Content for One: Developing a Personal Content Management System

Doug L. Simpson
University of Oregon
1225 Kincaid St.
Eugene, OR 97403-1212
1-541-346-1736
dsimpson@uoregon.edu

ABSTRACT
You may have browsed or searched one of the large Content Management Systems that store and display useful information; many universities, departments, and library research desks have turned to these to manage both their long-term reference materials and daily-changing data. What if we'd like to manage our own content, but our staff consists of one overworked, time-challenged, long-term-memory-impaired worker: ourself?

My own information storage system was "heap-based:" several overflowing desktops of stacks of papers, journals, post-its, and scribbled memos. I needed something better, but didn't have anyone to give this project to and only a modest amount of time to devote to a new project myself.

I decided to look at a range of open-source Content Management Systems (CMS) and compare freely-available packaged systems with a completely handmade system based on PHP and MySQL -- all topics with which I had little or no experience.

Advantages to setting up a CMS like this (beyond organizing your own data) include gaining familiarity with concepts and systems that are likely to be increasingly important in the future. You may, if you wish, become a mentor to others with the same needs. The goal is to allow the wide-spread sharing of ideas and content without having to spend a lot of time fooling around with the technical details of presentation – a useful tool many people might be interested in.

I hope my experience and conclusions will help you decide if a CMS is the answer to your data problems, and which path might be the best route for you to take.

1. INTRODUCTION
Keeping up with an ever-changing list of tasks is always a challenge. The temptation to move from a list on paper to a neatly printed list on a computer is strong, but then the challenge is finding ways to access and update the list when you’re moving around on-campus – perhaps in an environment where you don’t have your regular editing tools available to you.

Even a simple one-page list of items can be difficult to keep up-to-date. If you modify the list on one local computer, for example, and forget to update the master stored elsewhere you’ll end with two “to-do” lists and no idea which is most current. Sharing a list of tasks with someone else almost guarantees extra phone calls or repeated messages between you to keep things straightened out.

You may have (or would like to have) more than just a “to-do” list to keep handy:

• A reference list of URLs
• Snippets of configuration details you need on a random basis
• Software serial numbers and version information
• One place to save new bits of information as you come across them
• A web log (blog) of current events
• A calendar list of upcoming appointments and events

You need more than a single document to keep different types of information sorted. This only multiplies the work of keeping them all current. A PDA or other portable device can help, but also limits the kind of records you can carry with you.

Setting up a CMS for personal use is much easier than creating one for a large organization. You can build the entire system from scratch, or use an existing CMS customized to your needs.
Enthusiasts have developed a very large number of open-source CMS’s, employing a number of different tools and techniques.

My original thought to create a CMS came from a poster presentation I saw at my first SIGUCCS Conference (Providence, RI in 2001). Tony Lovgren from Idaho State University displayed a very polished system his staff had created (from scratch) using PHP and MySQL to manage lab assistants, lab documentation, and student scheduling.

What time commitment and beginning technical background is required to create a basic, but useful CMS? I’ll contrast homebrew vs. packaged systems with an idea of how much time you might spend setting up each. I’ll also discuss some ways to help choose the right CMS for you.

2. CONTENT MANAGEMENT SYSTEMS

2.1 CMS Definition and Components
A Content Management System (CMS) can be defined as a database of information and a way to change and display that information, without spending a lot of time dealing with the technical details of presentation. Information is usually displayed in a web browser window.

Sharing information becomes far easier. Levels of access can be defined to control who has access to what data. The CMS can keep track of changes made by different users (an auditing trail) and won’t let two people change the same piece of data at the same time (record locking).

Although commercial CMS’s exist I won’t refer to them here as they usually are designed for very large-scale enterprises (with price-tags to match). Open-source CMS’s are powerful enough to compete both in quality and power with their commercial cousins.

There are three pieces to a CMS, in addition to the host computer where everything is stored: a database to organize the information; a script language to provide an interface to the information; and a web server to present the information as an HTML file for a web browser (see Figure 1).

Figure 1. How a CMS page is generated

2.2 The Host Computer
The hardware or operating system of the machine that will host the CMS isn’t an issue, as there are implementations for Windows, Mac OS X, Linux, Unix, and other platforms. For easy access to the data from anywhere else on the Internet, the host machine should have a static (fixed) IP. You can use your regular desktop machine, a spare computer, or even your laptop. You just need to have the computer turned on and connected to the Internet to access your data!

2.3 The Database
Using a database makes it possible to easily search the entire set of information for a particular item we need. MySQL, an open-source relational database, is often chosen for storing the information.

2.4 The Script Language
You view and change the CMS data through what looks like a standard HTML webpage. An underlying system of scripts controls how the CMS information is shown and altered. The scripts handle the details of accessing the database information. Things like web page links, formatting, and other underlying static information are also taken care of for you by the scripts. PHP is often chosen for the script language.

2.5 The Web Server
Under the direction of the scripts, the web server takes the information from your request and any data retrieved from the database and creates an HTML file. This can be viewed on any standard web browser connected to the Internet. Apache [1] is most commonly used for the web server.

3. MY BACKGROUND
I have an ancient B.S. degree in Computer Science. While I have knocked about in some programming languages, I have never done much with any of the languages usually seen (Perl, Python, Java and PHP) in the scripting component of the CMS’s I surveyed. I had little familiarity with MySQL (or any other database package of note) before starting this project.

A person with no previous computer experience whatsoever would profit from a local mentor to help him/her over the gaps in the documentation of all of the systems I’ve looked at thus far.

4. HOMEBREW VS. PACKAGED OPEN-SOURCE

4.1 Either Way Yields Results
In the last few years a wealth of books and on-line support has appeared for the novice system administrator. Five years ago, someone with little or no practical experience in setting up databases and web servers would have had little choice but to purchase a commercial CMS and pay for training and support. The emergence of books and information pitched for inexperienced users has changed all that. CMS For Dummies hasn’t appeared on bookshelves yet, but may not be far off!

4.2 Building It Yourself (Homebrew)
I began my whole experiment in working with a CMS by deciding I’d like to build everything myself (i.e., install all necessary software and write all the scripts that managed the data). I knew pre-built packages existed that combined scripts and some amount of presentation data, but hadn’t looked too closely at them. I was
sure the learning experience of beginning with an empty database and a blank page would outweigh any time saved by using a pre-built package.

I bought what looked like a good book [2] about creating a CMS from scratch using PHP and MySQL. I was taken step-by-step through the mechanics of installing first MySQL (the database), then PHP (to manage the data stored in MySQL), then the changes in setup and configuration so everything would work together as a whole.

There were small complications along the way when the assumed versions of either the software or my computer’s OS didn’t match what I actually had to work with. This usually resulted in visiting several websites in search of the best fit for the several choices before me. Sometimes the consensus would be to choose a stable version over an experimental (beta) version of a particular piece of software, sometimes the reverse.

This was my first experience with the extended support available through the Internet. Information I gleaned from forums and websites devoted to creation of CMS’s updated, expanded, and corrected the information I worked with in the book. ([3], [4], and [5] are examples of this type of resource.) I began to enjoy the sense of collaboration I felt with the anonymous posters of these tips and short tutorials.

Installing MySQL and PHP on my test machine so they would work properly together took between five to eight hours, undertaken in stretches of an hour or two at a time. I was then put to work learning the structure of PHP and slowly taught how to interact with the data I stored in MySQL. By the end of about 40 hours or so of study, I’d gone through two-thirds of the book and created a skeletal system wherein I could add, edit, display and delete jokes from a database entirely under my control.

The results were encouraging, but I was impatient to achieve something grander that would solve my original problem – the multiple streams of data and backlogged information that I wanted to organize.

### 4.3 Homebrew Conclusions

By the conclusion of my build-it-yourself construction phase I picked up a good deal of useful background knowledge about setting up a CMS and the rudiments of dealing with the low-level details of the greater system. To continue, I would have needed another book or books to build the kind of system most open-source packages offer.

I decided to suspend my homebrew solution and choose a packaged CMS instead. However, I didn’t feel my time thus far was wasted. To customize or add to a packaged CMS sometimes requires the kind of knowledge I picked up so far. (Especially if the package I choose uses the same scripting language I became familiar with – see “How To Pick a Package” below.)

### 4.4 Using a Packaged CMS

Packaged systems offer a bundled selection of scripts that are combined with the pre-selected database to eliminate most of the initial work of bringing up a CMS. The best of them save lots of setup time at the beginning but still can’t customize every script to your particular needs.

Doing the setup for you shields you from the details of the process, but may leave you more dependent on what’s built into the stock system or available as an add-on package. Taking time to pick the system that looks closest to your needs is important.

Different systems are offered at varying levels of assumed end-user knowledge. For the less-experienced user this becomes an important variable in the package selection process, below.

### 5. HOW TO PICK A PACKAGE

#### 5.1 Know What You Want

The first thing you need before looking at the large number of packaged systems available is a clear idea of the features you want. Some systems are centralized around a blog, forums, an online chat function, or other components you may not need. Try to find one that looks as much as possible like what you want.

On the other hand, being able to add extra components later may be something you find compelling. Most CMS’s offer ways to expand or add features later, so you aren’t limited to your initial setup forever.

The CMS scripts are most often written in PHP, Python, Perl, or Java. You may have a local expert in one of these who would be willing to work with you.

#### 5.2 Compare Packages by Feature

One way to sort through the many offerings is to visit one or more websites that have taken on the task of comparing systems [6]. Some even offer on-the-spot test runs with you as the administrator of a functioning system [7]. Look closely at the screenshots that show the admin screens; you’ll spend most of your time working with these controls.

#### 5.3 Examine Documentation Carefully

When you’ve identified one or more systems that might fit your needs, go through the documentation provided. Is it complete? Does it make assumptions about your technical knowledge that leave you in the dark? If there is a link to user forums or other online support, look to see if the postings there seem helpful and relevant to questions you may have about the package.

#### 5.4 Two Packages Selected

I ended up choosing Drupal [8] as my most promising candidate. The Drupal package uses the popular PHP & MySQL combination of scripting language and database. To see what I might be missing I also spent some time installing and using Mambo [9], which also uses PHP & MySQL. Both packages offered the features I wanted to start with and I was curious how similar they would be when set up in the same way.

### 6. INSTALLATION

Having already set up MySQL and PHP on my target machine, I could look at the installation and configuration screens of these packages with an eye to what might have been confusing to me if I hadn’t done it before.

Although every effort is made in the documentation of both packages to streamline the setup, there are times when it is
necessary to open a terminal window and type a command or two. Sometimes a missing folder needs to be created, or a pathname in an initialization file needs to be changed.

Here’s an early step in the installation of the master Drupal folder:

1. DOWNLOAD DRUPAL

You can obtain the latest Drupal release from http://drupal.org/.

The files are in .tar.gz format and can be extracted using most compression tools. On a typical Unix command line, use:

```
wget http://drupal.org/files/projects/drupal-x.x.x.tar.gz tar -zxvf drupal-x.x.x.tar.gz
```

This will create a new directory `drupal-x.x.x/` containing all Drupal files and directories. Move the contents of that directory into a directory within your web server's document root or your public HTML directory:

```
mv drupal-x.x.x/* drupal-x.x.x/.htaccess /var/www/html
```

Here’s that same step in the Mambo documentation:

The main distribution files for Mambo can be downloaded from mamboforge.net. The file will be a compressed archive so you will need a utility program, either locally on on [sic] your host server, that can "unzip" the file.

**Installing Locally**

Unzip the distribution file into a directory under your web server root. If you are using the Apache web server, this is typically /usr/local/apache/htdocs on a Windows and /usr/local/apache/htdocs on a POSIX system but it my [sic] vary. We are assuming [sic] you have a working web server and that you now [sic] where to put files to display on the web server.

If either of these looks incomprehensible, I hasten to add that these steps are expanded on and explained further in other areas on their respective websites. Both these sites have an active, enthusiastic group of users that enjoy helping with questions posted to their forums.

A local expert that can help you in person over occasional rough spots like this is an alternative to posting a question to the user forums and waiting for a response. Once the package is installed you won’t usually need to deal with issues like this.

### 6.1 First Impressions

After pushing through the installation it was a thrill (both times) to see the first web page load. When run as the administrator, both packages share most of the same maintenance tasks yet appear to regard them differently.

Drupal seems to expect a more thoughtful approach. One of the words with a weighted meaning in Drupal’s glossary is *Taxonomy*; to define vocabularies (sets of categories) which are used to classify content. *Nodes* and *blocks* and other special words must be learned and used to control the structure and placement of the data. Frustrated new users complain in the forums that they’ve been reading for half a day about concepts of data without being able to start data entry. More experienced Drupal users counsel that it’s all part of the plan and the slow start will prove worthwhile.

Mambo seems to reward the desire for immediate gratification. It’s easy to post messages and create pages with anything you want on them. More structure is available, but Mambo seems far less motivated than Drupal to encourage you to think about your data’s structure. Large colorful buttons are used (and required) for common actions.

Because Drupal asks you to define categories and types of data, it allows searching and sorting by the keywords you establish. Both packages provide full-text search functions. Both offer context-sensitive help.

### 7. RESULTS TO DATE

I find myself more interested in using Drupal than Mambo, but I realize that either can do the tasks I originally had in mind. Most of my time to date has been spent setting up these two competing systems for data storage. I still have a lot of data to enter. Please visit my webpage I’ll have up to show how things are going with this project if you’re curious [10].

By this fall’s SIGUCCS conference I should have ample experience with data entry and will know whether my heap-based desktop storage system is history – or not!

### 8. TO-DO

One of our next steps will be to create a CMS for our student workers in our public labs. They will use this to keep abreast of problems, to consult the (yet-to-be-written) FAQ, and to communicate with each other and the lab managers.

I want to explore the available add-on modules created by other Drupal and Mambo users. These are (they say) easily added to existing systems. If I identify a function that I need that hasn’t already been addressed I’d like to be able to build it myself.

There is one CMS package I haven’t looked closely at yet that I’d like to explore: Plone [11], a CMS built using Zope, which is in turn driven by Python. I’ve read glowing tributes to Python’s versatility and Oregon State University, our sister university to the north, has done some good work with this [12].

### 9. CONCLUSIONS

Content Management Systems bring many types of data together in one convenient place and makes it easy to access and update from many places. Although traditionally used for businesses, a CMS can also be set up for individual use.

Knowledge of the underlying programming isn’t required for setup, though access to on-line forums or some local talent may be useful to help you through occasional spotty documentation.

Although a packaged system may be quicker to set up, building everything from scratch is still possible. It also leaves no part of the system a mystery when setup is complete.

Allow sufficient time to choose the package that best matches your needs. A packaged open-source CMS can be setup and...
configured on a personal computer in as little as a few hours or perhaps a weekend, depending on your experience level.

Once everything is in place you can ignore the separate pieces of the system; day-to-day administration deals with the look and content of the data displayed. Let’s get organized!

10. REFERENCES