1

INTRODUCTION

Drupal is a powerful tool for managing web content, one that has become very popular in recent years. It is used in many organizations, including libraries, because of its flexibility and extensibility. But what makes it so useful in the library setting? This book will answer this question and walk you through creating a basic library website using freely available modules.

Before you decide on Drupal, however, you should begin with a more fundamental question: "Do I need a content management system (CMS)?" Because you're reading this book, your answer to this question is likely "yes." To help you get to an unqualified "yes," we will begin by looking at a series of decision points—whether you need a content management system in the first place and, if you do, whether it should be an open source or proprietary system. We will then provide some guidelines for making a well-reasoned decision and, finally, spend some time describing Drupal and the way it is structured.

In the coming chapters, we will examine the ways you can get Drupal for your site, plan a CMS installation, work with your library staff and technical support groups, and then delve into building a Drupal site of your own.

CONTENT MANAGEMENT SYSTEMS VERSUS FLAT FILES

If your website is more than a few years old, you may still be living in the world of plain HTML files stored on a web server. If you are, and you have more than a few dozen pages to manage, you are probably looking for the

efficiency a content management system (CMS) can bring you. These efficiencies include:

- the creation of a small number of templates so that all of your webpages look the same and can be changed quickly as the need arises.
- the separation of layout from content so that anyone in your organization may be given authority to write or maintain content for your website without the need to provide training on HTML, CSS, or any programming tools.
- the ability to reuse text multiple times within your site. For example, blog posts about circulation could appear in your news section as well as on your circulation page. Hours for libraries or service desks can appear both centrally and within the site they refer to. Staff names and contact information can be displayed everywhere they are needed, in the correct context.
- consistency across the organization for how web data is managed and stored.
- work flows that allow for content review and approval, where appropriate, before publication.
- the ability to borrow from other people using the same CMS by downloading and installing modules—small applications—to enhance your site.

When we began our redesign at the University of Michigan Library, we were faced with a monster of a website. It had more than 50,000 pages. About 10,000 were static files, coded in HTML and organized into several dozen subsites. The remaining 40,000 pages were generated dynamically, using Perl or PHP and a database back end, through a number of different legacy applications. There was no standardization between the HTML parts of the site and the dynamically generated portions—not even within the static pages or dynamic pages. This was the result of having different developers and authors building their own pieces of the site, independently, over the course of almost two decades. The maintenance effort in keeping the static part of the site updated was a large one, even though it was distributed across many individuals around the library. The not insignificant effort needed to keep the dozen or so different applications up-to-date and functioning was concentrated in the Web Systems group.

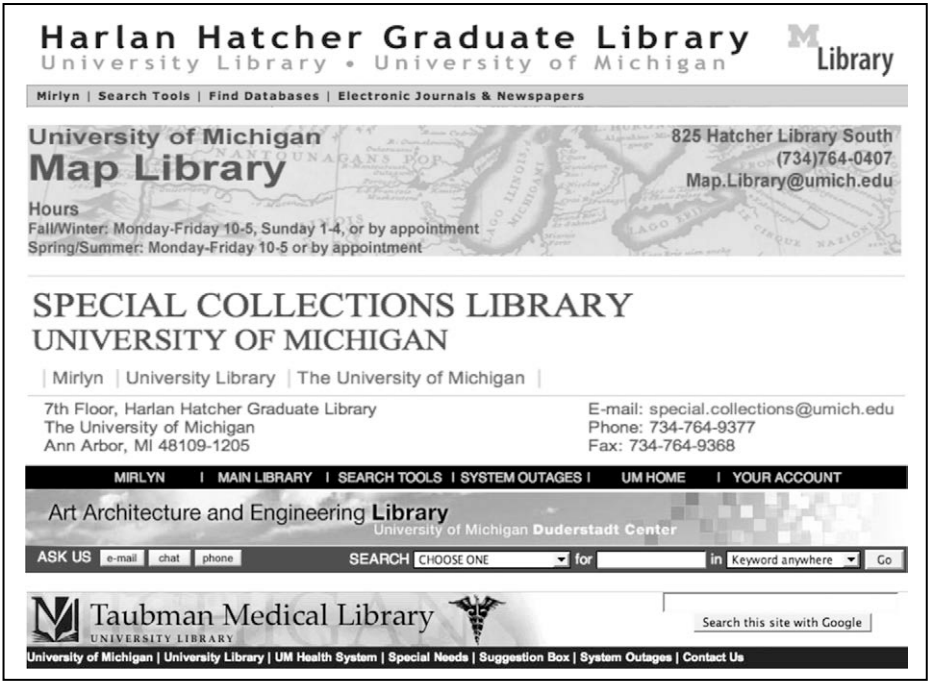
From the user perspective, the library's site had dozens of graphic identities for different parts of the library, many of which bore little resemblance to the homepage (see Figure 1.1). Constituent libraries, service points, departments, and information pages had radically different designs. There was almost no consistent navigation across the site; many pages did not link to the main

library page. Those that did used different logos or graphics, and the links were placed on different parts of the webpage. This complicated user interface made it very difficult for a site visitor to move from one library to another or from a library to a particular resource. The site was an exercise in frustration for our users and our staff.

Furthermore, there was no consistent application of site search—some search boxes covered the whole library website, while others included only that particular subsite's content. We arrived at this state of affairs much as you may have: by allowing the web to grow organically over years without finding the time or energy to bring it together.

When we were done with our redesign, we had pruned a tremendous amount of stale content, reducing the number of pages considerably, to about 10,000. This total represents a large number of brief pages describing particular resources in detail, along with information about the library and its services. We put in place a navigation across the entire site and had given more than 100 library staff members (out of a total staff of nearly 500) the authority to write, edit, and delete pages. Changes we now make to standard elements are instantly rolled out to the entire site through templates and common CSS files.

▶ **Figure 1.1: Sample Screenshots of Our Heterogeneous Site**



▶ OPEN SOURCE VERSUS CLOSED

Assuming that you have decided to go the CMS route, there is another fork in the road to navigate. Like most software systems, content management systems come in two flavors: closed (proprietary) and open source. In a proprietary or closed system, the vendor maintains control over the application and the interface, providing a defined, sometimes limited, suite of functions. These applications can range from "black box" to well-documented systems. In a "black box" system, the computer code is not accessible to anyone but the system's developer. Customers cannot look inside to see how the software works. In contrast, well-documented systems have detailed explanations of what is going on within the system, even if customers do not have the ability to change the way the application works. These changes are generally made through requests to the vendor for feature enhancements. At the same time, you usually have control over the display templates and styling.

Open systems differ in that you, the library, have access to the source code so you can make changes to meet your particular needs as well as add new functionality that is not offered by the software company. Plus, you can benefit from the many developers working in the same programming environment to help quash bugs and add features.

▶ SELECTING A CONTENT MANAGEMENT SYSTEM

What kinds of questions should you ask yourself when you're selecting a CMS? When we established a task force to evaluate the open source CMS landscape in April 2008, we used a fairly broad set of criteria to review the product landscape and settle on a short list of products that might work well for us:

- Is support available? If so:
  - How is it delivered?
  - What is the cost?
- Will the product be around for a while? As Yogi Berra said, "It's tough to make predictions, especially about the future." But you can make an educated guess.
- How modular is the tool?
  - Do you need to turn on every available function, or can you leave those that aren't of immediate use to your users inactive?
  - Can you add other modules available now or that might be available in the future?
- How active is the developer community?

  - How many other organizations are developing for this tool?
  - How many of those developments are available for others to use?
  - How many developers are working with contributed modules?
- Is there a formal release mechanism for new versions of the software? Open source does not mean disorganized—well-done open source efforts exert careful version control and planning. The popular Firefox web browser, for example, is open source.
- How much documentation is there, and how good is it?
  - Do developers provide instructions for using their code? Can you understand it?
  - Is there documentation for the version of the software you would be using?
- How easy is the tool to develop (for developers) and use (for content authors)? Is this a tool only a geek could use, or are the user functions understandable to you?
- How scalable is the tool?
  - Can it handle the amount of content you expect to put in it?
  - Can it serve the anticipated number of online users?
- Can you get your data out of it? In other words, what's your exit strategy if times change, new products emerge, or you simply realize you made the wrong choice for some reason? How can you get your content, and in what formats?

When we did this review, we assigned a score from 0 to 2 for each category (0 meant the application was unacceptable in that dimension, and 2 meant that it exceeded our expectations in that dimension). We looked at the average score for each tool, winnowing it down to a short list of three (see Table 1.1). From there developers and project managers reviewed the tools and sought feedback from the respective user and developer communities. This analysis led us to select Drupal.

▶ DRUPAL

What is Drupal? Drupal is an open source content management system. Drupal was designed to be the starting point for the functionality you want your site to have. It provides the framework, and developers across the Internet have provided the functionality. In many ways, it is a toolbox with a handful of common tools that will get the basic job done; it comes with screwdrivers, hammers, and saws already in it. To make it truly useful for you, though, you need to look at the problem you are trying to solve and acquire the specific tools you need to do those tasks well.

▶ **Table 1.1: Feature and Functionality Review Matrix for Selecting a CMS**

Dimension	SilverStripe	Alfresco	MODx	Drupal	Plone	Joomla!	Daisy	WebGUI	TYPO3
Is support available?	1.5	2.0	0.5	0.5	1.5	0.5	1.5	1.5	1.0
Will the product be around for a while?	1.0	1.5	1.5	2.0	2.0	2.0	1.5	1.0	1.5
How modular is the tool?	1.0	1.0	1.5	2.0	2.0	2.0	1.5	1.0	1.0
How active is the developer community?	1.5	1.5	2.0	2.0	2.0	1.0	0	1.0	1.0
Is there a formal release mechanism?	1.5	1.0	1.5	2.0	2.0	1.5	0	1.0	1.5
How much documentation is there?	1.0	1.0	0.5	2.0	2.0	2.0	1.0	0.5	1.5
How easy is the tool to develop and use?	1.5	1.0	1.0	1.0	1.0	1.0	1.5	1.0	1.0
How scalable is the tool?	1.0	2.0	1.0	2.0	2.0	1.0	1.5	1.0	1.0
Can you get your data out of it?	0.5	1.0	0.5	0.5	1.0	0.5	1.0	0.5	1.0
Average score	1.17	1.33	1.11	1.56	1.72	1.28	1.06	0.94	1.17

Drupal is sufficiently flexible that you can use it to create work flows and organize content that reflect your organization rather than forcing your organization into a one-size-fits-all model. If you have a very flat organization with little hierarchy, Drupal will allow you to set up a site and establish authoring and editing roles that match your culture. Likewise, if your organization is more vertically structured with clear lines of communication and management, Drupal can enable your website to reflect that kind of organization in page authoring, editing, and presentation.

The Power of the Library Developer Community

In our decision process, Drupal won out over the competition because of two key factors: the strength of the library developer community and its relatively wide

adoption in the library world. If you are going to put your content in a single basket, it is wise to use a basket that many of your colleagues are also using.

There is a self-maintained list of libraries using Drupal on the Drupal Groups website (http://groups.drupal.org/libraries). At the time of this writing, more than 180 libraries are listed, ranging from special to public to academic. There is certain to be a library similar to yours, facing similar challenges and needs, that has already accomplished something like what you want to do.

A similar list of library-related Drupal modules (more about that later) is on the "Drupalib" community site (http://drupalib.interoperating.info/library_modules). Modules are available for a range of typical library functions—from creating links to an OpenURL resolver or EZproxy to managing bibliographies to providing a frequently asked question (FAQ) interface.

Drupal Concepts

As with any system, Drupal has its own vocabulary unique to its specific components. This section will help you understand Drupal's fundamental building blocks as we go forward.

How Drupal Manages Content

Drupal's basic building block for the information used within and displayed on your site is the "content type." A content type is a set of fields that are created by a Drupal administrator to handle pieces of similar information in a uniform way. For example, a content type might be simple, such as an image (a URL to the file and a title) or a blog post (a title, an author, text, and tags) or as complex as a webpage with title, images, an author, text, and one or more terms from your website subject thesaurus and free-text keywords. Other examples might include descriptions of service points in your library: a location (building, floor, room number, etc.), hours of operation, description of the service, and contact information (a person, an e-mail address, a phone number, etc.). When you define the kind of information each content type provides, you are setting up a database table in which Drupal stores the information. (See the Quick Tip to help you decide what should be a content type.) Tools within Drupal give you access to

Quick Tip

While each content type may have its own presentation design, you should avoid thinking of content types as having a specific layout. If you want to apply a particular layout to a content type, handle that in the theme section (see How Drupal Handles Site Design, pp. 9–10). Although it is tempting to create content types for different presentations of similar material, this can cause a maintenance challenge down the road when you find yourself needing to change a theme. You would need to change multiple theme files. It is generally more efficient to keep similar content in the same content type to improve efficiency of maintenance.

individual fields for each type; you can display each type as a freestanding webpage as well.

How Drupal Displays Content

A content type defines a kind of information. A "node" is what the user sees. At its simplest, a node is a database entry—a page in the site, something with a unique and persistent URL. For example, a page showing a staff member's name, title, contact information, and other pertinent details would be a node, as would be a page describing how to access a particular online database. At the more complex end, it is a database entry but one that is not necessarily intended to have a page of its own. It might be a set of contact information for a particular service or staff member—intended to be searched for and displayed within the context of another, larger, framework.

How Drupal Functions

The basic building block of functionality is the "module." You can think of a module as a custom application (as you might find on your iPad or Android phone), a piece of code designed to do something specific in your Drupal site. There are more than 9,400 modules available from the open source community; 3,100 are ready for Drupal 7. Some modules come with Drupal and are required. Nodes, for example, the building block of Drupal content, are controlled by a core module, as are users (each person who has a log-in on the site) and blocks (discussed in more detail later, these are regions on a page into which content can be placed). Modules are the tools in your toolbox. When you install Drupal out of the box, you get a set of core functions that includes user account management, search, taxonomy management, navigation, and so forth. You can add additional tools to meet specific needs.

Other common modules you might use, but that are not part of Drupal's core, include these:

- **Calendars**—to display events happening at your library
- **FCKeditor**—to create a Microsoft Word–like toolbar for formatting text as it's typed when you are writing content for your site
- **Taxonomy Manager**—to manage and apply a thesaurus of terms to label content in your site
- **Pathauto**—to create alternate, human-readable URLs for your pages so that you can use "http://library.org/service-hours" rather than Drupal's default "http://library.org/node/12345"
- **Organic Groups**—to control membership and access rights to specific parts of your site for authoring and display

We'll talk more about these modules in Chapter 5, "Implementation."

How Drupal Works with User Roles

All visitors to your website have a user role. Most of them will be "anonymous users"—people who do not log in and have absolutely no authority to do anything on the site other than read the content, perform searches, and so forth. Authenticated users are those whom you permit to log in to your site to access some specific functionality (reserve a book, add a comment to a blog post, etc.). Then there are users whose roles you define—for example, you can create a user role for those library staff who have editing responsibilities so that they can edit any—or specified pieces—of the site's content. You can create as many user roles with as much—or as little—hierarchy as makes sense for your organization. Your library's patrons might even be given more advanced roles (moderator of a bulletin board, for example), if you give them the ability to log in to perform some function.

You control the level of access for particular users through Drupal's user administrator tools. Here, you create user roles (e.g., page authors, editors, reviewers) and assign each type a specific set of permissions for the kinds of content they can edit. One kind of page author, for example, might be able to write and update content for your site but not publish it. Another kind might be able to write and publish event summaries but not create content elsewhere. A content reviewer might also be able to approve content written by another staff member and make it public. An editor could have these capabilities but also be able to delete anything on the site. Access controls in Drupal are incredibly granular and can be assigned for each content type in the site.

An additional module, Organic Groups, allows you to further control access by section of your site. So, for example, a reference librarian might be a page author in one section of the website but not be able to add to the content in the interlibrary loan section.

How Drupal Handles Site Design

Drupal uses "themes" to manage and define your site's look and feel—the user interface to the site. You can think of themes as skins—with a quick change of templates you can change the way your site looks, from color schemes to layout. Themes come in three flavors: default, contributed, and customized. Drupal comes with four default themes (these will be discussed in more detail in "Install Themes" in Chapter 5, pp. 52–55). Default themes are ready to use and require no customization to work. On the other hand, they are bland and clearly identify a site as a Drupal site.

The second flavor of theme is contributed themes. Drupal developers have created thousands of themes for their own sites. Some of these themes are based on core themes or on previously contributed themes. Others are

written from the ground up. As of this writing, the Drupal website has about 250 contributed themes for Drupal 7. These are available for you to download and install and—most important—customize. If you like the layout of one, but want to use your library's color scheme, just change it.

The third flavor of theme is customized. If you are comfortable with Cascading Style Sheets (CSS) and HTML, you can tweak the colors and basic layouts of most themes. To develop your own, or heavily customize a contributed theme, you need to know some PHP, because the theme files interact with Drupal modules.

If you are interested in developing your own theme, an excellent starting point is the Zen theme (see http://drupal.org/project/zen). It is standards-compliant and very flexible. The Flexible theme (http://drupal.org/project/flexible) is designed with accessibility for users with varying physical or visual abilities. There are other starter themes, as well. See http://drupal.org/node/323993 for a current list. With some CSS and PHP knowledge you can customize a theme to look like anything you want.

At a more granular level, Drupal displays content in "blocks" on the webpage. Blocks are regions of the page, defined by the site administrator, into which various elements go. Site-wide templates control some blocks—your library's logo, name, and basic navigation presumably appear on all pages in the same place. Content-specific templates control other blocks—the display of textual information as compared to the display of graphical information, for example. With Drupal's blocks tool, you define a set of content areas (such as main navigation, secondary navigation, content, contact information, etc.) and place specific modules within each block. Each module has its own content template, used to control the way it looks. Blocks depend on the theme in which you are working, so changes you make to blocks in one theme do not automatically carry over to other themes.

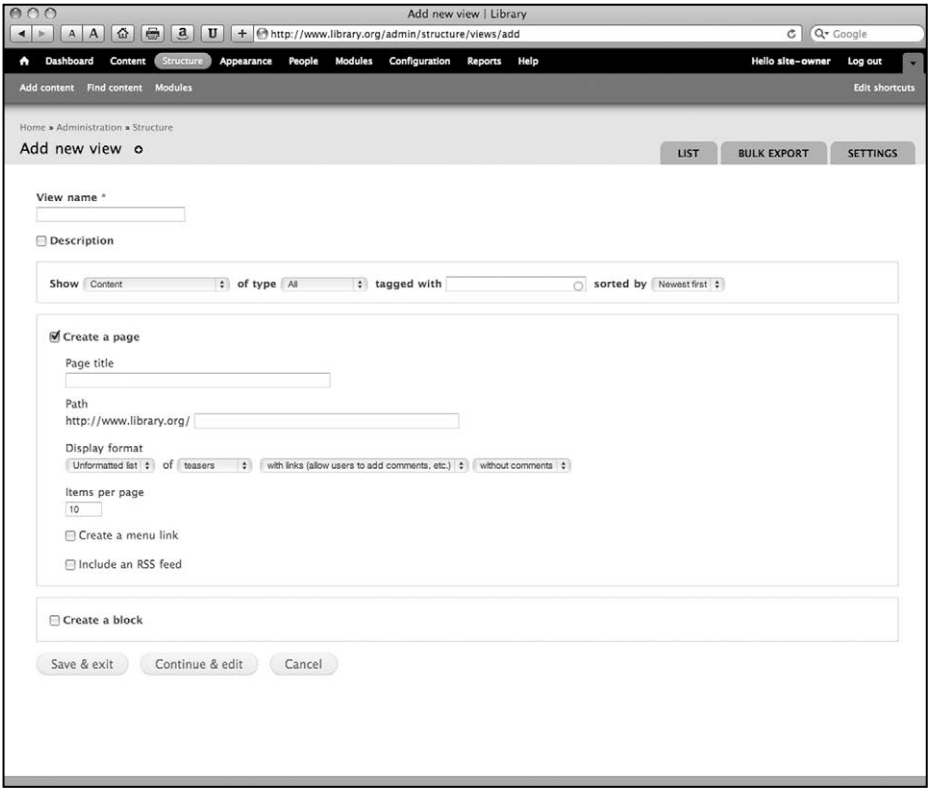
How Drupal Lets You Repurpose Content

Two related sets of modules allow you to reuse Drupal content in flexible ways. The first is "Panels," which lets you arrange Drupal content outside of its original structure. They are very powerful tools. They allow you to display individual pieces of information from one kind of content within other pages. For example, an implementation of panels on the front page of your news section might display the headline and first sentence for the most recently added ten stories as a table of contents. The title would link to the full news article. This "new news" panel can be replicated elsewhere on the site. The Panels module allows you to build a layout for this front page without having to do any significant custom coding, simply by applying

the panel to the content and setting the rules through the administrative interface.

The second module helpful for reusing content is "Views." Views are similar to Panels in that they allow you to pull out specific fields from across many nodes of the same type. However, where Panels gives you the ability to display content out of its original context, the Views module is more powerful: it is a query-building tool that gives you more flexibility in the outputs of your query. (In fact, the results of a View can be displayed within a Panel along with other content.) The Views module comes with a range of preset output formats: an HTML table, a grid, a photo gallery, an HTML list (bulleted or numbered), slide shows (for images), and so forth. You might use Views to present your site's taxonomy in alphabetical order. Why would you do this? Well, the built-in taxonomy tool is great for applying subject terms to your content but is less useful for displaying an index (listing the taxonomy terms), because it displays the list of terms in the order in which the terms were entered. The Views module lets you display a list like this in alphabetical order (see Figure 1.2).

▶ **Figure 1.2: Views Configuration Page**



▶ A NOTE ABOUT DRUPAL VERSIONS

An important note: This book focuses on Drupal 7, which was released in January 2011. The new core application offers a range of administrative interface improvements and many architectural changes over Drupal 6. It is likely that Drupal 6 will continue to be used by many sites for some time because modules written for Drupal 6 will need to be updated to work properly with Drupal 7. This work is ongoing in the community, but if experience with the Drupal 5 to Drupal 6 migration is any guide, it will take some time, on the order of 6–18 months, for the majority of Drupal modules to be rereleased for the new version.