11 12 13 **14** 15 16 17 18 19 20

THE TECH SET

Ellyssa Kroski, Series Editor

#14

Drupal in Libraries

Kenneth J. Varnum

LIBRARY AND INFORMATION TECHNOLOGY ASSOCIATION

lita

11 12 13 **14** 15 16 17 18 19 20

THE TECH SET

Ellyssa Kroski, Series Editor

#14

Drupal in Libraries

Kenneth J. Varnum

lita

ALA TechSource
An imprint of the American Library Association
Chicago 2012

© 2012 by Kenneth J. Varnum. Any claim of copyright is subject to applicable limitations and exceptions, such as rights of fair use and library copying pursuant to Sections 107 and 108 of the U.S. Copyright Act. No copyright is claimed for content in the public domain, such as works of the U.S. government.

Library of Congress Cataloging-in-Publication Data
Varnum, Kenneth J., 1967–
Drupal in libraries / Kenneth J. Varnum.
p. cm. — (The tech set; #14)
Includes bibliographical references and index.
ISBN 978-1-55570-778-1 (alk. paper)
1. Library Web sites—Design. 2. Drupal (Computer file) 3. Web site development. I. Title.

Z674.75.W67V37 2012
006.7—dc23
2012007201

Contents

Foreword *by Ellyssa Kroski*

Preface

Acknowledgments

1. **Introduction**

2. **Types of Solutions Available**

3. **Planning**

4. **Social Mechanics**

5. **Implementation**

6. **Marketing**

7. **Best Practices**

8. **Metrics**

9. **Developing Trends**

Recommended Reading

Index

About the Author

Don't miss this book's companion website!

Turn the page for details.

THE TECH SET® Volumes 11–20 is more than just the book you're holding!

These 10 titles, along with the 10 titles that preceded them, in THE TECH SET® series feature three components:

1. This book
2. Companion web content that provides more details on the topic and keeps you current
3. Author podcasts that will extend your knowledge and give you insight into the author's experience

The companion webpages and podcasts can be found at:

www.alatechsource.org/techset/

On the website, you'll go far beyond the printed pages you're holding and:

- Access author updates that are packed with new advice and recommended resources
- Use the website comments section to interact, ask questions, and share advice with the authors and your LIS peers
- Hear these pros in screencasts, podcasts, and other videos providing great instruction on getting the most out of the latest library technologies

For more information on THE TECH SET® series and the individual titles, visit www.neal-schuman.com/techset-11-to-20.

FOREWORD

Drupal is a powerful, free, open source content management system that enables organizations to create extensive, flexible websites incorporating social and 2.0 functionality. *Drupal in Libraries* is a comprehensive guide to harnessing the power of this increasingly popular technology to create community-oriented websites. Drupal guru Ken Varnum deftly conveys the entire process of setting up a library website as well as provides tips and best practices for creating and customizing themes, incorporating library resources into Drupal, and enhancing websites with social media features. Everything from creating and configuring Views and Panels, creating new content types, and designing your own modules can be found in this all-in-one manual.

The ten new TECH SET volumes are designed to be even more cutting-edge than the original ten. After the first ten were published and we received such positive feedback from librarians who were using the books to implement technology in their libraries as well as train their staff, it seemed that there would be a need for another TECH SET. And I wanted this next set of books to be even more forward-looking and tackle today's hottest technologies, trends, and practices to help libraries stay on the forefront of technology innovation. Librarians have ceased sitting on the sidelines and have become technology leaders in their own right. This series was created to offer guidance and inspiration to all those aspiring to be library technology leaders themselves.

I originally envisioned a series of books that would offer accessible, practical information that would teach librarians not only how to use new technologies as individuals but also how to plan and implement particular types of library services using them. And when THE TECH SET won the ALA's Greenwood Publishing Group Award for the Best Book in Library Literature, it seemed that we had achieved our goal of becoming the go-to resource for libraries wanting hands-on

technology primers. For these new ten books, I thought it was important to incorporate reader feedback by adding two new chapters to each volume that would better facilitate learning how to put these new technologies into practice in libraries. The new chapter called "Social Mechanics" discusses strategies for gaining buy-in and support from organizational stakeholders, and the additional "Developing Trends" chapter looks ahead to future directions of these technologies. These new chapters round out the books that discuss the entire life cycle of these tech initiatives, including everything from what it takes to plan, strategize, implement, market, and measure the success of these projects.

While each book covers the A–Zs of the technology being discussed, the hands-on "Implementation" chapters, chock-full of detailed project instructions, account for the largest portions of the books. These chapters start off with a basic "recipe" for how to effectively use the technology in a library and then build on that foundation to offer more and more advanced project ideas. Because these books are designed to appeal to readers of all levels of expertise, both the novice and advanced technologist will find something useful in these chapters, as the proposed projects and initiatives run the gamut from the basic how to create a Foursquare campaign for your library to how to build an iPhone application. Similarly, the new Drupal webmaster will benefit from the instructions for how to configure a basic library website, while the advanced web services librarian may be interested in the instructions for powering a dynamic library website in the cloud using Amazon's EC2 service.

Ken Varnum has been writing and speaking about web technologies for many years and has been keeping the University of Michigan Library on the cutting-edge as the Web Systems Manager. When I realized that the series needed a book on Drupal, I knew that Ken was the person to write it. He brings his extensive knowledge and expertise to this outstanding book on *Drupal in Libraries*, in which he guides readers through all aspects of creating a robust website using this powerful open source application. And he does so in a

clear and articulate manner that is exceedingly accessible for the reader.

Ellyssa Kroski
Manager of Information Systems
New York Law Institute
http://www.ellyssakroski.com/
http://oedb.org/blogs/ilibrarian/
ellyssakroski@yahoo.com

Ellyssa Kroski is the Manager of Information Systems at the New York Law Institute as well as a writer, educator, and international conference speaker. In 2011, she won the ALA's Greenwood Publishing Group Award for the Best Book in Library Literature for THE TECH SET, the ten-book technology series that she created and edited. She's also the author of *Web 2.0 for Librarians and Information Professionals*, a well-reviewed book on web technologies and libraries. She speaks at several conferences a year, mainly about new tech trends, digital strategy, and libraries. She is an adjunct faculty member at Pratt Institute and blogs at *iLibrarian*.

PREFACE

You've likely picked up *Drupal in Libraries* because you've heard something about Drupal and want to know if it is a good fit for your organization. Or perhaps you've been told to "fix the website" and you're exploring various technologies to make that request a reality.

I wrote *Drupal in Libraries* because there was no similar book available when I was managing a major website redesign in Drupal. Much of what I include in this book is derived from the processes and lessons I learned as I investigated content management software, managed the development team that customized our Drupal installation, and worked closely with the project management team to redesign our library's website. The lessons I learned—some the hard way—will save you time and effort. The days when an organization could essentially put up some HTML pages and call it a done deal are long past (if they ever truly existed). A multitude of tools and technologies exist to help libraries organize, publish, and maintain their content. One of the rising stars in this arena is the Drupal open source content management framework.

So what is an "open source content management framework"? Healthy open source software tools create a community from their user base. This community helps augment and develop the tool itself, creating new functionality that is relevant to its users. Drupal and libraries have forged a particularly symbiotic relationship, with libraries large and small creating—and, most importantly, sharing—chunks of code to accomplish specific tasks.

• Organization and Audience

Drupal in Libraries will guide readers step by step through the decisions and tasks needed to develop and launch a Drupal-powered website. Chapter 1 discusses open source and proprietary software and helps you understand the pros and cons of an open source

approach. It ends with a discussion of Drupal concepts to help you understand the rest of the book. Chapter 2, "Types of Solutions Available," describes the range of solutions available—from doing it yourself to outsourcing the entire development process, with various gradations in between. In Chapter 3, "Planning," you will go through a planning process that will guide you to your initial functional specifications for your site. "Social Mechanics," Chapter 4, gives you hints and suggestions to work with your IT department, colleagues, and management as you develop your technical specifications.

The bulk of the how-to in *Drupal in Libraries* is in Chapter 5, "Implementation." This chapter guides you through installing Drupal, adding modules, and developing your own themes (page layouts) and describes the modules created by other librarians for use on their sites. Chapters 6–8 discuss marketing your site, best practices for project management and development, and measuring the success and impact of the site once it launches. The book wraps up with a chapter on emerging trends and tools in Drupal and a look at the changes that might be expected when the next versions of Drupal (versions 8 and 9) are released in the years to come.

Drupal in Libraries is for you, the information professional. It assumes that you have some knowledge of website design and architecture, but it does not require you to be a specialist. You may have some programming skills, but they are not required for you to make use of this book. Being a framework, not a "solution," means that Drupal is powerful enough to accomplish almost any web content management task yet is focused on one thing—organizing web content—effectively. It is a tool set to manage web content and allow you, the web project manager, to rapidly customize the functionality you need.

Acknowledgments

I am indebted to the many people who have taught me much about Drupal and project management. Scott Ash, Albert Bertram, and Tyler Frankenstein are excellent developers who have taught me much of what I know, and I'm grateful to them for their many lessons and pointers. I learned a great deal from the other members of our Drupal project team at the University of Michigan Library: Karen Reiman-Sendi, Mike Creech, and Liene Karels. They have openly shared their advice, opinions, and knowledge over the past four years and this book is richer for it.

I am particularly grateful to Karen, who graciously read and commented on the complete manuscript, showing me places where my assumptions and the reader's diverged. Lucinda Varnum, my perennial proofreader, deserves a great deal of gratitude and credit for improving my writing, both of this book and in general. Ellyssa Kroski, my editor, has suggested innumerable ways to improve the narrative. Any mistakes in the finished book are mine alone, despite the best efforts of my colleagues and family to eliminate them.

Finally, I dedicate this book to my wife, Nieka Apell, who has so generously put up with the great time sink that is a book. Thank you for your love and support, in this and everything.

1

INTRODUCTION

- **Content Management Systems versus Flat Files**
- **Open Source versus Closed**
- **Selecting a Content Management System**
- **Drupal**
- **A Note about Drupal Versions**

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.

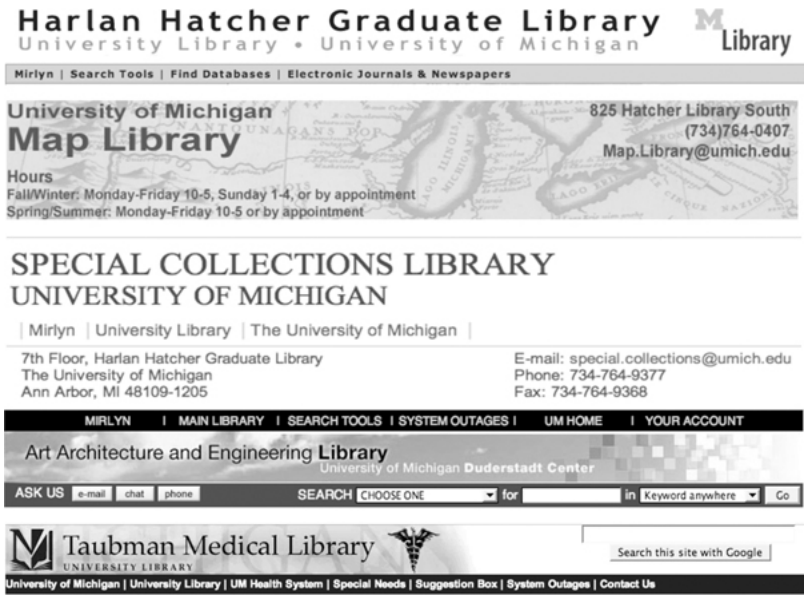

Figure 1.1: Sample Screenshots of Our Heterogeneous Site

• 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	WebGU
Is support available?	1.5	2.0	0.5	0.5	1.5	0.5	1.5	1.5
Will the product be around for a while?	1.0	1.5	1.5	2.0	2.0	2.0	1.5	1.0
How modular is the tool?	1.0	1.0	1.5	2.0	2.0	2.0	1.5	1.0
How active is the developer community?	1.5	1.5	2.0	2.0	2.0	1.0	0	1.0
Is there a formal release mechanism?	1.5	1.0	1.5	2.0	2.0	1.5	0	1.0
How much documentation is there?	1.0	1.0	0.5	2.0	2.0	2.0	1.0	0.5
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
How scalable is the tool?	1.0	2.0	1.0	2.0	2.0	1.0	1.5	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
Average score	1.17	1.33	1.11	1.56	1.72	1.28	1.06	0.94

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

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.

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).

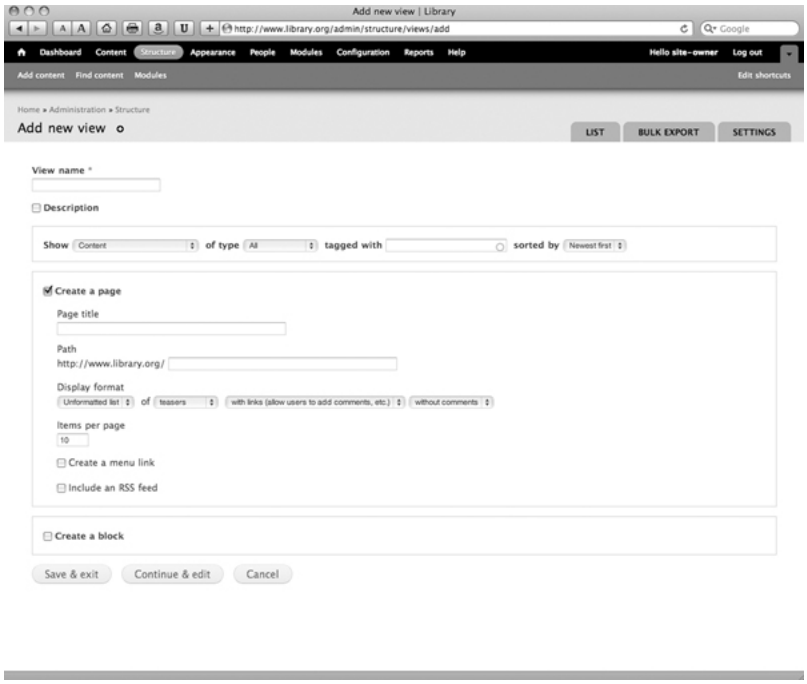

Figure 1.2: Views Configuration Page

• 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.

2

Types of Solutions Available

- **Choose Your Host**
- **Pick a Developer**

As with any software development project, you have a number of options when it comes to building the infrastructure and developing the application. Important choices you will make include selecting where to host the Drupal environment and how you will manage the development process. You may wish to treat these as separate items or to consider them a package deal. There are pros and cons, as summarized in Table 2.1. We'll discuss them at the most granular level, as if you were considering them as separate decisions, even though they probably are not for most libraries.

• Choose Your Host

The first set of issues surrounds where your website should be hosted. For some libraries, this may already be answered—you have your own server infrastructure or are required to use data centers provided by your parent organization. For others, though, this can be an open question. The answer depends on a wide range of factors. Here are some factors to consider:

- Does your library already maintain servers for your current website, integrated circulation and catalog system, and other services? If so, then adding a new piece of hardware—or repurposing an existing server—may not add that much extra work. If you rely on others to maintain this "big iron," then an outsourced solution might be appropriate for your Drupal server as well.
- Do you have the capital resources available to update server and backup hardware every few years? While the total cost of hosting your own server may not be significantly different from the total cost of outsourcing the hosting, the cash flow is different. A periodic contract

payment to a service provider may be more or less attractive from a budget perspective than a portion of a salary and a periodic capital expense to keep things in-house.

Table 2.1: Hosting and Developing Options		
	Outsourced	Developed Internally
Hardware	Pros: • Someone else is responsible for the hardware, networking, and operating system. • You can purchase shared access to more powerful servers than you might be able to afford on your own. • Service-level agreements (SLAs) give you an “or else” if something goes wrong. Cons: • You do not have physical control over	Pros: • You have complete control over your environment and can customize it to your specific needs. • Having physical control over patron data (comments on webpages, book recommendations, or anything else you enable your patrons to do on your site) may be important to some libraries. Cons: • You have full responsibility for your hardware and backups; depending on the depth of your library’s technical staff, meeting your expectations for uptime may be a challenge.

	your server. • There may be restrictions on other tools, etc., that you can install and run on the same hardware.	
Development	Pros: • You will likely benefit from a range of experts, from coding to user interface design to usability. • A qualified firm has likely done similar projects before, perhaps even for libraries, and will have understanding and experience of your particular needs. Cons: • Your organization may not	Pros: • The people who build the site are the people who maintain it. • Your new site will be built by developers whose main mission is aligned with that of your organization. • Maintenance may be much easier because there's no hand-off from creator to maintainer. Cons: • Your programming staff may need some time to learn Drupal's particular framework. • You can face the same hand-off

	understand the application you receive at the end of the development cycle. • Changes to the specification may be expensive.	problem as with an external vendor if your developer changes jobs. • Projects developed in-house can suffer from the "cobbler's children" problem: it can be a struggle for those assigned to work on a project to find time to work on it along with all their other responsibilities. It can be easy for them to get sidetracked or delayed.

- What kind of uptime do you require from your website, and are you staffed to provide it? Do you have employees who are on call during evenings and weekends who can, in an emergency, troubleshoot and fix a problem? While servers and Drupal can be considered fairly stable, does the site need to be available 24/7/365? If so, is your staffing such that there are people on call overnight, on weekends, and on holidays to fix problems (from power outages to software problems) if they occur?
- What level of control do you (or your administration) want to have over the entire website, soup to nuts? Some organizations simply want to be in charge of all aspects of the website, while others are satisfied leaving the infrastructure to someone else and are concerned only with the content.
- What level of flexibility and integration with other services do you hope to achieve? If you host everything yourself, and have the staff to maintain and work with the software tools, there may be functionality you can develop locally by virtue of having knowledge and access to

all pieces of your technology puzzle. While this is not impossible with distributed management of hardware, it becomes more difficult because additional steps—accessing and modifying software on other system—become necessary.

As you think about the answers to these questions, you will likely discern a fairly clear path to resolving the question of whether to host your site locally or at an external site.

Hosting Services

The first option to consider for hosting your site is a commercial hosting service. If you don't already have your own web servers, or don't wish to take on the overhead of managing another one, you can take advantage of a hosting service. Hosting services handle the hardware and (often) the server software for you, giving you access as a Drupal administrator to configure the site as you like. The hosting service takes care of hardware and software upgrades and makes sure the latest operating system and application security patches are in place.

If you choose the hosted services route, there is yet another layer of options. Many Internet service providers offer Drupal along with other tools (blogging software, wikis, etc.). The degree of support varies widely—some companies offer add-on services to design and develop your Drupal site, while others merely install the core code and leave it to you to make it work. In this sense, a locally and externally hosted solution can be very similar, with the main difference being who keeps the hardware running.

At the other end of the hosted spectrum are full-service companies that will host, develop, update, and maintain your site (we'll talk more about the pros and cons of this later). Some are Drupal-specific, such as Acquia (http://acquia.com/). Others are library-focused, such as LisHost (http://lishost.org/). Many, many such service companies are out there, ranging from the local to the national.

Your Own Server

If you already have server infrastructure in place, running your own web server with Drupal installed on it is another option. Deciding which option is best requires some understanding of how intensively you expect your site to be used in terms of simultaneous users and the complexity of the pages you are presenting. You will want to make sure that you size your server (computing power and memory) appropriately to handle this anticipated load. (We'll talk in more detail about technical requirements for Drupal in Chapter 5, "Implementation.")

If you do not already have your own servers, though, you will want to consider what you will need to build a solid web infrastructure for your Drupal site. At a minimum, you will need sufficient bandwidth at your library to serve your community (you will now be pushing more information out than before, where perhaps before you had only public computers downloading information from the web), good power, a backup plan for your hardware in case of data loss, and (strongly advised) a separate development server so that you work out of the public eye (more on this later).

If you aren't already hosting your own site, why would you want to start? At a basic level, hosting your own site gives you complete control over your own infrastructure. This benefit may not appeal to all libraries, particularly smaller ones. Perhaps a stronger driver for managing your own site is to reflect institutional priorities such as maintaining control over all your data and your patrons' interactions with it, whether in the catalog or on the website. Libraries with strong policies to protect user data may find that managing their Drupal server themselves is the only way to go, particularly if they plan to encourage user-generated content—comments, ratings, tags, and so forth. While this sort of data may not be as protected as circulation data (it is, after all, by its nature publicly available), some libraries may feel the need to keep control over it.

The Cloud

A hybrid option that many organizations are looking at is a cloud service such as Amazon Web Services (http://aws.amazon.com/).

Amazon offers, at a reasonable cost, space on their vast server infrastructure. Amazon, for example, has built their e-commerce platform to handle the highest peak demand they foresee. This gives them a tremendous amount of unused capacity most of the year. They provide server space to all comers. You can pick an operating system (various flavors of Windows or Unix) and install software. You can even provide your own domain name. They keep your data and applications up and running while you have full administrative control over the applications. You pay by the volume of use (size of the database table and gigabytes of data transferred to users). Should your use increase, Amazon automatically adds additional server resources to handle the load. Conversely, when demand decreases, Amazon scales things back. For some, this solution offers the best of both worlds.

• Pick a Developer

The discussion of how to develop your Drupal site is analogous to the discussion of where to host it. At one level, Drupal is a PHP application with a database, often MySQL, in the back end. Any competent PHP/MySQL developer can successfully work on Drupal. At the same time, it is more than that. You often hear Drupal developers talk about the "Drupal way" of creating code. The "Drupal way" is not clearly defined but mostly centers on ensuring that the code you create is logically consistent with the way Drupal does things. While this is perhaps obvious, many developers find becoming fluent in Drupal something like becoming fluent in a spoken language; there are vocabulary and grammar, which can be memorized, but the accent marks you as a native. Once you've learned Drupal's "accent," Drupal becomes more powerful.

Questions to ask yourself as you decide how to source the development of your Drupal site include these:

- What are your short- and long-term funding prospects for development and maintenance? This question is really getting at your staffing levels. Do you have programmers on your library's staff who can be redirected to work on Drupal for the project development cycle? Can

you hire a developer on a short-term contract for the ramp-up to launch?

- If the development of the site is outsourced, who will maintain and manage the site after launch? Do you (or will you then) have staff who are sufficiently trained in PHP and MySQL to keep the site running, or does the long-term maintenance also need to be arranged for? Development and maintenance are tasks at different ends of the scale. While there are obvious benefits to having the same person or people maintain the site they built, it is often easier for someone with limited Drupal experience, but some programming background, to modify and maintain existing code than to create it from scratch.
- How confident are you in your organization's long-term ability to maintain either resident or outsourced staff? Depending on the environment in which you operate, there may be institutional preferences for one over the other.

The answers to these questions should help you decide which direction you might go. We will now discuss, briefly, the relative merits of each approach.

Outsourced Developers

If you work with developers outside your organization, you set up a contract, provide the developers with the parameters of the website you want built, and they, in turn, deliver the code. There are many companies—ranging from full-fledged design studios to freelance developers—that could take on your project. As with any software purchase, it is wise to perform due diligence on the company or individual you select: checking references, looking at sample work, and learning how the potential contractor manages the work flow are all obvious steps.

When you hire a company to handle your website development, you should expect to spend a significant amount of time meeting with that company's design team. They will want to understand your needs and desires (your "requirements") so they can build the site according to your specification. The requirements should be very specific and detailed and will likely involve mocked-up interfaces for virtually every kind of page content or user interaction. While careful

planning (see Chapter 3, “Planning”) is a must in any project, it is critical when working with an outside company that must learn not only what needs to be done within your site but also your library’s particular cultural and user environment.

On the development side, once you have achieved your set of specifications, you can expect development and testing work to proceed fairly quickly. The company you hire probably has a large number of developers, graphic artists, and usability experts to draw on and can make quick progress once the slower planning phases are completed. However, it is important to keep in mind that once development work starts, it is likely to be both inconvenient to the process and expensive for the budget to change direction along the way. This is hardly unique to outsourced development projects; it is simply that the costs are more explicit and contractual rather than “just” additional time spent by the development team if they are on staff.

Make code documentation an explicit requirement of the contract. It is almost never easy for one developer to pick up rapidly on the work of another. Making sure that the code you accept is clearly documented, that the community modules that are used are noted and their functions explained in the context of your site, and that any custom modules are particularly well documented and explained will hold you in good stead down the road. Excellent documentation will ensure that not just the development company—whose personnel may change by the time you request updates—but your staff, or another company that you may hire later, can quickly gain a solid understanding of how the application is built.

These details might make it sound as if outsourcing development is not a good choice. That is not necessarily true. Outsourcing carries with it some advantages, particularly in the speed and often the quality of the product that is delivered. Depending on the staffing and budget capabilities of your organization, it may well be a logical and practical choice.

Internal Developers

Doing your development in-house in some ways gives you more flexibility and control over the project. Building it yourself conveys several advantages, if you have staff who are well versed in PHP (or another programming language) and MySQL and can be given training in PHP (if needed) and Drupal. As noted earlier, the "Drupal way" is not significantly different from other PHP programming, but it does require learning certain methods to make development and maintenance as easy as possible.

One of the largest advantages of managing your own Drupal development, and one I feel cannot be overstated, is that the development effort will be clearly aligned with the organization's vision and goals. This alignment, as with virtually any endeavor, can be critical to the success of the project. Even if a developer's approach is different from a librarian's, having both work toward the same organizational goal will create a site that is more likely to be viewed as a success by the organization. Internal developers will be much more able to understand and act upon the organizational mission than an external developer.

There are additional, significant benefits to having the developers work for the library. First, a developer who is a part of the staff can have full participation in the planning and decision-making processes in addition to taking the lead role implementing those decisions. This integration not only provides better decisions—informed, in part, by someone knowledgeable about the technical ramifications of design or functionality ideas—but also gives the developer a sense of shared ownership in the site, not just the code behind it.

Thinking in the longer term, maintenance and enhancement of the site can be accomplished by staff who understand the code at a fundamental level; the hand-off from developer to maintainer is minimized—except, perhaps, when the developer in a one-developer shop changes positions. The risk of this is not exclusive to in-house development, of course.

In the next chapter, we will continue our discussion of the preparations for developing a Drupal site with an overview of planning. What are the broad areas that you will want to start thinking

about now, before you begin developing your site? How can you create a solid project plan that will guide development work?

3

PLANNING

- **Inventory Your Resources**
- **Determine Your Goals**
- **Determine Scope of Development Effort**
- **Assess Staffing Needs**
- **Create Site Design**
- **Develop Functionality**

A good project plan will get you through the project even when you run into unexpected speed bumps or problems along the way. Aside from building a firm understanding of the functionality you want to achieve in your new library website, you also need to address a range of fundamental questions at the outset. The adage goes, "failing to plan is planning to fail." Like most adages, this one is a gross oversimplification. It still contains more than a grain of truth. You will not be able to identify, let alone arrive at a contingency option for, every possible problem or challenge you will face. However, by developing a collective understanding among all project stakeholders of the major desired outcomes, critical functionality, and interface features and developing a technology plan based on these items, you will reduce uncertainty and generate goodwill and understanding within your library.

By the time you begin active development (see Chapter 5), you should be able to answer such basic questions as these:

- How much time, budget, and expertise do we have (or have access to) to build the site?
- What functionality is core to the project's success, and what is secondary?
- What should the finished site look like?

- Who will do the work during construction, and who will maintain it once it's built?
- Where will you host your site?

It's important to keep in mind that these are not necessarily sequential steps. You will want to start with an assessment of where you are and where you want to go, but then much of the rest of the process will happen simultaneously with answers to one question influencing options in other areas.

• Inventory Your Resources

Any large web project starts with a chicken-and-egg question ("How many resources do I need to build the site that I want to build given the time and resources I have available?") and ends with a vague answer ("It depends."). If you have ever contemplated an addition or renovation to your home, you know the initial conversations with your contractor can be maddening. You start with, "I want to add a family room," to which the contractor responds, "How much do you want to spend?" You give a figure, which then turns into an estimate of what you can do for that money, which turns into a new figure, or changed features, or different finishes. Each decision you make informs the others; you rarely arrive directly at a precise target. Rather, you gradually go half the distance until you have ended up with a clearly defined goal.

A good place to start is to take a set of basic inventories of resources. Who is available in your organization to work on the project? You will need to identify people capable of working in several areas. Managing Drupal itself is the obvious one; who will be doing the installations, configurations, and maintenance? Depending on the size of your library, the answer to this will vary. It might be your existing systems staff. It could be technology-savvy librarians who are interested and willing (or able to be convinced) to take on this task. You could hire short-term staff to handle development and turn over long-term maintenance to permanent staff.

You also need to figure out who will be doing the design. Drupal is a modular development environment that separates the interface from the functionality, allowing different "themes" to be overlaid on the site with a relatively small impact on programming. An early decision point, then, is deciding whether to use an existing theme from the Drupal community, take an existing theme and make relatively minor modifications to it to suit your library's existing graphic identity (color scheme, placement of logos, etc.), or develop a theme from scratch. The closer you get to ground-up development, the more time and efforts are required.

Having a basic usability assessment process in place for stages of design from early prototypes to after you've launched your Drupal site is also important. This kind of assessment—designed to answer basic questions such as "Can users find functionality X?" or "Is the link to Y labeled properly?" to "Where should the link to the circulation desk be located?"—can be done effectively yet informally. It is important to the ultimate success of the project to do usability verification early and often. When we designed the University of Michigan Library's site, we tested it repeatedly with library patrons using paper prototypes (printouts of pages mocked up in Adobe Photoshop), early versions in Drupal, and the finished version in Drupal.

• Determine Your Goals

The second most important area of decisions centers on the functionality you want your site to have. What do patrons do when they come to your site? What resources do you offer them? How do they want to interact with your library when visiting virtually?

You probably have a great deal of information about how your patrons interact with your current site. This could come from server log file analysis or tools such as Google Analytics. Perhaps you have conducted small focus groups of your patrons to ask them what they like and do not like about your site. It is likely that your library's public services staff have a great deal of information about what patrons find

easy to use and hard to use—after all, they are the ones who talk to the patrons after the site has failed—but it is still important to rely on first-person evidence.

As you develop a list of functions for your site, start prioritizing them. What are the absolutely must-have items, what would be nice to have, and what would be icing on the cake? For most libraries, the must-have list will include things like these:

- Catalog search
- Database finder
- Online journal finder
- Event calendar
- Directions to the library, perhaps with an interactive map
- Library hours
- Managing patron accounts
- Contact links for the library, reference service, and circulation desks
- Signing up for the library newsletter

the student in need of research materials at the last minute; the genealogist looking for family history information; the small business owner doing competitive intelligence for his company; and a faculty member putting together a research proposal.

Put yourself in the place of each of your personas and think about the kinds of tools and resources they will need from your website to get them through their tasks from start to finish. Then, combining this information with what you know from usage statistics, interactions with your patrons, and the tools and services the library feels are most valuable, come up with a priority list of functions your site cannot do without. Aaron Schmidt and Amanda Etches's book *User Experience (UX) Design for Libraries* (THE TECH SET #18) has more on persona creation and use.

• Determine Scope of Development Effort

Regardless of where your Drupal site lives, you have three basic options for developing it: using Drupal out of the box, adding modules

from the community pool, or modifying/creating modules to match your needs. Which option you choose depends largely on what you want to do—how different is the functionality you want to create from what others have already done—and what set of skills and capabilities you have to work with. We'll discuss each of these three options in turn. It is likely that your site will end up reflecting a mixture of the second and third options—you will probably end up with a mix of community modules, community modules you adapt in small ways to meet your needs, and custom modules you create from scratch.

The more custom modules you develop, and the more you customize community modules, the greater the long-term maintenance burden you take on. Let me explain. The Drupal community of users maintains modules and keeps them up-to-date (to improve functionality, to patch bugs, to close inadvertent security holes, and to keep the modules functional as Drupal's core code evolves). To the extent that your modules are different from the modules shared with others, you will have a larger unshared burden to make sure that your site can survive an update to Drupal's core with a minimum of effort on your part. This is the basic hierarchy you should follow as you consider module development:

1. Use a community module. There are more than 3,100 to choose from.
2. Modify a community module to meet your need, if you cannot find a way to use community modules to meet your needs.
3. As a last option, develop your own module, but only go this route if you cannot find a sufficiently similar module to modify. If you do pursue this option, consider submitting your new module back to the community so that others might adopt it and find ways to further improve it. Your site's needs are probably not unique in the world of libraries or even in the world of Drupal libraries. What you create may be helpful to others.

Out-of-the-Box Drupal

One of Drupal's strengths is that you have a fully functioning content management system as soon as you've installed it and performed the initial setup steps. Drupal comes with four basic themes (Bartik,

Stark, Garland, and Seven). Each theme has a slightly different purpose. Bartik is the default theme for the public view of the site. Seven is the default administrative view (see Figure 3.1). Stark is a bare-bones theme designed to show the novice user how Drupal pages are structured, while Garland, which has been part of Drupal since Drupal 5, is more detailed and feature-rich.

Drupal's core, or basic, functionality is there as well. This means that you can turn the site on and start creating and publishing content immediately. Few organizations will want to have their site go live to the public without making some adjustments to the basic theme and functionality. However, with a bit of CSS customization, you can quickly change colors, font sizes, and so forth, to update the basic Garland theme. The Drupal community has created an extensive collection of themes, all of which are available to you to "reskin" your site—all through a simple configuration interface (see Figure 3.2).

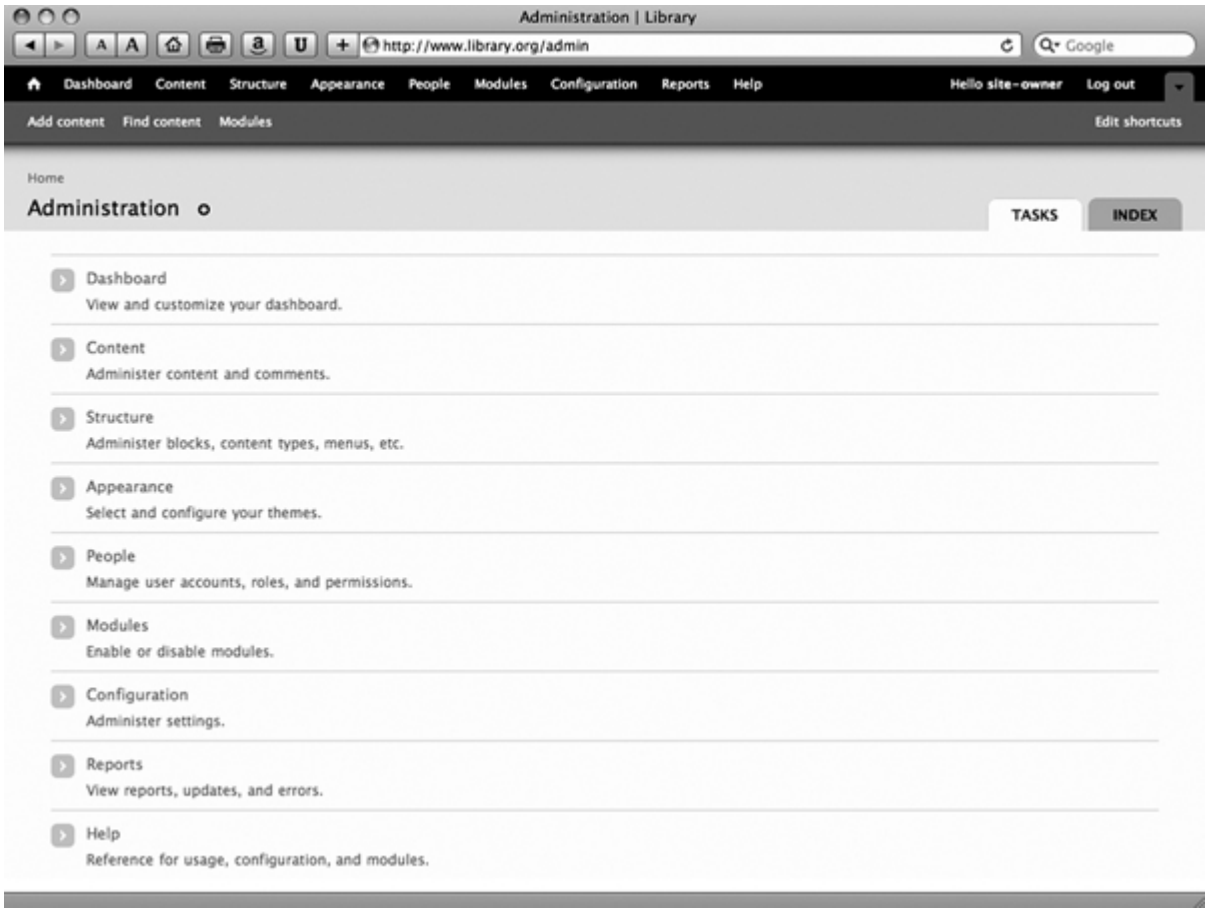

Figure 3.1: Drupal's Site Administration Page in the Garland Theme

Use Community Modules

Figure 3.2: Drupal's Bartik and Seven Default Themes and the Contributed Zen Theme

A plain download-and-install version of Drupal gives you the ability to create users and assign them roles in the site, publish pages, and

build navigation menus, among other basic functions. All in all, this is sufficient to get a basic site up and running very quickly and allow you to establish authoring roles to match your organization's needs. It will, for example, give some individuals the authority to write and edit pages, others authority to write, edit, and delete pages, and still others who have full administrator access to the configuration panels to change the site's look or functionality. You can change themes, establish navigation, and create page templates. In short, you can have a functioning site using Drupal with a modicum of development effort. You may even find that the content is more time-consuming to work with than the system itself. However, there are other functions that do not come "out of the box" and that must be downloaded and installed. I'll discuss the mechanics of installing a new module in "Install Modules" in Chapter 5 (pp. 49–52); here, I'll give an example of specific functionality a module can give you.

Let's say that you want to add the popular Pathauto module to give pages in your site human-readable URLs rather than Drupal's default, somewhat unfriendly system-generated URLs. Without this module, a page titled "Getting to the Library" might have the unmemorable URL http://www.library.org/node/31. Conveying no particularly important meaning to the user, the number is the identifier of the database entry for this particular page. A more user-friendly URL for this page might be http://www.library.org/getting-to-the-library. The Pathauto module automatically converts the page title into an alias for the actual URL and sets up the behind-the-scenes mappings for the human-readable URL and Drupal's machine-generated one.

Pathauto can also set up multiple aliases so that you can have multiple URLs—both http://www.library.org/directions and http://www.library.org/getting-to-the-library direct to the same page. You might want to do this if the page title changed from the previous site but you wanted to make sure that bookmarks for the old link still worked.

Customize Your Own Modules

Drupal's greatest strength, ease of customization, can be its weakness as well. Because of Drupal's flexible architecture, a site administrator can add just about any kind of functionality through a new custom module. As a rule of thumb, avoid this practice unless there are truly no existing community modules that achieve the same goal or come close.

If you do customize your own module, you should create it in such a way that it could be accepted as a community module if you chose to do so (see guidelines for this process at http://drupal.org/node/7765). At the least, following these good practices will reduce your future maintenance chores to the bare minimum, because a well-architected module that plays nicely with the current version of Drupal core will be more likely to play nicely with an updated version. And if Drupal core is updated, the migration path to new functionality equivalent to the superseded functionality is generally documented, making the module developer's job easier.

• Assess Staffing Needs

In addition to the staff, whether internal or outsourced, who will be doing the programming and design work, your Drupal site will likely involve a large portion of your library's staff on the content side. While you can realistically outsource graphic and application design, you and your library staff will take responsibility for your content. It is the whole point of the site after all.

During Development

You are presumably replacing your current website, which may or may not be in a content management system, with a Drupal-powered one. The first step is to perform a content review so that you can separate content that needs to be moved to the new system from content that is no longer needed. There are several approaches to such an inventory. If your site is built of plain old HTML files, you can start from the file system's directory to list out all of the nested files and folders. If your site is already in a content management system of

some kind, it almost certainly has a site map or export tool that can list all of your site's content. See "Recommended Reading" for resources on conducting a site inventory.

The people who are responsible for the content should likewise be responsible for reviewing it and migrating it into the new system when the time comes.

Reviewing without Responsibility

When the University of Michigan Library migrated from flat files into Drupal, we asked the dozens of people systemwide to review their content for migration, with the understanding that any content they selected would be moved for them. We hired students to copy and paste from the webpages selected by staff into Drupal. Because there was no penalty for moving lots of content, many staff were lenient when it came to the content review and opted to keep everything. This resulted in our students moving a significant amount of outdated, redundant, or just plain not useful content into Drupal initially.

When the project management group reviewed the often lengthy lists of content to be migrated and communicated more directly with the content owners, the second pass resulted in a much shorter list of content to migrate. As a related note, migrating the content centrally may be politically expedient, but giving the content owners a role in this process helps it go more smoothly and allows last-minute content decisions to happen in the flow.

As you review your content, there are several categories of content you may wish to consider leaving behind:

- **Outdated information**—pages describing services, events, tools, and so forth that are no longer offered by your library but that are still on your current site (even if those pages are "orphans," no longer linked from anywhere).

- **Orphaned pages**—if it's not linked to anywhere on your site, is it really useful?
- **Duplicate content**—for example, if you have the same information on library hours in multiple places, it is more difficult for your staff to keep the site accurate, and it is more confusing for the user. There is probably a great deal of similar content that could, and should, be consolidated.

After Launch

Once your Drupal site is launched, the big content push is over and you move into maintenance mode. You may not need to have all hands involved in content management, but it makes sense to establish a few roles for staff on the website. If you have a large organization, it may be useful to have people taking care of these roles in each unit. The simple hierarchy described here will likely work well for many organizations, but tailor the roles and responsibilities to meet your library's particular needs:

- **Content Author**—the person who is authorized to write and edit content (whether at the unit or library level, depending on the size and style of your organization)
- **Content Editor**—the person who reviews new content, approves edits, and deletes unneeded pages from the site
- **Site Administrator**—the person who has the ability to manage Drupal itself, such as adding modules and updating themes (often several individuals have this role so that changes can be made without waiting for *the* admin to get back from vacation or illness)

In some organizations, content authors are authorized to publish directly to the live site. In others, there is a review and editing stage. It generally makes sense to delegate the content questions to library staff who know most about the topic and not have this function in the systems office. Systems personnel understand the application but may not be best positioned to review content for accuracy or library style.

• Create Site Design

Drupal keeps the look and feel of your site separate from functionality and content. As you think about what you say and how users interact with your site (the content and functionality), you should also be working on the design of the site. You may have an existing site whose design works well for you, even if the content is hard to manage. Or you may view moving to a content management system as an opportune time to revamp the interface as well. Here are some points you should consider as you plan your design.

- Does your current site's design represent the image your library wants to portray? If your library is on the cutting edge of technology and services, do you want your site to be "edgy" as well? If your clientele's needs are better met by a simple, functional design, then that is the way to go.
- Is your current site accessible to those with vision or other impairments? If you have not recently reviewed your site for compliance with web accessibility guidelines, a redesign is a terrific time to do so. (Handy resources for this are listed in "Recommended Reading.")

Accessibility Is Part of Design

You should build accessibility reviews into your site's development plan so that your website is fully accessible to those who use screen readers or other assistive technology. Validating against these tools early and often will make accessibility an easy outcome. While there is no single standard for defining an accessible site, two sets of guidelines are commonly followed (and may be required by state law or local policy). The first is required for U.S. Government agencies (and has been adopted by many state and local governments and other organizations) and is known as "Section 508" (see http://www.section508.gov/). The second, the Web Content Accessibility Guidelines [WCAG], is managed by the World Wide Web Consortium (see http://www.w3.org/WAI/intro/wcag). Developing your site

with accessibility in mind generally leads to improved use for all users regardless of level of physical or cognitive disability.

- What do your users (and your staff) think of your current design? If you are not sure, ask them. This can be done through a simple online survey. Another method is to provide images of a typical page in your current system or the new system as you design it, and ask patrons to simply circle design elements they like, cross out elements they do not like, and add anything that might be missing. You will receive a wealth of information about your users' perceptions of your design in short order and can begin identifying parts of the design that work well and those that do not.

Once you have this basic information about the site's design needs you can start working on a theme. Themes are the way Drupal separates the "look and feel" of the site from the functionality of the site. They are written largely in HTML and CSS, although some PHP is needed to import the content for a particular page type display. As with modules, there are hundreds of community-contributed themes to start from, so, if you don't see one you like, you can build your own from scratch or modify an existing theme that has most of what you need.

• Develop Functionality

This is where the rubber meets the road. What do you want users to do when they are on your Drupal site? They might want to search the catalog, reserve a meeting room, renew a book, get reference help, connect to a database, enroll in a class or workshop, or perform any number of other tasks. Figuring out the functions you want should be based significantly on the goals you are setting out to achieve. As in other stages of planning, perform a needs assessment to explore what your patrons would like to do if they could and what they use on your current site and listen to feedback (either actively solicited for this purpose or already submitted through other channels). Some goals may be met by a single function; others may require multiple functions or features to achieve.

As you develop the list of functions, you can begin to compare them with both your goals and the pool of community modules. You will thus build a model of what you need and how you will achieve it and begin calculating the time needed to do any custom work. This exploration could be done through a simple table with a few columns:

- **Function**—What would this function do, who will use it, and how often will it be used?
- **Goals Met**—Of the goals you established, which ones does this function enable?
- **Priority**—How important is this function to the overall function of the site? Is this a must-have item, a nice-to-have item, or a bonus if you achieve it?
- **Community Modules**—Are there community modules that (in whole or in part) achieve this goal?
- **Scope of Work**—How much will it take to make the community module behave the way you would like or to develop your own module from scratch? At this early stage of your exploration, you may want to characterize this broadly as a small, moderate, or significant effort and then refine these estimates to actual development time as you learn more.

Now comes the hard part—making decisions about priorities based on the information you have gathered. Having some consistency in the inputs will make these decisions understandable and communicable to other staff and to whatever constituencies are following the web redesign effort. As we'll explore in the following chapter, keeping these constituencies informed about what is going on, and having consistent and open rationales for decisions, will go a long way toward smoothing the inevitable disagreements among them.

4

Social Mechanics

- **Collaborate with Your Systems/IT Department**
- **Bring Library Staff on Board**

Moving your library's website to Drupal shares much with the process of launching any other new service in your library. The project is made up of many parts, each depending on and interacting with others. At times, it can seem as if the "tangential" pieces—those not dealing directly with coding or content—take more time than the "core." This is probably not an illusion. Making sure that the people across the whole organization stay focused on the end goal and feel both responsibility and ownership for the outcomes is vital to the project overall.

In real estate, the phrase "location, location, location" describes a successful property. In project management, think "communication, communication, communication." Making sure that all parties who feel they have a say in the process are heard and that those parties who have an active role in the process at a minimum understand why decisions are made (ideally, they agree with them—but that's asking a great deal) is time-consuming but ultimately invaluable. Here are some strategies for keeping communication open and successfully navigating the organizational seas.

• Collaborate with Your Systems/IT Department

Depending on the organization, responsibility for building the system and maintaining the content are distributed across the library or even across the broader organization. When the hardware and server folks are in a different organization, special care needs to be taken in communicating the needs of the library and, sometimes, the benefits of Drupal (or other open source software). Convincing a central IT

organization to install an open source product (regardless of its nature) can at times require a broader conversation about maintenance, updating, security risks, and the like.

If you encounter resistance to installing and using Drupal, respond with some of these facts:

- Drupal is used by many companies and large organizations, including the White House website, *Fast Company* magazine, Sun Microsystems, FedEx, Sony Music, and Harvard University, to name a few. All of these customers have serious and significant concerns about the security and stability of their websites and have opted for Drupal.
- Drupal is open source, which means that system administrators and others in your IT infrastructure groups can see exactly what is going on within the code, which should give them a higher degree of confidence that it has no unintended effects.
- Because Drupal has an active developer community, the chances of security threats or other bugs lasting for a long time before being patched are small—on par with what you might expect with well-maintained commercial software.

• Bring Library Staff on Board

Your library's staff is the obvious group with significant concern about and interest in the library website. After all, not only does your site represent the collective public face of the library to a large segment of your user base, but it is also where many of your staff do their work. Making sure that your colleagues have a firm understanding of the need to move to the Drupal content management system (or to switch from another product to Drupal), grasp the basics of how Drupal works, and have become familiar with the authoring environment are all key milestones on the path to a successful implementation. In this section, we will explore strategies for bringing your staff on board with Drupal and keeping them there.

Change Management

Change management is, broadly speaking, a process for making transitions well understood and transparent. Any large-scale project can benefit from a careful plan to bring people "into the loop" in terms of defining the need to change and the parameters of what is going to change and providing staff opportunities to be part of the implementation process. Members of a staff who have been in on the implementation from the beginning are more likely to feel they own the outcome.

Depending on the scope of your project there will be more or fewer opportunities for staff to participate. If your Drupal implementation is largely a change in content management software, but not the basic information architecture of your site, you may find that staff involvement in planning is at a more superficial level. However, if you are redesigning both the site's information layout and system software, there may be much more need for broad participation. Keep in mind that it is all but impossible to communicate in too much detail, or too frequently, what is planned, what will happen next, and what the end goal is.

Buy-In

The change management process starts before you define the change you want to make. Invite key library stakeholders—administration, public service librarians, people responsible for major content sections of your current website, and your IT partners (internal or external)—to participate and help make decisions from the onset of the project. You will find it greatly beneficial to say, "The library selected Drupal," rather than "I selected Drupal," innumerable times throughout the development process.

The more broadly based the decision-making bodies, the more opportunity you will have to educate and explain Drupal's advantages to core constituencies in your organization. All possible major milestones in your project—determining the need to move to a CMS or to change CMS systems, selecting the CMS, prioritizing functionality, designing the user interface—should be undertaken with input and participation from across your organization. If you have

a large group, this might be multiple task forces or advisory committees; if yours is a small one, it may end up being a smaller web redesign committee.

Committees and Meetings

When my library at the University of Michigan undertook its site redesign, we established a series of task-focused committees to help us with planning and implementation. Our library is sufficiently large that we were able to create whole committees and working groups around each topic; smaller libraries can assign responsibility for these tasks to individuals. While our structure as outlined here may not be exactly right for your organization, it illustrates areas where responsibility for decision making might be distributed:

- **Technology Advisory Group**—a group of library IT and technically minded public services staff who established a set of criteria, reviewed a suite of CMS tools against this criteria, and ultimately made a recommendation in favor of Drupal
- **Faculty Advisory Group**—a group of campus faculty members (some of whom were heavy users and "fans" of the library, others of whom were identified as nonusers) who discussed specific information needs they had that were, or were not, being met by the current website
- **Information Architecture Advisory Group**—a group of library staff who helped determine the high-level structure of the new library website and worked on the taxonomy for it
- **Focus Groups**—volunteers from the student community who participated in guided discussions about the current website and their information needs
- **Authoring Group**—the roughly 100 people who would be given authoring responsibility in the new site served as a broad focus group and became the source of many ad hoc groups formed to answer specific questions

In a smaller library, one person could be responsible for each of the functional areas, with a group of individuals making up your project steering committee.

All of these groups, collectively, were convened and reported back to the "Web Team," a set of four library staff (the Web Systems Manager, the Web Content Manager, the Digital Information Services Librarian, and the Director of Communications). The Web Team was chartered by, and reported directly to, the library's management team. This reporting structure reflected, in our case, the desire to revamp the website expressed by the library's senior leadership. Regardless of its provenance, the reporting structure encouraged the Web Team to make strategic decisions based on the library's goals rather than the goals of any of our respective departments.

Constant Communication

Put bluntly: No matter how hard you might try, you will not be able to communicate too much with your colleagues about the project.

Your library's staff will be interested in what is going on, even if they are not active participants. Consider taking such steps as sending regular all-staff e-mail updates (every week or two) to let everyone know what has happened in the project recently and what is up next. Invite feedback from staff early and often through all stages of the project, from start to finish. If you're the project manager, make yourself available at staff meetings (or departmental meetings if the library is sufficiently large). The effort you spend in talking about the project, hearing—and adapting to—the feedback you will receive, and telling people what is happening next will be returned to you in shared vision, collective acceptance of the changes that are being implemented, and a broad understanding of where the library website is going. Not least important, you will get a wealth of ideas that may never have surfaced through other means.

After the Launch

As we'll discuss in Chapter 8, "Metrics," your job is not done when the site launches. There is always version 2. Setting up a habit and expectation of good communication through the planning and development process will smooth the way toward a more efficient

project review and, ultimately, planning for the next generation of the site when the current site reaches the end of its useful life span.

Now that you have some strategies for managing the project to help guide you toward a successful outcome, we will turn to developing in Drupal. In the following chapter, we will walk through all the steps you will need to install Drupal and configure it for a basic library website.

5

IMPLEMENTATION

- **Install Drupal**
- **Find Your Way around Your Installation (Where Things Are)**
- **Install Modules**
- **Install Themes**
- **Configure a Basic Library Website with Basic Modules and Tools**
- **Style Your Site by Going Beyond the Standard Themes**
- **Integrate Library Resources into Drupal**
- **Enable Social Media Features on Your Site**
- **Get Content into Your Site**
- **Keep Content Up-to-Date**
- **Create Your Own Modules**
- **Give Back to the Community**

Now we turn to the heart of the matter: installing, configuring, theming, and using Drupal. In this chapter, we will go step by step through the process of building a basic site for your library, starting with installing the core software and continuing through installing modules and themes, using the most common suite of core modules, and populating your website with content. We will then move on to more advanced topics such as adding your library's catalog data to your site, strategies for integrating other kinds of content, and building a social network around your pages. Finally, we will turn to a discussion of building your own modules and contributing your innovations and creations back to the Drupal community.

As a reminder, we are focusing in this book on Drupal 7 (released in January 2011). The main changes between the most widely used versions of Drupal—Drupal 6 and 7—are on the administrative and architectural sides of things. The Drupal development team focused on the usability of the user interface for administrators and authors as well as performance improvements and functionality changes. The latest version does not make significant changes to the installation and configuration options, although most of the core modules (Views, CCK, etc.) discussed in this chapter will have added functionality.

• Install Drupal

Drupal can be easily installed on web servers running the Windows, Linux, or Unix operating systems. Because Drupal depends on other tools that are commonly part of the operating environment on a web server, installing Drupal is often as simple as downloading the application and running the installation script. The requirements to install Drupal are straightforward, no matter the platform (Unix or Windows) you use.

If you are using a commercial hosting service, you may find there are "one-click installs" of Drupal available to you through the hosting service's customer site. When that's the case, simply use the easy installation option your host provides, skip the rest of this section, and go ahead to the section Find Your Way around Your Installation (p. 44).

Creating Drupal's Operating Environment

The first thing you need to do in order to install Drupal is set up a proper operating environment for the software to run. Drupal needs a web server, a database application, and PHP in order to operate. Although there are many different configurations you could choose, the most common operating environment for Drupal, as well as many other software applications, is a "LAMP" or "WAMP" stack. LAMP

stands for “Linux, Apache, MySQL, and PHP,” and WAMP stands for “Windows, Apache, MySQL, and PHP.” If you’re using Linux or UNIX, you will be setting up a LAMP environment for this portion of the project. If you’re a Windows user, you’ll be installing a WAMP environment.

1. The Apache web server (version 2.0 or greater is recommended) is available at http://httpd.apache.org/. It is a free, open source application that can be easily downloaded and installed by following the documentation on the project website. For the purposes of installing Drupal, all that needs to be done is a basic installation of the web server program to your hardware.
2. After you’ve installed the web server the next step is to install a database such as MySQL 5.0.15 or higher. (The PostgreSQL, SQLite, and MariaDB databases are alternatives.) MySQL is a free, open source application with ample documentation on the project website at: http://www.mysql.com/.
3. PHP version 5.2.4 or greater is recommended. Check with your ISP or system administrator to see if this is available or if you have to install it to your server space.

Installing Drupal

Once you have the operating environment in place, you’re ready to install Drupal. It is available as a compressed file in the “Download & Extend” section of the main Drupal website (http://www.drupal.org/). Figure 5.1 shows the main download page, from which the current versions of Drupal can be downloaded. You will need to select the version (the Drupal community actively supports the current and previous main releases so you will see both the final version of Drupal 6, with security patches, as well as the current release of Drupal 7). The package that you download includes “Drupal core”—the basic Drupal toolkit. This is a fully functioning version of Drupal with a suite of modules to provide a basic website and several themes to style the interface. As we’ll discuss later in this chapter, you will almost certainly want to expand on this core installation but you do not need to in order to start working with Drupal.

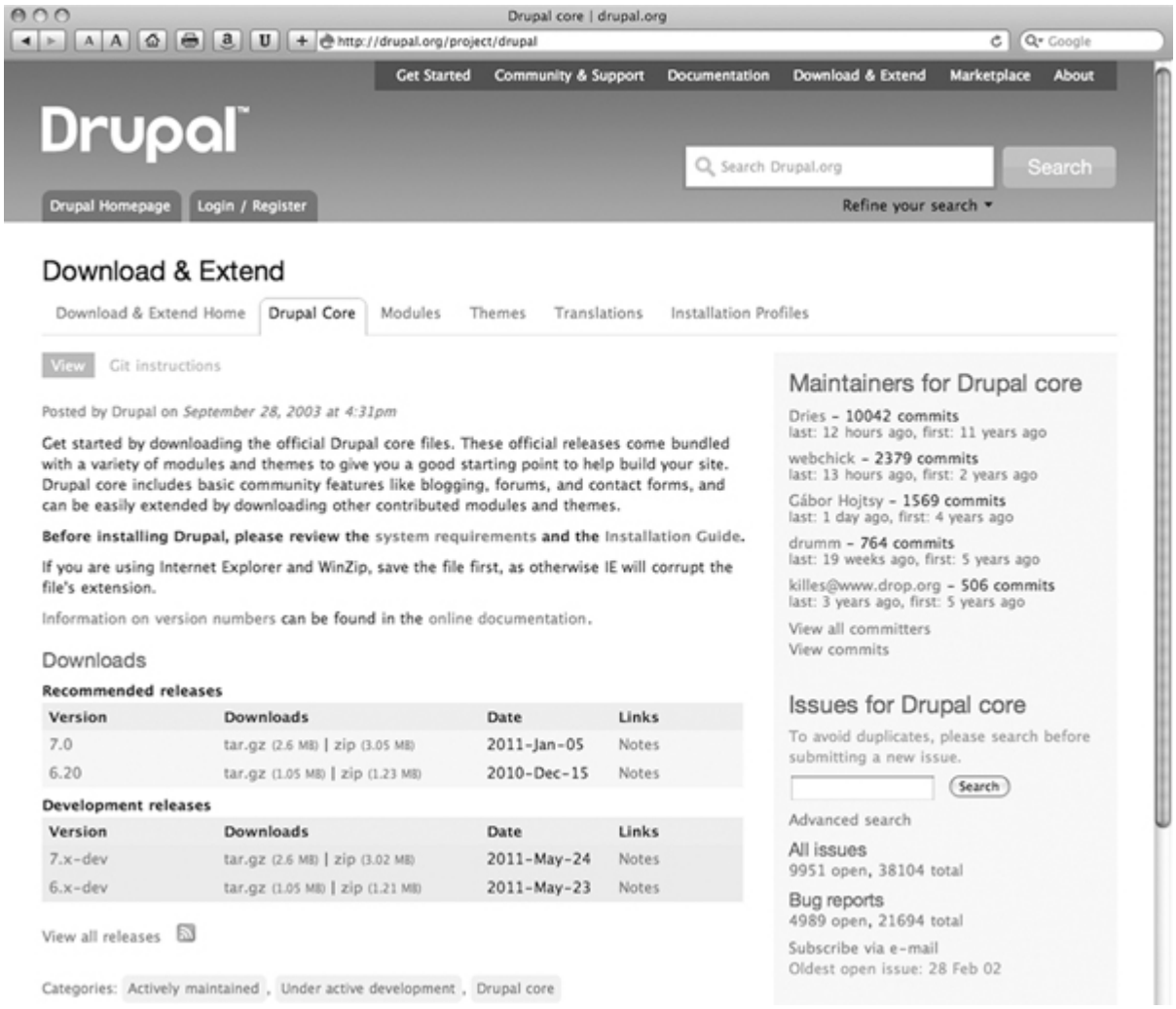

Figure 5.1: Screenshot of Drupal's Main Download Page

Drupal core does not take up extensive disk space—3–4 MB of free space on your server's storage, depending on the platform and specific configuration of your Drupal installation. However, this amount can grow quickly as you install additional modules, develop new content types, add images and media files, and, of course, upload actual content. Count on having 30–50 MB free on the server, and have a plan to expand this amount should it be needed as your site grows.

The instructions included in the Drupal download package are tailored for Unix installations and assume that you are managing the

installation yourself. There are three basic steps to get your Drupal site for Unix/Linux or for Windows up and running:

1. Download and extract Drupal to a web-accessible location on your web server. Drupal comes in a compressed file, so you need to download the compressed file and expand it into its constituent parts.
2. Create a database for Drupal to use. Drupal uses the database to store content, content types, and essentially everything it needs (other than modules and themes, which are files stored on the file system) in the database. You do this either from the server's command line or, if you have one, a web interface, such as MyPHPAdmin. This tool, if it is not already available, can be downloaded and installed on your server. With MyPHPAdmin, you set up a MySQL database and create user accounts in it through a webform. (See Figure 5.2 for a sample screenshot to create a database for the basic library.org website.) In this example, we are creating a database called "library-web-site" on a MySQL server called mysql.drupalinlibraries.org. The first user—the first administrator of the database—has a username (admin) and password. There's a comment field to identify the purpose of the database; this is optional but well worth the keystrokes if you end up creating many databases.

   If you are not sure what database applications are available on your web server or how to create a database in it, check with the group or company that runs your web server to find out if you have a web interface to your database administration functions.
3. Run the install script. Drupal automates most of the installation and basic configuration procedures. Because you moved the basic files to a web-accessible location as part of the extraction step, you can now point your web browser at the install script. For example, if you put the Drupal files in your "web root"—the top of the file space for your website—you would point to http://library.org/install.php (with your library website's server address in place of "library.org"). You will then be asked questions (such as the location and name of the database you created in step 2) to configure the installation. Depending on how your server's system administrators have configured your server, you may receive reports of problems. These problems do not, as a rule, prevent Drupal from being installed but may require you to manually create directories where content is eventually stored or to change permissions on files or directories.

Create a new MySQL database:

Database Name: Try putting your domain name in front of your database name if the name you want is already taken.	library-web-site
Use Hostname:	Create a new hostname now...
New Hostname: There is a delay while DNS propagates for new hostnames.	mysql . drupalinlibraries.org
First User:	library-db
New Username:	admin
New Password: Must be at least 6 characters	•••••••
New Password Again:	•••••••
Database Comment Optional - for your own organizational use!	For the new Drupal in Libraries site
Your database will be created right away, however new hostnames will need time to propagate.	Add new database now!

Figure 5.2: Screenshot of the MyPHPAdmin Page to Create a Database

Finishing Setup for a More Secure Site

You have now set up a basic Drupal installation, but you're not quite done yet. There are two additional steps you should take now to make sure that you keep your installation secure from potentially malicious hackers:

1. **Change permissions.** You will want to change permissions on certain directories or files so that other, more nefarious people cannot run the installer script themselves or view the log file in which Drupal records the modules and versions of modules you have installed. Drupal's documentation provides suggestions for these changes.
2. **Set up cron jobs.** A cron job is a scheduled task the server performs. Drupal sets up a number of these tasks automatically that are triggered periodically—every minute, every day, and so forth—while others might be triggered whenever a user visits a particular part of

the site so that content is updated, for example. The documentation at http://drupal.org/cron provides suggestions for specific tasks you might want to set up; more will become clear to you as you develop your site's functionality.

Congratulations! You should now have a working Drupal installation ready for you to start populating with content and customization to meet your needs.

Acquia Drupal

If you are interested in experimenting with Drupal but do not have access to a web server, Acquia offers a free set of software that will allow you to install Drupal on a desktop or laptop computer. The software download includes everything you need—Drupal, MySQL, and a web server—to get started with Drupal 7 on your Macintosh or Windows computer. As of this writing, Linux users can access only Drupal 6, the previous version. Full information and download links can be found at http://acquia.com/downloads.

The Acquia download gives you a quick and zero-cost sandbox to explore Drupal without involving the IT group in your organization. Acquia is a company that provides Drupal consulting as well as beginning-to-end Drupal development.

• Find Your Way around Your Installation (Where Things Are)

Once you have completed initial installation, most of the additional work you will do as a Drupal administrator happens through the administrative interface. This can be reached by adding "admin" at the end of your site's main URL. For example, if your main page is at http://library.org/, your administrative interface is at http://library.org/admin.

The Administrator's Interface

Unlike Drupal 6, Drupal 7's administrative screens are AJAX-based and provide clearer, more intuitive options for novice or intermediate users. For people familiar with WordPress (a blog and publishing tool) the basic layout and functioning of the administrative interface will seem generally familiar.

In a default installation of Drupal 7 using the Bartik theme, the administrative interface is divided into eight sections. These are displayed across the top of the page when you are logged into the administrator's view:

1. **Dashboard:** The Dashboard is the row of links directly below the main administrative list. You can add links to specific Drupal functions (such as a commonly configured module or a link to create a new page of a certain content type) to the Dashboard.
2. **Content:** Here you create, edit, and delete content and comments.
3. **Structure:** Here are the building blocks of a Drupal website, including Drupal's blocks, content types, menus, and the like.
4. **Appearance:** Here you select and configure themes.
5. **People:** Here you manage user accounts, giving users roles and managing their permissions (e.g., who are content authors and what sections of the site they can work in).
6. **Modules:** Here you enable, configure, and disable modules to add functionality to your site.
7. **Configuration:** Here you can change the settings for the entire site.
8. **Reports:** Drupal offers various reports, such as broken links from your site to other web pages and most frequently requested, but missing, pages on your site, along with other errors.

The Content Creator's Interface

Author-entered content in Drupal needs to be defined as a content type. (We discussed content types and nodes in Chapter 1.) A clean, out-of-the box installation of Drupal 7 gives you seven distinct content types (articles, basic pages, blog entry, book page, forums, polls, and

comments). Drupal lets you easily create additional content types that meet your specific needs; we'll get to that later in this chapter.

The first two content types are articles and basic pages. Broadly speaking, articles are meant to be shorter, frequently updated items that appear as the result of some automatic sorting process, while basic pages are single-page, relatively static content, often linked from the site's navigation. For example, a newsletter might be made up of a set of articles with a given publication date or topic, whereas the instructions for subscribing to the newsletter might be a basic page. These definitions are more operational than practical. These content types are active by default. You can, of course, deactivate them if you choose.

Four other standard content types—blog entry, book page, forums, and polls—are inactive when you install Drupal. Each is controlled by a separate module. To use these content types, you will first need to activate the corresponding module through the administrative interface (see the Install Modules section, pp. 49–52). As it sounds, blog entries are parts of weblogs. By default, Drupal lets each registered user create a blog within your site. This function can be restricted to certain registered users—library staff, for example—or turned off completely by the site administrator. Blog entries attach themselves to the user's blog, have comments turned on (by default), and appear in the blog's RSS feed.

With the book page content type, books are created by the Book module and, despite the name, are intended for documentation, not to describe actual books. Book page nodes function something like a wiki in that they are intended to be collaborative. The prime example of this content type is the developer documentation on the Drupal website (http://drupal.org/documentation). Any Drupal user may comment on and augment the documentation.

The other two inactive content types are forums and polls. Forums allow threaded discussions—a topic is followed by user comments, with additional comments allowed off each previous one. Each forum node is a particular topic of discussion (created by the administrator or, if configured to allow it, by members of the public). Individual users

contribute their comments to this topic in a threaded way. For example, a user could comment on the initial post or on a particular comment. Subsequent users see the comments in a long list, with the conversation thread made clear through indented margins so that comments on the post are on the left, and comments to comments are indented. In addition to providing a place for conversation to happen, forums are a great method of gaining input and interaction from the public on your site or on other services or tools the library offers. The poll inactive content type, much as the name suggests, is a way of asking a question and receiving answers. Users select an option that you provide (yes/no or multiple choice) or enter text. You can create simple questions with multiple responses; your users cast their votes and the tool keeps track of the total.

The last basic content element, comments, is not technically a content type—it does not define a node—but it helps to think of it as such. Comments are attached to nodes much like comments go along with weblog posts. As a site administrator, you can activate comments for any content type in general or for specific nodes. Content authors can, if the administrator allows, activate or deactivate comments on their own particular nodes. Where they have been activated, they allow any user (or just registered users, if you choose) to add comments to those pages (see the Enable Social Media Features on Your Site section, pp. 76–82).

The File System

Drupal installs files on your web server's file system at the server's web root level. The web root is the directory on the server that is visible to the web server. Anything inside this directory, including subdirectories, can be served up by the web server to a user; anything above or parallel to this directory is invisible to the web server. (Of course, your system administrator can make exceptions to these normal conditions, blocking subdirectories beneath the web root or enabling access above it.) A Drupal installation contains a number of files and directories, as listed in Table 5.1.

Table 5.1: Important Files and Where to Find Them	
File/Directory	Purpose
cron.php	Drupal uses this to manage recurring tasks—checking for module or core updates, for example. This file is updated through the administrative menu, the site building item, and then the Cron option.
includes	The includes directory contains many PHP include files that Drupal uses (to handle, among other things, authorization, menus, and updating Drupal's core and modules).
misc	The misc directory contains various files (images, JavaScript scripts, style sheets).
modules	This directory is where Drupal's core modules are located. Contributed modules should be added to the sites/all/modules directory (see the "sites" entry).
profiles	These are used during installation. A profile defines the kind of Drupal site you are creating and is, essentially, a preset bundle of modules and themes. Broadly speaking, when you install Drupal, you would select the "Standard" profile.
robots.txt	This file instructs web indexers (Google, Bing, etc.) what they are allowed to index and what they are forbidden, along with giving you as webmaster control over how often these indexing bots visit your site. For more details,

	see http://www.w3.org/TR/html4/appendix/notes.html.
scripts	This directory contains the various scripts Drupal uses.
sites	Drupal allows you to run multiple websites, each with its own domain name and independent configuration and functionality from a single instance of the application. You could choose to run a separate site for each of several branch libraries, each with its own theme and functionality. Each of these independent sites would have its own directory here to place local themes and modules. Common themes and modules would go in the “all” subdirectory. It is a best practice to install contributed modules and themes in sites/all/modules and sites/all/themes rather than in the main modules and themes directory.
themes	Themes are the “skin” of the site—the graphic design and the layout. Drupal 7 has four core themes to choose from (Bartik, Garland, Stark, and Seven). The user community has contributed countless others, and you can design your own if you wish. Contributed modules should be placed in sites/all/themes, as described previously.

Depending on what you do with Drupal you may need to edit files in the file system. Many changes and updates to Drupal’s functionality that you will make are implemented through the administrative web interface. Configurations, for the most part, are stored in the database

and not on the file system. In some cases, Drupal creates files on the file system and stores them in the locations noted earlier. Functionality or theme changes are, generally speaking, made on the file system. Functionality is largely controlled in the modules directory, while the interface is largely controlled through files in the themes directory. As you build your site, you will likely spend the most time in these two parts of the site.

This is a good time to remind you of an important habit to establish immediately: practice "safe developing." It is foolhardy to make untested changes directly on your live production site. Setting up a development environment, a second Drupal installation where you customize modules, themes, and content types, is wise. Having a development environment will allow you to experiment and work with your site out of the public eye. Then, when you are ready to release new modules or features to the public, you simply copy the updated files to your production site. How you copy files from your development environment to the production environment depends somewhat on the system you are using (Linux/Unix, Windows, etc.). Assuming you are running in either a LAMP or WAMP environment, you can use the "scp" command to copy individual files or entire directories between servers.

Another approach is to use some form of version control software for this purpose. Popular version control tools are Subversion (SVN) and Git. (Version control tools are beyond the scope of this book; please see "Recommended Reading" for resources to help you learn more.) No development project should exist without reliable and frequent backups so that even if version control fails you, you are protected against larger disasters such as power surges, storage failures, and hardware damage.

Drupal's strength is its "extensibility"—it is designed to allow users to augment its functionality by adding modular chunks of code. These additional modules provide specific functionality as needed for a particular Drupal site. These modules, once created, can be easily shared and installed in other Drupal sites. There is a large subset of the user community that is also contributing modules for all to benefit

from. It is, however, important to note that modules are dependent on the version of Drupal that is installed. In most cases, a module that works with Drupal 6 will not work, unaltered, with Drupal 7. This is the case because the modules exchange data with the core of Drupal, and a new version of Drupal will have new or altered ways of accomplishing similar tasks. It's as if you, as an international traveler, went to different countries speaking strikingly different dialects of the same language: you would not be able to communicate fluently with native speakers of one dialect even if you knew the other without changing, even if slightly, your vocabulary and grammar.

Now that we have established a basic understanding of what Drupal 7 comes with and how things are organized, we will build on this foundation. In the following sections, we will explain how to extend Drupal in these two important ways: functionality (using modules) and interface (using themes).

• Install Modules

Do you need a way to post an updated webpage to Twitter? Do you want to include Google Analytics on every webpage? Do you need an event calendar that exports events in standard calendar formats? Drupal modules allow site administrators to add new functionality, such as the previous examples, to a site. Chances are there are already modules for these functions. What if you wanted to create some functionality specific to your library—for example, give your patrons a way to look up all the books on their catalog on-hold list to see which ones are available at Amazon.com and let them buy them (perhaps even giving you a referral fee along the way)? You can even create modules from scratch. We'll talk more about custom module development toward the end of this chapter.

Finding modules is straightforward: they are listed on the Drupal website at http://drupal.org/project/modules. You filter the 9,400 modules by category, Drupal version, title, author, and so forth, and can search by keyword. When you have identified a module that meets your needs, you:

1. add it to your Drupal installation and
2. activate it in the administrative interface.

"Install from a URL" and "Upload a module or theme to install." We'll talk about each of these options.

Installing Modules from a URL

The easiest way to install a module is to simply copy the URL of the module you've found on the Drupal site into the Modules webpage in Drupal's administrative interface and click "Install." Drupal will then download the file and install it in the appropriate directory, making it available to a site administrator to activate and configure. You activate and configure modules through a webpage in Drupal's administrative interface.

When a module is contributed to Drupal.org, it has been thoroughly reviewed and added to the public repository by trusted members of the Drupal developer community. While there is no guarantee the module you download will be completely bug free, you can be confident that it is not malicious software. Modules you download from other sources should generally be viewed with some skepticism, as the module will not have gone through this peer review process for security and stability. We will talk about contributing your own modules to the Drupal community later in the chapter.

In some cases you may want to download the module yourself and make changes to it or simply review the code before you install it. Or, you might have written your own module and want to install it in your production version of Drupal. When this is your preference, you can download the file to your computer, perform whatever steps you would like, and then compress it again for installation. The second option on the Add Modules page in the administrative interface is the one you would then use—"Upload a module or theme to install."

Installing Modules by File Upload

You may want to have more control over the code you install on your development environment or simply want to know exactly what is

being placed in which directories. If so, you can install modules from the command line, bypassing the installation interface.

To do a completely manual installation, you can either use your web browser to download the module to your desktop and then upload it to your web server or, if you have a utility on your server such as Wget (http://www.gnu.org/s/wget/) or cURL (http://curl.haxx.se/), download the module directly to your modules directory from the command line. Modules are compressed as either .zip or tar.gz files, so, once you have downloaded the compressed file, expand it using the appropriate method. You now have a new directory in your modules directory that contains the module you just expanded. When you load the Modules page, Drupal scans the modules directory for modules and presents them in a list.

Activating a New Module

Once the module is installed in Drupal's modules directory, you need to turn it on and, depending on the module, set any necessary preferences. Manually activating a module is good security—a safeguard against someone installing a potentially buggy module by mistake without the Drupal administrator's knowledge. Incidentally, this is a good reason to make sure the user log-ins to the machines on which Drupal runs are different from the log-ins to the administrative interfaces—someone with access to one will not necessarily have access to the other.

To activate a module, go to your site's module interface in your web browser (e.g., http://library.org/admin/modules) and log in. Drupal groups modules by kind. First are all the core modules—the ones that come with a clean Drupal installation. All of these will be active by default; they are shown in Figure 5.3.

Modules | Library

http://www.library.org/admin/modules

Dashboard Content Structure Appearance People Modules Configuration Reports Help Hello site-owner Log out

Add content Find content Modules Edit shortcuts

CORE

ENABLED	NAME	VERSION	DESCRIPTION	OPERATIONS
☐	Aggregator	7.0	Aggregates syndicated content (RSS, RDF, and Atom feeds).	
☑	Block	7.0	Controls the visual building blocks a page is constructed with. Blocks are boxes of content rendered into an area, or region, of a web page. Required by: Dashboard (enabled)	Help Permissions Configure
☐	Blog	7.0	Enables multi-user blogs.	
☐	Book	7.0	Allows users to create and organize related content in an outline.	
☑	Color	7.0	Allows administrators to change the color scheme of compatible themes. Required by: Stylizer (disabled)	Help
☑	Comment	7.0	Allows users to comment on and discuss published content. Requires: Text (enabled), Field (enabled), Field SQL storage (enabled) Required by: Forum (disabled), Tracker (disabled)	Help Permissions Configure
☐	Contact	7.0	Enables the use of both personal and site-wide contact forms.	
☐	Content translation	7.0	Allows content to be translated into different languages. Requires: Locale (enabled)	
☑	Contextual links	7.0	Provides contextual links to perform actions related to elements on a page.	Help Permissions
☑	Dashboard	7.0	Provides a dashboard page in the administrative interface for organizing administrative tasks and tracking information within your site. Requires: Block (enabled)	Help Permissions Configure
☑	Database logging	7.0	Logs and records system events to the database.	Help
☑	Field	7.0	Field API to add fields to entities like nodes and users. Requires: Field SQL storage (enabled), Field (enabled) Required by: Drupal, Field SQL storage (enabled), Field (enabled), Text (enabled), Comment (enabled), Options (enabled), Taxonomy (enabled), Community Tags (enabled), Field UI (enabled), File (enabled), Forum (disabled), Image (enabled), List (enabled), Number (enabled), Tracker (disabled)	Help
☑	Field SQL storage	7.0	Stores field data in an SQL database. Requires: Field (enabled), Field SQL storage (enabled) Required by: Drupal, Field SQL storage (enabled), Field (enabled), Text (enabled), Comment (enabled), Options (enabled), Taxonomy (enabled), Community Tags (enabled), Field UI (enabled), File (enabled), Forum (disabled), Image (enabled), List (enabled), Number (enabled), Tracker (disabled)	Help
☑	Field UI	7.0	User interface for the Field API. Requires: Field (enabled), Field SQL storage (enabled)	Help

Figure 5.3: Screenshot of Module List with Module Name, Settings, and Other Functions

After the list of core modules you will see displayed those modules that have been added by the local administrator, listed in alphabetical order by module name. Each module has its own section. Some modules are actually themselves bundles of functionality that can be independently activated. For example, the Views module contains five separate components. Each component lists its dependencies,

the other modules it needs in order to function. You can turn on the individual pieces that you need.

Find the module you added and check the box to the left of its name. Then click the “Save configuration” button at the bottom of the page. When you see the confirmation message that your configuration has been saved, you’re done—you have added new functionality to your site.

For all but the simplest modules, you will probably need to configure the module to work properly in your environment. These configurations are available through the “Settings” link in the list of modules, to the right of the module name. Obviously, the specific configuration for a module depends on the module itself. The documentation for that module, generally found on the Drupal site, is a great place to start learning how to make the module work.

• Install Themes

Drupal themes are installed and managed similarly to modules: find the theme you want to install (on Drupal.org, elsewhere, or after having developed it yourself). Then, go to the Themes page in the administrative interface (http://library.org/admin/appearance) and click “Install new theme.”

As with modules, you have the options of installing themes directly from the Internet or from a file on your computer or installing the theme files manually on the server. Regardless of how the theme gets into Drupal’s themes directory, you need to manually activate it by going to the themes section of the administrative interface and selecting it.

The themes, with a thumbnail picture of the basic layout, are listed. These correspond to the themes installed in the themes directory. You can quickly change themes by simply selecting the new theme and clicking save. The change is instant; you can see the new theme on your site the next time you reload the page. However, and this is a big “however,” applying a new theme in practice will not be this straightforward. Themes are not as easy to implement as modules.

The difference is that modules simply provide functionality, while themes define the output and the look and feel of the site. The module may work fine, but if the page layout does not accommodate the functionality of the site, users are left without the information they need.

Drupal organizes pages in a modular way—much as it does everything modularly. A theme not only defines the colors and text styles of the site, it also defines blocks on the page that have specific purposes. One or more blocks grouped together make a region. There could be a block for your site's logo, a block for main navigation, a block for content, another block for standard footer information, and so forth. These blocks are consistent across your entire site. The logo and navigation blocks could be grouped together into a region (see Figure 5.4). Some kinds of pages may have special blocks—for search results, or an online chat box, or library hours. If you change the theme of the site, you need to configure the new theme to have the same regions and block names—even if they are in different places on the page—as the old site. Alternately, you can update the modules that provide the functionality in question to put their content in a new block.

In the basic Bartik theme that comes with Drupal 7, site navigation is across the top of the page and the site log-in is on the left. These areas are defined as "blocks" in the theme, and various modules are configured to display their content in specific blocks. When you install a new theme, with differently named and organized blocks, your modules may not know where to put their content or may put it in unintended places.

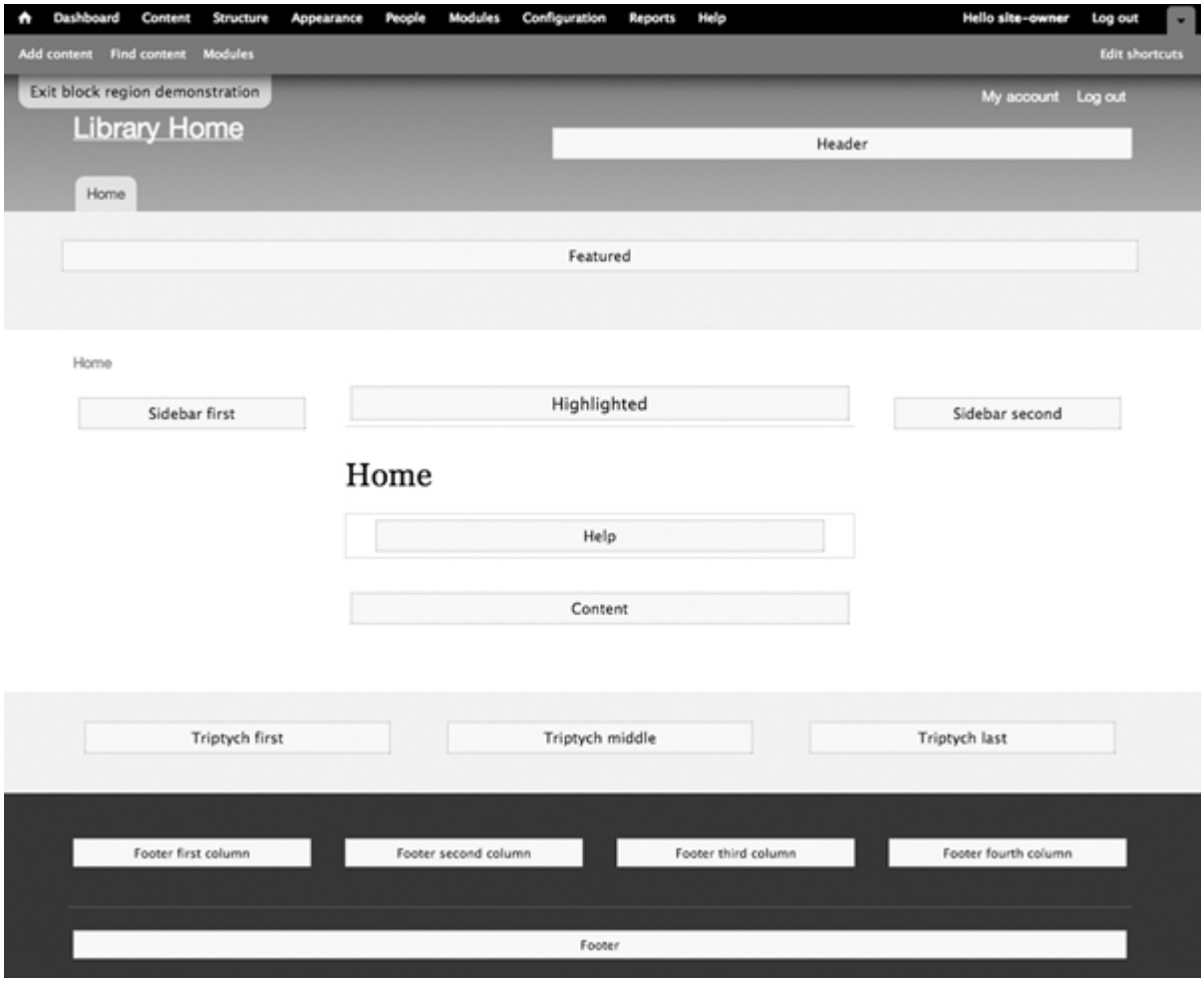

Figure 5.4: Regions and Blocks in Drupal 7's Bartik Theme

If you change themes, you will need to spend some significant time updating the site to match. This involves renaming blocks so that areas with similar functionality in the old and new themes are given the same name. You may need to adjust template files to accommodate changed layouts (see the Style Your Site by Going Beyond the Standard Themes section, pp. 72–74). You may also need to update any custom code you have created.

Much as when you may have considered a site redesign before moving to a content management system, redesigning the layouts of the site with a content management system can be complicated. This is not to say that once you have picked a design you are locked into it. When contemplating a redesign down the road, the effort is

invested in building the templates rather than in migrating the content from the old style to the new. Once the content and functionality are in place, they can be more or less left alone and effort can be focused on design for the next iteration.

Keeping Modules and Themes Up-to-Date

Just as Drupal makes installing and managing modules straightforward, there is a similarly easy way to keep your Drupal installation up-to-date. Within the reports section of the administrative interface is an "Available updates" report. This report checks with Drupal.org to see which of the core and contributed modules and themes installed on your site have newer versions. In the example shown in Figure 5.5, Drupal core is up-to-date but one contributed module (Community Tags) has a newer version available. While Drupal provides links directly to the module's download page (the newly released version number), there is also a link to "Download release notes." It is always a good idea to review the release notes before installing an update so that you know what is being fixed.

If you are a cautious Drupal administrator, you will probably not download and install a Drupal module the instant it is available. Despite the collaborative testing regimen open source software projects follow, it is possible that minor (or even major) incompatibilities between an updated module and the current one exist. Unless the release is truly critical (meaning that the update fixes a serious bug that might lead to lost data or a hacked site), wait a while. And, as with everything else in Drupal, it is always prudent to test the new version of the module in your development environment first.

There are a couple additional caveats to using the updates report to keep your modules and themes at the latest release:

- Any updates to your installed themes will also be listed here. As with installing themes, applying updates requires a bit more caution and testing. Changes to themes can impair use of your entire site and so should be approached with caution.

- If you have customized either modules or themes, you may find using the automated update mechanism to install a new version will make your custom code vanish.

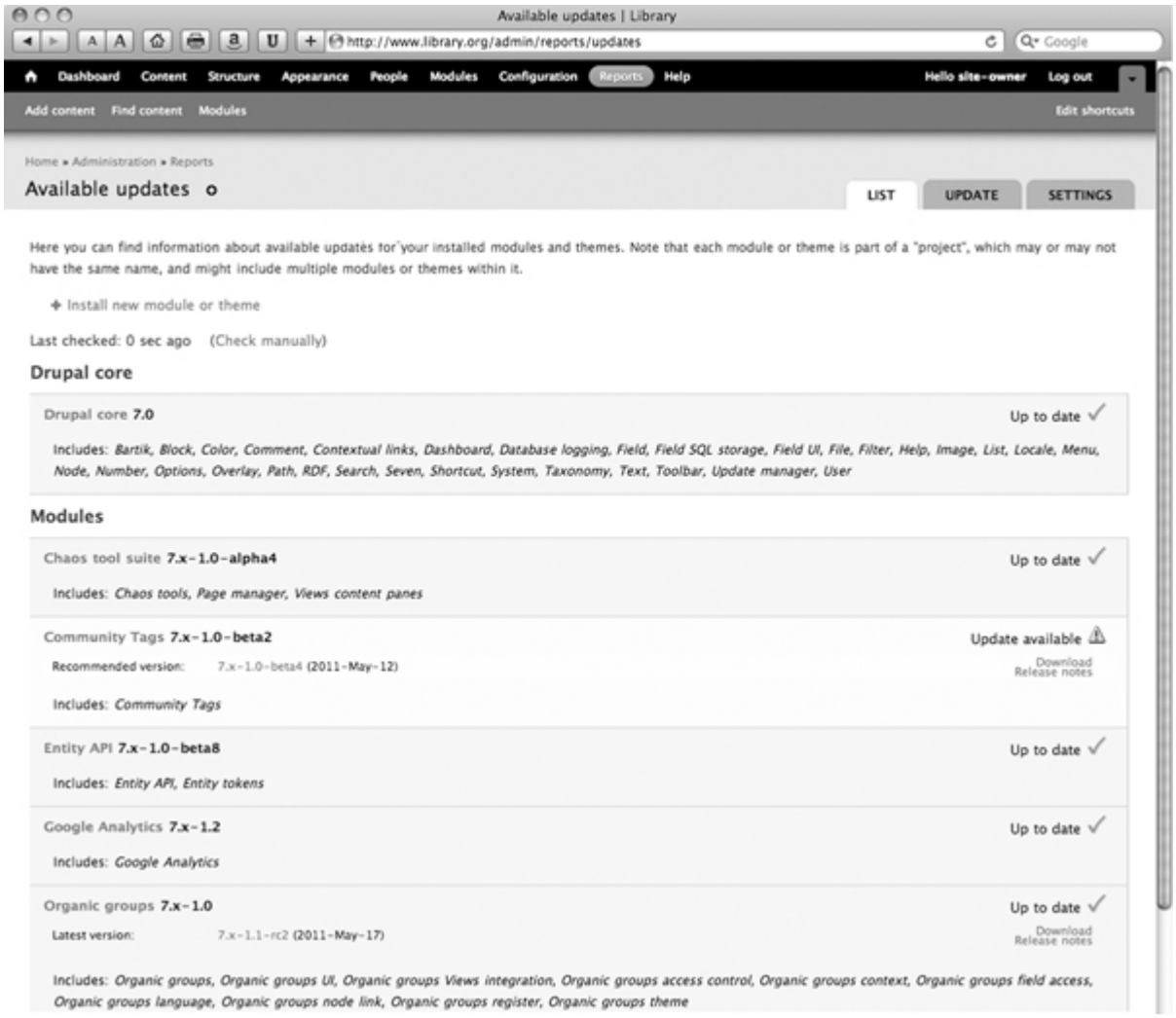

Figure 5.5: Screenshot of the Available Updates Report Page in Drupal's Administrative Interface

If you set up cron jobs when you installed Drupal (see the previous Install Drupal section, pp. 40–44), Drupal will check for updates periodically and alert you to any updates that are available on this reports page.

• Configure a Basic Library Website with Basic Modules and Tools

After you have set up your operating environment, installed Drupal, and completed the setup process, you have a Drupal-powered website. Your Drupal website consists of an administrative panel and one page of content—a default front page with only a log-in link, a search box, and your site's title in the upper left (along with the cheerful Drupal logo).

Because this basic cookie-cutter site probably does not meet your library's needs, we will now talk about using specific modules and tools to make your site feel like yours. We'll discuss functionality (modules) first and then interface design ("theming" in Drupal terms), though it is wise to read both sections first before proceeding.

Of course, this introductory book will only scratch the surface of what is possible within Drupal. What we cover here will give you a basic website that will meet basic needs. See "Recommended Reading" for books devoted to the details of module development, Drupal administration, and the like.

Using the Content Creation Kit

One of the most powerful modules in Drupal 6, the Content Creation Kit (known as CCK), has been built into Drupal 7 and is now part of the core functionality. CCK offered a flexible and intuitive way to extend existing content types or to create wholly new ones. Whereas Drupal originally came with a fixed set of content types, CCK allowed site administrators to add new fields to existing types or to create wholly new content types through a web interface. CCK also allowed fields from one content type to be shared with other content types so that a field that held the same kind of information, regardless of the content type it resided in, could be used in multiple places without needing to duplicate it.

The CCK module also enabled a simple interface to specify the kind of content that is expected in a given field (e.g., a character limit)

or to add fields of a certain kind (date or time, e-mail address, etc.). CCK provided an easy way for site administrators, and not just developers, to make and extend content types. A content author would then fill in the forms, and the result would be a standard page for that particular kind of content. CCK became such a critical piece of functionality for Drupal administrators that it was incorporated into Drupal 7 and is now part of Drupal's basic functionality.

Creating a Content Type

We'll create a basic content type for describing a library's branches. We'll make a very simple example with the following set of fields:

- Name of library branch
- Description of branch
- Contact information
    - Street address
    - Phone number
    - Director's name

Log in to the Drupal administrative site by going to http://library.org/admin. Content types are defined through the Structure menu, so click on that (see Figure 5.6).

Then, click on the "Content types" link. Here you will see the basic content types Drupal offers by default (article and basic page). We need to add an additional content type, "Library Branch," which we do by clicking "Add content type."

There are two required fields: name and title. Each content type in your website must have a unique name. You can use spaces, mixed capitalization, and so forth—this is the name that you and your content authors will see. Drupal will translate this into an all-lowercase, machine-readable version for its own use. For example, if you gave this content type the (awkward) name "System's Branch Libraries," Drupal would create the machine name "system_s_branch_libraries." (You can edit the machine name as long as you keep it free of uppercase letters, punctuation, and spaces. The only

time you would need to interact with the machine name for your content type is if you create a custom theme for the output. See the Style Your Site by Going Beyond the Standard Themes section, pp. 72–74.) For this example, we'll call this content type a "Library Branch," which Drupal will give the name "library_branch."

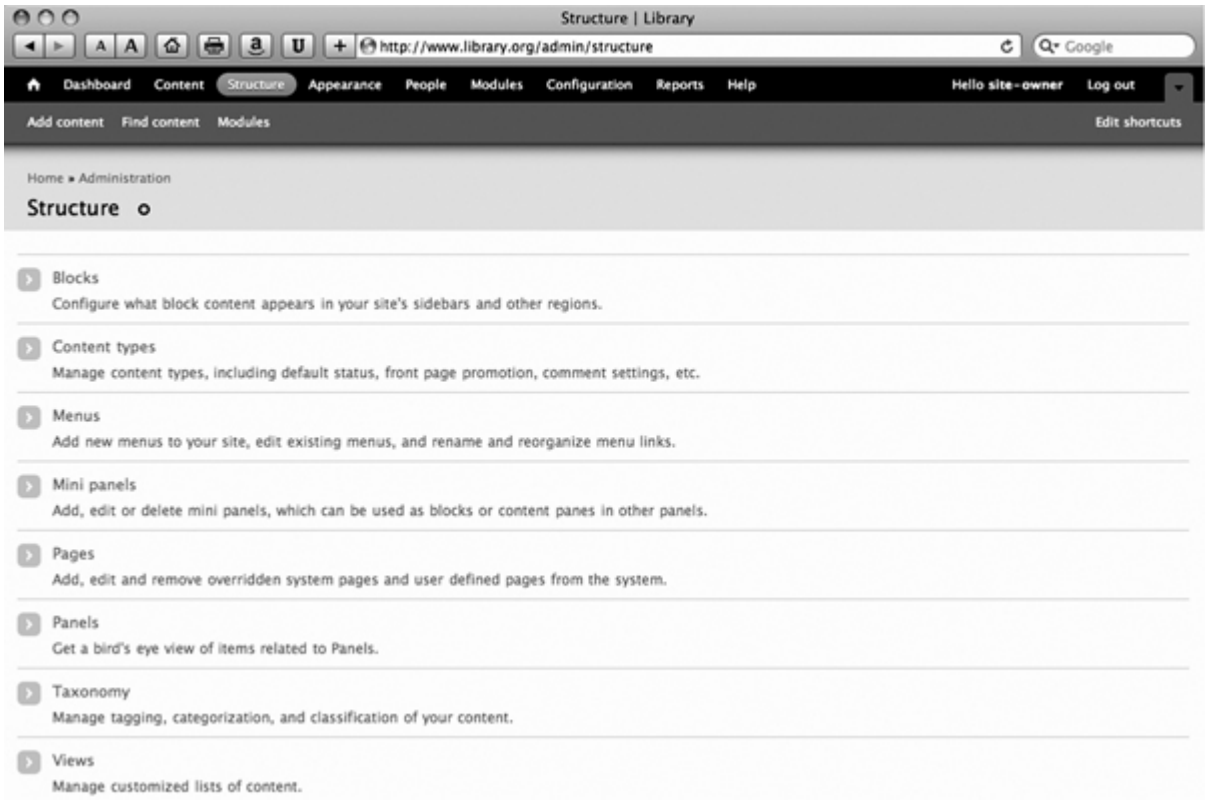

Figure 5.6: Screenshot of the Structure Menu in Drupal's Administrative Interface

Drupal encourages you to add a description for the content type. This field can be edited later and can be thought of as a scope note for the content type—it provides guidance to content authors when they are trying to decide which content type fits their needs best.

The title is the name of the field that the page's title goes in. This name defaults to "Title." While it is fine to leave it as such, you might find it more intuitive to give this field a more representative name, such as "Branch Name" (see Figure 5.7).

There are two more fields: “Preview before submitting” and “Explanation or submission guidelines.” “Preview before submitting” gives the site administrator the option of forcing page authors to see a preview of the content before they save it. You might want to set this to “required” if the content is high-visibility—such as the main page of your site—and you want to make sure the authors haven’t inadvertently created a display problem with their changes. This setting defaults to “optional.” “Explanation or submission guidelines” is text that is displayed to content authors above the form. It might give them specific instructions for filling in the form, explain in more detail what this content type is for, or serve some other function.

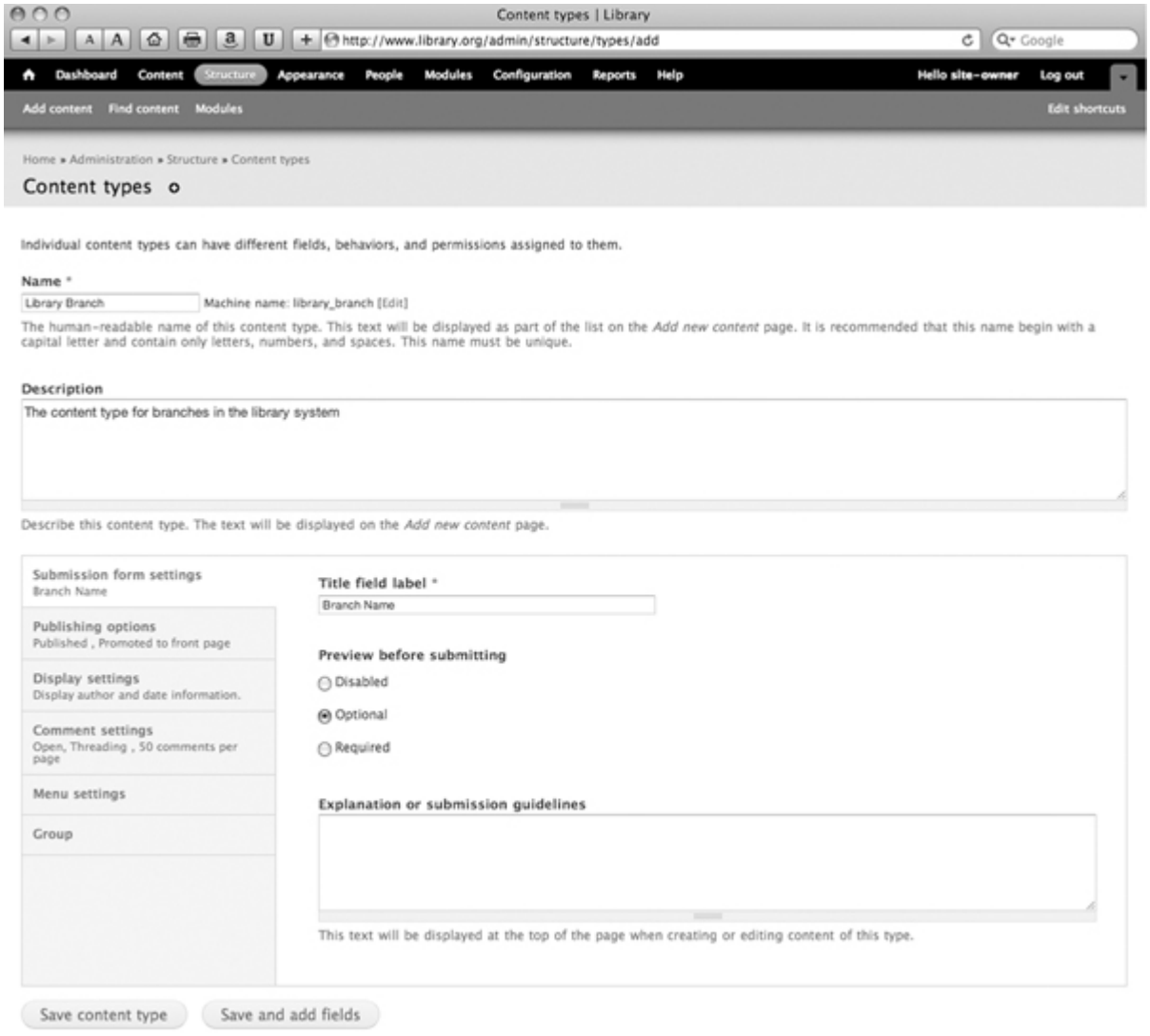

Figure 5.7: Example of a Completed Add Content Type Screen in Drupal's Administrative Interface

Finally, there are two buttons: "Save content type" and "Save and add fields." If this content type only needed a title and text, you could click the first button. But we want to add additional fields for the branch library (contact information and a description of the services available here), so we'll click "Save and add fields."

Now that you've saved the basic content type and started to add more fields, you see a summary of the fields in the content type (see Figure 5.8). Every node has a body—this is where, by default, Drupal expects you to put your content. You can use it, or you can create other fields into which to place the information that will appear on this page. You can also delete it if you like, but it is important to remember that if you delete a field that has information in it, not only is the field gone from the database, but all the information that you added to that field, on any node, is gone as well. This can be a catastrophe if you are using a default field like body, which appears in every content type. Deleting it from one content type can inadvertently delete information across your entire site. Two cautions: one, be careful about deleting fields, and, two, back up your site and your database frequently.

Dashboard Content Structure Appearance People Modules Configuration Reports Help Hello site-owner Log out

Add content Find content Modules Edit shortcuts

Home » Administration » Structure » Content types » Library Branch

Library Branch

EDIT | MANAGE FIELDS | MANAGE DISPLAY | COMMENT FIELDS | COMMENT DISPLAY

Hide row weights

LABEL	WEIGHT	PARENT	NAME	FIELD	WIDGET	OPERATIONS
Branch Name	-5	- None -	title	Node module element		
Body	-4	- None -	body	Long text and summary	Text area with a summary	edit delete
Street Address	-3	- None -	field_street_address	Text	Text field	edit delete
Phone Number	-2	- None -	field_phone_number	Text	Text field	edit delete
Director's Name	-1	- None -	field_branch_director	Text	Text field	edit delete
Group roles and permissions	0	- None -	og_roles_permissions	Boolean	Check boxes/radio buttons	edit delete
tags	2	- None -	field_tags	Term reference	Autocomplete term widget (tagging)	edit delete
Add new field						
Label	3	- None -	field_ Field name (a-z, 0-9, _)	- Select a field type - Type of data to store.	- Select a widget - Form element to edit the data.	
Add existing field						
Label	4	- None -	- Select an existing field - Field to share		- Select a widget - Form element to edit the data.	

Save

Figure 5.8: Screenshot of Manage Fields Screen in Drupal's Administrative Interface after Clicking "Save and Add Fields"

For this example, you can use the body field as the place to hold the description of the branch. We still need to add fields for contact information. You will notice two options beneath the existing fields: "Add a new field" and "Add existing field." Perhaps you have already created another content type that has a street address field in it. You could put all your addresses together into one, commonly named field. In this case, you would select "Add existing field" and select the existing address field. As a rule, this is a good practice when fields have identical functions across content types. However, remember the admonition from the previous paragraph: if you think you might

want to rename it or delete it, be sure you know the scope of the action you're about to take.

For this example, we'll create a new field, "street address." Unlike on the previous screen, you get to give the machine name of the field. You are still restricted to the lowercase alphabet, the numbers 0–9, and underscores. While this name is used primarily by Drupal, it is always helpful for future developers—including you—if this name bears some resemblance to the displayed field name. We'll call it "street_address." We need to define the kind of data to store—this establishes the database field type. We'll make this one a text field. This decision, in turn, forces the widget entry to be the same: text field. The widget is how a content author enters the information. The options vary depending on the kind of data. If we had picked List, the options in widget would be radio buttons or checkboxes from which the author can pick one, or more than one, selection.

Now, click "Save" to create this particular field. When you do this, you are given a chance to enter any field-specific criteria. For a text field, you can specify the maximum length of the text box. For a list, you would enter all the possible valid options. A text field defaults to 255 characters; if this seems sufficient for all of your branch mailing addresses, leave it as such. Or, you can make it larger or smaller. Then, click "Save field settings."

The final step is to make any special settings for this field when it is used in the context of the Branch Library content type. As noted earlier, the street address field can be added to any Drupal content type. But you can customize the authoring interface for it when it is being edited on a Branch Library node. By clicking the "edit" link to the far right of any field name, you can change settings, such as adding a label, making the field required (by default, it is not), defining how big the editing window for this field is, and setting any HTML filters (do you want authors to enter plain or formatted text?). For our example, we will make it required (all branches need a street address or patrons will not be able to find the branch), leave it plain text (HTML does not make sense for a mailing address), and leave the

default value blank (having a default street address for all library branches does not make much sense).

Further down this editing screen, you can also change the default settings in case you've changed your mind or thought of something new. You can always return to this screen when you are creating a new branch by going to the administrative interface, clicking "Structure," then "Content types," selecting this content type, and then clicking the "Edit" tab.

Now you can repeat the process for the other two text fields—phone number and director's name. When you've done that, you have created your first custom content type and could begin creating nodes for each of your library's branches. This would be a pretty simple site, but it's a solid start. From here, creating additional content types to match the different sorts of content your library's site will require is straightforward.

Using Views

Content types are a solid building block for developing your site. However, one of the true powers of a content management system is that it is possible to reuse not just entire pages, but individual fields, from one kind of content to another. For example, if you wanted to create a directory page that listed all your branches and service desks—assuming you had built content types for branches and service desks—you might be tempted to create such a page manually and keep it up-to-date. It would be much more convenient if only there were a way to build some kind of custom report and display only the selected fields from these content types.

Views is that custom report tool for Drupal. Views exposes the intricate data structures of your content and allows you to select individual items, organize them (by name, modified date, etc.), and display them in tabular or page format. Even though it is hard to imagine building a site without Views, it has not been added to Drupal's core code and is still a module to download.

Creating a View

We'll go through a simple example with Views, pulling out the names and phone numbers from the library branch content type and displaying them on a new page. The first step is to create a new view, which is accomplished in the administrative interface by selecting "Structure" and then "Views" (see Figure 5.9).

From the list of existing views, we want to add a new one. To do this, click the "Add view" screen. Then, give the view a name (human-readable, to appear in the listing of views), a description (to remind you of its function later on), and any optional tags—keywords to help you find it again. We'll call this view "Director List" and give it a description, "List of branch directors." You also need to pick a "View type." For this view we are working with content from nodes so we will select "Node" and then click "Next." There are several view types. Each has access to a different part of Drupal's database. Once we have picked a view type we get to the page where we configure the view to display just the information we want.

The Views module offers a large number of ways to select, organize, and display the content you want. This simple example will get you started; much more detailed instructions can be found on the Views project page in Drupal. First, we give the View a name and (optionally) a description. The description is shown in the administrative interface and tells Drupal administrators what the view is for. It is good practice to use the description field. We will create a view with the name "Director List" and a description "List of branch directors." Now, select the content type we want to draw from. In this example, we're working from the Library Branch content type. We want the result of this view to be a table, so select "Table" from the display format menu (see Figure 5.10). Finally, select the "Continue & edit" button to specify the fields that will appear in the view.

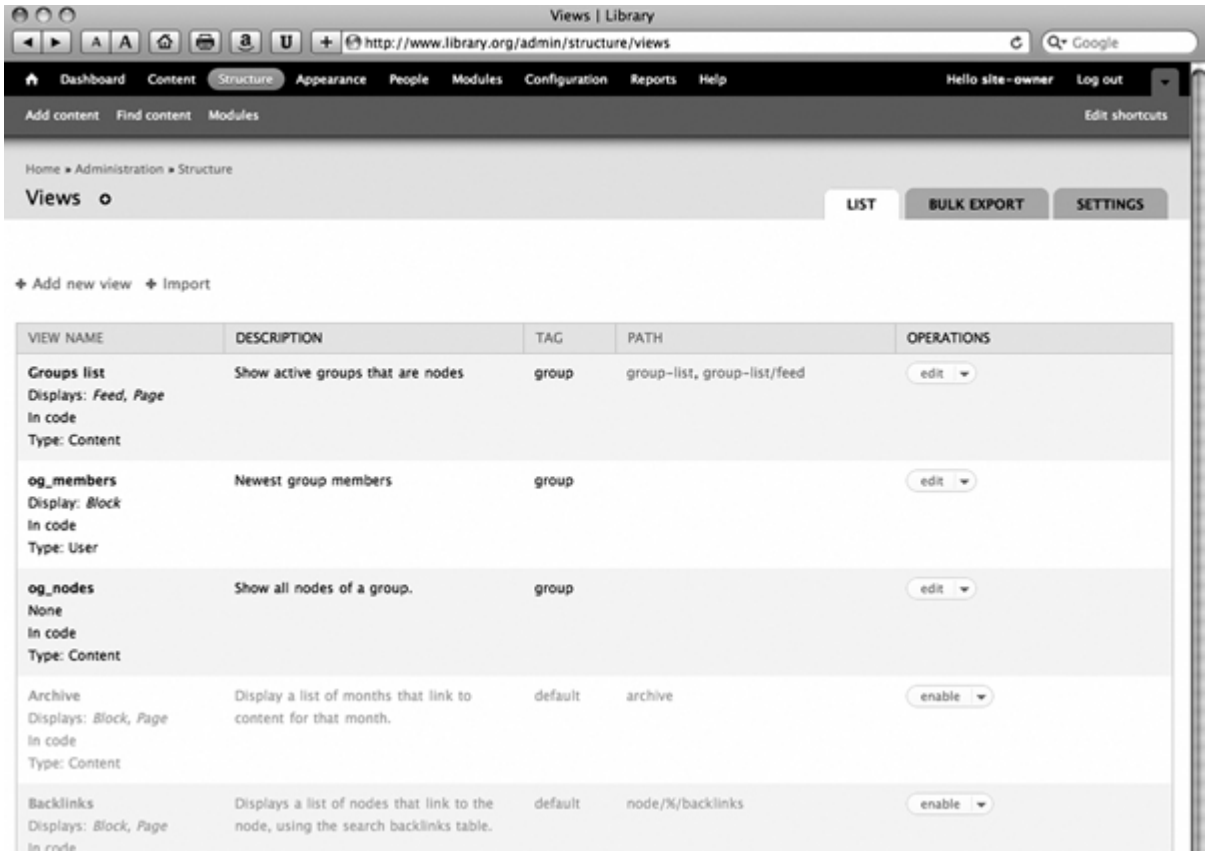

Figure 5.9: Screenshot of Views Listing in Drupal's Administrative Interface with "Add a New View" Selected

In the "Filter Criteria" section of this page, click the word "add." You will see a new section of choices appear below the list of options—these represent all the characteristics of Drupal content that you can work with. We will start by selecting "Content: Type" so that we can work with only content in the Library Branch content type, ignoring everything else. Check that box and click "Add and configure filter criteria."

Next we will specify the content type or types that we want to use. The filter defaults to "is one of," but you could equally well use all content types except for specified kinds. Then check the box next to "Library Branch" and click "Apply (all displays)."

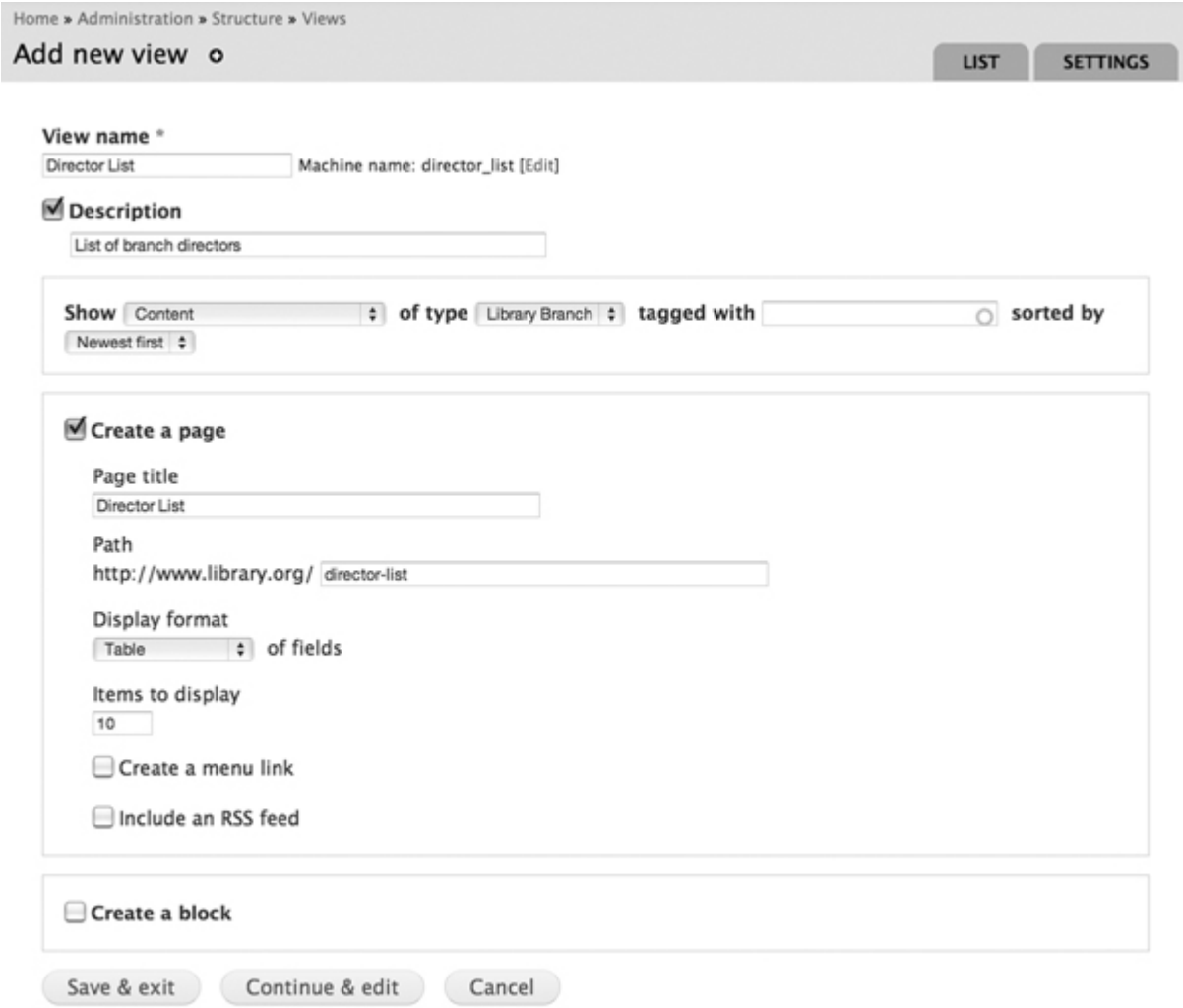

Figure 5.10: Screenshot of the View Configuration Screen in Drupal's Administrative Interface

Now we want to select fields from the Library Branch content type to display. Click the "add" link next to "Fields" in the list of options. Drupal will show you a list of all the fields available in the content types you selected in the previous step; each field lists the content types to which it belongs. We want to add three items from the Library Branch content type, one at a time: Content: Director's Name, Content: Title (the name of the branch), and Content: Phone Number. For each, select the appropriate checkbox and click "Add and configure fields."

Here is where we tell Views how to display the information contained in the field. You can add a label to each field, hide it from display, or apply a particular CSS style to the content of that field. The fields are presented in the order they are selected in the Fields menu, which defaults to alphabetical by name, so the order is:

1. Content: field_branch_director
2. Fields: field_phone_number
3. Node: Title Title

Director's Name," and the phone number last by giving that field a weight of "3." Click "Apply (all displays)."

The next step is to define the output format. We'll use a simple table (other preset layout options are available) sorted alphabetically by the name of the branch library. In the "Format" section of the Views page, click "Unformatted list." Select "Table" and click "Apply (all displays)." The next screen shows the display options. We want the list to sort by branch name, so check the radio button on the right side of the list, on the "Content: Title" line. Then click "Apply (all displays)."

You have now created a list of branches, directors, and phone numbers that will be updated automatically whenever information on the branch page is updated. Each branch title links to the branch's page (this is automatic, but can be disabled). Common views in a library might include lists of electronic resources, a staff directory, or list of events happening this week.

This view automatically gets its own URL, one that is based on the view name we started with. Since we called this view "Director List," the URL is automatically set as http://www.library.org/director-list. You can edit this name by clicking on the default title under the "Page Settings" heading and creating any URL you like.

The view you just created can also be embedded into other pages using Panels (which we'll talk about in the next section). With different displays, Views can exist in blocks, can create RSS feeds, and more. Our rudimentary example only scratches the surface of what Views

can do. Even though it is not part of the Drupal core code, it is one of the most powerful tools in the Drupal suite. The more content you have in your site, and the more specific content types, the greater the power of Views to bring information from different places together, dynamically, without needing to reenter it. If the information is in Drupal once, it can be reused anywhere, multiple times.

Using Panels

What Views is to accessing your content out of context, Panels is to presenting your content in context. Panels is a powerful Drupal module that gives administrators the flexibility to display multiple nodes, blocks, and views within a single page. The Panels module opens up almost endless possibilities to display information from your site. For example, you could take the display of library directors and phone numbers you created in the previous section and embed it within another webpage. Both the full directory and the embedded list would be updated, automatically, whenever you edited the branch information for a particular library.

Panels has a number of page layouts you can pick from. Panel layouts include fixed-width one-, two-, and three-column displays; two- and three-column displays with headers and footers; and alternating rows of one- and two-column displays. (You can also make your own custom layouts if the existing ones do not meet your needs.)

When you create a panel, you then select the specific content that you want to appear in each section. This can be entered directly into the panel, drawn from the body of an existing node, or drawn from a view you have created. We will walk through a very simple panel construction using the branch information view we created in the previous section and text we enter into an introductory paragraph. For this example, we'll create a two-column panel with a header and a footer.

Start by going into Drupal's administration menu and selecting "Structure" and then "Panels." We'll pick a "Panel Node" for this

example, so click on that. On the next page, there is a menu from which you can pick a style of page and a corresponding graphic for each page type that fits the category. Pick “Columns: 2” to see the list of two-column layouts, and then pick “two column stacked.” You’ll next be asked to give this panel a title and a “CSS ID” before continuing (see Figure 5.11). We’ll call this one “About the Library” and will leave the CSS ID field blank. This field gives the contents a specific ID that can be used to add custom styling to the text that appears in this panel.

Leave the other settings as is and click “Save” at the bottom of the page. You will be taken to a view of the panel as the public will see it. Right now, of course, it has no content, but you can see the authoring tabs just above it. We want to set the content for this panel, so click the “Panel content” tab. You’ll see four gray boxes, each with a label such as “Top” or “Left side.” To the left of each label is a light gray cogwheel icon. This icon is the way to get to the panel’s configuration settings (see Figure 5.12).

We’ll start by putting introductory text in the Top section. Click the cogwheel icon to the left of the word “Top” and select “Add content.” You will get a list of kinds of content you can use; for this section, we want to type in text directly, so pick the next-to-last item, “New custom content.”

Next, place the names of the directors in the left-hand column. Click the cogwheel to the left of “Left side,” and then click “Views.” A list of the views you have created appears; select “director_list” to pick the list we created in the previous section. Pick the page layout you want to use (select default unless you created a custom layout for this view just to be embedded on this page).

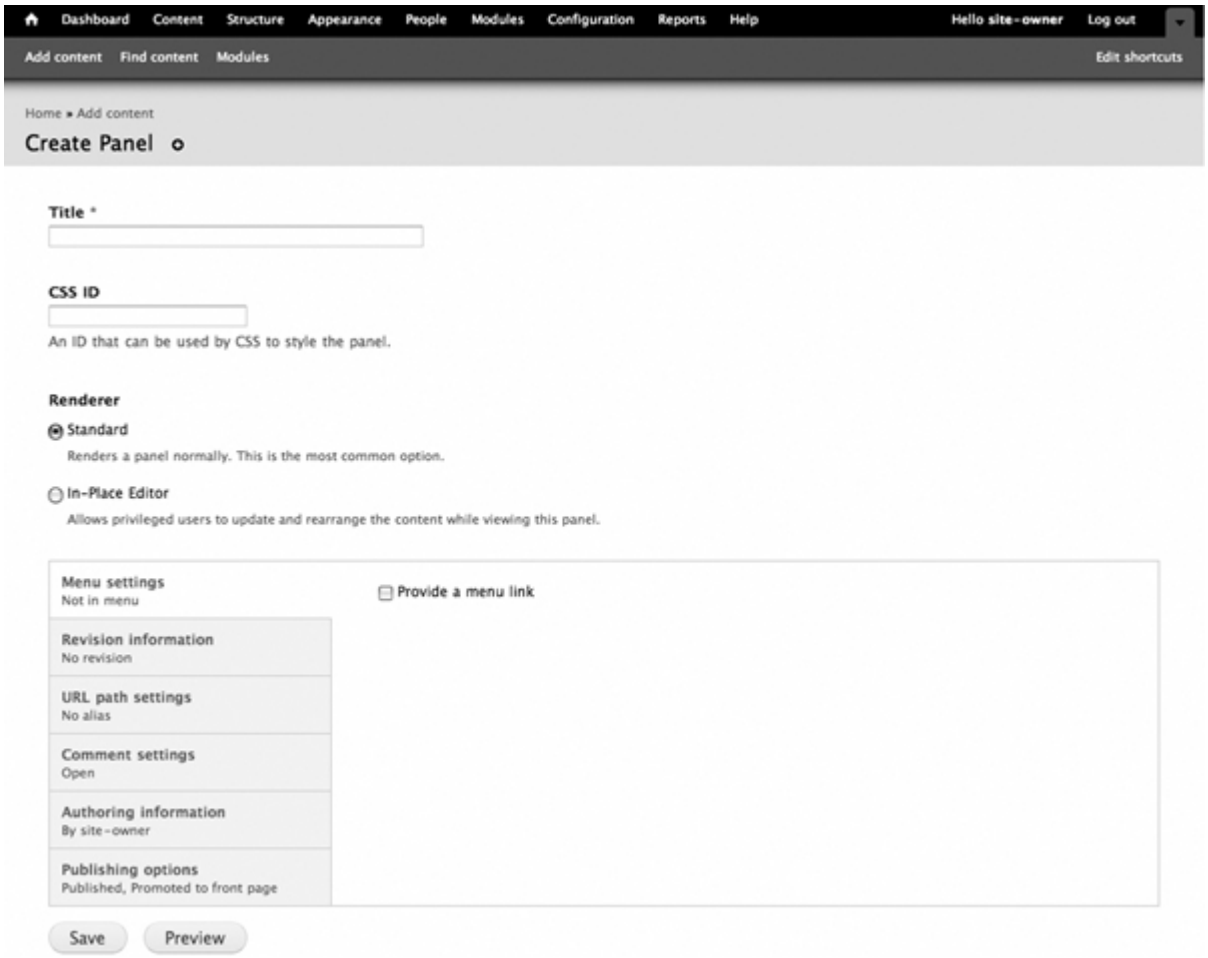

Figure 5.11: Screenshot of the Add Panel Screen in Drupal's Administrative Interface

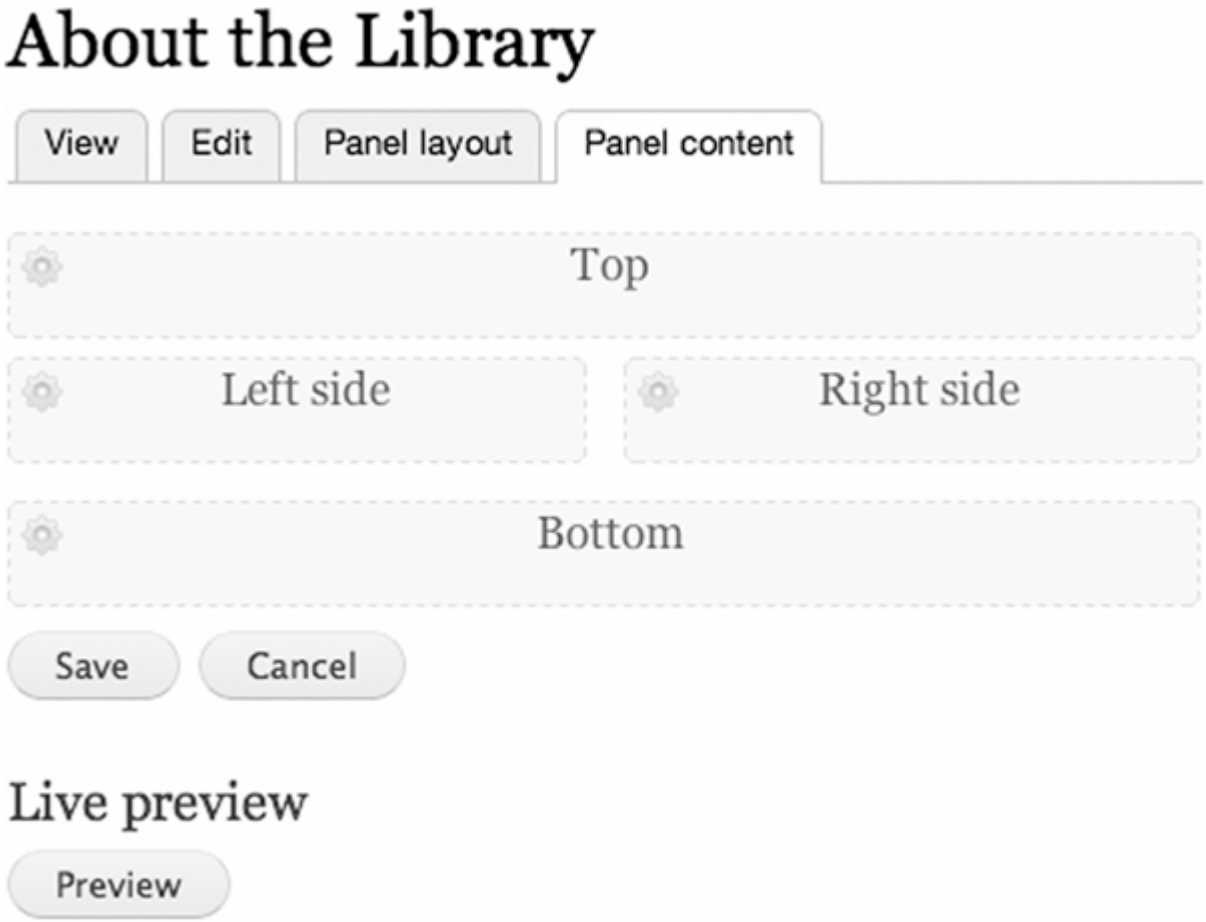

Figure 5.12: Screenshot of the Panel Content Tab with Nothing Configured

Now we will select content to place in the right column. We might have a list of libraries, or our site's navigation, or perhaps a photostream from the library's Flickr account. For this example, we'll add a list of recently modified pages on the site. Select the cogwheel to the left of "Right side," and click "Settings." Select "Widgets" and then "Recent content." You can give this panel its own title (such as "Recently Updated Pages") or leave the title blank. Then click "Finish." The panel defaults to the most recent ten updated pages; once you are back on the View tab, you can click the cogwheel to the right of "Recently Updated Pages" to adjust this setting, along with others. You can similarly add content to the "Bottom" section of the panel.

Panels can be used to create entire pages or to create smaller, reusable pieces that can be placed elsewhere in the site. “Mini panels” have the flexibility of a full panel but can, in turn, be inserted into another panel. They can also be used to fill in blocks on the site so that similar content, dynamically generated, can be placed across the site in a standard place. An example use of this functionality is creating a branch-specific view of today’s events that appears in a specific place on all pages within that branch’s set of pages but not on other branches’ pages. Similarly, you could use panels to place navigation menus for your site within the page, rather than across the top, as a way to keep sitewide navigation separate, visually, from subsite navigation.

Soliciting User Feedback

In addition to having comments placed in public view on the site, you can also actively solicit feedback through a form. The Webform module (http://drupal.org/project/webform) is built to handle precisely this; it is an easy-to-configure module that allows you to build forms and have the data users submit processed in some way. At a basic level, the webform could ask for a commenter’s name and e-mail address and give a text box for comments. You could extend this a bit by asking for additional information to classify the comment as a reference question, suggestion, complaint, compliment, and so forth, or to request additional ways to contact the person. You could route the e-mail the form generates to a different address depending on the kind of comment that was being made—so that suggestions, compliments, and complaints could go to the director while reference questions could go to the reference desk.

Once you have installed the Webform module (see the earlier section on installing modules), you create a new form by creating a new node of content type “Webform” through the Create Content menu. You are creating the page on which the form resides. Like all Drupal pages, it has a title and a body—the text you display to the user as an introduction or preface to the form (see Figure 5.13).

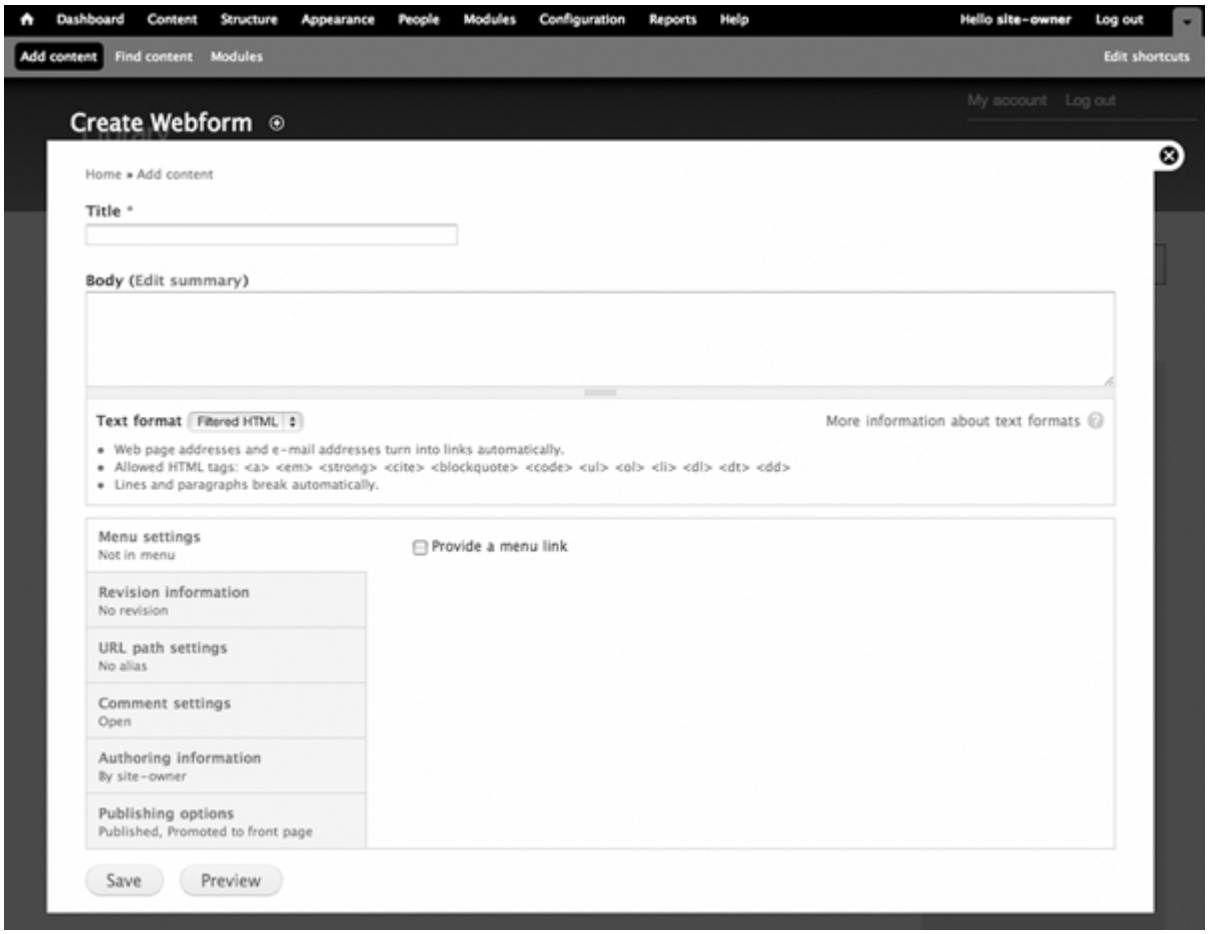

Figure 5.13: Screenshot of the Create Webform Page in Drupal's Administrative Interface

You start to create the form by adding fields of various kinds. For the basic form outlined earlier, you will need three fields: commenter's name, commenter's e-mail address, and a comment. Much as when you create a content type (described earlier under Content Creation Kit), you add fields one at a time. We'll create three fields: commenter name, commenter e-mail, and comment body. For each field, you give it a name and pick a type (text field, text area, select list, etc.) and then, on the following page, provide more details such as maximum number of characters, whether or not the field is mandatory, and what the label for the field should be. We now have a simple, three-field comment form set up (see Figure 5.14).

The next step is to set up the behavior of the form once the commenter fills it in and clicks "Submit." Switch to the "E-mails" tab.

First, enter the destination e-mail address for user feedback and click "Add." You can now configure custom subjects, a "from" address (using what the commenter entered or something else), and a "from" name (again, the commenter's or something else that you determine). You can then customize the resulting e-mail; each field the user filled in is available, plus you can add some "environment variables," such as the date and time, the user's IP address, the kind of computer and browser the user was using, and so forth. When you click "Save e-mail settings" you have created a working e-mail form. You can expand on this simple outline by creating more narrowly focused forms to gather information and direct it to the appropriate office or person.

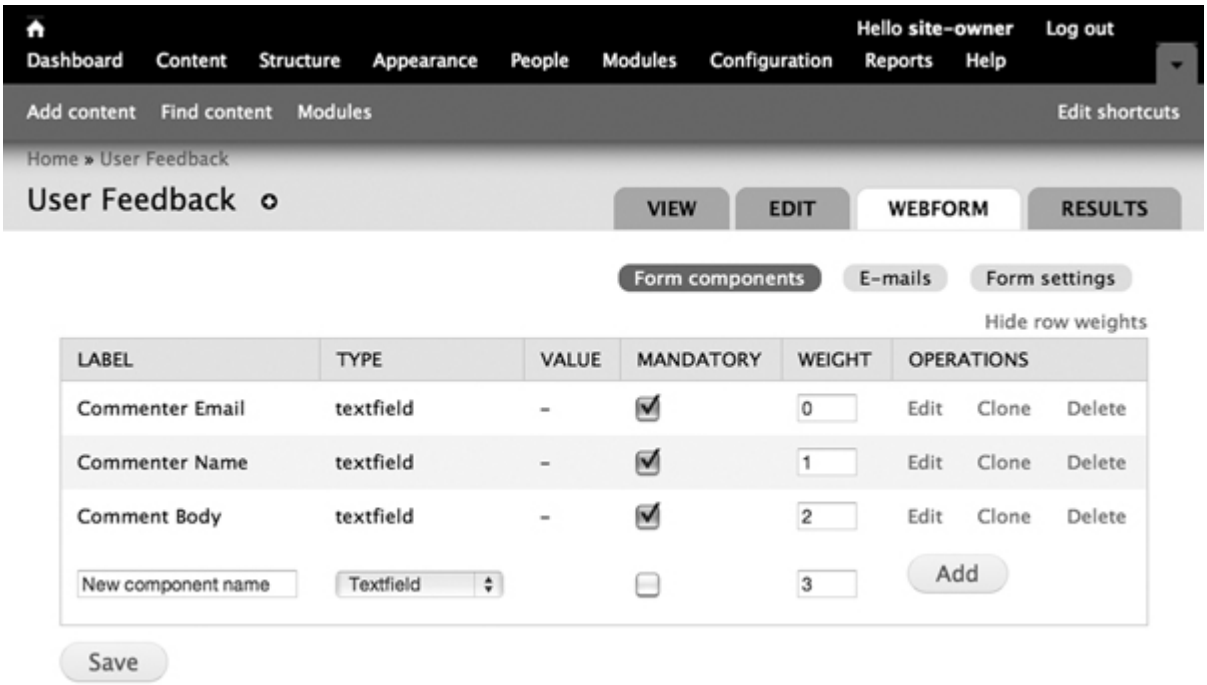

Figure 5.14: Screenshot of the Form Components Screen in the Create Webform Process with Three Fields Configured

Creating Organic Groups

In Drupal, loosely speaking, everything is created equal. Drupal itself provides little in the way of organizing content or users into coherent clusters. Creating a site that gives sets of content authors

responsibility for defined parts of the site (content and navigation, for example, in a particular set of pages), or allowing your patrons to belong to certain parts of the site (to become online members of a particular library branch and comment on pages there but not elsewhere in the site, for example) are tasks handled by the Organic Groups module (http://drupal.org/project/og).

Before we talk more about Organic Groups, it would be useful to define, briefly, Drupal's approach to users, roles, and permissions (see Figure 5.15):

- Users in Drupal are individuals who access the site, regardless of whether they are visitors to the site or staff who work on it. By default, Drupal classes users in one of three roles: anonymous (anyone who visits the site without logging on), authenticated (anyone who has a log-in to the site), and administrator (the small number of individuals who manage the site).
- Roles are sets of actions a particular user type can exercise. A Drupal administrator can create as many roles as needed. Commonly, there are user roles for specific content types. You might have a role for book review authors that is separate from staff event calendar authors. Being able to contribute a book review does not give you permission to add events to the staff calendar.
- Permissions are what matches users to roles. When a Drupal administrator creates a user account, she can assign specific roles to that user. Roles are very granular and therefore very powerful. You might create one role for "content author" for staff that gives users with this role the permission to create, edit, and delete any kind of library webpage content. Another role might be more limited, giving specific patrons the ability to contribute book reviews and edit their own contributions but nobody else's.

The Organic Groups module adds power and flexibility to Drupal's roles and permissions. Content of a particular type might belong to different organizational structures in the library; some book reviews might be part of the children's library, while others are adult biography. Patrons might be allowed to contribute reviews in one area but not another. Organic Groups allows the Drupal administrator to implement this sort of modular assignment of content and users. Groups can be tightly controlled by the Drupal administrator, in which

case only individual users who have been explicitly granted permission may see, edit, or comment on content. Or, they can be very open, in which case users can create their own account and join groups, thereby gaining the ability to comment on or even edit content on your site.

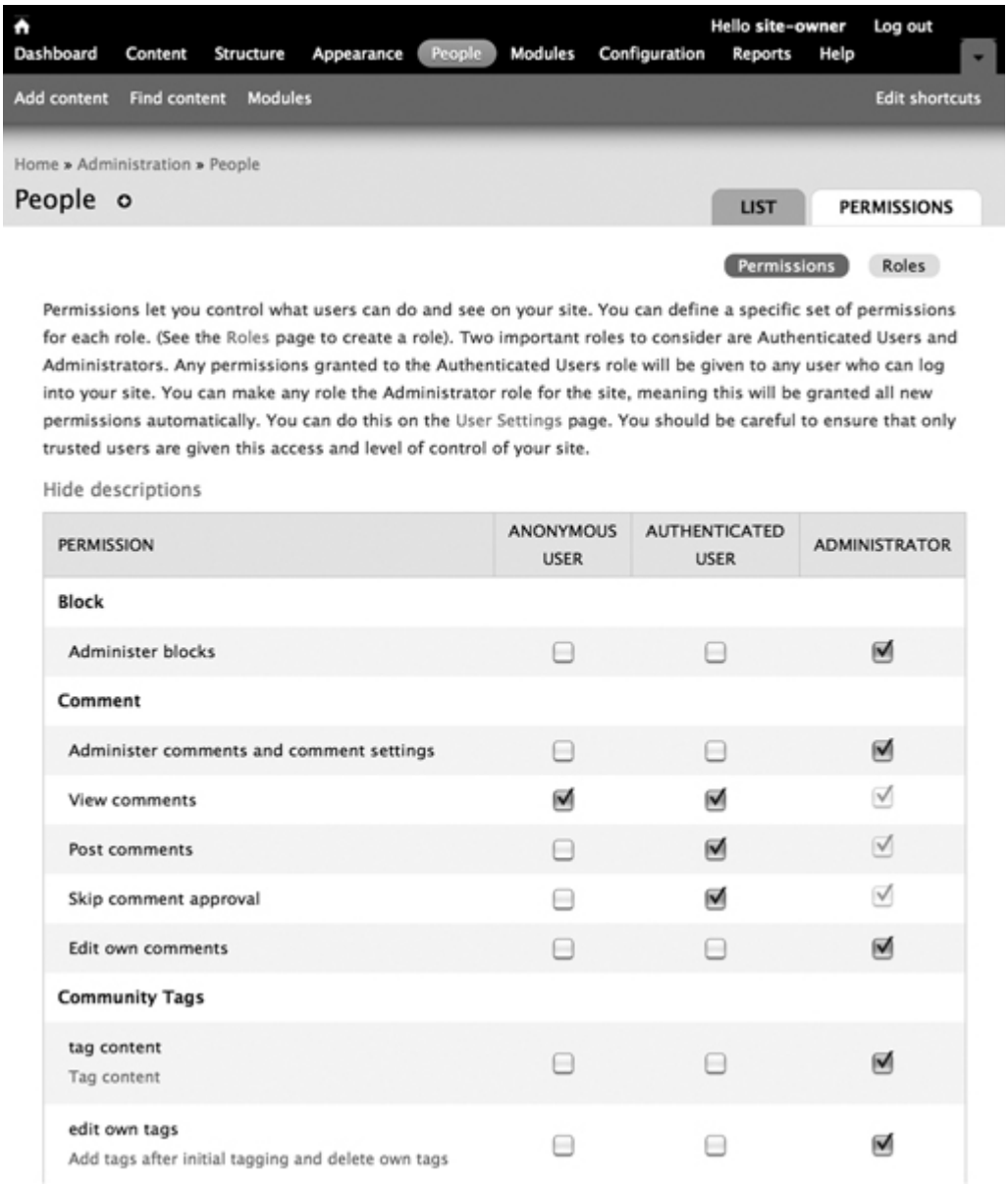

Figure 5.15: Screenshot of the Global Comment Settings Window in Drupal's User Permissions Window

The open end of the spectrum might be most applicable in a public book discussion forum or feedback area—parts of the site where you wish to have broad participation tied to individuals. At the latter end of the spectrum, you might use Organic Groups to control site authoring and clearly demarcate roles and responsibilities within the site for staff while giving the broader user community read-only (or comment-only) access but not allowing site visitors to alter the site's content beyond commenting.

Because Organic Groups is a module, you install it the same as any other—by going to your administrative interface, selecting "Modules" and then "Add module," and then entering the URL of the most recent version of the module from Drupal.org. Once you've installed the module, you'll need to make sure all the dependencies are also installed; the full functionality of Organic Groups modules requires the Entities, Views, Locale, and Chaos Tools modules as of this writing. And, once these are installed, you will need to rebuild content access permissions so that Organic Groups can interact with your content types and determine which members of which groups have permissions to work with the content.

Once Organic Groups is installed, you have the ability to set permissions for each field in each content type separately for members, nonmembers, and administrators of the group. So, for example, you could allow everyone to see comments but only group administrators to edit comments. You could make this setting global—comments no matter where they appear on the site—or just within a particular content type.

Making such permissions changes globally is one thing; making them to select pieces of the site is where Organic Groups becomes much more powerful. When creating a new node with Organic Groups installed on your site, you are able to select an "Audience" for that node. "Audience" is one or more groups to which site visitors belong, as defined by Organic Groups.

• Style Your Site by Going beyond the Standard Themes

The way your site looks says a great deal about you (many people believe you can judge a library by its website). The generic themes that come with Drupal are functional and easy to use, but they tend to make your site look like many others. Even with the new flexibility in Drupal 7 to adjust colors of certain elements of your site from within the administrative interface (to turn the blue used by Bartik into green, for example), the theme controls layout as well as colors, font faces, and placement of navigational elements.

There are three main options for styling your site. One, the least complicated, is to simply use a default theme as it is or make minor tweaks to it (either through the administrative interface or through editing the style sheet). A second, in the middle, is to take a community-contributed theme and work with it to make sure that all your modules are able to use it. At the most complicated end are rebuilding a generic into something that is all your own and building a theme from scratch. These three options are described later in this section. We'll start with an overview of themes and what is on the file system so you can understand where to customize your site's appearance.

It's worth noting that, unlike most of the configuration we've discussed up to this point, you will need to have some proficiency with PHP to update and alter themes and have some understanding of Drupal's programming architecture. Because these topics are beyond the scope of this introductory book, we will not delve into specifics of how to edit these files but will instead focus on what needs to be changed and to what effect. See "Recommended Reading" for books and web resources on learning how to program for Drupal.

The Structure of Themes

Drupal themes are a series of PHP and CSS files stored on your site's file system. From your web root, look inside the themes directory and then inside the directory that matches your theme's name. For this example, we will look at the Bartik directory, where

there are several files (bartik.info, logo.png, screenshot.png, template.php) and directories (color, css, images, and templates).

Each theme has a file that describes the theme. In this case, bartik.info tells Drupal what version of the theme this is and where the main CSS files are located, and it defines regions on the screen. The two image files are used in Drupal's administrative screens—the logo is the standard Drupal logo displayed by default in the upper left of Bartik pages, and the screenshot is displayed on the themes administrative page to show what the theme looks like.

The final file, template.php, is the script that controls the basic look of all pages in the site and defines content-specific template files when they have been created. It is a PHP file because it contains a lot of logic—if this element is present, then display it here; otherwise, do something else. So, for example, if comments are enabled for a particular node, the template file will display the comments. If they are not enabled, then it moves on to the next piece of data. You can edit this file to change the behavior of all pages in your site.

The directories have similarly organized functions. The color and css directories contain all the CSS files the theme needs, including specific files for defining the site's color palette (mentioned earlier), the layout (separate from other styling), print, and specific style sheets designed to accommodate those browsers that take an idiosyncratic approach to CSS (there are specific style sheets for Internet Explorer 6 and Internet Explorer 7 and higher).

The images directory contains the icons that the theme uses. Bartik has a minimal set—an add icon, comment arrows (for next and previous set of comments), and the border that surrounds tabs on the navigation interface. Other themes may have more icons, and you may of course wish to add additional elements yourself.

Theme Customization

Finally, and most important, is the templates directory. This is where templates for specific content types are located. We've already discussed the sitewide template.php file. Template.php controls the

framework of the site. There are also template files for nodes, pages, and comments. These, in essence, define the behavior of content within the site template, the area of the page reserved for page content. Output of a particular content type in the basic template is defined in a template file with a corresponding name. So, once you have created a “library_branch” content type (as we did in an earlier section of this chapter), you could then create a custom theme for it by creating a new template file, library_branch.tpl.php.

Once you have built your custom template, you need to update your theme’s main template.php file to instruct Drupal that there is a custom template for the library_branch content type. This custom template file’s PHP code then controls the layout and presentation of content within the framework of the general template.php file.

Best Practices for Editing Themes

When you are working on themes, it is always best to work in a development environment because you are editing files that the entire site may use. Unlike adding a new module, where changes you make are likely to affect only the content type the module manages, theme changes can inadvertently take down the entire site from the user’s perspective.

It is also wise to work with a copy of your theme so that you have a quick fallback should things go badly. By copying the entire theme directory (from “Bartik” to “Bartik-test,” for example) and making your changes in Bartik-test, you can help ensure that even if the updates have negative effects, you can roll back from them quickly by changing the site’s theme back to the original theme (in this example, Bartik).

• Integrate Library Resources into Drupal

Until Drupal 7 was released in January 2011, integrating the library's specialized content into Drupal was a greater challenge. The challenge was that data that was displayed in Drupal, for the most part, had to be within Drupal's databases.

Drupal 7, or, more accurately, the Views module developed for Drupal 7, changes that model significantly. Views are able to access and interact with data that is outside Drupal's control (as long as it is available for access via standard database connections). What this means is that access to library data—the catalog, for example—can be accomplished from within Drupal's administrative interface without exporting the data from its source, importing it into Drupal, and then working with it. This also makes tools like the Social OPAC (SOPAC) easier for libraries to work with. Additionally, modules already written for Drupal make it easy to import and access data libraries access through Serials Solutions' Summon article discovery tool. We will talk about those two tools in more detail in Chapter 9, "Developing Trends."

LibGuides

LibGuides, a web-based subject guide software package developed by Springshare (http://www.springshare.com/) and used by many libraries of all kinds, provides libraries a simple authoring interface for "how-to" material. Libraries often use LibGuides to present their subject-related research guides, showing patrons what databases are available and how to use them and how to get help, sometimes including video or other content. The power of LibGuides as a purpose-built content management system for this one class of library information has made it a popular tool. While there are modules for Drupal that will allow the creation of simple guides, many libraries have found LibGuides to be the better solution for their needs.

Putting an important subset of your content in another silo seems to go against the grain of moving your content into a single content management system. And, to a degree, it does. However, many libraries felt that LibGuides provided a better user experience with

less significant investment of resources than would building something equivalent in Drupal. Some libraries may want to access this data within their Drupal sites. Drupal gives libraries the flexibility of maintaining the data externally while providing access to it through the library site.

Accessing the Data

LibGuides, as many other data services, offers an on-demand XML export of your libraries' guide content. (This data export service is an extra service, available at a relatively low cost compared to the annual contract price.) Everything about your guide and your guide authors is provided in an XML file, including the totality of the content of the guide as HTML data within the XML file. (Items you embed in your guide, such as YouTube videos, are included as URLs.)

At the University of Michigan Library, we used this XML file to add research guides to our Solr-powered search index so that our research guides appear in search results along with pages in the Drupal site. This is one approach.

Another approach might be to use the RSS import modules (WordPress Import and Feeds) described later in the section Get Content into Your Site (pp. 82–84). Making this work would require some programming assistance to convert the content you want from the LibGuides XML document into an RSS-style feed, but, once that was done, each guide could be imported as, in essence, a blog entry. You might want to present your guide content in Drupal or perhaps provide just the summary information for the guide and link to the full guide in the LibGuides environment.

A third option is to build your own local database and import the XML data from LibGuides into it every day or week. Using Views, you could then pull data out of this database—even if it is external to Drupal—and present the information in it (e.g., the guide's title, abstract, author and author contact information, and link to the full guide) within your Drupal site.

There is more discussion about ways to bring "traditional" library content into your site in Chapter 9. For now, though, we will move on to a discussion of encouraging your patrons to interact with your site.

• Enable Social Media Features on Your Site

Now that you are building a website that is dynamically constructed based on a wide range of content, content that is reused and repurposed across your site, you will probably want to make the user experience more dynamic, as well. Earlier, we talked about creating feedback forms. Now, we are going to discuss some Drupal tools for enabling your site's visitors to comment on individual pages, add their own tags to your site, access your content via RSS, and easily share your content with others via Internet-based social media tools such as Facebook, Twitter, and the like. These are all standard tools in the two-way web, and all are readily available in Drupal's toolbox.

Commenting

The first tool is perhaps the most basic and the most expected: offering users a way to comment on, ask questions about, and discuss the contents of a particular page. Making comments available to your site's visitors is simple. In fact, not only are comments built into Drupal, but they are the default behavior when a new node is created unless you make changes to the content type on a global basis. We'll take a look at the "Comment settings" configuration pane on the Branch Library content type we've been working with. Find the content type (go to Content under the Structure menu) and look at the bottom of the screen. On the left you will notice several options; click "Comment settings" (see Figure 5.16).

When you create a new content type, Comments are activated and have some default behaviors: they are "threaded," which means that one user's reply to another's comment appears immediately beneath and is indented from the comment being referred to. You can set how many comments to display at a time as well as offering the comment author a chance to preview the comment before it is submitted—or

forcing them to if you're so inclined. Turning comments off is similarly easy: change the "Default comment setting for new content" to "Closed." Note that this will not change the behavior of nodes that have already been created—you may need to edit any existing nodes manually.

But what about "spam"? You have several options with the core Drupal code, as well as several additional modules that can help reduce spam comments. Drupal's permissions allow you to give anonymous or authenticated users the ability to post comments without review. By default, anonymous users can comment (but comments must be reviewed and approved by a Drupal administrator), while authenticated users (those whom you have given a log-in) can post comments without review. This default setting can be changed in the administrative interface by going to the People section and clicking on the "Permissions" tab. You can add, or remove, permission to post comments without review by granting or revoking the "Skip comment approval" permission. Without this box checked, that group of users will need to have comments reviewed and approved before they appear on the site.

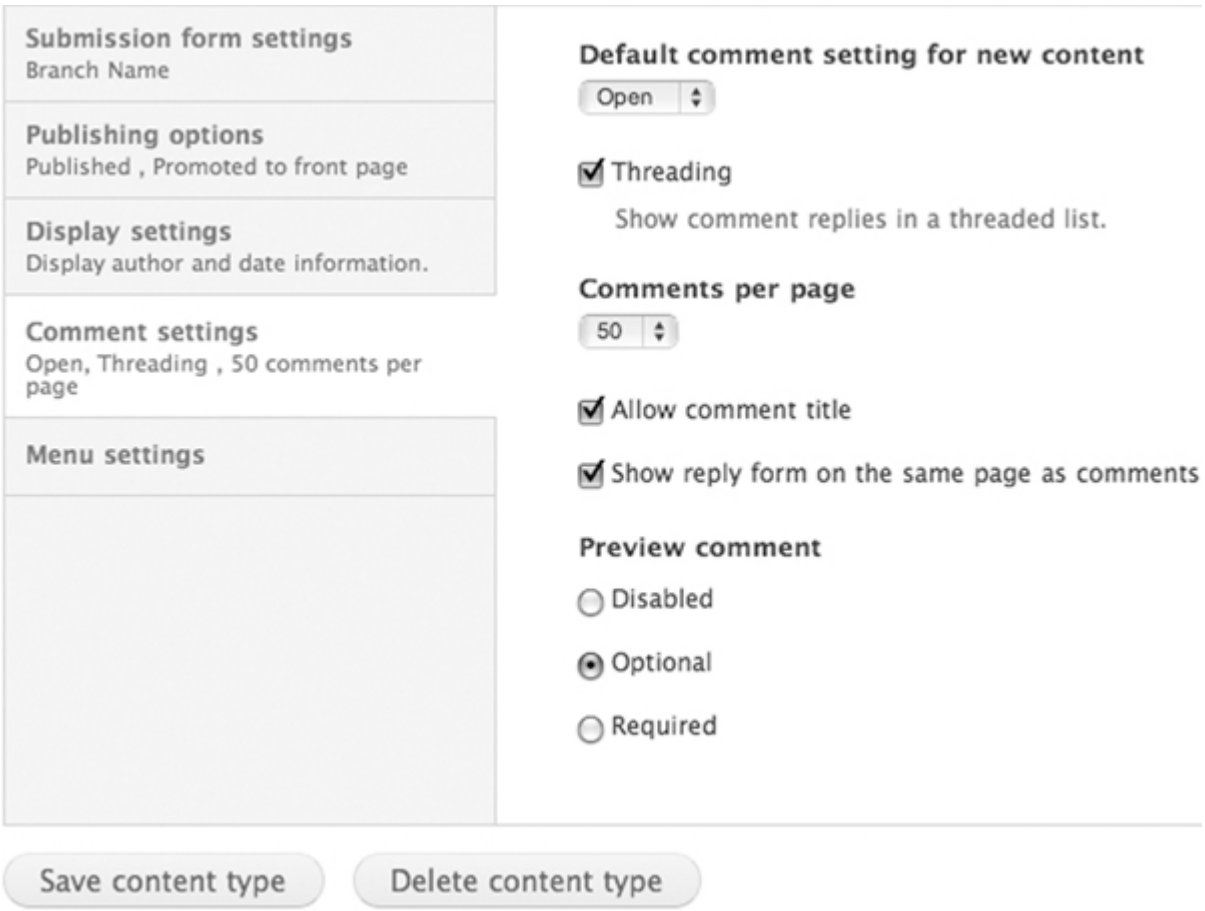

Figure 5.16: Activating Commenting for a Specific Content Type

You may wish to add additional levels of security on top of comment moderation. Doing so will reduce the workload on the administrator by filtering out more of the obvious spam. While automatic spam detection is not 100 percent effective, there are several useful modules to consider installing:

1. **CAPTCHA** (http://drupal.org/project/captcha). A "CAPTCHA" is a picture of letters, numbers, or words that are hard for a computer to understand but easy for a human; the commenter must first type in the characters they see before the comment can be submitted. While CAPTCHAs are fairly effective, improved software used by the spammers is beginning to defeat this technique.
2. **AntiSpam** (http://drupal.org/project/antispam). This powerful spam filtering module works with three Internet services (Akismet, TypePad AntiSpam, and Defensio); you can pick the one that suits you best. AntiSpam watches the comments that are submitted and, depending

on any number of cues available to it (the content of the comment, the number of links, where the links go, the IP address of the person leaving the comment, etc.), determines the likelihood the comment is spam. Comments that pass the threshold you set are not published; comments below this threshold are. You can always reverse the software's decision, so you still need to monitor your site, but perhaps not as stringently.

3. **Mollom** (http://mollom.com/). Mollom is a web service through which you can choose to direct user content. Mollom analyzes the comment and gives one of three answers: spam (don't publish it), ham (it's good), or unsure (in which case you can present a CAPTCHA to verify if the commenter is a person or a program). Mollom was cofounded by Drupal's original creator. There is a free version usable by most nonprofit organizations, with fee-based services for high-volume users based on the amount of content Mollom filters per day.

Depending on how sensitive you are to unwanted comments on your site, you may wish to use one, or both, of these methods—moderating comments and installing an antispam module. You can also force commenters to be registered users, which raises the bar a bit—but inherently prohibits anonymous feedback to your site.

Tagging

Commenting is generally a thoughtful act meant to be explicitly interactive either with the page's author or with others who have already added their comments. Tagging is more idiosyncratic. A tag is a word or phrase applied by an individual to a particular page. People tag pages as a sort of bookmarking tool so that they can find the page again or perhaps to allow others to find it. (Tagging is almost always a selfish act; people rarely tag pages for the explicit benefit of others. At the same time, the tags applied to a page by a range of users can often better define the page's content and uses than the content itself.) For the Drupal administrator, tagging is a way to allow your patrons to build a hybrid index and table of contents to your site.

Quite a few Drupal modules enable tagging. Community Tagging is the one we will discuss here. Once you have installed the module and activated it, you will need to make tagging available to users through

the “Permission” screen in the modules list (see Figure 5.17). When you activate the module, only the administrator can work with tags. This prevents the feature from being seen or used before you are ready. When making tags available to your patrons, you must first decide whether you want to allow only authenticated users to see and edit them or to make this available also to all (“anonymous”) users as well.

Begin by adding tags to a content type. We’ll add tags to the Branch Library content type by going to the Structure section of the administrative interface. Then select “Content types” and then “Library Branch.” On the Manage Fields tab, we want to add an existing field—tags exists in Drupal, even if there are no data in it yet. Give it a label that makes sense to you (“tags” might be appropriate), and select “Term reference: field_tags (Tags)” from the “Field to share” menu. Finally, select “Autocomplete term widget (tagging)” from the “Form element to edit the data” menu (see Figure 5.18). Autocomplete suggests already-entered tags to users as they type their tag. So, for example, if someone else has tagged a page with “communities,” when another user starts typing “commu” Drupal will suggest “communities” as the tag. It is just a suggestion that the user can ignore, but it can help standardize tags (singular or plural, standard spellings, etc.). As when you created this content type and added fields, you are prompted to add labels and help text for the field.

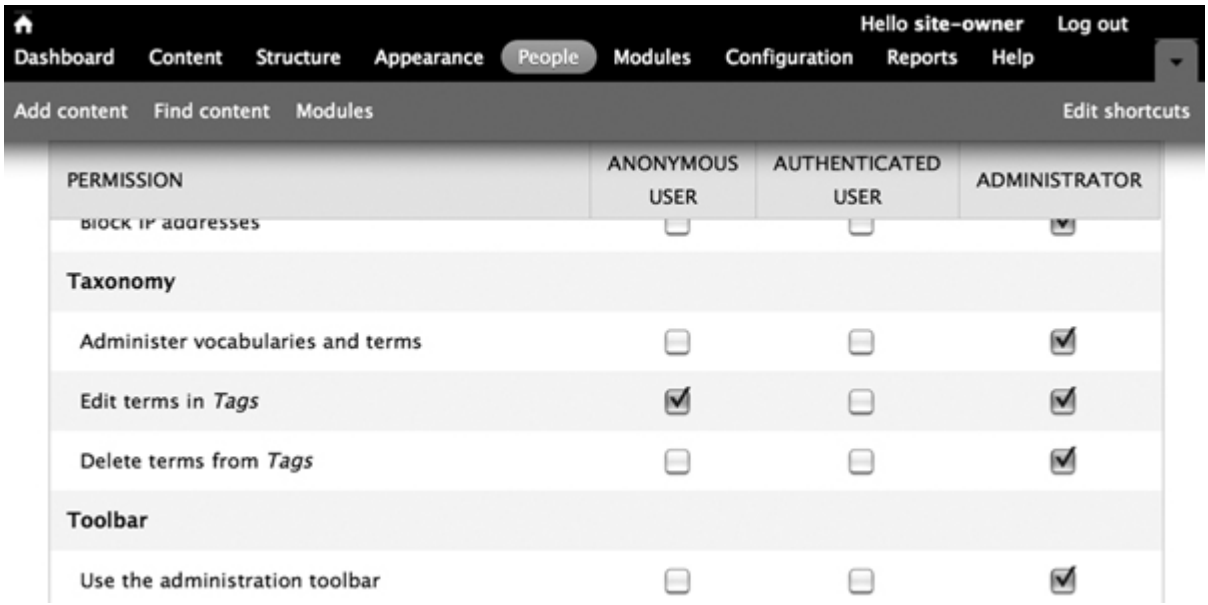

Figure 5.17: Community Tags Admin Settings

Figure 5.18: Changing Tagging Mechanism to Autocomplete

Tags are by default shown on their own tab in the interface and only to authenticated users. This default setting can be changed, for each content type, by navigating to the content type's configuration page (start at the Structure menu, then select "Content types," and finally click "Edit" for the specific content type you wish to work with). Select the "Community tags settings" tab" (see Figure 5.19), and then change the setting for "Community tagging form" to the one you want. There are three options. The default is "Tab," which puts user tags in its own tab. The other two are more common: "Block" and "Inline." Block creates a tag block that can be positioned on the content type template, whereas Inline simply adds the tags to the content type in the order in which it appears on the Edit Fields panel. If tags are the last element in the list of fields, that is where they will appear. If they are listed after Body, then that is where they will go.

Sharing

Commenting and tagging are social interactions with your content within the context of your library's website. The interactions with your site these tools engender are inherently limited, though, to people who have already found your site. If you are interested in pushing your library's content into the larger social web, then you should consider adding an additional tool that makes it easy for your site's visitors to share what they've found with others. While there are a number of Internet services that allow you to plug in a small block of custom code into your page templates (these work for any webpage, regardless of whether you are using a content management system or not), building this functionality into your Drupal site with a module is a more standardized way to achieve the goal. You will get a consistent interface to social media sites that, as with any module, you can place on specific content types where you want.

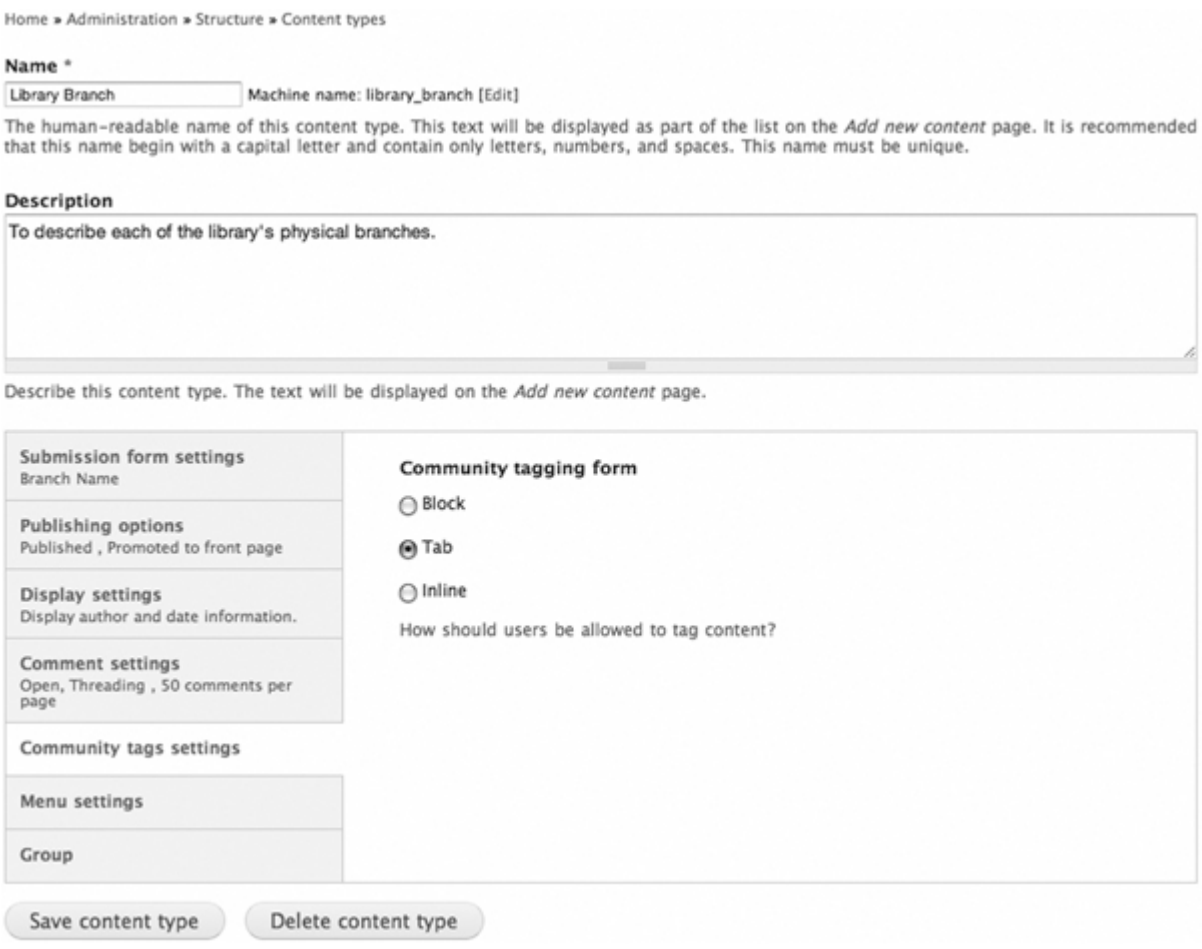

Figure 5.19: Setting Display of Tags to Block from Tab

There are several modules to choose from, including “AddtoAny” and “Social Share.” They both work in essentially the same way: once you have installed the module and configured it to display the links in the appropriate part of your site, a list of links to common social media sites (such as Facebook, LinkedIn, Twitter, Digg, Delicious, etc.) will be displayed on your site’s pages. Users clicking the link can then log in at the social media site and rapidly share the URL of the page from your site.

This functionality makes the most sense on pages within your library that describe news, events, or other content that changes. While it’s certainly possible someone would share your circulation policies, it is far more likely that next week’s lecture on doing genealogy research would be shared by a potential attendee to her friends.

Offering RSS

The third type of social sharing we’ll discuss is the least individualized: RSS feeds. Drupal makes it simple to create RSS feeds for just about any kind of content on your site. Once created, the feed will include new nodes created and published for that content type; anyone who subscribes to the feed can find out when a new page has been added to your site within that feed. As with sharing, the likely candidates for this sort of updating are content types like library news or events, the library newsletter, new or highlighted books, and that sort of thing; it is unlikely that many of your patrons will want to know every time an existing page is edited or a new page is created. Building an RSS output for a particular content type is as simple as setting up a view with the parameters you need—select one or more content types, sort by “newest first,” and be sure to check the “Include an RSS feed” option under the “Create a page” section on the View “Add new view” screen. You specify the URL on your server where you want Drupal to create the RSS feed.

• Get Content into Your Site

Even if you are redesigning the architecture of your site at the same time you are building it in Drupal, you will probably have a great deal of content from the old site to move into the new site. Migrating content is a large task in itself. There is no single, simple answer to the question of how to accomplish content migration; options range from copying and pasting text from the old site into the new to trying to automate the content importing through a script. Frankly, neither approach is wholly satisfactory or without problems, and both approaches offer their own challenges.

At the same time, implementing a content management system gives you a wonderful opportunity to review what you have on the website and streamline your offerings. Live websites, short of a redesign, rarely lose content in a systematic way; they tend to grow steadily, with unwanted content ending up hard to find through the navigation (by intent or by accident), but still available through search engines. Google (and other web-scale engines) never forgets if the page is still there.

Content Triage

The first question you want to answer is, "What content do I want to carry over to the new site?" The corollary to this question is, "What content do I have in the first place?" That may be an easy or difficult question to answer, depending on how your current site is organized. If it's already in a content management system, you can easily run reports showing all the pages that you have. If your current site is in old-fashioned HTML files, you'll have to explore the directory structure.

However you review your content, you'll want to perform triage on it, dividing it into three categories:

1. Content you want to keep as is
2. Content that needs to be updated or refreshed when it is migrated
3. Content that is no longer needed and will not be migrated

Your instinct may be to move it all and sort it out later. This is probably not a good idea. From our experience at the University of Michigan Library, where we did not perform a strictly enforced triage before moving content into Drupal, we ended up with a large number (on the order of hundreds) of pages that, ultimately, proved to be redundant or superfluous to other content on the site.

It is a good practice to figure out who will be responsible for various content areas in the new site and give them the primary responsibility for selecting content to move. This can, and should, be done with input and advice from other library staff, but if the content responsibility starts with the people who will have it after the new site goes live, the content that gets moved will have a stronger likelihood of being useful and updated going forward. Nobody likes extra work, and if you are given ongoing responsibility for the content you decide to transfer, you will make more judicious decisions.

From a Database to Drupal

What seems like the easiest thing to do—migrate content from one database to another—can in many instances be one of the most complicated. Drupal's table structures are not straightforward; information about a particular piece of content can be stored in multiple tables. (As you may recall from our earlier discussion, fields that are shared among content types are stored together, while fields that are unique to a content type are in a different place.) Devising scripts to read data from the source database, parse it, and insert it into Drupal's tables is certainly possible, but requires a great deal of work. If the content you are migrating is heavily interlinked—if each rendered page links to different other pages in the text (rather than through some standard element, such as navigation)—the task is further complicated because, until the content is imported, you will not know how to reference it in Drupal when building new links.

With that large caveat out of the way, there are some modules that can import data in selected formats into Drupal. There are modules (such as WordPress Import and Feeds) that are designed to take RSS files for weblogs and import them into Drupal's blogs content

type. There are numerous other importers for specific kinds of content in the Drupal modules directory. However, there are no general-purpose tools. Based on experience at the University of Michigan Library, the return on investment is not worth the effort.

Our experience was that it took about three weeks of a developer's time to import about 100 pages from one library's custom-built PHP content management tool into Drupal. This involved writing scripts to harvest the content from the original database and write it to Drupal, updating links within this one library's site to match the new linking structure of the new site, and identifying the links to pages within the broader institution site to pages not managed by this content management system. For much less cost, and with less effort, we could have given this content to one or two temporary hourly workers who could have copied and pasted the content from the old site into the new, rebuilding the links as they went.

The cost–benefit analysis can shift in favor of automation, in either case, if your site is sufficiently large and sufficiently consistent. If you are already in a full-fledged content management system (such as Plone or Joomla), you may have an export function available from the original content management system that will, at a minimum, give you consistent output. It is still not likely to fully resolve the challenges of maintaining links between nodes in the new site, as the links you will need are not fully known when you begin importing.

From HTML to Drupal

The situation with a plain HTML website is similar to that of databases but may actually be easier. If the HTML is consistent and well formed so that it can be read and parsed by a script, it could be turned into an RSS-style document and then imported using one of the blog import tools mentioned earlier. The consistent and well-formed HTML is important because a script would need to read the HTML file and identify the title (either the HTML title tag or another consistently identified page element), page author (perhaps in the page's metadata or in a consistently marked-up line in the document), and the page body (skipping over any navigational elements, which

means they would need to be clearly identified in the mark-up). This need for consistency creates many potential failure points. Much as is the case with a database-driven site, the globally cheaper solution may well be to manually migrate the content from the old site to the new.

• Keep Content Up-to-Date

There are many helpful reports you can build (using Views) that show content that has not been updated in more than some determined period of time. For example, you may want content authors to review their content at least every 6 months. Because Views has access to metadata about nodes (such as updated date and author information) as well as the contents, it's straightforward to set up a view, organized by author, of content that has not been updated in some specific period of time. A brief example will help illustrate. Create a new view (from the Administrative menu, select "Structure," then "Views," then click "Add new view") and give it a title like "Stale Content," check the Description checkbox, and give it a suitable description (e.g., "Content not reviewed for 6 months"). By default, Drupal will create a page for this view. Click the "Continue & edit" button at the bottom of the screen (see Figure 5.20).

On the next screen you configure the view. Click on the "add" text in the Filters section to select the kinds of content you want to include in your report. (You may want to exclude news items, for example, as once they are published they will not change.) When you click this symbol, you'll get a list of all the possible criteria you can filter on. You can add multiple filters—inclusions and exclusions—but we'll use just two: published state and revised date. We'll first limit the selection to published content by checking the box next to "Content: Published" in the list. Then click "Add and configure filter criteria." Next, select the radio button next to "Yes" and click "Apply (all displays)." Then we'll follow the same process, this time selecting the "Content: Type" criteria and then picking the specific content types we want. For this example, we will select content that is either a Library Branch or an Article by checking those. We want it to be either, so select the "Is

one of" operator. This setting will restrict the report we are creating to only content of type Library Branch or Article—everything else will be ignored. You could select all or some content types or use the opposite "is not one of" and pick the one or two things you wish to exclude. Make your choice, and click "Apply (all displays)" (see Figure 5.21).

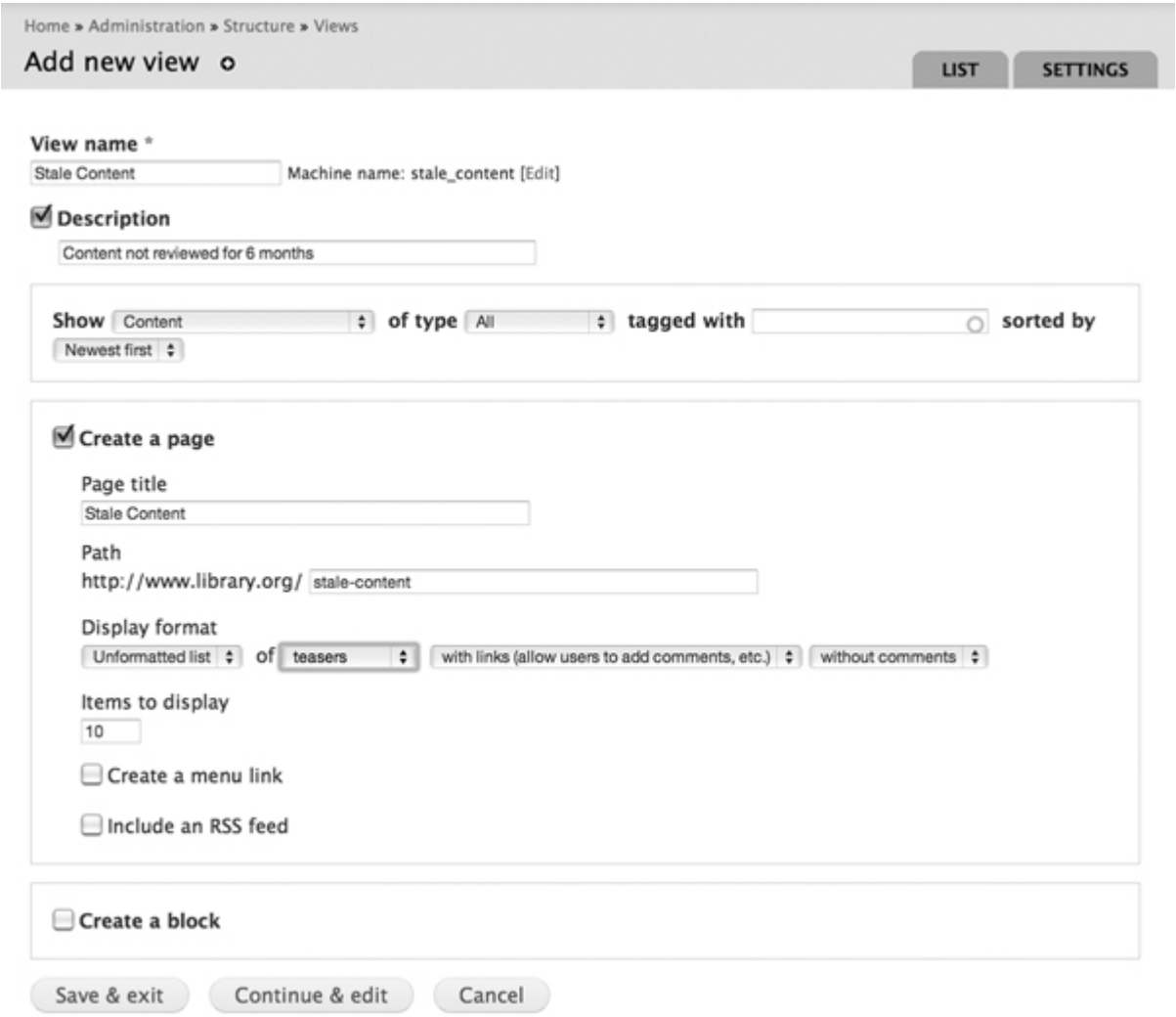

Figure 5.20: Creating a New View for the Stale Content Report

We'll add a third filter to include nodes that have not been revised in more than 6 months. Go through the same steps until you pick the specific piece of Node information you want. This time, we'll use "Content: Updated date." Here, we'll want to restrict the filter to dates

more than 6 months ago. So we'll select "Is greater than" and enter "–6 months" in the "Value" option (see Figure 5.22). Then click "Apply (all displays)." You may have noticed the "Expose this filter to visitors, to allow them to change it" checkbox at the top of the last screen. If you check this, users will be able to edit the view themselves to change the date range.

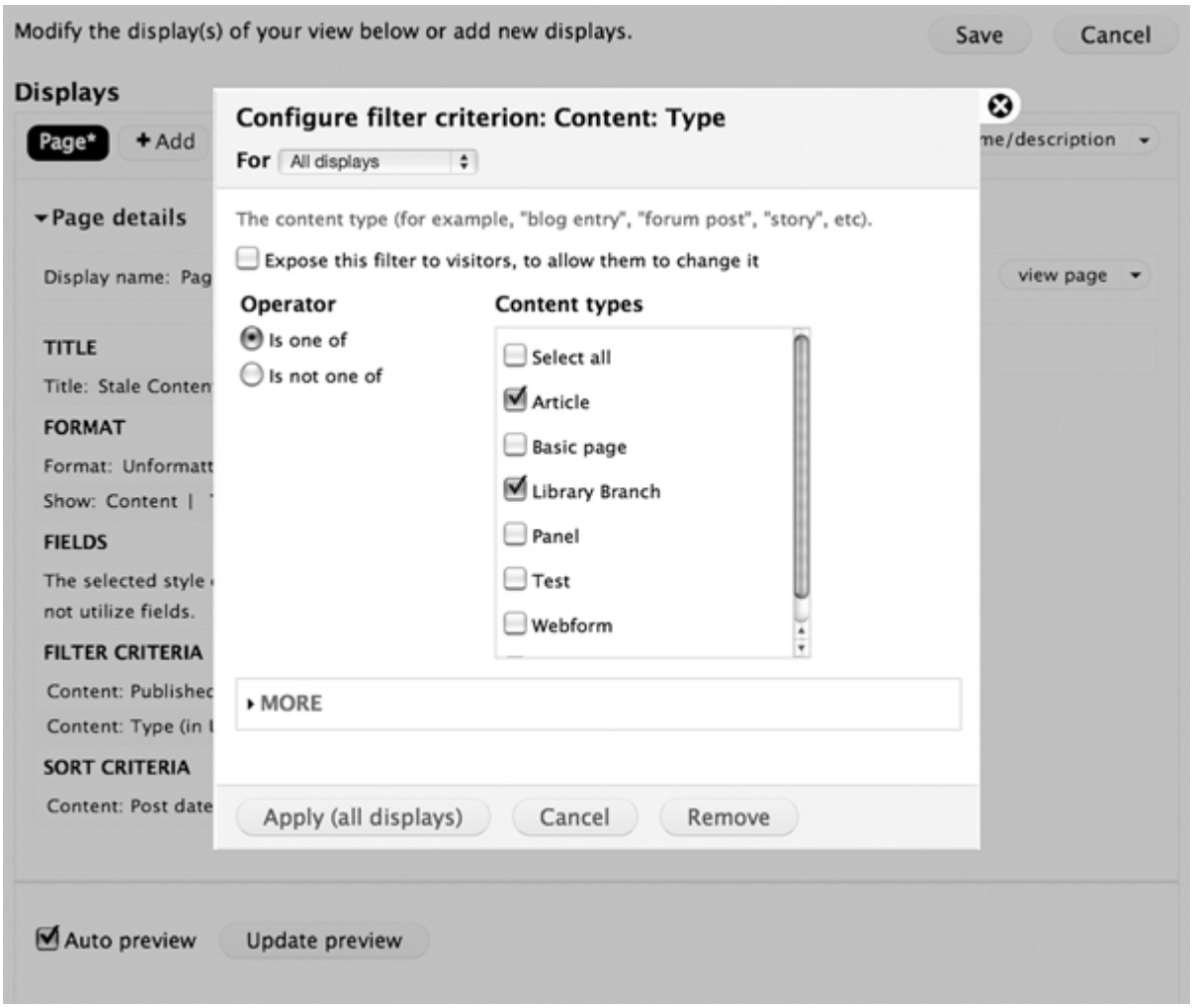

Figure 5.21: Selecting Content Types to Exclude from the Stale Content Report

Next, you'll need to pick a set of fields to display so that the Drupal page author knows which pages need work. An obvious starting set is node title, edit link, and last updated date, but you may find other fields are helpful for your own internal processes. Finally, you'll want

to create a display for this view—a URL that a staff member can access to see the potentially stale content. (This was explained in the Using Views section of this chapter, pp. 61–65.)

Every time a page author pulls up this view, he will see a list of pages for which he is the author that are at least 6 months old. If the page needs updating, it can be edited and resaved. Perhaps it is no longer needed and can be deleted. If it's fine as is, he can simply save the page without making any changes to reset the "last updated" date.

• Create Your Own Modules

Despite the vast number and diversity of modules available for Drupal, it can happen that the functionality you need is not available in a community module and you find yourself in the position of developing one yourself. Because this book is geared more toward librarian project managers than to Drupal developers, we will not go into extensive depth, but we will cover the broad steps to take to make a module for your site. In the final section of this chapter, we'll discuss contributing your work back to the Drupal community so that another organization might take advantage of your work.

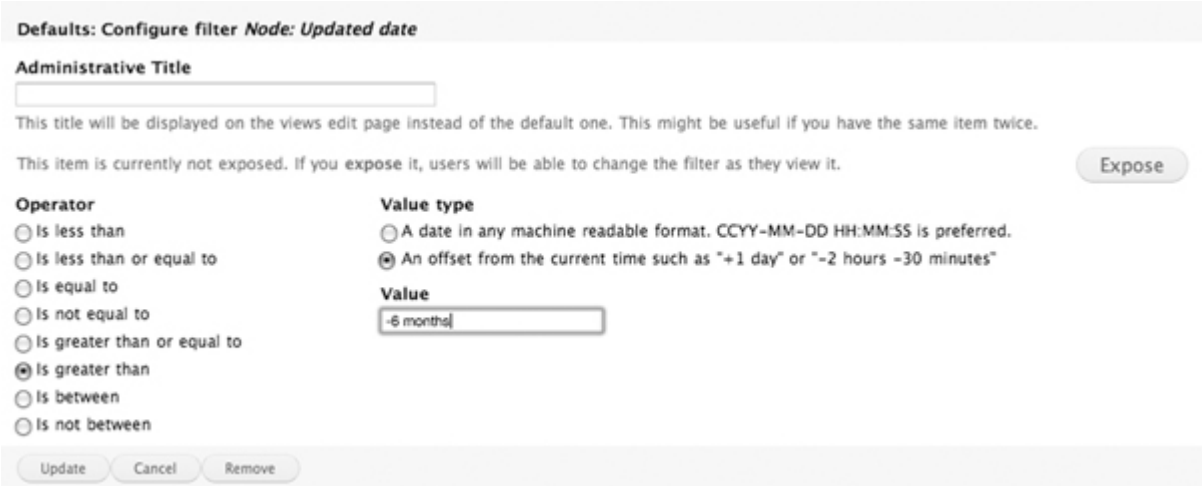

Figure 5.22: Setting the Age of Content to Include in the Stale Content Report

A module is a set of files in a directory that, when placed in your Drupal installation's sites/all/modules directory, appears on your modules list in the administrative view of the site. A module has several component files. For this discussion, we'll assume your module has the exciting name "module_name." Drupal requires that modules be named with only lowercase letters and underscores; numbers and other characters, including uppercase letters, are not allowed. Each module contains at least two files, and likely three, according to the specifications described in Table 5.2.

Modules pass information to and receive information from Drupal itself using Drupal's APIs and callback functions. In brief, a module might list all the nodes in the site created in the past week. The module would first do the computing of what nodes fit that description using Drupal's API to get a list of nodes with a created date in the past seven days. Next the module would create a block on the page into which the list of nodes is inserted. Finally, the module goes through the list of nodes and creates the HTML code to display them in the block.

Modules have some unique requirements compared to "standard" PHP applications. For example, while you always start your module's code with the PHP declaration "<?php" you do not end your PHP application with the typical "?>" but instead omit that line. There is a tutorial, "Creating Drupal 7.x modules," on the Drupal.org website (see http://drupal.org/node/361112) that walks through much of this process. For a much deeper exploration of Drupal's APIs, hooks, and module development, see the Drupal (General Resources) section of "Recommended Reading." It is wise to use the Drupal "Coder" module—available through the Drupal site—which reviews module code for compliance with Drupal's syntax, current API and hooks, and so forth. Using Coder to review your code and correct any errors it reports is not a guarantee that your module will work, but it does help ensure that it works "the Drupal way" and is a prerequisite for contributing your module to the Drupal.org for others to use.

Table 5.2: Components of a Module

File Name	Specification
module_name.info	This file contains information about the module that Drupal uses for the modules list display and to tell when there is an update available. It is a short file, containing (in a specified way) the full name of the module, the revision date, the version number, and which version of Drupal it is designed to work with.
module_name.module	This is the PHP code that makes your module work. The name can contain only lowercase letters and underscores.
module_name.css	This is the Cascading Style Sheet that customizes the display of any onscreen output generated by your module.

• Give Back to the Community

In the final section of this chapter, we will go over one of the more important, but often neglected, aspects of using Drupal: contributing back to the community. As an open source project, Drupal gets better when its users are also contributors. When you develop an interesting, useful, or clever functionality in the form of a Drupal module, do your part by contributing it to the Drupal community. It's possible that the code you write is so specific to your institution's needs that nobody else in any other Drupal-powered library will ever have need of the code, but it isn't highly likely (see Dale Askey's insightful article, "We Love Open Source Software. No, You Can't Have Our Code" in *Code4lib Journal* 5, http://journal.code4lib.org/articles/527). If there is library functionality

that you use, it is likely that other, similar libraries will have the same need.

Submitting a module to Drupal.org is a peer-review process that starts with your request for Git—the version control software used by Drupal.org—and a sandbox project. (A sandbox project is one that is in development; you can share the code with others on Drupal.org, but it is not formally endorsed or reviewed.) When you feel your code is ready for review, create a new issue in the Project Applications queue. This is a list, a fairly lengthy one, of Drupal projects submitted for review. Reviewers—in the Drupal "adhocracy," volunteers who have extensive experience and community credibility and act as gatekeepers for Drupal—will look through your code and documentation, possibly request changes, and, ultimately, provide you with the authority to promote your in-review project to being a full project.

The review process can take some time—there are many more projects submitted than there are reviewers—and it is likely (depending on the complexity of the module and the number of iterations the module goes through before being approved) that the review process can take weeks. Once your module is approved, you will be given permission to add it to the full repository. Your work with it will not be finished then; you will need to keep it up-to-date as other modules that yours depends on (if any) change and certainly as Drupal itself evolves. This is not particularly burdensome work, because you will undoubtedly be making all these updates and changes to the module yourself to keep it functioning on your own site.

6

MARKETING

- **Market Internally and Externally**
- **Market to Your Staff**
- **Market to the Public**
- **Create a Marketing Plan**

Now that you have built a Drupal site and are ready to launch it to the world, it is time to implement your marketing plan for the new site. The marketing plan is the guide you create for explaining your new website to your staff and the public. A site relaunch is an excellent opportunity to promote your library by focusing on the resources and services you have always offered while highlighting the improvements and additions you have just added. In this chapter we'll discuss marketing the site both internally and externally as well as describing, briefly, what goes into a marketing plan.

• Market Internally and Externally

The launch of your library's new website is an opportunity to reintroduce your library—both its physical location and its online presence—to your users. As with any change, individual members of your patron community will perceive it in different ways. Some will instantly appreciate the new functionality and site design; others will need some tutoring to understand the new tools and functions you offer, while still others—a minority, to be sure—will long for the familiar version of the site.

You will want to address all three of these reactions in your prelaunch marketing efforts, but focus on the latter two. Give plenty of notice a change is coming via your current website and via signs and posters in the library. Make sure that all staff, and public services staff in particular, understand the goals of the new site and are able to

explain, even if in a cursory manner, how common tasks are now done in the site. Anyone who interacts with the public will be asked questions about the site.

Ideally, your entire library staff has been brought along for the ride as you developed your new library website. As noted at the end of Chapter 4, constant communication with the entire library staff—not just with the staff directly involved in the development of the new site—will grease the wheels for a smooth roll-out when you reach that milestone. Still, as you near completion of your website, you should renew your communication of the new features and functionality in ways that focus on the user experience. For example, don't simply tell staff about the wonderful new book renewal tool; focus on the user experience and describe how a patron would get from the front page to a successful renewal of materials.

At the same time, keep in mind that your users include library staff, some of whom may not have been deeply involved in creating the new site and may have been paying only cursory attention over the redesign period. Regardless of the amount of communication to staff you conducted from the project start, as you approach the completion of development work and start moving content into the site, you should increase the pace. This will let your whole staff know about the new site and what is different—and, more important, improved—about the patron and staff interfaces. Their active investment in the site will make them better ambassadors to the public when the site launches.

In the following sections of this chapter, we will focus in more detail on marketing internally (to your staff) and externally (to your broader user community) and then briefly discuss how to create a marketing plan.

• Market to Your Staff

Sell Staff on New Features and Functionality

The scope of what you will need to teach your staff depends on your staff's general level of technical expertise and also on the system you

previously used. If you are moving from a different content management system to Drupal, then the basic concepts for page authors will be similar: staff will be familiar with needing to edit a webpage through a form, being able to easily roll back a changed page to a previous version, and (depending on the old system and how you have Drupal configured) having a built-in review process. In this case, training should be more focused on the specifics of the Drupal interface and work flow rather than on the conceptual level.

If you are moving from a static site with plain HTML files or something similar, the transition will provide more opportunities for more staff to be involved in the website. The simplicity of content creation and management is one point to highlight. The ability for staff with little technical training in HTML or CSS to manage the site is another. Drupal can give more of your staff across the organization the opportunity to play a role in developing content for the web.

Sell Staff on the Easy Interface

One of Drupal's advantages is that it simplifies tasks that might have been onerous in your old site. Drupal provides reports through its authoring and administrative interfaces to identify areas that might need work. Examples of this kind of reports are:

- **Broken inbound links:** These are pages on your site that visitors are trying to reach but cannot. During a site redesign, and to a lesser extent as part of routine maintenance, old content is deleted. Review the broken link reports to identify missing pages, allow users to re-create the old URLs (pointing to an updated page, if necessary), or restore content if it turns out that a deleted page might actually have value to the community.
- **Broken outbound links:** Your site has links from one page to another and, almost certainly, links to other websites. These links need periodic review to make sure your site's users aren't being sent to dead ends. This report shows links that are no longer working so that they can be repaired or removed. (See Figure 6.1 for a sample; the left column shows the URL that is broken—status 404—while the right column shows the Drupal nodes that contain that link.)

- **Top search phrases:** This report shows you a summary of the searches your users conduct on your site. This information can help you tailor the pages you present. If people are searching for something that is not there, perhaps a page talking about that topic should be added. Perhaps they are searching for something that is there but is described with a different word or phrase.

As described in previous chapters, page creation and editing are done through webforms using toolbars for formatting similar to what your staff might use in Microsoft Word or other similar applications. Most formatting is applied through selecting text and clicking formatting buttons. This makes content creation easier for everyone in the organization, allows more people to participate in the website, and has a low learning curve.

Position It as an Instruction Tool

Depending on what the new site does, compared to the old site, you may need to offer introductory training sessions for those members of your clientele who want them. It could be argued that a well-built website does not need instruction, but it is also better to provide that safety net for the portion of your user community that will not find it easy to adapt to the changed interface and that wants to be shown the new site and not thrown into it.

URL	Response	Error	Operations
http://www.lib.umich.edu/asia/korea/new/...	404	Not Found	Edit node 2169 Edit node 2170 Edit node 2260 Edit node 2261 Edit node 2262 Edit node 2263
http://www.lib.umich.edu/asia/korea/new/...	404	Not Found	Edit node 2169 Edit node 2171 Edit node 2260 Edit node 2261 Edit node 2262 Edit node 2263
http://www.lib.umich.edu/asia/korea/new/...	404	Not Found	Edit node 2169 Edit node 2170 Edit node 2171 Edit node 2260 Edit node 2261 Edit node 2262 Edit node 2263
http://www.lib.umich.edu/asia/korea/new/...	404	Not Found	Edit node 2170 Edit node 2171 Edit node 2260 Edit node 2261 Edit node 2262 Edit node 2263

Figure 6.1: Sample Broken Link Report for the University of Michigan Website

Additionally, the launch of a new site provides a great opportunity for your staff, especially those in public service, to interact with your patrons in a positive way. This is an opportunity to remind them of all the services your library offers even if that suite of offerings is the same in both the old site and the new. Moreover, any library training session your library offers should include references to the new site.

Offering introductory sessions to the new website not only allows your staff to present their work to the public and teach the new tools it offers but also—this can be even more important—to listen to your patrons' comments and find out where problems might be. There is nothing like real-world use to highlight the strengths and point out the

areas for improvement with any website, no matter how thoroughly tested.

Working with staff to bring them into the marketing process is one thing. But the true focus is on marketing to the public. That is the topic of the next section.

• Market to the Public

We talked about the new site as an instructional opportunity for your library's staff. How else can you promote your new site and services to your patrons? We'll discuss some venues here.

Library Publications and News Outlets

As you get closer to your launch date and have confidence that the planned date will, in fact, be the *real* date to introduce your new site, you have a series of opportunities to help you communicate. Your library probably has newsletters (online or in paper) to discuss current and future offerings. Start talking about the new site, the new features, and the new design as soon as they are established. Share screenshots of the design with your community before the pages are implemented or seek comments on them before you finalize the layout.

Much as with instruction, discussed earlier, you may find that the approaching introduction of your new site is an opportunity to talk about the library and its services in other venues, such as (for an academic library) a campus publication or (for public libraries) local news outlets. Create press releases talking about the redesign process and goals and highlight what people can do in the site—and don't forget to give credit to the staff who worked on it.

Note on Press Releases

A good press release should have a meaningful, descriptive title, a summary of what is new, and a contact person—with

phone and e-mail address—so that someone wanting more information can follow up directly.

Social Media

We have talked about old media so far. A number of new-media methods can help you connect with your patrons. You can reach a different set of your user community through these means.

For some libraries, one or more of these tools may be new. However, if you are introducing a "share this" function within your site (as discussed in Chapter 5), then it is more important for you to "eat your own dog food"—talk about the redesign process through the same media you encourage your patrons to use.

Tweet It

If your library has a Twitter account, be sure to talk about the new site there, particularly when you want to share screen designs, beta testing, or the site launch. Twitter is an excellent promotional tool to spread the word widely, but is not as good for engendering detailed feedback. Ask for feedback and comments—you are sure to get them! Because you may find that a 140-character tweet is not as detailed as might be needed to describe the issue, provide a link that leads to a webform so that your patrons can give you more detailed reports.

Share It on Facebook

If your library has a Facebook page, let your fans and followers know about the upcoming changes here as well. Show screenshots of the new site, talk about functionality, and start discussions about the changes. Facebook may be helpful to you in generating ideas about what to change from your old site. Ask questions about services or tools you currently offer to get ideas for redesigning them as you build your new site.

Blog It

Your library's blog is yet another venue to discuss the site. Blogs lend themselves to a deeper discussion than do either Facebook or Twitter but are less likely to draw instant feedback the way the other, more immediate social media tools do. Your blog is likely the best place to start a serious discussion or to discuss the philosophical underpinnings of your site redesign.

You may wish to use your blog to talk about the project from start to finish—a diary of planning and development work. While not of interest to all your users, opening up the process and decision points to public awareness, even if not full public participation, will broaden the sense of ownership your patrons have in the new site.

Create QR Codes with Your Website Address

A "QR code" is something like a barcode: in a graphical way, it encodes a site's URL so that people with camera-equipped cell phones can take a picture with their camera and open up that URL in their phone's web browser (see Figure 6.2).

If you take a picture of the QR code in Figure 6.2 with your cell phone (assuming it has a QR code reader application), you will end up at a webpage for this book. These codes can be included in any print materials, particularly posters or flyers, you distribute announcing the new site. A number of online tools will convert a URL into a QR code—search for "create QR code" to find them.

Figure 6.2: Sample QR Code

• Create a Marketing Plan

Conducting an extensive marketing campaign requires some planning in itself. Here are a few things you should consider when developing your own: this is by no means an exhaustive list but will give you a starting point for your planning.

- Who is your primary audience? Are you trying to reach all current library users? All library website users? Every potential user in your

service area?

- What is the main message? Is this an opportunity for you to simply announce a new, improved website—itself a major accomplishment—or is the redesign a part of a larger effort to reposition your library as a different type of service?
- What is changing and why? If you are changing the interface, but leaving URLs and tools in place, say so. Likewise, if you have completely redesigned the entire site, be up front about that—acknowledge that some people's bookmarks or links may no longer work, and provide suggestions for how they can find what they want in the new site. Be sure to provide contact information (to an individual or to a service point) so that those who need it can seek and get help.
- When is all this happening? Let people know the timeline. The sooner you begin talking about the site, the better.
- What input are you seeking? Early in the design process, you can invite volunteers to participate in focus groups and feedback sessions. Later in the development process, you can ask for testers of all or part of the site.

As you think through these questions, start putting together outlines of the messages you want to convey, to whom you want to convey them, and through what media. You will start putting your plan into effect before your site launches and continue after the launch. Think about how your message will evolve as you move from talking about a future change, to announcing the change itself, to talking about the new services and functions offered by your library through your new Drupal site.

As a final point, take advantage of whatever public relations, marketing, and instructional expertise your library has. Many medium-sized and large libraries have an individual or even a department that focuses on marketing and public relations. Others might belong to a larger organization (a school district, city government, or other institution) that has such resources available. Take advantage of these people by talking with them early in your redesign process. They will be able to guide you in marketing opportunities for the new site and provide language for your communication message and vehicles for getting it out to your public.

In this chapter, we have discussed marketing your new site to both internal and external audiences, as well as given a brief overview of creating a marketing plan for your site. Now, we will turn to a review and discussion of best practices for planning, building, and maintaining your Drupal site.

7

Best Practices

- **Plan and Manage Your Project**
- **Follow the Drupal Way**
- **Manage Content**
- **Launch Your Site**

In this chapter, we discuss a number of techniques and best practices that will help make the development process go as smoothly as possible. The three major aspects to developing a website are planning and managing the project, building the site, and filling it with your library's content. Some of these items have been mentioned in previous chapters, but we'll go into a bit more depth here. Other aspects will be new.

• Plan and Manage Your Project

Getting a project started, keeping it on track, and leading it to a successful conclusion are as critical to launching a site as doing the technical aspects well. Here are some suggestions to make this part of the process run smoothly.

Create a Project Plan and Stick to It

Spend the time up front developing a project plan, and involve as much of your library's staff as you can. Everyone has good ideas about how to make a successful site; bring as much of this input to the table as you can, and then winnow down the features and functions to a manageable, coherent whole. These will be tough decisions; making them with rich and frequent communication with the staff will help avoid disappointment or resentment. When you arrive at the end goal, make the plans and the timeline you have

developed available to everyone. Even more important, keep both documents up-to-date as things—big or small—change.

Change the Project Plan When You Need To

The flip side to having a project plan and sticking to it is being able to revise it when the need arises. What you knew or wanted several months ago may change. New technologies may appear, and new patron concerns may emerge. No plan goes according to plan: make sure that everyone knows when deadlines change or are missed and what the impact of those changes is going to be. Be a pest: tell everyone what will happen, when it will happen, and that it has taken place once it does happen.

Build the Site Your Library Wants but Be Sensible

It is a good idea to keep in mind the total workload, both now and in the future, as you figure out what functionality you want in your site. Temper your excitement for the absolutely perfect, but heavily customized, calendar function with the knowledge that if you build something to exactly match a current, non-Drupal work flow, you may well be building something that will, in the long run, be much harder to maintain, more complicated to integrate into the rest of your site, and (as Drupal evolves), require more development time to work with new releases of the code base. Yes, Drupal gives you immense flexibility to do things just the way you want, but this flexibility can come at a cost: long-term maintenance. There is almost certainly a module developed and used by another organization (whether or not it is a library) that comes close to your needs. If not, and you opt to build your own, make sure you use common functions wherever possible.

• Follow the Drupal Way

One of the most important things you can do as the person responsible for a Drupal site is to follow good practices for application development. Even if you are not doing the application development

yourself, you should have a good grasp of some basic principles for creating a long-lasting, stable, and low-maintenance site.

If it is not clear by this point, spending the time to learn the particular idiom of Drupal (compared to other content management systems or PHP applications you may be familiar with) will make your experience, in the long run, much more satisfactory. Drupal's architecture and way of passing data back and forth between modules and the Drupal core provides immense flexibility to you as the site administrator. Short-term expediency—composing code that works, even if it bends (or breaks!) the "rules"—will almost always come around to haunt you.

Work in a Development Environment

Put simply, do not install a new module, develop a module, or make changes to Drupal core on the production server until you have thoroughly tested it in your development environment. In fact, change Drupal core only if you are both very sure of the need and very sure you can undo the changes later if you need to. While Drupal is modular and changes to one part of the site as a rule do not harm other parts, it is possible for two modules to interact poorly and cause interface or interaction problems. Iron out all the kinks in your development environment before making changes to production.

Maintain the Site

Maintaining a Drupal site is a bit more involved than maintaining an HTML site. Not only must you make sure the content on your webpages is up-to-date and accurate, but you also need to keep up with changes to Drupal itself along with the contributed modules and themes you have used to build your site.

Drupal itself will tell you when newer versions of Drupal core or contributed modules are available. Some of these updates will be optional—for improved functionality, better interface, and so forth. Others will be highly recommended security updates. When updates

of either kind are available, they are shown on the update tab of the Available Updates report (see Figure 7.1).

All of the modules with updates available are listed along with a one-click link to download and install the updated module. As shown in Figure 7.1, Drupal core cannot be automatically updated; you must perform this update manually by downloading the update to your server and running the update script. Never perform updates of modules or Drupal core until you have made a backup of the Drupal site's file system and the database.

Backing up your site is critical when you are updating Drupal core because it is always possible, if unlikely, that an update to Drupal core can go wrong and bring down your entire site. The risk when you update a module is smaller; should something go wrong, it is likely only to bring down the specific functionality that the module handles. Additionally, when you apply updates to modules or themes that you have customized for your site, you need to be doubly careful to reapply any changes you made to the newly updated module. As discussed in the next section, it is highly advisable to do the update first in an identical development installation to make sure that nothing untoward happens.

It is good to get in the habit of performing frequent (weekly or monthly) checks for updates and installing the latest releases. The more time that passes between updates, the more likely it is that you will be leapfrogging over several incremental changes. Additionally, you want to make sure that any security holes in modules or Drupal core are fixed as soon as the updated code is released.

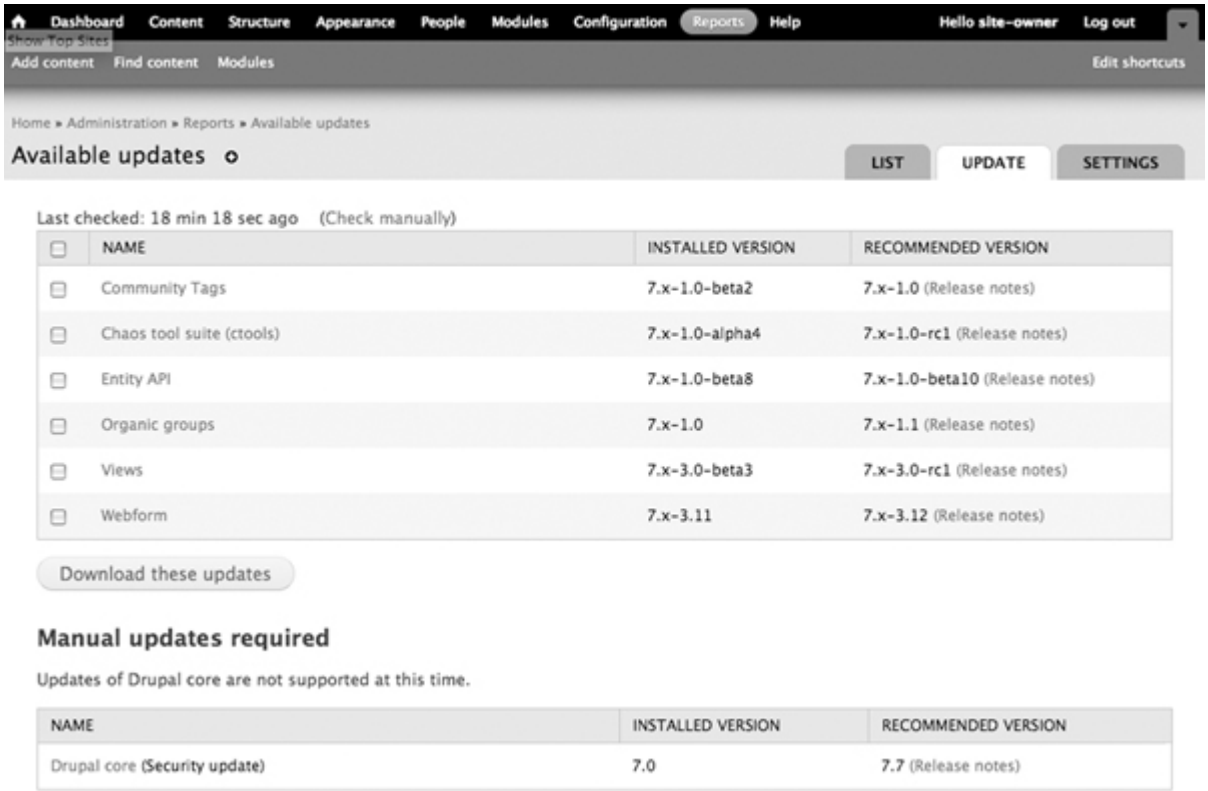

Figure 7.1: Available Updates Report Showing Updates to Drupal Core and a Contributed Module

Foster Safe Computing

Be exceptionally cautious about downloading Drupal modules and themes from any site other than Drupal.org. Anyone can make a module available via the web. Drupal's "Add a module" and "Add a theme" tools are not particular about where the downloadable file resides, but you should be. Software available through Drupal.org has been reviewed and tested by a number of Drupal experts who have built a reputation on the quality and reliability of the code they approve. This is not to say that code from other sites is automatically dangerous. At the same time, you should exercise caution in downloading modules and themes from other locations.

If you are truly security conscious, when you download new modules or themes from any source, you should do so to a part of your server's file system that is not visible from the web. This will protect you from the software you installed interacting with other

servers on the Internet. Then, once you have reviewed the code, or had a programmer look it over, you can move the extracted directory to the appropriate web-accessible space on your server and install it through Drupal's modules or themes interface. For this you will need to have access to the command line at your website. And, as always, it's best to try new software in your development environment.

Back up your production and development servers often—and always before you make changes to Drupal core, Drupal modules, and Drupal themes. Things go wrong: programmers make mistakes, hard drives fail, and data are mistakenly erased. Protect yourself from accidents by backing up the file system (Drupal core, modules, themes, and files and images you have included in your site) as well as the database structure and contents. Put the backups on autopilot and schedule them to run at least nightly. Ideally, your backups should be stored somewhere else (at a minimum, in a different building); you can send files across the Internet to a remote location for safety's sake.

Share Your Code

By the time you finish developing and launching your site, you have probably used many modules contributed to Drupal.org by other developers. You may have even written your own custom modules. If you do, share them. Drupal is a community, made stronger by the contributions of everyone who uses it.

• Manage Content

Content is a third complicated area. How to evaluate what you already have on your current site? What are the areas of strength and weakness? Where do your library's patrons spend the most time or ask the most questions?

Establish Style and Content Guidelines

You may already have style and content guidelines in place. If so, review them to make sure they still fit your needs, and apply them to

the new site. If not, take some time to determine what you want your site to say and how you want to say it. Pick a "voice" for the website. Is your site—or are parts of it—formal or conversational? Does it need translation into one or more languages spoken by your community, and, if so, to what degree? Are you adhering to a formal style guide for English, or do different authors get to write as themselves? Establish or use existing rules for writing for the web.

Be Ruthless in Culling the Herd

As you review your current website for content, take advantage of this rare opportunity to review what you have and be selective about what goes into the new site. Much as you might do when you move your residence from one house to another, ask yourself—or have members of your library's staff ask themselves—does this page really go with the new site? Do we need a new one? Have I (or anyone at all) looked at this page since the last time we moved it? If content has not been updated in years, it is probably either timeless and critically important or junk. A gambler would put odds on the latter option. Review your content carefully, and remove what is not needed.

• Launch Your Site

Take a gradual approach to launching your new website. Run the new version of the site in parallel with the old for a period of time with an invitation to explore the new site to the test audience.

Alpha Testing

An "alpha test" in programming parlance is a test of a site or software late in the design process but before the developers of the site consider it done. In an alpha test, most of the functionality and content are in place, but it is understood that there may be dead ends or (temporarily) broken functionality while the bugs are being worked out. You may wish to start by inviting all library staff to explore the site at this stage so that they can help identify true problems that have not otherwise been observed and confirm that the overall structure of the

site works. You may also bring in advisory groups you consulted early in the process to make sure users think you are meeting their needs.

Beta Testing

A “beta test” is the prerelease stage of testing. At this point, you should be fairly confident you have resolved all, or nearly all, of the critical problems identified so far and are ready to invite a broader audience in to see the site. This is a great time to invite the public to test out the new site. There is nothing like real-world use by users unfamiliar with the site to find the last areas needing improvement. At our library, we ran a staff-only beta test for several months while we finished up a few pieces of functionality and cleaned up programming bugs and content inconsistencies that the full staff helped us find.

Switching to the New Site

Once you have decided that the new site is ready to go, you can choose to run both the old and new versions of the site in parallel or make a complete cutover. If you choose to have a transition period, you can start by displaying an invitation to try the new site and then move to redirecting all users accessing the old site to the new site (with a way to get back to the “classic” site for those who wish to use it), and finally deactivating the old site completely, leaving only the new. Alternately, you can make a hard switchover.

While in years past users felt less comfortable with rapid change, the pace of the Internet has made rapid change of websites seem more normal. Unless you are feeling particular pressures to move gradually, it is generally better to move boldly and replace the old version of the site with the new at one fell swoop.

8

METRICS

- **Listen to User Feedback and Questions**
- **Use Web Analytics**
- **Monitor User Studies and Surveys**

By this point, you have established the need for a new site, picked a content management system, installed and configured it, customized or developed functionality, created an appropriate and elegant user interface, and trained your staff and patrons. You're done, right?

No! Here is where you start measuring your success and identifying areas that might still need tweaking or improvement. Truly, the process of measuring your success should have begun when you evaluated your old site and decided it was time to replace it. Whatever that decision process was, it probably involved concerns about modernity, ease of updating and maintaining it, patron use, and patron comments (not to mention staff use and comments, which are also important).

This is a great time to go back and look at the project proposal you created when the idea of redoing the website first came up. Go through that proposal and look for the specific reasons that the old site needed updating. You may have measured features or usage of the old site in some way. For example, secondary pages of the site, where you felt the important information was contained, were visited only a fraction of the time. Or perhaps the reference or circulation desk received some number of in-person, telephone, or e-mail inquiries asking for help performing some specific task. Or there may have been a concern that updating webpage content was a laborious, time-consuming process handled by one or two individuals as time allowed and the mean time to completion of an edit was four days.

You likely used some form of web analytics on your previous site, whether it was something basic like weblog analysis—looking at the web server's access logs using software like Analog—or something more sophisticated like Google Analytics (http://www.google.com/analytics) or AWStats (http://awstats.sourceforge.net/). What did this tell you about site usage in the old site? If you could track—even anonymously—individual users through your site from the front page to their final stop, what were the common paths? Were these the paths you wanted them to follow?

In this chapter, we'll talk about specific measures of website use and traffic and provide a suite of tools that you can use to describe and categorize the way your new website is being used. If you are reading this section before embarking on the redesign, you may find it helpful to apply some of these tools to your current site so that you have a baseline from which to compare your new site's characteristic use patterns.

• Listen to User Feedback and Questions

The easiest possible way to find out what your patrons think about your site is simply to listen to them. You undoubtedly have interactions with your patrons through various channels (in person, by e-mail, through feedback forms, etc.) that are not specific to your website but that garner feedback about the site, among other topics. Make a conscious effort to pull all of this together and review it. It may not be statistically rigorous, but you can be sure that your patrons will tell you what they think.

• Use Web Analytics

Figuring out how your patrons use your website is more of an art than a science. Looking at their footprints as they travel through your site is perhaps the easiest method, though it is far from perfect as it shows only what the users do, not what they intended to do or why they did it. However, gathering this data and analyzing it is easy and

can be informative, providing quantitative information about the traffic patterns on your site. We'll talk about two kinds of analysis: one using the log files your web server generates automatically, the other using JavaScript code you add to your website that talks to third-party software.

Web Server Log Files

Every time someone views a webpage on your site, your server notes the date and time, the page that was requested, the page the user was on when the user clicked a link to your site, the kind of web browser being used, and—if your site uses them and the user is logged in—the user's user ID. This set of information is recorded for every page, and every image on every page, whenever a page is loaded. Web search engines generate the same sort of rich data on your server every time they crawl your site. You can exclude them—at least the main engines like Google and Bing—from your analysis, because polite search engines identify themselves in the browser section of a log file. The "Web Server Log Files" sidebar shows a sample log file, with four log file lines, and provides a brief description of the example.

Web Server Log Files

127.0.0.1 - - [22/Apr/2011:08:55:00 -0700] "GET /news HTTP/1.1" 200 8223 "http://www.library.org/" "Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_5_8; en-us) AppleWebKit/533.21.1 (KHTML, like Gecko) Version/5.0.5 Safari/533.21.1"

127.0.0.1 - - [22/Apr/2011:08:55:01 -0700] "GET /images/library.jpg HTTP/1.1" 200 8223 "http://www.library.org/news" "Mozilla/ 5.0 (Macintosh; U; Intel Mac OS X 10_5_8; en-us) AppleWebKit/ 533.21.1 (KHTML, like Gecko) Version/5.0.5 Safari/533.21.1"

127.0.0.1 - - [22/Apr/2011:08:55:01 -0700] "GET /images/director.jpg HTTP/1.1" 200 28623 "http://www.library.org/news" "Mozilla/ 5.0 (Macintosh; U; Intel Mac OS X 10_5_8; en-us) AppleWebKit/ 533.21.1 (KHTML, like Gecko) Version/5.0.5 Safari/533.21.1"

66.249.72.52 - - [22/Apr/2011:08:57:03 -0700] “GET / HTTP/1.1” 200 8223 “-” “ “Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)”

127.0.0.1 - - [22/Apr/2011:08:57:43 -0700] “GET /newts HTTP/1.1” 404 356 “-” “ “Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_5_8; en-us) AppleWebKit/533.21.1 (KHTML, like Gecko) Version/5.0.5 Safari/533.21.1”

Log files contain one line for each component of a webpage requested by any user (human or computer). The above four log file lines represent two different interactions with a website. In the first line, one user (sitting at a computer with IP address 127.0.0.1, which is “localhost”—meaning that the user is working on the same computer as the web server) follows a link from the main library webpage (http://www.library.org/) to the news page (/news). The news page, in turn, has two separate images (/images/library.jpg and /images/director.jpg); these are requested with a referring URL of the main news page (http://www.library.org/news). This user has the Safari web browser and is using a Macintosh. The third log file line represents a different transaction from a second user at a different IP address (66.249.72.52). This user happens to be the Google search engine’s web indexing software, known as “Googlebot.” It requests and receives the site’s main page. The final log file line represents a request for a page (/newts) that does not exist. The server reports this as a status code 404 File Not Found, not the code 200 reported for successful downloads.

Server Log Analysis

The first kind of log analysis tool uses the web server’s log files directly. It provides broad usage data about your website using the web server’s log files but does not provide much expanded data about how users travel through the site or how they might behave on repeated visits. There are several tools that run through the server

log files and present reports; perhaps the best known is the venerable Analog (http://www.analog.cx).

The strength of log analysis tools is that they rely on data the web server captures anyway and, since the log analysis application is something that you install and run (either on your computer desktop or on the server), you can easily configure the reports to focus on particular segments of your site that you are interested in. You could dive deeply into all usage of your children's library pages over the course of a year, for example, or all uses of a particular page for a week.

Log analysis tools can help answer very specific questions such as which are the most frequently visited pages on your site, and what other pages (on or off your site) users click from to get there. These tools may provide information about the IP address of the computers that visit your site (geographic location or name of the Internet service provider). They generally do not track a particular user through time over repeated visits and are stronger at providing aggregate data (number of visitors per page) than they are about providing details about common navigation paths through the site.

Log analysis tools can give you information about where your users are located. Depending on the tool, it will produce varying degrees of geographic specificity—all the way from country, based on the domain name of the user's computer, down to city, by looking up the user's IP address in a database. For a local library, it would be surprising if the bulk of your users came from far beyond your local region; for a larger regional or academic library, such broad use might be expected. But they do not tell you much about a particular user's history with your site over time (IP addresses for a particular computer can—and often do—change over time, making it challenging to derive longitudinal data about a particular user out of log files).

Click Path Analysis

The second class of tool is something that we will call “Click Path Analysis.” These tools rely on more rigorous data gathering. The most commonly known example of this kind of tool is Google Analytics though there are many others. These tools make use of extra computer code that you include in your website template. The code is actually hosted on, in the case of Google Analytics, Google’s server. Whenever one of your pages is loaded, Google Analytics knows about it because a piece of code is also downloaded from Google. The code allows Google Analytics to capture more information about the particular user’s computer (screen resolution, plug-ins—like Flash Player—available, etc.) that can help you refine the content of your site.

Thanks to the code that gets downloaded, and the use of “cookies”—small text files that a web server can provide to a web browser—Google can track, in extensive detail, the interactions of a particular computer over time. These advantages make Google Analytics a particularly powerful tool because you can build up histories of how users flow through your site in the aggregate. You can also drill down into these paths to learn, for example, that people using a mobile device (i.e., a smartphone) might take one path through your site and perform a particular set of functions while users at desktops do something else.

The Google Analytics module for Drupal (http://drupal.org/project/google_analytics) makes adding a tracking code very simple by automatically adding the appropriate tracking code to your site’s pages. Once you have installed the module, you configure it with your Google tracking number. You can also have it exclude tracking of by specific groups of users—for example, you could skip tracking library staff’s clicks, or only track users who were not logged in.

A couple caveats about Google, and this kind of tool, are in order. Many people raise concerns about the concentration of so much user data in the hands of a third party. Providers like Google can track users across sites, not just within a site, and presumably do so even if they do not share user-specific data with anyone. Still, the privacy of

individual data is a valid concern for many libraries and something to be considered. There are other software packages that work similarly, ranging from other remotely hosted tools that work much like Google Analytics to those that run on a library server. In the former case, the same privacy concerns may apply, although other companies might have different legal arrangements with their customers. In the latter, the library maintains complete control over the data but needs to acquire and maintain the necessary software and, possibly, hardware.

• Monitor User Studies and Surveys

Web analytics of both varieties can provide aggregated usage information and can often point out unexpected trends, but rarely answer what can be the most important question: why do users behave the way they do? The answer to "why" is often only learned through talking to or observing users. In this section, we briefly cover some techniques for surveying, observing, and interviewing your website's users to help find out why. Readers in academic libraries may wish to see the sidebar on institutional review boards and user studies.

Surveys

A survey can be an easy way to identify broad causes of user behavior, particularly by asking questions about satisfaction, ease of use, and understandability. A survey is a baseline tool of exploring trends you might have spotted in web analytics or through feedback received at the public service desks.

You can ask very broad questions about the site to understand how your patrons interact with it though it is wise to frame these questions in the specific. Rather than asking them to rate how well the search function works, for example, ask them to think about the last search they did, and then ask whether they found what they were looking for. Rather than ask how often they use the library website or a particular function, ask them when they last used whatever portion you are

exploring. Specific questions will help frame the respondents' thoughts into a specific instance rather than making them respond in general terms. The general is also important, particularly as you measure the degree to which your new site is an improvement over the previous version, but it is often difficult to get information that leads to a direct response by asking general questions.

A survey is perhaps most valuable as a tool to understand how your users perceive or use your website or to help set priorities among a number of areas that you feel need improvement. The results of a survey—either the quantitative yes/no or degree of satisfaction responses—can point you in the direction of areas for additional investigation. But to really find out what is going on with your users, you need to observe and interact with them directly.

Institutional Review Boards

Most librarians working in academic libraries are subject to the university's human subject research policies. Academic institutions usually have an Institutional Review Board (IRB) that sets and enforces guidelines about ways to conduct research with people, even if it only involves a survey. These guidelines almost always apply to user interviews or user observation studies. The guidelines are in place to ensure that participants are aware they are part of a user survey and that the information they provide may be recorded or published, even if anonymously, and that the participants have the option of not continuing with the study at any time. These guidelines likewise set standards for protecting the privacy of study participants.

While such policies are rarely in place in public or special libraries, all librarians are advised to conduct their user research with these concepts in mind. It may also be advisable to review the proposed testing protocol with library administration and perhaps with legal counsel.

Observation

There are several ways you can observe your patrons as they use the library. This does not have to be a formal process though it can be. At the less formal end, position yourself at the reference or circulation desk (particularly if you do not generally work there) and pay attention to the questions and problems that users report. Take off your librarian hat for a moment and look for opportunities to improve the website or to identify places where users stumble in their research process.

At a more formal level, you can look for volunteers among your patrons—post ads on the website or near the public computers in your library—to participate in a study. They need do nothing except the work they were going to do anyway; the only difference is, they do it in your presence and think out loud, explaining what they are doing each step of the way. Good note taking on your part is essential; if it's not too off-putting for you or your patron, you could record (audio or video) or install screen capture software on the workstation the patron uses so that you can replay the search session. The novel workarounds your patrons will have found to solve perceived problems on your site will be eye opening.

Ideally, you will perform this sort of observational study both in the redesign process and after you have launched your new site. This sort of observational interaction will help you learn where you succeeded in your redesign and where patrons still are finding things unclear. Do not become disheartened if there are problems in the new site—there almost inevitably will be, for some users, for any site—but take this as an opportunity to continue improving your site's functionality and design.

User Testing

User testing is at the most formal end of the scale. It does not require any huge investment of technology but does require more planning and implementation. You will most often conduct more formal user testing as the final step in the investigative process or as you try to

hone your improvements to issues raised in earlier, more ad hoc, testing.

User testing is more rigorous in that you carefully control the design of the experiment. You come up with a scenario—either testing the usability of a feature of the current site or working with a prototype to make the design or work flow better. You devise a script so that all of your test subjects have the same experience and interaction with you and then record (again, either in writing, using audio or video captures, or using screen capture software) all of the steps the user takes in trying to resolve the question at hand.

In all of these approaches, you are likely to identify potential problems or concerns that are “off topic”—not in the specific area you set out to investigate. Use these “bonus” findings to plan future testing or enhancements.

9

Developing Trends

- **Investigate Forthcoming Modules**
- **Anticipate Upcoming Drupal Versions**

Until now, we have focused on what you can do with Drupal 7 today. In this chapter, we will briefly explore some tools that are not yet ready for Drupal 7 and then look ahead to Drupal versions 8 and 9.

• Investigate Forthcoming Modules

There are several library-focused Drupal modules for Drupal 6 that, as of late 2011, have not yet been updated to work with Drupal 7. These modules provide extra functionality that some libraries may want to explore. The first integrates the library catalog into a Drupal website, making each catalog record a page in the site. The second integrates article discovery via the Summon product from Serials Solutions.

The Social OPAC

The Social OPAC, known as SOPAC, is the combination of a set of utilities that runs on your web server and a Drupal module (http://drupal.org/project/sopac). In brief, SOPAC makes a copy of your library catalog data using a connector—a special utility that harvests your catalog's data and imports it into SOPAC—and then makes that data accessible within Drupal. However, accessing the data is just the start. The truly interesting and exciting features of SOPAC revolve around the "social" part of the name. SOPAC allows your site visitors to tag, rate, and review your library's holdings, giving them a way to interact with your catalog similarly to how they might be accustomed to interacting on Facebook, Flickr, or other Internet sites.

Libraries using SOPAC in the spring of 2011 included the Ann Arbor (MI) District Library, the Darien (CT) Public Library, and the Palos Verdes (CA) Library District. SOPAC's main developer, Jon Blyberg, started working on it when he was at the Ann Arbor District Library and continued at the Darien Public Library.

SOPAC is not yet updated to work in Drupal 7, but Drupal's new ability to work with external data through Views and the Content Creation Kit will make tools like SOPAC more powerful and flexible, because they will reduce reliance on the connectors described earlier.

Summon

The Summon product from Serials Solutions is an article discovery tool that indexes well over half a billion articles from scholarly and popular journals, institutional repositories, library catalogs, and newspapers. For those libraries that are using Drupal and are Summon customers, my library (the University of Michigan) has released a module, called "Article Discovery" (http://drupal.org/sandbox/bertrama/1119778) that interacts with the Summon service via its API and displays article citation information within the library website.

Taming the API

Through implementation of modules like SOPAC and Article Discovery, a library can provide a more streamlined, uniform interface to information silos. By keeping the user within a single, familiar interface for at least the initial searching and discovery process, the library can wrap that experience in the same services and tools as are offered on the regular website and do so in a consistent way. (Of course, neither of these modules necessarily replaces the "native" interface to either the catalog or Summon; they can be used as starting points from which a user may go to the native interface to have the full—and possibly more familiar—interactions.)

Both modules make heavy use of "APIs"—application programming interfaces—essentially, methods a programmer can use to pull data from one system into another to bring content and functionality into a Drupal site that actually reside in another location. As more library data becomes accessible through openly available APIs and as the use of this "linked data" expands, modules similar in function to these two will become more common.

• Anticipate Upcoming Drupal Versions

Drupal 8

Even as Drupal 7 was being launched at the annual DrupalCon in March 2011, work began on Drupal 8. Although it is certainly early to know exactly how Drupal 8 will differ from Drupal 7 or what the upgrade path is likely to be, the development road map laid out by Drupal's creator, Dries Buytaert, included these main goals (http://buytaert.net/starting-to-work-on-drupal-8, viewed on April 28, 2011):

1. Multidevice publishing (aka mobile): clean HTML/CSS, HTML5, contexts, web services APIs, etc.
2. Interoperability and integration with cloud services: web service APIs, pluggable components, clean data models, etc.
3. Delightful experience: accessibility, usability, performance
4. Configuration management: better separation between content and configuration, universally unique identifiers (UUIDs), exportables, more consistent CRUD APIs, etc.
5. Content staging

Three of these goals (numbers 1, 2, and 4) are firmly on the back end and deal with the way Drupal is structured, the way Drupal works with other external systems, and better interoperability between modules and Drupal core. Item 3 is focused squarely on the user and content developer. Drupal's administrative and authoring interfaces took a huge leap forward between versions 6 and 7 with improved web-based interfaces to create content types, build views and panels, and so forth. Accessibility saw a marked improvement as well. Going

forward, Drupal will be further enhanced for administrators, content creators, and users.

The last point, goal 5, is one that will resolve one of the biggest problems content creators have in bringing new parts of a website to life. In Drupal 7 and before, there is no particularly satisfactory way to create a section of the website in “draft” form and then release it, with content and new functionality, in a straightforward way. You can build modules on the development side and later move that improved code to the production side, but synchronizing the content—the information that appears on the page—has always been challenging. Drupal 8 promises to resolve this challenge.

Beyond Drupal 8

It is likewise far too early to predict when Drupal 8 will be released or what the upgrade path from Drupal 7 to Drupal 8 will be. However, early discussions point to Drupal 8 as being a smaller release than were either Drupal 6 or Drupal 7, leaving major architectural changes and overhauls to the subsequent version, Drupal 9. Because Drupal has a long track record of maintaining both the current and immediately previous version of Drupal, supplying bug fixes and security patches on a regular basis, there will likely not be a compelling need to move quickly to Drupal 8.

Recommended Reading

Listed here are various resources that will help you explore many of the topics covered in this book in more depth. If you have a resource that you'd like to share, please do so on the companion website at http://www.alatechsource.org/techset/.

• Drupal

General Resources

Butcher, Matt, Larry Garfield, John Wilkins, Matt Farina, Ken Rickard, and Greg Dunlap. 2010. *Drupal 7 Module Development*. Birmingham, England: Packt Publishing.

This guide to module development focuses on theme design.

Drupal—Open Source CMS. 2011. http://drupal.org/.

The main Drupal website is a wealth of information about installing, managing, and working with Drupal, including manuals and instructions for Drupal core and contributed modules and themes.

Herremans, Dorien. 2010. *Building Drupal 7 Community Sites*. New York: Apress.

This book focuses on social networking and the community aspects of a Drupal website.

Kelsey, Todd. 2011. *Drupal 7 Primer: Creating CMS-Based Websites: A Guide for Beginners*. Boston: Course Technology PTR.

This is an in-depth how-to for Drupal 7.

Knaddison, Greg James. 2009. *Cracking Drupal: A Drop in the Bucket*. Indianapolis, IN: Wiley.

This guide to maintaining Drupal is written by a Drupal insider.

Mercer, David. 2010. *Drupal 7*. Birmingham, England: Packt Publishing.

This is a basic overview of setting up and running a Drupal 7 site.

Shreves, Ric, and Brice Dunwoodie. 2011. *Drupal 7 Bible*. Indianapolis, IN: Wiley.

This book provides in-depth coverage of installing, managing, and authoring content for a Drupal 7 site.

Tomlinson, Todd. 2010. *Beginning Drupal 7*. New York: Apress.

This guide is for Drupal beginners.

Tomlinson, Todd, and John K. VanDyk. 2010. *Pro Drupal 7 Development*. New York: Apress.

This guide is aimed at advanced Drupal users.

Townsend, Robert J., and Stephanie Pakrul. 2010. *Foundation Drupal 7: Learn How to Use the Drupal Framework to Quickly Build Feature-Rich Websites*. New York: Springer Science+Business Media.

This is a nonprogrammer's guide to using Drupal 7 modules and themes.

West, Ray. 2011. *Drupal 7: A Beginner's Guide*. Emeryville, CA; London: McGraw-Hill/ Osborne.

Here is a beginner's look at installing and configuring Drupal 7.

Library-Specific

Austin, Andy, and Christopher Harris. "Drupal in Libraries." *Library Technology Reports* 44, no. 4 (May/June 2008): 1–40.

The authors take a case-study approach to Drupal's use in libraries.

Battles, Jason J. 2010. "Designing and Building a Collaborative Library Intranet for All." *Journal of Web Librarianship* 4, no. 2: 251–263.

This case study involves building an intranet using Drupal at the University of Alabama Libraries.

Becker, Danielle A. 2010. "Adventures in Drupal: Designing a Library Website Using a CMS." *Online* 34, no. 5 (Sept./Oct.): 19–21.

This article describes using Drupal for library websites.

Coombs, Karen. 2009. "Drupal Done Right." *Library Journal* 134, no. 19: 30–32.

The author discusses what modules to use to make Drupal work for your library.

Farkas, Meredith. 2008. "CMS for Next-Gen Websites: Using Drupal to Manage Library Web Content." *American Libraries* 39, no. 10 (Nov.): 36.

This article discusses the advantages of using Drupal in a library website.

Gonzalez, Alejandro Garza. 2009. "From OPAC to CMS: Drupal as an Extensible Library Platform." *Library Hi Tech* 27, no. 2: 252–267.

This article reviews the selection and implementation process for Drupal at the Tecnológico de Monterrey.

Harris, Christopher. 2008. "Modular Management." *American Libraries* 39, no. 8 (September): 48.

This is an introduction to Drupal concepts.

———. 2010. "Drupal Gets Easier: Making a Site? Ning-less? Try Drupal Gardens." *School Library Journal* 56, no. 5 (May): 14.

Harris describes using the Drupal Gardens shared hosting environment to set up small-scale Drupal sites.

Nelson, Walter. 2010. "Some Thoughts on Drupalization." *Searcher* 18, no. 5: 7–11.

Nelson candidly discusses the pros and cons of moving to a Drupal environment.

Rosenthal, Danielle, and Mario Bernardo. 2010. "Creating a Library Database Search Using Drupal." *Code4Lib Journal* 10. http://journal.code4lib.org/articles/2920.

This article describes how to create a database finder using Drupal.

Vacek, Rachel, Sean Watkins, Christina M. Morris, and Derek Keller. "Improving the Drupal User Experience." *Code4Lib Journal* 12. http://journal.code4lib.org/articles/4578.

Here are some tips and tweaks for improving Drupal's user experience for content authors.

Wiersma, Gabrielle. 2009. "Building Online Content and Community with Drupal." *Collaborative Librarianship* 1, no. 4: 169–171.

This article focuses on using Drupal to create community in your library's website.

• Content Inventories

Halvorson, Kristina, and Melissa Rach. 2012. *Content Strategy for the Web*. 2nd ed. Berkeley, CA: New Riders.

This is the bible for developing a content strategy for your organization's website.

U.S. Department of Health & Human Services. 2011. "Content Inventory." U.S. DHHS. http://www.usability.gov/methods/design_site/inventory.html.

This describes a process for conducting content inventories developed for U.S. government websites.

Veen, Jeffrey. 2002. "Doing a Content Inventory (Or, a Mind-Numbingly Detailed Odyssey through Your Web Site)." Adaptive Path (database). Accessed September 2, 2011. http://adaptivepath.com/ideas/doing-content-inventory.

This readable guide describes how to conduct a content inventory of your site.

• Open Source Software

Askey, Dale. 2008. "We Love Open Source Software. No, You Can't Have Our Code." *Code4lib Journal* 5. http://journal.code4lib.org/articles/527.

Askey discusses the tension in organizations between consuming and producing open source software.

Morton-Owens, Emily, Karen L. Hanson, and Ian Walls. 2011. "Implementing Open-Source Software for Three Core Library Functions: A Stage-by-Stage Comparison." *Journal of Electronic Resources in Medical Libraries* 8, no. 1: 1–14.

The authors discuss using open source software in a library setting.

Sheehan, Kate. 2009. "Creating Open Source Conversation." *Computers in Libraries* 29, no. 2 (Feb.): 8–11.

The author discusses how to use Drupal to enable sharing of information across a website and allow patrons to be part of the conversation.

• Usability/User Experience

Borchers, Jan. 2001. *A Pattern Approach to Interaction Design*. New York: J. Wiley and Sons.

This book provides patterns—recipes—for defining and describing common design features on a website.

Krug, Steve. 2005. *Don't Make Me Think: A Common Sense Approach to Web Usability*. New York: New Riders Press.

This classic guide to website usability is by one of the field's experts.

Morville, Peter, and Louis Rosenfeld. 2006. *Information Architecture for the World Wide Web*. 3rd ed. Sebastopol, CA: O'Reilly.

This seminal work addresses organizing information on a website.

Nielsen, Jakob. 1999. *Designing Web Usability*. Indianapolis, IN: New Riders.

Despite its age, this book provides highly relevant and useful tips for making a website usable.

• Version Control

Loeliger, Jon. 2009. *Version Control with Git: Powerful Tools and Techniques for Collaborative Software Development*. Farnham, England: O'Reilly.

This is a step-by-step guide to using Git for version control.

Pilato, C. Michael, Ben Collins-Sussman, and Brian W. Fitzpatrick. 2008. *Version Control with Subversion*. Sebastopol, CA: O'Reilly.

Written by Subversion's authors, this guide provides detailed instructions for using this version control software.

• Web Accessibility

AChecker. 2011. "Web Accessibility Checker." Accessed November 1. http://achecker.ca/checker/.

This Canadian resource provides detailed reports on accessibility issues ranging from bad HTML to poor color contrast and beyond.

A List Apart. 2011. "User Science: Accessibility." Accessed September 1. http://www.alistapart.com/topics/userscience/accessibility/.

This list of online articles relating to accessibility is a helpful index to recent blog posts and articles on web accessibility.

U.S. Government Services Administration. 2011. "Section 508 Guidelines." U.S. GSA. Accessed November 21. http://www.section508.gov.

These are the guidelines that U.S. government and many state and local government agencies require web developers to follow.

World Wide Web Consortium. 2008. "Web Content Accessibility Guidelines (WCAG) 2.0." Accessed September 1. http://www.w3.org/TR/WCAG20/.

These guidelines were developed by the W3C and are broadly used across government, commercial, and nongovernmental websites.

INDEX

Page numbers followed by the letter "f" indicate figures; those followed by the letter "t" indicate tables.

A

Accessibility, 10, 30, 43, 102, 115–117
Acquia Drupal, 44
Activating, 50–52, 51f
"AddtoAny," 81
Administrative interface, 44, 45
  organic groups, 8, 70–72, 71f
  reports and, 84–87, 85f, 86f, 87f
Adobe Photoshop, 23
AJAX-based, 44–45
Akismet, 78
Alfresco, 6t
Alpha testing, 104
Amazon.com, 16–17, 49
Analog, 110
Analytics, 107–111
  click path analysis, 110–111
  web server log files, 108–109
Ann Arbor (MI) District Library, 115–116
AntiSpam, 78
Apache, 40–41
API. *See* Application programming interface (API)
Apple.com, 109
Application programming interface (API), 116–117
  CRUD, 117
  Drupal's API, 87–89, 116–117
Architecture
  Drupal, 27, 100
  information, 35–36
  programming, 73
  website, 82
Article discovery, 75, 115–116
Askey, Dale, 89
Audio, 113
Authoring group, 36

Available updates report, 101–102, 102f
AWStats, 107

B

Backup, 48
  hardware, 13–14, 16
  safe computing,101–103
Bartik theme
  administrative interface, 45
  colors, 50–52, 51f, 72
  default for public view, 25
  editing, 72, 74
  example, 26f
  files, regions and blocks, 47t, 53f
  navigation, 53, 53f
  structure of, 73–74
Bartik.info, 73
Bartik-test, 74
Berra, Yogi, 4
Best practices
  content management, 103–104
  Drupal way, 100–102, 102f
  launching site, 104–105
  planning, 47t, 74, 99–100
  safe computing, 102–103
Beta testing, 104
Bing, 47, 47f, 108
Black box system, 4
Blocks, 10, 45, 53–54, 53f
  module creation, 87–89
  navigation, 53
  panels and, 65, 67
  tagging and, 78–80,79f, 80f
  themes and, 52
  views and, 61, 65
Blogs, 2, 96
  activating, 50–52, 51f
  RSS feed, 46, 81–82
  software, 15
Blyberg, Jon, 115–116
Book page nodes, 45–46
Broken links, 93, 94f
Browsers, 5, 50–51, 69, 73, 96, 111

analytics and, 108–109
Buytaert, Dries, 117

C

Calendars, 8
CAPTCHA, 78
Cascading Style Sheets (CSS), 2–3, 10, 30, 73, 92–93
customization, 25
defaults, 64
forthcoming updates, 117
ID fields, 65–66
module components, 88t
panels and, 65–66
theme structure and, 73
Catalog (OPAC), 116
access, 75
integrating, 13, 39, 115
modules, 49
searching, 23, 31
serials and, 116
Cell phone, QR codes, 96, 97f
Click path analysis, 110–111
Closed (proprietary) system, 4, 77
Cloud computing, 16–17, 117
Amazon Web Services, 16
CMS. *See* Content management system (CMS)
Code sharing, 103
"Coder" module, 88
Code4lib Journal, 89
Color, activating, 50–52, 51f, 72
Communication, 34–37
Community, giving back to, 26–27, 26f, 89
Configuring
library website, 55–61, 57f, 58f, 59f
panels and, 65–67, 66f, 67f
views and, 61–65, 62f, 63f
Content
categories, 82–83
creation, 56–59, 57f, 58f, 59f, 60–61, 92–93
creation interface, 45–47, 47t
defined, 7
editor, 29
links, 3, 7, 57, 93, 94f, 96f, 97

management, 103–104
themes, 9–10
Content Creation Kit (CCK)
constructing, 56–59, 57f, 58f, 59f, 60–61
panels, 65–67, 66f, 67f, 68f
SOPAC and, 116
view and, 61–65, 62f, 63f
Content management system (CMS)
advisory groups, 35–36
buy-in, 35
change management, 34–35
efficiencies, 2–3
feature and functionality matrix, 6t
vs. flat file, 1–3
open source vs. closed, 4
selecting, 4–5, 6t
Cookies, 111
"Creating Drupal 7.x modules," 88
Cron jobs, 44, 47t, 55
CRUD, 117
CSS. *See* Cascading Style Sheets (CSS)
cURL, 50
Customize, 9–10, 14t, 24, 44, 73, 100–101
CSS and, 25
e-mail and, 69
functionality, 10
installation and, 48
modules and, 27, 55, 88t
theme, 27, 74–75

D

Daisy, 6t
Darien (CT) Public Library, 115–116
Dashboard, 4, 50–52, 51f
Databases
activating, 50–52, 51f
application, 40–42
backup, 101–103
connecting to, 31
CMSs and, 2
creation, 42, 43f
Drupal, 74–76
entry, 8, 27

files, 48, 60
finder, 23
IP address and, 110
migration, 83–84
MyPHPAdmin, 42, 43
structure, 103
tables and, 7–8
Delicious, 81
Development
community, 6t
CMS and, 6t
environment, 101
hosting options, 14t
website, 17–20
Digg, 81
Digital Information Services Librarian, 36
Director of Communications, 36
Drupal
activating, 50–52, 51f
administrative interface, 59f, 65
architecture, 27, 35–36, 100
backup, 101–103
blocks, 10, 45, 53–54, 53f
“Coder” module, 88
community giveback, 89
concepts, 2, 7–11, 23–25
configuring, 55–59, 57f, 58f, 59f, 60–65, 62f, 63f
Content Creation Kit (CCK), 56–59, 57f, 58f, 59f, 60–61, 92–93, 116
content management, 4–5, 7–8, 10–11
customized, 9–10
database, 42, 43f, 74–75
defined, 5–6
development, 14t, 17–20, 24, 101
feature and function matrix, 6t
file system and, 1–3, 47–49, 47f
filter, 63–64, 78, 85–87
functions, 8, 31–32
Groups website, 7
installation, 8, 41–45, 41f, 52–54
integrating library resources, 74–76
library developer community and, 6–7
maintenance, 29
operating system creation, 40–41

outsourcing, 13, 14t, 18–19, 28–31
reports and, 45, 55f, 84–87, 85f, 86f, 87f
script installation, 43
security, 34, 43–44, 77–78, 117
selecting, 4–5, 6t
site design, 9–10, 29–31
SOPAC and CCK, 116
tagging, 78–80, 79f, 80f
tools, 6t
trends, 115–117
tutorials, 88
versions, 12, 115–117
vocabulary, 7, 17, 49
web root, 47–49, 47t
See also Installation; Modules; Themes
Drupal API, 87–89, 116–117
Drupal 5
migration, 12, 83–84
out-of-the-box, 25–26, 25f, 45
Drupal 6
Acquia, 44
administrative interface, 44
Content Creation Kit (CCK), 56
installation, 40–42
modules, 12, 48
Drupal 7
Content Creation Kit (CCK), 56
files, where to find them, 47t
installation, 40–45
integrating library resources, 74–76
modules, 12, 48–49, 115
regions and blocks in Bartik theme, 53f
reports, 25
themes, 72–73
trends, 115–117
tutorials, 88
Drupal 8, 116–117
Drupal 9, 116–117
DrupalCon, 116–117
Drupalib, 7
Drupal.org, 41, 44, 46, 49–50, 82, 88, 89

E

E-mail form, 69
Etches, Amanda, 24
Evaluation. *See* Measuring achievement
Event calendar, 23
EZproxy, 7

F

Facebook, 76, 81, 96, 115
Faculty Advisory Group, 35
FAQ interface, 9
Fast Company (magazine), 34
FCKeditor, 8
Feature and Functionality Matrix for Selecting a CMS, 6t
FedEx, 34
Feedback, 31
  social media and, 76–80, 77f, 79f, 80f
  Webform module, 67–69, 68f, 69f
Feeds, 76, 83
Fields, 7–8, 56–61, 64–65, 69, 79–80, 87
  activating, 50–52, 51f
File system, 41–45, 47–49, 47f
Filter, 63–64, 78, 85–87
Firefox, 5
Flash Player, 110
Flat file system, 1–3, 28
Flickr, 67, 115
Focus groups, 36
Footer, 53, 65
Functionality, 31–32

G

Garland theme, 25, 25f, 47t
Git, 48
Global Comment Settings Window, 71f
Google, 47f, 82, 108–109
Google Analytics, 23, 49, 107, 110–111
Googlebot, 109
Graphics
  Drupal and, 65
  file system and, 47f
  identity and, 2–3, 3f, 18, 22
  outsourcing, 18, 28

QR codes, 96, 97f
Guides, 75–76

H

Hardware
collaboration, 33
hosting and developing options, 13, 14t, 17–20
maintaining, 111
options, 14–16, 40
protection, 48
Harlan Hatcher Graduate Library, 3f
Harvard University, 34
Headers, 65
Heterogeneous site, example of, 3f
Homepage, 2, 3f
Hosting
cloud, 16–17
commercial, 40
developer, 17–21, 101
hardware, 13, 14t, 17–20
options, 14t, 13–17
server, 15–16
web server log files, 109–110, 109f
HTML
accessing data, 75
content management and, 1–3, 28, 30, 82
creating a document, 61
Drupal and, 10, 84, 92–93, 101, 117
file system, 47t
filter, 61
modules and, 88
web server log files, 108–109

I

Icons, 66, 73–74
Images, 7, 11
adding, 42
backing up, 103
file system, 47t
libraries, 30
themes and, 73–74
Implementation. *See* Installation; Methods and practices

Information Architecture Advisory Group, 35
Inline, 80
Installation
  administrator interface, 44–45
  configuring website, 55–61, 57f, 58f, 59f
    panels and, 65–67, 66f, 67f
    views and, 61–65, 62f, 63f
  content creating, 42–43, 43f, 45–46
  files, 41–45, 41f, 47–49, 47f
  hardware, 39–41
  modules, 49–52, 51f, 87–89, 88t
  operating environment, 40–41
  panels, 65–67, 66f, 67f
  script, 43
  security, 34, 43–44, 77–78, 117
  setup, 42–44
  themes, 52–54, 53f
  updates, 54–55, 55f
  web root, 47–49, 47t
  website download, 41
Institutional Review Board (IRB), 112
Interactive
  design, 23–24
  site, 9–10, 29–31
  tagging, 78–80, 79f, 80f
Interface
  administrative, 57–62, 66, 57f, 58f, 59f, 62f, 63f, 66f
  content management systems and, 3–4
  Drupal and, 11–12, 14t, 26, 40, 103
  installing, 42, 44–47, 47t, 48–49
  LibGuides, 75
  modules and, 116–117
  organic groups, 72
  staff and, 93–94, 94f
  themes, 52–56, 55f
  URL and, 49–52
  Webform page, 68f
  website and, 7, 9, 97, 101, 107
Internal developer, 19–20
Internet Explorer, 73
Internet provider, 15
IP address, 69
  spam and, 78

web log files, 109f, 110
IT department, 33–34, 44

J

JavaScript, 47, 108
Joomla!, 6t

L

LAMP, 40, 48
Launch, 36–37
marketing, 91–92, 94–98
testing, 104–105
Layout, page, 2
Legacy application, 2
LibGuides, 75–76
Library
administration, 15, 35, 112
categories, 82–83
contact, 3, 7, 23, 93, 94f, 96f, 97, 115
content creation, 56–59, 57f, 58f, 59f, 60–61
content creation management, 103–104
developer community and, 6–7, 101
interactive design needs, 23–24, 30
IT department, 33–34, 44
LibGuides, 75–76
migration, 83–84
newsletter, 23
patron accounts, 23
site, 9–10, 29–31
staff, 34, 92–95, 94f
web team, 36
Library branch, 56–61, 70, 86
panels and, 65–67
social media and, 76, 79
theme customization, 74
views and, 61–65, 63f
LinkedIn, 81
Links, 2–3, 7, 97
adding, 64
administrative, 45–46
broken, 27, 29, 93, 94f
database and, 83–84

downloading, 44
log-in, 56
module and, 52, 54
panels and, 10
resources and, 23, 56–57, 61, 76, 87
social media and, 76, 78, 81
Linux, 40, 42, 44, 48
Localhost, 109
Log analysis, 107, 109–110
Logo, 3, 10, 22, 53, 53f, 56
logo.png, 73

M

Macintosh, 109
Maintenance, 29
Malicious software, 50
See also Security
Marketing
change and, 34–35, 91–92, 94, 96, 98
external, 91–92
integrating library resources, 74–76
interface, easy, 93
links and, 93, 94f, 96f, 97
plan, 97–98
public, 95–97
QR codes, 96, 97f
search engine optimization (SEO), 82
social media, 95–97
staff, 91–95, 94f
Measuring achievement
analytics, 107–111
AWStats, 107
click path analysis, 110–111
Google Analytics, 107–108
log files, 108–110, 109f
observation, 112–113
questions, 108
selecting a CMS, 6t
server log analysis, 109–110
surveys, 111–112
testing, 113
user feedback, 31, 67–69, 68f, 69f, 108–109
social media and, 76–80, 77f

tagging, 79–80, 79f, 80f
web analytics, 108–111, 109f
web log files, 108–110, 109f
Methods and practices
administrator interface, 44–45
Amazon.com, 79
community giveback, 89
configuring, 55–59, 57f, 58f, 59f, 60–65, 62f, 63f
content, 45–46, 70, 82–84
creating operating environment, 40–41, 43f
database migration, 83–84
feedback, 67–69, 68f, 69f
file system, 47–49, 47f
HTML, 45, 53, 67, 70, 72, 74, 80, 84
installation, 40–43, 41f
integrating resources into, 83–84
LibGuides, 75–76
organic groups, 8, 70–72, 71f
page layout, 2
panels, 65–67, 66f, 67f
reports and, 45, 55f, 84–87, 85f, 86f, 87f
RSS feeds, 81–82
search engine optimization (SEO), 82
security, 34, 43–44, 77–78, 117
setup, 43–44
social media enabling, 76–78, 77f
sharing, 80–81, 81f
tagging, 78–80, 79f, 80f
style, 72–73
theme installation, 9–10, 52–54, 53f, 73
theme installation, customization of, 74–75
up-to-date, 54–55, 55f, 84–87, 85f, 86f, 87f
See also Installation; Modules
Metrics. *See* Measuring achievement
Microsoft Word, 93
Migration, 83–84
Mobile, 117
Modules
activating, 50–52, 51f, 77
blocks, 87–89
community, 26–27, 26f, 31, 89
creation, 87–89, 88t
customized, 27, 55–56, 88t

environment, 101–103
file system, 47–49, 47f
functions, 2, 6t, 7–8, 10–12, 11f, 31, 75
installation, 40, 49–52, 51f, 75, 77–81
interface, 116–117
options, 4, 24–27, 42, 45–46, 83
organic groups, 8, 70–72, 71f
out-of-the-box, 24–27, 25f
panels, 10–11, 27, 65–67, 66f, 67f, 80
reports, 54–55, 55f, 88–89, 111
up-to-date, 53–55, 55f, 102f, 115–117
views, 11, 11f, 40, 52, 61–66, 62f, 63f, 66f
integrating, 75–76
updating, 84–87, 116–117
web root, 47–49, 47t
See also Themes
MODx, 6t
Mollom, 78
MyPHPAdmin, 42, 43f
MySQL, 17–19, 40–42
Acquia Drupal, 44

N

Navigation
content, 2–3, 8, 10, 45, 53, 70, 72, 80, 83
interface, 74
menu, 26–27, 26f, 67
panels, 65
Node, 11, 60–62, 86–87, 93
book page nodes, 46
defined, 8
panel and, 65, 68
social media and, 76–77, 81, 84
themes and, 72–74
views and, 64

O

Observation, 112–113
Online journal finder, 23
OPAC. *See* Catalog (OPAC)
Open source systems
black box system, 4

vs. closed systems, 4–5
Drupal and, 5–6, 6t, 8, 40–41, 54, 89
pros and cons, 1–3, 33–34
OpenURL resolver, 7
Operating systems, 17, 40
closed (proprietary), 4, 77
environment, 40–41, 55
open source, 2–6, 5–6, 6t, 8, 33–34, 40–41, 54
Organic groups, 8, 70–72, 71f
Orphaned pages, 28
Outdated information, 28–29
Out-of-the-box modules, 25–26, 25f, 45
Outsourcing, 13, 14t
developers, 18–19
staffing, 28–31
Overlays, 22

P

Palos Verdes (CA) Library District, 115
Panels. *See* Modules
Pathauto, 8
Perl, 2
PHP
content management and, 2
customization, 74, 84
design and, 30, 88
development and, 19, 100
Drupal and, 10, 17
file system and, 47f
operating environment, 40–43
style and structure with, 73
Planning. *See* Project management
Plone, 6t
PostgreSQL, 41
Press release, 95
Priority, 31
Product availability, 6t
Profiles, 47t
Programming, 19, 73, 76
API, 116
staff, 14t, 17, 22, 28
Project management
change and, 34–35

communication, 34–37
content, 21–24, 28–29
development, 21–22, 101
functionality, 23–24, 31–32
goals, 23–24
interactive, 23–24
inventory resources, 22–23
IT department, 33–34, 44
modules, 26–27
needs assessment, 31–32
out-of-the-box, 25–26, 25f, 45
planning, 21–22
redesign, 30, 32
scope of, 24–25
site design, 29–31
staffing needs, 28–29
stakeholders, 21, 35
transition, 34
usability assessment, 22–23
web team, 36
See also Best practices; Marketing
Public relations. *See* Marketing

Q

QR codes, 96, 97f

R

Regions and Blocks in Drupal 7's Bartik Theme, 53f
Release mechanism, 6t
Reports, 45, 82, 96, 112
administrative interface, 93–94, 94f
Available Updates report, 101–102, 102f
modules and, 54–55, 55f, 88–89
views and, 61, 84–87, 85f, 86f, 87f
web server log files, 109–110
Robots.txt, 47
Roles, organic groups and, 8, 70–72, 71f
RSS feeds, 46, 65, 76, 81–84

S

Safari, 109

Safe computing, 101–103
Sandbox project, 89
Scalability, 6t
Schmidt, Aaron, 24
Scope of project, 31
Screenshot.png, 73
Script installation, 43
Search engine optimization (SEO), 82
Search phrases, 93
Security
  Drupal and, 43–44
  holes, 24
  malicious software, 50
  patches, 15, 117
  risks and, 33–34
  safeguards, 77–78, 101–102, 102f
Serials solutions
  article discovery integration, 115–116
Summon, 116
Server
  choosing, 1, 13–16, 14t
  cloud, 16–17
  log analysis, 109–110
Seven theme design, 25, 26f, 47t
SilverStripe, 6t
Site
  administrator, 29
  design, 9–10, 29–31
Skin, 9
  file system, 47t
  reskin, 26
  See also Themes
Social mechanics
  buy-in, 35
  change management, 34–35
  colleagues, 34
  committees, 35–36
  communications, 33, 36
  IT department, 33–34, 44
  staff, 34–37
  web team, 36
Social media
  commenting, 76–78

enabling, 76–78, 77f
RSS feeds, 81–82
sharing, 80–81, 81f
tagging, 78–80, 79f, 80f
Social OPAC. *See* SOPAC
"Social Share," 81
Software
development project, 13–15, 17–18
internal development, 19–20
malicious, 50
outsourcing, 18–19
See also Drupal; Hosting; Security
Sony Music, 34
SOPAC, 75, 115–116
Spam, 78
Springshare, 75
SQLite, 41
Staff
development and, 14f, 15, 17–19, 30, 34–36, 99
marketing to, 92–95, 94f
needs, 8–9, 22, 28–29, 70, 72
Stark theme design, 25, 47t
Statistics. *See* Measuring achievement
Styling, 72–73
Subversion (SVN), 48
Summon
article discovery integration, 115–116
Serials Solutions, 116
Sun Microsystems, 34
Support availability, 6t
Surveys, 111–112

T

Tagging, 78–80, 79f, 80f
Taxonomy Manager, 8
Technology Advisory Group, 35
Template.php, 73
Templates, 2–4, 9–10, 54, 80, 110
creating, 27
template.php, 72
themes and, 73–74
Testing, 94, 96, 97, 101–102
Bartik-test, 74

changes, 54–55
development, 18, 48
Institutional Review Board (IRB), 112
launch, 104–105
user, 23, 113
Themes
basic, 25–27, 25f, 26f
customization, 27, 74–75
default, 9–10
description, 9–10
installation, 52–54, 53f
structure, 73–74
style and, 73
up-to-date, 54–55, 55f
web root, 47–49, 47t
Timeline, 97, 99
Trends, future
API, 116
article discovery integration, 115–116
forthcoming, 116–117
modules, forthcoming, 115–116
Serials Solutions, 116
Social OPAC, 115–116
Tutorials, 88
Twitter, 49, 81, 96
TypePad, 78
TYPO3, 6t

U

Universally unique identifiers (UUIDs), 117
University of Michigan Library
categories, 82–83
committees and, 35–36
Harlan Hatcher Graduate Library, 3f
homepage, 3f
needs, 28
report, 94f
Serials Solutions, Summon, 116
website, 2, 23, 28
XML and, 75–76
Unix, 17, 40, 42, 48
Updates, 54–55
Available Updates report, 101–102, 102f

installation, 49–50
OpenURL resolver, 7–8
reports and, 45, 55f, 84–87, 85f, 86f, 87f
URL, 87, 97
User Experience (UX) Design for Libraries (Schmidt and Etches), 24
Users
ID, 108
marketing plan, 97–98
needs, 31–32
organic groups, 8, 70–72, 71f
roles, 9
testing, 113

V

Video, 75, 113
Views. *See* Modules

W

WAMP, 40, 48
Web Content Manager, 36
Web log files, 108–110, 109f
Web root, 47–49, 47t
Web Systems Manager, 36
Webform module, 67–69, 68f, 69f
WebGU, 6t
Website
accessibility, 10, 30, 43, 102, 115–117
categories, 82–83
configuring, 55–61, 57f, 58f, 59f
panels and, 65–67, 66f, 67f
views and, 61–65, 62f, 63f
content creation, 56–59, 57f, 58f, 59f, 60–61
content creation management, 2, 103–104
customize, 9–10, 14t, 24, 44, 73, 100–101
CSS and, 25
e-mail and, 69
installation and, 48
modules and, 27, 55, 88t
theme, 27, 74–75
design, 9–10, 29–31
development, 14t, 17–20, 101
FAQ interface, 9

Website *(cont'd.)*
feedback, 31, 67–69, 68f, 69f
social media and, 76–80, 77f
tagging, 79–80, 79f, 80f
filters, 63–64, 78, 85–87
functionality, 31–32
goals, 23–25
identity and, 2–3, 3f, 18, 22
interactive design, 23–24
LibGuides, 75–76
QR codes, 96, 97f
updating, 84–87, 85f, 86f, 87f, 117
user, 31–32, 108
web server log files, 108–109
See also Drupal; Hosting; Measuring achievement
"We Love Open Source Software. No, You Can't Have Our Code" (Askey), 89
Wget, 50
White House website, 34
Wiki, 15
Windows, 17, 40, 42, 44, 48
WordPress Import, 45, 76, 83

X

XML, 75–76

Y

YouTube, 75

Z

Zen theme, 10, 26f

About the Author

Kenneth J. Varnum has been the Web Systems Manager at the University of Michigan Library since 2007, a position that involves all aspects of the libraries' websites. In his 17 years working as a technology librarian, he has served in a variety of libraries, including a small special library at the Open Media Research Institute in Prague, a large corporate library at Ford Motor Company, and a small academic library at Tufts University. He received his MLIS from the University of Michigan and a Master's of Russian and East European Studies. Prior to his postgraduate education, he received a BA from Grinnell College. His research interests include open source software, discovery environments, and personalized information delivery systems.

Ken can be found on the web at http://www.varnum.org/ and on Facebook, Twitter, and Google Plus under the username varnum. His blog, *RSS4Lib*, at http://www.rss4lib.com/, is where he writes occasionally about technologies and libraries.

He lives in Ann Arbor, Michigan, with his wife and their two boys.