



**MEAP Edition
Manning Early Access Program**

Copyright 2008 Manning Publications

For more information on this and other Manning titles go to
www.manning.com

©Manning Publications Co. Please post comments or corrections to the Author Online forum:
<http://www.manning-sandbox.com/forum.jspa?forumID=504>

Part 1: Introducing Drupal

- 1 What is Drupal? Why open source? Why Drupal?**
- 2 Technical introduction to Drupal**
- 3 Features of modern, interactive websites**

Part 2: Fundamentals of publishing with Drupal

- 4 Basic publishing workflow with Drupal**
- 5 Views: easily present and format lists of content without writing database queries**
- 6 Using panels to create sophisticated landing pages**
- 7 User-recommended content**
- 8 Complete control of visual presentation**
- 9 Images and image galleries**

Part 3 Advanced publishing with Drupal

- 10 Advanced workflow with workflow/actions modules**
- 11 Syndicating content and data migrations**
- 12 Generating revenue**
- 13 Performance and scalability**
- 14 Managing repeatable site upgrades and deployments**
- 15 The Drupal community**

1

What is Drupal? Why Open Source? Why Drupal?

Get excited: Drupal is an amazing tool. You are probably already somewhat excited about Drupal – that's why you're reading this book. However, this book goes beyond the basics of Drupal and into the entire world of building and growing an enterprise publishing website. Perhaps you are about to embark on building a site and want some guidance on how to do it best or you already run a mailing list and blog about a particular topic but are unhappy about how it scales to handle multiple people. Drupal, and this book, are here to help you.

Drupal will help reduce the amount of time you spend to get your content online, increase the quality of your website, increase your ability to add new features, and do all these things with a limited investment of your time or money. The first three chapters give you a lot of background information about Drupal, the Internet, and Open Source software. It covers how some technical details and also gives insight into why they are increasingly useful in the creation of enterprise publishing websites. In the rest of the book we'll cover more advanced topics and get to Drupal in action.

1.1 What is Drupal?

Drupal is an Open-Source Social Publishing Platform written in PHP. It works with both MySQL and PostgreSQL databases and can run on any web server that supports PHP such as Apache, LightTPD, and Microsoft's IIS. Well, that's a mouthful. It also happens to be a great way to get your content online.

The project was started near 2000 as a way for Belgian college students to leave notes for each other and coordinate their use of a shared Internet connection. The same basic needs that those college students had - user management, permission levels, creating and publishing content - are fundamentally the same set of needs of nearly every website. So, in

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=504>

2001 Drupal was released to the world as an Open Source project so that other people could use and modify the source code to make it better. Fast forward to 2009 and the software is so widely used that there are conferences, location and interest-based user groups, companies, and even books being written about it.

1.2 What is Open Source Software?



Figure 1.1 A Typical Open Source Developer (cc-by <http://flickr.com/photos/jurvetson/70704300/>)

The processes of Open Source Software are a lot like termites building a termite colony, a process known as stigmergy. There is no centralized leader telling the termites where to build and what the colony should look like. Rather, one of them picks up a ball of mud, mixes it with pheromones as a signal to others that he is invested in this location, and then leaves it sitting on the ground. Other termites encounter this ball of mud and pheromones, decide that it would benefit them to have a nest, and do the same. Eventually many termites have gone through this process of finding the work of others and contributing a little bit more and together they have built a nest, though without any central coordination. It's a form of self-organization that provides a great way to think about Open Source. Humans have the benefit of e-mail, instant messaging, and VOIP to improve our communication. The Drupal project has a centralized Issue Tracker and code repository (our pheromone infused mud-balls) that keep changes to the codebase well organized.

One of the major events in the history of Open Source software (OSS) was the release of the first version of the GNU General Public License (GPL) in 1989. The GPL is a widely used license and Drupal itself is licensed under GPL Version 2 or later. The benefit of a common ©Manning Publications Co. Please post comments or corrections to the Author Online forum: <http://www.manning-sandbox.com/forum.jspa?forumID=504>

or general license is that everyone can be familiar with it and trust its contents without needing to review the license or hire a lawyer. It's common to view OSS as a "new" trend that sprouted from the 1989 GPL and flourished with GNU/Linux in the 1990s.

In fact, much of the original software distributed in the 1960s and 1970s was "open source," though the ideas and philosophy of the OSS community hadn't fully developed at that time. The reason we point out that software started as open source is to help understand that the collaboration involved in OSS is a natural result of the way that software works. Historically, our economy is based on what economists call rival goods. A typical bicycle is a rival good: if you are riding the bicycle then I cannot also ride it. In contrast, software (and music and movies and...) is a non rival good: Your use of a piece of software doesn't diminish the benefit I get from my use of it.

1.21 Software Written Collaboratively Around the World

So, software is a non-rival good and that means there's little reason not to share, but it also doesn't explain a motivation to share and especially not across international boundaries and cultural barriers. If we look at the history of Drupal it becomes clearer why and where these exchanges happen. In January of 2001 Dries released Drupal to the world under the GPL. Other users started downloading it and using it. They found ways to improve the software for their site and, since they had the source code, made those changes.

Then they had a choice: share those changes with Dries or keep the changes to themselves. If they kept the change to themselves then every time a new version of Drupal came out they would have to make that same change. There's a chance that in-between the change they made and the next version of Drupal that the code would have changed so significantly that what they wanted would no longer be as easy to achieve. On the other hand, it only takes a few seconds to share the code changes and resulting improvements using a "patch file," and standardized way of describing changes to source code. If it is agreed that the change is an improvement, Dries would apply it and the software would be better for everyone. The next release of Drupal would include your change so you don't have to worry about losing that functionality in the future. The developer or company who spent their time on the improvement is given credit which is useful for self-esteem and even for marketing. While this is a bit of a simplification, it clearly provides some of the motivations to collaborate. And, the proof is in the software: Drupal is but one of many thousands of open source projects all of which follow basically this same model.

WHAT DO WE KNOW ABOUT THESE THOUSANDS OF DEVELOPERS AROUND THE WORLD?

Looking at the Linux Kernel project, a relatively mature project, can help provide some rough indications about Open Source in general. In 2004 a review of contributors to the Linux Kernel found over 1,000 people who contributed a total of 38,000 "patches" to the project. Of those 1,000 people about 100 people are paid by their employers to work on Linux and

those 100 people were responsible for 37,000 of the 38,000 of the changes.¹ One conclusion is that while open source software is developed by "volunteers" most of the work being done is financially compensated.

In general, 1.5% of them are Female compared to 28% of the Proprietary Software world which is female². The Drupal project has reviewed its members and found 7% to be female.

1.2.2 Proven, reliable way of developing great software

Saying that OSS is reliable implicitly means "reliable compared to closed source." However, system software is a complex field. Comparing reliability numbers, feature sets, total cost of ownership, and other metrics is extremely difficult. Many people point out that OSS is better because it is "more secure" and yet for years the only operating systems to gain the United States of America Defense Department's "Common Operating Environment" certification were closed source and included Microsoft's Windows, a favorite punching bag for the security conscious. So, what is the reality? One indicator is that the major closed source companies have all decided to embrace Open Source to one extent or another. Oracle purchased and funds development of various MySQL database engines. Nokia has decided to turn their major phone operating system into an OSS program. In 2007 Microsoft hosted individuals from many OSS projects including Gabor Hojtsy and John VanDyk from the Drupal project at their Web Development Summit.

You may think that OSS is just for people on the fringe - that it is still only a very small number of people who rely on OSS every day. Ask your friends or extended family how many of them use open source software. Depending on your friends, you will mostly be greeted by looks of puzzlement and responses that "Of course I don't use that open source stuff. That's for techies." However, how many of them use Google? Yahoo? Hotmail? Linksys routers? Macintosh OS? An iPod or iPhone? Chances are that every one of them uses at least one of these products every day - they are all using OSS! And, are they satisfied? Generally speaking the answer is yes. Many individual OSS projects have matured to the point that they are as good or better than proprietary counterparts.

Part of the reason for the reliability of the software is the open process of feedback and peer review. As OSS proponent Eric S. Raymond says, "given enough eyeballs, all bugs are shallow." This manifests itself through public, early releases of the software where users are encouraged to report any problems they may find. The end result is that OSS is often extremely high quality, reliable software.

¹ http://www.gen.com/online/vol1_no1/26641-1.html?topic=daily-updates

² http://www.flosspols.org/deliverables/FLOSSPOLS-D16-Gender_Integrated_Report_of_Findings.pdf

1.2.3 Free (in terms of rights) and often free (in terms of price)

There are several terms that are used to describe OSS such as "free software" or "software libre" or even "free/libre/open source software". Using the wrong phrase can get you beat up in the wrong parts of town (admittedly an enormously geeky part of town). One of the many distinctions is between software that is "free as in free beer" or "free as in free speech." If you get a "free beer" then it costs you nothing, but you don't necessarily get the recipe to the beer along with it. People with "free speech" have rights and liberties which they can exercise. "Free" is primarily focused on these rights you have, but a big part of the attraction to OSS is that it often is available for little or no price.

How many times have you been using a piece of software and thought to yourself, or even said out loud, "I wish that it did {something else} instead of the way that it {does this}." If that's a piece of proprietary software the best you can do is send an e-mail to the support team or their website and hope that they'll make the change. Maybe they will, but most often they won't. No matter how much you care you generally have very little control in the process. The Freedom to make a change that you want is one of the real benefits of OSS.

If you're a developer, you make that change directly and hopefully pass back the change to the project. If you're not a developer, there is generally a large number of developers with familiarity with the project who can do the work for you. With proprietary software there is generally one company to call to make a change, or perhaps a few integrators who can make customizations. In successful OSS projects there are generally thousands of companies available to make customizations - it's the exact opposite of vendor lock-in.

1.2.4 New Economics of Open Source

OSS is always "free as in speech" and often "free as in beer." However, it's a mistake to believe that Open Source software has no costs. Several studies have found that the total cost of ownership over the entire life of the software compared between Microsoft Products and OSS alternatives are comparable. Many of those studies were funded by Microsoft, so once again, it's hard to know what to believe in the complex world of software.

One of the most common complaints about OSS is that it lacks documentation and support. This is largely a difference in perception between OSS and proprietary software. When you purchase Microsoft Windows you pay for the software but get a free user's manual and help pages on the Microsoft web page that are written by professional writers and sometimes can also call a toll-free customer support line if there is a problem with the software. When you download a piece of OSS (for free) you are much less likely to get documentation and if you do it is often lower quality than it's closed source version - forget a toll-free support number, though you may get some free support from other users of the software in a forum or mailing list. However, you can usually purchase a book which will explain the software and there are often paid support companies who will pick up the phone if you need help.

Which one is more expensive? Which support option is more responsive? Popular media and proprietary software companies have worked hard to convince you that OSS is confusing, that there is no support, and that it's only appropriate for geeks who enjoy spending weekends writing code. However, proprietary software has its flaws in the support and ease of use department. Who hasn't been enormously frustrated with Windows? Was the software support person you called helpful? Even the paid support from proprietary vendors is often low quality. In the end, perhaps the most important decision factor for people is what they are comfortable with. Therefore, the best practice is to extend your level of comfort with as many pieces of software as you can.

Drupal is a highly successful project in many of these areas. It generally has great documentation and good support all the way from free forums on drupal.org to many hundreds of consulting companies offering support and customization services.

1.3 What is a Social Publishing Platform? What is PHP?

In discussion about Drupal you may have heard technical jargon like "Social Publishing Platform," "Content Management System" and PHP. In this section, we'll review the fundamental concepts necessary for having a solid understanding of what Drupal really is and how it works.

1.3.1 Review of Key Terms

WEB BROWSER, HTML, CSS, PHP, CONTENT MANAGEMENT SYSTEM

If you're interested in building a Drupal website, you're likely already familiar with key concepts such as HTML, the web browser and web server. Each of these things plays an important part in accessing a website served by a content management system, so let's review these key terms before we talk about Drupal's benefits and guiding philosophies.

A content management system is software that enables people to create a frequently changing, or dynamic website.

NOTE

While the term social publishing platform lacks a standardized definition, it usually refers to a content management system that, in addition to facilitating the creation of content, enables multiple users to communicate and collaborate with one another using features such as wikis, blogs, forums and user profiles. We'll save the discussion of whether content management system or social publishing platform is a more appropriate title for Drupal for section 1.3.5, Social Publishing Platform, Content Management System, or Framework?

Each of the terms listed in the heading plays an important role in the process of viewing a website.

The browser, or web browser (such as Firefox, Safari or Internet Explorer), is a program that runs on the computer that you use to access the Internet. A browser works by interpreting HTML code that describes the contents of a website. When viewing a webpage, your web browser requests HTML code from the web server, which is, basically, the computer where the website lives. Modern websites also serve CSS code (cascading stylesheet), which control the visual appearance of a website. A content management system is software that runs on the web server and generates HTML code. PHP code runs on the web server, retrieves information stored in a database and generates the HTML which is sent to your browser.

While the end-product of a content management system – HTML that describes a website – might be identical to the HTML generated by a static website consisting of “flat” HTML files, the dynamic, database-driven nature of content management systems makes possible websites that are far more sophisticated and interactive than were previously possible.

1.3.2 Static HTML and Dynamic PHP

In the early days of the Worldwide Web, if you created a website, you probably had one or more files containing HTML code stored on your web server. Suppose we operated a website about marmalade. If we wanted to add a sentence about our newest marmalade flavor, we would have to open an HTML file containing the code describing our website, enter new information, save the file, and upload it to our webserver. This might require us to write additional HTML and CSS code in order to format the page correctly.

Many modern websites have content that changes frequently, and some, such as Youtube, Facebook and MySpace, have thousands and even millions of users interacting and posting content simultaneously. Websites where content changes frequently are considered dynamic websites. On a dynamic website, manually editing HTML files for every change to the website would be impractical, if not impossible. It would also require laypersons to learn HTML code, which many are not interested in doing. This is where a content management system, such as Drupal, comes in.

In section 1.3, we said that a content management system, or CMS, is software that enables interactive, dynamic websites. Generally speaking, a content management system replaces static HTML files by dynamically generating HTML on-the-fly when someone visits the website. The workflow for creating and editing content with a CMS is different from the workflow for a website consisting of static files. With a content management system, instead of interacting with the website by entering HTML code into a file, we use our web browser and are provided with a graphic user interface that (hopefully) makes managing our website easier, and maybe even more fun.

Content management systems like Drupal consist of two important parts, the database and the scripting language. The database stores all of the content for the website, such as

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=504>

blog posts, user accounts, and some of the configuration settings. The scripting language is computer code that runs on the web server, and contains a set of instructions for the content management system to follow when someone requests a webpage.

NOTE

There are many different types of scripting languages, but this book will talk primarily about one scripting language called PHP, since Drupal is built in this language.

1.3.3 The Relationship between PHP and HTML

To demonstrate concretely how PHP and HTML are related, let's use an example website and see how it might be handled by a content management system.

Supposing we build a website to share our marmalade recipes:

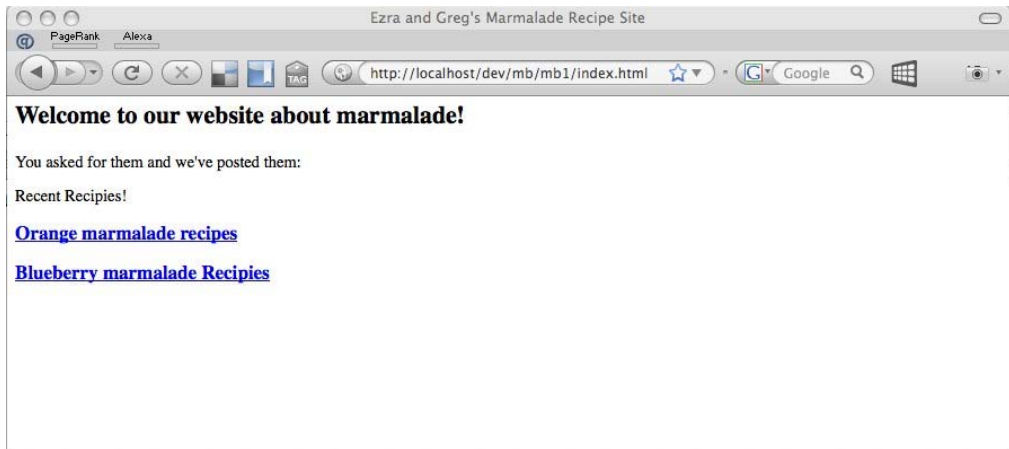


Figure 1.2: Our example Marmalade website.

Listing 1.1 The underlying HTML code for our Marmalade website

```
<html>
<head>
<title>Ezra and Greg's Marmalade Recipe Site</title>
</head>
<body>
<h2> Welcome to our website about marmalade!.</p> </h2>
```

©Manning Publications Co. Please post comments or corrections to the Author Online forum:
<http://www.manning-sandbox.com/forum.jspa?forumID=504>

```

<p> You asked for them and we've posted them:</p>
<p> Recent Recipies!</p>
<h3><a href="recipies/orange.html">Orange marmalade recipes</a></h3>
<h3><a href="recipies/blueberry.html">Blueberry marmalade Recipies</a></h3>
</body>
</html>

```

This code could be stored in a flat HTML file on the web server, or it could be generated with PHP.

Code listing 1.2 shows how PHP code might be used to generate the HTML code for this website. This code isn't from a Drupal website, but illustrates the fundamental concept at work in many content management systems, including Drupal.

Listing 1.2 PHP code generates HTML

```

<html>
<head>
  <style type="text/css" media="all">@import "style.css";</style>
<title><?php print $title;?></title>
</head>
<body>
<h2> Welcome to our website about marmalade!</p> </h2>
<p> You asked for them and we've posted them:</p>
<p> Recent Recipies!</p>
<h3><a class="<?php print $article1->class;?>" href="<?php print $article1->url;?>"><?php print $article1->title;?></a></h3>
<h3><a class="<?php print $article2->class;?>" href="<?php print $article2->url;?>"><?php print $article2->title;?></a></h3>
<h3><a class="<?php print $article3->class;?>" href="<?php print $article3->url;?>"><?php print $article3->title;?></a></h3>
<h3><a class="<?php print $article4->class;?>" href="<?php print $article4->url;?>"><?php print $article4->title;?></a></h3>
<?php print $destroy;?>
</body>
</html>

```

The difference between this version of our website's code and the last version, is that in this version, PHP code replaces some of the HTML on the server. When the client requests the webpage, PHP takes a variable, such as `$article1->title`, and looks in the corresponding location in the database to retrieve the title for article 1. PHP fills this information in and sends the HTML to the client. The result is that the client receives exactly the same HTML code as in our first example.

Let's look at a single line and see where PHP replaced PHP code with HTML:

```

<h3><a class="<?php print $article1->class;?>" href="<?php print $article1->url;?>"><?php print $article1->title;?></a></h3>

```

became

```
<h3><a class="orange" href="recipies/orange.html">Orange marmalade
recipies</a></h3>
```

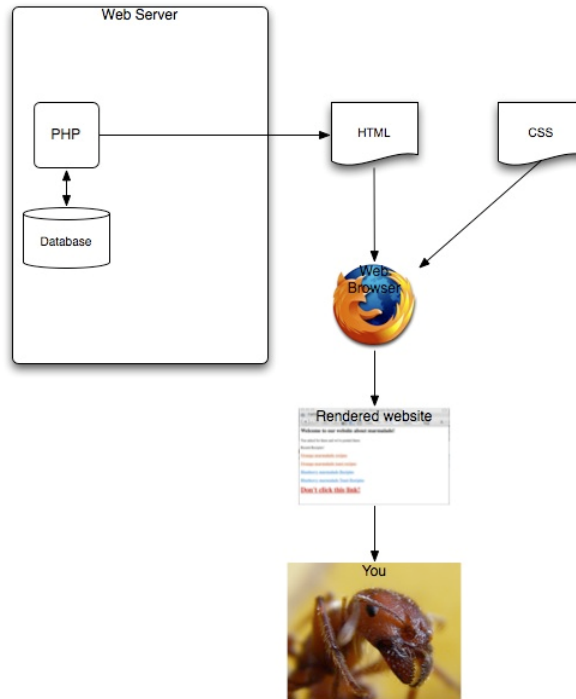


Figure 1.3: Page Request Cycle: PHP, Database, HTML and CSS producing a page.

1.3.4 The Relationship between Drupal and PHP

One question that is frequently asked about writing code within Drupal, is whether Drupal code is written in PHP or its own language. The answer is there are particular ways of using Drupal code that are specific to Drupal, and that Drupal is written in PHP.

One way of thinking about Drupal is as a system of APIs. API stands for Application Programming Interface, which simply means a standardized way for different computer programs, or components of one computer program, to communicate with one another. Drupal provides a great deal of functionality to users who build websites through Drupal's interface and do not have computer programming skills. We will discuss Drupal's APIs in more detail in chapter 2, a Technical Introduction to Drupal. Someone writing code to extend

or customize Drupal's functionality would write code in PHP, while making use of API functions provided by Drupal.

PHP can be thought of as clay that is used to make mechanical components, which are Drupal's API. Another metaphor could be that PHP is the atomic particles that make up the Drupal organism. Therefore in many cases with Drupal, including our PHP examples above, it is equally correct to say that PHP or Drupal takes a particular action and that our code is PHP code or Drupal Code.

Now that we've reviewed the fundamental concepts about how Drupal and other content management systems work, let's discuss some of Drupal's key attributes, benefits and philosophies.

DRUPAL IS A MODULAR SYSTEM

A module is a collection of files containing code that adds functionality to a Drupal website. Each module handles a particular domain of functionality and new features are often implemented by "connecting" one or more modules. This modular design is at the heart of what makes Drupal so flexible, extensible and powerful in the hands of site administrators and developers.

DON'T REINVENT THE WHEEL

You may have heard the expression, "Don't re-invent the wheel" used in explaining why Drupal is a better choice for building a sophisticated website than writing all of the website's code from scratch. That's because Drupal makes implementing features common to dynamic community websites and web applications more efficient by providing flexible systems for addressing these tasks.

For example: Suppose you are creating a website where one of the requirements is that users are able to create an account and log in. You could write lots of custom code, develop a database architecture, and test the code. Or, you can use Drupal's Core User system, which handles basic tasks relating to user account management, and is flexible enough to support new and customized features. When using Drupal's Core User module, loading a user's account can be done with the following single line of code.

```
$this_user = user_load((array('uid' => 23)));
```

Instead of investing lots of time building the basic components of your site, you can focus on other areas such as the important, custom features. And because Drupal Core receives the careful attention of over 1,000 highly skilled developers and designers, you can be confident that it runs efficiently and securely.

Custom websites and applications often require features that work in a specific, custom or otherwise unique way. Because Drupal's APIs are flexible, they can be used as the foundation for a wide range of truly custom functionality.

Re-inventing the wheel is a common metaphor for how developing with Drupal is more efficient than coding from scratch. Another way of thinking about building a custom website or web application is as if it were a custom automobile. It would require far more work, with little benefit, to custom mold the washers that are part of the automobile's engine, rather than building it out of existing components. Of course, this is something of an oversimplification; there are some cases when using Drupal might not be the best solution. But, Drupal is an excellent framework for building a wide range of custom web applications and websites.

1.3.5 Social Publishing Platform, Content Management System, or Framework?

We've just reviewed the basic components of a content management system. Drupal is one of many content management systems. However, you may have heard Drupal classified as something other than a content management system.

A common topic of discussion is whether Drupal is more appropriately considered a Platform, Content Management system, or Framework. While each of these terms has a unique technical implication they are all similar, and perhaps appropriate to describing Drupal.

The answer, until the Drupal project conclusively decides otherwise, is that none of these terms is singularly correct. Drupal offers a great deal of powerful functionality, some of which is valuable to end-users, such as built in blogging capability, and some of which is more important to application developers, such as its many APIs. Whether Drupal is a platform, CMS or framework depends on how you wish to look at it and take advantage of all that it has to offer.

1.4 Benefits of Drupal and Open Source: Cutting-edge, Robust, Extensible and Enterprise Ready

1.4.1 Drupal Core vs. Drupal Contrib: What's the Difference?

When discussing Drupal's features, it's important to understand that there are two different groups of code that make up the Drupal codebase. These two groups are Drupal Core and Drupal Contrib -- Drupal's library of modules and themes that have been contributed by community members. Drupal core provides a great deal of functionality by itself, or "out of the box," and it also provides APIs that make it extensible, so that developers can create custom modules that enhance its functionality. We'll discuss Drupal's extensibility more in the next section, as well as in section 2.4, Hooks: How Drupal's Modularity Works.

Everyone who uses Drupal is using Drupal Core. Many Drupal sites make use of different combinations of custom Drupal modules, either from the Drupal Contrib code repository or using site-specific modules.

Both Drupal Core and Drupal Contrib are open source software, and anyone is welcome to contribute patches or bug reports to either. However, because Drupal Core is the foundation upon which every Drupal site is built, the Drupal Core codebase and proposed changes to it generally receive a great deal more scrutiny by a larger number of people for each individual released version than do changes to Drupal Contrib. Only a few leading Drupal community members have the authority to finally approve and apply changes to the Drupal Core codebase, usually after a consensus has been reached about the change in question. The frequency with which Drupal Core is released, or its release cycle, is approximately once per year. Once a Drupal Core version is released, with the exception of security updates and minor bug fixes, it does not change. New features and major bug fixes must wait for the next Drupal Core version release.

On the other hand, Drupal Contributed modules and themes are generally released much more frequently. Usually, an individual or small team of people have jurisdiction over a contributed project and can approve changes. As a result, contributed projects receive new features and improvements more quickly than Drupal Core.

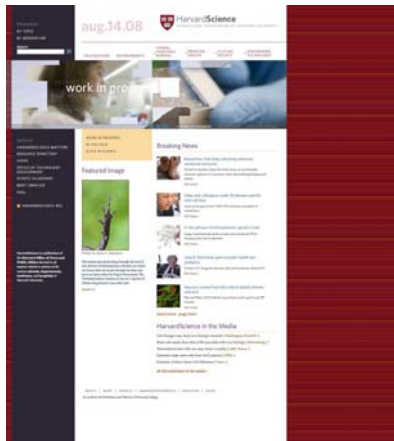
While Drupal Core comes with an impressive list of functionality, most Drupal sites use contributed modules to take advantage of and build upon that functionality. Therefore, when speak of Drupal in general, they are often referring to the combination of Drupal Core and Contributed modules or themes, as well as Drupal's thousands of contributors worldwide.

1.4.2 Enterprise-ready and Robust

Drupal is ready to be used for your enterprise-level website. Drupal's Core and extensive repository of functionality-enhancing contributed modules means that the features you need for your site or the foundation for building them likely already exists. A great deal of attention goes into making sure that given the proper resources, those features will perform efficiently under a high load. If the functionality you require doesn't exist yet, Drupal's extensibility makes it efficient to create that functionality yourself.

As we discussed in section 1.4.1, Drupal is highly extensible. That means that you can use it to build virtually any kind of website or application you can imagine.

Here is a list of just a tiny fraction of organizations whose either primary or subsite enterprise websites are running Drupal: Popular Science Magazine, FastCompany Magazine, The Onion, Macworld, MTV, Lifetime, Forbes, Minnesota Public Radio, Harvard, NASA, Fedex, Nike, Sony BMG and virtually all of its artists, Warner Brothers Records and many of their artists (Britney Spears), Sun, Novell, Ubuntu, Yahoo, Amnesty International, and the United Nations. Below are screenshots from websites that exemplify the diversity of advanced features and visual design possible with Drupal:



Harvard Science



The Onion Newspaper



Amnesty International



NowPublic.com

Figure 1.4 Screenshots of various sites using Drupal.

1.4.3 Extensibility, Rapid Development of Features

New modules and features are added to Drupal's Contrib repository far more frequently than to a proprietary content management system. That makes sense, because there are thousands of people collaborating to produce those features for Drupal, and usually only a limited, private group of developers working on a proprietary content management system. When comparing Drupal to an enterprise content management system, one is often choosing between enjoying the free benefit of the careful collaboration of thousands of developers, or paying (and waiting) for the development of features by a private group of people.

1.4.4 Ready for Web 2.0, 3.0 already in Mind

"Web 2.0" is often regarded as an overextended buzz-word, but it does have a useful, commonly agreed-upon meaning. Web 2.0 usually refers to website functionality that facilitates communication and collaboration between users of the website. Commonly editable Wikis, user profiles, forums, "friends lists" and other social-networking features are all examples of the so-called Web 2.0. With Drupal, there is no need to code these systems from scratch: they already exist and can be customized to your exacting specifications.

In fact, Drupal is already focused beyond Web 2.0. Whereas Web 2.0 is more focused on communication between individual website users, development has already begun on integrating so-called Web 3.0 features into Drupal, which call for more communication between websites by means of technologies such as RDF.

1.4.5 Great Support

Drupal's active user community provides excellent support and there are multiple avenues for support within the Drupal project. If you're having a problem with your website, it's likely that someone else in the community has experienced and reported that problem and possibly even posted a solution. Because Drupal is used around the world, the Drupal online chatrooms almost always have someone who is awake, and often times can help you or point you to where you might find support. Of course, paid support is available for Drupal from a wide variety of providers. For more about Drupal support, see Chapter 15.

1.5 The Development Process: Needs, Goals and Architecture

Drupal is so powerful that it can be tempting to simply dive in and start building things. This also satisfies our natural desire to explore. In general, it's best to limit this exploration work to a test site. Whether your project is just for you or it is for a large company it's still a best practice to follow a basic process while building your site.

1.5.1 Take a step back: What are the Needs? What are the Goals?

Before figuring out which features a website should provide, it can be helpful to take a step back and consider more generally, the purpose of the website. In other words, which needs will this website fulfill for the client, organization or target user community? For example, "communication with other users about upcoming and past events" is an example of a need that an organization might have. Having a list of needs in mind is helpful for developing an appropriate list of complementary features while remaining focused on the end goal: fulfilling needs for the website's (human) users.

Another technique for assembling a potential list of features is to use Actors and Use Cases. Actors and Use Cases are just technical words that system requirements people use for "groups of people who might visit your site" and "what kinds of things each group of people will do."

Creating this list will help guide you in making decisions about how to assemble the site. It's also a good idea to prioritize this list to determine which features are most important.

1.5.2 Architecting Drupal-based Solutions: One Tool - Multiple Uses

One of the key approaches in architecting Drupal solutions is to use existing modules and code to achieve custom functionality. This might sound somewhat contradictory -- How can one use existing code to achieve custom functionality?

What do a blog post, forum post, wiki, and article all have in common? They all have author, publishing date, title, and body fields. What makes a blog a blog is the presentation: Blogs show a stream of content with a title and teaser and the word "Blog" at the top of the page. Forum posts are generally shown in groupings by category with just their title, the number of comments, last comment date, and the author. However, at the individual page level the content of a forum thread is very similar to a blog post with comments.

Drupal's core "node" system is based on this generalized idea of content. Different types of content, which take many different forms, share a similar underlying data structure. This similar structure makes all types of content, including custom content types with custom fields, accessible to other modules in a uniform way. Drupal's node system is only one of many core and contributed APIs that utilize this approach. For more, see chapter 2, a Technical Introduction to Drupal.

While gaining familiarity with Drupal's important APIs and basic methodologies does require some learning investment, the benefit is well worth it. By creating modules which solve problems in a general way, Drupal's core and contributed authors have created an environment where you can work far more efficiently than by custom coding every feature from scratch, yet you are limited only by your creativity.

TIP: HOW TO CHOOSE MODULES AND THEMES

One of the major benefits of using Drupal is the rich feature set provided by Drupal's core modules and the broad set of capabilities available from the contributed modules. This

can also be one of the most frustrating aspects of working with Drupal when you have to decide which one of the modules to use when multiple modules, or sets of modules, might achieve the same feature. As you are evaluating potential solutions it's good to take a review of several characteristics of the modules involved.

First, how close is it to the features that you've imagined? A module that works for "images" may not handle images in exactly the way that you want. Or it may work fine for you but not as well for some of your site users who are less technical.

Consider the popularity of the module and how often it is updated. Look at the issue queue to get a sense of whether other people are using it and whether there is a community of users. Look at the "cvs messages" for the module to see when the last changes were made. While there are sometimes high quality popular modules which seldom need to be updated it is more likely that a module without updates is no longer maintained.

If possible, review some of the code in the module. Look to see if the developer uses the Drupal API functions for common tasks like creating links with the l() function instead of hand-coding string concatenation.

For more about evaluating Drupal modules, see chapter 15.

1.5.3 The Ideal and the Practical: Dealing with Infinite Possibilities

It may sound like Drupal is capable of anything and that is fairly true. There are currently over 3,000 modules available as part of the Drupal contributed repository, each of which may have multiple purposes. By the time you read this, there may even be more than 4,000 modules. But part of the reason is that Drupal is simply a modular collection of code, so you can always write more code to get whatever feature you need.

USE AN EXISTING SOLUTION

Where possible, it's almost always best to use an existing solution to a problem. It's quite likely that multiple other people have tried to build a feature similar to yours and then contributed their module back to the community. This wealth of modules is one of the most amazing aspects of the Drupal project.

EXTEND AN EXISTING SOLUTION

However, there are times when an existing module doesn't do exactly what you need. In these cases it is generally best to collaborate with the author of the existing module. Joining forces with them will generally provide the best feature set for the least amount of work, and

Drupal's modular nature makes it easy to "override" or "extend" existing code so that you don't have to create a fork.

DEFINITION:FORKS

Forks in software are much like forks in a road or path - it's a point where a single set of code breaks into two. Forks often happen when two people working on a single codebase can't agree about how to change the code and they decide to split apart the project into two projects. Sometimes this is a short-term problem and the users may merge them back together. Sometimes the projects may go on to have great individual success.

BUILD A NEW SOLUTION FROM SCRATCH

If there is no existing solution then you have to build the functionality yourself or hire someone to do it. Luckily the Drupal framework and APIs make it easy to rapidly build new modules and it's often possible to leverage existing modules to gain much of the necessary functionality limiting the actual development to "glue code" which fills in the feature gaps for what you need.

1.6 Why Write About Publishing?

You might be wondering, "If Drupal is so flexible, then why write about a single application such as publishing?"

The answer is that working within the context of Newspaper and Publishing websites allows us to refine the potentially infinite scope of discussion about Drupal's technical capabilities. Newspaper and publishing sites provide many excellent use-cases for some of the most important features and development techniques used in Drupal core and contrib. Using real-world examples also allows us to stay true to the hands-on learning approach employed in the rest of the In Action series.

Review: Drupal vs Proprietary Content Management Systems

	Drupal	Proprietary CMS
Acquisition Cost	Free	Typically starts at 6 figures
Yearly License Cost	Free	Often 20% of acquisition fee
Number of Users	Unlimited	Limited by license agreement
Cost, time spent waiting for new features	Often already available through contributed modules. If not, efficient to create based on core + contributed	At the mercy of vendor

	modules, Drupal APIs.	
Development Team	Thousands of developers around the world	Vendor's internal team
Upgrades and new features	Released for free You can help shape new features and help them happen faster.	Provider determines availability, timing and cost
Security Model	Dedicated security team audits code, receives confidential vulnerability reports from the community, issues free security updates. Drupal APIs eliminate many security risks.	Limited number of eyes focused on securing code.
Performance Improvements	Thousands of Core and Contrib maintainers work to make sure code is optimized out-of-the-box. Your choice amongst competition for additional server, Drupal optimization services.	
Usability improvements	Focused on empowering site administrators and end-users. Usability receives increasing levels of attention from developers, interactive designers, and usability testers.	Depends on preference of the provider and resources of the closed development team.
Pairs of Eyeballs	Tens of thousands around the clock	Depends on the size of the internal development team, almost always far fewer
Choice in providers	Your choice amongst International marketplace of providers	Vendor lock-in: All services provided by CMS provider
Potential for mastery of the software	All of Drupal code is available. Changes to Drupal APIs, functionality have been documented. Enormous body of written material about code and techniques.	Depends on provider

Costs	Installation, maintenance, new features	Acquisition of code, yearly cost installation, maintenance, security updates, new features, number of users, amount of content.
-------	---	---

1.7 Summary

A content management system or (social publishing platform) is a type of software used for building advanced websites. The web browser makes a request to the web server, which processes the data in the request using a programming language like PHP, and then sends HTML, images, and CSS back to the browser. The browser renders this information into a complete web page. Using an open source social publishing platform is a great way to get the cutting-edge, scalable features you need for your site.

Drupal is a modular, extensible, high quality, open source, social publishing platform written in PHP. Open Source Software is a reliable way of developing great software and often results in the creation of powerful, resourceful communities. While there is often no purchase price to obtain the software, you may end up paying for training or support. Drupal itself is free to download, free to modify, and has an amazing community of coders, documenters, and companies who can support you.

In the next chapter, we'll install Drupal and introduce its most important and widely used components. We'll also look at examples of some of the most important technical concepts that underlie Drupal's diverse functionality, so that we have a good grasp of how different components of Drupal interact, "under the hood."