which then have specific design formats applied to them in an XSL:FO formatter. The result of the XSL transformation and the XSL:FO formatting is a pdf document, which contains all of the information from the original document (title, sections, paragraphs etc.), that is indistinguishable from one that was created in InDesign.

HTML Documents

Our group’s HTML document team used XSL to create HTML files that would be hosted on a webpage. This process is different from the print document generation process; first a static html “skeleton” is created, and then XSL statements are placed in the skeleton. These XSL statements grab certain information from an XML document and place that information within the HTML skeleton.

For example, one XSL statement would search for a heading in the XML document and then place that heading in an area of the HTML document surrounded by an appropriate HTML tag.

With the organized information from the old documents in HTML form, we could now apply CSS styles and other interactive elements to the documents to improve the user’s experience.

Wireframes

Wireframes were created to aid in the planning of the website that would host the finished HTML documents. The wireframes specified the possible appearance of navigation elements, menus, and the website homepage and document page. Wireframes for the mobile version of the website were also produced.

Website

In the planning stages of this project, my group determined that accessibility and ease of use were paramount to our users. With this in mind we created a desktop and mobile layout for the website so that users could access and use the online documents from a computer or mobile device.

The website is simple: the homepage presents a user with three document categories choices and the desktop layout includes dropdown lists of the category’s documents. If a user selects a category they go to a page that shows each document, a description of that document’s content, and a link to that document page. Menus on the document page allow users to switch to a new document category, or switch between the documents in their current category.

*AJAX*

The website incorporates Asynchronous JavaScript and XML (AJAX) in the navigation between categories and documents. AJAX allows the website to only reload the parts of the webpage that need to change when a user switches documents or categories, which is must faster than reloading the entire webpage. If a user changes to a new document, the information in the document area is erased and quickly replaced with the new document information that was stored on the server.

Summary

This project took students through all of the steps of creating a real-life cross media publishing application. The project was very process heavy—we spent much of our time planning and creating information structures so that we could streamline the process of publishing the documents to multiple media, and so that users would be able to access the documents and understand the information better than they originally had.

Future Improvements

The homepage should have an initial description that explains the Document Publishing Center and the document categories to first time users.

Because the website loads new content using AJAX, and does not reload the entire webpage, if a user hits the back button it will take them off the website. This is disorienting for users who are used to the back button taking them to the last page—I think the initial selection from the homepage should reload the entire page, so that if a user clicks the back button the browser will load the homepage instead of taking them away from the website.

The document paragraphs have a first line indent and a space before the next paragraph. This is unnecessary and wrong, and the problem could be quickly fixed by changing the paragraph’s CSS properties.

The dropdown to switch to other documents in the same category provides document descriptions, however, the descriptions are long and their font size is very small, especially on mobile devices. The descriptions should be shorter and more readable.