Part I: Before You Start

1

Thinking Like WordPress

WordPress provides you with the tools to organize your website content, but 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 you can use lists in your documents. That's what this chapter is about: learning to think like WordPress so that you can organize your content in an efficient and flexible manner right from the start, and be able to use it in new and useful ways later.

Dynamic vs. Static Websites

When you open a website in your browser, you see a single page filled with text and media (graphics, photos, video, and so on) like the page in a magazine or newspaper is a single entity made up of text and images. But what you see in a browser window is created from a series of instructions: the HTML code. So ultimately the HTML is the single entity behind what you see onscreen; the equivalent of the printed page.

However, there's an important difference between an HTML page and a printed page. The HTML that's fed to your browser may be a single entity when it arrives at the browser, but it may or may not be a single entity sitting on the server waiting for browsers to retrieve it, like a magazine on a newsstand waiting to be purchased. The HTML may be made up of chunks of code that get assembled into a whole in that split second when the browser pulls it off the shelf.

That's the difference between dynamic and static web pages. Static pages are complete sets of HTML waiting to be retrieved, whereas dynamic pages are chunks of HTML that are assembled at the moment of retrieval into a single entity that's displayed in your browser (some systems store the most recent static version of a dynamically created page to keep the server from being overworked, but ultimately the browser pages were created dynamically).

What I want you to take away from this lesson in particular, but the book in general, is to reject static thinking in favor of dynamic. You might have a vision right now for the content of a particular page on your website, but if you learn to view the content in chunks, there may be ways to use part of that content on another page as well. *Dynamic thinking* means you want to keep that chunk of content separate and reusable, not welded to the other content.

Content Management Systems

Creating HTML pages dynamically is one half of what a content management system (CMS) does: it takes chunks of code (your content) and pieces them together into a single HTML page. The other function of a CMS is to provide an easy way for you, the user, to manage all those chunks of content.

Managing content does not just mean allowing you to enter text or upload images; it also means making it easy for you to determine the relationships between chunks of content. Selecting a category for the article you're working on, for example, tells the CMS to assemble that chunk in a particular way when someone on the Internet requests a page on your website.

Everybody understands the role of a CMS when it comes to managing content: it saves having to know HTML coding. But why not just have the CMS manage the content of individual HTML pages? All this assembling business seems like a lot of extra work. If you had a five-page website that never changed, that might be true. But suppose, even on a five-page website, that you decided you didn't like the top section or header that appears on all the pages of your site. Although a CMS for static pages would make it easy to change, you'd still need to change the graphics on all five pages separately, because they're all individual, physical pieces of coding. Now imagine that task on a site with five hundred pages or five thousand! Even with search-and-replace capabilities you would need to upload all five thousand pages back onto the server to replace the old version, then do it all again for the next change. Ouch!

By separating the content of individual HTML pages into chunks, a CMS offers tremendous flexibility. Say you wanted three thousand of your pages to have a different kind of header than the other two thousand. Easy, with a CMS. What if your business partner decides that your line of five hundred different wuzzbuzzes should be categorized under buzz instead of wuzz? Easy, with a CMS.

We're always being told to embrace change, and one of the advantages of a website over print is that it allows you to change things as much as you want, as often as you want. The advantage of using a CMS instead of manually creating manual or dynamic web pages is that the managing of change is much easier and more flexible, which is exactly what WordPress does.

WordPress as a CMS

Even if you get the bit about dynamic thinking and managing chunks of content, you might still be asking yourself: isn't WordPress blogging software? Yes, and blogging software is nothing more than a CMS that's geared toward a type of website structure––the blog. Real estate CMSs, for example, are set up to manage the kind of structure and content that you typically see on a real estate site. Online stores are managed by CMSs that organize content into catalogues or what we call shopping carts.

But I don't want a blog, you say, so why would I use WordPress for my website? Fair question. Part of the answer is that you could use any CMS to build any website; it's a matter of how much work it would take to do it, how much customization, how much training to use the interface in a way it was not intended, and so on. The rest of the answer is that WordPress's design — the simplicity and the flexibility — make it an ideal CMS for a huge variety of uses. Yes, there will need to be creative thinking, sometimes add-on software, sometimes customization of the coding, but if that weren't needed, we'd be talking about a custom CMS for every website.

The point is that all websites have a lot of common elements that may have different names and different functions, but from the standpoint of HTML coding they operate in basically the same way. For instance, I need a page full of testimonials whereas you need a page of all your current specials. If a testimonial and a special are the chunks of content, all we need the CMS to do is assemble our chunks into whole pages. Your header and footer may be very different in look and content from mine, but we both need a header and a footer. A good CMS could care less which is which––it just assembles and manages, easily and efficiently. Like WordPress does.

How WordPress Assembles Pages

Three basic structures in WordPress interact to create HTML pages: the *engine*, the *theme*, and the *database* (where content is stored). What I call the engine is the set of files that perform the tasks of storing, retrieving, and assembling content. The database is where the content is stored and the theme is made up of template files that provide instructions to the engine about what to retrieve and how to assemble it, as I've tried to illustrate in Figure 1-1.

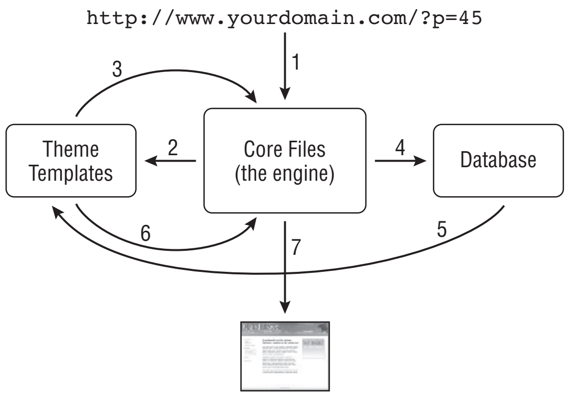


Figure 1-1

The web address you type into a browser window goes to the WordPress engine and tells it which template file to look for in the theme. The engine then reads the template file and follows the instructions about what chunks of content to retrieve. Depending on the complexity of the template, there may be dozens of chunks to be located in the database where they're stored (technically, not all chunks are physically stored in the database, but at least the information on how to find them is stored there). Having found the content, the engine then assembles the chunks according to the template's instructions and you see the result as the HTML page in your browser. And of course all of this has happened in a split second (or two).

Why Separate Is Good

You saw earlier why it's important that a CMS keep form (design and structure) and content (text and media files) separate, and now you're seeing the particular power of the way WordPress achieves this. Remember that earlier example of wanting three thousand pages to have one header and two thousand pages a different header? Depending on exactly how WordPress generates those pages, you might have to add only one template file to your theme to accomplish the change.

If you want to see a dramatic example of how the separation of form and content works on the Web, visit a site called CSS Zen Garden (`www.csszengarden.com`). You can instantly switch between dozens of incredibly different looks, all presenting exactly the same content.

But separating form and content isn't the only useful kind of separation that WordPress employs. It also separates the form from what I've called the WordPress engine — the set of files that do the actual assembling and managing. That engine is completely separate from the theme and the content, which is a good thing from a number of standpoints, the most important of which is the ability to easily update the engine.

Software of any kind is constantly being given new features, strengthened for security, made more efficient, and so on. If you had to completely redo your theme every time the engine needed an update, it would be very inefficient, just as having to redo your website content because of a new structure or look would be inefficient. As I said earlier, WordPress at its heart is a set of three separate structures — the engine, the theme, and the content (in a database) — each of which can be tweaked, updated, or completely replaced, all independently.

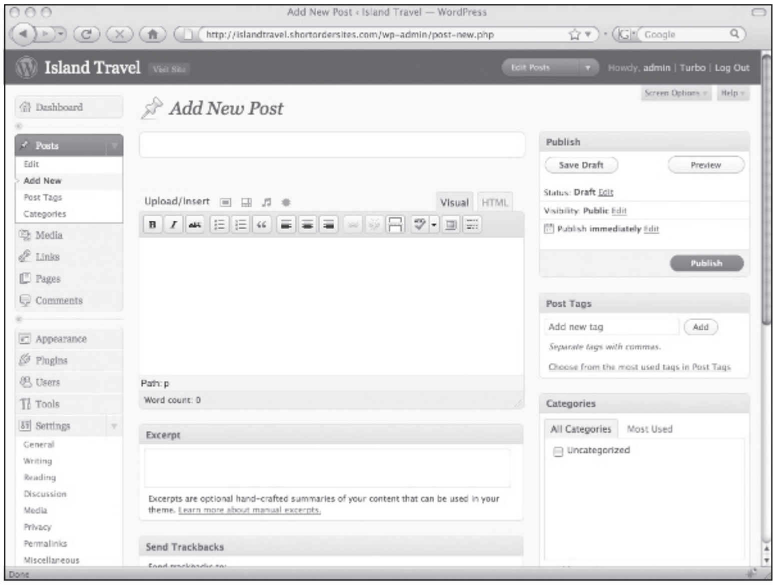
There's a fourth separate structure to WordPress that is entirely optional: plugins. These are bits of extra code that you literally plug into the WordPress system and they provide additional functionality, from letting people rate the content on your site to automatically creating tweets on Twitter.

Sometimes people ask why they don't just incorporate the plugins into the engine, but that would be defeating the whole purpose of this elegant and flexible system. To begin with, plugins are meant to address specific needs. Why clutter the engine with features that not everyone uses? Sometimes a plugin is so useful to everyone that it is eventually incorporated into the engine, but most plugins aren't like that. Also, the more complex the engine, the better the chance things will break down. Keep the engine simple and add on extras as you need them. I have some WordPress sites with only two plugins, and others have dozens.

Another reason for keeping extra features as plugins is that there can be many variations of a plugin, each one serving the needs of a group of users. A good example would be plugins for photos — some are very simple, some are very complex, some work better than others. Having a choice of those plugins, rather than being stuck with only one, is another important advantage.

How WordPress Manages Content

Very easily, thank you. Like any CMS, WordPress stores the chunks of content it uses to assemble HTML pages in a database. Getting that content into the database, letting you edit that content, and then storing instructions about how that content relates to other content is really what managing the content means. All databases work pretty much the same way, and though part of WordPress's simplicity and flexibility stems from the way its creators built the database and the files to run it, what ultimately matters to users is the interface that's used to do the managing. It's this administrative interface (a sample screen is shown in Figure 1-2) that my clients and hundreds of thousands of users around the world find so easy to use — for them, it *is* WordPress.



Status: Draft Edit
Visibility: Public Edit
Publish immediately Edit

Separate tags with commas.

Choose from the most used tags in Post Tags

Excerpts are optional hand-crafted summaries of your content that can be used in your theme. Learn more about manual excerpts.

Figure 1-2

Every CMS has its particular way of dealing with content and though WordPress is extremely easy to use, you still need to understand how it refers to content and the methods it uses to organize content. Take *posts* for example. In the world of blogging, people refer to the act of creating a new blog entry as posting. So it's not surprising that the primary kind of content chunk in WordPress is called a post, but that doesn't mean we have to use WordPress posts exclusively for a blog. A post is just a block of text and some instructions stored in a database. They could just as well have called them *chunks*. We don't want to get tied to how we use posts simply because they were originally intended for and named after an element within blogs.

WordPress has another type of content chunk called a *page*, but not the HTML pages you see in your browser. Like posts, WordPress pages are essentially blocks of text and accompanying instructions stored in a database. They're different from posts, though, in several ways. For a start, you can put only one WordPress page at a time into the final assembled HTML page. On the other hand, you can have dozens or even hundreds of posts displayed on a single assembled HTML page.

Suppose you set up WordPress so that each press release for your company is entered as an individual post. Then, you tell WordPress to show the five most recent press release posts. Whenever you add a new press release, it goes to the top of the list. On the other hand, the content describing your company's mission statement doesn't change that often — it's static in comparison to press releases — so you set up a WordPress page for that content. That's how you'll hear people describe the difference between posts and pages: one is for dynamic content and the other is for static content. *The main thing is not to confuse a WordPress page with the final HTML page that gets generated and viewed by the public*. WordPress pages and posts are both chunks of content that just get utilized in different ways.

There's another important difference between posts and pages: posts can be categorized whereas pages cannot. Pages can be a sub-page of another page, but it's a very limited relationship. There's a lot you can do with categories as you see later, but I'll mention one here: a post can be placed in multiple categories at the same time. That has enormous consequences for how you use posts. It makes it very simple for the content of a post to appear in several or even dozens of places on a website.

For instance, if I don't go to the press release area of your site I won't see your announcement of a new wuzzbuzz for kids. But if the post for that press release also appears in the products section of the website, as well as in a section on helpful tips for keeping kids busy, it's more likely I'll notice this new product. Yet, you only had to enter that press release once and assign it to several categories. WordPress then automatically displays it in multiple locations, saving you the time of entering the same information two or more times, let alone having to remember all the places on the website where that information is needed.

Okay, having just told you how posts and pages differ in WordPress, I'll be using the term posts *throughout this book to mean both posts and pages. Partly it's to avoid potential confusion over the term* page*, but mainly it's for the sake of simplicity. The way you enter and edit content for posts and pages is virtually 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.*

Now that you're thinking like WordPress, it's time to take an actual website plan and see how it can be organized using WordPress, which is what I'll do in the next lesson.

Try It

There isn't anything specific to try based on the material in this lesson, but one thing you could do is examine your favorite news website and count how many different chunks of content are on one page. Then go to another page on the site and think about 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.

There is no video to accompany this lesson.