

# 1

## First Post

### WHAT'S IN THIS CHAPTER?

---

- Appreciating the provenance of the WordPress platform
- Choosing a suitable platform for your WordPress installation
- Downloading, installing, and performing basic configuration of WordPress
- Diagnosing and resolving common installation problems

If displaying “Hello World” on an appropriate device defines minimum competence in a programming language, generating your first post is the equivalent in the online publishing world. This chapter provides a brief history of WordPress and then explores several options for hosting a WordPress installation. Common miscues and misperceptions along with their resolutions round out the chapter and put you on the edge of publishing your wit and wisdom.

Once you’ve installed, configured, and completed the barebones administration, you’re ready to take advantage of the code walk-throughs and detailed component descriptions in later chapters. Of course, if you already have a functional WordPress website, you can skip this chapter, and dive in headfirst to explore the core code in Chapter 2, “Code Overview.”

### WHAT IS WORDPRESS?

WordPress is one of the most popular open source content management systems available, with global and vibrant user, developer, and support communities. While it can be compared to Drupal and Joomla as a user-generated content workhorse, WordPress distinguishes itself with a broad array of hosting options, functional extensions (plugins), and aesthetic designs and elements (themes).

With the rise of self-publishing, low-cost web hosting, and freely available core components such as the MySQL database, blogging software followed the same trend as most other digital technologies, moving from high-end, high-cost products to widely available, low-cost consumer or “hobbyist” systems. WordPress isn’t simply about creating a blog so that you can have a digital diary attached to your vanity URL; it has evolved into a full-fledged content management system and burgeoning application development framework used by individuals and enterprises alike. This section takes a brief tour through the early history of WordPress and brings you up to speed on the current release and user community.

WordPress started similarly to many other popular open source software packages: Some talented developers saw a need to create a powerful, simple tool based on an existing project licensed under the GPL. Michel Valdrighi’s b2/cafelog system provided the starting point, and WordPress was built as a fork of that code base by developers Matt Mullenweg and Mike Little. WordPress first appeared in 2003 and was also built on the MySQL open source database for persisting content with PHP as the development platform. Valdrighi remains a contributor to the project, which is thriving as it has a growing and interested community of users and developers.

As with other systems written in PHP, it is self-contained in the sense that installation, configuration, operation, and administration tasks are all contained in PHP modules. WordPress’s popularity has been driven in part by its simplicity, with the phrase “five-minute installation” making appearances in nearly every description or book about WordPress. Beyond getting to a first post, WordPress was designed to be extended and adaptable to the different needs of different people.

WordPress today is supported by a handful of core developers and many key contributors. Mike Little runs the WordPress specialty shop [zed1.com](http://zed1.com) and he contributes the occasional patch to the code. Matt Mullenweg’s company, Automattic, continues to operate the [wordpress.com](http://wordpress.com) hosting service as well as fund development of related content and site management tools, including Akismet, multi-site WordPress, Gravatar, and most recently plugins such as JetPack. Akismet is a robust, Automattic-hosted spam detection and protection service with a statistically (and incredibly) low failure-to-detect rate. Previously known as WordPress MU, multi-site WordPress functions are at the heart of the [wordpress.com](http://wordpress.com) hosting system and are now merged into the main WordPress source tree. Gravatar dynamically serves images tied to e-mail addresses, providing a hosted icon with a variety of display options. Think of it as a service to make hot-linking your profile picture technically and socially acceptable. JetPack is a multifunction plugin offering a vast array of common needs for the website owner. The JetPack plugin is covered further in Chapter 16.

As a content management system, the WordPress system definition does not stop at time-serialized posts with comments. BuddyPress is a set of themes and plugins that extends WordPress into a functional social networking platform, allowing registered users to message and interact with each other, again with all content managed within the WordPress framework. Similarly, bbPress is a PHP- and MySQL-based system designed for forums (bulletin boards) that is distinct from WordPress but is commonly integrated with it.

Chapter 16 covers some of the WordPress adjunct systems in more detail, but they are included here to provide a sense of how WordPress has expanded beyond a basic single-user-oriented tool. At the same time, we are not endorsing or making a commercial for Automattic, but delving into the

guts of WordPress without a spin of the propeller hat toward Mullenweg and Little is somewhere between incorrigible and bad community behavior.

## POPULARITY OF WORDPRESS

This book is based on the WordPress 4.1 major release, but really focuses on foundational WordPress tactics. Each successive release of WordPress has included improvements in the administration and control functions (Dashboard); backup, export, and import functions; and installation and upgrade features. Even if you start with a slightly down-rev version of WordPress, you will be able to bring it up to the current release and maintain the freshness of your install. Install and upgrade paths are touched on later in this chapter. But just how popular is WordPress?

### Current State

Interest in WordPress and WordPress usage is booming. You're holding in your hands a testament to that. Just four years ago, very few WordPress books were available. Now this third edition has been published. "Popular" is always a subjective metric, but statistics add some weight to those perceptions. According to Automattic, as of 2014, tens of thousands of new WordPress sites are created every day (<http://en.wordpress.com/stats/>) not including standalone self-hosted WordPress sites. That includes sites using WordPress for content management, blogging, and personal rants, and has to be discounted by those of you who have multiple WordPress installations to their names, but even with that estimate of the order of magnitude, WordPress is immensely popular. Automattic no longer discloses how many sites they host on WordPress.com, but in 2012 they reported nearly 74 million WordPress websites globally with about half of them hosted at WordPress.com, and in 2010 that number was at only 5 million sites. In 2008, the official WordPress plugin repository hosted over 6,300 plugins, double the number from 2007. In 2012, the second edition of this book cited 19,000 plugins in the repository, and at the time of this writing, the number of plugins is nearing 32,000 (<http://wordpress.org/plugins/>). Since the last publication of this book, the community has contributed over 1,000 unique themes to the official WordPress theme repository, which now has more than 2,500 listed. This does not include all the commercial theme vendors and independent developers creating their own custom themes.

The combinations of plugins and themes require scientific notation to represent in complexity, but at the same time, they are all equally simple to locate, integrate, and use. That's the result of a solid architecture and an equally solid community using it. In short, the ecosystem surrounding WordPress is alive and thriving—even booming.

Today, WordPress powers many large media companies' websites or portions thereof, including CNN's blogs, the *Wall Street Journal's All Things D*, Reuters, and Forbes. Fortune 500 companies such as GM, UPS, and Sony use WordPress. WordPress is a viable choice for a range of users, from international conglomerates to major recording artists to huge media publishing companies. Some need reassurance before choosing WordPress and focus on which big boys are using it; you can find a list online at the WordPress Notable Users showcase (<http://en.wordpress.com/notable-users/>).

But the simplicity, ease of use, and ultimately the power of the plugins and themes also makes WordPress suitable for your mom's family information website, your local elementary school teacher's classroom newsletter, and the hobbyist. These are truly some of the WordPress success stories of today and these widely accessible, more narrowly popular websites are what makes WordPress popular. WordPress is adaptable and will be as simple or complex as you need it to be. Empowering "lower tech" users to be web publishers and then spreading the word (pun intended) to their families and friends about how easy WordPress is to use have fueled this explosive growth and adoption.

Where do you get started? Wordpress.org is the home for the current released and in-development versions of the code. Click through to [wordpress.org](http://wordpress.org) for a starting point in finding plugins, themes, and wish lists of ideas and features to be implemented.

Wordpress.com has both free and paid hosting services. Over at [www.wordpress.org/hosting](http://www.wordpress.org/hosting) you will find a list of hosting providers that support WordPress and often include some additional first-time installation and configuration support in their packaging of the code for delivery as part of their hosting services. You will also find concentrated WordPress hosting providers that strictly host WordPress sites and offer additional specialization features and options.

## Intersecting the Community

WordPress thrives and grows based on community contributions in addition to sheer usage. Like high school gym class, participation is the name of the game, and several semi-formal avenues along which to channel your efforts and energies are available.

WordCamp events are community-hosted and locally operated, and now happen in dozens of cities around the world. Official WordCamps are listed on [wordcamp.org](http://wordcamp.org), but you will do just as well to search for a WordCamp event in a major city close to you. WordCamps occur nearly every weekend with bloggers, photographers, writers, editors, developers, and designers of all experience and skill levels counted among their attendees. WordCamps are a low-cost introduction to the local community and often a good opportunity to meet WordPress celebrities. Visit [www.wordcamp.org](http://www.wordcamp.org) to find the next WordCamp.

Less structured but more frequently convened than WordCamps are WordPress Meetups, comprising local users and developers in nearly 400 (up from the 200 mentioned in the second edition of this book, and 40 in the first) cities. You'll need a [meetup.com](http://meetup.com) account, but once you're registered, you can check on locations and timetables at [www.wordpress.meetup.com](http://www.wordpress.meetup.com) to see when and where people are talking about content management.

A rich, multi-language documentation repository is hosted at [www.codex.wordpress.org](http://www.codex.wordpress.org). The WordPress Codex, with all due respect to the term reserved for ancient handwritten manuscripts, represents the community-contributed tips and tricks for every facet of WordPress, from installation to debugging. If you feel the urge to contribute to the WordPress documentation, register and then write to your heart's content in the WordPress Codex. We hope that you will find this book a cross between a companion and a travel guide to the Codex.

Finally, mailing lists (and their archives) exist for various WordPress contributors and communities. A current roster is available online at [www.codex.wordpress.org/Mailing\\_Lists](http://www.codex.wordpress.org/Mailing_Lists); of particular interest may be the `wp-docs` list for Codex contributors and the `wp-hackers` list for those who work on the WordPress core and steer its future directions.

## WordPress and the GPL

WordPress is licensed under the Gnu Public License (GPL) version 2, contained in the `license.txt` file that you'll find in the top-level code distribution. Most people do not read the license and simply understand that WordPress is an open source project; however, pockets of corporate legal departments still worry about the viral component of a GPL license and its implications for additional code or content that gets added to, used with, or layered on top of the original distribution. Much of this confusion stems from liberal use of the words “free” and “copyright” in contexts where they are inappropriately applied.

The authors of this book are not lawyers—nor do they play them on the Internet or on television—and if you really want to understand the nuances of copyright law and what constitutes a “conveyance” of code, pick up some of Lawrence Lessig’s or Cory Doctorow’s work in those areas. This section is included to minimize the concerns of IT departments who may be dissuaded from using WordPress as an enterprise content management system by overly zealous legal teams. Do not let this happen to you; again, if WordPress is acceptable to CNN and the *Wall Street Journal*, two companies that survive on the copyrights granted to their content, it probably fits within the legal strictures of most corporate users as well.

The core tenet of the GPL ensures that you can always get the source code for any distribution of GPL-licensed software. If a company modifies a GPL-licensed software package and then redistributes that newer version, it has to make the source code available as well. This is the “viral” nature of GPL at work; its goal is to make sure that access to the software and its derivatives is never reduced in scope. If you plan on modifying the WordPress core and then distributing that code, you will need to make sure your changes are covered by the GPL and that the code is available in source code form. Given that WordPress is written in PHP, an interpreted language, distributing the software and distributing the source code are effectively the same action.

Following are some common misperceptions and associated explanations about using WordPress in commercial situations.

- **“Free software” means you cannot commercialize its use.** You can charge people to use your installation of WordPress, or make money from advertisements running in your website, or use a WordPress content management platform as the foundation of an online store. That is how `wordpress.com` works; it also enables Google to charge advertisers for using their Linux-based services. You can find professional quality WordPress themes with non-trivial price tags, or you can pay a hosting provider hundreds or even thousands of dollars a year to run your MySQL, PHP, Apache, and WordPress software stack; both involve commercialization of WordPress.
- **If you customize the code to handle your own {content types, security policies, or obscure navigational requirements} you will have to publish those changes.** You are only required to make the source code available for software that you distribute. If you choose to make those changes inside your company, you do not have to redistribute them. On the other hand, if you’ve made some improvements to the WordPress core, the entire community would benefit from them. Getting more staid employers to understand the value of community contribution and relax copyright and employee contribution rules is sometimes a bit challenging, but the fact that you had a solid starting point is proof that other employers made precisely that set of choices on behalf of the greater WordPress community.

- **The GPL will “infect” content that you put into WordPress.** Content—including graphic elements of themes, posts, and pages managed by WordPress—is separated out from the WordPress core. It is managed by the software, but not a derivative of or part of the software. Themes, however, are a derivative of the WordPress code and therefore also fall under the GPL, requiring you to make the source code for the theme available. Note that you can still charge for the theme if you want to make it commercially available. Again, the key point here is that you make the source code available to anyone who uses the software. If you are going to charge for the use of a theme, you need to make the source code available under the GPL as well, but as pointed out previously, users installing the theme effectively get the source code.

More important than a WordPress history lesson and licensing examination are the issues of what you can do with WordPress and why you would want to enjoy its robustness. The next section looks at WordPress as a full-fledged content management system, rather than simply a blog editing tool.

## CONTENT AND CONVERSATION

Multiple linear feet of shelves in bookstores are filled with volumes that will improve your writing voice, literary style, blogging techniques, and other aspects of your content creation abilities. One of the goals of this book is to define the visual, stylistic, and context management mechanisms you can build with WordPress to shape vibrant user communities around your content. That context stimulates conversation with your readers. Publishing is not just about the words in each post, or even if you are an interesting writer. How will people find you? How will you stand out in the crowd? How do you put your own imprint on your site, and personalize it for whatever purpose: personal, enterprise, community, or commercial?

## WordPress as a Content Management System

Blogging systems have their roots in simple content management operations: Create a post, persist it in stable storage such as a filesystem or database, and display the formatted output based on some set of temporal or keyword criteria. As the richness and types of content presented in blog pages expanded, and the requirements for sorting, searching, selecting, and presenting content grew to include metadata and content taxonomies, the line between vanilla, single-user-targeted blogging software and enterprise-grade content management systems blurred.

Content management systems (CMS) handle the creation, storage, retrieval, description or annotation, and publication or display of a variety of content types. CMS also covers workflow tasks, typically from an editorial or publishing perspective, and includes actions such as approval and marking content for additional editing or review. The WordPress Dashboard provides those elements of workflow management and editorial control. WordPress is not the only open source content management system in widespread use today; the Drupal and Joomla projects are equally popular choices. Drupal and Joomla start from the perspective of managing content repositories; they handle a variety of content types, multiple authors in multiple roles, and delivering the content to a consumer that requests it. WordPress is at its heart a publishing system, and the end focus is on displaying content to a reader. Although areas of functional overlap exist, you can integrate WordPress with other content management systems, a process covered in detail in Chapter 15.



WordPress has established itself as a *bona fide* content management system through its design for extensibility and the separation of content persistence from content display. Taking some liberties with the Model-View-Controller design pattern, WordPress separates the MySQL persistence layer as a data model, the theme-driven user interface and display functions, and the plugin architecture that interposes functionality into the data to presentation flow. Most important, WordPress stores content in raw form, as input by the user or an application posting through the WordPress APIs. Content is not formatted, run through templates, or laid out until the page is rendered, yielding immense power to the functions that generate the actual HTML. At the same time, the data model used by WordPress uses a rich set of tables to manage categories (taxonomies), content tags (folksonomies), author information, comments, and other pieces of cross-reference value. The WordPress database schema that makes this possible is explored in Chapter 6.

Although that design gives WordPress incredible power and flexibility as a content management system, it also requires knowledge of how those data persistence and control flows are related. (It was a search for such a dissection of WordPress in functional terms that got us together to write this book.)

## Creating Conversation

*Conversation is king; content is just something to talk about.*

—CORY DOCTOROW

A robust CMS is measured by the utility of its content. Even the richest content types and most well-managed processes are of low return if nobody actually consumes the outputs. It is not sufficient to install blogging software, write a few posts, and hope the world shows up on your virtual doorstep; you need to create what Tim O'Reilly calls an “architecture of participation.” Social networking, advertising, feeds, and taking steps to ensure your site shows up in search engine results will drive readers to your site; the design, branding, and graphic elements coupled with the quality of your content will encourage them to take the steps toward active participation.

Look at the problem from the perspective of a reader: In a world of tens of millions of websites (many of which have a “first post” and not much else), how will you be found, heard, and echoed? Your Twitter followers should want to read your site, and your WordPress site can update your Twitter feed. Conversely, your Twitter updates may appear in your WordPress sidebar, marrying the ultra-short content timeline to the more thoughtful one. If you are active on Facebook, you can import entries into a public figure page and Facebook readership will drive traffic back to your website. If you cover specific, detailed, or arcane areas in your writing, Google searches for those terms should direct readers to you, where they will join the conversation. Chapter 12 looks at how your WordPress content can be more broadly distributed.

## GETTING STARTED

Before any serious work on presentation, style, or content begins, you need a home for your website (despite the previous discussion about WordPress and content management systems, we will refer

to your website and the actual WordPress installation that implements it interchangeably, mostly for convenience and brevity). Factors affecting your choice include:

- **Cost**—Free hosting services limit your options as a developer and frequently preclude you from generating money from advertising services. More expensive offerings may include better support, higher storage or bandwidth limits, or multiple database instances for additional applications.
- **Control**—What tools are provided for you to manage your MySQL database, files comprising the WordPress installation, and other content types? If you want to be able to muck around at the SQL level, or manage MySQL through a command-line interface, you should ensure your hosting provider supports those interfaces.
- **Complexity**—You can install the Apache or nginx web server with a PHP interpreter, MySQL, and the WordPress distribution yourself, but most hosting providers have wrapped up the installation process so that some of the rough edges are hidden from view. If you expect to need technical support on the underlying operating system platform, find a provider (including your own IT department) that provides that support in a reasonable time frame.

This section takes a quick look at some hosting options, walks you through the basics of a do-it-yourself installation, and concludes with an overview of the ways in which WordPress and MySQL choose to ignore each other when installation goes into the weeds.

## Hosting Options

Three broad categories of WordPress hosting exist, each with trade-offs between administrative complexity and depth of control. The easiest and most popular is to use `wordpress.com`, a free hosting service run by Automattic using the multi-site version of WordPress (originally WordPress MU). You can install themes and plugins through the Dashboard but you can only enable or disable the choices that come preinstalled. Further, you will not have access to the underlying MySQL databases and core code, or be able to integrate WordPress with other systems. You can redirect one of your own URLs to `wordpress.com`, but if you want full control over everything from the code to the URLs used, you are probably looking at a paid option. The free route may be a reasonable first step for you, but for this book it is assumed that you are going to want to perform surgery on your installation.

You will find a starter list of for-fee hosting providers on `www.wordpress.org`, including the paid option on `wordpress.com`. Most have the latest, or close to latest, releases of the WordPress core available as a package to be installed in conjunction with MySQL and a web server. The third hosting option is to install everything on servers that you own and operate. If your servers live in a hosting facility but you enjoy root administrative access that is equivalent to a do-it-yourself installation. These are all options for putting your WordPress installation on the public Internet. If you are just looking to explore, Chapter 3 covers running WordPress locally for development.

WordPress requires a web server with PHP support, a URL rewriting facility, and an instance of MySQL. Apache is the most popular option for front-ending WordPress because it provides PHP interpretation through `mod_php` and URL rewriting in `mod_rewrite`. There is growing interest in `lighttpd` (Lighty) and `nginx` as replacements for Apache. Finally, you can use Microsoft's IIS 7.0 as a



web server with its `URL _rewrite` module. The emphasis on URL rewriting stems from WordPress's support for "pretty" permalinks to content entries, allowing you to create a URL tree organized by date, category, tag, or other metadata. Those mnemonic, or human-readable, URLs are converted into MySQL database queries to extract the right WordPress content based on titles or other keywords as part of the WordPress main loop, which is covered in detail in Chapter 5. Your web server decides whether the URL should be parsed by WordPress or if it refers to a specific HTML file based on what is in the `.htaccess` file, and the URL rewriting rules ensure that its contents are interpreted properly. Technically, URL rewriting is not required to install WordPress, but it is good to have because it gives you tremendous flexibility in the presentation and naming conventions used for your content's URLs. Permalink design and practices are covered in more detail in Chapter 2, but keep the requirement in mind as you select your WordPress substrate.

Up to this point, MySQL has been mentioned only in passing, but a brief review of MySQL requirements rounds out the hosting prerequisite list. It is worth establishing some terminology and distinguishing between the MySQL software, database instances, and WordPress instances using MySQL. When you install and configure MySQL, you have a full-fledged relational database system up and running. It does not have to be configured on the same machine as your web server, and some hosting providers will create horizontally scalable MySQL "farms" in parallel to their web server front ends. An *instance* of MySQL running on a server can support multiple *databases*, each with a unique name. When you install WordPress, you will need to know the *name* of the MySQL database reserved for your content, although this information may be auto-generated and configured for you if you are using a provider that supports WordPress and MySQL as an integrated package. WordPress creates a number of relational data *tables* in that named database for each website that you create.

Confusion can result from nomenclature and complexity. You (or your hosting provider) may run multiple MySQL instances on multiple servers, and you will need to know where your database is hosted. Because each instance of MySQL can run multiple databases, and each database contains groups of tables, it is possible, even common, to run multiple MySQL-based applications on the same hosting platform, using one MySQL instance or even one MySQL database.

If you want to have multiple WordPress sites on the same server, you can share a single MySQL database instance for all of them provided you configure WordPress to distinguish the MySQL database table names within the MySQL database. It is a simple configuration option that is covered in the next section, and it highlights the distinction between multiple sets of tables in a database and multiple databases for distinct applications.

Once you have secured the necessary foundation, it is time to get the code up and running. Even if you are using a hosting provider that installs MySQL and WordPress for you, it is worth knowing how the server-side components interact in case you need to track down a problem when you're deep in plugin development.

## Do It Yourself Installation

The famous, fabled, fabulous five-minute WordPress installation is a reality when everything is configured and coordinated properly. This section walks you through the steps that are often hidden from view when you use a provider with packaged installs, and highlights some of the common misfires between WordPress and MySQL instances.

The installation process is quite simple (assuming that your web server and MySQL server are already running): Download the WordPress package and install it in your web server's directory tree, and then navigate to your top-level URL and complete the configuration. One (compound) sentence describes it completely.

It is possible and even advisable to install a fully functioning WordPress instance on your laptop or development machine, particularly if you are going to be working on the core, developing plugins, or otherwise making changes that would create embarrassing failures during testing on a public website. Mac OS X comes with an Apache web server (with PHP and URL rewriting); download MySQL from [www.mysql.com](http://www.mysql.com), or use a prepackaged configuration such as MAMP ([www.mamp.info](http://www.mamp.info), which includes the phpMyAdmin tool), and you will have a self-contained development and deployment lab. For other platforms, XAMPP ([www.apachefriends.org](http://www.apachefriends.org)) has a neatly integrated platform stack that runs on Windows, Mac OS, and Linux foundations. Furthermore, the use of virtual machines for your development environment has grown immensely, and now there are packaged VM solutions to get you started. Having everything under one hood is a powerful option for examining failure modes, as you will see in the next two sections. More information on working with WordPress locally is covered in Chapter 3.

## Installing WordPress Files

If you download the WordPress code from [wordpress.org](http://wordpress.org), you will get a zip (or tarball) archive that expands into a directory called `wordpress`. The first part of a WordPress installation is to get the code into your web server's directory structure; ensuring you have it in the right place is a critical step. Gloss over this part and you will find your website ends up with a URL like `http://example.com/wordpress` and you will either have to start over or e-mail ugly URLs to your friends and family. If that is what you want—to distinguish your WordPress site from other content on your website or to isolate multiple sections—choosing the filesystem layout is equally important.

Pick the top-level directory where you want to install WordPress. Most commonly, this is the root directory for your web server, and if you are using a hosting provider it is probably the subdirectory called `public_html` in the file tree. If you are using a packaged install where there is a menu asking you for the target location, make sure you pick this top-level directory (and yes, you know that it already exists, that's the point!); if you are copying files from your local machine to the web server target using an FTP client, make sure you pick the right destination. The somewhat obvious move to copy the zip file to the server and then unpack it will put everything into a `wordpress` subdirectory, and if you want your WordPress site's URL to be `http://example.com` rather than `http://example.com/wordpress`, move the files up one directory level before proceeding. There is a configuration option to have your WordPress installation in a subdirectory to your top-level URL, so it is not fatal if you drop WordPress into a less-than-desirable filesystem geography. That is covered at the end of this section.

Once the WordPress files are installed, your filesystem browser should show you something like Figure 1-1, with an `index.php` and template `wp-config-sample.php` file. That's the entirety of the WordPress system, which runs effectively within the web server's PHP interpreter.

At this point, if you are doing a manual installation, you will want to create your own `wp-config.php` file by editing the provided sample file, `wp-config-sample.php`, and saving it in your top-level WordPress directory. As an alternative, you can navigate to your website's URL, and

the WordPress code will notice there is no configuration file. After you select your installation language, WordPress presents you with dialog boxes like those in Figures 1-2 and 1-3 where you can fill in the details. You will need the MySQL database name, database username, and some idea of the WordPress database table prefix (other than the default `wp_`). These lower-level details are the guts of the next section on database configuration. If you are using a hosting provider with packaged installations, you probably will not see this step because the WordPress files will be extracted and the MySQL database information will be automatically inserted into a configuration file, no end user-serviceable parts inside.

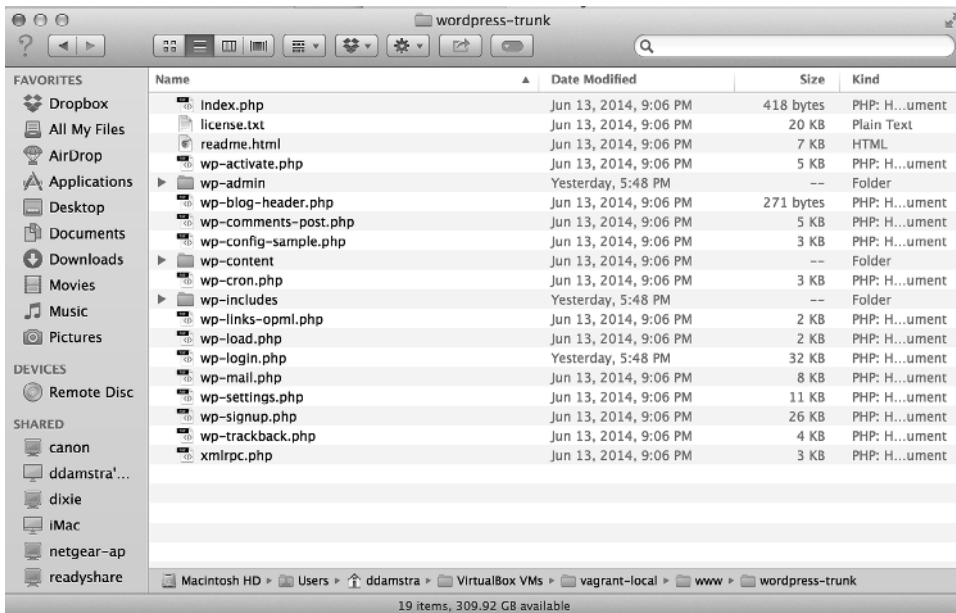


FIGURE 1-1: A clean but unconfigured WordPress installation

What do you do if you already have HTML or other content at your target URL and you want to add WordPress to an existing site? Disposition of existing files depends on your desired first user experience upon navigating to your URL. To use WordPress as a content management system as described here, your best choice is to save existing content and convert it into new posts or pages, effectively making your previous site color commentary and context for your WordPress-driven site. Alternatively, you can install WordPress in a subdirectory, keep your existing `index.html` file, and direct readers to your new content through a button or link on your extant home page. Do not leave this to chance; if you have an `index.html` file and then install WordPress, you will have an `index.php` and an `index.html` file side by side and users will see one or the other depending upon the Directory Index configuration of your site's web server. Actions on existing content should be informed by how much traffic that content is driving to your site: if your pages are responsible for search engine traffic, you probably do not want to disrupt the existing URLs that have been cached and should install WordPress in a subdirectory. If you feel strongly about making WordPress the wrapper around the user experience, move the content and include URL rewriting or redirection for pages that move into the WordPress world.

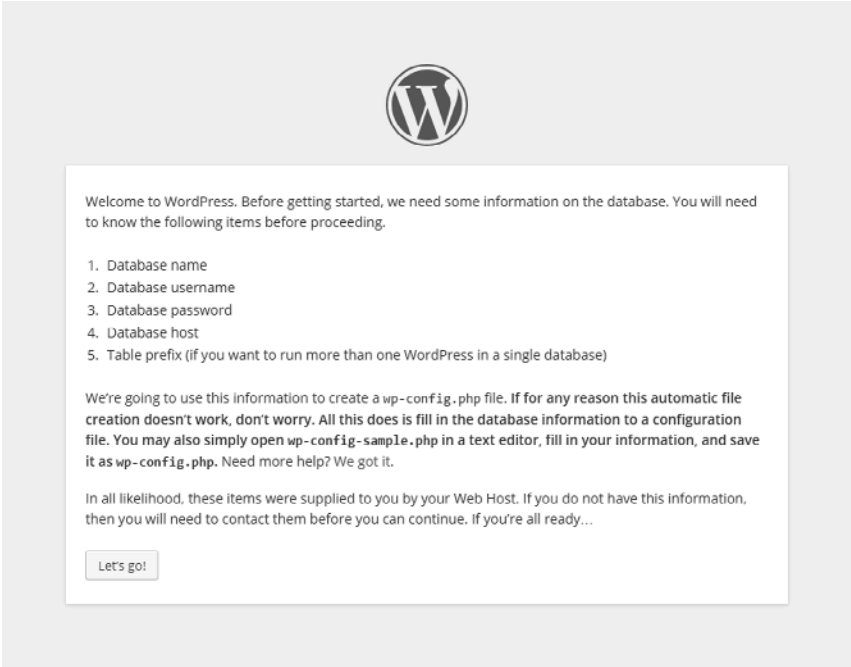


FIGURE 1-2: WordPress will create a new wp-config file if one does not exist.

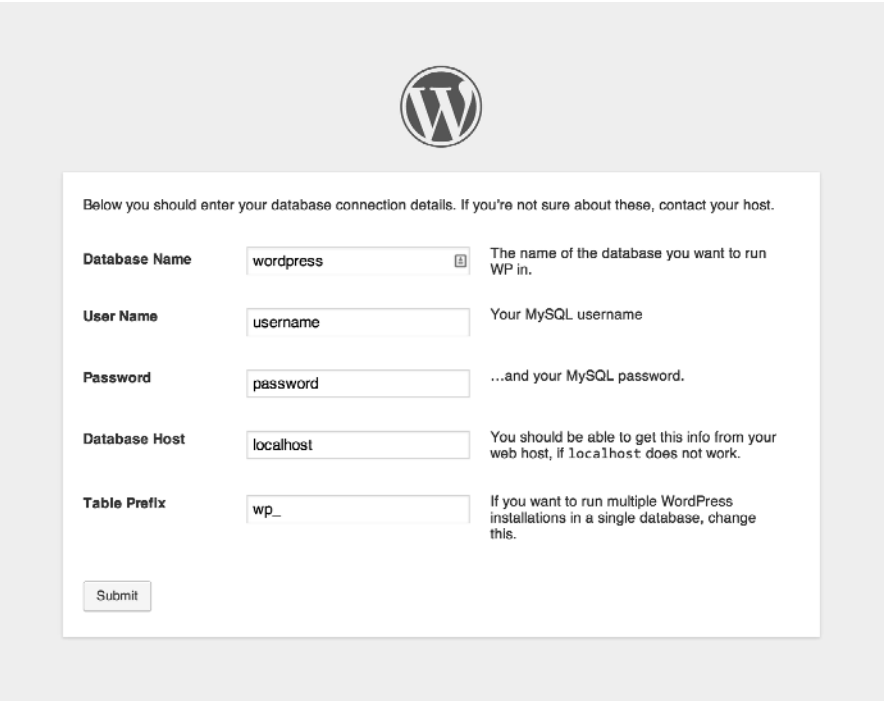



FIGURE 1-3: Database configuration dialog box



## Welcome

Welcome to the famous five minute WordPress installation process! You may want to browse the ReadMe documentation at your leisure. Otherwise, just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

### Information needed

Please provide the following information. Don't worry, you can always change these settings later.

**Site Title**

**Username**   
Usenames can have only alphanumeric characters, spaces, underscores, hyphens, periods and the @ symbol.

**Password, twice**  
A password will be automatically generated for you if you leave this blank.

Hint: The password should be at least seven characters long. To make it stronger, use upper and lower case letters, numbers, and symbols like ! \* ? \$ % ^ & .

**Your E-mail**   
Double-check your email address before continuing.

**Privacy** ☒ Allow search engines to index this site.

**FIGURE 1-4:** Complete website details and set up admin user.

If you used a hosting provider's packaged installation, or if you manually created a `wp-config.php` file and then navigated to your top-level URL, WordPress should have completed creating the database tables, created an administrative user for your WordPress, and set an initial password, as shown in Figure 1-4. Make sure you change the username to something different than admin.

Upon a successful installation, you should see a box like Figure 1-5 that indicates your five minutes of famed installation is done.

The next section covers the MySQL-WordPress configuration dance in more detail and is suitable reading even if thinking about SQL gives you hives. If you are up and running, you can skip the next section and go right to the section "Finishing Up."

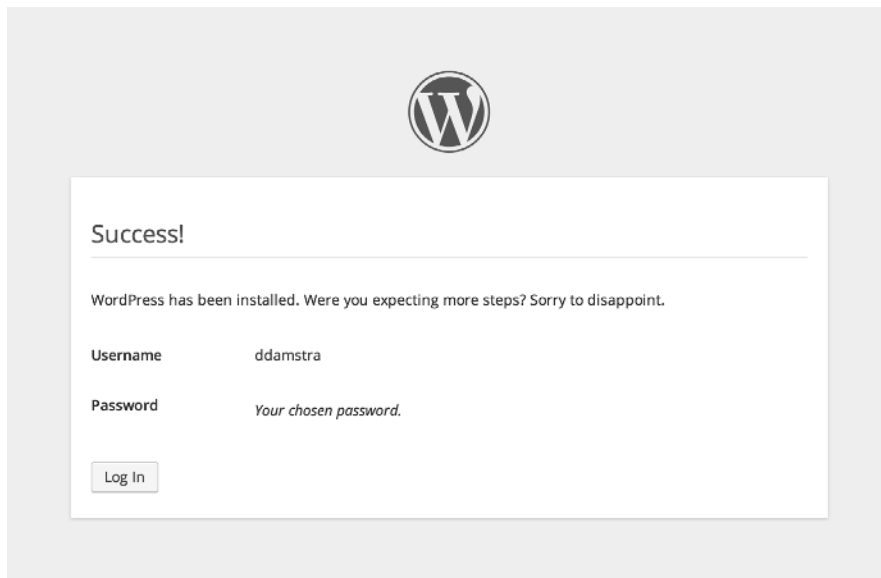


FIGURE 1-5: Administrative information at the conclusion of a clean install

## Database Configuration

If your hosting provider spun up a MySQL database and created a user for you, check your resultant `wp-config.php` file to gather this information. It is necessary for the MySQL probing covered in this section, and it is good to have in case you run into MySQL problems later on. There is a username and password combination included in that file, so treat it the way you would treat other login information. On the other hand, if you are going deep on the do-it-yourself route, this section gives you a sense of what is likely to create confusion or consternation as you pull the pieces together.

In theory, MySQL setup for WordPress is trivial: Make sure MySQL is up and running, create a WordPress user in MySQL, and then have that user create a database to hold the WordPress tables. You can use the MySQL command line or tools such as phpMyAdmin or Chive for these tasks, but bear in mind that MySQL has its own set of users and permissions granted to those users, distinct from those used by your (or your hosting provider's) operating system. Once MySQL is installed, it will create a default table of users and grants, adding a `root` user on Unix systems that is a MySQL superuser, unrelated to the Unix `root` user. However, if you are attempting to connect to your MySQL instance as the MySQL `root` user, those connections can only be made from `localhost`—the same machine on which MySQL is running. If you want to learn more about MySQL permissions, the table governing grants of those permissions to users, and how MySQL users are managed, refer to the “MySQL Reference Manual” (<http://dev.mysql.com/doc/>) and the sections on securing the initial MySQL accounts.

No set naming conventions exist for WordPress users or databases; hosting providers will typically append the name of the package or your account information to distinguish users that benefit from MySQL database co-tenancy. Again, it is possible to have multiple databases, owned by the same user or different MySQL users, running in a single MySQL database server instance. In the

example shown in Figure 1-3, `wp_` is used as a prefix for both usernames and database names, at least providing a hint to the database administrator that these belong to a WordPress installation. Security best practices recommend not using `wp_` as your table prefix; this is covered more in Chapter 13.

What can go wrong between WordPress and MySQL? The following are the three primary root causes of installation failure. Note that all of these conditions need to be fulfilled at installation time; there has to be some basic database structure to contain the admin user before you can log in as that admin.

- **Web server cannot find MySQL.** Either you have the hostname for the MySQL server noted incorrectly in the `wp-config.php` file, or the web server is looking for a local MySQL instance and cannot open the socket connection to it. Here is a simple example: when you run WordPress locally on Mac OS, MySQL creates the socket `/tmp/mysql.sock` for local connections, but the WordPress PHP code is going to look for `/var/mysql/mysql.sock` through the PHP engine's MySQL module. Simply symbolically link one to the other:

```
# ln -s /tmp/mysql.sock /var/mysql/mysql.sock
```

The actual filesystem path to the local MySQL socket is a function of the database configuration; when it starts up, it creates the local socket. Where the PHP engine, and therefore any PHP-based applications, looks for this socket is PHP configuration dependent. If you want to figure out exactly where the mismatch is, a bit of heavy-handed `printf()` style debugging helps.

Edit `wp-includes/wp-db.php`, the set of functions that establish WordPress's database connection. If you are seeing the "Error establishing a database connection" message during installation, insert an `echo(mysql_error());` statement where the error is detected to see the details displayed along with the generic message, as shown in Figure 1-6:

```
if (!$this->dbh) {
    echo(mysql_error());
    $this->bail(sprintf(*WP_I18N_DB_CONN_ERROR*/"
    <h1>Error establishing a database connection</h1>
```

The `mysql_error()` function is a PHP library function that spits out the error generated by the last MySQL function called.

- **WordPress finds MySQL but cannot log in.** Most of the time, the MySQL username or password is wrong, particularly when you have to copy some arbitrary username generated by a hosting provider. Double-check your username data, and verify that it is reflected properly in your `wp-config.php` file. You may also run into a password authentication issue when using MySQL 4.1 or MySQL 5.0 with some web servers' PHP implementations; they only support the older MySQL 4.0 password hashing scheme. If this is the case, use MySQL's `OLD_PASSWORD()` function to hash your WordPress user's password in the backward-compatible format; use the magic SQL incantation (at the MySQL command-line prompt or within the SQL window of MAMP) to address the following:

```
SET PASSWORD FOR user@host = OLD_PASSWORD('password');
```



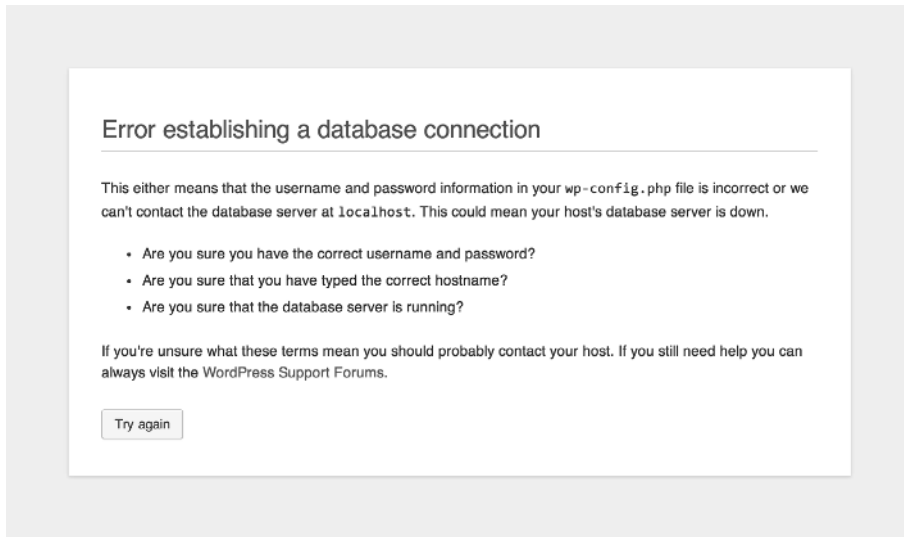


FIGURE 1-6: `mysql_error()` reporting a socket problem

In this instance, `user@host` is your WordPress database username and database hostname, and `password` is the (clear text) password you provided in the configuration file.

- **WordPress connects to MySQL but cannot select the database.** Just because the web server can log in to the database server with your WordPress database user information does not mean that there is necessarily a database available to that user. This is another scenario best diagnosed with `mysql_error()`, by inserting it in `wp-db.php` where the selection error is identified:

```
function select($db) {
    if (!@mysql_select_db($db, $this->dbh)) {
        $this->ready = false;
        echo(mysql_error());
        $this->bail(sprintf(/*WP_I18N_DB_SELECT_DB*/'
... <h1>Can&#8217;t select database</h1>
..
```

If, after inserting the `mysql_error()` statement as described earlier, your attempts to complete installation result in an error box like that shown in Figure 1-7, your MySQL database was not created under the appropriate database user, or the database user does not have privileges to use it. Double-check what MySQL believes using the following command line:

```
vagrant@vrv:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 98
Server version: 5.5.37-0ubuntu0.14.04.1 (Ubuntu)
```

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| wordpress_default |
| wordpress_develop |
| wordpress_trunk |
| wordpress_unit_tests |
+-----+
7 rows in set (0.00 sec)

mysql>
```

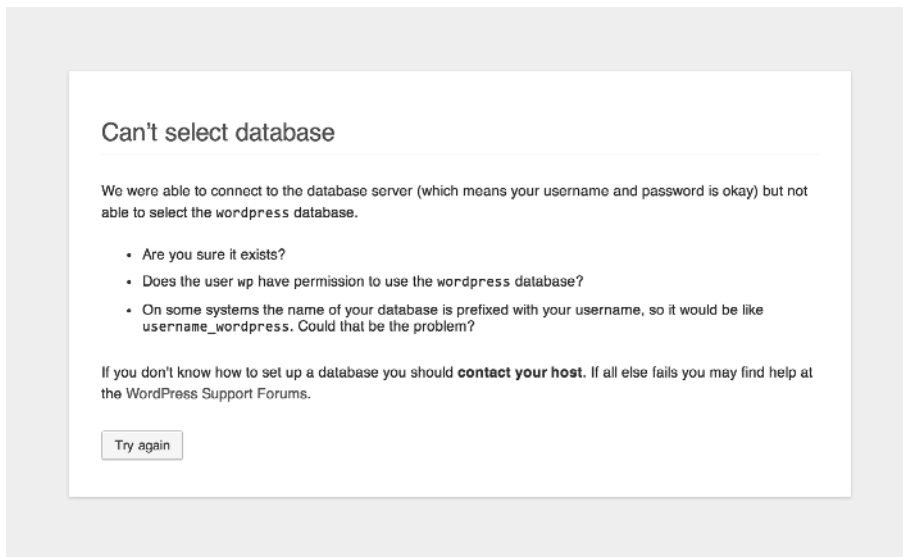


FIGURE 1-7: MySQL database selection error

Once you logged in as your designated MySQL database user, you did not see the MySQL database—in this case, it was probably created by the MySQL user root, and permissions to access or modify it were not granted to the WordPress installation's MySQL user. If you have MySQL root access, or sufficient MySQL user privileges to create new databases within the MySQL instance, it is easy enough to create a database once logged in on the command line:

```
mysql> create database wordpress_trunk;
Query OK, 1 row affected (0.00 sec)
```

Again, it is important to distinguish operating system users from MySQL users from WordPress users. MySQL users are defined in the database and granted privileges to create databases, muck with tables, and otherwise generate useful data. WordPress users exist within the WordPress database tables created during install; they only have privileges, context, and meaning once you are logged in to WordPress.

Once you have a clean WordPress installation, you should see a collection of tables named according to the table prefix you set in `wp-config.php`; again, this is easy enough to verify using the MySQL command line:

```
mysql> use wordpress_trunk; show tables;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
+-----+
| Tables_in_wordpress_trunk |
+-----+
| wp_commentmeta            |
| wp_comments               |
| wp_links                  |
| wp_options                |
| wp_postmeta               |
| wp_posts                  |
| wp_term_relationships     |
| wp_term_taxonomy          |
| wp_terms                  |
| wp_usermeta               |
| wp_users                  |
+-----+
11 rows in set (0.00 sec)

mysql>
```

In this example, you set the database table prefix to `wp_`; if you later add another WordPress installation using the same database user and instance, you can simply set a different prefix and have the two sites co-mingled in the same database table. You dig into the schema and uses of the basic WordPress database tables in Chapter 6. For now, once you are happily connected to MySQL, you are ready for some final clean-up and first-time administration.

## FINISHING UP

At this point, your MySQL database is up and running. There is a home for your content, and your web server is happily executing the WordPress core code. There are just a couple more things to discuss.

## First-Time Administration

Once you have completed the installation, proceed to log in with the credentials you set up in Figure 1-4 and you'll see the basic WordPress Dashboard captured in Figure 1-8.

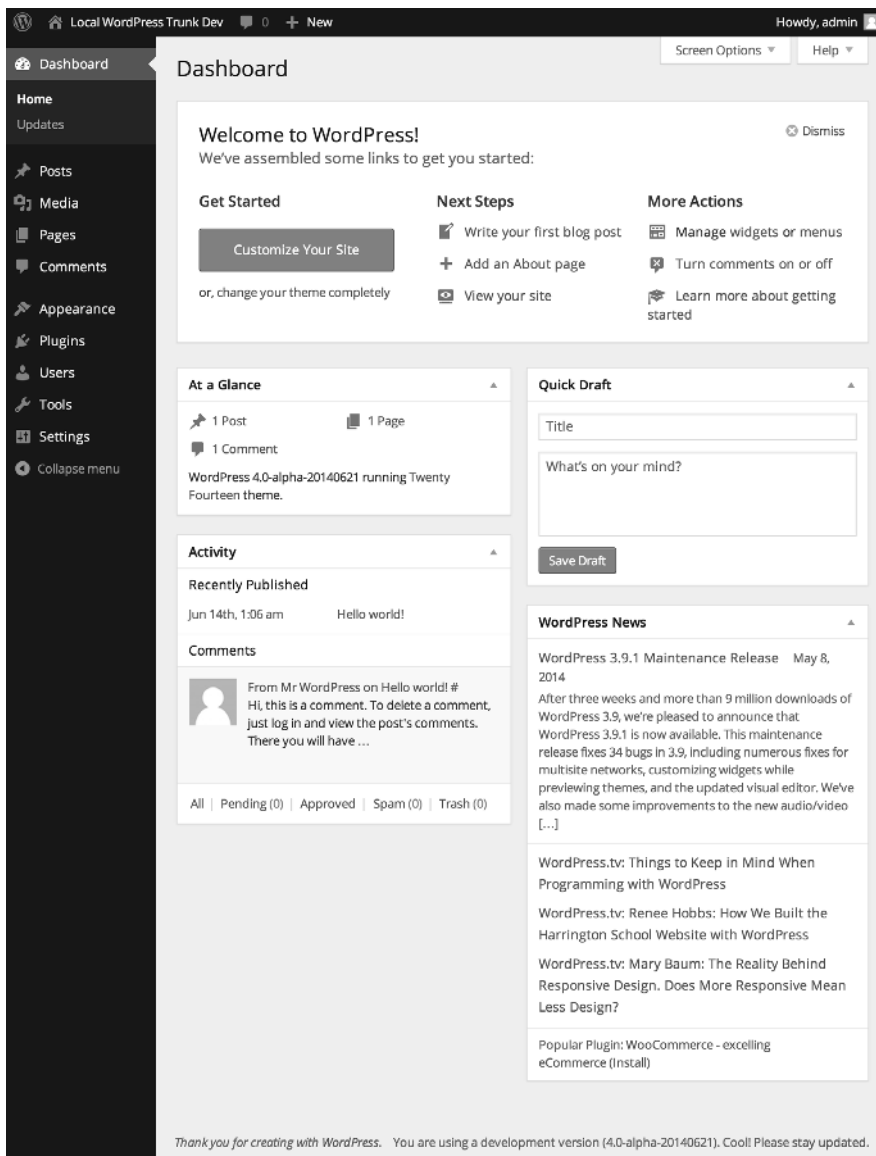


FIGURE 1-8: Dashboard view upon a first-time login

If you are not redirected to the Dashboard through the Log In button, or if you happen to visit your website's top-level URL first, either click the Log In link on your website or explicitly go to the `wp-admin` subdirectory (`example.com/wp-admin`) to be presented with a login dialog box. Logging in to your website takes you to the WordPress Dashboard, which is both amazingly simple in its power and rich in its complexity and exposed features.

What you do next with the Dashboard depends on how happy you are with the basic installation. If, as in the preceding example, you ended up with an older version of WordPress, click the Update button to do an in-place upgrade to the latest distribution. In addition to having a strong self-installation feature, WordPress includes self-update functions (in `wp-admin/includes/update.php` if you are looking for them).

You may decide to change some basic configuration options, such as the database name or the MySQL database user, although you will only change the default of `root@localhost` if you have full control over the web and database servers. The configuration file also has entries for “security keys” that are used to provide stronger security for browser cookies. Security keys are discussed in more detail in Chapter 13. Editing your `wp-config.php` file affects the changes right away. Changing the database table prefix, for example, causes WordPress to instantiate a new set of tables and create a clean-slate installation. Make those edits and then go back to your top-level URL and you will find yourself with new admin user information and logged in to a starter Dashboard, as in Figure 1-8. Old tables are not removed from MySQL, so you’ll have to do manual cleanup.

At this point, if you want to set your URL to be different from the location in which you installed WordPress, you can choose Settings and General from the Dashboard and change the URLs for both your top-level address as well as the WordPress installation directory. If you dissociate your site’s URL and the WordPress directory, make sure you move the `index.php` file to the desired top-level URL, and then edit the last line to include the proper subdirectory path to WordPress.

Before creating your first post, it is also a good idea to establish a permalink structure so that everything you write follows the naming conventions you have chosen to make it relatively easy for readers to find, share, and link to your content. As expected, it is another option in the Settings portion of the Dashboard; options for permalink naming and their impact on performance and database schema are covered in more detail in the next chapter.

Whether it has really been five minutes, or a few hours of tracking down mismatches in hostnames, usernames, and database configurations, you are now ready to publish the first post of your own writing.

## First Post

A successful WordPress installation already has a first post and comment published, thus ensuring that all of the moving pieces are moving in unison, and giving your website some initial content. When you are ready to add your own first words, either use the right-hand QuickDraft panel in the Dashboard to post an entry (you may need to dismiss the new website help first), or go to Posts and click Add New to be taken to the built-in WordPress editor. Figure 1-9 shows an entry in progress in the QuickDraft panel, followed by the updated Dashboard after it has been successfully posted.

If your tastes run more old-school, you can always crank out content in your favorite text editor and then copy it into the editing pane. Be careful with WYSIWIG word processors such as Microsoft Word or OpenOffice if you want to copy into the WordPress HTML composition window because the HTML will be riddled with additional tag and style information. Finally, a variety of standalone editors publish to WordPress using the Atom Publishing Protocol or XML-RPC. Options for

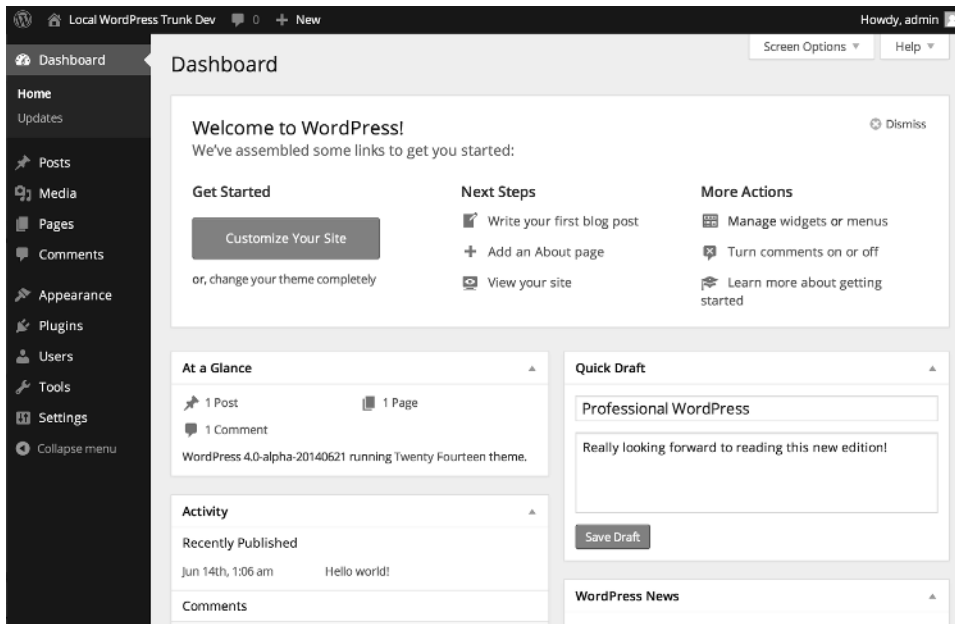


FIGURE 1-9: Publishing from the QuickDraft panel

enabling posts to be published remotely are, as you would expect, in the Dashboard’s Settings section under Writing options.

Click Publish for your own “Hello World” moment. Multiple subsystems created that editing pane, saved the content in a database, generated and saved the referential metadata, and then emitted nice-looking HTML. Most of the user-visible pieces are governed through the Dashboard and certain functions will be covered in various chapters.

## SUMMARY

This chapter covered how WordPress got to where it is today with a brief history lesson and also touched on its current popularity. Part of WordPress’s rise in the web realm is attributed to the simplicity of the installation process. The next chapter dives into the core of WordPress so that you can take advantage of its extensibility, friendly design, and function.

