SECTION I
Before You Start

- **LESSON 1:** Thinking Like WordPress
- **LESSON 2:** Planning Your Site for WordPress

1

Thinking Like WordPress

WordPress provides you with the tools to create, organize, and update your website content. Those tools function in specific ways, just as one type of word processing software has its specific buttons for creating, say, lists. But there's a difference between knowing which button to press to create a list and thinking about ways to use lists in your documents. That's what this lesson is about: learning to think like WordPress so that you can build or rebuild your website in an efficient and flexible manner from the start, and to use it in new and useful ways.

The driving principle behind this way of thinking is: *Store everything in the smallest possible piece; then assemble as needed*. It's the way of the digital world—photographs assembled out of pixels, data stored in database fields, or video recorded in bytes. WordPress operates with this kind of thinking, and you can make better use of its power if you think of your website and its content in this way.

STATIC VERSUS DYNAMIC WEB PAGES

If you right-click while viewing a page in your web browser, you'll see a tool called View Source, which displays the HTML of the page you're currently viewing. If you try this tool, it appears as though you're viewing a single file, but for most websites today, that's an illusion. In most cases there is no corresponding file sitting on a web server. Instead, the server has combined dozens and dozens of files in a split second to create what you're seeing with View Source.

That was not the case in the early days of the Internet, when most web pages were stored as single HTML files. The fact that no assembly was required to produce the code you see in your browser is why they are called *static files*. They're easy to create and they load quickly (an important factor at a time when computers and Internet speeds were slow), but they aren't flexible. Suppose you decided to change the logo at the top of each of your website's pages, and it had a new file name. With static files, you would need to manually

go in and replace the HTML in every file. Not so bad on a 5-page site, but what if you had 5,000 pages? Yes, there's such a thing as search-and-replace functions in HTML editors, but aside from the fact that methods like that are not user-friendly, they solve only one limitation of static files. Suppose you wanted an entirely different header area depending on what part of your site the visitor is on?

The answer is to break up the structure of web pages in such a way that different files control different parts of the final page. So instead of storing web pages as single files, the server would store a series of files that are then assembled into a single file at the moment the page is requested by someone's browser. It is this assembly process that leads us to refer to these types of web pages as *dynamic*. Figure 1-1 shows one way to split up a static HTML file.

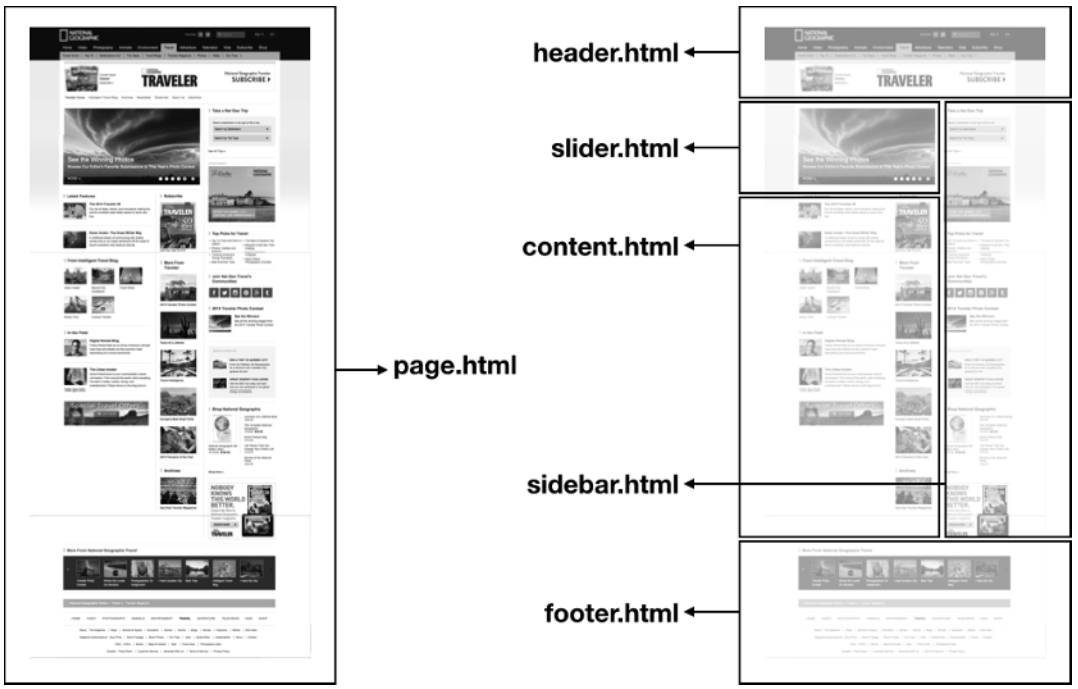

FIGURE 1-1

Notice in Figure 1-1 that the only file that would be unique to this particular web page is content .html. The rest of the files—header, footer, and so on—would be shared by the other pages on the site. So changing that logo for 5,000 pages would simply be a matter of changing the header .html file. If you can start thinking of your web pages in this way—as a set of parts that can be assembled on-the-fly in different ways—you're more likely to think of ways to use this ability to your advantage.

For example, it could be that the actual content of a web page (the material in content.html in Figure 1-1) might be broken down further to allow for greater flexibility. News stories, press

releases, or testimonials are good instances of this kind of content. Using testimonials as an example, you can see in Figure 1-2 how dynamic web page thinking could be applied:

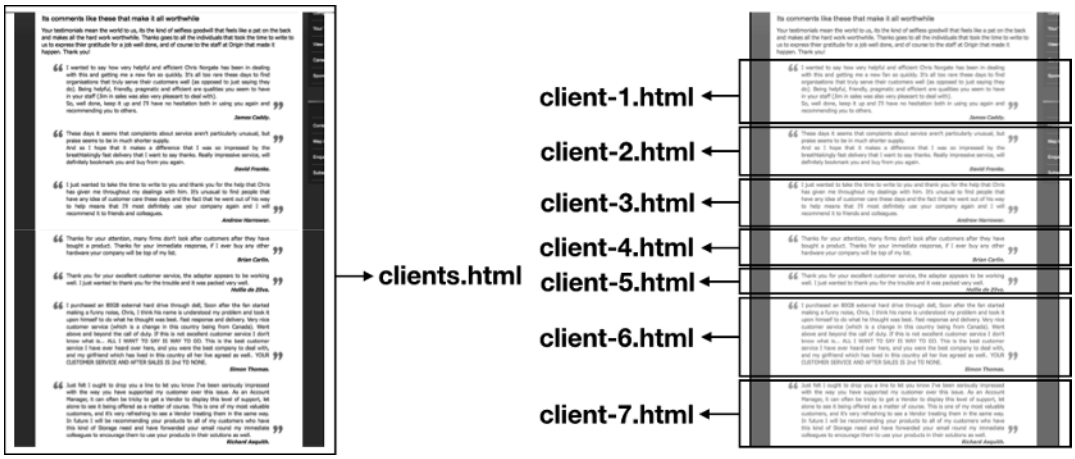

FIGURE 1-2

Although the value of dynamic web pages is obvious, the concept is not of much use to website owners unless the files required to run them are easy to manage. You could build a dynamic website—even a sophisticated one—with just a set of simple text files. But that would require the website manager to know HTML and other assorted languages, and to be comfortable working with tools such as file transfer programs. Moving those simple text files into a database to increase their flexibility only further complicates the work of the website manager. Enter the *content management system,* or *CMS*.

CONTENT MANAGEMENT SYSTEMS

Most of us tend to think of a CMS merely as a way to avoid having to learn HTML, but editing the text and media on a web page is just a part of what a CMS does. A content management system is a user interface for dynamic web pages.

Imagine for a moment you had a CMS that provided only a WYSIWYG interface for the full HTML of each individual web page. If you had a 5-page website that rarely changed, that might be enough. But suppose, even on a 5-page website, that you decided you didn't like the top section or header that appears on all the pages of your site. Yes, the CMS makes it easy to drag and drop a new graphic into the header area of the pages, but you'd still need to change the graphic on all 5 pages separately. Now imagine that task on a site with 500 pages or 5,000! Even with search-and-replace capabilities, you would need to upload all 5,000 pages back onto the server to replace the old version, and then do it all again for the next change. Ouch!

A CMS, however, is much more than a WYSIWYG editor. You want the CMS to keep separate all those elements of a dynamic web page that you saw earlier: the header, footer, sidebar, and so on,

and to manage not only their contents, but also how they interact with each other. A CMS instructs the server how to assemble any particular web page based in part on elements you control using its interface.

From the look of the page to which sidebar elements to include, a CMS provides ways for nontechnical users to control their web pages. The question then becomes: Does your CMS offer a lot of control and an easy-to-use interface?

Why Choose WordPress?

There's no shortage of content management systems these days—good ones—but the reason for choosing WordPress as your CMS is twofold:

- The simplicity and flexibility of WordPress's design make it easy to learn, easy to expand, and easy to customize.
- The WordPress community is so large and so vibrant that you have a huge number of add-on functions and designs to choose from, as well as excellent support, and will have for years to come.

It's important to keep in mind that no CMS can fulfill everyone's needs right out-of-the-box. The more a CMS tries to be all things to all people, the more bloated it becomes, and that means a steeper learning curve and a greater chance it will break down. A good CMS follows the principles of digital thinking and keeps as many elements separate as possible, meaning each one is fairly simple but when assembled offers great power.

WordPress is built on this principle. The core software does only basic functionality, to which you then easily add other functions as you need them or remove them when you don't. The look of WordPress is entirely separate from the core software, so it, too, is easily changed. And all these elements outside the core can be modified or new ones can be created to match your exact needs.

But even the best CMS is only as useful as the community that supports it, and WordPress has community. Whether there is someone building new add-on functionality, offering advice in forums and blog posts, or selling development services, WordPress is the most-supported CMS on the planet.

How WordPress Assembles Pages

Part of thinking like WordPress is having a general grasp of how it works. There are four elements of WordPress that interact to create HTML pages: the *core files*, the *theme files*, *plugin files*, and the *database*.

The core is the set of files that you download from `WordPress.org` that provide not only the basic functionality, but also the coordination between all the other elements.

The theme files have two key functions: Provide the set of HTML files that assemble the final web pages, and control the design of those pages.

Plugins are groups of files that add more functionality to WordPress. Some plugins consist of a single file, whereas others can have dozens or even hundreds.

The database has several functions. It keeps track of all the parameters of your WordPress installation, from which themes and plugins you have to the preferences of each individual user. It also stores the text portion of your content. When you write a blog post, for example, it is stored in the database, along with any references to media files, which are stored in folders on the server.

Figure 1-3 shows a simple diagram of how these four elements interact.

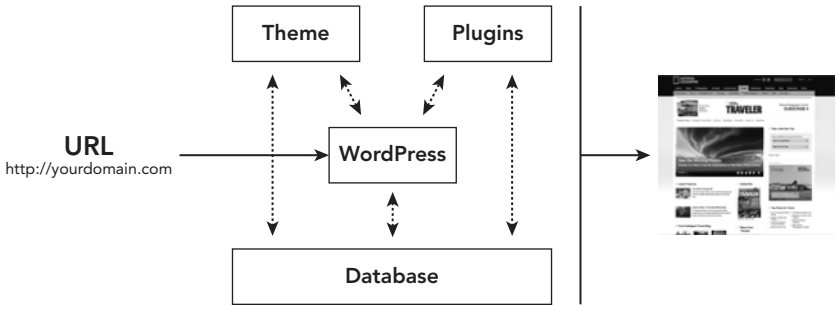

FIGURE 1-3

The arrows going back and forth between the elements begin to demonstrate the incredible amount of interaction required to generate a web page in a matter of seconds. The number of requests or queries made to the database averages approximately 40 or 50 for a typical WordPress page. Dynamic web pages are not cheap. They can place a heavy load on server resources if a lot of people try to access the site at one time. That's why many sites use a caching system with their CMS, which means that snapshots are taken of each page and stored as single, static HTML pages, avoiding all the back and forth between files and the database. The caching system keeps track if any changes are made to a page and takes a new snapshot as needed. That way you have the advantages of both dynamic page generation and static page serving.

The advantages of keeping these four elements of WordPress separate are many. Changing the look of your site is as simple as changing the theme. If a plugin starts causing problems for your site, you simply remove it and plug in a new one. If a new social media platform becomes the next big thing, someone will come up with a plugin to interact with it—or you can have your own made. When WordPress needs updating, your site's options and preferences remain untouched in the database. And if your host gives you poor service, you just copy all the files and the database, and move them to a new server.

But, although WordPress by its nature as a CMS and through its particular design produces dynamic web pages, its built-in tools can go only so far. Sites using WordPress can be more or less dynamic in nature depending on how the user works with WordPress's tools. Some of the power and flexibility in your site comes down to you.

YOUR ROLE IN MAKING YOUR SITE FLEXIBLE

As mentioned earlier, a good CMS needs to have a simple user interface, and WordPress lives up to this requirement. Actually, it was that ease of use that first led me to use WordPress on my clients' sites. Even as it has grown more complex, the developers of WordPress have continually worked to keep the interface user-friendly.

The menu system, for example, enables an unlimited number of menus; each can handle dozens and dozens of menu items with multiple levels of drop-downs, and those menu items can be virtually any type of content within WordPress: pages, posts, categories, tags, and more. Yet, as you can see in Figure 1-4, all this complexity is handled with simple check boxes, drag-and-drop interfaces, and drop-down selections.

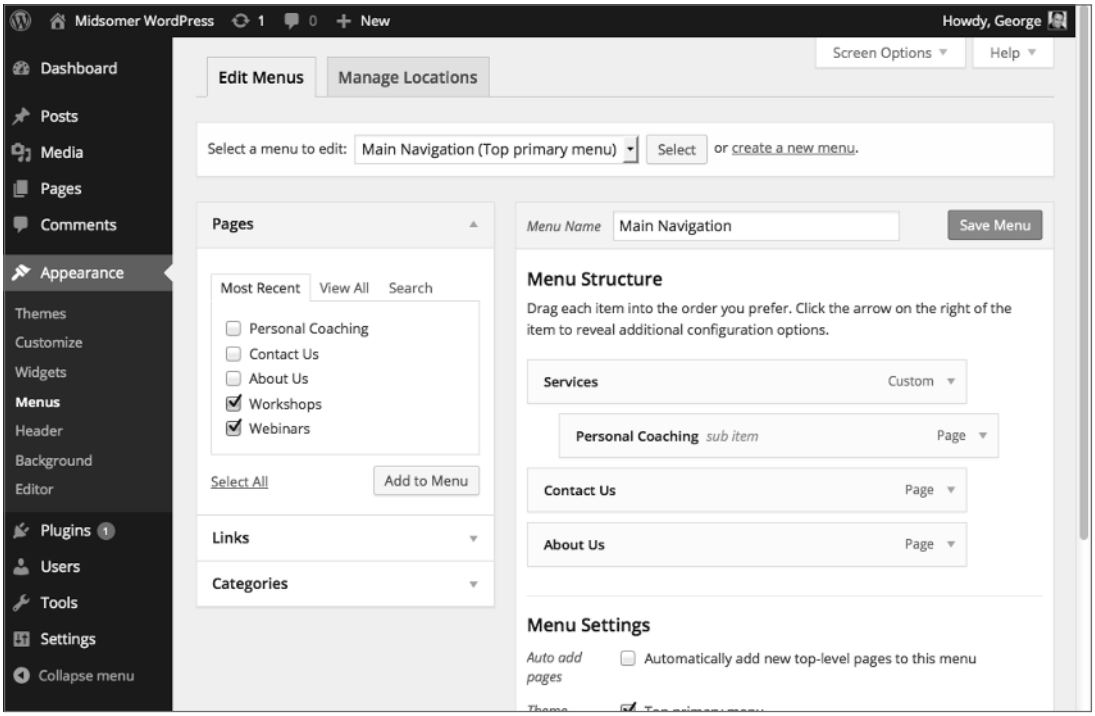

FIGURE 1-4

But with great power comes great responsibility. You're the one who has to create a useful and easy-to-follow navigation system for your users; WordPress cannot do that for you. It just makes it easy for you to do the creating. The more you can understand what's possible with WordPress, the better you'll become at making use of its powerful tools to create the best website for your visitors.

Understanding the difference between two ways of handling content in WordPress is particularly important to making your site easier for you to use and more useful to your visitors, and that is the difference between posts and pages.

Posts and pages share a lot of similarities, in particular the way their user interfaces or, in WordPress terminology, their admin screens work (which means there are many functions you'll have to learn only once). Where they differ brings you to the heart of the principle in this lesson: Store everything in the smallest possible pieces.

Take testimonials. For many websites it is absolutely crucial to have testimonials from satisfied customers or clients. So you grab those e-mails and letters people have written praising your services and you open up WordPress. You want a web page of all those testimonials, so under Pages in the WordPress menu, you select New Page. The WYSIWYG editor makes it easy to copy the text over from your e-mail or Word document, add any extra formatting, and get the page looking nice indeed.

Fast forward 6 months, and you decide that you would like to group some of the testimonials onto another web page because they all relate to a particular service. Not so bad; you just copy the text of the page, create a new WordPress page, paste it in, and then delete the testimonials you don't need. Then a month later you get a new testimonial that belongs on both pages. You copy and paste into both. But suppose you have several categories of testimonials; you can see how this could get both tedious and complex to manage.

Remember earlier in this lesson the diagram of the testimonials page (refer to Figure 1-2) and how you might break it up instead into individual testimonials? Well that's what posts in WordPress are for: groups of related content that you can categorize. That's all we mean when we talk about a blog: groups of similar content. So now, instead of putting all the testimonials in a WordPress page, you can create a category of posts called Testimonials and enter individual testimonials as posts, assigning them to that category. Then put the category on your site menu, and when visitors click that link, WordPress gathers all the relevant posts and outputs them as a single web page.

Fast forward 6 months again and you decide you need a category of testimonials for a particular service. Just create a subcategory of Testimonials called, say, Workshop Testimonials. Through an easy WordPress interface, you can assign 2, 20, or 200 testimonials to that new category with a single click. Then you just put the Workshop Testimonials on your menu or create a link to the category wherever you want. When visitors click, they see only testimonials about your workshop.

By keeping those testimonials individual, the possibilities are endless. Need rotating testimonials on the front page? Want to feature a particular testimonial on the sidebar of a specific page? It's only possible by keeping them as separate posts. That's the power of thinking like WordPress. In the next lesson you start applying that thinking to a sample site, and throughout this book you see ways to use WordPress's tools to increase the power and flexibility of your site.

> **NOTE** *Okay, having just told you how posts and pages differ in WordPress, I use the term* posts *throughout this book to mean both posts and pages. Partly, it's to avoid potential confusion with the term* page*, but mainly, it's for the sake of simplicity. The way you enter and edit content for posts and pages is basically identical because they both share the majority of content management features. Where necessary, I'll distinguish between them, but unless I do you can assume that when I say* posts*, I mean both posts and pages.*

TRY IT

There isn't anything specific to try based on the material in this lesson, but one thing you can do is examine your favorite news website, and in the content area of the site, try to image how those pieces of content might be separated in the site's CMS. Then go to another page, compare what's common with the previous page, and try to imagine how the builders have divided up the structure of the page—map it out on paper.

REFERENCE *There is no video to accompany this lesson.*