

A Simple Web Content Management Tool as the Solution to a Web Site Redesign

David Thomas Dudek
Senior Programmer Analyst
University at Buffalo
201 Computing Center
Buffalo, New York 14260-1408
001-716-645-6363
dudek@buffalo.edu

Heidi A. Wieczorek
Senior Programmer Analyst
University at Buffalo
201 Computing Center
Buffalo, New York 14260-1408
001-716-645-5069
lotter@buffalo.edu

ABSTRACT

Before embarking upon a large web site project, whether creating a new site or just redesigning one, it is generally accepted practice not to begin work until all the problem specifications are ironed out. Some of the information that is usually required is the audience of the site, the intended purpose of the site, and the scope of the content provided in the site. The technical specifications of the site, such as accessibility, usability, file format, layout, and style, are then usually derived from this information. This is textbook web project management methodology.

At the University at Buffalo, we found ourselves in a real-world, less-than-ideal situation with our IT web site redesign. Without possessing all the preliminary information, we had no choice but to begin writing content and designing our site. This was with the full knowledge that several of the project specifications would change, or not be known to us until we were well into building the site. Interesting questions arose such as how we were going to efficiently make future layout and navigation changes that would propagate throughout the site.

Additionally, there were other challenges we had to face that were typical of any large-scale web project. For example, with multiple authors with varying degrees of web publishing expertise, how were we going to maintain consistency of style, layout, and file format throughout the site?

We felt that these challenges provided a perfect opportunity to implement a simple web content management tool. By creating units of content separate from their presentation, we gave ourselves the flexibility we needed to easily rearrange and redesign our site along the way. In addition, we simplified the process of editing and overhauling the site in the future.

Categories and Subject Descriptors

H.5.3 [Group and Organization Interfaces]: *Computer-supported cooperative work, Web-based interaction, Collaborative computing;*

H.5.4 [Hypertext/Hypermedia]: *Architectures*

General Terms: Documentation, Design, Human Factors

Keywords: web, content management

INTRODUCTION

The University at Buffalo is the largest institution in the State University of New York (SUNY) with approximately 17,300 undergraduate students, 8,500 graduate students and 7,000 full-time faculty and staff. It is a public university with approximately 50% funding by the State of New York.

For several years, the Computing and Information Technology (CIT) website was merely a collective of individually maintained websites. Obviously, this had its problems, mostly as the result of several inconsistencies between the pages. In 1999, a staff member was appointed full-time central webmaster. His job was to tie all these individual sites together with improved navigation and a consistent look and feel. He accomplished this with varying levels of success. When he left CIT in 2001, his webmaster duties went mostly unfilled. Once again, the website fell into disarray. In January of 2002, we became part of a group within CIT that was charged with overhauling the aging CIT central computing services website by the beginning of fall 2002.

MISSION AND CHALLENGES

Our mission was to provide an information service regarding central IT resources using at least three methods:

1. A centralized web site at <http://www.cit.buffalo.edu/>
2. Our own campus web search engine and other third-party search engines such as Google.
3. Various web portals on campus - now and in the future.

We were required to provide an information architecture design that was clean and simple, and accessible directly to the primary document authors. We also needed to ensure that information was accurate, timely, useful, accessible and include expirations for temporary information. Therefore, we had to provide tools to facilitate the periodic review of our document collection. Since information technology has become critical on campus, it was imperative that this information service be highly available.

It may seem at this point that our mission was cut and dry. However, due to problems that go beyond the extent of this paper, the scope of our project was uncertain. As noted earlier, the CIT website was a loose collective of individually maintained websites. We knew that it was impossible to get all of those sites migrated into the new one within the timeframe allowed. Some of these sites were not just static HTML; some of them were dynamic sites or required specific technologies. Furthermore, there were rumors of a major reorganization of the campus IT departments. The only thing for certain was change.

APPROACH

Early on in our project, we decided that there would be two parts to our solution. The first part of our solution was the separation of content from its presentation. This would allow us to make changes to our site without rewriting documents. Furthermore, it would allow the web portals on campus to use our content in their site, providing a seamless experience for the campus.

The other part of our solution to this problem was a formal editorial process. Although there are several authors for content on our site, we decided that there should be editors that have the final say as to what gets published. A style guide was created for the authors and the editors that covered language and markup styles, as well as a common lexicon for us to use.

We needed to get the majority of our content up on our new website in a matter of two months. We came up with many ideas: just using Dreamweaver to installing a commercial or free ware version of a content management system. We needed to come up with a main page, that was dynamic or that could be built on the fly. We wanted the navigation to be on all pages. We also wanted a consistent look and feel for the entire site.

SOLUTION

Obviously, we needed a content management system. Given the time frame in getting a CIT Website up and working, we had many restraints. First, we did not have time to fully investigate a commercial or shareware version of a content management system. Second, we had limited technical resources in creating this web site. We were very limited because there was only one server available to us that would provide the high availability that was required. That server has a very stable environment, and installation of new software requires a long period of testing. The only currently available programs or publishing tools on this server were Perl CGI, HTML, and PHP. Perl CGI and HTML were our main focus, due to the programmer's expertise in these areas. The HTML was also an issue because we had a variety of contributors and their knowledge or lack there of HTML. Most of the content contributors use a HTML editor like Dreamweaver, and did not worry which tags were being used, or whether they were accessible or not.

TECHNICAL DETAILS

The content management tool is a Perl/CGI script that uses DCE for authentication. Perl was chosen because it was a fast and easy way to script all of the pages to be built and to build the system itself. Authentication is used to determine access to the tool. The tool is designed so that multiple authors may provide content for the site. Authors may create new directories or files in a test site via a web interface. The test site has a separate URL than the production site via Apache virtual hosts. Authors may also modify files in the test site via the web interface. Essentially, the web interface provides text fields for the document title, a short description of the document, search keywords, and the actual text body of the document. We allow HTML markup in the body. At any point during editing, an author may click a button to create the document in the test site. Two files are created: a marked-up file containing only the page content and the final HTML page with the site's look and feel. The CGI builds the static HTML page using using HTML templates which provide the banner graphics, navigation bars, header and footer text, etc... The CGI also "stamps" the document with an HTML comment field which

includes a timestamp and the authenticated author of the document.

A small group of users of this content management tool were designated as editors. In addition to the ability to modify content in the test site, editors have the ability to click a button on the content management tool web interface labeled "commit to live". Clicking this button copies the files in the test site over to the production site to be made public. Also, if the look and feel or the navigation of the site changes, editors may change the template and run a script to rebuild the entire site using the content text files.

We wanted to create a content management system that would be accessible, easy to edit or deploy mass changes without touching each file. We choose to use as a layout, a generic header, footer, and navigation frame that would appear on every page. This, we felt would give it a general look and feel and would make it appear that all of the content was coming from one source but in actuality was coming from multiple sources. For this we choose to separate content from layout, and to do this we would have to build each page of the site on the fly. We wrote a few additional Perl scripts to join content and look and feel (layout). We had a generic header, footer, and navigation files, which join with the content, would build a file with all four parts.

We choose to use Cascading Style Sheets (CSS) to give the site a similar look which could be changed in one place. We wanted to change the look on a yearly cycle. Also, CSS was very much needed to implement the style that we originally agreed upon. So many of the deprecated HTML tags needed to be addressed by using CSS. We choose to create our own way of managing content based on the short time frame that we needed something to be done, based on the group's general knowledge of HTML, tags, CGI, and based on giving the site a general look and feel. We decided Dreamweaver or Front Page were not an option based on a learning curve for some and based on the program adding its own tags to our original content.

To address the content issue we wanted our content to be separate and easily available to load into something else in the future. Some considerations for future use are XML and/or a database driven content site. A delimited flat file was the simplest way to separate content for later re-use. It was also an easy way to build our HTML files on the fly with the header, footer and side bar attached. The script to do this basically takes seconds to run and reproduce the pages if a change needs to be made.

The directory design began to play a major part in the easy of content separation, for each topic we created a separate directory. For each file in that directory there is a corresponding html page, named by the specific topic. Backups were an issue so every time content was changed a backup file was retained. The content tool that was created had to be able to let the user create new directories so that they could separate content by topic, in a relatively easy way. Also, the editor of the content was also saved, as to track down any changes to a specific person.

Security was another issue, we wanted to restrict user to our software by User Id, and so we added authentication to this program. Because we had so many users adding content we added a cgiwrap to the content tool so that all files were owned by an admin user.

To make the tool user friendly we came up with certain items that were needed to create each page. They were Title, Meta

Description, Meta Keywords and Body. So basically we had data entry boxes for each item in which the content would be entered, and the CGI program would add the appropriate tags to it. The only deviance from this was the BODY input. This input could contain tags such as bold, underline, bullets, etc.

We also needed an area for test and for production. First all content was added to test and previewed via the preview button or by going to the test URL. Once a writer had completed their content an editor would review it and publish it to production. We wanted the flow and the grammar to appear as one and the some person. So we created a Style Guide for general use and then a final edit was done to finalize the content. The publishing of content was locked down to one person with a supervisor as a backup. The keep the content under control with one point person to question in regards to what was written.

RESULTS

Our content management system worked for most of the pages that we needed to incorporate into our site. However we did encounter some limitations. First of all, our system could only

work with a flat directory structure. This was not a problem for most of the pages, but it was for our documentation staff, who maintain such a large number of documents. Also, we were unable to incorporate web forms into the system. This is a bug that we would need to fix if we decided to continue to use the system.

Overall, the project was a success. However, we will be looking into replacing our home-grown system with a commercial or open-source content management system. We believe that we formatted our individual documents in such a way that it will be relatively easy to migrate them into a new system.

REFERENCES

Student Computing Web Project website:

<http://wings.buffalo.edu/computing/scw2002/>

The new CIT website:

<http://www.cit.buffalo.edu/>