

Report on Content Management Systems

University Communications
Web Services Office
March, 29 2010

Table of Contents

Overview	1
Our Current Situation:.....	1
Our Problems:	1
What We Need in a CMS:.....	1
concrete5	3
Our Criteria	3
Review Notes	3
Impress CMS	4
Our Criteria	4
Review Notes	4
Joomla!.....	5
Our Criteria	5
Review Notes	5
MODx 1.0.2	6
Our Criteria	6
Review Notes	6
Symphony CMS	7
Our Criteria	7
Review Notes	7
TYPO3.....	8
Our Criteria	8
Review Notes	8
Ektron CMS400 .NET	9
Our Criteria	9
Review Notes	9
dotCMS.....	11
Our Criteria	11
Review Notes	11

Overview

Our Current Situation:

- Web pages exist as files on the server – traditional HTML and PHP
- The look and feel are produced by a complicated set of include files and CSS
- Editors edit files locally and transfer them to and from the web server via SFTP
- Access to files is controlled through standard UNIX user/group/other file permissions
- Editors upload media to whichever folders they are able to publish web pages

Our Problems:

- Our editors are either unknowledgeable about web authoring or woefully out of date, but they have access to edit (and possibly break) *every* part of the page.
- Because our editors are generally not technically competent enough to manage traditional web pages, they rely on technical support. For administrative departments, that responsibility falls on the web office –but for academic offices that responsibility falls on each college or school’s own IT department. This approach ignores the obvious problem presented by the college and school IT departments being completely unable to perform any of the IT functions of website administration (which is wholly the domain of OIT systems and security) and that these IT departments are being tasked with functions of the University Communications office.
- The look and feel of the pages can be changed by editors who have access to the HTML header and includes.
- Editors often work on out-dated pages and overwrite newer copy. They are also unable to save a draft on the server or to work on a page from a computer without specialized software.
- UNIX file permissions are *extremely* limited. Also, they must be administered via the UNIX prompt making it difficult to get a broad overview or to make changes without being destructive.
- Editors aren’t adequately skilled in using HTML to properly embed media files. Also, with distributed editing, there’s no standard way to present media files and repackaging media into other forms (i.e. podcasts) is very difficult.
- While files are backed up at regular intervals not every version of each page is stored and retrieving files can be a difficult process requiring IT intervention.

What We Need in a CMS:

With respect to the specific problems we face in maintaining and authoring the Montclair State University website, the following list represents a set of features and properties we hope to find in a CMS and how they should help alleviate the above-listed problems.

- **A separation of content and presentation** – Our website editors are not designers or programmers. They should be focused wholly on writing content, not fonts, colors, or code.

- **Page versioning** – every version of each webpage should be backed up at the time that it's created and should be retrievable by the editors without requiring IT intervention.
- **Robust permission settings**
 - Group-based permissions – content assigned to groups, people can be moved in and out of groups by an administrator
 - Publishing and Editing – the CMS must support the ability to edit and save content without making those changes visible to the visitor.
 - Granular permissions – the CMS must be able to assign permissions to content items or groups of content, not merely have site-wide roles into which users are placed
- **Centralized editing** – editors should be able to log on to a web-based interface and edit content from any desktop computer, preferably without the need to install any sort of software or browser plugins.
- **Site administration controls** – the CMS should allow for the separation between website administration and web server administration. This will help reduce the overlap in OIT and University Communications roles.
- **Media management** – editors should be able to put certain types of files on the server. Editors should not have access to the server file system, and should have no need for such access.
- **Centralized templates** – University Communications should be able to effect site-wide look and feel changes or effect changes to individual sections of the website.
- **LDAP Integration** – Our editors already have a username and password via LDAP. The CMS should be able to authenticate against it instead of requiring us to create new accounts and passwords. Furthermore, editors should log in as themselves, not as a generic account, so that content editing can be tracked.

Several CMS have been investigated and reviewed by University Communications. The following pages outline the general information about each, how they do or do not meet our needs, and other relevant but miscellaneous information.

concrete5

Server OS: UNIX/Linux

Server Software: Apache

Scripting Language: PHP

Database: MySQL

Cost: Free/Open Source

Website: <http://www.concrete5.org>

Our Criteria

- Separation of Content and Design – Design managed in templates, content stored in a database as “blocks,” pages constructed with multiple blocks.
- Page Versioning – each version of each block is stored in the database and can be retrieved at any time.
- Robust Permission Settings – Users can be placed in groups and groups can be assigned permissions to pages (read, write, delete, approve, versions, admin). Page permissions do not appear to be inherited from page to page – so reassigning an existing group of pages is laborious.
- Centralized editing – editors log in to a web interface. They then browse the site as though they were visitors but they gain access to controls on each page that allow editing.
- Site admin controls – administrators when logged in are able to view an administrative console
- Media management – users can be granted access to upload files to the web server
- Centralized Templates – the CMS supports templates for design but has no mechanism to determine which users have access to which templates.
- LDAP Integration – there is currently no LDAP authentication plugin. There’s some speculation that the OpenID plugin can be modified to support LDAP but nobody appears to have made this happen yet.

Review Notes

While concrete5 presented a clear and easy to use interface, its permission settings were somewhat lacking. Like many open-sourced projects the focus is on adding features and a clean interface, not enforcing business rules.

There are a myriad of add-ons that are purchasable through a marketplace accessible via the CMS console. Unfortunately there doesn’t appear to be any way to prevent users from accessing the marketplace, purchasing, and installing add-ons.

Impress CMS

Server OS: Platform Independent

Server Software: Apache

Scripting Language: PHP

Database: MySQL

Cost: Free/Open Source

Website: <http://www.impresscms.org/>

Our Criteria

- Separation of Content & Presentation – Content stored in a database, design in page templates.
- Page Versioning – Available
- Robust Permission Settings – Permissions can be set per object
- Centralized Editing – Editing is done through a web-based interface on the main web server
- Site admin controls – administrators have access to a back-end control panel
- Media management – media management functions are available
- Centralized template – template structure unknown
- LDAP Integration – not only available, but accomplished successfully with the MSU LDAP server

Review Notes

Impress CMS is not among the most recommended CMS because it is inherently not well-suited for a traditional hierarchical website like the MSU website. Instead of creating pages, editors create “articles” which are then published to the website, but they do not become individual and distinct pages. Because of its unsuitability it was not reviewed in as much detail as other CMS.

Joomla!

Server OS: Platform Independent

Server Software: Apache

Scripting Language: PHP

Database: MySQL

Cost: Free/Open Source

Website: <http://www.joomla.org>

Our Criteria

- Separation of content & presentation – the content is stored in a MySQL database, the presentation managed by templates
- Page Versioning – A version of each content item is stored in the database
- Robust Permission Settings – unknown
- Centralized Editing – editors log in to the website and edit content through a back-end
- Site admin controls – logged-in administrators can view a back-end console
- Media management – unknown
- Centralized Templates – a templating system is accessible through the administrative console.
- LDAP Integration – available

Review Notes

Like ImpressCMS, review of this CMS halted relatively early due to its being focused around publishing “articles,” not separate pages. Also, pages are dumped into a massive unorganized content bin and users are required search for the content they want to edit.

Joomla appears to be geared around creating an interactive community portal, not a traditional website. While it is an award winning piece of software, it simply does not provide the type of website we want to build.

MODx 1.0.2

Server OS: Platform Independent

Server Software: Apache

Scripting Language: PHP

Database: MySQL

Cost: Free/Open Source

Website: <http://modx.com/>

Our Criteria

- Separation of content & presentation – pages are stored in the database, presentation is managed through page templates
- Page versioning – Available through a free add-on. Not tested in this review
- Robust Permission Settings – Roles define a set of permissions available to an editor. Users can be placed in groups. Those groups can then be assigned a role per page. For example the group “Human Relations” can be assigned the role “Content Editor” on the Human Relations pages. These roles and groups are defined by the site administrator.
- Centralized Editing – Editors log in to the main website, browse the front-end website like normal and use CMS controls on the page to edit content.
- Site admin controls – logged-in administrators can view an administrative console.
- Media management – File management is available, file upload permissions unknown
- Centralized Templates – templates are files stored on the server. An administrator can create and upload templates and then add them to the CMS.
- LDAP Integration – While there is no included LDAP integration, we were able to build a plug-in for MODx to provide LDAP integration in a few hours that functions with the MSU LDAP server.

Review Notes

Of the open-source CMS, MODx is probably the most viable CMS for MSU. It is easy to use and has a clear, understandable hierarchical site structure. Its group and role based permissions work effectively and provide a great degree of flexibility.

It does lack some of the niceties that other CMS have, but in reviewing we found the plug-in architecture to be extremely useful, and the actual site source code to be relatively easy to edit and customize. For example, editors without the ability to edit the homepage couldn't see the site tree in the back-end view (since the root of the tree is unavailable to them – even though they have sufficient permissions to view that page). It took only a few hours to change that behavior and to make all pages visible while only allowing editors the ability to edit pages through the backend as defined in the page permission settings.

MODx 1.x is considered outdated, even though version 2.0 has not yet been released. Version 2.0 is in beta and is not yet viable as a production CMS. It exhibits several critical flaws, including the inability to properly create users.

Symphony CMS

Server OS: Platform Independent

Server Software: Apache

Scripting Language: PHP

Database: MySQL

Cost: Free/Open Source

Website: <http://symphony-cms.com/>

Our Criteria

- Separation of content & presentation – content is stored as generic content in the database, presentation is dictated by a series of XML, XSLT, and template files.
- Page Versioning – not available
- Robust Permission Settings – unknown
- Centralized Editing – Editing is done online through the website
- Site admin controls – editors are able to view a back-end console, but it has very limited functionality
- Media management – unknown
- Centralized templates – handled through a series of XML and XSLT files set up by the site administrator
- LDAP Integration – unknown

Review Notes

Symphony CMS was dismissed almost immediately as it dumps all of the content into a massive, unstructured content bin. Editors are obliged to search for content to edit. This makes managing a website of more than a few dozen pages an almost Sisyphean effort.

While it's nice to have a CMS support open standards like XML and XSLT, the unstructured nature of the CMS extends beyond the management console and into the site itself. The CMS has no concept of a "page" and the administrator is obliged to create the data types, the filters that convert the data into a page, and then the templates that create the look and feel for that content. In the end, Symphony is barely more than a GUI for the PHP XSLT parser.

TYPO3

Server OS: UNIX/Linux
Server Software: Apache
Scripting Language: PHP
Database: MySQL
Cost: Free/Open Source
Website: <http://typo3.com/>

Our Criteria

- Separation of content and presentation – The content is stored in the database, and presentation is stored separately, but it appears initially as though the content is linked to fields defined in the template used at the time of page creation. This effectively locks a page into a single design at the time of creation
- Page versioning – available but not tested
- Robust Permission Settings – unknown
- Centralized Editing – Editing is done on the website through a back-end interface
- Site admin controls – an administrative console is available to administrators on login
- Media management – unknown
- Centralized templates – templates exist as entities in the CMS database separate from content. TYPO3 parses HTML to build the structure, including the parts of the page into which editors can place content. This leads to the potential problems discussed above.
- LDAP Integration – available through a free add-on, but not tested in this review

Review Notes

TYPO3 is the most dense CMS reviewed in the entire process. It is *extremely* difficult to administer and not merely confusing, but outright cryptic at times. In about a week of testing the page and template structure is still completely foreign and we were never able to get a full grasp on how page creation works, much less how to build and maintain a full website.

It is at least encouraging that pages are stored in a tree structure but beyond that most aspects of management are still unknown.

It would be nice to dismiss TYPO3 as a non-contender simply because it seems like such a hassle but knowing that it's successfully being employed by other colleges (specifically New Jersey City University) means that it should be taken seriously. Nevertheless, if we are unable to push through the confusing nature of this CMS it will be impossible to use on a day-to-day basis.

Ektron CMS400 .NET

Server OS: Windows Server

Server Software: IIS 7

Scripting Language: VBScript .NET

Database: Microsoft SQL

Cost: one-time \$30,000 - \$70,000; recurring support fees

Website: <http://www.ektron.com/cms400.aspx>

Our Criteria

- Separation of content & presentation – pages are assembled by templates and “content blocks,” which can be either a page or a portion of a page. Content blocks are stored as entries in the database while templates are stored as VB Script files on the web server. Content blocks are organized into folders in the back-end system, but those folders don’t automatically determine how content is structured on the website.
- Page versioning – each version of each content block is saved and can be retrieved instantly
- Robust permission settings – administrators can place editors into groups and then assign detailed group permissions to folders in the back-end, granting rights to do a large number of individual tasks.
- Centralized editing – editors log in to the website and can either edit content through the back-end folder structure or edit content on pages via the front-end view.
- Site admin controls – Administrators have access to edit almost the entire site configuration via the back-end console.
- Media management – media files are placed in the back-end file structure as content blocks and are subject to the same permission settings as pages. The web page editor is able to handle most media types intelligently and let users either link to or embed content through a library browser.
- Centralized templates – templates for pages are stored as VB Script files on the main web server. They employ the IIS “master page” model. This allows a site administrator to create one main template that defines the essential structure and design of the web pages and then several sub-templates that amend and alter the master page. Editing templates requires filesystem access.
- LDAP Integration – the CMS offers LDAP integration for authentication, but we were unable to make it function properly in the review process. However, ensuring the LDAP authentication functions properly would be the responsibility of Ektron under their service contract.

Review Notes

Ektron’s CMS was far and away the most robust one we tested. Its ability to organize content into folders without affecting the site structure makes it extremely flexible for large websites. It’s also one of

the only CMS we tested that was able to apply permission settings to media uploads or to handle them in any reasonably intelligent way.

dotCMS

Server OS: Platform Independent

Server Software: dotCMS Java-based executable binary

Scripting Language: Java/J2EE

Database: modified Postgre SQL

Cost: Free community version, \$4,750/CPU + [\$6,000-\$21,000] annually for support

Website: <http://www.dotcms.org/>

Our Criteria

- Separation of content and presentation – content is stored in a database and parsed using templates, which are also stored in the same database. Content is organized into folders on the back-end which can have their own permissions.
- Page versioning – each version of each piece of content is stored and can be retrieved instantly
- Robust permission settings – Users can be arranged into groups which can then be assigned roles. The roles apply site-wide, but then those roles are granted permissions on pages. In practice, the “roles” in dotCMS are effectively groups of other groups. Permissions on individual pages don’t seem to go any deeper than “read/write.”
- Centralized editing – editors log in to the website and edit pages either through the back-end or by navigating the front-end in “preview mode.”
- Site admin controls – administrators are presented with administrative controls after logging in to the website.
- Media management – media files can be uploaded to content folders in the back-end. The files inherit the permissions of the containing folder. Images can be added to pages using a browser that traverses the back-end content folders.
- Centralized templates – templates are stored as entities in the back-end system. Content must be placed into “containers” which can then be included in templates.
- LDAP Integration – purportedly available via a paid plug-in. We did not purchase the plug-in to test this claim.

Review Notes

The major crutch for dotCMS is that it’s a relatively complicated CMS to use. Each page is an entity in the back-end system, but the actual content has to be placed on the page separately through “Edit Mode” where an editor views the front-end page and edits individual pieces of content there.

Also, while it’s tempting to say that this is a low cost or even free solution, the reality is that the actual cost is very vague. \$21,000/year buys us 10 help-desk tickets per month of support, but it’s very difficult to know if that will be enough or too little.

Also, because dotCMS uses its own application server, it's unknown whether the CMS will function alongside existing technologies like PHP and MySQL. We have several systems built using those technologies that must still exist on the future website regardless of CMS.