Software Developer's JOURNAL

new ideas & solutions for professional programmers

Vol.2 No.13 Monthly Issue 13/2013 (19) ISSN 1734-3933

Building Your Own WordPress Sites from Scratch

Most Exciting Tips and Instructions Written by Best Experts

Dear Readers,

We would like to present you with our first WordPress issue! As promised, you will find tutorials varying on different levels of complexity and a wide range of topics.

With a strong following worldwide, WordPress has become one of the most popular content managements systems (CMS) used for web development. Its popularity continues to grow as the flexibility and dynamics of the platform are being pushed further everyday. Without further adieu, here it is a brand new SDJ issue!

We've started a partnership with some of the best developers in the industry who are willing to share their knowledge and practical skills with you. Some of them will contribute to our magazine on a regular basis, so we would be very thankful for any feedback.

What's inside? Jeremy Holcombe and Lynda Gilmore explain the very basics of building a WordPress site, while Jim Gallaher shares secrets of Self Hosted WordPress and Gary Glasscock tells how to create a Simple Responsive Website.

You'll also find Jeffrey Zinn's great networking concepts, clearly expressed in the article 'How To Speed Up Your WordPress Website', Tish Briseno's tricks in 'How to Enhance your Website with Plugins & Widgets', and Brett Widmann's article covering the basics of keeping WordPress secure from getting hacked.

Don't forget to look through our unique +EXTRA articles. Dr. Yildirim Kocdag presents his 'Beyond Mobile Gestures' and Kevin Donnigan shares some news about 'Mobile Navigation: User Friendly Techniques For Small Devices'. Futhermore, Matthew Ruppert's new input about 'Mistakes' and a special gift – additional ebook devoted to author's WordPress instructions.

Enjoy your reading!

Anna Łakomy

& SDJ Team

Publisher: Hakin9 Media SK
02-676 Warsaw, Poland Postepu 17D
Phone: 1 917 338 3631
http://en.sdjournal.org/

Table of Contents

First steps:

Beginner's Guide to Building a WordPress Website 5
by Jeremy Holcombe
Self Hosted WordPress Based Website From Scratch 13
by Jim Gallaher
Step By Step building a Website with WordPress 23
by Lynda Gilmore
Installing and Creating a Simple Responsive Website 31
With WordPress *by Gary Glasscock*

Getting involved:

How to Enhance your Website with Plugins & Widgets 39
by Tish Briseno
Building Advanced Search Queries in WordPress 45
by Woo Themes
Customize your WordPress Theme the Right Way with 59
Child Themes *by Ruth Maude*
How To Speed Up Your WordPress Website *by Jeffrey Zinn* 63
Developing and deploying a website with WordPress 69
and Git *by Martin Prunell*
Wordpress: Maintain & Deploy *by GJ Petersen* 80
Mobile Navigation: User Friendly Techniques For Small 90
Devices *by Kevin Donnigan*
Holding Down the Fort: Five Tips for a More Secure 97
WordPress Site *by Brett Widmann*

EXTRA +

Beyond Mobile Gestures *by Dr. Yildirim Kocdag* 103
Expert MySQL Design Practices *by Ronald Bradford* 109
How to set up multi-master replication with Galera 121
Cluster for MySQL *by Ashrif Sharif*

Developer@Life *by Matthew Rupert*
1. Mistakes *by Matthew Rupert*

EXTRA ++

Matthew Rupert – extra ebook

Meet the **Developer-Friendly** Payment Solution

3 easy steps to optimized checkouts:

1

Create the checkout page

With Gate2Shop, you can optimize your payment pages by using ready-made templates or by customizing payment pages to your site look and feel.

2

Test and optimize

An effective payment page variant testing tool, A/B Testing helps you gain insight into user behaviour, increase payment conversion in the short and long term.

3

Accept payments worldwide

With dozens of alternative and local payment methods offered in multiple currencies, the personalized checkout allows you to reach users from all around the world.

✔ Easy integration ✔ Cross-platform ✔ Secure

G2S gate2shop Sell. More.

Call for a free consultation: +44 20 3051 0330

www.g2s.com

Software Developer's JOURNAL
new ideas & solutions for professional programmers

Beginner's Guide to Building a WordPress Website

Jeremy Holcombe

This article will explain everything you need to know about building a WordPress website. It will give a little information on what WordPress is and where it came from. This beginner's guide is geared toward those who are new to WordPress, or have never used it before. While building a viable WordPress site is something that many professional companies and individuals will pay for, there are a few steps you can take to building your very own WordPress website.

Introduction to WordPress

First introduced in 2003, WordPress was originally written as a single piece of code that was to be used to enhance and style to typography for everyday writers. Over the last decade that single piece of code has developed into the world's largest self-hosted blogging platform, and then even further, into one of the most sought after website development tools around.

WordPress is an "open source" project, which technically means that there are literally hundreds, if not thousands, of people all over the world working on it. The amount of people that work on WordPress is more than almost any other commercial platform in the world. From website templates to plugins and other add-ons, everything related to WordPress and implementing WordPress is contributed by programmers around the world. This is just one of the many reasons that you will see small, personal websites and large industry and professional companies all using the WordPress layout and code for their websites.

Today WordPress is still used as a personal blogging system, but it is also used as a full website building and content management system. With thousands of themes, plugins and widgets, WordPress continues to push the envelope and will quickly continue to rise as one of the preferred website building tools in the world.

What is WordPress Built On

The whole point of WordPress was to be a desirable personal publishing platform. It accomplished this and so much more. It is a very mature and very stable product built on PHP and MySQL. The system licensed under the GPLv2. With fresh updates always coming out, the WordPress system continues to stay ahead of all other personal publishing software and also continues to grow in the realm of business and personal website design.

Building a WordPress Website

A good website is the key to many things, especially in the world that we live in today. Have something to say? A website is one of the best ways to get your message out. Have a business? Then having

a well designed website is an absolute must in order to develop relationships with customers, make sales, and get your brand recognized. With that being said, having a website is by far the best way to get recognized today, whether professionally or personally.

While there are still several options when it comes to websites, using the WordPress platform is arguably one of the easiest ways to get your own website going, especially if you are a beginner. While there is of course coding and other work involved in building a professional WordPress site, the platform does offer enough that you can get a decent site going with only a little knowledge of how plugins and add-ons work. Follow the steps below and with just a little effort you will be getting your message out with your very own WordPress based website.

**Note: You may be able to jump around some in the steps below. However, following them in the order they are laid out will be the easiest and fastest way building yourself a new WordPress website.*

Step One: Domain Name

Whether you are going to let WordPress host your site or have your site hosted by an outside hosting company you have to have a domain name. There are numerous places on the web to go and pick up a domain name. Most domain names will cost you anywhere from seven dollars to fifteen dollars, depending on what website you purchase from, what coupon codes you have, and what add-ons you choose to buy with the domain name.

Your domain name can be anything you want, provided that the name you choose is available. Want your name as the title of your website? No problem, go see if it is available. Once you have purchased your domain name of choice you will be ready to jump to the next step, which will be to figure out your hosting situation.

Step Two: Hosting

Okay, so WordPress, unlike most other options, allows you to host a domain name for free if you use the *wordpress.org* platform system. There are some other blogging platforms out there that allow for free hosting, however, if you ever wanted to transition into a self-hosted site, the other blogging platforms don't make available the use of their platform. This is just another reason that WordPress is considered by most, including myself, to be the very best website building tool around. With that being said, here are your two options when it comes to hosting a WordPress based website.

1. *WordPress.org* (the WordPress system will host the domain name for you for free and you can use their website and blogging platform)
2. Host the website yourself using one of many great hosting companies. You can still use all of the WordPress platform themes, plugins and add-ons.

Hosting Differences

There are a few things to remember with each type of hosting. Yes, the *WordPress.org* hosted system is free and they do offer you an abundance of tools. However, you will not be able to make as many modifications to your website design as you may like. You cannot

add much code, if any, and you may not be able to link to outside Social Media outlets the way you want to. The system is well written and secure, so you won't have to worry about bugs and spam. Modifications to your site will be limited to what the free WordPress based template you choose allows.

Conversely, hosting your site with an outside hosting company gives you full and total creative control. It allows for any modifications and coding you want and also allow for better social networking, buying and selling of goods, and just about anything else you can imagine. You still get to use all of the WordPress themes, plugins and add-ons, but you can go deeper, add more, be more creative with design, and really get into the building and coding design side of WordPress. You will have to be more aware of security and spam, but there are a ton of applications and techniques that can help you with this.

You will need to pay for outside hosting, but most plans are affordable and if you have to pay more because of traffic, then your site is probably doing well enough that it won't matter. Finally, the difference in the way your domain name looks is as follows:

**Note: Using the WordPress.org free hosting platform your domain will read as www.mysite.wordpress.org. Using an outside hosting solution your domain will read as www.mysite.com (.org, .co, .net, .biz) whatever you choose to buy.*

Step Three: Downloading WordPress

Once you are all set with your domain name and your choice of hosting it will be time to download WordPress. Yes, WordPress is not like other website development tools, you actually download it for free and start using the system. If you have chosen to go ahead and use *WordPress.org* for their free hosting tools, then you simply build a WordPress account and follow the directions given by WordPress.

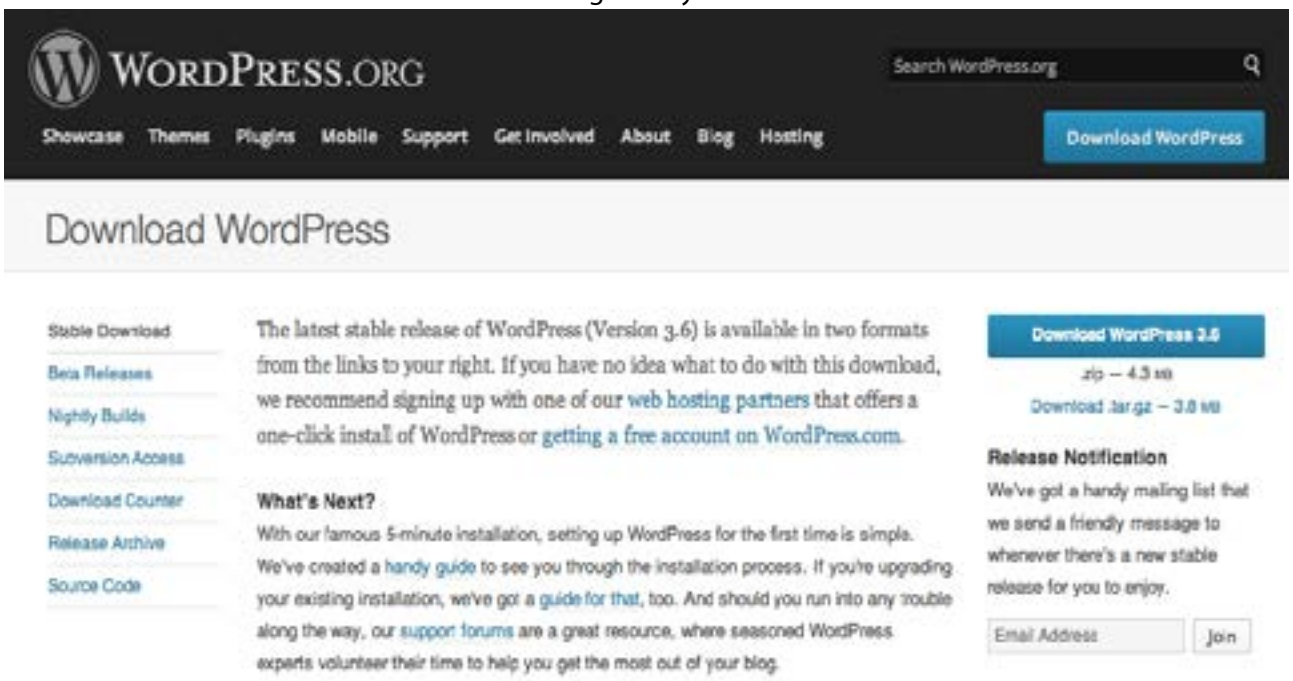


Figure 1. WordPress download page

Your themes and system will already be integrated, it is simply a matter of setting things up how you prefer.

If you have chosen to go with your won hosting, then you need to go download the WordPress system from the site and upload it into your hosting system. This can be done one of two ways.

1. *You can do it through an FTP client (that is an entirely different article)*
2. *You can upload it with a few clicks through your hosting system (recommended)*

Most, if not all, hosting companies offer easy one or two click WordPress platform uploads that you direct right to the domain name of choice. If you do indeed choose to use an FTP client (again, that is for another article), then you can go download WordPress to your desktop from *http://wordpress.org/download/*.

WordPress download page

Congratulations, whatever way you chose to download WordPress you should now be successfully running it on your website domain name. Now it is time to login to your website.

Step Four: Logging In To Your WordPress Site

Another great aspect of WordPress is that once you have uploaded the platform you do not need to stay logged into your hosting account to change your site and log into the site itself. While some high-end changes and coding may need to be implemented via FTP from the server itself, most coding and design can be done right from the backend of the site once you are logged in. That being said, this is a beginner's guide, so even at this point you will probably be slowly mapping your way around the backend of the site to become familiar with how everything looks. Adding custom CSS and messing

Figure 2. WordPress login page

with system files should be left to a professional unless you think you know what you are doing.

Once you have uploaded WordPress into your hosting account and pointed it to the domain name you want you will be provided with a simple login link that reads as follows, *www.mysite.com/wp-admin/*. Going to that provided link will bring up a login screen and you simply punch in your chosen username and password and you will be directed to the editing area of your WordPress website. Whether you want to post pages, articles, upload files, plugins, add-ons, or make entire changes to your site, this is where you will do it. Again, performing all those functions are best reserved for another article, but at this point you should be ready to choose your WordPress theme and move forward.

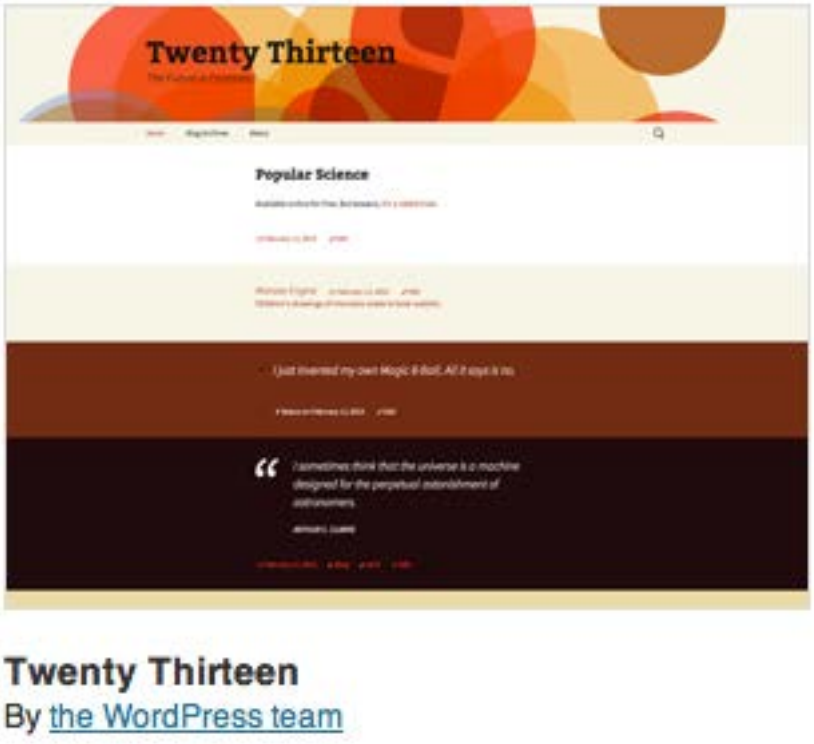


Figure 3. Default Twenty Thirteen theme

Step Five: Choosing a WordPress Theme

At this point you are nearing the end of the process. Lets take this last major step and look at it in two different ways. As stated above, if you have chosen to go the route of being hosted by *WordPress.org*, then all your themes will be readily available and easy to access. Most of them will be free and you will be able to play with a few different layouts, colors and other fun formatting designs and functions. The main downfall here is that you will not have access to many of the most dynamic WordPress themes available from great theme designers around the globe. Don't worry; you will still have thousands of options available to you.

Conversely, if you have chosen to buy hosting and host the site yourself, then you have a couple different options as well. Once you upload WordPress into your system the platform will provide you with a default them that they build for you to use. Right now the default theme is called Twenty Thirteen. As you may guess, they

update that theme every year and add new functions to it and name it after the most relevant year. Example, next year's default theme will be named Twenty Fourteen.

Default Twenty Thirteen theme

Here is where it gets really fun if you are hosting your own site. There are literally hundreds of top-notch designers and companies that have created amazing WordPress themes. While there are plenty of free themes to use, the very best ones are the ones you will have to pay a small sum for. Most of the best ones allow you (or your designer) to do just about anything imaginable. The best WordPress theme designers will offer excellent theme support and guide you through any questions you may have. There are all sorts of themes available, depending on the type of WordPress based site you are trying to build.

Child Themes

You can of course build a WordPress site from the ground up, but you will need plenty of coding and computer writing language experience. That is why many of the top WordPress theme designers (both individual and companies) offer "child themes." A child theme is basically a pre-built WordPress theme that has plenty of built-in functionality. You can use what is there to build your side and use code (CSS and HTML) to do modifications. You can also change theme files using PHP if you want to. A child theme is yet another reason that WordPress is so popular, as it gives even those with little to no website experience a chance to put a website together.

Uploading Your Theme

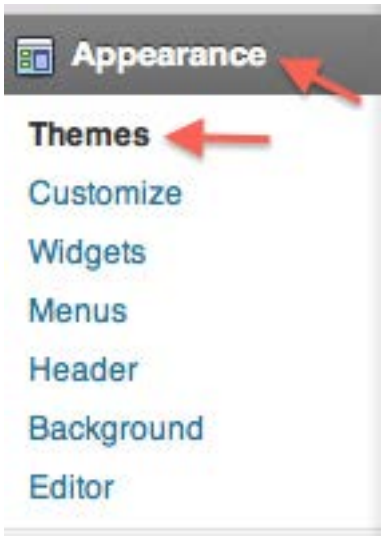


Figure 4. Uploading your theme

Uploading the theme you finally settle on is simple enough. Go to "Appearance" on the left side of the dashboard and click on it. It will bring a popup menu up that will allow you to click on "themes." Follow the instructions that WordPress provides from there. You will simply "upload" the file to your site and activate it.

**Note: if you are uploading themes, plugins, or other add-ons from your desktop you have to compress the file first before uploading. WordPress will only upload compressed (zipped) files.*

Where to Go From Here

You will now be at a point where your site is up and running. From here you have several options. There are literally thousands of plugins you can download, many of which are free. These plugins will allow you to accomplish almost anything you want within your new WordPress site. There are also all types of add-ons you can implement within the site.

You will want to get the most popular plugins, ones that are used to protect your WordPress site from hackers and spam and ones that are used in conjunction with SEO optimization. The plugins and add-ons you choose to implement within your WordPress website will be based on what you are trying to accomplish, as well as what your personal needs and desires are for the website.

Blogging

Finally, remember what WordPress was originally created for… blogging, or personal writing. While the WordPress platform has evolved and become a whole lot more than just a blogging platform, the system is still the most popular way to blog and get your message out. Whether you are putting together a personal blogging website or trying to build a high-end business website, the WordPress platform is arguably the very best website building platform around. If you follow the steps given above then you should be able to get your WordPress based website up and running, even if you are a beginner or new to the WordPress system.

Final Thoughts

The steps listed above will help get you going as far as starting your own WordPress based website. The platform allows for designers and website builders to implement just about any function imaginable. However, there is still the issue of coding and certain CSS styling that you may want to hire a professional to do, depending on the functions and ability you want your WordPress website to have. You may be able to implement a lot of things, but if you are building from the ground up, then you will definitely want to study CSS styling and PHP code, or hire a WordPress website designer to design the site for you.

Summary

This article gave some information on what WordPress is, how it came to be, and what it has become over the last decade. It also provided information to get your own WordPress based website off the ground by following a few simple, uncomplicated steps. Information on how to build the site after you get it going is also provided.

On The Web

- *http://wordpress.org - Main WordPress website*
- *http://wordpress.org/themes/ - WordPress themes directory*
- *http://wordpress.org/plugins/ - WordPress plugin directory*
- *http://opensource.org - Information on the Open Source initiative*

Glossary

Open Source Initiative: The Open Source Initiative (OSI) is a non-profit corporation with global scope formed to educate about and advocate for the benefits of open source, and to build bridges among different constituencies in the open source community. See our about and history pages for more.

Source: *http://opensource.org*

PHP: PHP is a widely used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.
Source: *http://php.net/manual/en/intro-whatis.php*

MySQL: MySQL is the most popular Open Source SQL database management system and is developed, distributed, and supported by Oracle Corporation.
Source: *http://dev.mysql.com/doc/refman/4.1/en/what-is-mysql.html*

FTP: File Transfer Protocol (FTP) is a standard Internet protocol for transmitting files between computers on the Internet. FTP is commonly used to transfer Web page files from their creator to the computer that acts as their server for everyone on the Internet.
Source: *http://searchenterprisewan.techtarget.com/definition/File-Transfer-Protocol*

CSS: Cascading Style Sheets, or CSS, is the recommended way to control the presentation layer in a web document. The main advantage of CSS over presentational HTML markup is that the styling can be kept entirely separate from the content.
Source: *http://reference.sitepoint.com/css/css*

About the Author:

Jeremy Holcombe is the founder and owner of My Writing Tree, *www.mywritingtree.com*, a WordPress website design and content writing company. He has been designing WordPress based websites and blogs for individuals and businesses for over 5 years and also owns several of his own WordPress based websites and blogs. He also writes content for websites and publications, as My Writing Tree also offers a unique lineup of writing services to go along with WordPress website design.

Software Developer's JOURNAL
new ideas & solutions for professional programmers

Self Hosted WordPress Based Website From Scratch

Jim Gallaher

In this tutorial I'll walk you through the steps on how to create your own self hosted WordPress based website from scratch. WordPress is a great way to get a fairly decent website up and running quickly without having a strong understanding of web based languages such as HTML, CSS, JavaScript, and PHP. (Of course if you know these languages, then you can have far greater control over how your WordPress site looks and functions.)

What You Will Learn…

Topics covered in this tutorial will include a basic introduction about what WordPress is, the general requirements you need to host a WordPress site, how to create a MySQL database for WordPress, a walk through on how to install WordPress on your server, some basic security practices, simple spam prevention, and how to create a basic site in WordPress.

What You Should Know...

To follow along with the steps in this tutorial you don't need to be a webserver administrator expert, but you do need to be comfortable with installing programs on a Microsoft Windows XP/7/8 based computer or on an Apple Macintosh based computer, some basic experience with using a text editor, and handling some simple administration tasks in the backend of your website's backend control panel.

Tools for Task

A Domain Name, a Web Server or Website Hosting

First off, before you can do anything, you'll need a domain name and website hosting service that provides PHP version 5.2.4 or higher and MySQL version 5.0 or higher. If you don't have a domain name and/or website hosting, companies such as Inmotion Hosting provide basic hosting packages for $9.00 or less a month that includes the requirements for WordPress. Contact them or your preferred hosting provider to get yourself setup with service, and then come back here to get your WordPress site up and running.

If you already have hosting or a webserver setup and you're not sure if you meet these requirements, check with your hosting provider or server administrator first, and have them get your webserver up to par if necessary.

If you know you meet these requirements and you're saying to yourself "Yes I do! Let's get on with this tutorial already!" Then let's continue on with our shopping list. (Ladies and Gents, the rest of tools you need are free, so keep your credit card in your wallet or purse for the remainder of this tutorial.)

A Copy of WordPress

Next on our list is WordPress. Obviously we won't get too far without it, so head over to *http://wordpress.org/download/* and click on Download WordPress button to download the zip file to a location you'll remember. (Please note that as of this writing, latest version WordPress is 3.6, but it's very likely that you'll see a newer version due to frequent updates. Don't worry if you see something higher than 3.6, the requirements should be the same.)

A FTP Program

Now we'll need a FTP program to upload the files onto the webserver. Any FTP program will work fine to accomplish this task, but for this tutorial I'll be using Filezila as the file transfer program of choice since it's free, they offer both a Macintosh and Windows version of their software, it's very easy to use, and it just works great! To get the program go to *https://filezilla-project.org* and click on Download Filezilla Client all platforms, and select the appropriate download for your operating system (Windows or Mac). When it's done downloading go ahead and install it on your computer. Just follow the on screen instructions to install it as you would any other program.

A Text Editor

The second to last tool we'll need is a text editor. Any basic pre-installed text editor on your computer will get the job done with no problems. However, I suggest using something a little fancier that offers at minimum line numbering and code coloring. Using a text editor with these features will make your life much easier when you need to find and edit a few lines of code that I'll cover in detail later on. My personal text editor of choice is Sublime Text 2 *http://www.sublimetext.com*/2 which works on both Windows and Mac and offers a free trial period. If you want to use Sublime Text, that's great! If you have something else installed that you prefer to use instead, that's fine too. It makes no difference whatsoever to me or WordPress.

A Web Browser

The last thing on the list you'll need is a web browser. I assume you have at least one installed that's fairly up to date, but for completeness I need to mention it. I recommend using one of the latest versions of Firefox, Google Chrome, Opera, or Safari which offer good support for HTML5 elements. For this tutorial, I'll be using the latest version of Firefox.

Creating the MySQL Database

If you've made this far, then I'll assume you got everything you need to begin installing WordPress. Let's first start off by creating the MySQL database that will hold all the content that will go into your WordPress site. You'll need to access the backend of your website, so check with your web hosting provider or server administrator on how to gain access the backend of your website. Once you have backend access, your hosting provider should offer a control panel of some kind which makes it pretty easy to setup a database and other server related tasks. Also, please note that the exact steps on

creating a database will vary from server to server, I can't tell you exactly on where to go or what to click, but I will provide a general overview of what you need to do If you're not sure about creating a database on your webserver or if you have additional questions, contact them for this information since they can provide you with the most exact information.

General Overview

- Create a new database and provide a name for it. For security purposes it shouldn't be easy to guess, but it should have at least some kind of reference to the site you applying it to. For example my website is *jcorbindesign.com*, so a database named, jcorwpdbmain would be a good name that provides a reference to what the database applies to, but can't be easily guessed by hackers. Write down your database name and continue to the next step,
- create a username and a password for the database. We don't want the username to match the database name and we don't want it to be easily guessed either. A good username could be wpjcoradmdb. The password for the database user should be really, really hard to crack. Create a password that uses random uppercase/lowercase letters and numbers. Using symbols will help too. The minimum length that the password should be is at least 8 characters. My hosting provider allows passwords up to 18 characters long, but other hosting providers may not allow passwords to be up to this length. A good example of a strong password could be RoK?GTL~&0SQ Write down your database username and his password, then continue,
- the last thing we need to do is add our database user to the database and provide him with all privileges.

Setup an FTP account

With the MySQL database, database user, and the database password setup (You did remember to write down the database name, username, and password didn't you?) the next logical step is to setup a FTP account so you can access your website's directory from an ftp program such as Filezilla. Check with your hosting provider or server administrator on how to create an FTP account, since the exact process will most likely be different for everyone. Also, ftp accounts can be setup to only connect to a specific directory within your website folder structure and cannot view anything outside of that directory. If you have any questions about this, contact your Website hosting provider or server administrator.

FTP Program Access

The next step is to get Filezila to connect to your website:

- Click on file,
- then Site manager,
- then new site,
- an icon will appear with the name new site,
- change the name of new site to your site's name,
- fill in the host, port, and additional ftp information.

Check with your web hosting provider or server administrator for the proper FTP settings, if you're unclear about them.

When you're done setting up Filezilla, verify that you can connect by clicking on file, site manager, the site that you want to connect to, and then connect. If everything goes well, you should see a message in the message log saying Directory listing successful. (The message log is located approximately in the middle of the screen. Below the message log, there are two windows named local site and remote site. If you have successfully connected to your website, the remote site window should show an empty directory. If the remote directory is populated with files and/or directories, verify that you are installing WordPress in the correct directory of your site and remove any files or directories before uploading the WordPress files to that directory.

Security note: If your web host supports encryption, do yourself a huge favor and use the encryption settings they provide since this will help keep your site more secure when accessing it via FTP.

For additional detailed documentation on Filezilla: *https://wiki.filezilla-project.org/Documentation*

Uploading WordPress to the Server

With your FTP account and FTP access setup, it's time to upload the WordPress files to the server via FTP.

- Unzip the WordPress zip file you download earlier if you haven't done so already,
- navigate to the WordPress unzipped files by using Filezilla's Local site directory menu and make sure in the Local site window, you see a list of directories and files (wp-admin, wp-content, wp-includes, *index.php*, etc.)
- re-verify that your WordPress installation is being uploaded to the correct location on your website in Filezilla by checking that the remote directory is empty and is the proper location,
- upload all the files and directories (wp-admin, wp-content, wp-includes, *index.php*, license, readme, and etc.) by selecting all the files/directories in the Local site window and click n' dragging them over to the remote site window.
- During the file transfer process, for one reason or another, there may occasionally be a file or two that will flag a message "This file already exists." If you see any messages like this popup, just click ok to overwrite the file.

Installing WordPress

Once all the files have been properly uploaded to the server, it's time to go through the installation process.

- Go to the site's address that you uploaded WordPress to. For example, my site is *http://jcorbindesign.com* so I would go here to begin the installation process,
- you should see a message that says "There doesn't seem to be a *wp-config.php* file. I need this before we can get started." Don't panic! Just click the "create a configuration file" button and WordPress will take care of this for you,
- next, you should see another message that says "Welcome to WordPress." And another reminder about what you'll need to complete the installation. (Database name, Database Username, Database password, etc.) Click the "let's go!" button,

- first, enter the name of the database you created earlier,
- next the database username and the database password,
- the Database host is something new. For most will be left at the default "localhost" however it could be something else depending on your hosting provider. Check with your hosting provider or server administrator if you're not sure,
- lastly the Table Prefix is set to wp_ by default, but for security reasons, I highly recommend changing it to something random, like rqw_ so hackers won't know the default table names and prefixes right off hand,
- click the "submit" button. If everything is correct, you'll get a message saying All right, sparky! You've made it through this part of the installation."
- click "run the install" and you'll be taken to the next screen where you'll create the site name, a username for logging into the WordPress backend of your site, (This is a different username than the database username), your administrator password, your administrator email address, and the option to allow or block search engines from indexing the site,
- the site name is pretty self-explanatory,
- the username is for the administrator account that you will use to log into the backend of your WordPress based site. By default the username is set admin, but for security purposes this should be changed to something much more secure. Hackers will almost certainly try using brute force attacks to login into a WordPress based site using admin because everyone in the world that knows anything about hacking a WordPress site, will assume that admin will be most likely be used as the username of choice. By using the default username admin, this only makes it much easier for hackers to break into your site since they just have to know the password. Also, note that the administrator username should be different from the database username,
- the password for the administrator account should, again, be something that isn't easy to guess. A random mixture of uppercase/lowercase letters, numbers, and symbols that span at least eight characters long should make it fairly difficult for hackers to break into your site. The administrator password should be different from the database password,
- enter your email address for the administrator account, this is important if you forget your password and you need it reset. Also this email address serves to notify you of comments from people who comment on your blog postings. I'll cover this in more detail later on,
- privacy is the last thing on the list before we start the installation. If you plan to work on this site for a while and you don't want submit to search engines just yet, you can uncheck this box to block search engines and enable it later when you're ready,
- Click on the "Install WordPress" button when you're done,
- You should see a success message and a login button. Let's go ahead and login!

Troubleshooting

If you get an "Error establishing a database connection" message. This message means that something is wrong with the information you entered or possibly the database host name is different from localhost. Check everything you entered to be sure it's correct. If you're still having issues, check with your hosting provider or server administrator for help.

Logging into your WordPress site for the first time

When you first login, you'll enter in what is called the dashboard. The dashboard is the backend control panel for your WordPress site. On the left side you'll see the Dashboard menu which provides menu items such as Posts, Media, Pages, Comments, Appearance, Plugins, Users, Tools, and Settings. Since we are signing in as an Administrator all of these menu choices are available. However, keep in mind that WordPress offers different user levels and depending on what user level you are signing in as, (administrator, editor, author, contributor, or a subscriber) there will be different menu options able to you. I won't go into detail about the different roles since this requires a much broader explanation. However to learn about user roles see the following link for more information *http://codex.wordpress.org/Roles_and_Capabilities*

Important First Time Settings

There are a few important first time settings that should be taken care of straight away anytime you perform a fresh installation of WordPress.

Change your public display name.

By default, any blog postings or comments you make will by default, show your administrator name. If you recall, I mentioned earlier to change the default administrator name from admin to something else that can't be easily guessed. What this means for security is, WordPress will publicly display your username, so hackers a solid lead on what your username is just by looking at your blog posts and/or comments.

- In the dashboard go to Users and then to Your Profile,
- under Nickname (required), delete the name in the field to something like your company name or first name,
- click on the menu option under Display Name Publicly to the new name you just made,
- scroll down to the bottom and click the Update Profile button to save.

Controlling Blog Comments

WordPress has a few methods that allow you to handle blog comments or more commonly, spam comments. You can either use a plugin to handle this automatically or handle it manually by creating a moderation/blacklist of any words that you don't find acceptable.

WordPress by default is setup so that anyone who posts a comment has to enter in an email address. This email address is used as a way to approve and/or identify a user who posts a comment. If this is the first time this person is posting a comment on your blog with their email address, their comment will be held in the moderation comment list,

until you approve it, trash it, or send it to the spam folder. Once that email address is approved, that user can post comments without requiring additional approval. However, as I mentioned before you can moderate certain words that are present in their post regardless of whether they've been approved and review it before approving it or you can blacklist words and have them sent directly to spam folder.

Controlling Spam with a Plugin

WordPress by itself is pretty feature rich. However, you have the option of installing plugins, which are mini programs that provide additional features that WordPress can't do straight out of the box. To see what plugins are installed already, activate/deactivate, add, or remove a plugin, go to the Plugins menu section. I won't go into detail about plugins since this would require a discussion all on its own, but if you want to learn more about plugins, go to *http://wordpress.org/plugins/*

One plugin I will mention, comes installed by default, but isn't activated. Akismet is a plugin for controlling blog spam. To use Akismet, you must first activate it and then register for an API key on the developer's website. The key is free for non-commercial sites with certain restrictions, but requires a subscription for commercial use. You can read more about their plans at *http://akismet.com/plans*

Controlling Spam without a Plugin

If you're looking for a way to control spam without a subscription based plugin, you can do so fairly easily. In the Dashboard menu, go to Settings, Discussion, and look for Comment Blacklist on this page. You'll see a blank box where you can add words or HTML tags that you want to blacklist. Over the last couple of years I've put together a small list of HTML tags that I've found spammers to commonly use, but legitimate users don't.

The current list I have black is the following below:

- *<a>*
- *<b>*
- *http://*
- **

However, there's still a small possibility that legitimate users might still enter in at least some of these HTML tags without knowing that they're banned. So what's the solution? Well, fortunately WordPress provides a list of acceptable HTML tags towards the bottom of the comment box for each blog posting so users will know what's allowed and what is not. However, we're going to have to edit this list manually by downloading the file using our FTP program, grab it from our remote site that we just uploaded, and edit it using a text editor. The file we need to edit is called *kses.php*.

- To get it, open your FTP program, connect to your WordPress site that you just uploaded, look for the wp-includes directory, find *kses.php*,
- download it to your computer,
- open it with your text editor of choice. (I use Sublime Text 2 for my text editor of choice)

- once the file is open, scroll down to line 372 which will be the start of a comment that says * Kses allowed HTML elements. This is the start of the place where WordPress stores the allowable HTML tags for commenters.

Ok, so earlier I mentioned there are a few HTML tags that blacklist because of spammers, so let's comment out some of those tags.

On line 377 you'll see $allowedtags = array(below that is the first section of code for <a> and http:// tags which is one of the tags that we want to remove from the list.

- Look for the following code:

```
'a' => array(
'href' => true,
'title' => true,
),
```

- add // in front of each line to comment it out rather than delete it.

The reason why I recommend commenting out the lines, rather than deleting them is so if you make a mistake by missing the wrong lines or not commenting out enough lines, you can correct it fairly easily. Also, there might come a time when you might want to allow these HTML tags again. By commenting out lines, rather than deleting them, you can undo these changes.

The following lines of code I just mentioned should look something like this below:

```
//'a' => array(
//'href' => true,
//'title' => true,
//),
```

The next tag we're removing is the <b> tag. Look for the following single line of code below and comment it out the same way we did with the previous one.

```
'b' => array(),
```

The last tag we're removing is . Find the single line of code below and comment it out.

```
'strong' => array(),
```

- Save the file,
- upload it back into the wp-includes directory.

If Filezilla prompts you that this file already exists, just click Ok to overwrite it. Remember, you can always modify the list of acceptable HTML tags by editing this file.

Creating Web Pages

We've taken care of our most important first time setup settings and of course there are plenty more settings that you'll want to explore on your own to customize your WordPress site, but let's move onto how to create webpages in WordPress and then we'll build a navigation menu for those pages.

To create a new page.

- Go to Pages | Add New in the Dashboard menu.
- We're in the Add New Page screen now.
- Let's create a title first by entering in a title in the first box that says "Enter title here". I'll call my first page "Home",
- the larger box is what holds the actual page content. I'll just enter in some text to fill the page. (Feel free to add in whatever you want.)

You might notice that something called Permalink with your site's url and page_id appears with a button that says Change Permalinks.

- Click on this button and another tab will appear with the Permalink Settings,
- for SEO purposes, I highly recommend changing the setting from Default to Post name so each page and/or blog post you make will have a SEO friendly name set to it rather than a page_id=4 or something,
- if you do change it to post name, click the Save Changes button and close this tab.
- Now, go back Edit Page when you're done adding in your content,
- click the Publish button on the right.
- You've now created your first Webpage in WordPress! If you want to add more pages.
- Simply click the Add New link which is located towards the top of the page or off to the left in the Dashboard menu.
- When you're done making pages, let's move onto creating the Navigation menu for those pages.

Creating and Editing a Navigation Menu

If you view your site, you might notice that you already have menu items already for the page(s) you created, but they may not be in the order you want and you might have an extra item called Sample Page. To fix this.

- Go to Appearance | Menus which brings us to the Edit Menus page.

Let's remove any menu items we don't want by editing the list of menu names under Menu Structure. For example, I created a Home page, but it appears twice in the list. I also want to remove Sample page because I don't want that appearing on my site either.

To remove any menu items.

- Click on the down arrow next to page,
- click on remove,
- when you're done, click on the Create Menu button.

The next step we need to take is change the menu name from Menu 1 to something more customized. (I renamed my menu to main so I know what it is in case I make additional menus, but you can call it anything you want.)

- Enter a new name for the menu,
- next, check navigation menu box under menu settings,
- click Save Menu.

Now our navigation will appear correctly on the site.

Rearranging the Menu Order

If you want to change the order of the menu items on your navigation menu, doing so is easy.

- Simply hover over one of the menu items in the list until a four point arrow appears,
- then click and drag up or down where you want that menu item to appear,
- when you're done, click the Save Menu button.

Creating Blog Posts

Creating a blog post is done pretty much the same way you create a page. To post a blog go to Posts | Add New and follow the same steps that you did for creating a page.

Themes

No doubt you've noticed that the look of your WordPress site is already defined for you. If you want to change the look of the site, you can do so by installing, creating, or editing a theme. A theme is a set of files that provides some extra functionality and the overall style of WordPress sites. They include much more than just a simple CSS based style sheet. I won't go into detail about themes in this article, but I will at least point you in the right direction for changing the default theme to something else.

- Under appearance | Themes you can either manage your current themes or install additional themes by either uploading one or searching for one.

If you want to learn more about themes, checkout *http://wordpress.org/themes/about/*

Logging Back Into Your WordPress Site After The First Time

Whenever you are done working in the backend of your WordPress site, make sure you logout. However, you might be wondering "How do I get back in later?" It's simple.

- Go to your site's url and add /wp-admin to the end of it. For example if I was logging into my site *jcorbindesign.com* I would enter in *jcorbindesign.com/wp-admin* and then enter in my username and password.

Summary

This tutorial covered the very basics of installing and setting up your own self hosted WordPress site. We cover topics such as what tools you need to get going, how to create a database, how to install the program, important first time settings, managing spam, creating pages, editing the the navigation, posting a blog, a brief mention about controlling the look of your site, and how to log back into later when you need to make additional changes or updates. There are plenty more things to learn about WordPress, but I hope this tutorial at least helped you to get off to a great start with using WordPress and managing your site with it.

About the Author

Jim Gallaher is a Professional Graphic Designer from Evanston, IL who started JCorbin Design as a personal business with a focus on working with smaller businesses to help them explore marketing ideas, improve the look of their business, and build long-term relationships. He graduated from Westwood College in 2011 with a Bachelor of Science Degree in Visual Communications and in 2006 he graduated from Elgin Community College with an Associates of Applied Science Degree in Computerized Graphic Design.

Software Developer's JOURNAL
new ideas & solutions for professional programmers

Step By Step building a Website with WordPress

Lynda Gilmore

WordPress is one of the most widely used Website and Blog building platforms around, powering over 65 Millions website globally. There are many notable sites that run on the WordPress platform some of these include "The New York Times Blog", "CNN Blog", "Forbes", "Mashable" and even The Rolling Stones and Katy Perry's Websites. So if you are considering building a website or a blog with WordPress, then you are in good company.

However, before you rush off and start your online venture there are a few things you need to consider.

What you will learn….

- How to build a simple plan for your website
- How to do effective market research
- Choosing your Domain Name
- What to look for in a good Hosting Provider
- Linking your Domain Name to your Hosting
- Installing WordPress automatically
- Installing Wordpress manually

What you should know….

There is no pre-requisite knowledge or skills required to understand this tutorial, except how to do a search and use a browser. Here we are starting at the very beginning of what every good web developer should do when building any website regardless of the platform they are building it on.

Planning your Website

First and foremost you need a plan. You need a clear and concise idea of what type of Website you are going to develop and what type of information you are going to present your readership with. The fact is you can build just about anything you want on the top of your WordPress installation, a blog, a wiki, an online shop, the list goes on and on as there are WordPress plugins for just about anything you can think of.

Having a plan will help keep you on track during the development phase of your website and also help you structure your navigation and content accordingly. The simplest way in which to develop your plan is to get a blank piece of paper, lay it out landscape and put your website name at the top. Next put your top level navigation underneath the title, for example "Home", " News", "Products", "Services", "Shop", "Contact Us", "About", whatever you want to be the main focus of your site. These heading will become your menu

navigation for your site. Under these heading your can place your sub menu topics.

Here is an example of a site plan I recently put together for a client of mine we just nutted this plan out together in about 20 minutes but it gave us both clear direction of what content would be required and a good base structure to work from. Simple but effective!

Figure 1. Example

How to do effective market research

Once you have your plan in place the next thing you should do is your market research. Why? Well there is no point in building a website if there is no one interested in what you have to offer. And just because your Mum said it was a good idea is not a good enough reason to spend hours building a website that might well end up just another lonely lemonade stand….(*my take on websites that no one ever visits*) .

So how do we find out if there is a market and an audience for our product or service offering.

Ask Google of course!

But more specifically, we check out the search volume for the keyword phrases that we think our potential clients and customer might be searching for.

To do this we need to sign up for a free Google Adwords Account and use the Keyword Planner. At this point I cannot go into depth on how to use the Keyword Planner as that would be a tutorial unto itself. If you have never used the Keyword Planner before Google do provide very good instructions to get you started.

Using our example above, we use our list of top level terms (products or services) to find out if people are entering these search term and also just how many people are looking for our products and or services.

Choosing your Domain Name

Choosing a good domain name is very important. You do not have to use your business name as your domain name. It's actually better to use a relevant domain name that's related to your product or service. For example if you are a dentist or a service provider in a particular town or city you might want to try and get a domain name that has

your locality included like *www.dentistparkville.com*, this is called an exact match domain and helps greatly with search engine ranking when people are searching for local services.

There is a reason why I am suggesting that you chose your domain name at this point in the process rather than being the first step, that reason is that your research may reveal a particular term or phrase that people are searching for regularly. If at all possible try and include this term or phrase in your domain name like "Home Care" as in the example above. These domains work very well with the search engines.

Now we need to test our selected domain name and see if it is available to register. So next go to *google.com* and type in "domain tools", there's a specific tool on *domaintools.com* called Domain Suggestions. I really like this tool because you can type in a number of keywords and figure out if the domain that you want is taken and it will also give you some suggestions on other domain names that are related to the specific keywords you have selected.

Once you find a suitable domain name that is available then you need to register it. There are literally hundreds of domain name registrars around.

Depending on your geographic location you will have top level domain extension for your location, for example in Australia our top level domain extension are ".*com.au*", ".*net.au*"".*org.au*" and so on. If you want to get a domain name with your top level domain extension you will have to purchase your domain from a registrar that can provide domain names for your region. Shop around as there is quite a difference in the price you will pay for your domain name. An example here in Australia is Melbourne IT, they charge $140.00 for 2 years domain name registration for a .*com.au* where as Crazy Domains cost $24.00 for exactly the same service. Guess which one I use…

Do not buy your hosting at this point until you read the next section.

What to look for in a good Hosting Provider

Now you have your Domain Name the next step is to select a good hosting provider. Again, like domain name registrars there are hundreds of Hosting companies out there all vying for your business.

To run a WordPress installation there are several server capabilities required and or preferred. These are as follows;

- Preferably cPanel Hosting
- PHP 5.3 – Preferred Minimum PHP 5.2.4
- MySQL 5 Database capabilities – A must have
- Apache mod_rewrite
- 99% Server Up Time – Preferred
- Secure Hosting Environment - A must have
- Good Customer Service – A must have
- Affordable pricing structure - Preferred

Ok, now lets look at why these features are needed.

1. cPanel Hosting make your life so much easier and offers a lot of other benefits like email set up and one click install. Installing Wordpress va cPanel is a breeze.

2. PHP 5.3 is the latest version of PHP so it is always good to get the most up- to-date software release and also PHP 5.2.4 has known security vulnerabilities that can be exploited by hackers.
3. MySQL Database – Wordpress is a database driven application therefore it is imperative that wordpress has its own MySQL Database.
4. Apache mod_rewrite need for permalinks (Search Engine Friendly URLS)
5. 99% Server up time. More importantly is the downtime when your website is not available. At 99% Up Time the annual Down time is 3 days 15 hrs, at 98% Up Time the annual Down time is 7 days 7 hours, at 95% Up Time the annual Down time is 18 days 6 hours. As you will see it might not look like a lot when looking at the percentages but there is a significant difference over time.
6. Secure Hosting Environment – You want to try and keep hackers out wherever possible therefore having a secure hosting platform is the first step in protecting your site and your work.
7. Good Customer Service - I prefer to get hosting that has 24/7 customer support as the hosting provider you chose might be on the other side of the world and it you can only get assistance in the middle of the night that would not very satisfactory. Also I recommend you read their customer reviews.
8. Affordable pricing structure – A lot of hosting companies offer a "Honeymoon" pricing offer to get you started. Make sure that you know what the real ongoing hosting costs are going to be after the honeymoon is over. Also ensure that there are scalable pricing structure if your website out grows the initial hosting package size.

My personal favorite, and I have no affiliation with them, is Site Ground. They cover all of the points outlined above and they have servers worldwide.

Linking your Domain Name to your Hosting

Once you have your domain name registered and you hosting account set up you should receive a confirmation email from both your domain registrar and your hosting provider. Keep these emails in a safe place as they will contain your login details and other important information for both accounts.

The email you receive from your hosting provider will also have your "Domain Name Server" information this will look something like this *ns1.hostingcompany.com* and *ns2.hostingcompany.com* you will need this information when you log in to your domain registration account.

When you log in to your Domain Name account you should find an area where you can update your domain name server information it will look something like this.

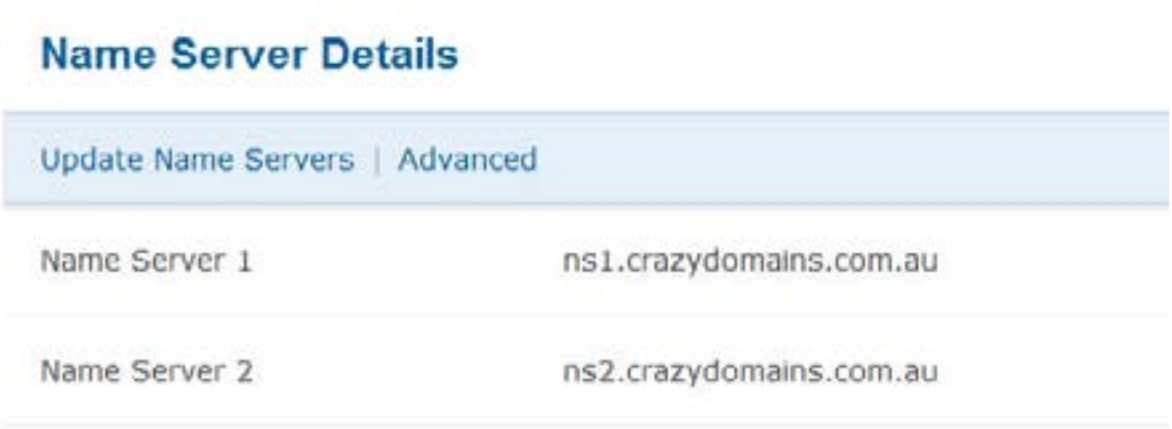


Figure 2. Account

Click on the Update Name Servers Link and another screen will appear where you enter your new name server information.

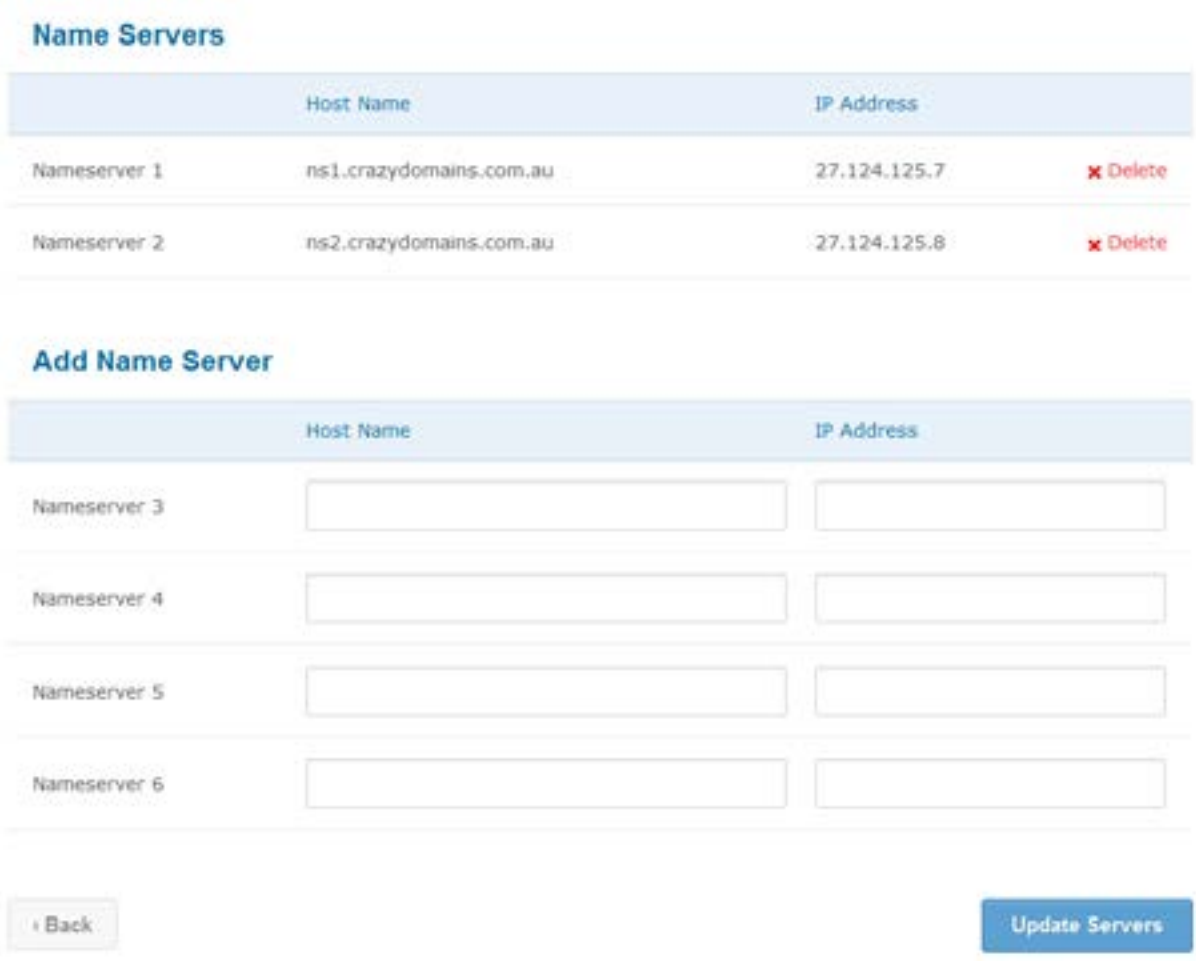


Figure 3. Domain Name account

Please note you're your domain registrar setup may look slightly different from the example but the principle will be the same.

Once you have your new name server information entered don't forget to delete the old name servers otherwise this will create a conflict and the domain name will not resolve. Also don't forget to

click update servers. Domain name delegation can take up to 48 hours for the domain name to resolve but I usually find that they become active within an hour or two. So be patient. Once your domain name has resolved you will be able to type in your domain url into your browser and generally see a landing page or something that looks like this.

Figure 4. Index

Now we are ready to install WordPress

Installing WordPress automatically

Firstly you will have to log into your hosting cPanel account. From there you will see a panel that looks like this.

Figure 5. Panel

In this example there are two ways to install WordPress, either by pressing the WordPress icon or by using Softaculous. Some cPanel hosting also has a site building program called Fantastico, this is also capable of doing an automated install of WordPress.

In This next example I selected the WordPress icon and the following screen appears.

Figure 6. Screen

This screen is pre-populated with information, you can change the detail as you like. The first step is to choose your protocol, that is if you want the "www" in front of your domain or just *http://yourdomainname.com.*

If you are installing to your root directory leave the "In Directory" field blank

Next enter your site name this is different to the url you can call your site whatever you want like Online Home Care Products as an example. Next you enter your site description, slogan or tag line if you like.

Pick a good username and a strong password that preferably, incorporates letters and numbers and even a special character symbol to add to the strength and security of your admin login. Next enter you email address where you want to receive your site administration emails and notifications.

Finalize the rest of the fields as you chose. Then click "Install" and let the installer weave its magic!

A progress bar will appear and once it is finished you will see a message like this.

Figure 7. Message

This message contains your frontend site url and your back end login url. You should also receive and email with your log in details that will be sent to the email address that you entered during the

installation process. If you visit your website address now it will look like this.

Figure 8. Your Website

So there you have it your WordPress site is installed and ready for you to start enhancing it with your own special touches.

In our next tutorial we will go through configuring your WordPress settings in the administration area as well as installing a theme, installing plugins, setting up menus and understanding the difference between pages and posts.

On the web

- *http://www.howtowordpress.com.au* - Our WordPress video tutorial website.

About the Author

Lynda Gilmore has been developing websites over the past 7 years through her own website development company Weblinx Marketing she is also co-founder of the website "How to Wordpress" with her business partner Chris Green.

Lynda has also been engaged as a trainer and presenter by Federal, State and Local Government organisations in Australia to teach small business owners how to get their business up and running online. If you would like to know more about her experience you can find Lynda's full business experience profile on LinkedIn.

Software Developer's JOURNAL
new ideas & solutions for professional programmers

Installing and Creating a Simple Responsive Website With WordPress

Gary L. Glasscock

WordPress has been on the rise as a web development platform for the past several years. I began developing websites professionally with WordPress in 2010, but had been using it to build my own sites for a while longer. One of the first things a developer new to WordPress has to learn is how to install the core software properly. With that in mind let's begin.

What you will learn...

- Three methods to install WordPress
- Basic WordPress customization

What you should know...

- How to download software from a website
- How to access your web server
- How to use and be comfortable with a plain-text editor
- How to use an FTP client
- Have a web browser

Since this issue is the start of a WordPress Section, I thought it appropriate to start with something similar to a "Hello World" type of tutorial. We won't actually write the code that says, "Hello World!", but we will still see it on the home page once we install WordPress.

The first thing you need to do is to check to see if your web server meets the requirements for a WordPress installation.

WordPress requirements

- PHP 5.2.4 or greater
- MySQL 5.0 or greater
- The mod_rewrite Apache module

Now that you've checked and everything is up to spec, let's do the installation of WordPress.

There are three variations on the install process for WordPress. The easiest, least time-consuming method is to install it from your web server's control panel. This is usually done with another program such as Fantastico, Simple Scripts, or some other script installation program. It is very simple and if you do not care about security it is the quickest and easiest method to get started with WordPress.

The second variation is to install it yourself using the "Famous Five-Minute Install" method. You will need to have access to create databases through your website control panel and an FTP client. The steps are:

- Download and unzip the latest version of WordPress from *http://wordpress.org*
- Create a database for the WordPress installation with MySQL.

This is usually done through your website control panel and takes less than a minute to complete

- Upload the unzipped WordPress core files to your webserver to the directory of your choice.
- Type the URL of the website in the location bar of your browser and you should be presented with a screen like the image below.
- Run the installation script using your web browser and follow the instructions. *(http://yourdomainname.com/WordPressInstallationDirectory/)*

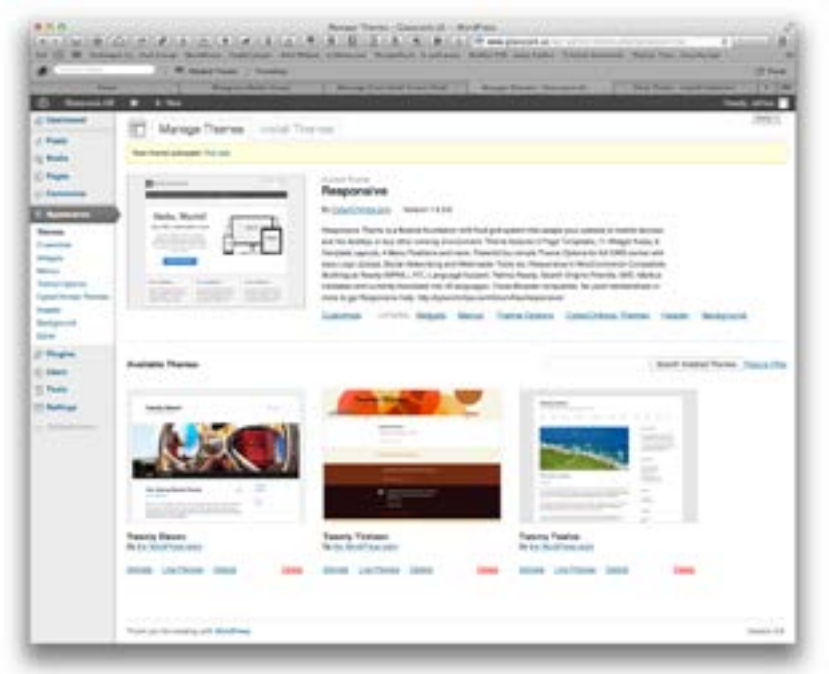


Figure 1. Input your information here and click "Submit." Almost done!

The next page you will see is to setup the Admin username and password. It's important that you choose something unique for the username and something completely and totally abstract to anything and everything for the password.

After that your installation is complete and the final page you see is telling to you go to the login page. Click that link and you are done. Now all you have to do is login and then begin your customization.

The third method is a completely manual but gives you total control over the installation. If you've done everything correctly, the first thing you will see when you type the URL of the site in the address bar of your browser will be the standard index page for WordPress which will show the "Hello World" post.

The first two steps of this method are the same as the "Famous Five-Minute Install" method (look above) so we will skip those steps. You will need to pick up a little something extra though to complete the WordPress installation.

- Visit *https://api.wordpress.org/secret-key/1.1/salt/* and you will want to copy what you see on the screen to a text-editor file for future reference.
- After that step we then use the below steps to complete installation.

- Before you upload the core files you need to create a *config.php* file
- Find the directory on your computer where you unzipped the core WordPress files and look for a file named *config-sample.php* and open it in your favorite text-editor. For future reference though you might want to have those files in a folder you have created for your client. You never know, at sometime in the future you may need those files.
- Once open scroll down until you see the following in the *config-sample.php* file and insert the proper information:

Listing 1. Information

```
<<LISTING1>>
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', '');

/** MySQL database username */
define('DB_USER', '');

/** MySQL database password */
define('DB_PASSWORD', '');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
<</LISTING1>>
```

- Now continue to scroll down through the file until you see

Listing 2. Example

```
<<LISTING2>>/**#@+
 * Authentication Unique Keys and Salts.
 *
 * Change these to different unique phrases!
 * You can generate these using the {@link https://api.wordpress.org/
secret-key/1.1/salt/ WordPress.org secret-key service}
 * You can change these at any point in time to invalidate all
existing cookies. This will force all users to have to log in again.
 *
 * @since 2.6.0
 */<</LISTING2>>
```

in the comments. Just below that comment section you will see define('AUTH_KEY') along with 7 more "defines" under the define('AUTH_KEY'). What you want to do here is to copy all the data you pasted into the keys document you created and then paste it into the *config-sample.php* file. Here is the PHP code you are specifically looking to replace in the *wp-config-sample.php* file.

```
<<LISTING3>>define('AUTH_KEY',         'put your unique phrase here');
define('SECURE_AUTH_KEY', 'put your unique phrase here');
define('LOGGED_IN_KEY',   'put your unique phrase here');
define('NONCE_KEY',       'put your unique phrase here');
define('AUTH_SALT',       'put your unique phrase here');
define('SECURE_AUTH_SALT', 'put your unique phrase here');
define('LOGGED_IN_SALT',  'put your unique phrase here');
define('NONCE_SALT',      'put your unique phrase here');<</LISTING3>>
```

Here is a sample code listing you might receive from the *WordPress.org* secret-key service:

```
<<LISTING4>>define('AUTH_KEY',         ':^-T@D|?0D:w M2](_@=/mtK~ENNJa+AxP%/
  {SI!$|-_FNwoqEiNW]Wuk-}W7-}t');
define('SECURE_AUTH_KEY', 'W|H:;FIhU)==P0SsQW!!|Cbbal-J20!JcFE+1e9J/
  [^q6mts?7-:<zRrNpc`L_fO');
define('LOGGED_IN_KEY',   '02@tmo~$@JUW}DGQ>imY:}f&Yc:z+xZz>_]w LO-
  S,1:gB9d#/w$4W{LtqP^}4|u');
define('NONCE_KEY',       '{4|h}MhB@%>l.jPKy!3gYJ+?frAU*}^PcnOD$-
  KSB`MlEJ}0~zU0`<eJjTCc_xoF');
define('AUTH_SALT',       '<t7*d*QE}t]]89|1u~?zGH#^1*+!!-a!)?px>mI]=?sdrQv
  n3G.V;[kEAFaKq#`_');
define('SECURE_AUTH_SALT', '^e=4m)Zj!^pN-_/O8T_t!#56-Y2VzO/
  S+^fUOe?>x|gK2gj3vB+7kj`N{Zpc9Cq@');
define('LOGGED_IN_SALT',   ',SL+|4bwR+C%4RKLr!:u BR$6cq:nnQ9jcr{C-
  #+K{koEd?g`XvO>cEYs8{g/}yA');
define('NONCE_SALT',       'mr[t*Ql1+ah+JI0v^%,@   MUM),NTta)6i6qe]i]0/
  M+2]Vmay:=Jj]t_+2-2V<swZ');<</LISTING4>>
```

- Similar code is created when you use the "Famous Five Minute Install" method, however, in order to make this even more secure you can add your own string of random numbers or characters to the randomly generated key strings already present. If you do decide to do so, find a random character generator online and run it a few times and get some characters you can add to the above code, or if you prefer, just make up some on your own. It's not like you will have to remember them, it will just make the site even more secure.
- Now save the file as *wp-config.php*
- Upload the core files and the newly created *config.php* file and when done you should be ready to go. Just type your domain in the address bar of your favorite browser and it should render perfectly and your installation is complete.
- Now that the installation is complete, what next?
- Well this is where the customization part comes in. To begin you should go to *http://wordpress.org/themes/* and begin your search for a theme to use for your site. It doesn't have to look exactly like what you want, but it's good if it does resemble the structure you wish to use. Colors can be customized as well as the CSS, PHP, and HTML code within WordPress. You just basically want to have something to start with. Of course there are other possibilities, but for now we'll work with a theme from the WordPress Repository.

A Note On Free Themes

The WordPress Repository is the #1 best place to get free WordPress themes and please be aware that free themes you download from other sources may have extra code in them that you do not ever want on your site or your clients' sites, e.g., internal links to porn sites, phishing sites, and possibly a trojan horse or virus.

The theme I will be using in this part of the tutorial is Responsive. It is a free theme available through the WordPress Repository and I do like it quite well. It's like some other responsive themes I've used in WP which make creating a responsive site a breeze and cuts down on your development time, allowing you to charge your clients or customers less and still have a higher profit margin. Also, most themes you get are fairly easy to customize. This one is a bare framework just begging for you to customize it.

I do want to mention that what I am going to show in this article is some basic customization that can be done with a free theme from the WordPress Repository. If there is to be a lot of customization to the theme, then you should use a Child-theme, which (author's name) has written about in this issue. Please when making serious code customizations to your theme, create and use a Child-theme. Believe me, you'll be glad you did in the long run.

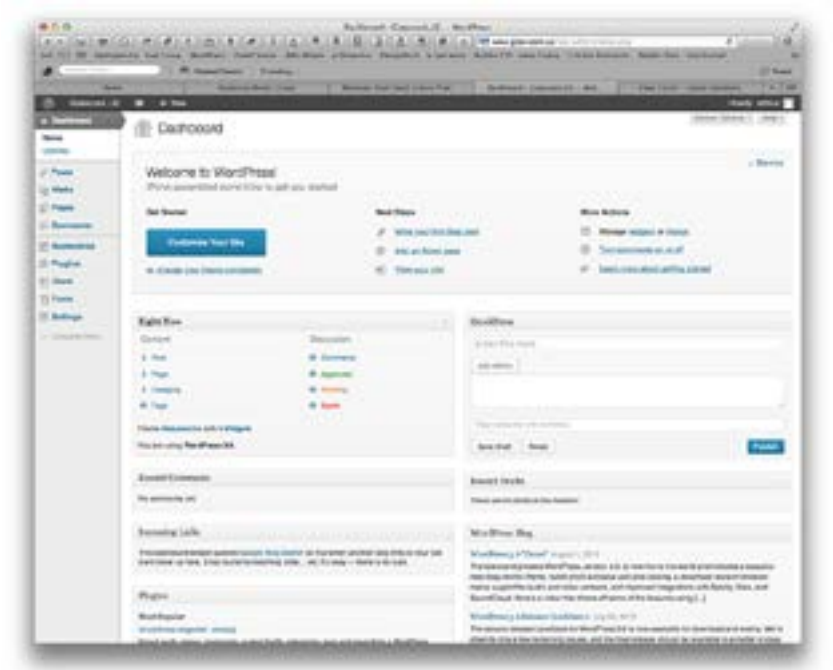


Figure 2. WordPress Dashboard

From the above screen you can do anything in WordPress. First though, we want to make sure we're using the correct theme.

On the left is the standard WordPress Admin menu. In order to change themes we need to go to Appearance>Themes and then we'll see a screen like the image below.

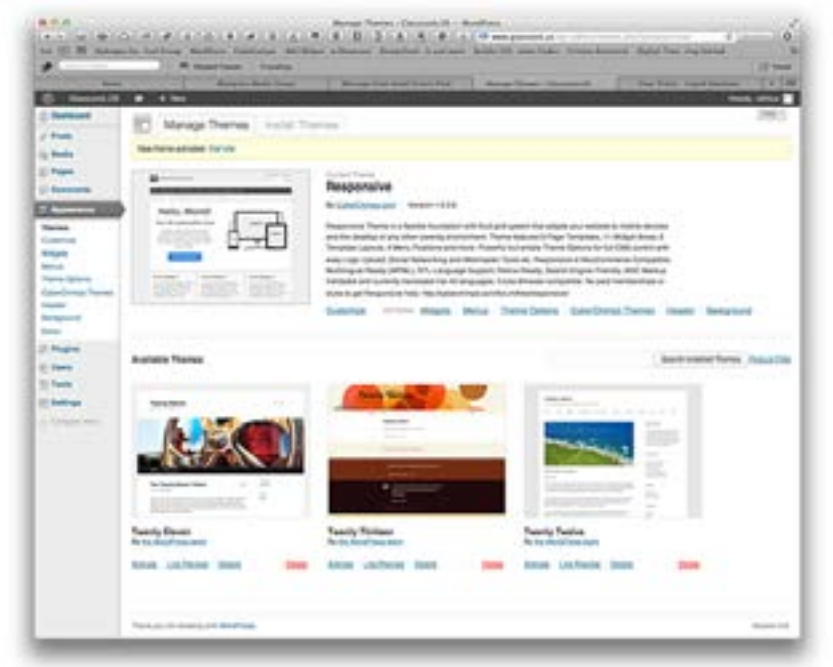


Figure 3. WordPress Theme Page

Since we are already in the Appearance menu, let's just stay for a few more steps and get this site rocking.

Go to Appearance>Header and you will be taken to a page where you can upload the header image of your choice, be a logo or a banner that goes all the way across the page. Standard sizing issues exist here like any other site you may have built so keep that in mind.

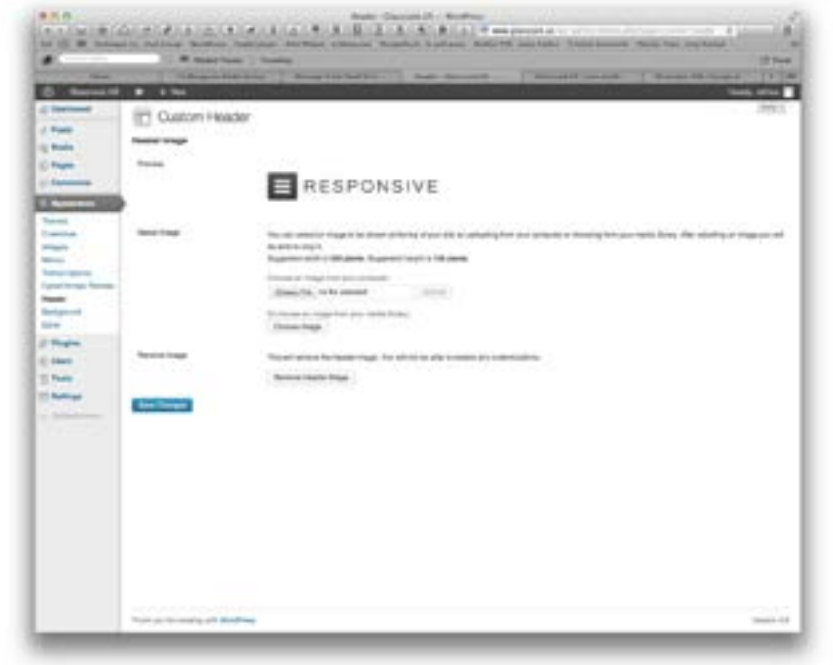


Figure 4. Change Header Image page

After you have saved that let's go to Appearance>Customize and change a setting or two.

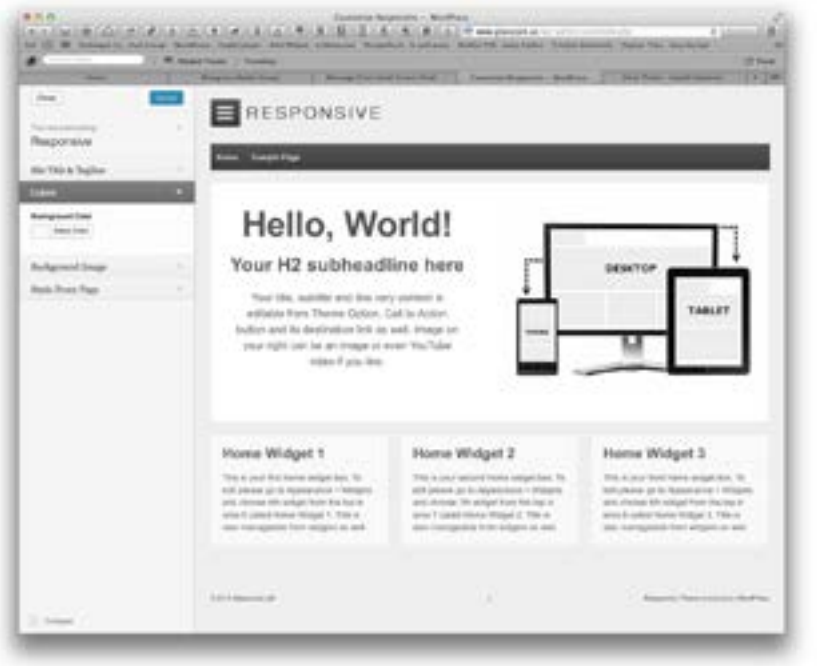


Figure 5. Customization Page

Now let's click on the Color option and it will expand. Now if we click on "Select Color" the section expands further and offers us a field to put in the hexadecimal equivalent of the color we wish, or you can pick the color from the color picker that is presented below the hex input field.

You can make a few other changes here such as, background image, static front page, even the tagline for the site. When you're finished click on the "Save & Publish" button and the settings will be saved and will show up immediately on the site.

There are other customizations available to you from the Appearance menu. You can change the content of any of the widgets you see on the website. You can also click on the "Themes Options" and do more customizations such as setting social icons, custom css coding, custom javascript snippets to be loaded in the header or footer, there's even a section for Webmaster scripts from Google, Yahoo, Bing, etc. The "Theme Options" section is inherent to the Responsive theme and other themes by CyberChimps, the makers of Responsive.

This will get you going on building a basic site with WordPress, but again, please read (author's name) article on Child Themes for future-proofing your customizations.

Summary

In this article you've been given step-by-step instructions on the three installation methods available to install WordPress. The article also goes in to some basic customizations that can be performed on the Responsive theme, a freely available theme on the WordPress Respository.

On the Web

- *http://wordpress.org* – The WordPress main site.
- *http://cyberchimps.com* – The CyberChimps website.

About the Author

Gary L. Glasscock has had an online presence since 1986 and coded his first webpage in 1993 after looking at the source code of websites online at the time in order to learn HTML. Over the years Gary has learned XHTML, CSS, CSS3, HTML5, Javascript, C++, and PHP. Along the way Gary also learned about marketing and advertising, and is President of Bluegrass Media Group, LLC, a digital marketing & design agency that focuses on bringing new strategies and tactics to their client's marketing profile in order to assist them in accessing profits from otherwise unreachable market arenas.

Software Developer's JOURNAL
new ideas & solutions for professional programmers

How to Enhance your Website with Plugins & Widgets

Tish Briseno

One of the things that really sets apart the WordPress platform is their Plugin feature. Essentially it allows you to "plug in" custom functions or features to your website. Plugins can transform a simple website into a blog, membership website, e commerce store, forum, or even an online magazine in minutes instead of hours. Learning about this functionality will save you hours that you would normally spend on development!

What you will learn…

- What is a Plugin?
- Why do you need Plugins?
- What is a Widget?
- Why do you need Widgets?
- What to consider before installing a Plugin/Widget?
- How do you install a plugin?
- How do you install a widget?
- What are the most Popular Plugins & Widgets?

What is a plugin?

A plugin is a small little program that when installed into your WordPress platform, adds additional features. Think of it like an electrical outlet, which are very common in our homes and places of business. We use electrical outlets to plugin items like lamps, printers, computers, fans, and appliances like refrigerators. Well WordPress Plugins work in a very similar fashion; they allow you to plug in certain features like social media sharing capabilities, enhanced security, forums, etc. This is awesome because it allows you to customize your website and add features in a flexible manner. You can access free plugins via the WordPress Plugin Directory. With over 26,000 Plugins available in the Plugin Directory it's safe to say the feature you'd like to add to your website is available.

Why do you need Plugins?

Imagine for a moment that your home didn't have electrical outlets. All of your TVs, fans, lamps, computers and appliances were built into your home instead. Not only would this be more expensive when building your home, but it would also mean less flexibility. What if you decided that you wanted to move your lamp to a different room? Or if it was time to replace your 5 year old clunky computer? You'd probably spend hours tearing down walls and rewiring the electrical just to make it work. Life without electrical outlets sounds like a nightmare. Well the same is true of life without Plugins. In this ever changing world of technology, website features are changing

with a rapid pace. On top of that it seems like every year a new social network comes into the space. For example, wouldn't it be hard if your website was only programmed to share social updates to MySpace? What would take hours of programming in the past, only takes a few minutes and a few simple clicks of a mouse when using a plugin. Plugins can easily be updated to include a new social network or a few feature- all you have to do is install a simple update. The functionality of Plugins can also greatly reduce the cost of development of a website.

What are widgets?

A widget is a small little program that only works in a specific area on your WordPress Website. They only appear in a "widgetized area" of your theme. Widget areas are predefined blocks where you place widgets, usually the sidebars and footer. Upon first glance, Plugins and Widgets almost seem like the exact same thing. They both add functionality to your website. They both are installed in the same manner. But the key difference is that Widgets are only used in certain areas whereas Plugins can have more of a blanket effect on a website. You can liken it to the difference between a chandelier and a nightlight. A chandelier can light the whole room whereas a nightlight can only light a very small portion of the room.

Why do you need Widgets?

Widgets offer many benefits to both the developer of the website and the owner of the website. Most likely over time owners want to add additional functionality to the website and widgets allow you to add and edit the functionality easily. Sidebars and Footers are very important areas to your website. They complement the main content of your website but can be used to get your website visitors to take a particular action, like signing up for your newsletter, liking your Facebook page or even submitting an estimate form.

The Importance Of Being Selective When Installing A Plugin

Since Plugins are basically a small program, they are susceptible to hacking, bugs, or abuse of security just like every other program out there. If you wouldn't want to hire an inexperienced contractor to work on your home, then don't install plugins that are poorly developed as they will only cause you problems. Some of the problems you'll run into with poorly developed plugins can range from being too complex, not operating with the most recent WordPress version, or decreasing your website speed. These are all reasons to research every plugin before you install it. In the similar way you'd research a contractor before hiring him to work on your home.

Which Plugin should you install?

Before installing any plugin, you should consider what need you are trying to meet and if the plugin version is regularly updated. Old Plugins are notorious for being defective resulting in major issues.

Some questions to ask yourself are: whether you need the plugin or not, does it meet your need, can you live without it and is there a better option you can choose? Your goal is to choose a Plugin that has a high rating, works with your version of WordPress, is updated frequently/recently, and created by a trusted developer.

The Rating Of The Plugin

Plugin ratings begin from one star to five stars. Usually, a five star plugin has only one rating. Steer clear from Plugins that only have one rating and a few downloads. Make sure to use a plugin that has numerous ratings and downloads. 3.5 stars is the average rating for the most common and useful plugins. To measure ratings, pay attention to how many times the plugin has been used, its popularity, plugin author, update history and how often it has been recommended throughout Wordpress system. Find out how many downloads it has, who is recommending the plugin, what they are saying about it and why.

Trustworthy Plugin Developers

Would you hire someone if you know they're going out of business soon after? Probably not. The reason why is all comes down to support. iIf something breaks you want to make sure they'll be around to stand behind their work. The same is true with it comes to authors of Plugins. You want to only download Plugins that are actively being supported by trustworthy authors. By taking a few minutes to do a google search to see how reputable the author is, you'll be saving yourself some hassle down the road.

The ability to easily install features into your website far outweigh the time it takes to shop around for a good plugin. When making a choice of plugins to install, be aware of the risks that your website will be exposed to.

How do you install a Plugin?

There are two ways to install your WordPress Plugin, through your WordPress Dashboard or FTP.

How to Install your WordPress Plugin- via WordPress Dashboard

If your Plugin is one that is found in the WordPress Plugin Directory you can easily install it via the WordPress Dashboard should work just fine.

When on your WordPress Dashboard click on Plugins > Add New

Go ahead and type the name of the Plugin you need in the Search field.

You should see listings of Plugins show up, when you find the one you need go ahead and click install.

How to install your WordPress Plugin- via FTP

The only time you'll really need to install a WordPress Plugin via FTP is when you purchased a premium plugin. When you purchase a plugin you'll be supplied with a folder containing your Plugin.

To install it via FTP you will need the following information:

- Hostname
- FTP Username
- FTP Password
- Connection Type

If you don't have the above information, please contact the support for your web host provider and they'll be happy to help you setup a login for your FTP account.

Step One: With your FTP program (I personally use FileZilla) access your website's server.

Step Two: Navigate to the wp-content > Plugins folder

Step Three: Upload the Plugin to this older

Step Four: Logon to your WordPress Dashboard and click on "Plugins" to see if the plugin you just uploaded is in this area.

Activate the plugin and Configure it's settings

Once the plugin installs onto your WordPress platform, you still have one more step to complete- click on Activate Plugin. This is like an on/off button for Plugins. Plugins that are active are "on" and plugins that are inactive are "off".

Once your Plugin is installed and activated you may need to configure the settings of your WordPress Plugin. Majority of WordPress Plugins require some sort of configuration of it's settings via your WordPress Dashboard.

Try looking under the "settings" tab. Unfortunately, not all Authors of Plugins place their settings in the same area- so if you are unable to find it there you may have to look under other tabs to find it. You can also try looking under Tools, Appearance, or your Plugin may have created it's own new tab.

If you're unable to find the settings for your Plugin check the Plugin's official page or documentation that may have been included with the Plugin. If you installed the Plugin via the WordPress Plugin Directory visit the Plugin's page and check under the "Installation" or "FAQs" tab.

Majority of Plugin authors provide clear instructions with how to configure their Plugin. If you're unable to find it, I suggest contacting the author for support or using search engines to find the answer.

How do you install a Widget?

When on your WordPress Dashboard click on Plugins > Add New

Go ahead and type the name of the Plugin you need in the Search field.

You should see listings of Plugins show up, when you find the one you need go ahead and click install.

Once the plugin installs onto your WordPress platform, click on Activate Plugin.

Now navigate to Appearance > Widgets

Open the widget area. On the left are all the widgets available and on the right hand side in a column are the options for where you'd like your widget to go to (the name of the widget areas varies according to the theme you're using).

With your mouse drag a widget to the right column widget area.

What are the most Popular Plugins & Widgets?

The following are the Plugins that I keep on file and use depending on the needs of my website project. While majority of the following resources are free a few are premium plugins which means that they do have some sort of cost associated with them, I've recommended

them because I've found in my experience that they are the best value.

WordPress Optimization Plugins:

- Akismet: Kills those Spam comments. It comes standard with your WordPress install and all you have to do is sign up to get the API key http://wordpress.org/plugins/akismet/
- WP Super Cache: The little plugin decreases the load time of your website! What it does is it makes a static copy of your website and stores it on your server. That way when someone visits your website, the server gives them the cache copy instead of generating your website, which takes a lot longer http://wordpress.org/plugins/wp-super-cache/
- Google Analyticator: Google Analytics is a great way to track how much traffic your website gets and where it's coming from. This plugin is the easiest way to set up Google Analytics on your website and allows you to see your recent traffic stats from your WordPress dashboard http://wordpress.org/plugins/google-analyticator/
- Google XML Sitemaps: Every website needs a Sitemap especially if you want to do well with the Search Engines. This is the easiest way to make one http://wordpress.org/plugins/google-sitemap-generator/

WordPress Security Plugins:

- Ultimate Coming Soon Page: Still building your website and don't want anyone to see it yet? This plugin creates a Coming Soon landing page http://wordpress.org/plugins/ultimate-coming-soon-page/
- Better WP Security: This plugin does all kinds of awesome things to keep the bad guys away, I wouldn't even know where to start. Just install it and thank me later http://wordpress.org/plugins/better-wp-security/
- Sucuri: Want to check your website for malware, spam, and other security issues? Sucuri is the best if you ever have security problems with your website http://wordpress.org/plugins/sucuri-scanner/
- Backup Buddy*: This plugin is a lifesaver to have a on hand. Just in case anything happens to your website, it's important to have a backup on hand http://ithemes.com/purchase/backupbuddy/
- BuddyPress: Want to make your website into a social network? Need users to sign up, create porfiles, and post messages? http://wordpress.org/plugins/buddypress/

WordPress Functionality Plugins:

- 1GravityForms*: If you have a contact form on your website, this is hands down the best plugin- it even integrates with third-parties like PayPal, MailChimp, and Aweber http://www.gravityforms.com/
- Contact Form 7: If you really can't afford Gravity Forms, this is a good alternative even though it doesn't have the same features

and ability to integrate with other services like GravityForms does http://wordpress.org/plugins/contact-form-7/

- WordPress SEO by Yoast: If you can only install one plugin, you need to install this one. Getting traffic to your website is a priority and this plugin helps you to do it http://wordpress.org/plugins/wordpress-seo/
- Shareaholic: Making your content easy to share on social networks can be a contributing factor to more traffic on your website. This plugin ads social media icons to the end of your blog posts making them easy to share http://wordpress.org/plugins/shareaholic/
- Events Manager: Great for if you need to add events or a calendar/list of events to your website http://wordpress.org/plugins/events-manager/

WordPress Widgets:

- Facebook Members: Allows visitors to like your Facebook page directly from your website. http://wordpress.org/plugins/facebook-members/
- Rotating Tweets: Want to showcase tweets from your Twitter account? http://wordpress.org/plugins/rotatingtweets/
- Easy Social Icons: Add social media icons to your website and show your website visitors where else they can find you online http://wordpress.org/plugins/easy-social-icons/
- Google Maps Widget: Display a Google map in your sidebar or footer with this great widget http://wordpress.org/plugins/google-maps-widget/
- Testimonials Widget: Showcase reviews, testimonials, or quotes on your website http://wordpress.org/plugins/testimonials-widget/
- Popular Widget: Display the most commented or most viewed posts in a tabbed widget, filter the post by date range or by category http://wordpress.org/plugins/popular-widget/

*Premium Paid Plugin

Summary

Plugins & Widgets are small programs that allow you to customize your WordPress website. Use to increase functionality to your website. You can access WordPress Plugins through the WordPress Plugin Directory and install them through your WordPress Dashboard.

About the Author

Tish Briseno is a WordPress website designer & trainer for small businesses who don't speak geek. She's the author of 8 Steps to a Successful Website.

Email info@tishbriseno.com
Twitter @iSocialTish

Building Advanced Search Queries in WordPress

Woo Themes

WordPress started out as a blogging platform, but recent developments have allowed developers to extend its features beyond that to be a fully fledged Content Management System. This article aims to show how to build advanced search functionality and extend WordPress by introducing a new content type to create a basic property listing plugin.

What you will learn...

- Main outcome: Implementing advanced search queries using meta queries and taxonomy queries.
- Secondary outcomes: Implementing a new custom post type, custom taxonomies, custom meta fields through a custom developed plugin.

What you should know...

- Basic knowledge of setting up a WordPress self hosted site.
- Intermediate PHP knowledge including that of Object Oriented (OO) Programming (simply put, the use of classes)
- Basic knowledge of WordPress development.

Setting the Scene

Since version 3.0 of WordPress, developers have had access to create new types of content using built in functions in the core of WordPress to register a new post type, new taxonomies, and add meta fields to the new post types. Developers have typically used these functions to expand WordPress' capabilities to be "more than a blog" and be more of a fully fledged Content Management System (CMS).

Theme companies were some of the initial adopters of these functions, using them to build themes that allowed functionality like being able to create a directory listing site, events, and real estate listings. What you will be able to implement through this article is a basic plugin that allows you to list properties for rent and perform advanced searches on these properties.

Initial Setup

What you need to get started with this build is

- An existing WordPress self hosted site, hosted either on your local development machine or on your live domain,
- Access to edit the files on your server using your favorite text editor (ours is Sublime Text 2).

We'll be using the Twenty Twelve theme that comes with WordPress, so make sure it is installed and activated. All the other default WordPress settings can be left unchanged.

We're going to be calling our plugin 'Seriously Simple Real Estate' - this will by no means be a complete plugin, but it will contain

everything you need to know to flesh it out into something more practical.

Creating the Plugin Structure

After planning, the first thing to do when creating a WordPress plugin is to set up the file structure. To do this you would need to create a folder inside your WordPress plugins folder as follows:

/wp-content/plugins/seriously-simple-real-estate/

Inside that folder you need to create a file:

/wp-content/plugins/seriously-simple-real-estate/seriously-simple-real-estate.php

In order for WordPress to recognise your plugin, you need to have the following code at the top of the above file:

Listing 1: Identifying your plugin

```
<<LISTING 1>>
/*
 * Plugin Name: Seriously Simple Real Estate
 * Version: 1.0
 * Plugin URI: http://www.woothemes.com/
 * Description: A demo WordPress plugin for Software Developer's
Journal
 * Author: Hugh Lashbrooke & Jeffrey Pearce
 * Author URI: http://www.woothemes.com/
 * Requires at least: 3.0
 * Tested up to: 3.6
 *
 * @package WordPress
 * @author Hugh Lashbrooke & Jeffrey Pearce
 * @since 1.0.0
 */
<</LISTING 1>>
```

Once you have added that code, you are now ready to activate your new plugin! It doesn't add any functionality yet, but it meets WordPress requirements for activation.

You will be able to add the code we give you here to this same file and your plugin will work perfectly. At the end of this article we will supply you with the full code for a plugin built using OO methods, so the ultimate structure will a bit different to what you see in this article, but the outcome will be exactly the same.

Creating a New Post Type

To create a new post type it's as easy as using 1 function -> register_post_type()

This function takes a number of arguments, but for our purposes the most important ones to note are:

- 'public' -> this should be set to true so that visitors to your site can view your individual custom posts.
- 'taxonomies' -> this links your post type with the custom taxonomies covered in the next section.
- 'has_archive' -> this allows the use of a custom archive page for your post types that makes use of the default WordPress archive template.

If you want to further customize your post type, consult the WordPress Codex
http://codex.wordpress.org/Function_Reference/register_post_type

Listing 2: Register the 'property' custom post

```
<<LISTING 2>>
type
function register_post_type() {

      $labels = array(
            'name' => _x( 'Properties', 'post type general name' , 'ss_
realestate' ),
            'singular_name' => _x( 'Property', 'post type singular name'
, 'ss_realestate' ),
            'add_new' => __( 'Add New', 'ss_realestate' ),
            'add_new_item' => sprintf( __( 'Add New %s', 'ss_realestate'
), __( 'Property' , 'ss_realestate' ) ),
            'edit_item' => sprintf( __( 'Edit %s', 'ss_realestate' ),
__( 'Property' , 'ss_realestate' ) ),
            'new_item' => sprintf( __( 'New %s', 'ss_realestate' ), __(
'Property' , 'ss_realestate' ) ),
            'all_items' => sprintf( __( 'All %s', 'ss_realestate' ), __(
'Properties' , 'ss_realestate' ) ),
            'view_item' => sprintf( __( 'View %s', 'ss_realestate' ),
__( 'Property' , 'ss_realestate' ) ),
            'search_items' => sprintf( __( 'Search %s', 'ss_realestate'
), __( 'Properties' , 'ss_realestate' ) ),
            'not_found' => sprintf( __( 'No %s Found', 'ss_realestate'
), __( 'Properties' , 'ss_realestate' ) ),
            'not_found_in_trash' => sprintf( __( 'No %s Found In Trash'
, 'ss_realestate' ), __( 'Properties' , 'ss_realestate' ) ),
            'parent_item_colon' => '',
            'menu_name' => _x( 'Properties', 'post type menu name', 'ss_
realestate' )
      );

      $args = array(
            'labels' => $labels,
            'public' => true,
            'publicly_queryable' => true,
            'exclude_from_search' => true,
            'show_ui' => true,
            'show_in_menu' => true,
            'show_in_nav_menus' => true,
            'query_var' => false,
            'rewrite' => true,
            'capability_type' => 'post',
            'has_archive' => true,
            'hierarchical' => true,
            'supports' => array( 'title', 'editor', 'excerpt' ),
            'taxonomies' => array( 'property_area', 'property_type' ),
            'menu_position' => 5,
      );

      register_post_type( 'property', $args );
}
<</LISTING 2>>
```

If you add this to your plugin's file and reload the backend of your site you will now see a new menu item called "Properties".

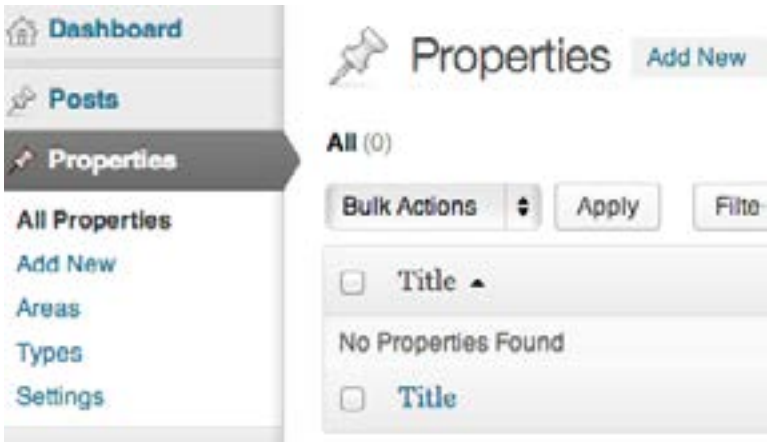


Figure 1. Properties Custom Post Type menu in the WordPress Admin

Note the use of _x() & __() in the above code - those are WordPress' built-in i18n functions that allow people to translate your plugin into any language they like. We are using our own text domain called 'ss_realestate' that can be used by anyone to target the text strings in the plugin.

Note: Sometimes you may need to revisit the WordPress Permalinks Settings page in order to see these changes. If at any point during this tutorial you don't see the desired result, revisit this page. You can find this page in the WordPress admin area under the "Settings -> Permalinks" menu.

Figure 2. Permalinks menu item in the WordPress Admin

Organizing your Post Type

By default, WordPress posts are organized using categories and tags, but what most people don't know is that they are both examples of Custom Taxonomies associated to the default Post Type.

We don't want to use categories and tags for our post type though, we want to be able to sort our properties by Area and Type. To do this is really easy, if you recall when setting up our post type we added a 'taxonomies' parameter with 'property_area' and 'property_type' in an array - this is part of how to link a taxonomy to a post type. The other part is to register the taxonomies, this is easily done using the register_taxonomy() function.

The main function arguments here that you need to take note of is the 2nd argument which links your taxonomy to the post type, and the 'hierarchical' argument that should be set to true.

Listing 3: Register two custom taxonomies for the new post type

```
<<LISTING 3>>
function register_taxonomies() {
     register_new_taxonomy( 'property_area', __( 'Areas', 'ss_
realestate' ), __( 'Area', 'ss_realestate' ) );
     register_new_taxonomy( 'property_type', __( 'Types', 'ss_
realestate' ), __( 'Type', 'ss_realestate' ) );
}

function register_new_taxonomy( $taxonomy, $general_name, $singular_
name, $hierarchical = true ) {

  $labels = array(
    'name' => sprintf( _x( '%s', 'taxonomy general name', 'ss_
realestate' ), $general_name ),
    'singular_name' => sprintf( _x( '%s', 'taxonomy singular name',
'ss_realestate' ), $singular_name ),
    'search_items' => sprintf( __( 'Search %s', 'ss_realestate' ),
$general_name ),
    'all_items' => sprintf( __( 'All %s', 'ss_realestate' ), $general_
name ),
    'parent_item' => sprintf( __( 'Parent %s', 'ss_realestate' ),
$singular_name ),
    'parent_item_colon' => sprintf( __( 'Parent %s:', 'ss_realestate'
), $singular_name ),
    'edit_item' => sprintf( __( 'Edit %s', 'ss_realestate' ),
$singular_name ),
    'update_item' => sprintf( __( 'Update %s', 'ss_realestate' ),
$singular_name ),
    'add_new_item' => sprintf( __( 'Add New %s', 'ss_realestate' ),
$singular_name ),
    'new_item_name' => sprintf( __( 'New %s Name', 'ss_realestate' ),
$singular_name ),
    'menu_name' => sprintf( _x( '%s', 'taxonomy menu name', 'ss_
realestate' ), $general_name ),
  );

  $args = array(
    'public' => true,
    'hierarchical' => $hierarchical,
    'rewrite' => true,
    'labels' => $labels
  );

  register_taxonomy( $taxonomy,'property', $args );
}
<</LISTING 3>>
```

If you want to further customize your taxonomy, consult the WordPress Codex *http://codex.wordpress.org/Function_Reference/register_taxonomy*

A Quick Pause

At this stage you can refresh your WordPress backend and will see that the Properties menu item will now have some new items below it, "Areas" and "Types". These items allow you to add new Areas and Types. You can also take a look at the "Add New" area for Properties now, where you will see a space to add new Areas and Types.

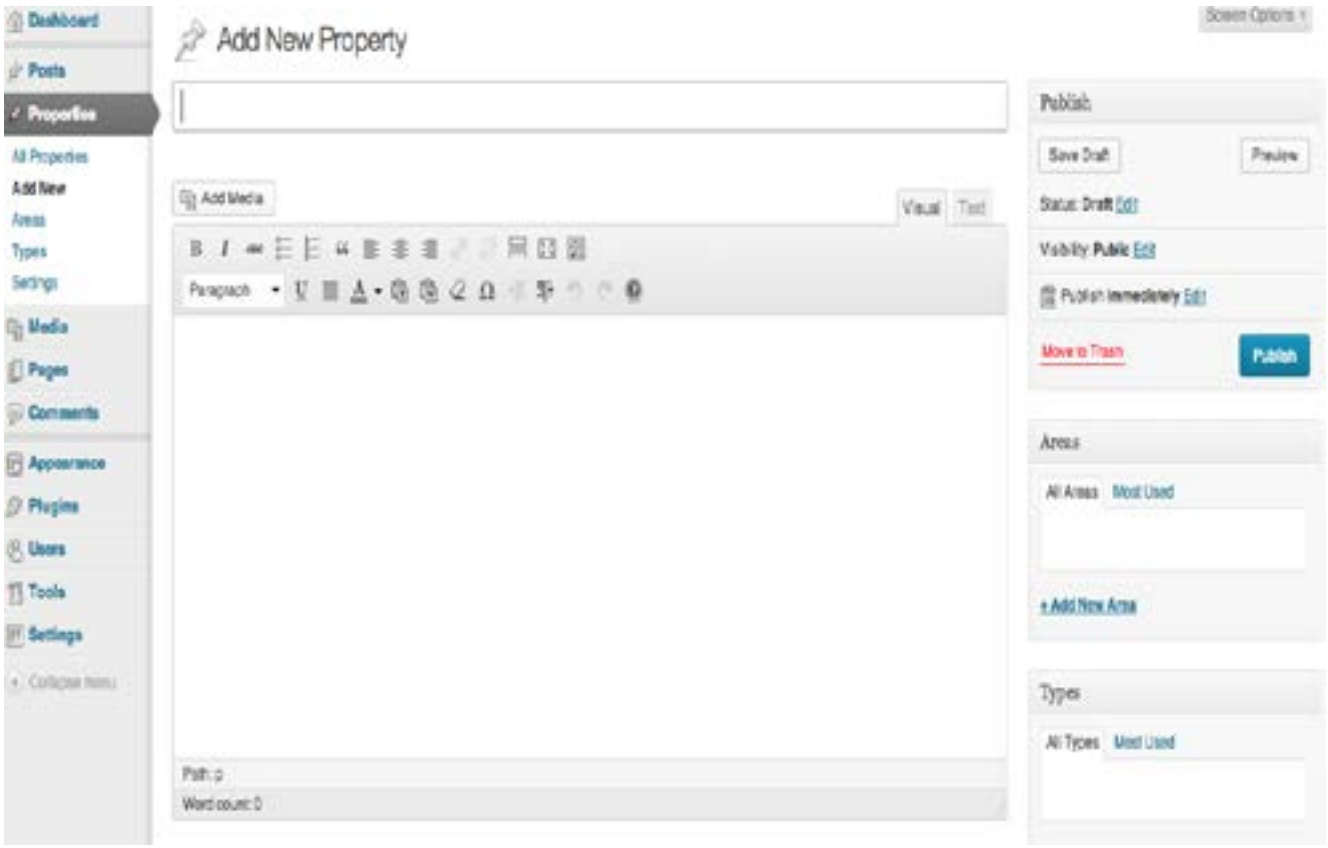


Figure 3. Add New Property screen in the WordPress Admin

Adding Custom Data to the Post Type

Other than Area and Type it would be awesome to add a price to our properties. To do this we want to add a custom meta field to capture this value. WordPress has some built in functions to help us with that.

Listing 4: Create a new meta box with a 'Price' field for each property

```
<<LISTING 4>>
add_action( 'admin_menu', 'ssre_meta_box_setup', 20 );
function ssre_meta_box_setup() {
      add_meta_box( 'post-data', __( 'Property Details' , 'ss_
realestate' ), 'ssre_meta_box_content', 'property, 'normal', 'high' );
}

function ssre_meta_box_content() {
      global $post_id;
      $fields = get_post_custom( $post_id );
      $field_data = ssre_get_custom_fields();

      $html = '';

      $html .= '<input type="hidden" name="property_nonce"
id="property_nonce" value="' . wp_create_nonce( plugin_basename(
dirname( __FILE__ ) ) ) . '" />';

      if ( 0 < count( $field_data ) ) {
            $html .= '<table class="form-table">' . "\n";
```

```
            $html .= '<tbody>' . "\n";
            foreach ( $field_data as $k => $v ) {
                  $data = $v['default'];

                  if ( isset( $fields[$k] ) && isset( $fields[$k][0] ) ) {
                        $data = $fields[$k][0];
                  }

                  $html .= '<tr valign="top"><th scope="row"><label
for="' . esc_attr( $k ) . '">' . $v['name'] . '</label></th><td><input
name="' . esc_attr( $k ) . '" type="' . esc_attr( $v['type'] ) . '"
id="' . esc_attr( $k ) . '" class="regular-text" value="' . esc_attr(
$data ) . '" />' . "\n";
                  $html .= '<p class="description">' . $v['description']
. '</p>' . "\n";
                  $html .= '</td><tr/>' . "\n";

            }

            $html .= '</tbody>' . "\n";
            $html .= '</table>' . "\n";
      }

      echo $html;
}

function ssre_get_custom_fields() {
      $fields = array();

      $fields['property_price'] = array(
        'name' => __( 'Price:' , 'ss_realestate' ),
        'description' => __( 'Property price' , 'ss_realestate' ),
        'type' => 'number',
        'default' => '',
        'section' => 'plugin-data'
      );

      return $fields;
}

add_action( 'save_post', 'ssre_meta_box_save' );
function ssre_meta_box_save( $post_id ) {
      global $post, $messages;

      // Verify nonce
      if ( ( get_post_type() != 'property' ) || ! wp_verify_nonce( $_
POST[ 'property_nonce'], plugin_basename( dirname( __FILE__ ) ) ) ) {
            return $post_id;
      }

      // Verify user permissions
      if ( ! current_user_can( 'edit_post', $post_id ) ) {
            return $post_id;
      }

      // Handle custom fields
      $field_data = $this->get_custom_fields();
      $fields = array_keys( $field_data );
```

```
      foreach ( $fields as $f ) {

            if( isset( $_POST[$f] ) ) {
                  ${$f} = strip_tags( trim( $_POST[$f] ) );
            }

            // Escape the URLs.
            if ( 'url' == $field_data[$f]['type'] ) {
                  ${$f} = esc_url( ${$f} );
            }

            if ( ${$f} == '' ) {
                  delete_post_meta( $post_id , $f , get_post_meta(
$post_id , $f , true ) );
            } else {
                  update_post_meta( $post_id , $f , ${$f} );
            }
      }

}
<</LISTING 4>>
```

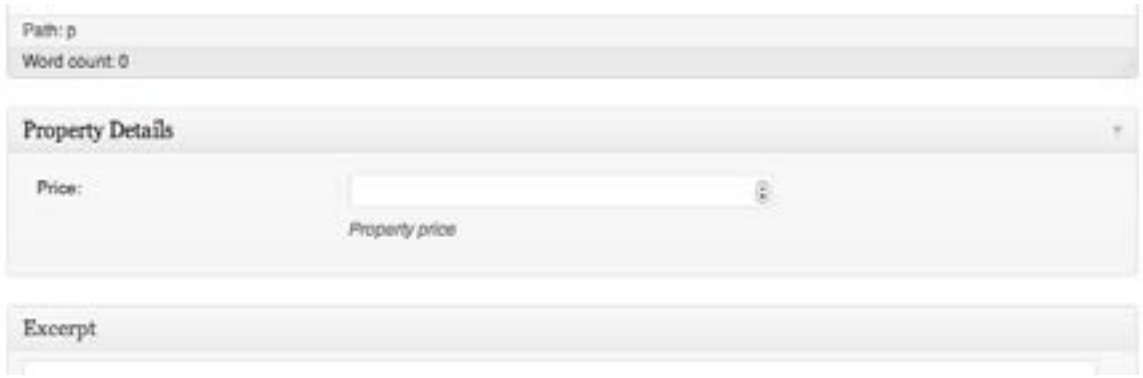


Figure 4. Price Meta Field in the Add New Property screen in the WordPress Admin

Your Content is Ready!

So at this stage your new content type is ready for data capture! Let's add some properties - you can either add your own or use the demo content that you can download (and import using WordPress' built-in importer).

- Dummy data XML import file: *https://raw.github.com/woothemes/software-developers-journal/master/seriously-simple-real-estate/dummy-data.xml*

Advanced Search

Now that you've extended WordPress features to have a new type of content, we can move onto the main goal of this article, to teach you how to implement an advanced search because the way the WordPress default search works is not suited to a property listings site. Your visitors will want to be able to search by Area and Type, as well as by Price Range.

In order to do this we need to implement Meta Query and Taxonomy Query filters.

http://codex.wordpress.org/Plugin_API#Filters - A brief overview of what a WordPress filter is and what it can be used for.

We want to use the existing WordPress search function as well as the search results template, and create a shortcode to output our advanced search form. To do this we needto first create the form adding field containing the taxonomies and the price ranges:

Listing 5: Creating a search form that can be called us the 'property_search' shortcode

```
<<LISTING 5>>
add_shortcode( 'property_search', 'ssre_property_search' );
function ssre_property_search() {

	$args = array(
		'hide_empty' => false
	);

	$areas = get_terms( 'property_area', $args );
	$types = get_terms( 'property_type', $args );

	$area_options = '';
	if( count( $areas ) ) {
		$area_options = '<option value="">-- ' . __( 'All areas',
'ss_realestate' ) . ' --</option>';
		foreach( $areas as $area ) {
			$area_options .= '<option value="' . esc_attr( $area-
>term_id ) . '">' . $area->name . '</option>';
		}
	}

	$type_options = '';
	if( count( $types ) ) {
		$type_options = '<option value="">-- ' . __( 'All types',
'ss_realestate' ) . ' --</option>';
		foreach( $types as $type ) {
			$type_options .= '<option value="' . esc_attr( $type-
>term_id ) . '">' . $type->name . '</option>';
		}
	}

	$prices = array();
		for( $i = 250000; $i <= 3500000; $i += 250000 ) {
			$prices[] = $i;
		}


	$price_options = '';
	foreach( $prices as $price ) {
		$price_options .= '<option value="' . esc_attr( absint(
$price ) ) . '">' . number_format( $price ) . '</option>';
	}

	$title = __( 'Property Search', 'ss_realestate' );

	$html = '<div class="property_search">
				<h3>' . $title . '</h3>
				<form name="property_search" action="' . get_
site_url() . '" method="get">
```

```
                                <input type="hidden" name="search"
value="property" />
                                <input type="text" name="s" value=""
placeholder="Text search" />
                                <br/>
                                <select name="area">' . $area_options . '</
select>
                                <select name="tax_relation">
                                      <option value="AND">' . __( 'AND',
'ss_realestate' ) . '</option>
                                      <option value="OR">' . __( 'OR', 'ss_
realestate' ) . '</option>
                                </select>
                                <select name="type">' . $type_options . '</
select>
                                <br/>
                                ' . __( 'Price:', 'ss_realestate' ) . '
                                <select name="price_min">
                                      <option value="0">0</option>
                                      ' . $price_options . '
                                </select>
                                ' . __( 'to', 'ss_realestate' ) . '
                                <select name="price_max">' . $price_options
. '</select>
                                <br/>
                                <input type="submit" value="' . __(
'Search', 'ss_realestate' ) . '" />
                          </form>
                    </div>';

      return $html;
}
<<</LISTING 5>>
```

To see this search form on your site, you can add it to a page by using the following shortcode in the content area:

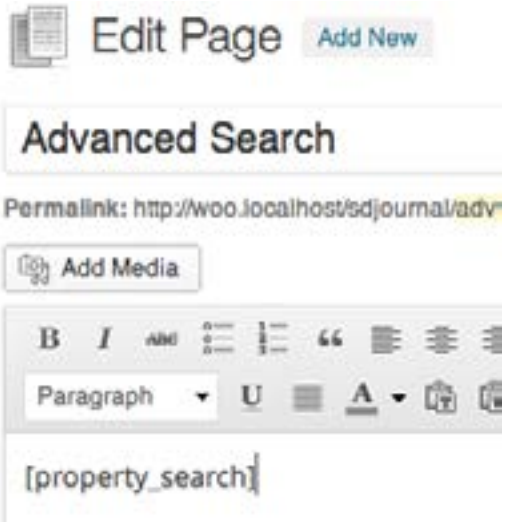


Figure 5. 'property_search' Shortcode in the WordPress Admin

This shortcode will output an advanced search form on the frontend of your website.

Figure 6. Property Search Form in a page in Twenty Twelve

Lastly we need to add a filter to handle the search form fields in the search results output:

Listing 6: Filtering the WordPress posts to show our property search results

```
<<LISTING 6>>
add_action( 'pre_get_posts', 'property_search_filter' );
function property_search_filter( $query ) {

     if( isset( $_GET['search'] ) && $_GET['search'] == 'property' ) {

          // Only apply the searc filter on the main page query
          if ( ! $query->is_main_query() )
               return;

          $query_args = array();

          // Basic arguments
          $query_args['post_type'] = 'property';
          $query_args['post_status'] = 'publish';

          // Area taxonomy query
          if( isset( $_GET['area'] ) && absint( $_GET['area'] ) > 0 )
{
               $query_args['tax_query'][] = array(
                    'taxonomy' => 'property_area',
                    'terms' => absint( $_GET['area'] ),
                    'field' => 'id',
                    'operator' => 'IN'
               );
          }

          // Type taxonomy query
          if( isset( $_GET['type'] ) && 0 < absint( $_GET['type'] ) )
{
               $query_args['tax_query'][] = array(
                    'taxonomy' => 'property_type',
                    'terms' => absint( $_GET['type'] ),
                    'field' => 'id',
                    'operator' => 'IN'
               );
          }
```

```
            // Taxonomy relation
            if( isset( $_GET['tax_relation'] ) && in_array( $_GET['tax_
relation'], array( 'AND', 'OR' ) ) ) {
            $query_args['tax_query']['relation'] = $_GET['tax_
relation'];
        }

        // Price meta query
        if( isset( $_GET['price_min'] ) && isset( $_GET['price_max'] )
) {
            $query_args['meta_query'][] = array(
                'key' => 'property_price',
                'value' => array( absint( $_GET['price_min'] ), absint(
$_GET['price_max'] ) ),
                'compare' => 'BETWEEN',
                'type' => 'NUMERIC'
            );
        }

            // Pagination
            $paged = ( get_query_var( 'paged' ) ) ? get_query_var(
'paged' ) : 1;
        $query_args['paged'] = $paged;

        // Check text search string
        if( get_query_var('s') ) {
            $string = esc_attr( get_query_var('s') );
            if( $string && strlen( $string ) > 0 ) {
                $query_args['s'] = $string;
            } else {
                $query_args['s'] = '';
            }
        }

        // Set query variables
        foreach ( $query_args as $key => $value ) {
                $query->set( $key, $value );
        }
    }

    return $query;
}
<</LISTING 6>>
```

What this filter will do is handle the standard search and apply filters to the results based on the post type, taxonomies, and meta field values.

There are some key elements to this filter:

- Post Type - you will see this is specified in the line $query_args['post_type'] = 'property'; and what this does is make sure that the search results only draw items from the post type that we created in this article.
- Area and Types Taxonomy Queries - what these 2 arguments do is similar to a SQL "WHERE" clause in a "JOIN" type query. This further specifies which results to return. And just like with SQL, you can specify various logical operators, in this case "AND" or "OR".

- Price Meta Query - this further specifies what results to return, returning results between the 2 values used from the search form price fields.
- The text keywords - this uses the default WordPress 's' variable to search listings titles and content for matches.

You can read more about WP_Query and it's various parameters here: *http://codex.wordpress.org/Class_Reference/WP_Query*

Additional Ideas and Next Steps

Congratulations! You have built a basic property listing plugin that allows you to list and search for your properties. The next step would be to expand the plugin to be fully featured by:

- Expanding the settings page to allow for more customization of the plugins features. You will find the Settings page code in the finished plugin
- Make the drop downs on the frontend search form more user friendly by implementing a JavaScript library such as Chosen. This will become very important when your list of Areas and Types grow larger.
- *http://harvesthq.github.io/chosen/* - Chosen JavaScript library
- Add more property meta data to the post type - follow the example of the Price field and add more data such as "on show dates" and "address".
- Templating - many plugins these days provide a basic templating engine for theme companies to use when supporting 3rd party plugins. This will also make your plugin look far superior right out of the box.

When you are happy with your plugin, consider sharing it by listing it on *WordPress.org* or on GitHub. And if you are thinking about making the jump to full time WordPress development, why not sell your extended version it as a premium plugin!

- You can download the completed plugin here: *https://github.com/woothemes/software-developers-journal*

About the Author

Woo Themes- What started in 2008 as 3 *WordPress* enthusiasts who met online, from 3 different countries, is now an *international team* of designers, developers and support ninjas catering for a passionate community of hundreds of thousands of users.

From our humble beginnings selling a handful of commercial WordPress themes we now offer a huge catalog of feature rich *themes*, and a suite of *plugins* that extend your WordPress experience. Proudly bootstrapped and built around our team's diverse lifestyles we have big ambitions to become the ultimate WordPress platform provider.

pixel jar

CREATING PROFESSIONAL WORDPRESS SOLUTIONS SINCE 2007

- Custom Plugins
- Responsive Theme Development
- BuddyPress Social Networks
- WordPress Multi-Site

Brandon and Jeff have helped us optimize our site, bring in new visitors, simplify our ad management, and get our site organized. Their knowledge of WordPress is top notch. We love Pixel Jar!

— Amy Squires, WeddingChicks.com

pixeljar.net

pixel_jar

pixeljar

Software Developer's JOURNAL
new ideas & solutions for professional programmers

Customize your WordPress Theme the Right Way with Child Themes

Ruth Maude

As you read this, WordPress dominates the Web. In fact, 18.9% of the Web is powered by WordPress[1]. As the popularity of WordPress grows, so does the demand for Web developers who can build custom themes or properly customize an existing Theme. The correct way to modify an existing theme is with a child theme.

What you will learn…

In this article you'll learn how to modify a WordPress theme by creating a "child theme". We'll look at what makes a theme a "child theme", why and how you would use them.

What you should know…

This article assumes that you are familiar with using an FTP client (such as Filezilla) and a file editing program (e.g. Komodo edit). You should be comfortable creating folders, downloading files from the host server to your computer, editing files and uploading files to server. You should also be familiar with CSS (Cascading Style Sheets). Knowledge of PHP is an asset.

You've searched hour upon hour for the perfect WordPress Theme. You find one that is very close to what you want in a website – if you could only change a few things. But here is the thing – if you make changes in the theme files, you won't be able to update the theme without losing all your customizations. Updating a theme may be necessary for security, added functionality or for compatibility with new versions of WordPress.

The solution to this problem is to create a child theme. If theme modifications are made in a child theme, the parent theme can be updated without overwriting your changes.

WHAT? Child Themes Defined

The WordPress Codex explains the concept of child themes as a …

Theme that inherits the functionality of another theme, called the parent theme, and allows you to modify, or add to, the functionality of that parent theme.

A child theme is the safest and easiest way to modify an existing theme, whether you want to make a few tiny changes or extensive changes. Instead of modifying the theme files directly, you can create a child theme.

A child theme inherits all of the templates and functionality from its parent theme, but allows you to make changes to the parent theme because code in the child theme overwrites code in the parent theme.[2]

WHY? Four Benefits of Child Theming

Choose a child theme if you want to:

- make a few customizations to an existing theme; it is safer to do it by creating a child theme than by modifying the parent theme directly.
- make a lot of customizations to the theme look and layout but use the underlying theme features without starting from scratch. Starting with a parent theme will speed up development time.
- ensure that your theme continues to work going forward with new versions of WordPress. You can build a child theme and let the developer of the parent theme deal with updates, adding new WordPress functionality and security releases.
- brand a theme. You can rename the theme for the client's project and list your name and company as the theme author. You can add a *screenshot.png* file to the child theme folder to display on the wp-admin themes page.

Child Theming Rules:

The parent theme must be installed on your site – You'll break your theme if you delete the parent theme. If you install a child theme by FTP you will also need to install the parent theme. If you upload a child theme from a zip file, WordPress will search the repository and install the parent theme if available.

Do not edit parent theme files directly – If you edit the parent theme then when you update the theme you will lose all of your changes. Besides, this totally defeats the purpose of creating a child theme in the first place!

Only put modified files in the child theme folder – The child theme folder only needs to contain the *style.css* file and any other modified files. You want to preserve your modifications only. Child theme files will take precedence over the files in the parent folder. There's no need to duplicate files from the parent folder that haven't been modified.

HOW? Create a Child Theme in 5 Easy Steps

For this example I'm going to use the free "*Responsive Theme*" from the WordPress repository as the parent Theme.

A child theme starts with the *style.css* file. This is the only file that is required to create a child theme.

1. Open the FTP client and create a new empty folder on the host server in "*.../wp-content/themes/child-theme-name*". Do not put the child theme folder inside of the parent folder. Use hyphens for spaces in the folder name.
2. Open a file editor and create a new *style.css* file using this sample code:

```
/*
  Theme Name:  My Theme
  Theme URI:    http://clientsite.com/
  Description:    Child theme for Responsive
Theme
  Author:         Ruth Maude
  Author URI:     http://dandelionwebdesign.com/
  Template:       responsive
```

```
  Version:       0.1
*/

@import url("../responsive/style.css");
```

Edit this as follows:

- start by changing "My Theme" to identify the theme, make sure it is a different name from the "parent" theme,
- add your client's website address,
- edit the theme description to describe your new theme and note that it is a child-theme,
- change the author name to give yourself credit,
- add your website address to create a link on the WordPress manage themes screen,
- the template line indicates the directory name of the parent theme (case-sensitive),
- change the version number to the child-theme's version,
- @import url("../parent/*style.css*"); calls the original *style.css* from the parent theme. Unmodified classes will loaded normally,
- include your custom css styles below the @import. You can create new entries in this file, or override the existing ones from the old *style.css*.

1. Now, go back to FTP and upload the *style.css* file to the child theme folder.
2. In the WordPress admin area go to Appearance -> Themes and activate your new theme.
3. You can change more than theme styles. A child theme can overwrite any file in the parent theme and add additional custom files. Here's how …

Child Theming Beyond Style

Include a file of the same name in the child theme directory, to supersede the equivalent file in the parent theme directory. For instance, you can copy the *footer.php* from the parent theme to the child theme, modify your copyright information, and then upload the modified *footer.php* file back to the child theme folder.

The *functions.php* of a child theme is one exception to the rule. It does not override its counterpart from the parent. A child theme's *functions.php* loads first. To add new functions, such as a menu or widget area, create a blank *functions.php* file with opening and closing PHP tags, between them add your bits of PHP.

To replace an original function from the parent's *functions.php*, declare the function conditionally, for example:

```
if ( ! function_exists( 'func_name' ) ) :
function func_name() {
//do something...
}
endif;
```

You can also add completely new files that aren't part of the original parent theme, for example a custom page template. The easiest way is to start with an existing page template. Give it a new

file name and edit the head of the page template to reflect the new template name, then edit the PHP code for your purposes.

Add a *screenshot.png* file to the child theme folder if you wish to display an image in the Available Themes list on the Manage Themes screen.

Test drive your theme thoroughly to ensure that it works well in different browsers an on different devices. If your parent theme is responsive, make sure that your modifications continue to work responsively.

Child themes are now eligible for acceptance into the WordPress repository. If your child theme is significantly different from the parent theme you can submit it for consideration.

Congratulations… you are now a WordPress Theme Developer!

Summary:

A child theme is the safest and easiest way to modify an existing WordPress Theme. In this article you learned how to create a child theme with a *style.css* file and how a child theme can modify, or extend the functionality of a parent theme.

References:

1. *"18.9% of the Web is powered by WordPress" http://wordpress.tv/2013/07/27/matt-mullenweg-state-of-the-word-2013/ - Matt Mullenweg State of the Word - July 27, 2013*
2. *http://codex.wordpress.org/Child_Themes*

About the Author

Ruth Maude, of *Dandelion Web Design Inc.*, has been designing and building websites since the late 1990's. Ruth works primarily in WordPress. She loves WordPress because it is open source, user-friendly and the best platform for SEO. Ruth teaches *WordPress Workshops in Toronto* for *LearnWP.ca* and *speaks at WordCamps and* conferences. You can find her on twitter @dandelionweb.

Software Developer's JOURNAL
new ideas & solutions for professional programmers

How To Speed Up Your WordPress Website

Jeffrey Zinn

Since the explosion of broadband access and the availability of rich media, users are expecting more and more, faster and faster. With WordPress at the helm of nearly 20% of websites, here are few tips on how to optimize and enhance your WordPress site to increase its speed for the ever eager user.

What you will learn...

- Simple tips on how to speed up a WordPress website
- What to look for in a web host to better your chances for speed
- What caching is and how to make use of it
- How a Content Delivery Network (CDN) can increase the speed and performance of a website

What you should know…

- Be familiar with the WordPress dashboard
- How to download and install plugins

Speeding up any website is important. Users are ever more impatient these days and a delayed page load can result in a dreaded bounce. Search engines also measure the load time of a website and factor those results into their ranking algorithms. So both from a usability standpoint and an search engine optimization (SEO) point of view it is important to speed up your site as much as possible. Nearly one in five websites active today are utilizing the WordPress platform, and using any content management system (CMS) requires server-side processing that will inherently impact site speed. So what can a WordPress site owner do to step on the gas pedal?

Hosting

Begin with the most basic decision when starting a website - hosting. When first starting out, it's easy to fall into the $2.95 per month shared hosting trap. As my WordPress meetup group is fond of repeating, not all hosts are created equal, and cheaper is not always better. A hosting provider is responsible for the hardware and software that delivers a website to its end user. Taking some time to research a host can make a big difference in a site's overall load time.

Though often not considered to account for much, hardware can be just as integral as software when rendering a website. Find out if the host is using modern hardware. Also take note of where there servers are located. If you know your primary target market is in North America, it doesn't make sense to host your servers in China. Look for hardware located in properly maintained data centers with proper environment and technological setups. If possible, use a host

that uses solid state drives (SSDs) instead of spindle drives as SSDs are significantly quicker. Will the site live on a shared server, a virtual private network (VPN), or have it's own dedicated server? Shared servers tend to run slowest as the hardware's resources are shared between all sites running on that server.

Software is obviously important as well, but tends to be more of an indicator of server upkeep rather than speed. Check that the host is running versions of PHP and MySQL that support the latest version of WordPress. If PHP and MySQL are not up to date, this can be telling of what kind of hardware is being used to run a webserver - older software means older hardware. At the time of writing this article, the latest version of WordPress is 3.6 which requires PHP 5.2.4 or greater, and MySQL 5.0 or greater.

If there is no information to be found for regarding hardware and software on a host's website, use the contact information to find out more. (If nothing else, this will provide a sense of what the host's customer support may be like.)

It is also worth considering using a host that is optimized to run WordPress websites such as WP Engine or *Page.ly*. The bigger hosting companies have their hardware and software configured generically to serve up a whole swath of webby goodness that is not necessary in supporting a WordPress website. With WordPress specific hosting, because a host knows the only sites to be served are WordPress sites, the hardware and software will be tuned and optimized to specifically run WordPress. This will generally result in a faster site compared to the same site is hosted on non-WordPress-specific host. Typically hosting fees for these types of hosts will be higher than the bigger brands, but this careful customization and expert support accounts for the higher price tag.

Caching

Think of rendering a website like building a Lego model. Every page is made up of small pieces of data that come together to form a whole page. A cached page can be thought of as a completed Lego model versus a bunch of pieces that need assembling. It's much faster to deliver a completed Lego to a user than to build the model from scratch each time it is requested.

Because WordPress is based on PHP and MySQL, no page actually exists until it is requested by a user typing in a URL. When Page A is requested, PHP and MySQL get the necessary templating and data together, render the page, and then serve it up to the user. Caching takes that final product that is shown to the user and stores it for later retrieval rather than go back to PHP and MySQL to rebuild the page every time Page A is requested by a user. This can be done both at the server level and at the browser level.

Admittedly, setting up caching is not always easy. There are many options and considerations to be made when setting up caching for a website. Caching is typically set up through a WordPress plugin and utilizes whatever the hosting environment has to offer.

With caching plugins, there are several options available, but two plugins are dominant among those choices - WP Super Cache (WPSC) and W3 Total Cache (W3TC). There are differences on how each plugin handles cached data, but ultimately setting up either plugin in its default state will have your site running faster once in operation. WPSC will take the rendered page and save a static copy to the server. Since web servers are faster at serving static pages, this provides a good increase in speed. W3TC provides a slightly different and more comprehensive approach to caching. W3TC can also utilize server side caching through APC or memcached, minify css and javascript, and integrate with a content delivery network. Detailed tutorials on configuring detailed settings for WPSC and W3TC can be found on *wpbeginner.com* and other websites (the configuration options are too detailed to cover in this article). However, do be careful when running a caching plugin when a site is on a shared host as that does not always generate a net gain in site speed.

Alternatively, a hosted caching solution can be used. Generally shared hosts do not provide any caching services. With specialty hosts, caching can be included and configured at the server level without the need for any plugins. In the case of WP Engine and *Page.ly*, caching services are included with hosting even at the most basic hosting levels.

Content Delivery Networks

Even with caching established there can still be a bottleneck of files moving from a web server to an end user. In the case of WordPress, all the templates, images, stylesheets, javascript, and every other file that is needed to produce a web page must be sent from the server to the user. Often times this can be a strain on the server as the delivery of those files is at the mercy of that server's delivery speed for every file. A content delivery network (CDN) is a great way to ease the burden from a server that is delivering web pages.

With a CDN such as MaxCDN or Amazon S3, many of the files that create a web page can be offloaded from the main web server to a separate server specifically designed to deliver those types of files at the fastest possible speeds. When the files are being served from many sources, the ultimate delivery rate is sped up as the user is now receiving files from several servers rather than one. Imagine needing to fill a bucket with water. The bucket will fill up quicker with several spigots pouring water into it rather than just one.

Furthermore those various files are then disseminated to various servers located around the world. When a file is requested by a user, it can be served from the server closest to the end user's physical location. As fast as the Internet often seems, physical distance between the user and the server is still a factor in page load speed. For example, if a website's files all live on a server on the east coast of America, but a user from Japan or Europe or America's west coast request a page, all of those files transmit from the East Coast to their final location. But if a CDN has stored copies of the same files on servers in Europe, Asia, California, and other locations, it will deliver the files from a location much nearer to the end user.

Many CDNs can be integrated with WordPress either by using custom plugins or often times through caching plugins. Both MaxCDN, Amazon S3, and many other CDNs can be set up using both the WP Super Cache and W3 Total Cache plugins mentioned above.

Database

From time to time the WordPress database can use a tune up. Just like a work desk, it can become cluttered and disorganized. Optimizing the database tables will essentially defragment the the data for easier access. Once optimized, a database will run quicker.

Normally this is done by logging in to the database server or phpMyAdmin and running commands to optimize the database tables. But this is WordPress, and just like with caching, there are plugins designed to optimize the database tables. One such plugin is WP Database Optimizer. Once installed and activated, there is a button for the admin to press that will immediately optimize all the WordPress database tables. With this particular plugin, the admin may also set up regular automatic optimizations for every set number of days, which is a time saving feature. This will keep the database optimized over time without the admin having to manually optimize the database on a regular basis.

Plugins

Plugins add functionality to a WordPress site and it is this extensibility that makes WordPress such a viable platform for so many websites. That said, plugins can also zap the speed from a site. It is important to remember that all plugins are created by third party developers. These developers can range from WordPress core developers to hobbyists. Each plugin has a different level of care and skill applied to its creation and maintenance. As such, it is important to analyze the performance of each plugin to see what amount of load it is putting on the site.

When maintaining plugins on a WordPress website, begin by removing any unused plugins. Often admins will deactivate unused plugins, but leave them in the system. And while these plugins are not active, WordPress still keeps track of them and their files, which of course takes processing power.

Minimizing plugin use will also help speed up a WordPress website. With all the versatility that plugins offer it can be easy to load up on plugins. But every plugin that is active on the site is another set of files that need to be loaded and rendered. Take some time to review existing plugins and determine if their use is truly necessary to the function of the website. If not, get rid of them.

Analyze a site's existing plugins. GoDaddy developed a useful plugin called P3 Profiler that allows administrators to analyze a site's plugin load time. Load hogs can be identified easily and either removed or replaced if necessary. Do be careful with this plugin's analysis, however. A large load time does not always equate to a resource intensive plugin. For example, the Jetpack plugin can

appear as the longest loading plugin. But since Jetpack is more of a suite of plugins rather that just a single-purpose plugin, the load time may be justified. Analyze each plugin individually against its own merit and assess whether its function justifies its loading costs.

Analysis

Lastly, it is important to quantify page speed. There are a number of tools on the web that will allow page speed to be measured such as Google Page Speed, GTmetrix, and YSlow. It's not necessary to measure every page, but it's a good idea to take a cross section of samples. The homepage is typically a site's heaviest load and should definitely be measured. In addition, test a typical static page, a typical blog post, a product page if there is ecommerce on the site, or any page that represents a typical layout on a site.

Start by testing the given page to set a baseline measurement to compare to future measurements. Apply changes slowly, preferably one at a time. Test again after each change and compare measurements to the baseline. This will provide a measurement of how significant each change has improved page's load speed.

And what should be the ultimate goal for a page load time? Web developers are always hoping for a sub-second load time. But with all the external and sometimes uncontrollable factors that factor into a page load, such as hardware, software, infrastructure, Internet speed, the user's device, and so on, a sub-second load can be unrealistic. So, while there is no real, official gold standard for load time, obviously less time is better.

It is important to consider the page being loaded. For example, loading *google.com* versus *youtube.com* is not a fair comparison. One is a very simple page and the other is much more media intensive, and both serve a specific and different purpose. Clearly, it would not make sense for *youtube.com* to simplify their site down to the point of *google.com* just for the sake of load speed. The ultimate goal is to have a web page load as quickly as possible, while considering the purpose of the page.

Summary

The basic building blocks of site speed are what files need to be delivered to the user and how files are delivered. Anything an administrator can do to ease that delivery will speed up the site. Every site is unique and not all the techniques described above can necessarily be applied to each site. But apply those that can be be done, because every little bit counts to a grander whole.

With all suggestions made here, be sure to perform due diligence. The Internet is an ever changing organism. What is true today, may not be true tomorrow. Stay vigilant and current of changes in software, hardware, algorithms, and so on.

On the Web

- http://wordpress.org/plugins/wp-super-cache/ - WP Super Cache Plugin
- http://wordpress.org/plugins/w3-total-cache/ - W3 Total Cache Plugin
- http://wordpress.org/plugins/p3-profiler/ - P3 Profiler Plugin
- https://developers.google.com/speed/pagespeed/ - Google Page Speed
- http://gtmetrix.com/ - GTMetrix
- http://developer.yahoo.com/yslow/ - YSlow
- http://www.wpbeginner.com/beginners-guide/how-to-install-and-setup-wp-super-cache-for-beginners/ - WP Beginner Tutorial for WP Super Cache
- http://www.wpbeginner.com/plugins/how-to-install-and-setup-w3-total-cache-for-beginners/ - WP Beginner Tutorial for W3 Total Cache
- http://wordpress.org/plugins/wp-database-optimizer/ - WP Database Optimizer Plugin

About the Author

Jeffrey Zinn has been a web designer since 1996. He is a co-founder of Pixel Jar, a WordPress development firm based in Orange County, California since 2004. He is a developer of the AdSanity WordPress ad management plugin. He also co-organizes WordCamp Orange County, an annual WordPress-based weekend conference geared to share knowledge and experience for WordPress users. In his spare time, Jeff enjoys surfing, hiking, traveling, tabletop games, and coffee.

Developing and deploying a website with WordPress and Git

Martin Prunell

WordPress is a free and open source content management system based on PHP and MySQL. It is an extremely simple and flexible tool for both creating simple blogs and complex websites due to the huge amount of plugins provided by its very active community.

What you will learn...

Developing a website using any content management system includes installing themes, tweaking templates and trying out modules and plugins. Using source control management software and a local development box can provide a good way of organizing development iterations and code deployment to a web server.

The suggested way to create a website from scratch in this article involves installing WordPress on a local environment, using a Git remote repository such as Github to store code changes, and deploying the actual site by copying the local database to the server and updating it with the latest code changes.

What you should know...

This tutorial will assume Ubuntu with Apache, MySQL, PHP and Git installed on both development and server and Github for remote code repository. Basic knowledge of Apache, MySQL, PHP, Git, Bash, Vim and general web development is also assumed.

Creating a local environment

First step is to open Ubuntu's terminal and choose a local directory for installing WordPress:

```
me@my_local_box:~$ mkdir -p ~/development
me@my_local_box:~$ cd ~/development
me@my_local_box:~/development/wordpress$ wget
http://WordPress.org/latest.tar.gz
me@my_local_box:~/development/wordpress$ tar -xvzf latest.tar.gz
me@my_local_box:~/development/wordpress$ rm latest.tar.gz
```

Wordpress will store data in a MySQL database. This database needs to be created for local development:

```
me@my_local_box:~/development/wordpress$mysql -uroot -p -e
"CREATE DATABASE my_website"
me@my_local_box:~/development/wordpress$mysql -uroot -p -e
"CREATE USER my_user"
me@my_local_box:~/development/wordpress$mysql -uroot -p -e
"GRANT ALL PRIVILEGES ON my_website.* TO 'my_user'@'localhost'
IDENTIFIED BY 'my_password'"
```

This will create a database, a user and assign all permissions for the newly created user to access the database properly. Database name, username and password values need to be written to a configuration file so that WordPress can connect to the database.

The plan is to deploy the local site to a server, so configuration files are needed for the local and the server website. The problem is that it is very likely that a different configuration for the production server is needed as typically server users have stronger passwords.

me@my_local_box:~/development/wordpress$ cp ~/wordpress/wp-config-sample.php ~/wordpress/wp-config.php

wp-config.php file will store database credentials for WordPress to access the database.

If a different database configuration is needed for the local development environment —as mentioned above, because database name, username or password are different— the following code needs to be added to *wp-config.php*:

Listing 1: Modifying wp-config.php to load local configuration if available:

```
<<LISTING 1>>

if ( file_exists( dirname( __FILE__ ) . '/local-wp-config.php' ) ) {
    include( dirname( __FILE__ ) . '/local-wp-config.php' );
}
else {
    define('DB_NAME', 'my_production_site');
    /** MySQL database username */
    define('DB_USER', 'my_production_username');
    /** MySQL database password */
    define('DB_PASSWORD', 'my_production_password');
    /** MySQL hostname */
    define('DB_HOST', 'my_production_host');
}
<</LISTING 1>>
```

Create a configuration file called local-wp-config.php that will store local database credentials:

```
<<LISTING 2>>

<?php
define('DB_NAME', 'my_database');
 /** MySQL database username */
 define('DB_USER', my_username');
 /** MySQL database password */
 define('DB_PASSWORD', 'my_password');
  /** MySQL hostname */
  define('DB_HOST', 'localhost');

<</LISTING 2>>
```

Now that WordPress has been installed on a local directory and an empty database is available, the local web server needs to be configured.

The goal is to develop the website locally before deploying to the server. That means a local URL is needed so that browsing to that location will show the site being developed.

Adding a local domain to /etc/hosts file: edit the hosts file by executing sudo vim /etc/hosts on Ubuntu's terminal and add the following entry:

127.0.0.1 local.my_site.com

This will point *local.my_site.com* to the local web server.

Create a virtual host file by running sudo vim /etc/apache2/sites-available/*local.my_site.com* and entering the following configuration:

Listing 3: Contents of Apache's virtual host file linking local.my_site.com to the location of Wordpress code on local disk.

```
<<LISTING 3>>

<VirtualHost *:80>
    DocumentRoot /home/me/development/wordpress
    ServerName local.my_site.com
    Options -Indexes
    RewriteEngine On
    RewriteOptions inherit
    ErrorLog /var/log/apache2/my_site_error.log
    LogLevel warn

    ServerSignature on
    <Directory "/home/me/development/wordpress">
        Options Indexes FollowSymLinks
        AllowOverride All
    </Directory>
</VirtualHost>

<</LISTING 3>>
```

Now enable the local site and restart Apache:

```
me@my_local_box:~/development/wordpress$ sudo a2ensite local.my_site.com
Enabling site local.my_site.com.
To activate the new configuration, you need to run:
 service apache2 reload
```

The local website is almost ready except that Apache needs write access to the website. The website code lives under the home directory in this example, preventing Apache's user www-data from writing there. There are two options:

1. Relaxing permissions on the development directory so

everyone can access. This represents security risks and should not be performed on a shared server: *me@my_local_box:~/development/wordpress$sudo chmod -R 777 **

2. Running Apache with your local user:

Edit Apache's environment variables file by sudo vim /etc/apache2/envvars and change the user and group to the current user:

Listing 4: modifying Apache's environment to run with current username. 'me' is the local user in this example. This configuration is meant for the local development environment and not for production.

```
<<LISTING 4>>
export APACHE_RUN_USER=me
export APACHE_RUN_GROUP=me
<</LISTING 4>>
```

Apache needs to be restarted in order to force new configuration to be reloaded:

me@my_local_box:~/development/wordpress$ sudo service apache2 restart

Now WordPress is waiting to be configured at local.my_site.com

Enter the usual fields and click on "Install" to start using the website.

Figure 1: WordPress installation page

Creating a custom WordPress theme

WordPress comes with some themes out of the box. These themes are built for blogging and it is very important to be able to use a customized theme with a specific HTML structure.

Fortunately, adding a custom theme is a matter of creating a few files on a predefined location,

me@my_local_box:~/development/wordpress$mkdir wp-content/themes/sample-theme

Create the following files:
style.css

Listing 5.

```
<<LISTING 5>>
/*
Theme Name: My Theme
Theme URI: http://example.com/example/
Description: A search engine optimized website framework for
WordPress.
Author: You
Author URI: http://my_site.com/
Version: 1.0
Tags: Comma-separated tags that describe your theme
.
Your theme can be your copyrighted work.
Like WordPress, this work is released under GNU General Public
License, version 2 (GPL).

http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

*/
 /* css code providing style to HTML */

<</LISTING 5>>
```

With this files created, the basic structure for the custom team is created, and it can be enabled on WordPress admin.

index.php

Listing 6: contents of index.php

```
<<LISTING 6>>
<?php get_header(); ?>
<div id="container">
    <div id="content">
    </div><!-- #content -->
</div><!-- #container -->
<div id="primary" class="widget-area">
</div><!-- #primary .widget-area -->
<div id="secondary" class="widget-area">
</div><!-- #secondary -->
<?php get_footer(); ?>
<</LISTING 6>>
```

header.php

Listing 7: Header file containing usual code to show in the head section of the document and the html 5 header section.

```
<<LISTING 7>>
<!DOCTYPE html>
<html>
<head>
    <meta charset="<?php bloginfo('charset'); ?>
    <title><?php wp_title(); ?></title>
    <meta name="viewport" content="width=device-width, minimum-
scale=1">
    <link rel="shortcut icon" href="<?php get_stylesheet_directory_
uri(); ?> /favicon.ico" />
    <link rel="stylesheet" media="all" href="<?php
bloginfo('stylesheet_url'); ?>" />
    <?php wp_head(); ?>
</head>
<body>
<div class="wrapper hfeed">
    <header>
      <h1><a href="<?php  echo get_option('home'); ?>"><?php
bloginfo('name'); ?></a></h1>
      <hr>
      <?php wp_nav_menu(array('menu' => 'Main', 'container' => 'nav'
)); ?>
    </header>
<</LISTING 7>>
```

footer.php

Listing 8: Footer with a structure to show footer info

```
<<LISTING 8>>
</div><!-- #main -->
    <div id="footer">
        <div id="colophon">
            <div id="site-info">
            </div><!-- #site-info -->
        </div><!-- #colophon -->
    </div><!-- #footer -->
</div><!-- #wrapper -->
</body>
</html>
<</LISTING 8>>
```

Template for a page. WordPress comes has two basic content types, posts for regular blog posts and pages, for static pages.

page.php

Listing 9: Page structure

```
<<LISTING 9>>
<?php get_header(); ?>
<div id="container">
    <div id="content">
      <?php the_post(); ?>
      <div id="post-<?php the_ID(); ?>" <?php post_class(); ?>>
          <h1 class="entry-title"><?php the_title(); ?></h1>
          <div class="entry-content">
              <?php the_content(); ?>
              <?php wp_link_pages('before=<div class="page-link">' . __
( 'Pages:', 'your-theme' ) . '&after=</div>') ?>
              <?php edit_post_link( __( 'Edit', 'your-theme' ), '<span
class="edit-link">', '</span>' ) ?>
          </div><!-- .entry-content -->
        </div><!-- #post-<?php the_ID(); ?> -->
    <?php if ( get_post_custom_values('comments') ) comments_template()
// Add a custom field with Name and Value of "comments" to enable
comments on this page ?>
    </div><!-- #content -->
</div><!-- #container -->
<?php get_sidebar(); ?>
<?php get_footer(); ?>
<</LISTING 9>>
```

functions.php

Listing 10: functions.php stores code for the theme

```
<<LISTING 10>>

<?php
add_theme_support('nav-menus');
if (function_exists('register_nav_menus')) {
     register_nav_menus(array('main' => 'Main'));
}
function theme_widgets_init() {
    register_sidebar( array (
        'name'          => 'Main Widgets',
        'id'            => 'main_widgets_area',
        'before_widget' => '<li class="main-widgets-area">',
        'after_widget'  => '</li>',
        'before_title'  => '<h3 class="widget-title">',
        'after_title'   => '</h3>',
    ) );
  register_sidebar( array (
        'name'          => 'Secondary Widgets',
        'id'            => 'secondary_widgets_area',
        'before_widget' => '<li class="secondary-widgets-area">',
        'after_widget'  => '</li>',
        'before_title'  => '<h3 class="widget-title">',
        'after_title'   => '</h3>',
```

```
    ) );
}

add_action( 'init', 'theme_widgets_init' );

$preset_widgets = array (
    'main_widgets_area'   => array( 'search', 'pages', 'categories',
'archives' ),
    'secondary_widgets_area' => array( 'links', 'meta' )
);
if ( isset( $_GET['activated'] ) ) {
    update_option( 'sidebars_widgets', $preset_widgets );
}

function is_sidebar_active( $index ){
    global $wp_registered_sidebars;

    $widgetcolums = wp_get_sidebars_widgets();

    if ( $widgetcolums[$index] ) return true;

    return false;
}

?>
<?php if ( is_sidebar_active('main_widgets_area') ) : ?>
        <div id="primary" class="widget-area">
            <ul class="xoxo">
                <?php dynamic_sidebar('primary_widget_area'); ?>
            </ul>
        </div>
<?php endif; ?>

<?php if ( is_sidebar_active('secondary_widget_area') ) : ?>
        <div id="secondary" class="widget-area">
            <ul class="xoxo">
                <?php dynamic_sidebar('secondary_widgets_area'); ?>
            </ul>
        </div>
<?php endif; ?>
<</LISTING 10>>
```

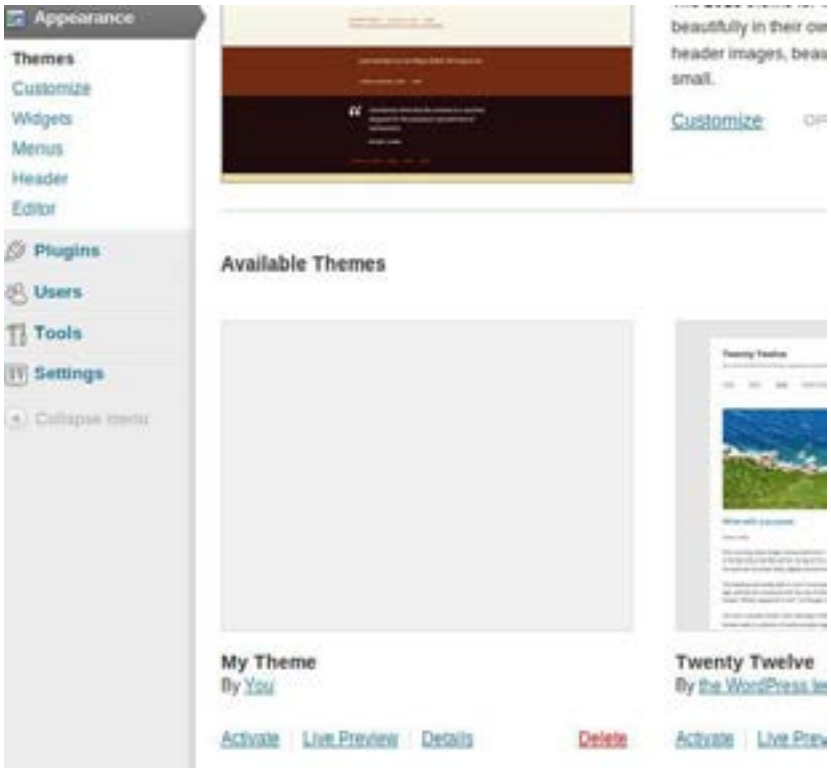


Figure 2: Activating custom theme

Managing your web site's code and deployments

Now the site is ready to add pages, styling, plugins, etc., it is a good time to add the code to the repository using Git:

me@my_local_box:~/development/wordpress$ git init

Create a file called .gitignore and add any files that should not be committed such as *local-wp-config.php*. Then commit the changes.

me@my_local_box:~/development/wordpress$ git add .

me@my_local_box:~/development/wordpress$ git commit -m "Initial commit"

Connect with a Github repository:

me@my_local_box:~/development/wordpress$git remote add origin git@github.com:me/my_site.git

Assuming that username "me" and repository "my_site" exist on Github.

Push to remote:

me@my_local_box:~/development/wordpress$ git push origin master

Log in to the remote server and execute the following steps:

1. Pull the code from Github to the production directory, usually /var/www. A good option for pulling code from Github (or any other Git server for that matter) is using deploy keys. After deploy keys are installed (installation depends on the Git server chosen) execute: git clone *git@github.com:me/my_site.git*

2. Create an empty database for the production website. Remember database name and user credentials should match those configured on *wp-config.php*.
3. Create Apache configuration for the production website in the same manner as for the local site, changing folders and domains accordingly. This time www-data should own the directory where the code is located. Enable the site and restart Apache.
4. Navigate to the production domain, WordPress installation should show up again.

Summary

There is a great number of plugins available for WordPress. That means a rapid development environment is needed for testing these plugins or writing customized ones. Any code or database changes can be tested locally and then moved to the server. Code changes are pushed to the server by using code repository, but databases changes cannot. WordPress is ultimately a content management system, so database stores basically content. Production server should be treated as the "Gold Standard" regarding database content.

About the Author

Martin Prunell is a co-founder at Sophilabs a software development company with focus on start-ups and Open Source projects. He has 20 years of experience in the IT industry and is dedicated to deliver beautiful software built with state-of-the-art technologies on top of solid architectures. He can be contacted at *mprunell@sophilabs.com* or at *http://www.martinprunell.com*.

HI, MY NAME IS KEVIN.

WORDPRESS

WEBSITE DESIGN & DEVELOPMENT
IS WHAT I DO.

www.anythinggraphic.net

wordpress • design • development • e-commerce • seo + more

ANYTHINGGRAPHIC

anythinggraphic.net (505) 975-6684 @anythinggraphic

Software Developer's JOURNAL
new ideas & solutions for professional programmers

Wordpress: Maintain & Deploy

GJ Petersen

Ever wondered how your brilliant code fits in the world? This article takes you on a journey to the outskirts of Wordpress. Not just the plugins, but to the environment where Wordpress lives and how you can deploy, configure, interact with and maintain Wordpress on a server. You might ask "But aren't there plugins for everything?" Yes there are and that's exactly my point. In my opinion there are problems solved in Wordpress nowadays that don't belong there. We'll touch that subject in the security section.

To have a good reading experience with this article you'll need to have at least some knowledge of web servers, system administration, provisioning software and a bit of an entrepreneurial spirit.

After having built websites myself in the distant past I ran into Wordpress not to far after it's birth and it was 'love at first sight'. Wordpress solved a problem for me at that time. I could do the programming myself but wanted an effective solution to easily and quickly put information online. Moreover, The biggest lessons I've re-learned in the last year is: "Don't do everything yourself". If there are tools available that can help you, use them. That sure can be hard if you're a software developer by heart, because then you can build anything and everything.

Stack building

The first thing when setting up a web service is a proper deployment strategy. Still very often the application or service is setup by hand which ends in a textbook end result. "Yeah, sorry, the guy that built it doesn't work here any more". And even if he did, rebuilding a service should be as standard as running an update. Remember, the blueprint of the service and server are more valuable then the server and service itself. In combination with a backup of course.

There is a slight discipline curve if you're used to 'quick fixes'. However, doing everything with deployment software makes your work safe and it pays out in the long run. All though deployment software starts to really pay out when you're maintaining multiple systems, it's a major step from a Business Continuity perspective. Just imagine you hosting company's storage platform completely failing. How quick can you set up elsewhere?

Whatever your pleasure

Having used and build several different deployment tools I ended up using Opscode's Chef. Chef has quite a thick layer of logic, not much less then puppet and the like, nevertheless, you can build the kitchen sink with it. The learning curve is 2 to 3 months but it is definitely worth it. Two distinct Chef definition used in this article are recipes

and run-lists A run-list in Chef is a set of recipes. A recipe in it's turn is a set of code blocks on how a software package is to be installed and configured on a system. Should your choice be Puppet a recipe is called a manifest. Is CFEngine your tool of choice a recipe is called a bundle. Your mileage may vary depending on the tool you use but in concept the definitions are the same.

Stack layers

The prerequisites needed for building a full web server stack running a Wordpress website basically are an installed linux OS of your choice, an ssh account and sudo rights. Your Wordpress OS dependancies should prevail over those that a deployment tool requires. Meaning, the application to be installed dictates the software and it's versions on the system and not the deployment tools used to build it. To have a good understanding of what a web-stack is comprised of I carry around a mental image of the following stack schematic. It shows, similar to the OSI-model, the logical layers of a server running Wordpress.

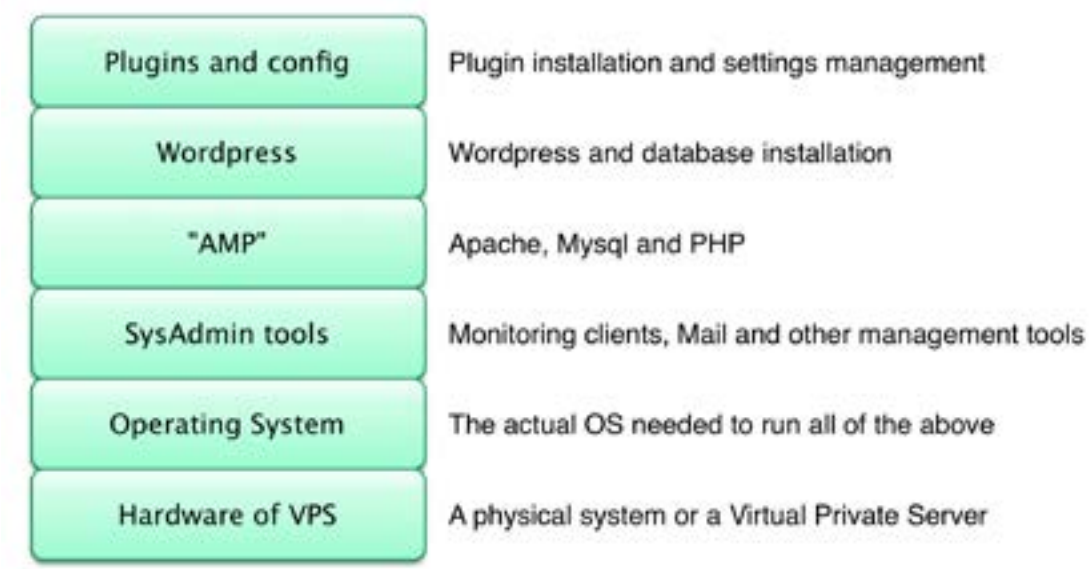


Figure 1: A logical web stack representation.

The different layers help to understand how, and in what order, a stack is built for long term maintainability. I'd advise you to group your deployment recipes in the same manner. Using Chef myself I group recipes in run-lists exactly as the stack schematic displays. To give an example, "AMP" would install Apache, Mysql and php.

There is some skill involved on installing tools and their dependancies. For instance, if you install Apache on Ubuntu, a set of default modules and (apt-get) packages come along for the ride. Should your setup need specific packages then you need to take care of that in your recipes. Moreover, should a package recipe contain a lot of customizations to fit your needs. You might consider giving it it's own recipe and simply include a link to that recipe in the first one. But I guess modularity is a given for a specialist like you.

When deploying a Wordpress website on a new server, you obviously need to deploy the full stack. Is it a second Wordpress website on an existing server then you only need to execute the run-

list for the top two parts of the stack. A new configuration is added beforehand so that your deployment software is aware of installing the second one. Chef uses JSON files (among others sorts) and for a website I usually don't need more information then the directives shown in Listing 1.

Listing 1: A .json file with the necessary directives for a Wordpress installation

```
<<LISTING 1>>
{
    "id": "somewhere",
    "description": "A WP site Somewhere",
    "customer" : "somebody",
    "node": "wn999",
    "fqdn": "www.somewhere.com",
    "serveraliases": [
        "somewhere.com"
        ],
    "database": "wdb999",
    "db_user": "wdb999_u",
    "wp_plugins": [
        "akismet",
        "contact-form-7",
        "easy-fancybox",
        "daikos-text-widget",
        "wordpress-seo",
        "google-analytics-dashboard",
        "google-analytics-for-wordpress",
        "p3-profiler",
        "broken-link-checker",
        "social-metrics"
        ]
}
<</LISTING 1>>
```

As you could see in the web-stack schematic I have separated the Wordpress and the Plugins (and settings) layers deliberately for maintainability reasons. If I want a setting changed in *wp-config.php* for all Wordpress instances I can do so running just the top run-list from the stack.

You might debate over the fact that it belongs in the second layer from the top, but that would conflict with some failsafe conditions I built in. My Chef configuration knows what websites are running on what server due to the "node" directive in the .json file. I check per Wordpress instance whether or not the destination folder or database exist. If either of these is found I progress to the next website. That way an automated install never breaks an already running website.

That also goes for the "Plugins and settings" run-list It addresses the plugin installations for all Wordpress instances on a given server. If a plugin directory exists I skip to the next one. Next to that it reads in the *wp-config.php* and adds anything from the recipe that is not configured yet. This might seem devious but it gives me one place to add an extra directive to which is the *wp-config.php.erb* template. Deploying this update simply means executing Chef with the "Plugin and settings" run-list on all my servers and it's done.

Search replace or add

Automated deployment introduces a scientific concept called idempotence (see references). Shortly meaning: "I can do this a hundred times, the result is always the same". For maintaining a file like *wp-config.php* this can result in problems. Adding something to the file without checking if it's already in there can result in duplicate entries. Best thing to do is use a regular expression based search function. That way you can explicitly define what entries you want to be in the file. With that regular expression you explicitly ascertain whether or not it's already in there.

Password unknown

When your automated installs start to get bigger you don't want them waiting for you to type in a password (twice :-). One of the entries in the aforementioned *wp-config.php* file is the password with which Wordpress connects to a database. This password is stored in two places, the database on the backend and the "*wp-config.php*" file. This password is of no interest. What I mean is, I don't care what it is, just as long as Wordpress can connect to the database. Deploying a Wordpress stack can do this in one go.

1. You generate a string (with openssl, pwgen, or any tool of your choice)
2. Use this string when writing out the *wp-config.php* variable DB_PASSWORD
3. Create the database user for your Wordpress instance and set this string as the database password.

If you do this in one go you'll never see the password but Wordpress has a working database it can connect to.

Command-line WordPress

With maintaining a growing set of Wordpress installations the need for scalable tools becomes imminent. There are tools out there like InfiniteWP that help people maintain multiple Wordpress websites but those are al UI based. These are mostly used by people that do not have extensive background knowledge on web servers. Understand me well, there's nothing wrong with that, as long as they help you get the job done that's ok.

However, this article is about the fauna Wordpress lives in. So, for good control and real upscale maintenance there's nothing like a command-line interface and decent scripts/tools. The backend tool I prefer and use in a lot of wrapper scripts is wp-cli (see references). It's an amazing CLI interface to everything Wordpress can do from the Dashboard and lot's more.

Since automated provisioning was where we started of, how does this work for wp-cli? It's maintainer provides the tool in .phar format for a while now. This simply means you place the *wp.phar* file (renamed to wp) in a directory that's in your $PATH and it's usable right away. Shortest recipe I ever wrote ;-).

Deployment

Once wp-cli is operational the possibilities are endless. For instance; Installing 'just another Wordpress website' is done with the command wp core install. You can add the parameters *--admin_name, --admin_email* and *--admin_password* and your admin account is set up in

the same instant. If you create the admin user based on a a hashed password stored in your recipe's code it's not readable to anyone but you know what it is. You can generate it like this:

echo "somepassword" | openssl passwd -crypt -stdin

From there on out, you initiate new users as needed. For illustration purposes a command-line example that says enough.

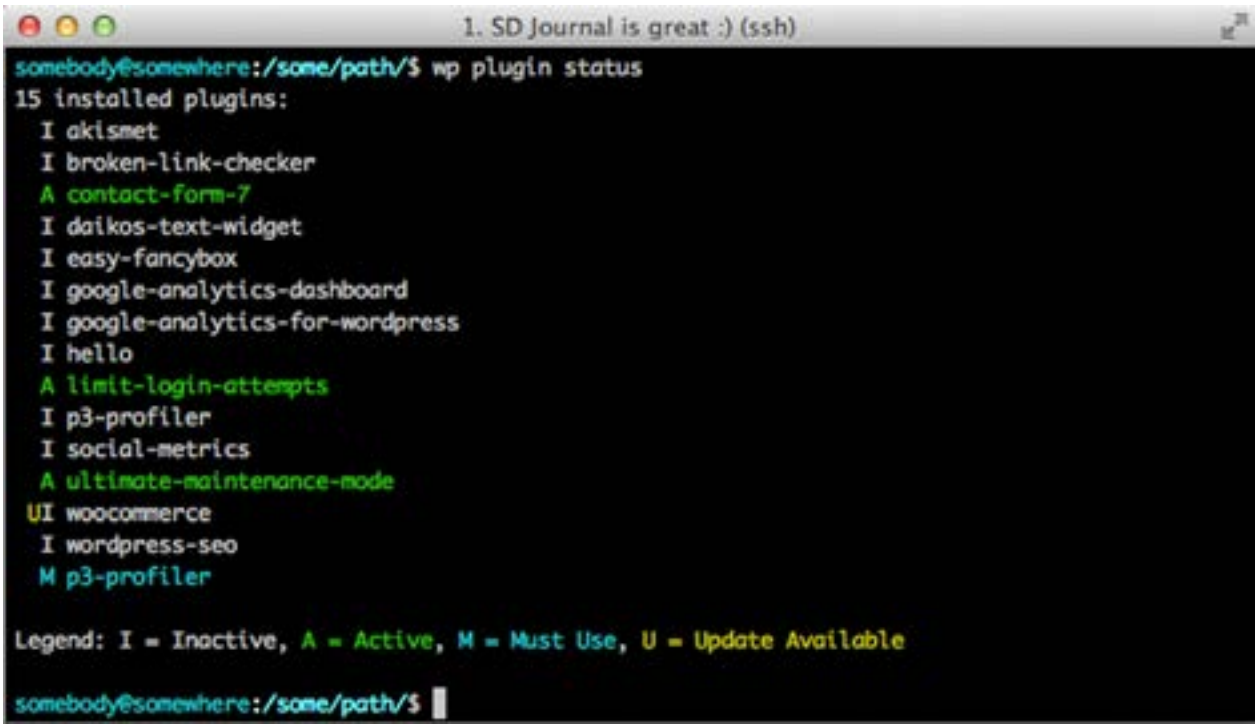
wp user create test_user *test_user@somewhere.com* | grep Password | mailx -s "Your password for 'test_user'" *test_user@somewhere.com*

Replace the hard values with parameters taken from a config file, like the json example, and deployments are fully unattended.

Updates

In case of updating Wordpress from the command-line, this is done with the command wp core update. The same works for plugins, where you can show a status overview with wp plugin status as shown below.

Figure 2: WP-CLI showing available plugin updates.



You can imagine writing a wrapper script parsing the output and returning a status code on whether there are updates available. If you let your monitoring software call this script your always in control. I use Nagios (see references) for this and the result is an email (or text message) the minute an update is available for any plugin the customers use.

Maintenance

Another built-in subcommand from wp-cli is wp db for database interaction. There are three options particularly handy being export, connect and query. From a business continuity perspective I run full Mysql exports but for all other temporary actions the wp db export command dumps a mysql backup in your current directory. Very smart to do just before migrations, copying websites, etc.

The second option, although for die-hard Wordpress database guru's, is wp db connect. This will instantly enter a mysql prompt for the given database.

Reporting

The third useful subcommand is "wp db query". This command can generate all sorts of CRON'ed reporting for you. For useful Wordpress queries there's Google that can help you, but to give you an example:

```
$ wp db query "SELECT count(*) FROM wp_comments WHERE wp_comments.comment_approved = 'spam' \G;" | grep -v "^\*\*"
```

Resulting in the response:

```
count(*): 113
```

Imagine writing a wrapper scripts invoked by your monitoring software, you could set an alert on when a spam box of a Wordpress site is flooding. This is good management information, because you get to advise a customer to install a spam blocker plugin.

To back-step to the previous chapter about the unknown password. When managing a database you often need to connect to the database using username/password. wp-cli does this for you by simply reading the password from the *wp-config.php* file. Meaning, as stated in the previous chapter and that's why I love wp-cli, I don't care what it is I just use it.

Last thing on CLI based Wordpress interaction with wp-cli I'd like to share is that the maintainer of wp-cli has made it real easy to include custom commands. Any function you like to call from the command-line can be easily wrapped in a php class and added to wp-cli as a new command.

The wp-cli maintainer has made this so easy that the community picked up on this. More and more plugin developers have their plugin commands included for wp-cli by default. Check the list of community commands for plugins (see reference) supported by wp-cli.

Wordpress and security

Last but certainly not least, security. An item that should get your constant attention. Reality, and my experience, learns that it always gets the attention it needs right after calamities. Big plans to avoid this kind of grief in the future, until it's laid to rest again somewhere in a dusty corner. You can have all sorts of tools in place to help you, but in the end it's the people that need to act on alerts or issues. Moreover, stay aware!

The most important rule of thumb out there when it regards security: "Always run your updates!". No, seriously, keeping track of your log files will show that bot's are always skimming the internet looking for exploitable web frameworks that are not updated.

The threat is real

To create some awareness; These are some log entries from the recent past that speak for themselves.

Listing 2: Log entries due to hack attempts.

```
<<LISTING 2>>
# Yet another Bulletin Board
xx.xx.xx.xx - - [29/May/2013:11:56:05 +0200] "GET /blog/join.php HTTP/1.1"
404 10358 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;
SV1;)"
xx.xx.xx.xx - - [29/May/2013:11:56:10 +0200] "GET /blog/signup.php
HTTP/1.1" 404 10360 "http://cap5.nl/blog/" "Mozilla/4.0 (compatible;
MSIE 6.0; Windows NT 5.1; SV1;)"
xx.xx.xx.xx - - [29/May/2013:11:56:12 +0200] "GET /blog/YaBB.cgi/
HTTP/1.1" 404 10362 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT
5.1; SV1;)"
xx.xx.xx.xx - - [29/May/2013:11:56:14 +0200] "GET /blog/YaBB.pl/
HTTP/1.1" 404 10361 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT
5.1; SV1;)"
xx.xx.xx.xx - - [29/May/2013:11:56:16 +0200] "GET /member/register
HTTP/1.1" 404 10363 "http://cap5.nl/" "Mozilla/4.0 (compatible; MSIE
6.0; Windows NT 5.1; SV1;)"

# Theme's and plugins
xx.xx.xx.xx - - [03/Sep/2013:12:13:34 +0200] "GET /wp-content/plugins/
zload.php HTTP/1.0" 404 43625 "http://cap5.nl/wp-content/plugins/
zload.php" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98; Win 9x
4.90)"
xx.xx.xx.xx - - [02/Sep/2013:11:37:52 +0200] "GET /wp-content/themes/
twentytwelve/404.php HTTP/1.0" 500 - "http://cap5.nl/wp-content/
themes/twentytwelve/404.php" "Mozilla/4.0 (compatible; MSIE 6.0;
Windows 98; Win 9x 4.90)"
<</LISTING 2>>
```

I run Wordpress and not a YaBB so these urls should not hit my website. And as you can see from the last two lines, even plugins and themes are a target. These should also be updated in a timely fashion. A very helpful metaphor I always use to explain security is comparing a website to a house. If you leave the door open or unlocked eventually somebody wil come in uninvited and do stuff you don't want them to. I hope by showing these log entries you are aware of the threat. So what to do to stay afloat?

Stay healthy

Never change something in code of others, or worse in the Wordpress framework itself. This will block the possibility to upgrade components when updates are available. Also make sure you honor and use the Wordpress internal update mechanisms. The Wordpress core developers are high on security (hmm, that sounds weird ;) so you should be to. The framework powers almost 19% of the websites on the internet and therewith becomes a target. When Wordpress is updated see to it that your plugins and/or themes use the latest Wordpress coding conventions. Deprecation warnings are not to annoy, they have a purpose. We should be grateful for the developers who use these. They give us the time to stay in sync and update our own code in a timely fashion.

The temptation to do 'quick hacks' is high because customers often want fixed fee and 1 time stuff. Let me tell you, if you convince them on longterm maintainability and continuity you'll make them confide in you and they'll turn into loyal and returning customers.

Security measures

As I stated in the beginning, there are plugins developed for Wordpress that needn't be if only people would have a bit more knowledge of web-stacks, servers, networks and (http) traffic. I know it's an illusion ;-).

Figure 3. Limit Login Attempts warning the user.

I want to highlight one plugin, called "Limit Login Attempts", that is on the edge in regards to the aforementioned statement. The reason I love this plugin is that it creates user awareness. Although there are tools that can do this stuff better, it is a fantastic plugin. The problem it solves is protecting Wordpress against brute-force attacks. I'll leave session hijacking for what it is because that's an article on it's own. Nevertheless, the good thing about the plugin is that users see it.

I makes a user more careful in handling their passwords. That being said, there's also a flip-side to this. I ran into a forum thread about this plugin a while back and read about a user originating a malicious IP address. He stated that the IP address came from Afghanistan. Ok, now what? What I mean to say is that users shouldn't be burdened with this kind of information. Let specialists do their thing (that's you ;) and make sure your customers acquire your carefree service with the utmost regards for their valuable business efforts.

Bigger means

The tools I was referring to just now that do the best job securing your web-stack are IDS's and Web Application firewalls. My two personal favorites are "OSSEC IDS" and OWASP's "mod_security" (see references). There's books on both of them, so I not going to try and fit it all in here. However, the one thing I want to mention is that they are of a pro-active nature.

OSSEC for instance can manipulate a firewall (iptables) in realtime dropping connections based on urls found in a websites access log. Paraphrasing the rule: "If more then a set number of 404's is detected from a certain IP source it's disallowed connections instantly for a certain period of time. That can also be configured for the Wordpress login POST, making the plugin "Limit Login Attempts" obsolete, and therewith not hassling users with this. The trick is convincing potential new customers the value of you taking care of it.

Then there's OWASP's "mod_security" which basically is an immense set of regular expressions filtering out anything out of the ordinary. Mysql injections for instance have certain fingerprints. These are defined in mod_security's filter rules. So if a MySQL injection (see references) is detected, the connection is disallowed before it even enters the Wordpress framework.

Both tools do the same thing but on a different level in the OSI model (ref: wiki for osi model). The beauty of it is that they solve the problem in the right place, not burdening Wordpress with this. Agreed that this is more of a system engineers cup-of-tea, I think they are things to consider. Worst of all, there are almost no hosting companies that provide you with these kind of security measures.

Coming to the end of my article the last tip I want to give you if you want to explore important aspects of security for Wordpress. Have a look at the Wordpress plugin Bulletproof Security (see references). This plugin gives you a good understanding of a lot of security items that are on the edges or even outside Wordpress. Although it its a plugin, Bulletproof Security definitely looks at the right stuff to make your environment more secure.

If you want these kind of security measures, a professional hosting platform or advice, please contact us via:

Email: *office@cap5.nl*
Phone: +31 10 236 23 21
Or go to: http://cap5.nl/go-dutch

References

- *WP-CLI – http://wp-cli.org/*
- *WP-CLI community commands – https://github.com/wp-cli/wp-cli/wiki/List-of-community-commands/*
- *Wordpress Plugin "Limit Login Attempts" – http://wordpress.org/plugins/limit-login-attempts/*
- *Idempotence – http://en.wikipedia.org/wiki/Idempotence*
- *MySQL injection – http://en.wikipedia.org/wiki/SQL_injection*
- *OSI Model – http://en.wikipedia.org/wiki/Osi_model*
- *Nagios Monitoring software – http://www.nagios.org/*
- *OSSEC IDS – http://www.ossec.net*
- *Mod_security – http://www.modsecurity.org*

Author: GJ Petersen

I've studied electronics and have been working in ICT for 25 years now. I have maintained a lot of different size environments during the years up to the scale of enterprise level backup infrastructures including tape silo's and storage area networks. Having owned and sold a self built online invoicing system recently (certified for use with Dutch TAX laws). Therefor I've been walking around with a DevOps mindset for at least 3 years. I currently own a Dutch hosting company specializing in Wordpress Hosting.

Software Developer's JOURNAL
new ideas & solutions for professional programmers

Mobile Navigation: User Friendly Techniques For Small Devices

Kevin Donnigan

The number of total website traffic is going to increase as data speeds accommodate for the things people want. And they want things fast. With the growth of traffic on mobile devices doubling since last year to 24%, the responsiveness of mobile apps and websites has skyrocketed. It is more important than ever to deliver a usable experience to mobile users.

What you will learn...

This tutorial is going to walk you through one of the most practical, functional, and usable ways to deliver navigation menus that users will recognize and find easy to use. Screen real estate is of the utmost importance and not paying attention to your navigation menu can take up a whole lot of it. You will learn a CSS method for turning navigation menus into sleek elements for these smaller screens. View the demo at *http://micjamking.github.io/Navigataur*.

What you should know...

Since this is a WordPress tutorial, you should know the basics of how navigation menus work in a fresh install. You should also have a basic understanding of how a child theme works and why you should not edit core files of a purchased premium theme or framework. If you wish to dig deeper into this subject, checkout the WordPress Codex: Child Themes at *http://codex.wordpress.org/Child_Themes*. Also, knowledge of HTML, CSS, and media queries are essential.

Before We Start

I personally develop using the Genesis framework for WordPress and build on top of that with custom child themes that I make for my clients. I also develop with progressive enhancement, meaning that users with newer browsers get the best experience while users with older browsers (namely Internet Explorer 8/9 and below) get a usable experience. They won't know the difference because they aren't switching browsers and testing the website like we do (which you should be doing).

If you are building your own custom theme or working in another theme, the line numbers mentioned in this tutorial may not match up. The Twenty Twelve and Twenty Thirteen themes already have a navigation system for mobile menus, but we want our own. Since I'll be using Twenty Twelve as an example, we will be removing these lines for the sake of this tutorial. You will need to look at your theme and adjust what to remove and where to add the new code. This CSS technique can be applied anywhere.

Let's Begin

This particular method is very easy to implement. It's called Navigataur and was put together by Mike King, an Interaction Designer at Ikayzo. Follow him on Twitter at *http://www.twitter.com/micjamking*.

I'll be implementing his technique into WordPress by changing selectors in the CSS and updating our *header.php* with two lines of new markup. Navigataur utilizes an input, a label, and CSS properties to make our navigation mobile friendly. You can view the Github Repository at *https://github.com/micjamking/Navigataur*.

The Markup

Below is some simple markup for navigation that displays with a default WordPress install using the Twenty Twelve theme.

**Note that I have removed classes on the <li> elements to shorten the length of the code for the example.*

Listing 1. Markup

```
<<LISTING 1>>
<div class="menu-main-nav-container">
    <ul class="nav-menu" id="menu-main-nav">
        <li class="menu-item"><a href="#">Home</a></li>
        <li class="menu-item"><a href="#">About</a></li>
        <li class="menu-item"><a href="#">Services</a></li>
        <li class="menu-item"><a href="#">Portfolio</a></li>
        <li class="menu-item"><a href="#">Contact</a></li>
    </ul>
</div>
<</LISTING 1>>
```

What we need to do is include two extra elements in the code directly above our menu container. To do this, copy *header.php* from your parent theme's folder (in this case it is twentytwelve) and paste it into your child theme's folder. Always use child themes so you can develop in an upgrade-safe manner.

Now open up your new *header.php* file in your favorite text editor. In the default install of Twenty Twelve, look for line 43 and delete it. This is the default theme's code for mobile navigation. Again, we want our own. Your particular theme may not have this.

```
<h3 class="menu-toggle"><?php _e( 'Menu',
'twentytwelve' ); ?></h3>
```

Now look for this line right below what we just removed.

```
<?php wp_nav_menu( array( 'theme_location' =>
'primary', 'menu_class' => 'nav-menu' ) ); ?>
```

We need to add our input and label. Put the following code directly above the code above.

```
<input type='checkbox' id='toggle'/>
<label for="toggle" class="toggle" data-
open="Main Menu" data-close="Close Menu"
onclick></label>
```

Here's a quick explanation:

The checkbox input box is hidden in our stylesheet, but when you click the menu it will become 'checked.' We will reference the checked state of the input in our CSS with a pseudo-class of :checked. When the input is checked, we will display the navigation menu. Genius.

The label element puts a label on our menu and has two attributes: data-open and data-close. In our CSS, we will target the content property within our selector to change the text on the menu depending on whether it is open or closed.

The content property allows us to add text and images to the markup using only CSS. It's not really considered markup anymore, but can be styled any way you choose. Note that this property only works with the :before and :after pseudo selectors. Chris Coyier has an awesome post about the content selector at *http://css-tricks.com/css-content* along with a slew of other tips and tricks.

The CSS

Here's where all the magic happens — the styles! If you look at the comments, the CSS is broken up into two pieces: essentials for functionality and styles for presentation.

Listing 2. Essentials and styles

```
<<LISTING 2>>
/* Functional Styles (Required) */
/* Tim Pietrusky advanced checkbox hack (Android <= 4.1.2) */
body { -webkit-animation: bugfix infinite 1s; }
@-webkit-keyframes bugfix { from {padding:0;} to {padding:0;} }

/* Nicolas Gallagher micro clearfix */
.clearfix:before, .clearfix:after {
     content: "";
     display: table;
}
.clearfix:after {
     clear: both;
}

#masthead {
     position: relative;
}

#toggle, .toggle {
     display: none;
}

.nav-menu > li {
     float:left;
     list-style: none;
}
```

```
@media only screen and (max-width: 768px) {
     .nav-menu {
          display: none;
          opacity: 0;
          position: absolute;
          right: 0;
          width: 100%;
     }

     .nav-menu li {
          display: block;
          margin: 0;
          width: 100%;
     }

     .nav-menu li a {
          -webkit-box-sizing: border-box;
          -moz-box-sizing: border-box;
          box-sizing: border-box;
          display: block;
          text-decoration: none;
          width: 100%;
     }

     .toggle {
          cursor: pointer;
          display: block;
          position: relative;
          -webkit-touch-callout: none; /* Disables OS callout when
touching and holding a link */
          -webkit-user-select: none;
          user-select: none;
     }

     #toggle:checked ~ .nav-menu, .main-navigation ul.nav-menu, .main-
navigation div.nav-menu > ul {
          display: block;
          opacity: 1;
     }
}
/*--------------------------------
 Presentation Styles (Editable)
--------------------------------*/
#masthead {
     height: 100%;
     min-height: 100px;
     padding: 0 20px;
}

#masthead > h1 {
     color: #DFDFDF;
     float: left;
     font-family: Georgia;
     font-size: 28px;
     font-style: italic;
     padding: 30px 0 0;
}
```

```
.menu-main-nav-container {
     display: block;
     float: left;
}

.nav-menu, .nav-menu > li, .nav-menu > li > a {
     height: 100%;
}

.nav-menu li a {
     -webkit-box-sizing: border-box;
     -moz-box-sizing: border-box;
     box-sizing: border-box;
     display: block;
     font-weight: normal;
     font-size: 16px;
     line-height: 1;
     padding: 10px 20px;
     text-align: left;
     text-decoration: none;
     -webkit-transition: all 0.25s linear;
     -moz-transition: all 0.25s linear;
     -o-transition: all 0.25s linear;
     transition: all 0.25s linear;
}


.nav-menu li a:hover, .nav-menu li a:focus {
     background: #F2F2F2;
     box-shadow: inset 5px 0 0 #51C1F1;
     color: #51C1F1;
     padding-left: 30px;
}

.toggle {
     margin: 0 auto;
     width: 200px;
     z-index: 2;
}


@media only screen and (max-width: 768px){

      .nav-menu {
           background: #FFF;
           border-top: 1px solid #51C1F1;
           padding-top: 1em;
      }

      .nav-menu, .nav-menu > li, .nav-menu > li > a {
           height: auto;
      }

      .nav-menu > li > a{
           padding: 15px 15px;
      }

      .nav-menu > li > a:hover, .nav-menu > li > a:focus {
```

```
            background: #F2F2F2;
            box-shadow: inset 5px 0px #51C1F1;
            padding: 15px 15px 15px 25px;
      }

      .toggle:after {
            background: #51C1F1;
            -webkit-border-radius: 2px;
            border-radius: 2px;
            -webkit-box-sizing: border-box;
            -moz-box-sizing: border-box;
            box-sizing: border-box;
            color: #FFFFFF;
            content: attr(data-open);
            display: block;
            font-size: 12px;
            margin: 33px 0;
            padding: 10px 50px;
            text-align: center;
            -webkit-transition: all 0.5s linear;
            -moz-transition: all 0.5s linear;
            -o-transition: all 0.5s linear;
            transition: all 0.5s linear;
            width: 200px;
      }

      .toggle:hover:after {
            background: #45ABD6;
      }

      #toggle:checked + .toggle:after {
            content: attr(data-close);
      }
}
@media only screen and (max-width: 479px){
      #masthead > h1 {
            text-align: center;
      }

      #masthead > h1, .toggle:after {
            float: none;
      }

      .toggle:after {
            text-align: center; width: 100%;
      }
}
<</LISTING 2>>
```

You will have to change the selectors depending on your markup. As always, style the visual properties as you wish. You now have a functional, usable, CSS-only responsive navigation menu that works great on mobile devices.

For now, let's go through some of the CSS techniques we used above. First is the bugfix at the very top that targets Android 4.1.2 and below. There is a bug in the Android operating system and it doesn't completely understand the pseudo-class + general selector.

Below that, we are declaring a clearfix hack that gives us a way to contain floats without resorting to using presentational markup. Next, we hide the checkbox input. We also hide the label on viewports higher than 769px. When the viewport of the device (or the width of your browser) is 768px or below, we will display the label and hide the navigation.

When a user clicks on the label, the checkbox becomes checked and then pseudo selectors tell the menu to show and the label to change. You can fancy this up by adding some styles of your own.

Summary

It's a very simple approach and is one of my favorites. It can be implemented for large navigation menus with secondary menu items as well. Just add some styling to the nested lists!

The reason I prefer CSS methods like this one is because it doesn't add a lot of weight to your site like a lot of the jQuery methods out there. If you pay attention to the world of search engine optimization, load times have an impact and building for speed is something everyone should do, if only for speed limitations on mobile networks.

On the other hand, there are a few nice looking jQuery methods out there that look and function really well. That is a tutorial for another day.

About the Author

My name is Kevin Donnigan and I'm the owner of Anything Graphic where I design and develop responsive WordPress websites for clients across that nation. Full-time, I am the front-end website designer and developer for a startup company called @Pay - The Two Click Checkout for Web + Email.

Website: *http://www.anythinggraphic.net*
Twitter: @anythinggraphic
Google +: *https://plus.google.com/u/0/109603281001417133017*
Facebook: *http://www.facebook.com/anything.is.graphic*
Dribbble: *http://dribbble.com/anythinggraphic*
LinkedIn: *http://www.linkedin.com/in/anythinggraphic*

Software Developer's JOURNAL
new ideas & solutions for professional programmers

Holding Down the Fort: Five Tips for a More Secure WordPress Site

Brett Widmann

WordPress is one of the most widely used open-source content management systems used for website development. With that in mind, it's crucial to ensure that any production-level install is secure from vulnerabilities and potential compromises. This article aims to introduce website administrators with best practices, tools, and techniques to better protect their WordPress powered websites.

What you will learn…

- Applying best practices for a more secure WordPress website
- Installing plugins to enhance WordPress security
- Configuring WordPress core files and the database

What you should know…

- Basic WordPress knowledge of installing and configuring plugins
- Experience with PHPMyAdmin and MySQL
- Familiarity of basic WordPress administration

WordPress is often regarded as the most widely used web content managements systems (CMS) in the world. WordPress creator Matt Mullenweg announced at WordCamp 2013 that WordPress currently accounts for 18.9% of the top 10 million websites in the world[1], which is an indicator that not only is the platform popular, but it justifies the CMS as a more than ideal target for hackers. By using a number of plugins, techniques, and best practices, administrators can significantly reduce the chances of their WordPress site getting compromised. Note that the following techniques provide a foundation for securing WordPress, and don't delve into all aspects of WordPress and security.

Security Tip #1 - Remove the Default *admin* Username

WordPress, by default, creates an administrator level account with *admin* as the username. Typically, this is a very simple entry point for a hacker has to compromise a WordPress website, and an easy vulnerability to fix.

1 http://wordpress.tv/2013/07/29/matt-mullenweg-state-of-the-word-2013/ (video)

For administrators that login with the *admin* username, please create a new username for yourself, then delete the original username supplied with WordPress. To do this, login in to the WordPress Dashboard and navgiate to *Users* > *Add New* and create a new user that has the Administrator role.

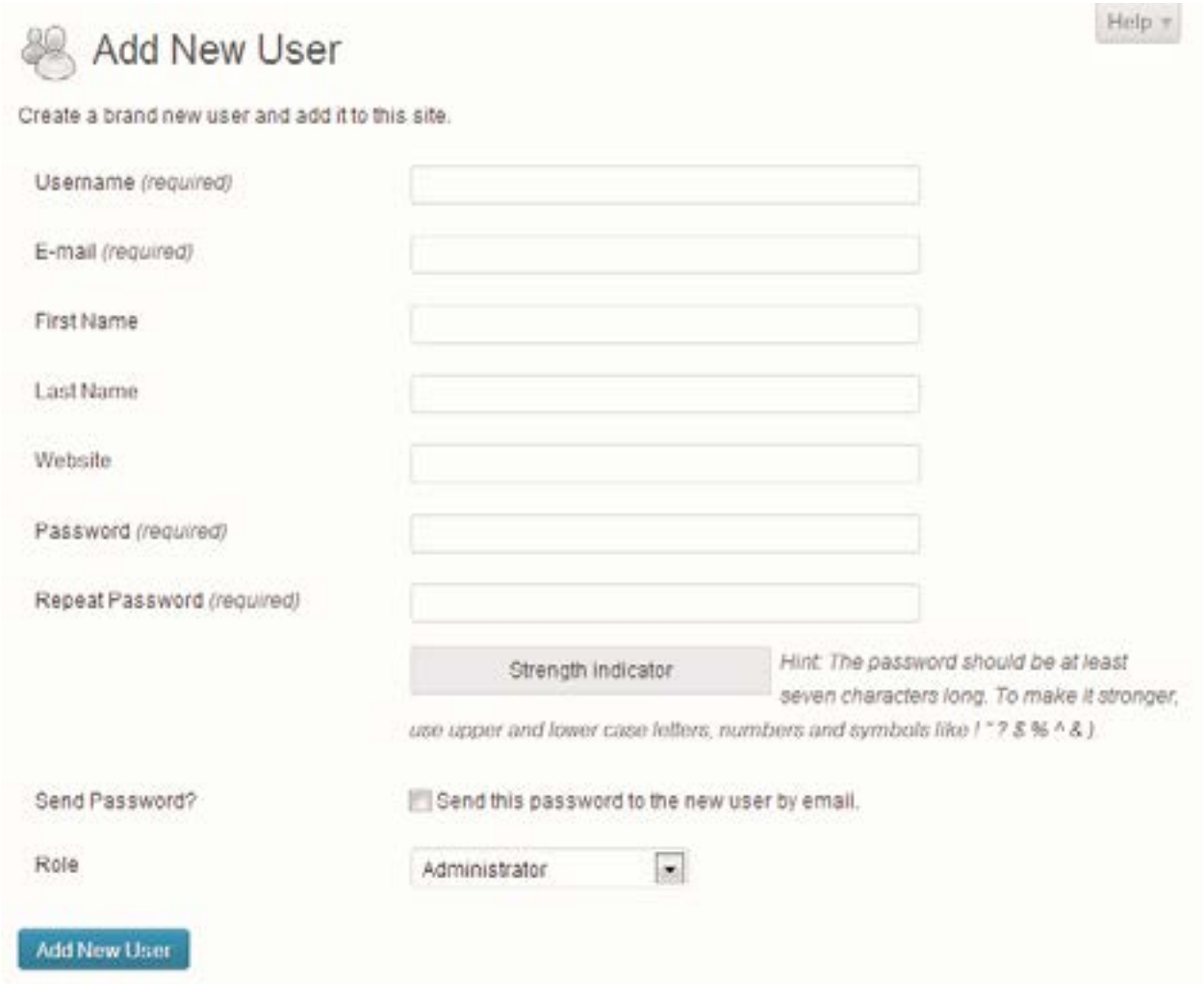


Figure 1. The Add New User Screen

This new account will serve as the new default administrator account. Once created, log out of the admin account and into the newly created account. Once logged in to the Dashboard, navigate to Users > All Users, select the admin account and delete it.

While creating the account, you may have noticed that no two accounts in the same WordPress install can have the same email address. When creating a new administrator account for yourself, use a secondary email address until you login with the new account, and delete the *admin* account. Once the account is deleted, navigate to the new account under *Users* > *All Users*, and select *Edit* to change the email to your original one.

Security Tip #2 - Create a Strong Password

A simple, one word, or few character password just doesn't make the cut when it comes to maintaining security. The WordPress password strength indicator suggests the "password should be at least seven characters long. To make it stronger, use upper and lower case letters, numbers and symbols like ! " ? $ % ^ &)". Anything over the seven character minimum is always welcome, too.

To change your own password, navigate to *Users* > *Your Profile* and enter the new password into both of the password fields. Save your changes, and the password has now been reset to the new one.

Figure 2. The Add New Password Section

It is highly recommended that all users follow this general rule of thumb. As an administrator, you have the option to change all passwords individually, or notify all users to do so. For future reference, using a plugin like *WordPress Password Expiry*[2] that forces users to change their passwords at certain intervals may also be useful.

Security Tip #3 – Keep WordPress Updated

A clear cut step an administrator can take to ensure their WordPress website is secure is to keep the CMS up to date with the latest stable releases and security patches. At the time of this article's composition, WordPress is currently on version 3.6.1, which addresses 13 bugs, three security issues, and additional security hardening for file uploads.[3] With this in mind, it's imperative to keep the WordPress core, as well as plugins and themes, up to date.

Thankfully, the update process is simple. Whenever an update for WordPress is available, a prominent yellow banner appears notifying the administrator that it's time to update. In most cases, the update is a simple one click process that does not require leaving the browser or any manual uploading via S/FTP. When taken to the update screen, the option to update plugins and themes is also available.

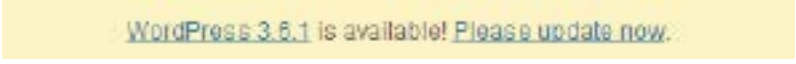


Figure 3. The WordPress update banner

2 http://wordpress.org/plugins/wordpresspasswordexpiry/
3 http://codex.wordpress.org/Version_3.6.1

Security Tip #4 – Backup Frequently

Keeping a backup of your WordPress database, or entire site, is crucial in the event that a website gets hacked beyond the point of repair. Sometimes, redeploying a backup of a website is necessary. Many times, a simple database restore can alleviate any issues from a hacker, as most vulnerabilities target stored data within the database. There are a number of approaches to backing up a WordPress website, below are a few suggestions:

Host Backups – Depending on your host, they may include or offer some sort of backup or feature within their control panel. In some cases, backups can be scheduled and deployed at the control of the user.

Manual Backups – The slowest of the options, but it gives the most control. Typically, a manual backup involves copying the entire install via S/FTP and exporting the database from PHPMyAdmin. This approach may be appropriate based on server restrictions, or if the site isn't regularly updated.

Plugins – Plugins typically offer the most flexible and versatile solution to what a site administrator is looking for. WordPress has a number of plugins and solutions for backup, ranging from a simple database backup, to a full-fledged website backup.

For the sake of keeping focused, we'll assume that plugins are the ideal scenario. For database backup, there are a plethora of options administrators can use, one of which is a free plugin called *BackWPup*[4]. Not only does this plugin backup the database, but it gives administrators the option of backing up an entire WordPress install that is fully customizable. Once customized, the backup can be scheduled to run at selected intervals via cron job. Administrators can also take advantage of storing backups via third party services, such as DropBox, Amazon S3, and Microsoft Azure to name a few.

Security Tip #5 – Limit Login Attempts

The most persistent way a hacker tries to get access to a site is to try to login to the Dashboard a multitude of times until they are successful, or their method of attack doesn't yield a correct password. One of the easiest ways to stifle this process is to limit the number of attempts a user has to unsuccessfully login to the WordPress Dashboard. While WordPress doesn't have this feature included, there's a handy plugin called *Limit Login Attempts* that allows administrators to customize settings such as maximum number of allowed retries, duration of lockout, and time until retries are reset. The figure below demonstrates the types of customizations and a basic log of IP addresses that have been locked out. To find these settings, navigate to *Settings* > *Limit Login Attempts* in the WordPress Dashboard after the plugin is installed and activated.

4 http://wordpress.org/plugins/backwpup/

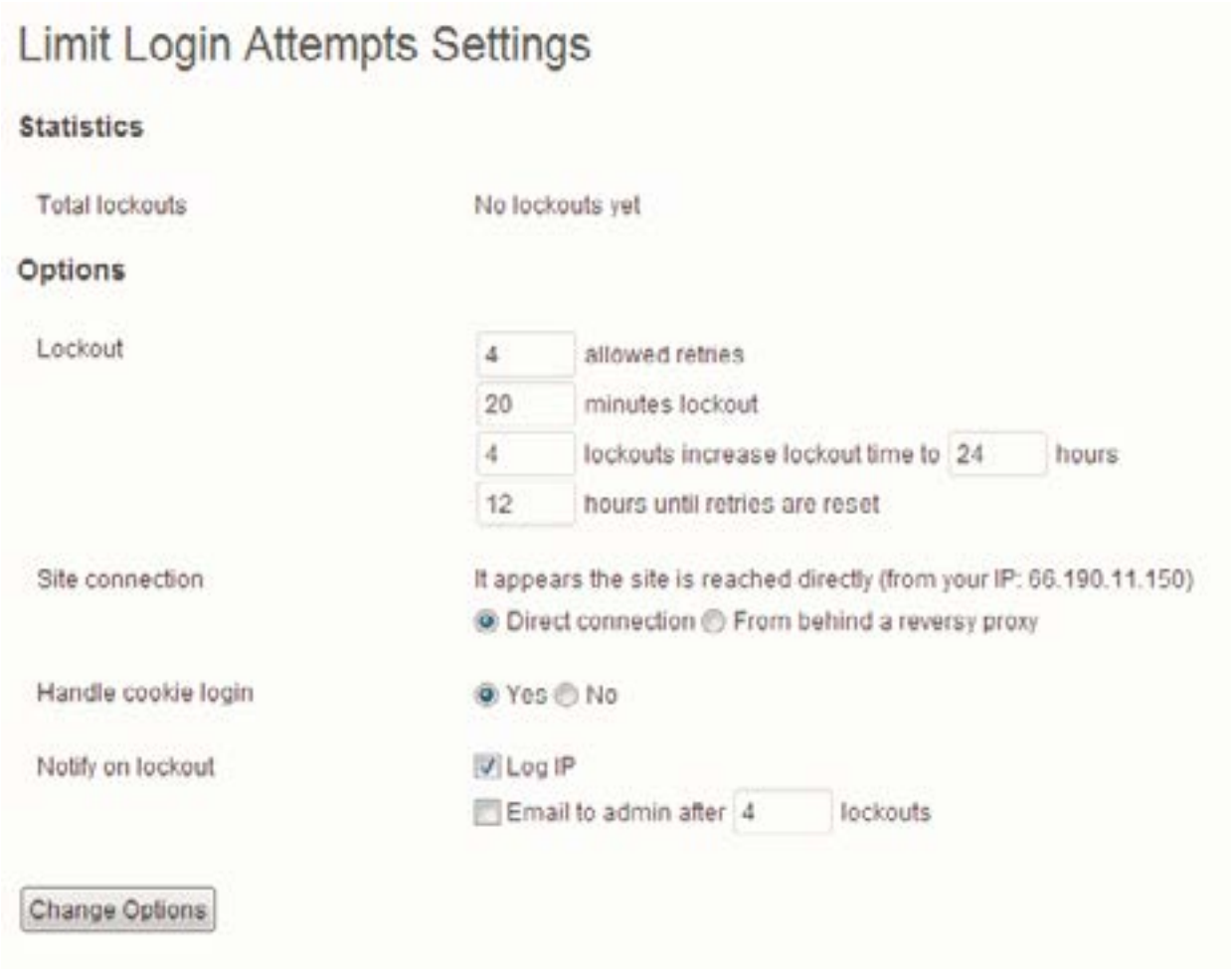


Figure 4. Limit Login Attempts plugin settings panel

Summary

With the ever growing popularity of WordPress, the platform has transformed itself from a blogging system to a full-fledged CMS. With its growth in popularity and extendibility, it's become a prime target for online hackers, so having a secure WordPress website is crucial to any administrator. By following the aforementioned tips and best practices, you've provided a solid foundation to having a more secure WordPress powered website.

About the Author:

Brett Widmann is a freelance web developer, specializing in WordPress and front end development. Originally a strong supporter of the Flash platform, Brett has changed his interests in animation using CSS3 and JavaScript and pushing the limits of the browser and what it can do. When not behind a computer screen, Brett serves as Digital Director for *Caught in the* Carousel, an online music magazine and can be seen attending concerts and music festivals in Chicago and Milwaukee.

Meet the **Developer-Friendly** Payment Solution

3 easy steps to optimized checkouts:

1

Create the checkout page

With Gate2Shop, you can optimize your payment pages by using ready-made templates or by customizing payment pages to your site look and feel.

2

Test and optimize

An effective payment page variant testing tool, A/B Testing helps you gain insight into user behaviour, increase payment conversion in the short and long term.

3

Accept payments worldwide

With dozens of alternative and local payment methods offered in multiple currencies, the personalized checkout allows you to reach users from all around the world.

✔ Easy integration ✔ Cross-platform ✔ Secure

gate2shop
Sell. More.

Call for a free consultation: +44 20 3051 0330

www.g2s.com

Beyond Mobile Gestures

Dr. Yildirim Kocdag

Mobile Gestures technology lets you control your device actions by movements or touchs. They can be likely some shortcuts of complex actions. The success of these shortcuts are depend on becoming natural behavior of users. The movement can be with device movement or movement on the camera of device. Touchs can be on screen or on device.

These gestures are becoming one of the main reasons while customers choosing their new smart phone such as battery consumption, screen size, weight or processor. The mobile phone companies are investing much on the features to be selected by customers. Especially the phone companies who offer their customer same Operating System (Android) are in the competition. The leading firms in this race are Samsung, Motorola, Nexus, LG, HTC, Sony, Huawei and ZTE. Apple is not in the same competition, however they have a strategy on the gestures which is obivious. On the other hand, these features are subject to fill patent applications.

History

While smart phones offers wide task list to their user such as voice comminucation, SMS, Multimedia, internet, GPS, mobile games, their input list gets limited. The only input way was hard keyboards. The hard keyboard surpassed by touch screens. After touch screens and multi touch options, shortcut movements gets more popular. The movements on the screen with touch called screen gestures.

Some of the known and popular screen gestures are

- spread or pinch for resize an image,
- swipe right for delete a message or answer a call,
- slide to unlock phone,

The screen gestures was an evolution in the gesture history, they are still in use, however the usage is reducing.

Figure 1. Slide

The second invention on the subject was motion gestures. They also called as 3D gestures. After the devices equipped with sensors, the motion gestures became favourites by users. The main

sensors in devices of today are accelerometers, gyroscopes, and orientation sensors.

Most popular gestures are using by phone users are:

- moving phone to face to answer a call while there is an incoming call,
- turn device over to mute a incoming call,
- double tap on top of your device to go on the top of a list.
- Moving your phone left or right down to control a craft in a game,

Mobile phone manufacturers add motion gestures to the rom of their products. They can be eligible by user via settings menu. The below image is the motion gestures menu of Samsung Galaxy S3.

Figure 2. Menu

User can turn on or turn off the gesture himself. Direct call and Smart Alert are my favourites for this phone. While Direct call option is on, if the phone brings to face of user, it calls directly the contact on the screen. if the phone placed down and turned off screen, smart Alert feature shows the notifications to user after user pick up the phone. The image below illusturates the usage of Smart Alerts.

Figure 3. Smart Alerts

If your phone does not have these gestures in settings menu, you can download a motion gesture application from Google Play. Listening sensors or registering a service to root does not need root permissions, therefore an application with appropriate permissions can be installed safely to use motion gestures. What is more you can create your own action for a specified input with these applications or your own program. For example you can send a message while your phone is in an appropriate slope.

The latest invention on gestures are air gestures. They also called wave control or hover control. These gestures help device owners to control their phone without touching them. With proximity sensor and image processing technology, the camera or sensor captures the hand movements and gets them as an input for an action.

Most adopted ones by user till now are:

- Hands on your device screen to mute a call,
- Pass your hand on camera to see next browser page,
- Move you hand for next the song,
- Moving your hand side from left to right or vica verse to capture screen.

The left air gesture is illusturates the mute a call. While phone rings user can mute the phone via holding his hand on phone for a short while. It raise up a service that catches hand movements when a call comes.

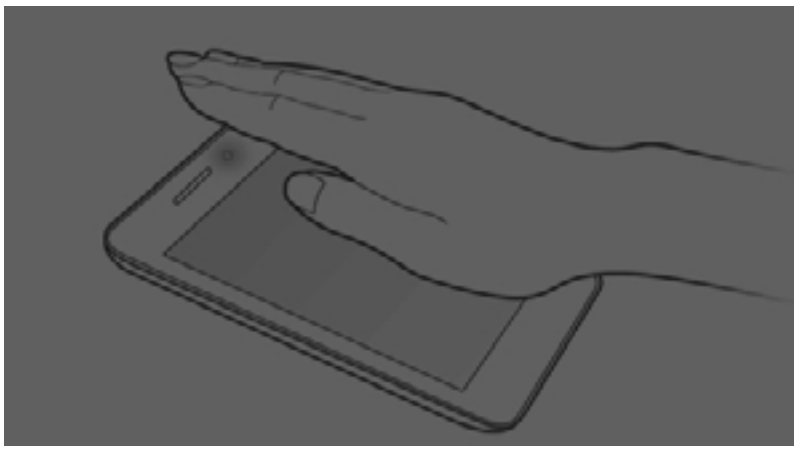


Figure 4. Left air gesture

Another air gesture is for accepting or rejecting an incoming call. The user can move his hand from left to right to accept a call, or visa verse to reject a call. Whenever you move your hand to accept it, the call accepts and speakers gets on. This is a handy feature if a call comes from your girl friend,while you wash dishes☺.

Figure 5. Accepting and rejecting call

In Samsung S4 and some models of Sony these gestures come default on rom, you can use them via turning on them in settings menu. In addition to this, there are applications in google play that provides some of these features. To use them, you should have an above average proximity sensor in the device.

The Architecture of Gestures

Sensors can not be up all the time, because of power consumption issues. That is why when need a special action to turn them on. Tha main action listener should be registered in root to make the gesture available after start up. After start up, the main action should be listened. If an expected action comes, the service attaches sensor listeners and if sensor listeners catch the input shows the output action. After the main action completed the sensors should be released with a detach sensor function. The below arcihtecture illustrates the point

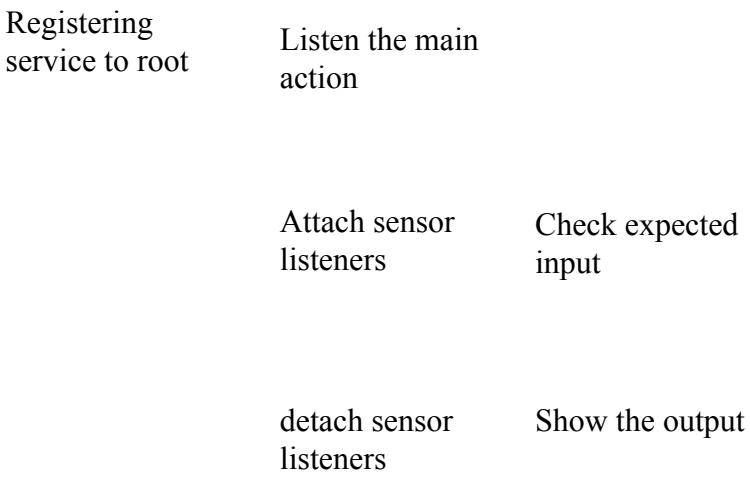


Figure 6. Architecture

Lets describe calling a contact gesture with this architecture. A broadcast receiver service should be registered for catching the contacts screen in root. Whenver the contacts screen is shown, the service starts the sensor listeners to catch movement of user. If the user move the phone to his face, the output action is shown which is calling the contact. After the contact screen is closed service should give an order to detach sensor listeners. Registering the main action can be done via a menu setting.

The architecture is designed for an android device, it can be different in other embedded operating systems.

Limitations

This handy gestures have some difficulties while we apply them. Without a main action which described above, they can not work. An application which turns on sensors all the time phone is up, can drain the battery fast. In addition to this, the sensivity of sensors are not in good for all phones. Some phones there is a delay for sensor updates. What is more, the range of sensor is different in different model. For example in some devices, the proximity sensor range is from 0-5. Equipped best sensor devices are expensive than others. The processor and memory are still an issue for image processing or sensor listening.

Next Step

Mobile technology is improving really fast. The gestures will be more popular in near future. The limitations will be solved soon. And the gestures will be stabile with new equipments.

1. I believe new sensors will be introduced in near future. Pressure or resistance can be candidates. The value range of sensors can be redefined and sensivity will be better.
2. Batter consumption problem will be solved. Drawing a letter on screen while screen off can be detected, because there will be no consumption problem and gesture wont need the main action.
3. Location, the gestures will be registered with different locations, for example if the user in a shopping center, the gesture will be defined to buy an item.
4. Speech technology will be more popular, the gesture will combine with speech. Speech and hand movement will get unique output.
5. After the phones gets wearable devices, the usage of gestures will increase dramatically. A new method will be introduced, the body gestures, the devices can be controlled with body movements.
6. Mind Waves, there is some equipments that can catch mindwaves. If you look at ios or android application market you can find some application or games that works with these equipments. They will be a part of gestures, users will be control their devices with mind waves.

Conclusion

Mobile Gestures technology will be most important preference for phone users while they choose their phone. There is still important limitations, however, the phone manufactures invest to solve these limitations. After the problems are solved, the gestures will be more popular and handy.

Screen gestures, Motion gestures and Air gestures are todays gestures. Body gestures and mind wave gestures will be future gestures. Today there is inputs for gestures, in future there will be also range or option list for this input. Location and speech services will provide a range or option list for the inputs.

About the Author

Dr. Yildirim Kocdag is a Computer Engineer from Istanbul. Currently using Android, Objective-c, c#, *vb.net*, *asp.net*, javascript, SQL and Oracle. His favourite areas in Computer Science are Compilers, Expert Systems, Digital Image Processing, AI and Extreme Programming. *ykocdag@yahoo.com*

Software Developer's JOURNAL
new ideas & solutions for professional programmers

Expert MySQL Design Practices

Ronald Bradford

All businesses, organizations, websites and governments use data to present information to users. This ranges from contact information of your company, products for sale, where you are and where you have been, and the transactions for your bank account. The right architectural practices for data management ensures successful scalability of your website, application or product managing data. There is no one way to architect a technology solution, however, there are many ways poor design can seriously impact the need for more hardware and resources, and most importantly, take significant time to re-architect causing development of new features to be affected.

Why is there a need for design practices?

Consider this question. How do you describe the need and role of a software architect to a non technical person? This is a generic description for the layperson.

You want to build a new home on a vacant block of land. In order to build a lasting home with all the expectations for today's living, a lot of planning, design, and development occurs long before you can move in, paint the walls, add furniture, cook, entertain and reside.

Your home is built on a foundation that is generally a solid and rigid block of concrete that was carefully planned on a level surface. The plumbing and electrical wiring was carefully prepared before the concrete was set. If the foundation is not level, solid, and meeting the design plan then no matter how much work you put into your home, you will spend more time compensating for the deficiencies of the foundation. Do you spend more and more time and expense trying to compensate, or do you have to give up, tear down the failed attempt and build again?

Designing and developing a website or a software product uses the same approach. With software the amount of investment, the time, and expense is far less than building a home, however the lack of a plan and design can easily lead to a poor product that nobody wants to use.

The availability of open source software, free hosting and ample online resources enables anybody to take an idea and create a software solution with very little effort. If you are building a shack on the beach, you do not invest a lot, you accept the cracks in the walls, the leaky roof, the lack of plumbing. Your first software application will likely be exactly the same. Do you continue to add on, or is there a time you decide you want to start over for your family to live at the beach over the summer with all the expectations of a holiday resort?

Not all plans are the same. If you live in the Amazon, your home is not built with concrete, and has additional different needs for building higher off the ground to avoid tidal flows and to have room for food storage and livestock to have shelter. One size does not suit all. Just like there are shanty towns to million dollar penthouses in large cities, the diversity of websites and software solutions are analogous.

Understanding how to design and develop a successful and scalable software solution takes practice, trial and error, and a good amount of training. Working in a team that includes experienced resources that have completed projects ensures you can learn from the mistakes of others and cultivate a healthy environment for accelerated learning.

It is unfortunate that many startups and open source projects, even the most popular LAMP applications suffer from common design mistakes. These practices misuse MySQL and therefore can shine a poor light on the software and the misconception that MySQL does not scale. Many of today's the largest websites, all the top telecommunication companies, online gaming sites and numerous other companies all rely heavily on your data stored in and managed with MySQL.

Today, many organizations have developers, IT managers and C level executives where the most senior resource has a handful of years of experience or less. They have little or no formal experience in design, data structures, software development, production deployment and testing methodologies. Combined with having never worked in a team environment with more skilled professionals, many software engineers today have a very limited world view of how large and successful products have been developed in the past. This lack of depth in skills reflects how the entire team learns and grows and without the right historical influences too many common beginner mistakes are repeated.

About Expert MySQL Design Practices

This new series by leading MySQL Expert Ronald Bradford helps the software engineer understand, appreciate and develop the right skills and techniques to build scalable software solutions. These proven and reproducible design practices will ensure your use of MySQL to improve performance, scalability and reliability.

These expert design practices are from 25 years of professional experience following formal university qualifications in computer science. All of these practices are written for use with a MySQL based data system however most of the content in these practices predate the existence of the MySQL product and have stood the test of time with emerging technologies and software development approaches. Many practices apply directly to other data stores, whether relational or the new NoSQL products and include working with persistent and non-persistent data storage products.

More information about the series can be found at *http://j.mp/MySQLDP*

DP#4 The importance of using *sql_mode*

What if the data you retrieved from the database did not match the data the application claimed to had successfully stored? How comfortable would your organization feel about your skills and the products that are being used to store important information if data integrity was not guaranteed?

MySQL employs a terrible default technique known as silent truncation where the product determines that it knows about your data better than you. Never has the saying "do not assume" because it makes an "ass" out of "u" and "me" been more applicable.

Customer Example

A HTML form for new customers provide input fields for the customer first and last name. Good design was considered with the HTML form client validation to ensure that each field could not exceed 20 characters in length. However, the database design is different, where the first name is only defined as 10 characters. In most cases this is sufficient, however for first names longer than 10 characters, the data retrieved does not match the data that was apparently successfully stored because there was no SQL error. The following SQL reproduces this situation.

```
DROP TABLE IF EXISTS dp13;
CREATE TABLE dp13 (
 customer_id  INT UNSIGNED NOT NULL AUTO_INCREMENT,
 first_name  VARCHAR(10) NOT NULL,
 last_name  VARCHAR(20) NOT NULL,
 PRIMARY KEY (customer_id)
) ENGINE=InnoDB DEFAULT CHARSET utf8;

INSERT INTO dp13 (customer_id, first_name, last_name)
   VALUES (NULL,'Evangeline','Jones');
INSERT INTO dp13 (customer_id, first_name, last_name)
   VALUES (NULL,'Christopher','Smith');
INSERT INTO dp13 (customer_id, first_name, last_name)
   VALUES (NULL,'Alexander','Bell');

SELECT * FROM dp13;
+-------------+------------+-----------+
| customer_id | first_name | last_name |
+-------------+------------+-----------+
|   1 | Evangeline | Jones  |
|   2 | Christophe | Smith  |
|   3 | Alexander  | Bell   |
+-------------+------------+-----------+
3 rows in set (0.00 sec)
```

As you can see, the first name of Christopher Smith is not actually correctly stored in the database. MySQL DID NOT produce an error message, rather it performed a silent truncation of the data.

Defining *sql_mode*

To demonstrate what level of data integrity you should expect with MySQL, you must define the *sql_mode* configuration option. The following example demonstrates the dynamic syntax for a given connection and the error you should expect.

```
SET SESSION sql_mode='STRICT_ALL_TABLES,NO_ZERO_DATE,NO_ZERO_IN_DATE';

TRUNCATE TABLE dp13;
INSERT INTO dp13 (customer_id, first_name, last_name)
   VALUES (NULL,'Christopher','James');
ERROR 1406 (22001): Data too long for column 'first_name' at row 1
SELECT * FROM dp13;
Empty set (0.00 sec)
```

When MySQL is first installed the following configuration option should always be added to all environments as a default.

```
$ cat /etc/my/cnf
[mysqld]
sql_mode=STRICT_ALL_TABLES,NO_ZERO_DATE,NO_ZERO_IN_DATE,NO_ENGINE_
SUBSTITUTION
```

For more information, refer to the MySQL Reference Manual for sql_mode at http://dev.mysql.com/doc/refman/5.6/en/server-sql-mode.html

NOTE: MySQL provides many different options with *sql_mode*. Careful consideration is needed to determine which options are best for your application. Some options help in providing syntax and compatibility with other database products however these can affect and even break existing products written specifically for MySQL.

MySQL Warnings

The underlying cause of this loss of data integrity is how MySQL handles success and error conditions with SQL Statements. There are the obvious success and failure states, however MySQL has a third state known was warnings, or more specifically success with warnings. As the use of warnings is uncommon with other data store products, many applications, developers and programming languages ignore checking for warnings, or are simply unaware of this inbuilt feature.

Using the MySQL command line client, you can get a visual indication of warnings following an SQL statement which then help the need for reviewing what warnings occurred.

```
INSERT INTO dp13 (customer_id, first_name, last_name)
VALUES (NULL,'Christopher','Smith');
Query OK, 1 row affected, 1 warning (0.00 sec)

SHOW WARNINGS;
+---------+------+-----------------------------------------------------+
| Level   | Code | Message                                             |
+---------+------+-----------------------------------------------------+
| Warning | 1265 | Data truncated for column 'first_name' at row 1 |
+---------+------+-----------------------------------------------------+
1 row in set (0.00 sec)
```

When using PHP there is no indication of SQL warnings unless you specifically check after every SQL statement. For example:

```
<?php
 $con = mysqli_connect('localhost', 'scott', 'sakila', 'design');
 if (mysqli_connect_errno()) {
 print 'Failed to connect to MySQL: ' . mysqli_connect_error() .
"\n";
 exit(1);
 }
 if (!mysqli_query($con, 'INSERT INTO dp13 (customer_id, first_name,
last_name) ' .
       'VALUES (NULL,"Christopher","Holt") ')) {
 print 'Failed to insert data: ' . mysqli_error($con) . "\n";
 }
 if (($warnings = mysqli_warning_count($con)) > 0) {
 if ($rs = mysqli_query($con, "SHOW WARNINGS")) {
  $row = mysqli_fetch_row($rs);
  printf("%s (%d): %s\n", $row[0], $row[1], $row[2]);
  mysqli_free_result($rs);
 }
 }
 mysqli_close($con);
?>
```

The best recommendation is to avoid all situations where MySQL can produce a warning and does not provide the best possible data integrity.

Refer to the MySQL Reference Manual for more information on SHOW WARNINGS at

http://dev.mysql.com/doc/refman/5.6/en/show-warnings.html

The Larger Problem

This underlying problem is actually more difficult to correct for an existing production system than enabling the correct *sql_mode* configuration option. Using the customer example, the identification of any rows that are 10 characters in length could be valid, or may have been truncated. There is no easy way to obtain the actual value that was originally submitted. The use of the correct numerical data type (DP #14) can provide a check constraint for values, however it can also suffer from the same truncation problem. You especially hope that this does not affect your payroll, your frequent flyer points balance or your accumulated score from your favorite online game.

The solution is to avoid the problem of producing incorrect data.

Review

While this example is using a character data type, field truncation can also occur with numeric and date data types. The use of applicable *sql_mode* configuration settings is a critical MySQL design practice to ensure adequate data integrity that all systems need to implement.

More References

- *http://ronaldbradford.com/blog/why-sql_mode-is-important-part-i-2008-07-17/*
- *http://ronaldbradford.com/blog/why-sql_mode-is-important-2011-06-01/*
- *http://ronaldbradford.com/blog/why-sql_mode-is-essentialeven-when-not-perfect-2012-02-16/*

- *http://ronaldbradford.com/blog/best-practices-additional-usersecurity-2010-06-03/*
- *http://ronaldbradford.com/blog/dont-assume-dataintegrity-2010-03-06/*
- *http://effectivemysql.com/presentation/mysql-idiosyncrasiesthat-bite/*

DP#8 The disadvantages of row at a time processing

It can be hard for software engineers to understand the following principle, however it is very important for improving performance and obtaining immediate scalability options. The principle is **"Do Less Work"**. That is, run less SQL statements.

Just one method to achieving the execution of less SQL statements is to eliminate Row At a Time (RAT) processing. In simple terms, do not perform identical repeating SQL statements in a loop. Relational algebra, and the Structure Query Language (SQL) specification is specifically designed to work with sets of data, or as I describe, Chunk At a Time (CAT) processing.

Customer Example

Your online social media website lets you send messages to multiple friends at one time. You enter the message, select the friends you want to receive the message and click send. While the user waits a moment and gets a success message, behind the scenes the application runs the following SQL statements to record your request.

```
START TRANSACTION;
INSERT INTO dp8_message_sent(message_id, user_id, message, created)
VALUES(NULL, 42, 'Hey guys. Just a reminder. The poker game will start on Friday at
8pm.',NOW());
SELECT @message_id :=LAST_INSERT_ID();
INSERT INTO dp8_message_recipient(message_id, from_user_id, to_user_id, status)
VALUES (@message_id,42,16,'New');
UPDATE dp8_user_notification
SET new_message = 'Y',
  new_message_count = new_message_count + 1
WHERE user_id = 16;
INSERT INTO dp8_message_recipient(message_id, from_user_id, to_user_id, status)
VALUES (@message_id,42,18,'New');
UPDATE dp8_user_notification
SET new_message = 'Y',
  new_message_count = new_message_count + 1
WHERE user_id = 18;
INSERT INTO dp8_message_recipient(message_id, from_user_id, to_user_id, status)
VALUES (@message_id,42,99,'New');
UPDATE dp8_user_notification
SET new_message = 'Y',
  new_message_count = new_message_count + 1
WHERE user_id = 99;
INSERT INTO dp8_message_recipient(message_id, from_user_id, to_user_id, status)
VALUES (@message_id,42,21,'New');
UPDATE dp8_user_notification
SET new_message = 'Y',
  new_message_count = new_message_count + 1
WHERE user_id = 21;
INSERT INTO dp8_message_recipient(message_id, from_user_id, to_user_id, status)
VALUES (@message_id,42,62,'New');
UPDATE dp8_user_notification
SET new_message = 'Y',
  new_message_count = new_message_count + 1
WHERE user_id = 62;
COMMIT;
```

You can define the table structures used in this example with:

```
DROP TABLE IF EXISTS dp8_message_sent;
CREATE TABLE dp8_message_sent(
 message_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
 user_id  INT UNSIGNED NOT NULL,
 message VARCHAR(500) NOT NULL,
 created  DATETIME NOT NULL,
 PRIMARY KEY(message_id),
 KEY(user_id)
) ENGINE=InnoDB CHARSET utf8;
DROP TABLE IF EXISTS dp8_message_recipient;
CREATE TABLE dp8_message_recipient(
 message_id  INT UNSIGNED NOT NULL,
 from_user_id  INT UNSIGNED NOT NULL,
 to_user_id  INT UNSIGNED NOT NULL,
 status ENUM('New','Read','Deleted') NOT NULL,
 PRIMARY KEY(message_id,to_user_id),
 KEY(from_user_id)
) ENGINE=InnoDB CHARSET utf8;
DROP TABLE IF EXISTS dp8_user_notification;
CREATE TABLE dp8_user_notification(
 user_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
 new_message ENUM ('Y','N') NOT NULL DEFAULT 'N',
 new_message_count INT UNSIGNED NOT NULL DEFAULT '0',
 PRIMARY KEY(user_id)
) ENGINE=InnoDB CHARSET utf8;
```

The average software developer may not see the problem here. In your test environment you executed 12 SQL statements and the code worked fine, i.e. it met the requirements for the function. However, while producing the correct result, this is a poor code approach.

This example shows not one repeating query, but two. Lucky you only sent the message to a few friends. If you sent it to 200 friends, you have a significant number more SQL statements to execute. This time the code executes 402 SQL statements for the same feature. The response time to the user is longer, the application connection has to remain open longer and the database has more work to do.

This popular site is sending thousands of messages per second, so the problem is compounded to produce an excess of unnecessary work, not just for the database, but the application web server connections as their are longer open requests.

The solution is straightforward. **Remove repeating queries**. It's not rocket science. This is a simple design practice I teach as the problem is evident on most consulting engagements. Popular products including Drupal and WordPress also implement this poor practice and developers that extend these products propagate this poor practice excessively. If this development approach can be easily found in a few common functions, in it generally a clear indicator this problem can be found throughout the code.

Here is the same operation performed efficiently.

```
START TRANSACTION;
INSERT INTO dp8_message_sent(message_id, user_id, message, created)
VALUES(NULL, 42, 'Hey guys. Just a better reminder. The poker game will start on
Friday at 8pm.',NOW());
INSERT INTO dp8_message_recipient(message_id, from_user_id, to_user_id, status)
VALUES
(LAST_INSERT_ID(),42,16,'New'),
(LAST_INSERT_ID(),42,18,'New'),
(LAST_INSERT_ID(),42,99,'New'),
(LAST_INSERT_ID(),42,21,'New'),
(LAST_INSERT_ID(),42,62,'New');
UPDATE dp8_user_notification
SET new_message = 'Y',
  new_message_count = new_message_count + 1
WHERE user_id IN (16,18,99,21,62);
COMMIT;
```

No matter how many friends you send a message to, only 3 SQL statements are executed. In these queries we see two different examples of leveraging the set capabilities of SQL to perform chunk at a time processing. We discuss the benefits of the multi-values INSERT in more detail with DP#10.

Customer Example 2

The following is a simple example for an online store processing function. Your shipping provider provides an update of all packages that were processed by them for a given date. For each packing tracking code that you have recorded with orders they provide a last known status. For example if the package were successfully delivered, is in transit, or has been returned.

A typical and very common developer process is to open the file, read each line looping through all the rows, and for each row perform a single update without using transactions like:

```
open file
for each line
do
 UPDATE dp8_order
 SET last_shipping_status=?, last_shipping_update=?
 WHERE tracking_code=?;
done
close file
```

As the size of data increases so does the processing time because you execute one statement per row. When there are are 10 packages, 10 SQL statements, when there are 300,000 packages, there are 300,000 SQL statements.

This batch process does not have a user response time requirement like online applications where performance is key to retaining your users. However, while eliminating row at a time processing is critical for providing a better user experience it is also just as important for batch processing.

```
stmt = 'INSERT INTO dp8_batch_tracking (batch_id, tracking_code, status, last_
update) VALUES'
sep = ''

open file
for each line
do
 stmt = stmt + sep + '(42, ?, ?, ?)'
 sep = ','
done
close file

START TRANSACTION;
EXECUTE IMMEDIATE stmt;
UPDATE dp8_order o, dp8_batch_tracking bt
SET o.last_shipping_status=bt.status, o.last_shipping_update=bt.last_update
WHERE bt.batch_id = 42
AND bt.tracking_code = o.tracking_code;
--DELETE FROM batch_tracking WHERE batch_id=42;
COMMIT;
```

This example removes the one query per row problem, and results in just 2 SQL queries for processing the file regardless of size.

NOTE: In MySQL there is a limit to the length of the SQL statement (i.e. The INSERT). This can be adjusted with the max_allowed_packet variable which can be set per SQL statement. If you are processing very large files, the following code would be modified to perform the INSERT for 'n' records, however only a single UPDATE is still required. See DP#10 for an example of using max_allowed_packet.

This example shows just one way to optimize this operation with the least amount of code changes to the existing application. An even better approach is to use the LOAD DATA INFILE syntax to populate the batch table directly. This requires additional SQL privileges and file system access and hence is a more complex solution.

Why is the impact of removing these repeating queries so significant? To answer that question we need to look at the anatomy of the execution of an SQL statement.

SQL statement workflow

To the end user viewing your website with a browser, the result of clicking send on a webpage is a [short] delay before the expected results are displayed or the applicable action occurs. Behind the scenes an extensive amount of work is performed. For anybody that has looked at a waterfall chart showing the response from a web server, there is a far greater complexity for rendering the page you are looking at. The following article gives a good introduction to browser waterfall graphs -- *http://www.webperformancetoday.com/2010/07/09/waterfalls-101/*. While the browser may render 100s of files, it is generally the first response, the actual page that is involved in executing the necessary SQL statements, and the focus of this design practice.

When a HTTP request is made to a web container the application performs a number of operations to satisfy the request and produce a response. With your application, regardless of the programming language, access to the MySQL database is performed by SQL statements. Each statement is passed to the language specific MySQL connector required with your web container. For example, when using the Apache HTTP server and the PHP programming language, the MySQL Native Driver (mysqlnd) is the necessary MySQL Connector. There are connectors for the popular languages including C, C++, Java, .Net, Python, Ruby etc.

Here is a short summarized list of what occurs with all SQL statements.

The application executes an SQL statement.

1. The MySQL client connector accepts the SQL statement then connects across the network to the specified MySQL server and passes the SQL statement to the MySQL server.
2. The MySQL server processes all incoming SQL statements in individual threads, so many SQL statements can be executed concurrently.
3. The MySQL server first parses the SQL statement for valid SQL syntax, and produces a summarized structure of the tables and columns used in the SQL statement.
4. The MySQL server performs a security check to ensure the user that is requesting this SQL statement has the necessary privileges to be able to access/modify the information requested in the SQL statement.
5. The MySQL server then passes the parsed SQL statement to the MySQL query optimizer. This is heart of the decision making process where the cost-based optimizer creates a decision tree, evaluates the various options by pruning the expensive paths to produce the optimal Query Execution Plan (QEP).
6. The MySQL server then passes the QEP to the applicable MySQL storage engine(s) to perform the physical work of storing and/or retrieving the data for the given SQL statement.
7. Depending on the type of query, the MySQL server may have to do additional work, for example to join multiple tables, sort results etc.
8. When the MySQL server has produced the results for the SQL statement, these results are send back across the network to the application server.

NOTE: This is a simplified representation of the execution path of an SQL statement in MySQL. The use of the MySQL Query Cache discussed in QP#9 introduces additional steps and can also produce a significantly simplified and faster execution path.

To summarize, every SQL statement is passed to the MySQL server, the network overhead of points 2 and 9 are the most expensive amount of time in a well tuned MySQL application. This alone is the greatest reason to run less SQL statements.

Every SQL statement is parsed, checked for application permissions and optimized before execution. This is most applicable for example when combining INSERT statements with multiple VALUES clauses. In addition to saving the network round trip, this overhead is also eliminated by combining SQL statements.

Universal Application

This same principle can be applied to other products that process data. For example, memcache is a popular product to improve performance and scalability of your application by providing a memory caching layer. The following figures are for an example benchmark with 28 objects in memcache using two cloud servers in Rackspace Cloud.

Using an individual get call 28 times sequentially in a single PHP file, simulating a client example, the total response time of the benchmarked ranged from 24 to 56 milliseconds. Using the same configuration with a single multi-get call for the same 28 objects the results ranged from 4 to 7 milliseconds.

It does not require a graph to see the **6x-10x** improvement in performance by eliminating row at a time processing. The saving of 20-50 milliseconds may seem small, however when multiplied in environments with thousands of concurrent users, thousands of times per second, has a large impact on resources.

Recap

This principle shows a simple technique for reducing the number of SQL statements by eliminate repeating queries. As a goal of **"Do Less Work",** this is only one case. DP#16 discusses several other query saving techniques that can eliminate repeating and unwanted queries providing improved performance.

More References

- *http://ronaldbradford.com/blog/the-rat-and-thecat-2006-08-24/*
- *http://ronaldbradford.com/blog/optimizing-sql-performance-the-art-of-elimination-2010-07-08/*
- *http://ronaldbradford.com/blog/we-need-morecats-2009-08-22/*
- *http://ronaldbradford.com/blog/simple-lessons-in-improving-scalability-2011-02-16/*

How to set up multi-master replication with Galera Cluster for MySQL

Ashraf Sharif

Database replication is used for a wide range of use cases, ranging from disaster recovery, reporting server, high availability to load balancing for higher performance. It has become an essential part of an IT infrastructure, and can be implemented using different mechanisms. The majority of MySQL and MariaDB users replicate data using the inbuilt MySQL Replication feature, however this has a number of drawbacks.

What you will learn...

- the basics of multi-master replication,
- the caveats of MySQL Replication in multi-master replication,
- how to deploy Galera Replication for MySQL and MariaDB,
- how to deploy HAProxy for SQL Load Balancing.

What you should know...

- basic knowledge of databases, distributed systems and Linux

Replication is one of the most popular features of MySQL. It allows data to be stored on multiple servers, which translates into a number of benefits:

- Scale out the database load and allow applications to handle more users and requests.
- Reduced downtime, since the applications can still access their data if a database server fails or a server needs to be taken offline for maintenance.
- Transparent live data backup.
- Disaster recovery by having a live copy in a different data center.

However, MySQL Replication has it's own drawbacks. Since it is a master-slave architecture, applications need to be aware of which server they can write to (the master), and which server(s) they can read from (the slave(s)). If the master fails, an administrator needs to configure another master and reconfigure applications to point to the new master. Slaves might also need to be reconfigured to read from the new master.

Therefore, a much better approach would be to have a multi-master replication setup, where applications can read and write on any of the MySQL servers.

What is MySQL Multi-Master Replication?

MySQL multi-master replication is a method of data replication which allows data to be stored by a group of MySQL servers, and updated by any member of the group. It works by replicating changes to all the MySQL servers in the group, while transparently solving conflicts if these arise. This guarantees that all nodes have the same data.

Why MySQL Replication is Not Appropriate for Multi-Master Setup?

MySQL Replication does not support locking protocol between master and slave to guarantee the atomicity of a distributed update across two different servers. Issues include:

- Manual failover – even with Global Transaction Identifiers in MySQL 5.6, recovery is easier but not automatic. Failover need to be performed by a skilled administrator.
- Asynchronous – it can not guarantee that your data is replicated, or that replicas are up-to-date.
- Binary log corruption – s crashing master potentially cause corruption of the binary log. The slaves will then unable to resume from the last binary log position when restarted.
- Inconsistency – slaves can end up out of sync with different data to the master. Promoting a slave to a master might incur data loss.
- Changes on application layer – you may need to re-write your application to split reads and writes, and redirecting writes to masters and reads to slaves.
- Single point of failure – if the master goes down, a new master needs to be promoted before the application could write to the database.

Although it is possible to set up circular replication across a ring of MySQL Servers and have updates propagate to all servers, such a setup is easy to break if one of the MySQL Servers go down.

Fortunately for MySQL users, there is some significant amount of innovation happening in the MySQL ecosystem. One of these is Galera replication that provides a multi-master setup.

What Can We Expect from Multi-Master Replication?

We would expect the following functionalities:

- read and write on any MySQL server,
- all nodes are automatically synchronized in real-time,
- strong consistency – all nodes have the update when a transaction is committed,
- a node can fail and resync automatically when coming back online,
- runs in active-active mode.

Introducing Galera Cluster for MySQL

Galera is a synchronous multi-master replication plug-in for InnoDB. It is very different from the regular MySQL Replication, and addresses a number of issues including write conflicts when writing on multiple masters, replication lag and slaves being out of sync with the master.

The Galera plugin is available in a MySQL build that has been patched with the wsrep API, which defines a set of application callbacks and replication library calls necessary to implement synchronous replication of writesets. Galera is developed by Codership Oy (*http://www.codership.com*), and provides synchronous replication guarantees for multi-master replication. Galera's primary focus is data consistency. A transaction is either applied on every node or not at all.

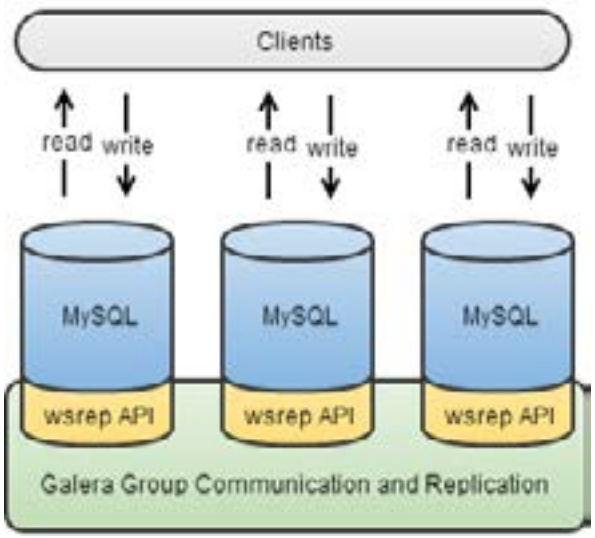


Figure 1. MySQL-wsrep with Galera Implementation

Some of the features of Galera include:

- synchronous replication (no committed transaction is lost, and no slave lag),
- true multi-master (read and write from any node, no read/write split),
- parallel replication of events,
- UDP multicast,
- no Single Point Of Failure,
- automatic node provisioning (new nodes automatically synchronize with the cluster).

Deploy a Multi-Master Setup with Galera Cluster

1. First, make sure that all tables in your database are using the InnoDB storage engine. If some tables are using MyISAM or another storage engine, you will need to convert them to InnoDB. To check your tables' storage engine, you can use following SQL query: SELECT table_name, engine FROM *information_schema.tables* WHERE table_schema =‚your_database_name';
2. We will use the Galera Configurator to automatically deploy Galera. For this example, we will need four nodes, three Galera nodes and one node for ClusterControl, the management server (Figure 2)

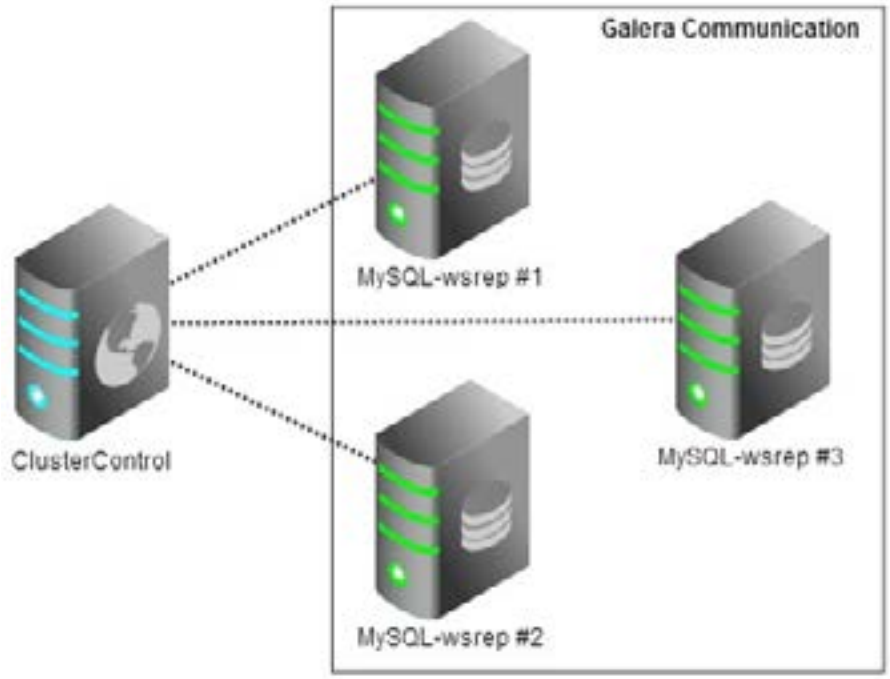


Figure 2. Example architecture of MySQL Galera Cluster with ClusterControl

3. Create the deployment script by going to *http://www.severalnines.com/galera-configurator.* Follow the wizard and generate a deployment package. Copy the download link to the deployment script and ClusterControl API token value.
4. Login to the ClusterControl node, download, extract and execute the deployment script to start the deployment process (Listing 1).

Listing 1: Example of bash commands to initiate the database cluster deployment

```
<<LISTING 1>>
$ wget http://www.severalnines.com/galera-configurator/tmp/
p4luco1r49uo8b5qlmgvsad086/s9s-galera-2.2.0.tar.gz
$ tar -xzf s9s-galera-2.2.0.tar.gz
$ cd s9s-galera-2.2.0/mysql/scripts/install
$ bash ./deploy.sh 2>&1 | tee cc.log
<</LISTING 1>>
```

5. You will be prompted for some input when you start the deployment. The deployment should take between 10 to 20 minutes.
6. Once the deployment is completed, login into ClusterControl at *https://192.168.0.20/clustercontrol* using the email address you provided in the Configurator. The default password is "admin". You will need to register the cluster by entering the address to the ClusterControl REST API, *https://192.168.0.20/cmonapi* and the API token (Figure 3).

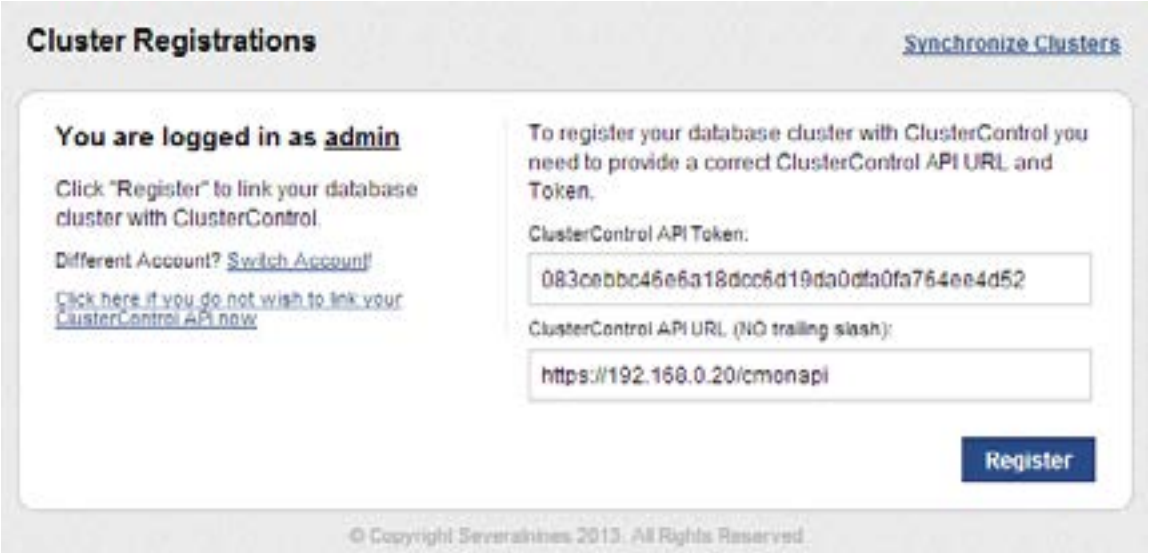


Figure 3. Register a database cluster using API Token

7. Once registered, you should able to see your Galera Cluster for MySQL (Figure 4).

Figure 4. Screenshot of ClusterControl overview page

SQL Load Balancing with HAProxy

There are different ways to provide connectivity to one or more database servers. One way is to use a database driver that supports load-balancing, such as MySQL JDBC driver Connector/J. HAProxy is a pretty neat way to do SQL load balancing, as it provides queuing and throttling of connections towards one or more MySQL Servers and provides overload protection. HAProxy can be deployed on the ClusterControl node, as shown in Figure 5.

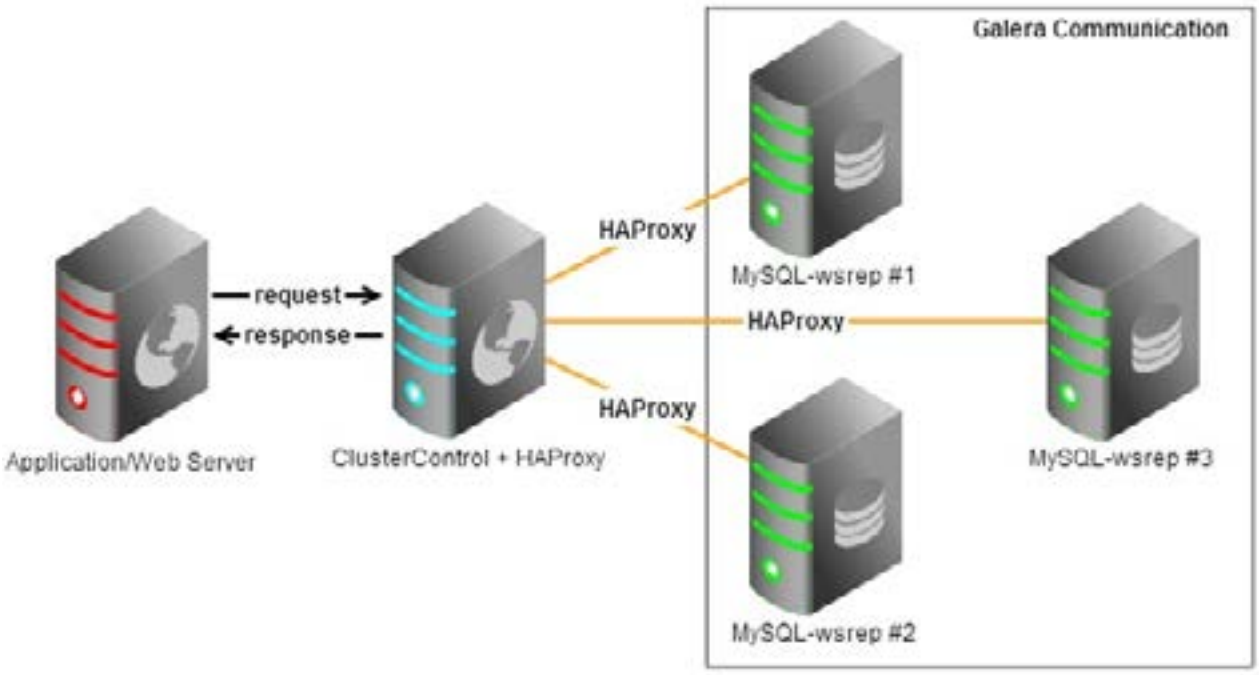


Figure 5. Example architecture of MySQL Galera Cluster with ClusterControl and HAProxy

1. From the ClusterControl node, download and execute the HA-Proxy deployment script from Severalnines Github repository: *https://github.com/severalnines/s9s-admin* (Listing 2).

   Listing 2: Example of bash commands to initiate the HAProxy deployment

```
<<LISTING 2>>
$ git clone https://github.com/severalnines/s9s-admin
$ cd s9s-admin/cluster
$ ./s9s haproxy --install -i 1 -h 192.168.0.20
<</LISTING 2>>
```

2. Check whether HAproxy is listening to the correct ports. MySQL should listen on port 33306 while HAProxy statistics page should listen on port 9600 (Listing 3).

   Listing 3: Verify the listening ports of HAproxy processes

```
<<LISTING 3>>
$ netstat -tulpn | grep haproxy
tcp  0    0 0.0.0.0:3330      0.0.0.0:*     LISTEN     23589/haproxy
tcp  0    0 0.0.0.0:9600      0.0.0.0:*     LISTEN     23589/haproxy
<</LISTING 3>>
```

3. Grant privileges for database user from the application side:
   GRANT SELECT, UPDATE, INSERT, DELETE ON database_name.* TO 'user'@'192.168.0.20' IDENTIFIED BY 'password';

4. Your application can now access to a single IP to read/write into database. We can simulate this by connecting to MySQL Galera Cluster via HAProxy:
   $ mysql -h 192.168.0.20 -P 33306 -u user -p
   The HAProxy statistics page can be accessed at port 9600 through web browser as shown in Figure 5.

Figure 5. Screenshot of HAProxy statistics page

Congratulations, you now have a fully load-balanced multi-master MySQL setup.

Summary

A multi-master setup addresses many shortcomings of a regular master-slave setup with MySQL Replication. Galera Replication is an elegant way of achieving that. All nodes are always synchronized, fail-over is automatic, and applications can read and write from any of the nodes.
to continue…

Additional Resources

- *http://www.codership.com/content/using-galera-cluster* - Galera documentation,
- *http://www.severalnines.com/galera-configurator/* - Galera configurator,
- *http://www.severalnines.com/clustercontrol-mysql-galera-tutorial* - Tutorial on Galera Cluster for MySQL,
- *http://haproxy.1wt.eu/* - HAProxy homepage,
- *http://www.severalnines.com/resources/clustercontrol-mysql-haproxy-load-balancing-tutorial* - Tutorial on SQL Load Balancing with HAProxy.

About the Author

ASHRAF SHARIF

Ashraf is a systems engineer at Severalnines, where he helps build automation and management systems for database clusters.

He previously worked as a senior Linux system administrator, with a specialisation on automation and security in hosting environments.

HI, MY NAME IS KEVIN.

WORDPRESS

WEBSITE DESIGN & DEVELOPMENT
IS WHAT I DO.

www.anythinggraphic.net

wordpress • design • development • e-commerce • seo + more

ANYTHINGGRAPHIC

anythinggraphic.net (505) 975-6684 @anythinggraphic

Software Developer's JOURNAL
new ideas & solutions for professional programmers

Matthew Rupert – extra ebook

Table of Contents

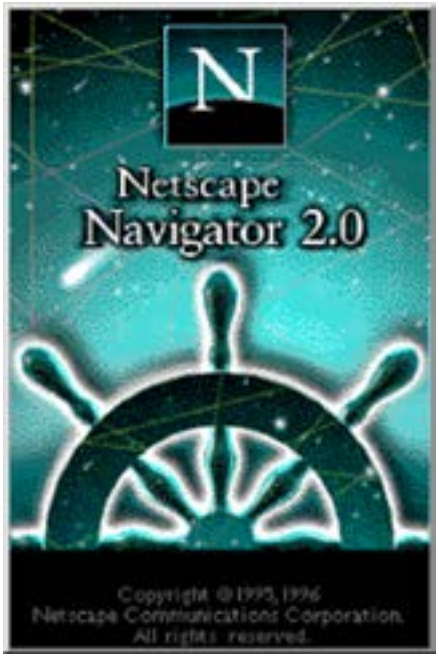
A Brief History of The Web (Very Brief) 129
WordPress From Scratch 131
WordPress.com vs. WordPress.org 133
Design 134
Hosting Providers 135
Before Starting 135
Installation 138
SSH/Command Line 142
Downloading Wordpress Manually 143
WordPress Installation 145
Apache/Web Server Configuration 147
MySQL Database 148
Installing WordPress using cPanel 150
Initial WordPress Configuration 151
WordPress Codex 153
Further WordPress Configuration 154
Themes 155
Importing an Existing Blog 160
Plugins 162
Important Plugins: Links 165
Permalinks 167
The Content: Posts and Pages 169
That's That! 174
Additional Recommended Reading (Just For Fun) 175
Links 175

Software Developer's JOURNAL
new ideas & solutions for professional programmers

A Brief History of The Web (Very Brief)

Not long ago I cam across an article about "The World's First Website." Without going into any debate about what exactly that means, or whether or not British physicist Tim Berners-Lee can be fully credited for inventing the World Wide Web at CERN in 1989, this much is true: The early days of the World Wide Web were, well, how should I put this? A bit static.

By *static* I mean a couple of things. There was a palpable excitement about it. The first time I ever viewed a website it was on a text-only browser called *Lynx*. The content was dynamic only insofar as it had been created at some point and uploaded to a web sever. I didn't realize that what I was viewing was being fed to my **VAX/VMS** terminal window was being served by something called web server, or that the web server was very likely **CERN httpd**. I didn't mind that the pages didn't change from day-to-day. Nor did I mind that the websites weren't pretty. For the time being, I was sufficiently fascinated by the endless links from topic to topic.



World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

The First Web Page at CERN

In retrospect, those early days of the World Wide Web moved by pretty quick. Soon we had things like Slipknot, Mozilla, and Netscape (or should I say *Netscape Navigator*?), to display web pages in increasingly flashy ways. Soon it seemed everyone had a website—and most of those websites were, well, awful. Using free hosing sites like Tripod, Angelfire, Earthlink, and The Globe. With anyone and everyone creating websites, we became accustomed to an overload of wasted web space. Remember the centered flashing red text? Remember the slow-loading GIFs? Remember the obligatory under construction images? YUCK!

Blogging became a trend around 2002, and it wasn't long before multitudes of what amounted to online journals full of amateur political commentary or teen angst exploded. With the advent of Twitter and Facebook, those who desired little more than public announcement of personal opinions found more 'appropriate'

outlets. Blogging survived, still survives (thrives, even), but only the best blogs receive significant traffic. WordPress and Blogger were two of the early blogging tools. Blogger, purchased by Google, has floundered. Sites created by Blogger stand out like a broken URL. WordPress, on the other hand, has grown and adapted. These days it is unfair to call WordPress a blogging tool. It can be just that, but its much more. It's a full functioning content management system. It's an expandable, customizable, scalable foundation upon which a vast range of websites are built.

Here's the bad news: The bad sites are still out there, and more of them are being created every day. The good news? The WWW audience has become more shrewd, with higher expectations of appearance and content. The even better news? WordPress makes it easy to create a foundation upon which those with something to share can do so in a way that contributes and does not detract.

WordPress From Scratch

About a week ago I got a call from a friend of mine who is looking to create a website for his small business. This friend is a small business owner—a contractor who worked on upgrades and additions to homes. "Archie," I said, "the best way to go about this is to let me set up a WordPress site for you. Once it's set up, I think you'll find it is extremely easy for you to maintain yourself."

"Isn't WordPress for blogging?" he asked.

It used to be it. It certainly still is. However, WordPress has evolved tremendously from its initial release in 2003. With thousands of templates, plugins, and easy customizability, WordPress has become one of the single best ways for businesses, small to large, to host their web presence. Don't think of WordPress as a blogging tool. Think of it, instead, as a powerful website creation tool, with powerful CMS, administration, feedback, sharing, and expansion capabilities.

So while WordPress has become a robust, powerful solution for websites of all kinds (see the WordPress showcase at http://wordpress.org/showcase/ to see a surprising and eclectic range of sites using WordPress), it remains a powerful blogging tool as well. Sure, some out there may suggest "blogging is soooo 2006," but

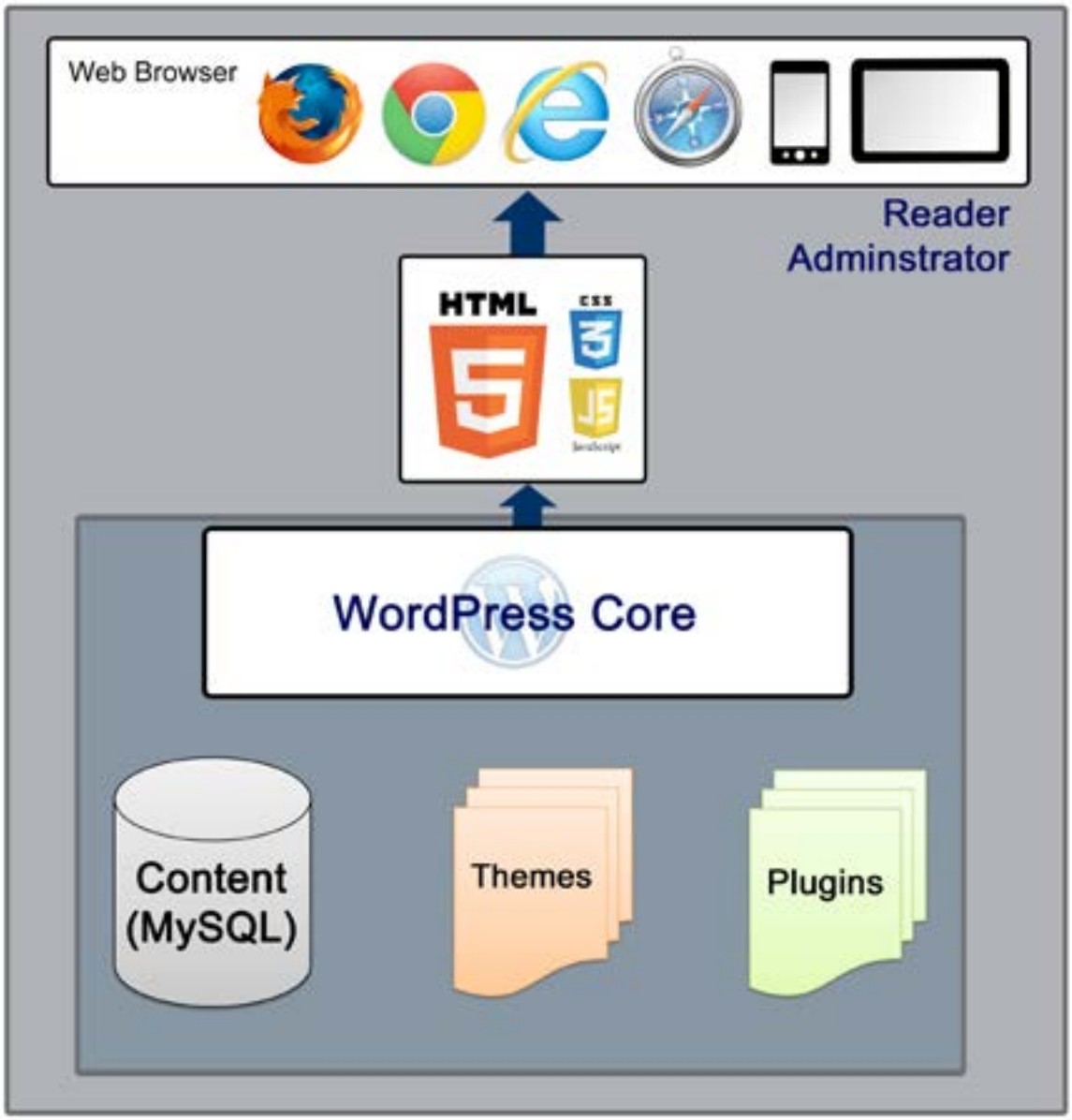
as for me, I remain an avid blogger and blog reader. The change in blogging over the years, with the growth of Facebook, Twitter and Google+, has been a good one. Those who created a blog simply as a way of keeping in touch with family or talking about themselves have found a voice in a more appropriate forum. Many of the remaining, most popular blogs, are those with original and interesting content. If I may editorialize for a moment, I suggest that before you start blogging, and even if you are continuing a history of blogging, strive to create content. Anyone can post links. A great blogger contributes!

What I personally love about Wordpress is this: It passes both the mom test and the uber-geek test. By *mom test*, I mean this: Could my mom use it? Yes (sorry mom!). By *uber-geek test*, I mean that it is open and configurable enough that even the most demanding power user will find satisfaction in the level of control granted by WordPress.

Due to its popularity, a key advantage is that the plugin architecture continues to grow. And with templates that are highly customizable, most of them free, it is indeed very easy for non-techies to administer their own sites. That said, for a self-hosted WordPress site, there is a bit of technical expertise required to get things up and running. With many providers, perhaps most, offering WordPress hosting, setup of a custom installation can be as easy as a few clicks. And for those running their own servers, given the knowledge required to do such a thing, installation of WordPress will seem quite simple.

It is necessary to clarify some terms here. WordPress is software. That is, WordPress, used generically, refers to all of the server-side PHP implementation that creates the functionality for management of all the blog/site settings, from posts to pages to plugins to settings. WordPress itself is not an ISP or hosting solution of any kind. WordPress is all the "stuff" that organizes and present content.

WordPress is developed by Automattic, Inc., a web development company that provides a number of other services, most of which tie-in nicely with it's flagship product. It is released under the GNU General Public License, meaning that all the code behind it is open source, and can be contributed to by anyone (anyone with the necessary skills, that is). One of the most popular blogging/CMS/website creation tools around, it is capable of handling very large amounts of data. It scales well enough for a small blog to a large news site.



WordPress.com vs. WordPress.org

When it comes to starting a WordPress site, there are two major options to consider: self-hosting or *wordpress.com* hosting. Each has its benefits and drawbacks, but self-hosting (either on your own server or through a hosting provider) provides the greatest amount of freedom. How to approach hosting is a question only you can answer. It's all a matter of cost, freedom of control, and technical ability/confidence.

Wordpress.com hosting may be a good option for many, but it is with limitations. In the case of a *wordpress.com* hosted site, available themes, plugins, and customizations are limited. A user cannot implement or install custom plugins, and tailoring is limited to that which is provided by *WordPress.com*. Still, it can be a great option for a number of reasons. First and foremost: It's free! This means that some users will see ads on your page (*free* isn't always ***free***).

It's simple and easy, and one need not pay for additional hosting. If you're willing to spend a few bucks, there are custom themes, fonts, colors, and domain name customization. *WordPress.com* hosting is easy, fast, stable, and secure. You'll never have to concern yourself with backups or scramble if your site gets hacked. All such security and damage control is handled by *WordPress.com*.

As a true business front or for hosting a personal site wherein one desires great customization, the limitations are likely more than such a customer desires. For this article, I am focusing on the setup of a self-hosted *wordpress.com* site.

Before continuing, note that there have been many updates to WordPress since it's initial release in 2003. For this article, WordPress 3.5+ is discussed.

Design

Web pages and style preferences are similar in many ways to fashion trends—What is considered sleek looking now will be considered dull (or downright ugly) in a year or two.

If you're setting up a website for a customer, this is easily overcome by simple switching of templates. New templates are always being created, and people spend many hours tailoring and testing them. Any web designer who has struggled with CSS and worked long hours to make sure a site renders correctly across multiple platforms and browsers will understand just how tedious this can be.

Because WordPress is so widely used, there is no shortage of new, trendy looking templates to choose from. Sure, you can spend hours upon hours creating your own (and some people, myself included, consider this *fun*), but starting with a great theme and customizing it a little here and a little there is a much safer, more timesaving approach.

Even for those who prefer to work start from scratch, there are a number of barebones templates available, created with the intent of laying out the basics (including specific WordPress hooks and tags). For the brave, those who prefer to go it alone and create something, I most definitely recommend starting from a barebones template.

Back to the shortcomings of a *WordPress.com* hosted site: With limited configurability and a limited number of template to choose from, your ability to create something entirely unique is very much limited. Some of the templates (many of the best ones) cost (anywhere from $50-120 per year), as does CSS, font, and color configurability. So if you do go the *WordPress.com* hosted route, be aware that viewers will likely stumble upon other *WordPress.com* sites that appear similar to your own.

That said, since it's free, it may not be a bad idea to get started and to understand WordPress is to set up a test blog for free at *WordPress.com* (assuming you have no prior experience with WordPress). If you do this, please be a good *web citizen*: Creation of a *WordPress.com* site ties up a name (e.g. *flimflam.wordpress.com*) that someone else may actually want to use. So be sure to remove the site and the blog so that others are not blocked from using it. It is frustrating when attempting to use a domain name or a blog name that is tied up by an unused or temporary site.

To sum up what I've stated above, there are really three options:

- *WordPress.com* hosting
- A paid hosting provider
- Self-hosting (Your own IP Address, server, domain name, installation, and configuration of both WordPress and the server.)

Hosting Providers

WordPress.org presents a few options to users in need of hosting. (For more details see *http://wordpress.org/hosting/*). For further details on the differences between *WordPress.com* and *WordPress.org*, see *http://en.support.wordpress.com/com*-vs-org/.

The remainder of this article is about setting up a self-hosted WordPress site. After setting up a WordPress self-hosted site, the most obvious immediate difference will be the appearance of the left-menu. Menu options for Upgrades, Ratings, Polls, and certain plugins will not be immediately available in the *WordPress.org* (self-hosted) dashboard.

Before Starting

If installing WordPress on a Windows server, there will be a few additional requirements, including installation and configuration of Apache, MySQL, and PHP. With so many differing server environments, from Mac to Windows to all of the various Linux distributions, environment needs are configured. The most simple installations will require some knowledge of hosting environments (assuming you intend on fully hosting yourself), including configuration of PHP, MySQL, and Apache. In a self-hosted setup, you will want to be well aware of security considerations. The overall configuration of Apache Web Server is beyond the scope of this article. Although I will touch on a few points, if self-hosting (not using a subscription hosting provider), take note that experience in web configuration and security is necessary.

Although any web server that supports MySQL and PHP will work, WordPress recommends either Apache (as do I) or Nginx.

Minimal WordPress installation requirements are:

- HP 5.2.4 or greater
- MySQL 5.0 or greater
- The mod_rewrite Apache module

In this article I will assume Apache HTTP Server is used (as is commonly the case).

PHP

WordPress and it's plugins are written in PHP. If you are a software developer, it is likely that you have at least some knowledge what PHP is. In case you do not, PHP is simply an interpreted scripting language used in many LAMP systems. LAMP is an acronym for common web applications that use Linux, Apache HTTP Server, MySQL database (although PostgreSQL is also commonly used), and PHP, Perl, or Python for backend serviced and rendering HTML. By this definition, WordPress is a LAMP application.

Definition

LAMP is a combination of free, open source software. The acronym LAMP refers to the first letters of Linux (operating system), Apache HTTP Server, MySQL (database software), and PHP, Perl or Python, principal components to build a viable general purpose web server.

Wikipedia.org, https://en.wikipedia.org/wiki/LAMP_(software_bundle)

While it is not necessary to know PHP, you will likely see some PHP code here and there. This is particularly true if you venture into editing the appearance of your theme beyond simple CSS modifications, modify, or write a plugin. Syntactically, PHP may seem foreign to those without prior experience, but it not a difficult language for a programmer to understand.

JavaScript

Modification of self-hosted *WordPress.org* sites will not require an expert level of JavaScript understanding. However, as with PHP, a number of bells and whistles can be added with JavaScript. (*WordPress.com* hosted blogs prevent inclusion of JavaScript enhancements.) This is another major benefit of using a hosted *WordPress.org* site (along with the ability to include jQuery or other JavaScript libraries).

Graphics

The default WordPress themes generally come with a canned set of graphics for use as headers, backgrounds, buttons, and separators. While it may be acceptable at first to use the default content, with so many other sites using similar themes, I encourage you to create, minimally, a custom header. Most themes will include a description of recommended header dimensions. I generally use Photoshop for my needs, but GIMP ("GNU Image Manipulation Program") is a powerful free alternative (*www.gimp.org*). Even if you are building a site for a customer, you will find it necessary to modify provided images to the necessary dimensions. As a sidebar, I don't recommend using an image that is too small for a particular area. For example, if a recommended header size is 800x600 pixels, a 200x100 image stretched to fit the area will look awful (not to mention unprofessional).

For buttons and headers, when customized, I prefer using the PNG format, which provides the compression and clarity of JPG, but allows transparent backgrounds.

Definition

Portable Network Graphics (PNG) *is a raster graphics file format that supports lossless data compression. PNG was created as an improved, non-patented replacement for Graphics Interchange Format (GIF), and is the most used lossless image compression format on the World Wide Web.*

Web Browsers

While modifying your WordPress site, always be sure to verify appearances in all of the most popular browsers, not simply your personal favorite. As I watch the hit statistics on my blog (*matthewrupert.net*), I've noticed a sharp increase in the number of users coming from mobile clients and Google Chrome. Perhaps because my blog is technical in nature, most readers are using either Google Chrome or Firefox. There are surprisingly few using Internet Explorer.

I always test my web pages on Chrome, Firefox, Internet Explorer, and Safari. If you do not happen to have a Mac available for testing, take comfort in knowing that if your site looks the way you want it to in Chrome and Firefox, it will likely render appropriately on Safari.

Newer WordPress themes have settings for mobile browsers, which may be enabled or disabled. I recommend enabling mobile rendering whenever possible. Think about your personal website reading. I'm willing to bet that you often use Android or iOS on a phone or tablet.

Installation

Guided vs. Non-Guided Installation

Assuming your hosting provider does not provide an interface for installation (e.g. cPanel), you'll need to know your way around the command line at least a bit. If your hosting provider does provide a cPanel interface, or something similar, chances are there is already a very easy WordPress installation path. In either case, after WordPress is installed, you'll want to configure some things. Installation through a hosted interface certainly simplifies the first few steps, and if this is your situation, skip ahead. Unless you are installing WordPress on your own server, changes are Apache is already installed.

As long as your server has PHP and MySQL installed, the platform matters little (although the vast majority of WordPress installations are hosted on Linux). Because there are so many possible configurations and hosting provider setups, there is no straightforward answers to some of the questions that may arise. Throughout this guide I will attempt to answer those questions as much as possible. Be aware that things like PHP installation, MySQL installation, database setup, and web server configuration are highly dependent on many variables.

That said, even if it is necessary to install everything from Apache to PHP, a knowledgeable software/administrator will find installation to be straightforward (although a bit longer than the "famous 5 minutes" that WordPress advertises—particularly if you are setting everything up from scratch). Even the database setup is easy. All you need to do as the installer is create an empty database with permissions for your WordPress account to access it. For anyone with limited operating system and web administration experience, I don't recommend attempting to install "from scratch" without first learning a few things about web hosting.

If installing our your own server, there's no reason you can't install WordPress locally on a laptop if you wish, just to figure things out before attempting a *production* install. All the tools needed, MySQL, PHP, and Apache, are all capable of running on nearly any OS. This isn't a bad idea, as you may find it necessary to tinker for a while before attempting to real thing.

Installing Using cPanel/Hosting Provider Services

My hunch is that those with limited technical knowledge are willing to pay a hosting provider, and hosting providers generally include web interfaces (such as cPanel) that make all of this simple. These web-hosting control panels, along with *autoinstallers* such as **Softaculous**, **Fantastico**, and **Installatron**, remove the difficult

parts. Some hosts provide multiple autoinstallers, and these programs often include all the database creation steps and domain mapping.

The cPanel software package a control panel commonly used by hosting providers. This software, and other packages like it, gives website owners easy ways to manage their websites and services. Installing WordPress, in this case, is very simple, especially is your MySQL user database is already configured (and it probably is). In this case, your host will provide an installation program such as **Installatron** or **Softaculous**.

In general, a **cPanel** installation of WordPress includes the following steps:

1. Log in to your cPanel. (this is typically something like https://<my-host-provider>/cpanel)
2. Use **MySQL Database Wizard** (These database steps may not be necessary. Your hosting provider may have already created your database. All you need to do is provide a WordPress table prefix and database access credentials.)
3. **Create a Database:** enter the database name and click Next Step.
4. **Create Database Users:** enter the database username and the password. Note that this is the database username and password—not the WordPress administrator username and password, which will be set up later.
5. **Add User to Database**: Select All Privileges checkbox before moving on to the next step.
6. **Complete the Database Setup:** Be sure to take note of the database name and user. There is nothing as frustrating as a forgotten hostname, username, databasename, or password. For this database, hostname will likely be localhost (127.0.0.1), as it is accessed by the server locally, not via some external path.

To set up the database, the installation process will request an administrator username, password and WordPress table prefixes. These table prefixes are used to distinguish WordPress tables from other tables (assuming the user has only one database schema per account).

Other hosting providers may allow for multiple MySQL databases, in which case it will be necessary to choose a database as well as WordPress table prefixes. Since there is no single approach among the many providers

Installation to Using Existing Database on Hosting Provider

If your hosting provider already provides a MySQL database created for each user (as many do), and you are not using a automatic installer (such as Softaculus), it is a good idea to check the *wp-config.php* file

to gather this information about the WordPress installastion. This step is necessary for the MySQL configuration settings required later. There will be a default administatory username and password in the *wp-confing.php* file, so take note of this information. When you have WordPress up and running, one of the first things you will want to do is change the admin password.

MySQL Installation (Self-Hosting or Hosting Provider without MySQL)

First things first, MySQL must be installed. MySQL, a popular choice of database for use in web applications and LAMP applications (such as WordPress, provides a powerful, free (it is open source) database. WAMP and MAMP are not unheard of acronyms (Windows, Linux, Apache, PHP/Perl and Mac, Linux, Apache, PHP/Perl), and as long as the configuration is correct, WordPress can be served from Windows and Mac OSX without issue.

In the unlikely scenario that are not self-hosting and your hosting provider does not provide MySQL, it will be necessary to find a new hosting provider or work with the provider to have MySQL set up for your account. If you are self-hosting, and in complete control of your server environment, fear not: Installing and configuring MySQL is straightforward.

Installing MySQL, assuming your system does not already have it available, can be just a bit confusing, as there are several editions to choose from. MySQL Enterprise Edition and MySQL Cluster CGE are commercial packages. Although MySQL is open source, these particular editions are meant for businesses. They cost, and provide much more functionality and support than what most WordPress users require. We want the MySQL Community Edition (licensed under GPL). This can be downloaded from *http://dev.mysql.com/downloads*. Here you'll see a number of other MySQL tools that are also available for free under the GPL, but for now, all that is really necessary is MySQL Community Server.

MySQL can be extracted from a tar file (on all operating systems) or as a native installer (on some operating systems). When possible, a native installer is the simplest approach.

Note: With MySQL now being "owned" and hosted by Oracle, it is necessary to create an account to download even the free edition. If downloading directly from MySQL/Oracle, the downloaded file is a zip or a tar.gz file. This means that installation will include expanding the contents in a desired location. For those on non-Linux systems without package management utilities such as yum and apt-get, installers and/or tar bundles are available at http://dev.mysql.com/downloads/mysql.

Only once instance of MySQL server is required, as multiple WordPress sites can utilize schemas on the same MySQL database. The only requirement here is that each WordPress site has unique table prefixes to distinguish the MySQL database table names of each installation of WordPress. This is very easy to do—In fact, WordPress installation prompts for a database table extension during setup.

Most hosting providers provide a single database instance for their customers, and this is not a problem whatsoever for hosting multiple WordPress sites. In fact, a number of different applications may use that database. Not to worry! WordPress will know which tables belong to it.

RedHat/Fedora

With RedLinux (RedHat Enterprise, Fedora, CentOS), installation is as easy as using the yum installer:

```
sudo yum install mysql-server mysql
```

If you are a RedHat user, chances are you're already quite familiar with yum. Yum is the RedHat software package manager, used for installation and updates (as well as removal) of packages on all RPM-based systems. The nice thing about installing via yum is that all dependences can easily be included in a single command. The entirety of WordPress cannot be installed via yum, but php, MySql, and Apache HTTP can be installed with the single line:

```
yum install mysql-server php httpd
```

Ubuntu

Ready for this? You're gonna like this! Everything you need to do to install WordPress on Ubuntu is as follows:

sudo apt-get install wordpress
sudo ln -s /usr/share/wordpress /var/www/wordpress
sudo apt-get install mysql-server
sudo bash /usr/share/doc/wordpress/examples/setup-mysql -n wordpress localhost
sudo /etc/init.d/apache2 restart

The second line above simply creates a symbolic link from /var/www/wordpress (the place Apache serves files from) to the WordPress installed path. In the third step, while installing MySQL, you'll be prompted for an administrator/root password. Enter one, and don't forget it!

The "setup-mysql" step creates the necessary MySQL WordPress tables (and similar steps can be taken on other platforms).

Ready for this? Open up a browser and navigate to sudo bash /usr/share/doc/wordpress/examples/setup-mysql -n wordpress localhost

Voila! Boom goes the dynamite! There you have it! There on that page you see a form with some things that need to be entered before your WordPress site is actually a *real site*, but for all intents and purposes, the heavy lifting is done. To be clear, this isn't what one might consider a complete installation. In this example WordPress is running as root, and therefore not servable to the outside world. There are a number of Apache owner and group configuration steps required here that, while not overly complex, run beyond the scope of this article.

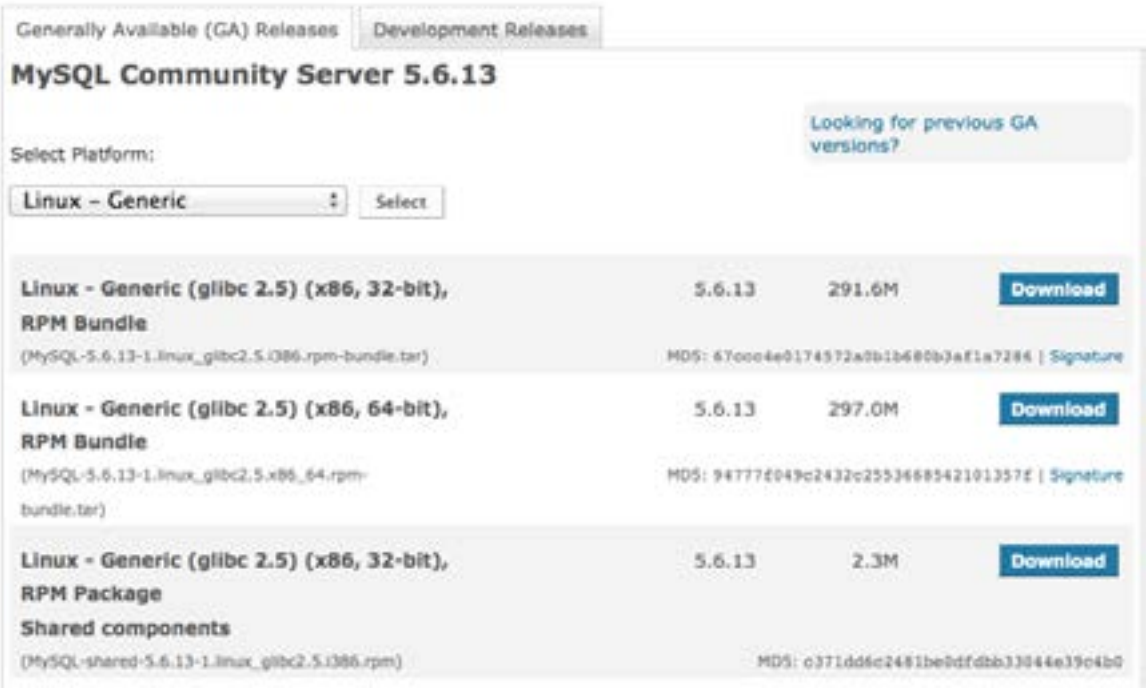
Microsoft Windows

A MySQL Installer for Windows (MSI) can be found at *http://dev.mysql.com/downloads/installer*.

Mac OS

MySQL provides a native dmg installer for Mac OS. See the MySQL downloads page.

Other Operating Systems



If using an OS for which there is no native MySQL installer available, it will be necessary to download and extract the MySQL application from a *tar.gz* file.

Now that MySQL has been installed, we won't worry about it for a bit (ore on this later). The next step is putting the WordPress core files in place.

SSH/Command Line

While the installation tools provide the capability of installation without ever touching the command line, those installing on their own servers, or using a server with SSH available, will find that knowledge of using the command line is necessary.

FTP Client

WordPress settings can usually be modified using the existing administrator pages. Even custom themes can be uploaded directly through the user interface. If you prefer to modify the files directly (rather than making changes locally, uploading, and testing), a good FTP client might speed the process. See *http://codex.wordpress.org/FTP_Clients*.

Downloading Wordpress Manually

The current version of WordPress (3.5.2 at the time of this article) can be downloaded from *http://wordpress.org/download/* as a *.tar.gz* or .zip file. WordPress touts a an easy "5-minute installation," and, assuming other requirements are already set up (hosting with MySQL available), this really is the case. There are ways to run WordPress with other databases, but MySQL is the standard. If the previous steps (using apt-get, yum, or another platform specific installer are not satisfactory), manual download and extraction is easy enough.

1. Download and Extract WordPress
   The first step is to download WordPress. Assuming you're working in a Linux environment (or on a Mac), grab the ***.tar.gz*** file. A .zip file is also available. Obviously you can grab it through your favorite web browser, but since we're going to be working at the command line anyway, it's just as easy to use curl or **wget** to retrieve the latest WordPress build. Fortunately, the folks have

   WordPress have made this very simple. Prior to downloading, go ahead and change to the directory in which you wish to install WordPress.

   Depending on your Apache configuration (see DocumentRoot in *httpd.conf*), this location will vary. In most Linux distributions it is /var/www (but don't assume this).

   Depending on your particular installation, and assuming this is a server that you have some degree of control over, the path to install depends a lot on personal preference. Do you plan on hosting *only* this WordPress site? If so, you may want to install directly in the Apache DocumentRoot.

   Chances are, however, that you want to install WordPress in a subdirectory. I recommend using a subdirectory, as it is much easier to configure Apache to point to another path than it is to move WordPress after it has been installed. This is certainly true if you have existing web content at the root level.

   Should you attempt to put WordPress's *index.php* and the same directory level of an existing *index.php*, chances are the results won't be your desired goal. If what you have at that

level is old content that you wish to get rid of, remove it first. If you instead want WordPress to be available at the same DocumentRoot (both the existing *index.html* and WordPress's *index.php*), you'll need to make a decision now about what you want available and how you want it to be accessed. It could be that Apache first reads PHP files when searching for an index. In this case, when a client hits http://<my-url> , your WordPress will be displayed. This would require clients wishing to hit the old *index.html* to explicitly enter *http://my-url/index.html* in the browser. This is not a recommend approach. Instead consider installing WordPress in a subdirectory (e.g., /var/www/html/wp). By installing in this manner, users (as well as site links from whatever your existing content is) can access your WordPress site in a consistent, expected way. This also enables better domain and subdomain mapping within your Apache configuration.

So here are my initial steps:

```
cd /var/www/html/
sudo mkdir wp
sudo wget http://wordpress.org/latest.tar.gz
or
sudo curl -O http://wordpress.org/latest.tar.gz
```

(For the sake of this article, I am skipping over explanation of sudo and other Unix commands. If you are uncomfortable with the above, and attempting to install WordPress on your own server, it is highly recommended that you first understand the basics.)

2. Set permissions
   The typical Apache installation is set up with User and Group Apache. Other systems may use www or _www. If unsure, just check the *httpd.conf* file and search for the User and Group settings:

```
User Apache
Group Apache
```

   In the previous step, while extracting the *.tar.gz* file, the WordPress directory, and all the files within were extract with the current user's Group and current user as the the owner. This needs to be corrected for Apache. The Group and Owner. To change this to something that Apache has the correct read/write permissions, we simply need to change the Group and Owner. To do so recursively:

```
cd /var/www/html
chown –R apache:apache wp
```

The above command changes the owner and group of each of the files and subdirectories in the wp directory to apache:apache. It may be possible to run WordPress without setting these permissions, but it is proper and most secure to have the correct Apache owner and group settings so that we need not worry about access permissions.

In the above steps, I assumed WordPress would be installed in the webroot in a /wp subdirectory. This may or may not be what you wish to do. WordPress can be placed in a subdirectory of wwwroot, in which case default user access is:

http://<url>/wp/

DNS mapping, virtual hosts, or Apache redirects can always be used to redirect from http://<url>/wp to http://<url>. This is my preferred approach, as it offers greater flexibility as a web administrator later. You may wish to configure a subdomain by way of Apache so that your WordPress site is accessed like so:

http://<my-site-name>.<my-domain-name>

Plugins will be addressed later, but it is worth noting here that there is a great plugin available for mapping various categories, post types, and pages to different subdomains: WP Subdomains: *http://wordpress.org/plugins/wp-subdomains-revisited/*

WordPress Installation

Now that WordPress has been extracted into your Apache served directory, you may be able to perform a quick test to see that the server is working by viewing *http://localhost/wp* in a browser. Here you'll see a list of the installed WordPress files (assuming Apache is configured to allow viewing of a directory index through mod_autoindex). The missing link here is the URL redirect and/or the document root for Apache. To be clear, this won't be the case if you did the more simple cPanel or easy installation steps including the configuration of MySQL using /usr/share/doc/wordpress/examples/setup-mysql -n wordpress localhost.

If, however, MySQL and/or PHP have yet to be configured, the following steps are necessary.

DocumentRoot, the Apache *httpd.conf* configuration setting is set by default in many Apache installations to be the root of your user web part (for example, /var/www or /user/var/www, or something similar). With full control of Apache, there are several approaches that can be taken to site redirection. If you intend only to serve your WordPress site, DocumentRoot can point to /var/www (for example). A more flexible solution is to utilize DNS redirects or VirtualHosts so that you are not limited to only hosting a single site. This is all a matter of Apache configuration, the details of which go well beyond the scope of this article.

At the root level of the served WordPress path (the DocumentRoot or the path that is redirected to when your domain is accessed), an *index.php* file appears (this already exists in the /wp directory that was created during WordPress extraction). This is the php file that redirects to the WordPress site. It is necessary that your web server is properly configured to automatically read *index.php* files upon access, and that no other index files compete with it (i.e., there should be only one index file at the DocumentRoot).

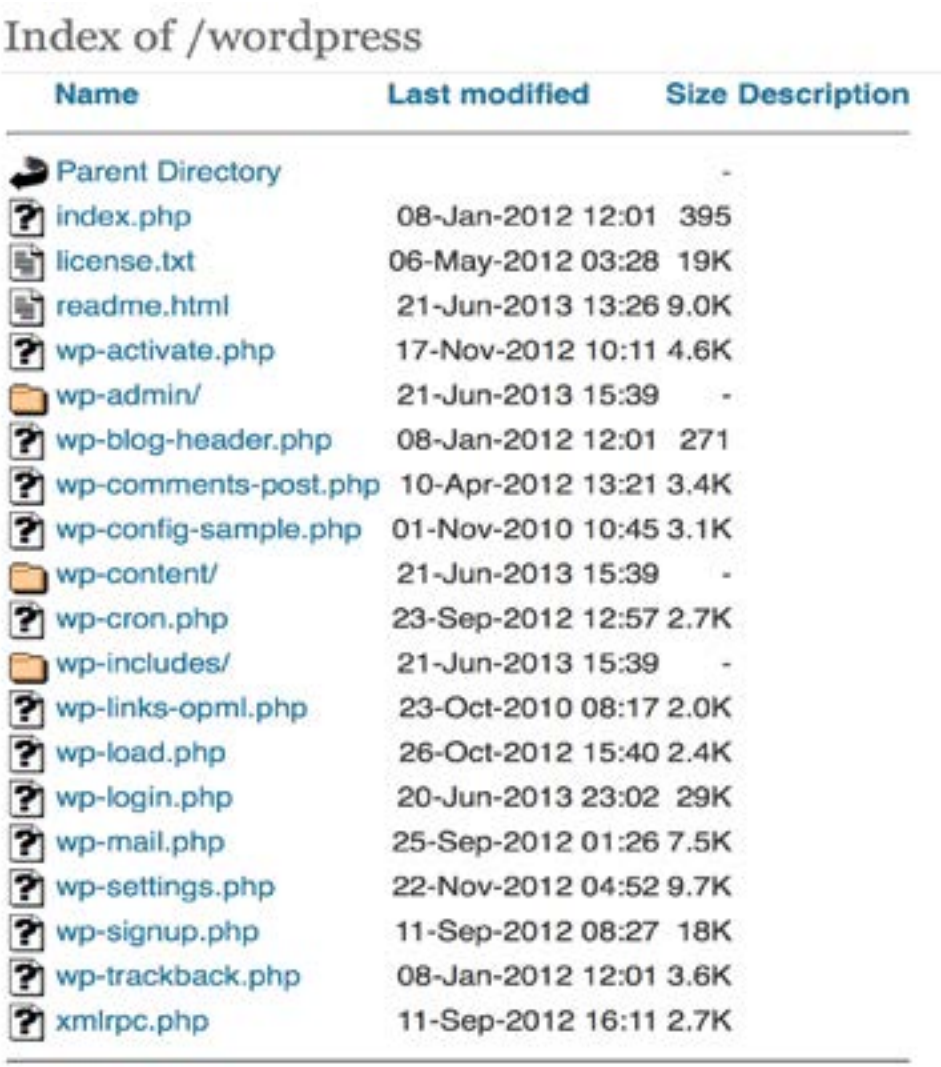
The contents of this file are as follows:

```
<?php
/**
 * Front to the WordPress application. This file
doesn't do anything, but loads
 * wp-blog-header.php which does and tells
WordPress to load the theme.
 *
 * @package WordPress
 */

/**
 * Tells WordPress to load the WordPress theme
and output it.
 *
 * @var bool
 */
define('WP_USE_THEMES', true);

/** Loads the WordPress Environment and
Template */
require('./wp-blog-header.php');

Of course, MySQL has has been installed, so
there may still be a few configuaton steps. so
you'll only see a listing of files (depending on
Apache configuration, you may get a Forbidden
error).
```


Index of /wordpress

Name	Last modified	Size	Description
Parent Directory		-	
index.php	08-Jan-2012 12:01	395	
license.txt	06-May-2012 03:28	19K	
readme.html	21-Jun-2013 13:26	9.0K	
wp-activate.php	17-Nov-2012 10:11	4.6K	
wp-admin/	21-Jun-2013 15:39	-	
wp-blog-header.php	08-Jan-2012 12:01	271	
wp-comments-post.php	10-Apr-2012 13:21	3.4K	
wp-config-sample.php	01-Nov-2010 10:45	3.1K	
wp-content/	21-Jun-2013 15:39	-	
wp-cron.php	23-Sep-2012 12:57	2.7K	
wp-includes/	21-Jun-2013 15:39	-	
wp-links-opml.php	23-Oct-2010 08:17	2.0K	
wp-load.php	26-Oct-2012 15:40	2.4K	
wp-login.php	20-Jun-2013 23:02	29K	
wp-mail.php	25-Sep-2012 01:26	7.5K	
wp-settings.php	22-Nov-2012 04:52	9.7K	
wp-signup.php	11-Sep-2012 08:27	18K	
wp-trackback.php	08-Jan-2012 12:01	3.6K	
xmlrpc.php	11-Sep-2012 16:11	2.7K	

Notice that there are three main directories in the WordPress extract: wp-admin, wp-content, and wp-includes. The files within wp-admin and wp-includes are, effectively, the core php files. The wp-content directory is where custom files, themes, plugins, and other media are placed. Expect the content of this directory to grow as your WordPress site is built and modified. Presentation code, that which is presented through the web browser (posts, pages, categories, links, comments, etc.), is generated dynamically by PHP code, so don't expect to find any html files in these paths. It is highly unlikely that you will need to directly manipulate any of the code in these paths. WordPress version updates and all customization is best done through the provided WordPress administration screens.

Apache/Web Server Configuration

Since WordPress is a PHP application, make sure Apache is set up for PHP (that is, the Apache PHP module is included and configured). This should be as simple as going to your Apache configuration file (*httpd.conf*) and uncommenting the following line:

```
LoadModule php5_module libexec/apache2/libphp5.
so
```

In a typical Linux environment. Apache configuration files are located in /var/conf/httpd of /var/www/conf. This all depends on where it was installed.

For this article, I will assume that you already have a basic installation of Apache Web Server complete. Configuration of Apache can be very simple or very complex, and it is a topic too large to cover here. In your hosting environment is under your control, some Apache configuration knowledge is required. There is no requirement that Apache be used as the web server, but it is the most commonly used web server, both for WordPress and most PHP applications.

Back at the command line, Apache needs to be restarted for this configuration change (inclusion of the PHP module) to take effect.:

```
sudo apachectl restart
```

Next we need to set up MySQL. Most Linux environments already have it installed, and again, if you're dealing with a hosted environment, much of this can be skipped. If this is your situation, you simply need to create a database to be used for WordPress.

MySQL Database

There are certainly ways that WordPress can be configured with databases other than MySQL, but such configurations are not officially supported, and may lead to a number of technical gotchas. That said, there is really no good reason to attempt a database other than MySQL. On the contrary, to do so will cause more problems than it is worth.

Because many hosting providers include WordPress, it configuring the MySQL database may not be necessary. Most hosting providers include a pre-configured MySQL database for each user, or installation scripts that make the process as simple as clicking and choosing schema names. If this is your situation, see your hosting provider's help and support pages (generally hosting providers provide some sort of control panel page, which may include handy services like Softaculous or Installatron).

While not necessary, myPHPAdmin offers a simple web-based interface for administering a MySQL database. If you prefer to avoid the command line for future administration, this may be worth looking into. Chances are, if your hosting provider uses guided installation programs like those already mentioned, your MySQL administration can already by done using myPHPAdmin or a similar tool.

If you have only one database and it is already in use, you can install WordPress in it - just make sure to have a distinctive prefix for your tables to avoid over-writing any existing database table.

The MySQL installation includes documents, manuals, and detailed installation instructions. Start by viewing the README and INSTALL-BINARY files in the MySQL path (e.g., /var/local/mysql).

The following steps are necessary to create a specific mysql user and group.

```
shell> groupadd mysql
shell> useradd -r -g mysql mysql
shell> cd /usr/local
shell> tar zxvf /path/to/mysql-VERSION-OS.tar.
gz
shell> ln -s full-path-to-mysql-VERSION-OS
mysql
shell> cd mysql
shell> chown -R mysql .
shell> chgrp -R mysql .
shell> scripts/mysql_install_db --user=mysql
shell> chown -R root .
shell> chown -R mysql data
shell> bin/mysqld_safe --user=mysql &
# Next command is optional
shell> cp support-files/mysql.server /etc/
init.d/mysql.server
```

Note: The above example assumes Linux commands. In a Mac or Windows environment, it is best to use the System Preferences/Control Panel interfaces to great users and groups.

As with Apache HTTP Server, MySQL administration is a very large subject. There are many factors to consider when hosting a database with MySQL. For WordPress hosting, all of the heavy lifting of database schema creation is handled. All we need to do is create the database, user, and permissions. To create all that, we're going to create a separate schema.

To do this, we need to go back into MySQL at the command line (MySql must be running, obviously. This setup of this can be done via the UI or as is done in the above shell commands, with the line "cp support-files/*mysql.server* /etc/*init.d/mysql.server*." In this case, the mysql server must first be started. If it is not running, go ahead and do /etc/*init.d/mysqld* start from the command line. You will probably want to set MySQL up to run as a default service.

In the following steps we'll create a schema specifically for WordPress. We'll create a database called "wordpress" and a user for that database called "wp." Remember the admin/mysql user password that you created previously. Alternatively, "root" can be substituted as the username. After this setup, however, there won't be much of a need to use root for WordPress tasks.

```
shell>  mysql -u mysql -p
Enter password: Welcome to the MySQL monitor.  Commands end with ; or
\g. Your MySQL connection id is 5340 to server version: 3.23.54   Type
'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>CREATE DATABASE wordpress;

Query OK, 1 row affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON wordpress.* "wordpressusername"@"hostname"
  -->IDENTIFIED BY "password";

Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;

Query OK, 0 rows affected (0.01 sec)

mysql> EXIT

Bye $
```

The above steps created the initial MySQL user. These steps did not create specific WordPress schema. This task is done by the WordPress setup. The beauty of this is that the rest of our setup can be done via the browser.

Installing WordPress using cPanel

If your hosting provider provides the *cPanel* hosting control panel (and it is very likely that it does), congratulations! The steps to installing WordPress are dramatically simplified.

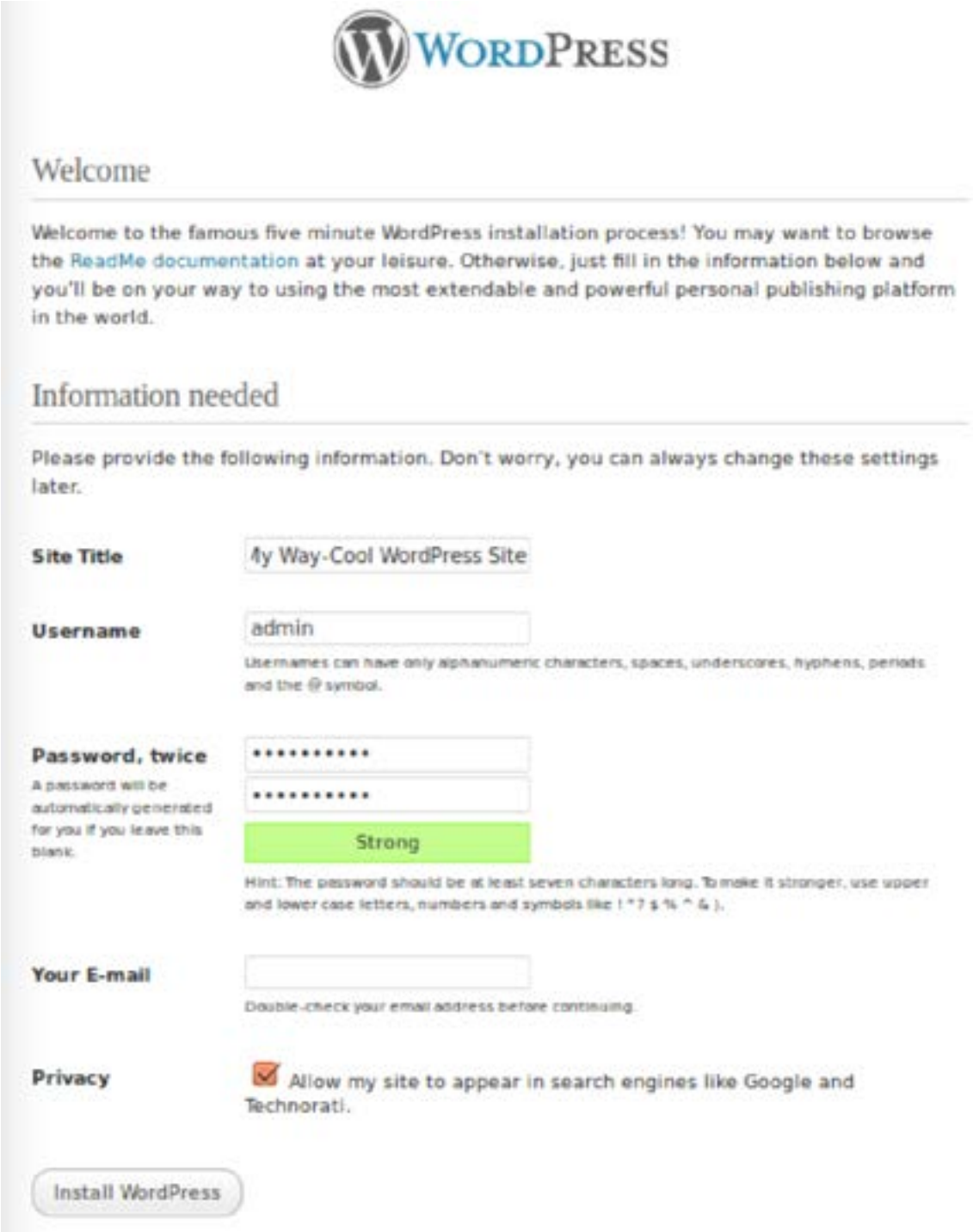
Often, this database has already been created. Not to worry—WordPress can use that single database. You will find during the cPanel WordPress installation that you are prompted to enter a WordPress table name prefix, the default being **wp_**. WordPress will create all the tables it needs within the single database that your hosting service has provided. If this is your situation, the following steps can be avoided. If you are unsure, go ahead and attempt the database creation steps. At some point it will become clear whether or not you can create a new database and if one already exists.

1. Log in to your *cPanel* user interface.
2. In the Databases section **MySQL Database Wizard**. Note: Some hosting providers allow only one database per user.
    1. **Create a Database**. You'll need to choose a database name. "WP" works, if you're not feeling creative. Your hosting provider may allow multiple databases. If this is the case, others can be created later. For now we are simply creating a database for use by WordPress.
    2. **Create Database Users.**
3. **Add User to Database.** Select the **All Privileges** checkbox.
4. **Complete. Take** note the database name and user—you do not want to forget these! At this point, it is wise to record the *hostname*, *username*, *databasename*, and the password you chose. In general, the hostname is localhost, as this is the host

that MySQL connects to, and it is very likely that the database that was created is on the same machine. This is not necessarily true, so take note.

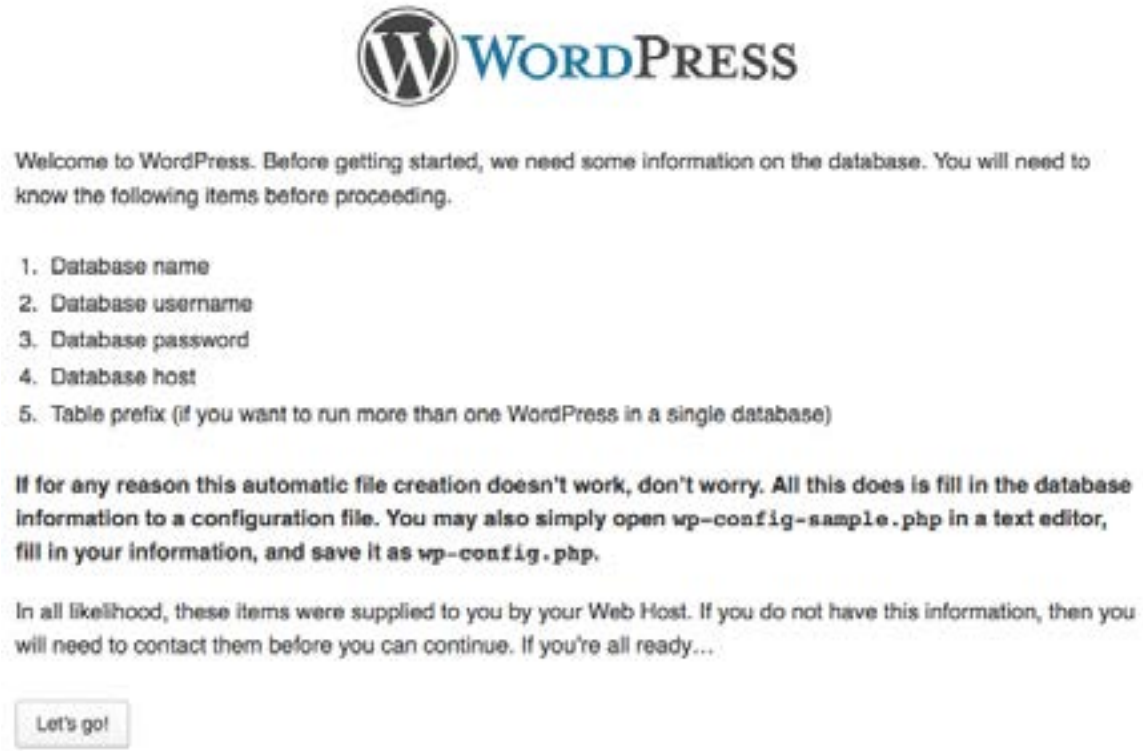
Permissions correct, database created, admininstrator username and password set, we're ready to get into WordPress and configure it.

Now that WordPress, MySQL, PHP, and Apache are installed and configured, try hitting *http://localhost/wordpress* again.



Initial WordPress Configuration

The first time WordPress is loaded it will recognize that it has not yet been configured. In your favorite web browser, go to *http://localhost/wp* (or http://localhost/, depending on where you installed). You should see a page like this:



Click on that "Let's Go" button to get to the following page. This is where WordPress prompts for database location.

We previously created a database named **wordpress**, and granted all privileges to the user **wp**. The database host, as far as WordPress is concerned, is localhost. It is possible that MySQL is hosted elsewhere, but in our case we are hosting our web content, WordPress, and database all on the save server. (If **localhost** does not work for you, try **127.0.0.1**.)

After submitting these initial details it's time to login to your administrative pages (http://<my-site>/<sub-directory>/wp-admin).

WordPress Codex

WordPress's central documentation system, the **WordPress Codex** can be found at *http://codex.wordpress.org*. If searching for help on a WordPress subject, it's likely that you'll end up here. The documentation and help system found here provides details on nearly all of the features, plugins, and customization. The Codex is where advanced usage details, for those who wish to dive in and take things to another level, are found.

Further WordPress Configuration

We've installed WordPress. Piece of cake! But for users accustomed to WordPress hosted blogs, it is necessary to highlight some of the differences. The clean WordPress installation we have just performed doesn't have all the plugins that are immediately available on a hosted WordPress site. When using a hosted WordPress site, features such as spam block, social network sharing, statistics, and subscriptions are already present. For those of us installing WordPress from scratch, these plugins need to be installed. Additionally, hosted WordPress sites present the user with a set up templates to choose from. This may seem handy, but plugins and template availability are limited on *WordPress.com* hosted sites.

Per your web server setup, you blog is available at one of the two following addresses:

http(s)://my-blog/
http(s)://my-blog/<sub-path>

And the administration user interface is available at one of the two addresses (again, depending on your web server setup):

http(s)://my-blog/wp-admin
http(s)://my-blog/<sub-path>/wp-admin

Themes

Appearance ->Themes

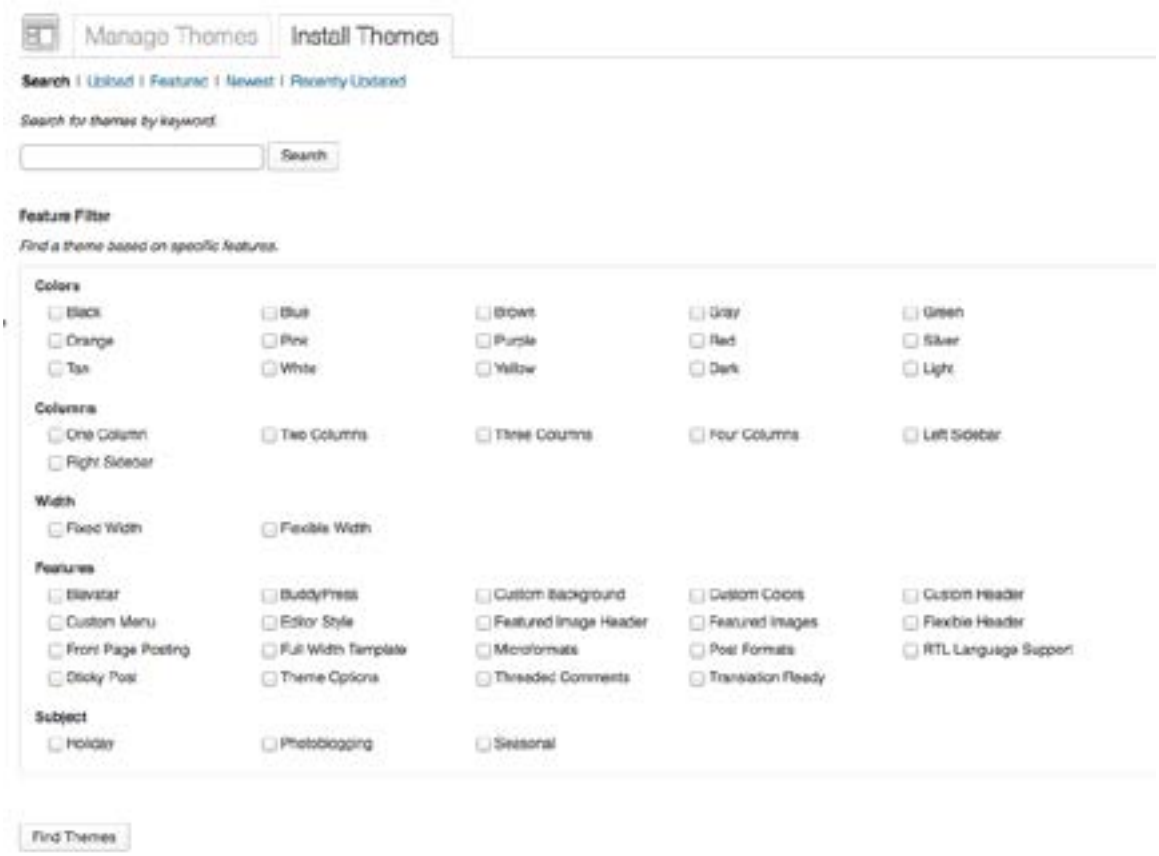
Thousands of themes exist, and new ones are being created daily. The pimary source of themes (as well as plugins) is the WordPress Codex (this is where WordPress locates themes that are displayed while browsing within the admin pages).

Themes are installed and managed in the administration interface at **Appearance-Themes**. There are two tabs available at the top of the Themes admin screen: **Manage Themes** and **Install Themes**. Manage Themes is where you'll go to view all the themes you've installed. From this tab you can preview and set themes as active. But we're not ready for this—not yet.

The second tab, **Install Themes**, is the first step. WordPress has shipped with a default theme, but let's go ahead and select something different.

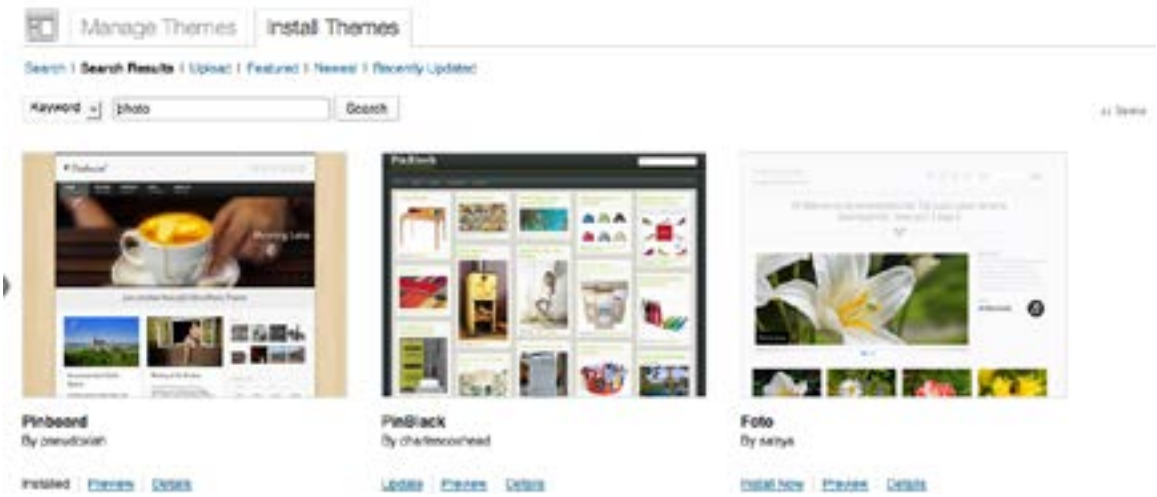
Selection of a theme for installation is pretty simple. On the Install Themes page, you will notice that there are a number of selection criteria options to help whittle down the search results when browsing a theme. Which criteria you select is largely dependent on the objectives and personal preferences you have in mind for your site. While theme colors can be selected from here, my suggestion is that you don't filter by color. Remember, colors can very easily be changed. Some of the other features, such as RTL support, number of columns, custom menus, and sticky posts are not as easy to modify. So choose a theme that, at a high level, meets your basic needs.

You can also search by keyword here, but you may find that this limits the results more than you may wish. On the other hand, it might be very useful. For example, if this site is intended as a showcase of your photography, go ahead and search using the keyword "photo" to see a number of themes that are tailored for just such a thing.



Below each theme thumbnail there are a few links:

- **Details:** See a description of the theme.
- **Preview:** See an example of what the installed theme might look like. This is not an Active Theme. Rather, this link displays a sample page (as screen shots) using the theme.
- **Install Now:** This link is displayed if the link has not yet been downloaded ("installed"). Download the theme to your local themes directory. This makes the theme available for activation.
- **Update:** The update link is available only if the theme has already been installed and a new version of the theme is available. If you update a theme that is currently the active theme of your site, the changes will take place immediately. Some of these changes may be significant. See the theme details or the theme creator's site for details.
- **Live Preview:** This link is available only if the theme has been installed. Clicking opens a preview window showing what your site will look like if the theme is activated (extremely useful).
- **Activate:** This link is displayed only after the theme has been Installed. This link applies the theme to your pages, posts, and overall appearance.



While there are thousands of themes available for download from various locations (some cost, many do not) WordPress has made it very easy to find almost any them you would ever want by hosting many of them. Themes available from WordPress can easily be downloaded through the administrator interface. Given the volume to choose from make use of the theme search filters that are available:

- **Search:** Search by keyword and feature.
- **Popular:** The link pretty much says it all: The most popular themes used. Be aware, sometimes it's nice to stand out in the crowd.
- **Trending:** Search for themes that are increasing in popularity. This is useful because popular themes may be popular only because of wide use, and not because they are currently among the best available.
- **Featured:** Browse a collection of WordPress selected, quality themes.
- **New and Recently Updated**: Browse the most recently created and updated themes. Many themes have been around for a while, but because of continuous improvements (such as the inclusion of responsive design), such themes can still appear appealing to a current web audience.

Should you decided on an initial theme that is not hosted by WordPress, it will need to be installed by uploading a .zip or *.tar.gz* theme file. This is done in the admin UI at Appearance->Themes->Install Themes->Upload. To start, let's just choose a theme that is available through WordPress. For those wanting to use a theme found elsewhere, this can be done after the blog/site is all set up.

Themes come in many varieties. Some are very basic, others extremely complex. Some themes provide a very general look and feel, while others are heavily tailored to a specific type of website. If you're a photo blogger, small storeowner, contractor, and just a simple blogger, start by looking for themes that best fit your desired use.

Before choosing a theme, it might be best to create a few dummy posts. This will help as you preview the appearance. While it can be fun to browse themes and see what your blog/site will look like, it can also be a distraction from completely the setup. Since this site is going to be self-hosted, remember that your theme will be fully customizable. So choose a theme that is close to what you like for now.

Definition

Installed Theme: *A theme that has been downloaded and unpacked in the WordPress wp-content/themes directory. Most themes include around 20 CSS and PHP files.*

(*http://wordpress.org/themes/*). Installing and setting a theme is a two-step process: Installing the theme(s) and activating a theme.

Definition

Active Theme: *A theme that has been downloaded and unpacked in the WordPress wp-content/themes directory and is selected as the current active theme.*

The first step is installing a theme. Note that an Installed Theme is not necessarily an active theme. The process of installing a theme in WordPress sets that theme as one that is available to your website. Only after a theme has been installed can it be viewed (Active Preview) or set as the active theme. While browsing themes to install, you can view the current version, community rating, and details. Some themes have great details highlighting theme features and purpose. Other themes are a little light on the details, so it may be necessary to install them theme and take a look.

Installed themes appear in the wp-content/themes. After installing a few, take a look there to see what has changed. You may see something like this:

```
twentythirteen
twentytwelve
my-custom-theme
/var/www/wp/wp-content/themes $
```

Looking more closely (cd into a theme directory), see the files that are installed for the theme:

```
/css                        functions.php
/languages                  footer.php
/js                         editor-style.css
/inc                        editor-style-rtl.css
tag.php                     content.php
style.css                   content-status.php
single.php                  content-quote.php
sidebar.php                 content-page.php
sidebar-front.php           content-none.php
search.php                  content-link.php
screenshot.png              content-image.php
rtl.css                     content-aside.php
page.php                    comments.php
page-templates              category.php
index.php                   author.php
image.php                   archive.php
header.php                  404.php
```

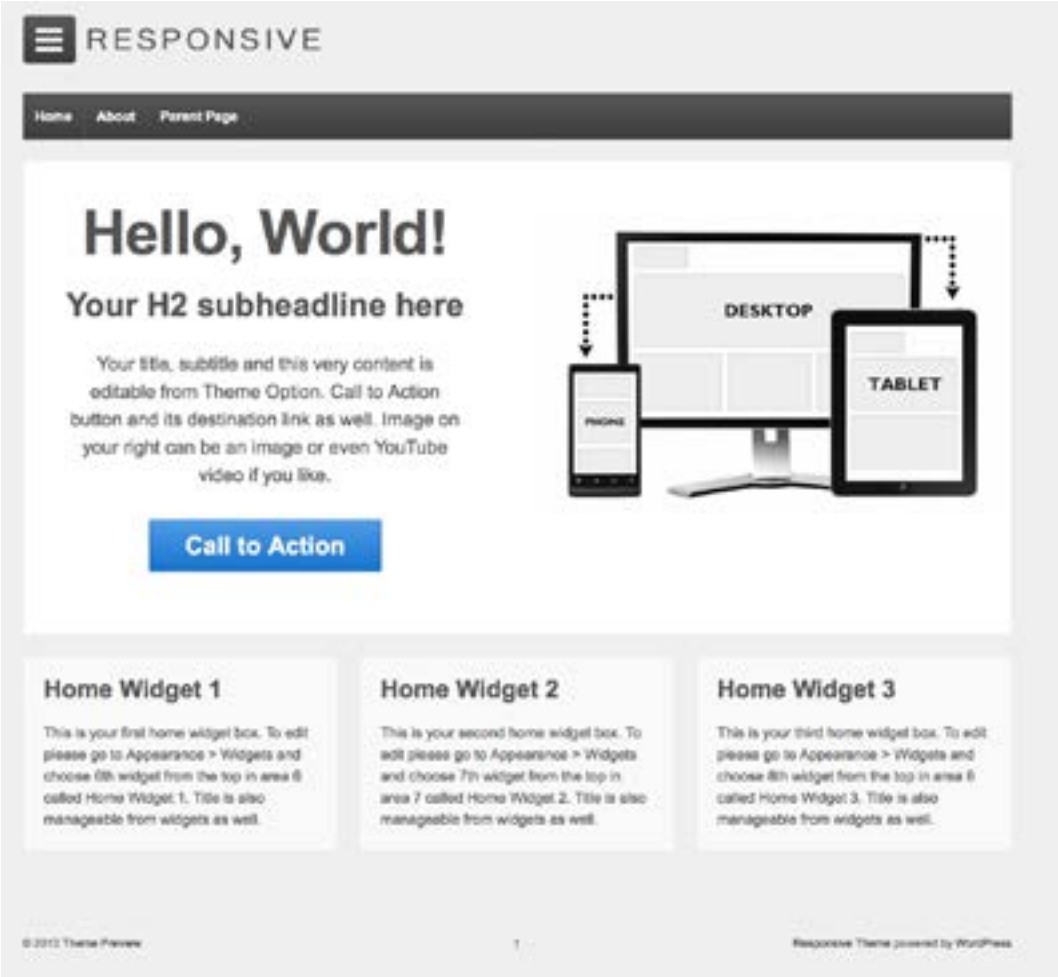
An experienced web developer may surmise (correctly) that the above files contain Multilanguage support, JavaScript, stylesheets and PHP specific to both end user/viewer experience as well as administrative pages. A theme may include certain bells and whistles that extend the administrator's ability to modify certain elements.

While browsing themes to install, you will notice a Preview link for each one. This preview will not show you what your site will look like with the theme (not exactly). This is simply a link to a sample of the theme. Before viewing an **Active Preview**, the theme must be installed.

Definition

Active Preview: *A preview window which is shows the blog/site with using an installed theme, pulling all posts, settings, and active plugins so as to display the appearance should the user activate the theme.*

Responsive Themes



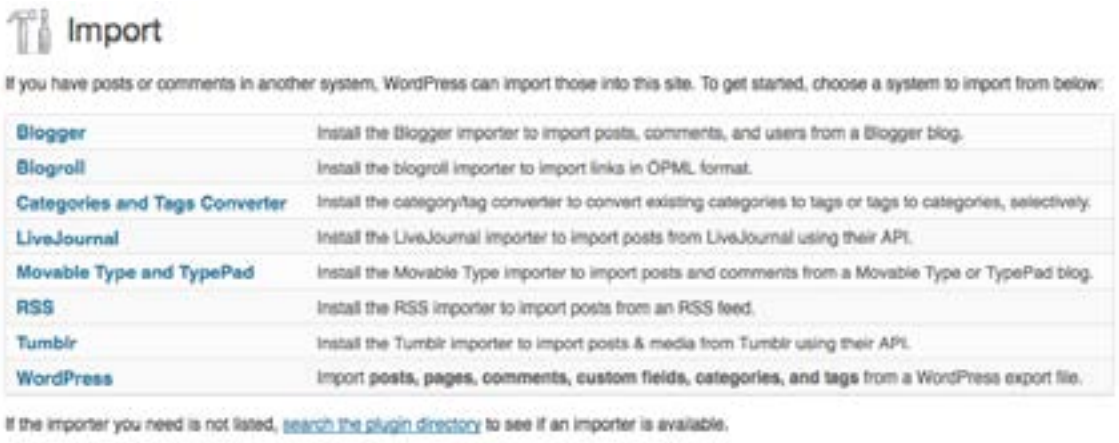
You will likely notice a number of themes that describe themselves as @media "responsive." Essentially, this just means that the theme uses CSS3 media queries to adjust the layout, size, and/or orientation of page for best display on a particular device (including hardcopy print, tablets, smartphones and, naturally, plain old computer screens). Many responsive themes include hideaway sidebars and/or horizontally scrollable content.

For some interesting details on the history behind responsive designs, see (*http://www.alistapart.com/articles/responsive-web-design*).

Importing an Existing Blog

Tools->**Import**

Import functionality is available as a plugin, but I mention it here because it may be an important early step of your configuration.


Import

If you have posts or comments in another system, WordPress can import those into this site. To get started, choose a system to import from below:

Blogger	Install the Blogger importer to import posts, comments, and users from a Blogger blog.
Blogroll	Install the blogroll importer to import links in OPML format.
Categories and Tags Converter	Install the category/tag converter to convert existing categories to tags or tags to categories, selectively.
LiveJournal	Install the LiveJournal importer to import posts from LiveJournal using their API.
Movable Type and TypePad	Install the Movable Type importer to import posts and comments from a Movable Type or TypePad blog.
RSS	Install the RSS importer to import posts from an RSS feed.
Tumblr	Install the Tumblr importer to import posts & media from Tumblr using their API.
WordPress	Import **posts, pages, comments, custom fields, categories, and tags** from a WordPress export file.

If the importer you need is not listed, search the plugin directory to see if an importer is available.

In the likely case that your new self-hosted WordPress site is intended to replace an existing blog, WordPress offers an import feature that works very well. For details on all of the various import possibilities, see *http://codex.wordpress.org/Importing_Content*.

As might be expected, importing from an existing WordPress site to a self-hosted site is the most straightforward process. This is often done when users import from a *WordPress.com* hosted site to a self-hosted WordPress site. In most scenarios, the import process is as follows:

1. Log into your old blog as administrator.
2. Export the blog. If your previous blog allows export as XML, you will want to do so.
3. Log into your new hosted WordPress site.
4. Navigate to Manage->Import in admin panels.
5. Choose the appropriate Import type the list. From here, you will be prompted to install the importer plugin for the type of site you are importing from. Install the plugin (click the Install Now button).
6. Upon successful installation of the plugin, you will see two links, ***Activate Plugin & Run Importer***, ***and Return to Importers***. Assuming you are now ready to import, click the **Activate Plugin & Run Importer** link.
7. In the final step, decisions will have to be made regarding the type of import. If importing from an existing WordPress site, the XML file created in step 2 must be selected and imported. For other blogs the import process may use the blog's RSS feed to auto generate posts on your new site.
8. In some import types, you may be prompted to map authors and categories.

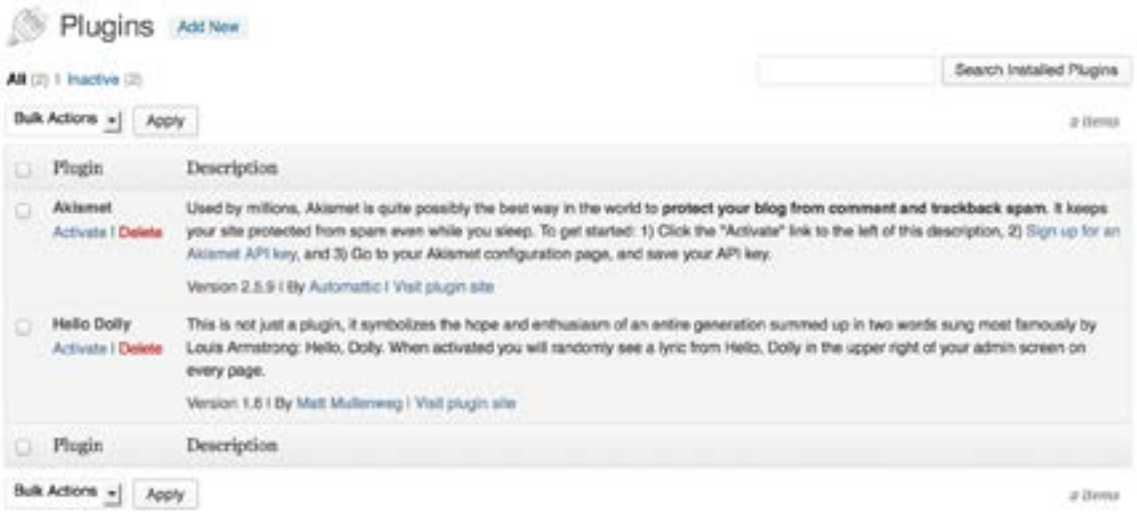
If importing from a non-WordPress site, there is likely to be some cleanup work required following the process. Depending on the type of site that is being imported from, the WordPress import plugin may not recognize authors, categories, tags, dates, or comments. Be aware that, while the WordPress import scripts often do import comments, in some cases it is not possible. Comments on old posts may not be imported to the new site during this process—it all depends on the import type.

Plugins

Plugins->**Installed Plugins**
Plugins->**Add New**
Plugins->**Edit**

As with themes, plugins are available directly through WordPress or via third party downloads. Always beware the source of a download, as, albeit uncommon, a plugin from an unknown provider could introduce a virus. Many plugins can be installed, but no plugin does anything until activated on the activation and configuration is

completed within the Plugins administration page. There are certain plugins that you may find absolutely critical to have. These include plugins for post editing, SEO statistics, social network sharing, and creating short links to posts. Others may add a certain amount of whiz-bang flare to your site.



Definition

Search engine optimization (SEO) *is the process of improving the visibility of a website or a web page in a search engine's "natural" or un-paid ("organic") search results. In general, the earlier (or higher ranked on the search results page), and more frequently a site appears in the search results list, the more visitors it will receive from the search engine's users. SEO may target different kinds of search, including image search, local search, video search, academic search,[1] news search and industry-specific vertical search engines.*

Wikipedia.org, https://en.wikipedia.org/wiki/Search_engine_optimization

Plugins are PHP extensions to existing WordPress features. From appearance improvements to SEO statistics to social networking, there are plugins for nearly everything imaginable. This is a very good thing… While you may be an experienced PHP developer, it is unlikely that you will find yourself needing to develop a custom plugin. There are plugins that are nice to have, and there are plugins that are necessary. With thousands available, I'll cover only a few of the most popular ones here. A plugin can be downloaded to your local computer from anywhere, and then uploaded to your WordPress site, but it is certainly much easier to

Step One: Locating a Plugin

WordPress provides a wealth of plugins, including many created by WordPress, and many more from 3rd party individuals and companies. Beware—A plugin may advertise itself as being free, but once installed, you may find that some of them push you to purchase additional features. I find that it is easier to remove that plugin and search for another option, with a few caveats (read on).

Behind the scenes, the plugin installation process is straightforward. Plugin files (PHP) are placed in the WordPress plugin directory (/wp-content/plugins). This directory is maintained by WordPress, so there is no need to maintain it outside of the admin pages. Plugins are packaged as .zip files. When a plugin is installed, the content is extracted into the /plugins directory.

If you are running on a server that does not have an FTP host installed, it will be necessary to download the plugin to a local path and then extract it to wp/wp-content/plugins. A much easier approach is to simply run an FTP server locally. This is because WordPress uses FTP or SFTP to transfer the plugin.

To start FTP on MacOS:

sudo -s launchctl load -w /System/Library/LaunchDaemons/*ftp.plist*

To start FTP on Linux:

```
/sbin/service vsftpd start
```

Step 2a: Adding a Plugin Through WordPress's Provided List

To install a plug-in, choose Plugins->Add New, which takes you to the Install Plugins page. You'll see five links at the top of the page that let you decide how to find the plug-in you want. Plugin's, unlike themes, do not provide active or inactive previews, so read all the details and understand what you are getting before you install. The plugin installation is pretty quick, but you will not notice any differences until the plugin is activated.

Step 2b: Adding a Third-Party or Custom Plugin by Upload

Once installed, a plugin still needs to be activated before it does anything, and plugins are easily uninstalled. In general, it is unlikely that you will damage your site by installing a plugin, but this is not always the case. Certain plugins permanently change database values For example, there are some that change URLs to posts, shortening them. Such plugins may be highly useful, but take heed: One such a plugin is installed it cannot be removed without breaking your site.

Step 3: Activating a Plugin

Activating a plugin is really a two-step process. Although many plugins require no customization, others do. Sometimes the configuration steps are clear. If they are not, try the plugin developer page for more details.

Installation of a plugin is similar to installation of a template in that it is a (simple) two-step process. The first step is uploading the template (which can be done through the WordPress administration interface). The second step is activating and configuring the template (some templates require no particular configuration, but many do). Some templates add entirely new links of the left side of the administration page. These are very convenient in many cases, know that your administrative experience may vary—it all depends on the plugin, and it might not be immediately clear where the changes took place. The best plugins provide details on such usage.

Plugins are not always consistent with regard to their setting modifications. Some templates, for example those that enhance post creation and editing features, are found in Posts section of the Administration pages. Others are seen in Appearance/Widgets.

Important Plugins

Believe it or not, WordPress does not come with built-in search functionality (the same goes for custom links and permalinks). There exists a plethora of search plugins (many of them in the form of widgets) the allow you to place a search box on the side of posts, pages or both. Whatever the nature of your site, it is generally good to provide readers with the ability to search content. Find a plugin that allows you to limit search criteria. You may not want the search plugin to search comments or certain pages (or posts beyond a certain date).

Important Plugins: Links

Links->**All Links**
Links->**Add New**
Links->**Link Categories**

The link features are not a part of the default *WordPress.org* installation. For those familiar with *WordPress.com* hosting, this is one of many differences in the basic setup. The official WordPress plugin for creating and managing links is Links Manager (see *http://wordpress.org/plugins/link-manager/*).

For a blogger, community site, or even a business site, it's always good to share links to friends, organizations of common interests, and partner. WordPress allows the creation of custom links, customizable by category. This isn't just a way to be a good web citizen—it may also help to drive traffic to your site.

Additionally, creative use of links can be used to link to content within your own site categories, and included as menus on a page or sidebar as a widget. Other link plugins include menu, category, tag and social media sharing plugins. Such plugins can make your site easy to navigate for a user and easily indexed by search engines.

Important Plugins: Fighting Spam

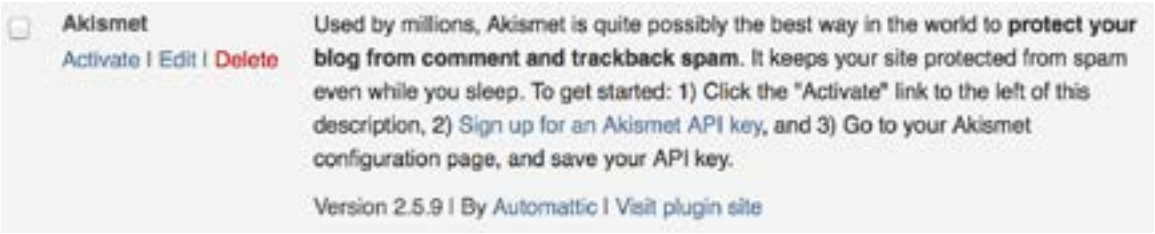
It may seem odd that the subject of spam appears in the plugins section, but plugins are the best way to battle the problem.

It's inevitable: If you open your site up to comments, you will get spam. There are ways to battle this, the simplest of which is by configuring your site to limit comments only to logged in users (those with WordPress accounts). This is limiting if you wish to allow for anyone to comment, and it doesn't ensure that the logged in user isn't a spammer.

You can also lock comments down so that they do not appear until you have approved them. Comments are "held for moderation." This is certainly something worth considering if you run a business. The options here are self-explanatory.

Comment settings can be configured at **Settings**->**Discussion** settings.

To automate much of the pain associated with spam prevention, the most popular plugin is Aksimet:



While this plug requires a certain amount of configuration, including registration Aksimet (you will need to create an Aksimet key), its effectiveness against spam is highly effective. For personal use an Aksimet key is free, and for business use the cost is minimal, and well worth it if you wish to prevent potentially embarrassing or unprofessional spam from appearing on your site.

Permalinks

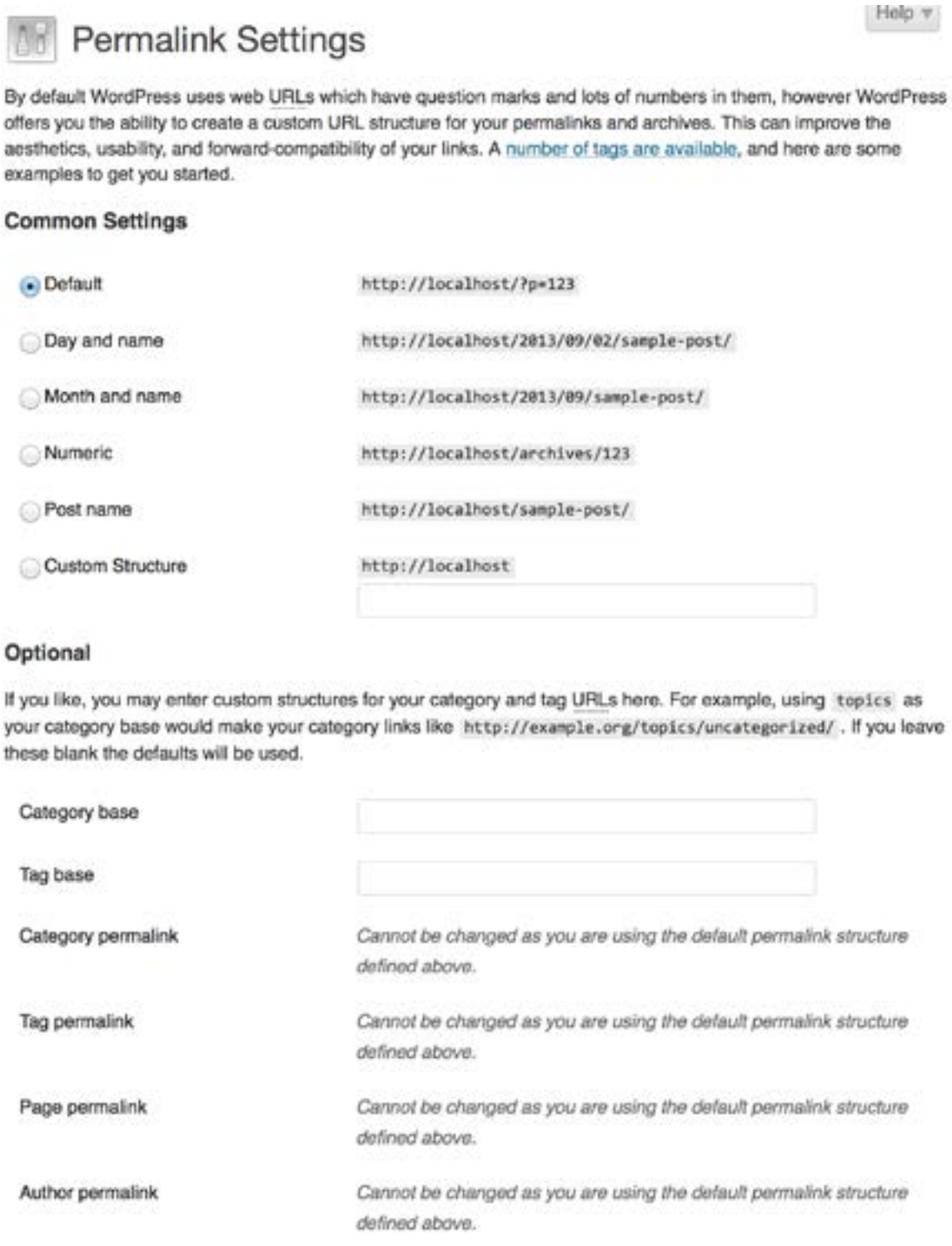
Settings->**Permalinks**

Without permalinks, you'll notice that page and post URLs contain GET data on the address line. For example, your very first post may look like this:

http://localhost/?p=1

There are other permalink plugin options, but the most common one (and the official WordPress plugin) is Permalink Editor (see *http://wordpress.org/plugins/permalink-editor/*).

After installing the permalink plugin, activated it at Plugins->Permalink Editor. The change won't be immediate obvious, as it must first be configured at Settings->Permalinks. This plugin is self-explanatory, with a number of permalink options being presented (see below).



My personal preference is to use the simple Post Name option. Combined with the option Category and Tag base names, each post and page can be assigned a unique pathname, generally one that is short enough to easily share. For a blog, you may wish to consider including months and or days in the post URLs.

Don't Overdo It With Plugins

As great as plugins can be, there are downsides to consider. The more plugins you have, the more risk you introduce for collision of features and need to maintenance and configuration, so consider the necessity of any plugin prior to installing it. Different plugins may work in slightly different ways, sometimes with overlapping features. For example, there are multiple plugins that provide social network sharing features. One plugin may provide features for sharing to Reddit, Facebook, and LinkedIn. Another may provide sharing to Reddit, Facebook, LinkedIn, Digg, Google+, and Email. If you install and activate both, the resulting user experience will be clumsy.

Test

Certain plugins can cause slight load delays, whether those plugins impact end-viewer experience, or your own administrative experience. As someone who has tinkered with a number of plugins, I have found more than a few that made with the admin pages, reader pages, or both, painfully slow. Sometimes this problem is made evident by user reviews. Other times, you'll have to discover this on your own. Always test response times, appearance, and function after installing a plugin. A plugin may result in some unexpected change. Therefore, while it may be tempting to install a bunch of great looking plugins all at once, it is good practice to install and configure one plugin at a time.

Impact on Load Times

I've personally been frustrated by plugins that advertise themselves as being free, only to find that, upon installation, they provide limited functionality and continuously prompt for purchase of the full version. Unless necessary, I tend to avoid such plugins.

Plugins can be written by anyone, so be cautious to install plugins from trusted sources. Read the plugin reviews and be careful to avoid any (or understand the risks) that may include security holes.

Finally, pay attention to the plugin activity, how widely it is used, whether or not it is maintained, and the age of the plugin. Some plugins can be rendered obsolete by newer versions of WordPress or newer plugins.

The Content: Posts and Pages

Setup is never really complete. Like the landscaping at a park, people expect new and ever-changing scenery.

This isn't to say that you're not ready to begin with content creation. Rather, a site that doesn't morph and change over time is, well… boring. Keep this in mind while as time passes by. New content—regular content—is a fair expectation of a reader. Even if your WordPress site is a semi-static business or informational page, pages that do not change over time begin to seem dull. Go ahead and try a new theme from time to time. Change your colors, logos, and arrangement. Dynamic content and new scenery are the things that keep visitors coming.

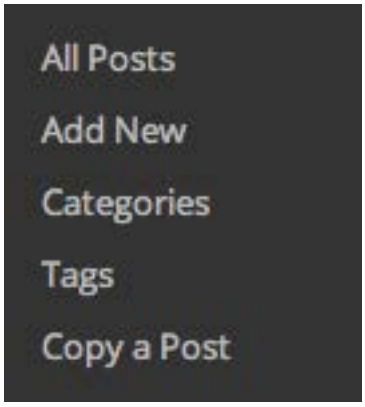
That said, dynamic content is why we're here in the first place. An unchanging set of web pages is created easily enough with HTML, CSS, and a few JavaScript goodies. With WordPress (as with any dynamic CMS, wiki, or blogging tool), the purpose is to promote and simplify ongoing dynamic content.

Even many businesses have embraced the idea of blogging—even if the regular posts are not specifically called *blogs*. Take a look at some of the biggies: *ibm.com*, *redhat.com*, *ge.com*, *exxonmobil.com*, *lilly.com*. Think of any company—large, successful company, and go to their website. What you'll find is not simply a website. It is more of an experience for the view--an opportunity to see what is new and interesting for the business in question. Typically, there on the first page you'll find information on careers, news, events, and conferences. It's what we've come to expect. See *https://wordpress.org/showcase* for an impressive list of WordPress powered sites.

With Posts and Pages, on the right side of the screen you will see a Publish section with a few options, including **Save Draft** and **Publish**. As a general rule, it's a good idea to Save Draft frequently while editing. This will save a draft of your post or page without publishing to the live website. As stable as WordPress is, we've all been frustrated by network connection losses, browser problems, and other applications when something unexpected happens and our work is lost.

Posts

Posts are the things that we think of when it comes to blogging. With WordPress, posts can be categorized, tagged, and formatted in different ways.



Post Format

Those already familiar with *WordPress.com* will know that various post formats are available:

- Standard
- Aside
- Image
- Video
- Quote
- Link
- Audio
- Gallery

These post formats allow various templates to modify the layout of a post depending on the format type. For example, and Aside Format post may appear with smaller texts, or off to one side. This all depends on the template that has been chosen. I typically select Standard for all of my posts, and rely on WordPress shortcodes and post editing, as I don't wish to have post appearances change profoundly when I choose a new template. This isn't to say that post formats should be avoided, but whether or not they meet your needs is subjective.

Pages

A key element separating WordPress from other traditional blogging tools is the ability to create pages that are not blog posts. These pages are still read from the database and read dynamically, but they can be displayed as separate entities—without a blog-like post date and time. It is perfectly reasonable, and common, to create an entire WordPress site without any blog posts.

Shortcodes

Shortcodes are WordPress tags that allow specific content to be embedded in a post. Like codes in a markup languages, **shortcodes** are replaced with dynamic content at presentation time. When using **WYSIWYG** editing mode, the fact that a shortcode is being used behind the scenes will not be obvious.

Shordcodes appear in square brackets ([ ]). There are many already included with WordPress and many plugins that add new ones or expand on existing.

A few **shortcodes** include:

- `[gallery]` – Displays an image gallery in a post or page
- `[slideshow]` – Embed a of photos within a post or page
- `[office]` Embed a document Microsoft Office Live document
- `[twitter-timeline]` - Embed a Twitter timeline

The list of available **shortcodes** is extensive and ever changing. Details can be found at *http://en.support.wordpress.com/shortcodes/*.

Tags

Admin Sidebar->**Tags**

Tags are different *things* than categories. Thing of Twitter *hashtags*—a set of short, preferably one word descriptions of a post –words that give a reader a quick view of the subject matter of a post. Tags can be especially handy for helping others to search for and find tops. A key difference between tags and categories is this: Tags are freeform words, created at post-time,, that label content in an informal, non-hierarchical manner. Along with a **Tag Cloud** plugin, regular use of tags can give a convenient visual of common site subject matter.

Categories

Posts->**Categories**

Post **Categories** (not to be confused with Link categories) may not always be necessary, but as a general rule, it is a good idea to create categories into which posts (and pages) can be organized. WordPress categories can be organized in a hierarchy, and this is a useful way to organize the appearance of menus (primary, secondary). WordPress is set up by default to categories posts as **uncategorized**. While categories are not critical, I will say that leaving all posts in this default category give the appearance of an unfinished website. By default, Pages are not categorized. However, there are plugins that allow this.

You do not necessarily need to determine all of your categories or the hierarchy of categories immediately. I've added and rearranged categories long after a site has been created. A great thing about WordPress is that the category organization isn't set in stone. For example, if you have two categories to begin with, say ***Operating Systems*** and ***Linux***, and each was initially created at the same level, if at some point you decided to move ***Linux*** underneath ***Operating Systems*** (making Operating Systems the parent category of ***Linux***), no links will break. If later still you decided to add a subcategory Windows to the ***Operating Systems*** parent, and ***RedHat***, ***Ubuntu***, and ***CentOS*** has subcategories of ***Linux***, it is easy to do. Keep in mind, however, that addition of new categories may prompt you go to back to old posts and re-categorize.

Creation of categories can also be done at post or page creation time. There have been a number of times when I've written a post, and, unsatisfied with my existing categories, I created one while editing the post. On the right side of the post editing window you

will see your existing categories (there are two tabs in the **Categories** section: **All** and **Most Used**). At the bottom of the categories section is a link: **Add New Category**. Don't worry—clicking this link will not take you away from your post. It is handled right there on the same screen, without a page reload. Likewise, any **Category** can be removed at a later time. Doing so will not remove a post that is tied to the category—the category will simply be removed.

There are a number of considerations when creating categories, including hierarchy, volume of posts, posts per category, and so on. In previous example, if you believe that you may have just a few posts in the ***CentOS*** subcategory, perhaps it should instead by labeled ***Others***. And if you do use a hierarchy, think about what makes sense. It may not be clear to a user that some OS related posts is in a sub-category while others appear only in the parent category. Again, as with most of this, it is subjective and depends on your needs and the expectations of your readers. In any case, choose categories and category relationships that distribute your posts in a sensible way.

Reader View of Categories

Categories are visible to the user in a number of ways (all configurable). A sidebar widget and/or category display with each page or post can be presented. The Categories widget, when added to a sidebar or embedded in a page, does not automatically display categories in a hierarchy. While editing widgets, you can check or uncheck whether or not you wish for them to be displayed as a hierarchy (**Appearance**->**Widgets**).

Comments

There's a certain thrill when a new comment arrives on a blog post. It means that someone has been reading, and that someone felt compelled to follow up. Unfortunately, it could also mean that someone is trolling or spamming your site. Comments can be disabled, but in general, I recommend opening up posts for comments. It creates a much more interesting user experience. This may make it necessary to lock comments down to only registered users, or setting comments up so that they must be approved (by you) prior to display. A much easier approach is to use a plugin that guards against spam (such as Akismet, which is discussed later).

Rating

Post Ratings is another feature that users with previous *WordPress.com* experience may be surprised not to find by default on a custom *WordPress.org* site. There are a number of Post Rating plugins/widgets available. While it can be fun to see ratings on a post, sometimes there is little point to making such a feature available. Once again, this is a matter that depends on the purpose of your site. For a new site, or a blog with a number of subjects that invite user feedback, ratings cam be appropriate. For a business blog, ratings are likely something to avoid. (See *http://wordpress.org/plugins/rating-widget/* for an example of a Rating Widget.) The same could be said of other plugins, such as ***Polls*** and **Like on Facebook** enabling widgets.

That's That!

I've touched on a range of subjects here, from the basic details of installation, to configuration and plugins. WordPress has become a very mature solution over the course of ten years, and as such, there is much to learn about all the various features. It is possible to expand on each subject here, and the possibilities for customization are endless. For non-technical users, including small business owners who cannot afford to hire a web developer, WordPress makes creation of a professional website attainable. For the most talented web developers, WordPress expands the possibilities by performing all of the heavy lifting.

What the Internet needs more of, and what people want, is meaningful content. The framework laid out by WordPress allows us to move forward and create that content. So I'll end with this: There are plenty of outlets for link sharing and repeating the thoughts of others. Don't do this… Create and contribute!

Snoop Dogg: www.snoopdogg.com - Created with WordPress

Whether you are setting up a web presence for a landscaper, a contractor, a group of hobbyists, or for your personal interests, remember that there are many others. Countless others! Any website worth the effort required to create it is equally worth the effort to maintain it. It's the ongoing updating that keeps readers coming back. While quality is important, it's the content that keeps readers coming. Case in point: The Drudge Report. Love it or hate it, one cannot deny it's popularity. It's simplicity has become legendary. People don't visit Matt Drudge's simple page (sometimes dozens of times a day) for its beauty. They visit for the content.

(Of course, you and I are not Matt Drudge. We want our pages to have both current, great content, as well as an elegant, attractive appearance.)

Additional Recommended Reading (Just For Fun)

20 Years of a Free, Open Web: http://info.cern.ch/
Tim Berners-Lee: http://www.w3.org/People/Berners-Lee/
History of Blogging: https://en.wikipedia.org/wiki/History_of_blogging
Lynx Web Browser: https://en.wikipedia.org/wiki/Lynx_(web_browser)

Links

Akismet: http://akismet.com/
A List Apart: Responsive Web Design:
http://alistapart.com/article/responsive-web-design
Automattic: http://automattic.com/ (the makes of WordPress)
WordPress.org: http://wordpress.org/
WordPress.com: http://wordpress.com/
WordPress Showcase: http://wordpress.org/showcase/
cPanel: https://cpanel.net/
MySQL: https://www.mysql.com/
PHP: http://php.net/
Softaculous: http://www.softaculous.com/

Matthew Rupert

Matthew Rupert lives in Wake Forest, NC, and is a software architect with Ateb Inc., (Raleigh, NC). He has been designing and developing software for over 15 years. A graduate of Ball State University (Computer Science), he has worked in a range of software development and lead roles ranging from defense contractors to healthcare. He blogs regularly on software subjects at *http://matthewrupert.net*. Find him on Twitter: @matthewrupert