



DIGGING INTO

WORDPRESS

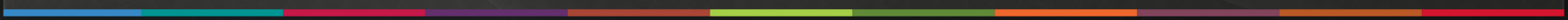
by Chris Coyier & Jeff Starr

How to set
things up right

Building themes
and how they work

Keeping sites **secure**
and **optimized**

Making the **most**
of **WordPress**



Acknowledgements

- Thank you to Thane Champie and James Starr for their help with proofreading.
- Thanks also to the many readers who have helped with further improvements.
- Thank you to everyone who allowed screenshots to be printed in this book.
- Thank you to the incredible WordPress community for making WordPress #1.
- And most of all, thanks to you, the reader, for sharing this adventure with us.

Chris would like to thank

- My mom for all the excellent printing advice (and for being a good mom).
- Jeff Penman for insisting the book was a good idea.
- Tim Chatman for giving me the time and assistance I needed to get it finished.

Jeff would like to thank

- My wife, Jennifer, for her loving support and encouragement.
- My two children, Josh and Lorelei, for being so awesome.
- Friends, family, and everyone who helps along the way.



DIGGING INTO WORDPRESS
By Chris Coyier and Jeff Starr
Version 3.7 • October 2013
© 2013 Digging Into WordPress
Visit us online @ DigWP.com

DIGGING INTO
WordPress

C H R I S C O Y I E R & J E F F S T A R R

Short URLs

The URLs in this book are so called “short URLs.” They look like this: <http://digwp.com/u/1> – When you click on one (PDF people) or enter one into a browser (Book people), you will be instantly redirected to the URL we are trying to get you to. What’s up with that? Are we trying to drive traffic to our own site? Nope, we are trying to do two things:

- 1) **Make it easier for you** (you don’t need to type in long awkward URLs)
- 2) **Make it easy for us** (it is easier to typeset and design around short URLs than long ones)

3.7

That is the current version of WordPress at the time this book was published. So if we say something like “...the current version of WordPress,” we are talking about 3.7. If we need to mention an older version, we’ll be specific about that.

So what if you are reading this and 3.8 is already out? **Don’t worry about it!** The information in here will still be valid. WordPress does a good job about not breaking existing stuff for new versions.

But there will be changes, and we intend to keep this book updated with those new things. All current owners of this book will get free PDF updates as it is updated!



See that? That’s dog food. It’s a metaphor.

We, the authors of Digging Into WordPress, eat our own dog food. We aren’t just here to stand on a pedestal and preach about how you should do things. We practice these things in the sites we work on every day.

Much of what you will read in this book is put into practice on the WordPress blog that accompanies this book.

<http://digwp.com>

Contents

1 Welcome to WordPress

1.1.1 Welcome	9
1.1.2 Why WordPress is Amazing	9
1.1.3. How to Set up and Configure WordPress	10
1.1.4 How to Implement Advanced Functionality	10
1.1.5 How to Optimize and Secure WordPress	11
1.1.6 How to Maintain Your WordPress Site	12
1.1.7 Don't Worry	12
1.2.1 So, You've Never Heard of WordPress	12
1.2.2 One Template, Many Pages	14
1.2.3 Powerful, Flexible and Extensible	14
1.3.1 Key Components of a WordPress Site	15
1.3.2 WordPress Core Files	15
1.3.3 The WordPress Database	15
1.3.4 The Back End	17
1.3.5 The Front End	17

1.4.1 Tools of the Trade	17
1.4.2 A Domain Name	17
1.4.3 Web Host / Server	18
1.4.4 Text / Code Editor	19
1.4.5 FTP Program	19

2 Setting Up WordPress

2.1.1 The Famous Five Minute Install	23
2.1.2 Where To Install?	23
2.1.3 Checking Default Performance and Proper Functionality	24
2.2.1 OK, I'm In. Now What?	25
2.2.2 Just Publish Something!	25
2.2.3 Go Look At It!	25
2.2.4 The Plan	26
2.3.1 Permalinks: Your URL Structure	26
2.3.2 HTAccess	27
2.3.3 Which Style of Permalinks?	28
2.3.4 Pick One and Stick With It	29
2.3.5 SEO Consideration: Mind Your Post "Slugs"	29
2.4.1 Categories and Tags	30
2.4.2 They are Basically the Same	32
2.4.3 Use Only One Category Per Post	32

2.4.4 Use Multiple Tags Per Post	33
2.4.5 Don't Go Overboard!	33
2.4.6 You Don't Need to Use Them At All	34
2.4.7 Custom Taxonomies	34
2.5.1 Users and Administrators	37
2.5.2 Add a New Account for Yourself	39
2.6.1 Choosing the Perfect Theme	40
2.6.2 Where to Find Awesome Themes	40
2.6.3 Previewing Themes	41
2.6.4 Key Things to Look For in a Theme	41
2.7.1 Getting Started with Plugins	44
2.7.2 Installing and Activating Plugins	44
2.7.3 Difference Between Disabling and Uninstalling	45
2.7.4 Recommended Plugins	46
 3 Anatomy of a Theme	
3.1.1 Understanding Theme Files	51
3.1.2 Every Theme is Different	51
3.1.3 Commonly Used Theme Files	51
3.1.4 How Theme Files Work Together	54
 3.2.1 Understanding Different Page Views	54

3.2.2 Page Views are for Pages	55
3.2.3 Single Views are for Posts	55
3.2.4 The Many Faces of Archive Views	56
3.2.5 How WordPress Decides Which File to Use for Rendering the View	56
3.3.1 Kicking It Off with the Header	58
3.3.2 The DOCTYPE and HTML Attributes	58
3.3.3 META Elements	59
3.3.4 The Title	59
3.3.5 Link Elements	61
3.3.6 The wp_head() Function	64
3.3.7 Template Tags	64
3.4.1 The WordPress Loop	67
3.4.2 The Loop in Plain English	68
3.4.3 The Loop Just Knows.....	69
3.4.4 Some Common "Loop Only" Functions	70
3.4.5 Some Common "Outside Loop" Functions	71
3.5.1 Comments	71
3.5.2 The comments.php File	71
3.5.3 Selective Inclusion for Different Views.....	72
3.6.1 The Sidebar	74
3.6.2 Purpose and Placement	74
3.6.3 Popular Sidebar Functions	75
3.6.4 Widgets, Widgets, Widgets	78

3.7.1 The Search Form	79
3.7.2 Why is This a Separate File?	79
3.7.3 Alternatives to WordPress Search	79
3.8.1 The Footer	81
3.8.2 The wp_footer() Hook	81
3.8.3 Mini Footers / Mega Footers	83
3.9.1 Theme Functions	83
3.9.2 Functions are for Specific Themes	83
3.9.3 Advantage Over Core Hacks	84

4 Theme Design & Development

4.1.1 Customizing the Loop	87
4.1.2 Customizing the Loop with query_posts	88
4.1.3 Customizing the Loop with WP_Query	90
4.1.4 Customizing the Loop with get_posts	92
4.1.5 The Loop Doesn't Care About Markup	93
4.1.6 The Power of WP_Query	94
4.1.7 Displaying Different Numbers of Posts	95
4.1.8 Excluding Specific Categories	96
4.1.9 Changing the Sort Order	96
4.1.10 Show Specific Pages, Embed a Page within a Page	97
4.1.11 Using Multiple Loops: An Example	97

4.2.1 Sidebars and Footers	100
4.3.1 Menus, Archive Lists & Tag Clouds	103
4.3.2 Page-Specific Menu Styles	105
4.3.3 Create the Perfect Archives Page	107
4.3.4 Impress Your Visitors with a Tag Cloud	108
4.4.1 Side Content & Useful Menu Items	109
4.4.2 Displaying Recent Comments	109
4.4.3 Displaying Recent Posts	111
4.4.4 Listing Popular Posts	112
4.4.5 Listing Recently Modified Posts	112
4.4.6 Listing Random Posts	113
4.4.7 Import and Display Twitter	113
4.4.8 Import and Display Delicious	116
4.4.9 Import & Display Other Feed Content	117
4.5.1 Creating and Using Child Themes	119
4.6.1 Styling Your Theme	120
4.6.2 Different Inclusion Methods	121
4.6.3 To Reset or Not To Reset?	122
4.7.1 Using Multiple Themes	125
4.8.1 Widgetizing	127

5 Extending Functionality

5.1.1 Extensibility	131
5.1.2 Extending WordPress with Plugins	131
5.1.3 A Plugin for (Almost) Everything	131
5.1.4 Do You Need a Plugin?	133
5.1.5 Choosing the Perfect Plugin	135
5.2.1 Plugin Usage and Maintenance	136
5.2.2 Sequential Installation	137
5.2.3 Keep Plugins Up-To-Date	137
5.2.4 Subscribe to Plugin Comment Threads	138
5.2.5 Getting Help with Plugins	138
5.2.6 Diagnosing Plugin Conflicts	139
5.2.7 Disabling and Uninstalling Plugins	139
5.3.1 Extending WordPress with Custom Functions	142
5.3.2 Plugins vs. Theme Functions (functions.php)	143
5.3.3 Examples of Useful Theme Functions	143
5.3.4 Example #1: Easy Admin Buttons for Comments	144
5.3.5 Example #2: Sitewide Shortcode Functionality	145
5.3.6 Example #3: Moving Plugins to functions.php	146
5.3.7 Example #4: Creating Plugins from Functions	146
5.4.1 Other Ways to Extend & Customize	147
5.4.2 Functions Within Theme Files	149
5.4.3 Hacking the WordPress Core	150

5.5.1 WordPress as a CMS	151
5.5.2 Working With Custom Fields	152
5.5.3 Users, Roles and Permissions	155
5.5.4 Categorizing, Tagging, and Custom Taxonomies	156
5.5.5 Page Templates	158
5.5.6 Page, Category, and Tag Hierarchies	159
5.5.7 Dynamic Menus	160
5.6.1 Extending CMS Functionality	161
But wait, there's more...	161

6 Working with RSS Feeds

6.1.1 Working with RSS Feeds	163
6.1.2 The Pros and Cons of Delivering Feeds	163
6.2.1 Different Types of WordPress Feeds	164
6.2.2 Posts Feed	164
6.2.3 Comments Feed	166
6.2.4 Individual Post Comments Feed	166
6.2.5 Category and Tag Feeds	167
6.2.6 Other Feed Types	167
6.3.1 Feed Configurations & Formats	168
6.3.2 Full Feeds	170
6.3.3 Partial Feeds	171

6.3.4 Number of Posts	171
6.3.5 WordPress Feed Formats	172
6.4.1 Using FeedBurner For Feed Delivery	175
6.4.2 Benefits of Using FeedBurner	175
6.4.3 Setting Up & Configuring a FeedBurner Account	176
6.4.4 Redirecting to FeedBurner via Plugin	177
6.4.5 Redirecting to FeedBurner via HTAccess	177
6.4.6 Redirecting to FeedBurner via PHP	179
6.5.1 Tracking & Displaying Feed Statistics	181
6.5.2 Types of Statistics Provided by FeedBurner	181
6.5.3 Displaying FeedBurner Statistics	182
6.5.4 Alternatives to FeedBurner	183
6.6.1 Customizing Feeds	184
6.6.2 Formatting Feed Images	184
6.6.3 Adding a Custom Feed Image	187
6.6.4 Include Comments in Feeds	189
6.6.5 Creating Custom Feeds	191
6.6.6 More Feed Customization Tricks	195
6.6.7 Styling Feeds	196
6.6.8 Removing the WordPress Version Number	196
6.6.9 Disable and Redirect Unwanted Feed Formats	197
6.6.10 Insert Custom Content into Feeds	198
6.6.11 Importing and Displaying External Feeds	200
6.6.12 Buffer Period After Posting	200

6.6.13 Protecting Feed Content	202
6.7.1 Validating Feeds	203

7 Working with Comments

7.1.1 Optimizing the WordPress Comments Area	207
7.1.2 Welcome to the WordPress Comments Area	207
7.1.3 About the WordPress Comment System	208
7.1.4 Comments, Pingbacks, and Trackbacks	208
7.1.5 Anatomy of the WordPress Comment Area	209
7.2.1 Syndicating WordPress Comments	213
7.2.2 WordPress Main Comments Feed	213
7.2.3 Post-Specific Comment Feeds	214
7.3.1 Formatting the Comments Area	215
7.3.2 Using wp_list_comments() or a Custom Loop?	217
7.3.3 Implementing Paged Comments	223
7.3.4 Implementing Threaded Comments	225
7.3.5 Separating Comments, Pingbacks, and Trackbacks	228
7.3.6 Eliminating Pingbacks and Trackbacks	232
7.3.7 Control Comments, Pingbacks, and Trackbacks	234
7.4.1 Customizing Comment Display	236
7.4.2 Numbering Comments Globally and Locally	236

7.4.3 Alternating Comment Styles.....	240
7.4.4 Custom Styles for Authors & Members.....	241
7.4.5 Styling Comments with Gravatars.....	243
7.4.6 Add a “Your comment is awaiting moderation” Message.....	246
7.4.7 Moderation Links in the Theme Itself.....	247
7.4.8 Display Comment, Ping/Trackback Counts.....	248
7.5.1 Optimizing the Comment Form.....	249
7.5.2 Set up Comment Previews.....	249
7.5.3 Rich-Text Editors for Comments.....	250
7.5.4 Adding Comment Quicktags.....	252
7.5.5 Comment Management and Spam Prevention.....	254
7.6.1 Controlling Comment Spam.....	254
7.6.2 WordPress’ Built-In Anti-Spam Functionality.....	255
7.6.3 Anti-Spam Plugins for WordPress.....	256
7.7.1 Other Considerations & Techniques.....	257
7.7.2 Enhancing and Encouraging Comments.....	257
7.7.3 Nofollow Links.....	258
7.7.4 Integrating Twitter.....	258
And so.....	259

8 Search Engine Optimization

8.1.1 SEO Strengths and Weaknesses.....	261

8.1.2 Strong Focus on Content	261
8.1.3 Built-In “nofollow” Comment Links	262
8.1.4 Duplicate Content Issues	262
8.2.1 Controlling Duplicate Content	263
8.2.2 Meta noindex and nofollow Tags	264
8.2.3 Nofollow Attributes	267
8.2.4 Robots.txt Directives	269
8.2.5 Canonical Meta Tags	273
8.2.6 Use Excerpts for Posts	275
8.3.1 Optimizing Permalink Structure	276
8.3.2 Default URLs vs. “Pretty” Permalinks	276
8.3.3 Keep Permalinks Short	277
8.3.4 Maximize Permalink Keywords	280
8.4.1 Scoring with Google	281
8.4.2 Content, Content, Content	281
8.4.3 Detecting Duplicate Content	282
8.4.4 Optimizing Heading Elements	283
8.4.5 Optimizing Title Tags	284
8.4.6 The nofollow Wars	286
8.4.7 Fixing Broken Links	287
8.4.8 Using a Sitemap	288
8.4.9 SEO-Related plugins	289
8.5.1 Tracking the Success of Your Site	290

8.5.2 Statistical Plugins	290
8.5.3 Mint Stats	291
8.5.4 Google Analytics	292
8.5.5 Other Metrics	293
8.6.1 Closing Thoughts on SEO	294
Into the future.....	295

9 Maintaining a Healthy Site

9.1.1 Keeping Your Site Healthy	297
9.1.2 Securing WordPress	297
9.1.3 Setting Secure File Permissions	298
9.1.4 Disabling Directory Views	300
9.1.5 Forbid Access to Sensitive Files	302
9.1.6 Remove the WordPress Version Number	313
9.1.7 Securing Your Database	313
9.1.8 Secure Multiple Installations	315
9.1.9 Prevent Hotlinking	315
9.1.10 More WordPress Security Help	317
9.2.1 Stopping Comment Spam	318
9.2.2 Configuring Your WordPress Admin Options	319
9.2.3 Using the Built-In Comment Moderation	320
9.2.4 Using the Built-In Comment Blacklist	320
9.2.5 Disabling Comments on Old Posts	321
9.2.6 Deny Access to No-Referrer Requests	321

9.3.1 Monitoring and Fixing Errors	322
9.3.2 Alex King's 404 Notifier Plugin	323
9.3.3 Broken Link Checker Plugin	323
9.3.4 Other Error-Logging Techniques	324
9.3.5 Online Monitoring Services	324
 9.4.1 Staying Current with WordPress	 325
9.4.2 Updating WordPress	326
9.4.3 Logging Changes	327
9.4.4 Backing Up Your Database and Files	328
 9.5.1 Optimizing WordPress	 329
9.5.2 Content and File Caching	329
9.5.3 File Compression Methods	331
9.5.4 Optimizing CSS and JavaScript	332
9.5.5 Reducing the Number of HTTP Requests	333
9.5.6 Plugin Maintenance	337
9.5.7 Database Maintenance	338
9.5.8 Other Optimization Techniques	338
To Infinity, and Beyond	341
Thank you	342

Don't make something unless it is both necessary and useful; but if it is both necessary and useful, don't hesitate to make it beautiful.

— SHAKER PHILOSOPHY

1

Welcome to WordPress

1.1.1 Welcome

Welcome to the wonderful world of WordPress. Say that ten times fast! Sincere thanks for purchasing our book and we hope the information will be as useful to you as it already has been for us.

1.1.2 Why WordPress is Amazing

There are all sorts of reasons you may have bought this book, but one of them might be because you are already a fan of working with WordPress. That's great, because we are too, and it's probably going to show.

WordPress makes controlling the content of a website easy. It can be done by you, or you can train another person to do it, because the Admin area is as intuitive as it is powerful. WordPress gives you the keys to build sites easily with all the modern features that clients crave. And because WordPress is free, open source, and has fostered a giant community around it, you'll always be in good hands. WordPress has changed the face of publishing content on the web, and we are all reaping the benefits. For all these reasons and more, we can safely say that WordPress is amazing.

If you are buying this because your boss is making you design a site around WordPress and you are none too happy about it... Bummer. But hopefully by the end we can turn your frown upside down and bring you into the light.

Plan

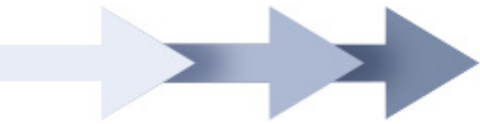
1. Buy Book

2. ???

3. Profit!

1.1.3. How to Set up and Configure WordPress

In this book you are going to learn how to set up WordPress. You'll learn that installing WordPress is "famously" easy. Then we'll discuss the various settings, explaining them as we go, and offering our professional opinions on why we recommend the things we do.



In **section 2.3.3** we discuss your choices with URL structure...

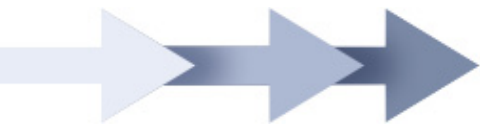
In **section 2.6.1** we go about choosing the perfect theme...

In **section 2.7.4** we go through some plugins that can be useful for any site...

1.1.4 How to Implement Advanced Functionality

We are going to go through some things that you can do with WordPress that are above and beyond the standard call of duty. For example, we'll cover how to use WordPress as a "full-blown Content Management System (CMS)," and try to dispel the myth that WordPress is just another blogging platform. We'll explore things like using page templates and page hierarchies, dynamic tiered menus, custom fields, and even fancy stuff like shortcodes.

Of course no discussion of advanced WordPress functionality can happen without discussing plugins. The flock of geniuses that is the WordPress community is responsible for an ocean of plugins that extend and advance the built-in functionality of WordPress. We will discuss quite a few of these plugins (all throughout the book), and even share the basics of writing your own.



In **section 5.5.1** you'll learn about using WordPress as a full blown CMS

In **section 5.6.1** you'll get a long list of plugins to check out for extending WordPress' functionality

In **section 6.6.5** you'll learn how to create custom RSS feeds of your content

In **section 7.4.4** you'll see how to style comments in special ways for your authors and other members

1.1.5 How to Optimize and Secure WordPress

Out of the box, and with a full cup of common sense, WordPress is a reasonably secure software package. But if you are up to snuff as a web designer/developer, you understand the importance of security and know that it is up to you to do everything possible to keep your site safe and secure. There are a slew of precautionary measures you can take to help with security, and we'll show you all of them. Things like locking down sensitive files, checking file permissions, and using various manifestations of "security-through-obscurity."

While we are at it, we'll customize our sites to streamline and optimize the efficiency of WordPress. Many of the things that can make WordPress faster are also things that can make *any* site faster, like reducing HTTP requests and optimizing images. But there are many WordPress-specific things you can do as well, like caching data, keeping your database optimized, and reducing the number of database queries.



In **section 8.2.1** you'll learn to control potential duplicate content problems

In **section 9.1.5** you will learn how to secure your sensitive files

In **section 9.4.4** you'll learn how to back up your database, in case of the worst

Many Features

To learn about great admin features such as the Theme Customizer and Media Library, check out the supplemental PDF (available in the DigWP Members Area: <http://digwp.com/u/638>)



Upgrade, Upgrade, Upgrade

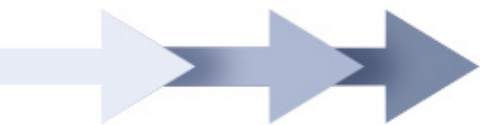
Around the time of the writing of the first version of this book, a serious security bug swept through the WordPress world. Any version of WordPress prior to 2.8.4 was affected by it. The exploit was able to create an additional administrator account inside WordPress giving access to do just about anything with your site. The bug was fixed, users just needed to upgrade!

WordPress' popularity makes it a big target for bad guys. But that same popularity ensures fixes happen fast and security gets better and better. We will go through lots of security measures in chapter 9.

The big take-away here is: **make sure to upgrade WordPress as soon as you can when new versions come out.** It only takes a few clicks.

1.1.6 How to Maintain Your WordPress Site

As time goes by and your site grows and matures, there are steps to be taken to make sure your site remains happy and healthy. It should go without saying that you should keep your WordPress core up to date with the latest version, and that also means keeping your themes updated with the latest functions. As a top-notch web star, you'll of course also want to keep an eye on your server logs, fight against 404's and broken links, and keep up with your moderation duties.



In **section 8.5.4** you'll get an intro to Google Analytics and other metrics you can monitor on your site

In **section 9.2.3** you'll start to learn how you can fight against comment spam using WordPress' built-in features

In **section 9.3.3** you'll learn about a plugin to watch for broken links on your site

1.1.7 Don't Worry

We sincerely hope none of this sounds daunting or over your head, because it's not. WordPress is easy, especially compared to its beastly competitors.

1.2.1 So, You've Never Heard of WordPress

If this is your first shindig with WordPress, there are a few things that may not be instantly obvious. If these things *are* obvious, good, you're ahead of the curve, and remember there was a time in all our lives these things were *not* obvious.

Static vs. dynamic sites

Your old Geocities page? That was a *static* site. Your new WordPress powered site? That is a *dynamic* site. Dynamic sites have content that is stored in databases and pulled out and displayed as instructed by templates. This is a powerful way to build

sites. It may seem complicated at first, but the flexibility offered by dynamic sites is well-worth the price.

Dynamic sites are...

- **Better for designers** – templates make it far easier to make global changes to a site.
- **Better for developers** – having content in databases allows them to get you what you need when you need it.
- **Better for site managers** – it puts publishing into their hands with tools meant for normal people to use.
- **Better for visitors** – because the site you create with it will be fantastic!

Self-Hosted ↓ 

WordPress.org

This book is about the self-hosted, download-and-install-it-yourself version of WordPress. This is the WordPress to which WordPress.org is dedicated, enabling you to download the latest version, participate in the forums, and browse the Codex (the extensive and official WordPress documentation).

This is the version that is for “real” designers and developers, as it offers you absolute and total control of your site, for use on your own servers, with no usage restrictions at all.

Not-Self-Hosted ↓

WordPress.com

Perhaps introducing slight confusion, there is also a WordPress.com, which is a hosted blogging service owned by the same parent company (Automattic). Sites hosted at WordPress.com are located at URLs like “yoursite.wordpress.com” and are largely used only for blogs. There, you have less control, can’t do things like use plugins, and cannot deliver your own advertising. Of course, it’s also free, easy, and uses essentially the same Admin as self-hosted WordPress, so it’s great for your Mom; but even so, we’ll be using the *real* WordPress for our websites!

1.2.2 One Template, Many Pages

If you are a web designer, you're probably already at least a little familiar with Cascading Style Sheets (CSS). It's such a beautiful concept. A single CSS file controls the design for lots of pages. So if one day you decide that your dark-red background color should be a little lighter red, you can make one little edit and cause a sweeping design change to your site (as opposed to opening hundreds of pages and making the change on each one). That's what a CMS like WordPress does, only instead of abstracting design away, it abstracts the content away.

With a template system like WordPress, you can make a change to the template and change the way all the pages that use that template are displayed. Perhaps you want to display the date above each Post title instead of below? No problem, just change the template. In this day and age, HTML isn't for actual content, just structuring content.

1.2.3 Powerful, Flexible and Extensible

You can witness the power and flexibility of the template system very easily, because WordPress allows you to quickly change the entire template you are using with just a few clicks. Find a new theme you like while browsing the web? You can upload it to your own site and activate it in seconds. All your content will then flow into this theme auto-magically! It's a wonder to behold, but remember not to change your theme too often or your visitors will be lost and confused.

The same things that make WordPress themes *flexible* also make them *extensible*. For example, adding new content to your sidebar is likely as easy as opening your `sidebar.php` and adding in the new stuff. Your theme might even be widget-ready (see chapter 4.8.1), meaning you can add, alter, or remove content and functionality from your sidebar (or any other widget-ready area) without looking at any code at all.

Appropriate Uses

CSS = Design

Database = Content

HTML/PHP = Get &
Describe Content

JavaScript = On-page
Functionality

Don't Cross the Streams!

1.3.1 Key Components of a WordPress Site

There are some things that need a bit of explaining before we get much deeper. For lack of a better term, we'll call them "components." Components represent the main parts of a WordPress site, including the actual files that make things go, and the database that houses all of the data. Together, these two components – the files and database – generate the web pages for your site. Once generated, there are two central types of these web pages, the Admin area that only you see, and the public pages that the whole world can see. Let's take a closer look at each of these different components.

1.3.2 WordPress Core Files

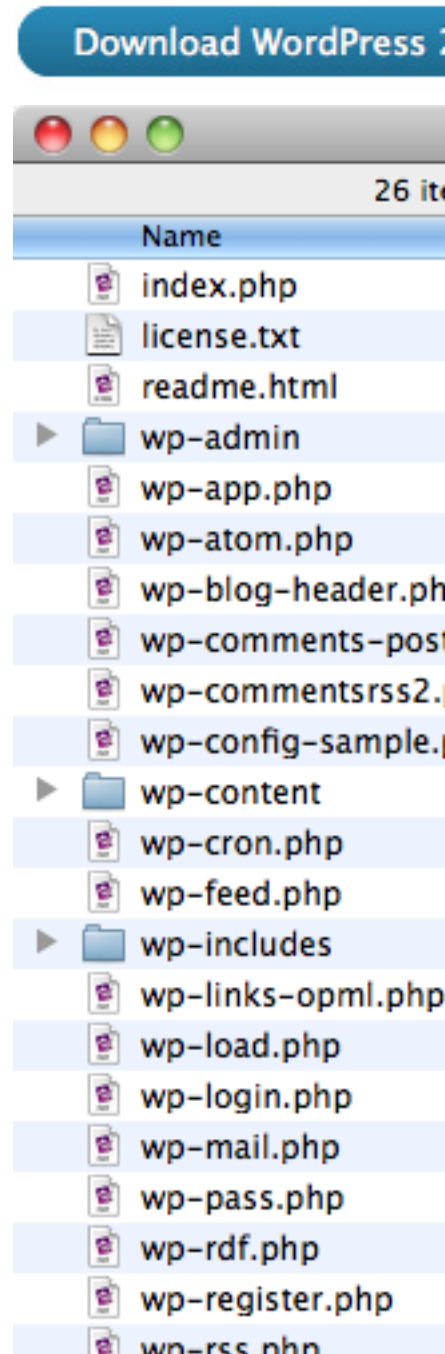
When you go to WordPress.org and download a copy of WordPress, you are downloading the WordPress **core files**. It's a big folder full of files and a few subfolders full of even more files. Once you download and unzip the WordPress package, you'll notice that most of the file and folder names begin with a "wp-" prefix.

The WordPress core files are the things that make WordPress tick, the things that make WordPress a complete software package. Most of these files you'll never touch. A few of them you'll touch very rarely, and a certain subset of these files you'll be in and out of all the time, especially when building a theme.

1.3.3 The WordPress Database

To get WordPress installed and running, you will need to set up and configure a MySQL database. The WordPress core files don't actually contain any of your site's content, they merely provide the functionality required to display the content stored in the database. All of your blog posts, plugin settings, and site options are contained in the database.

This can be a bit of an abstract concept, especially since the database is something

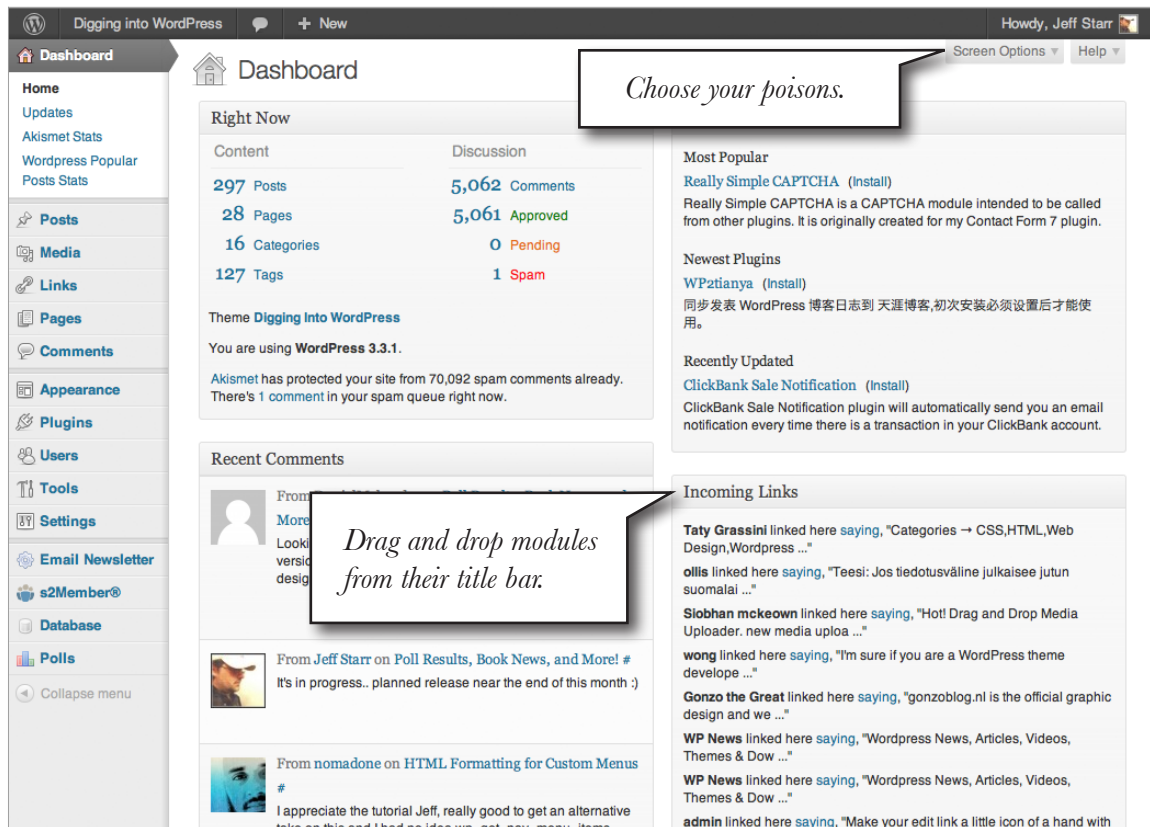


Database Backups

For information about backing up your database, see chapter 9.4.4.

you may never actually see. It's not even a "file" on your server that you could see and download (unless it's a backup copy). It just kind of lives in the cloud that is your server, humming along and doing its thing.

Despite being this elusive and abstract entity, the database is arguably the most *important* part of your WordPress site. All the Posts, Pages, Revisions, Comments, Users, and all of your other content and settings reside exclusively within the database. Even if you had a total server meltdown and lost *everything*, with a backup copy of the database you would be okay. You could reinstall WordPress, find a workable theme, and be back up and running in minutes while you worked on rebuilding the design. On the other hand, if you also lost the database, you would have lost every single bit of content on your site forever. In other words, **don't lose your database!**



This is what the "Dashboard" of the Admin area looks like. There is lots of functionality, information, and navigation here. Fortunately it is beautifully designed, and feels natural to use after a short learning curve.

The Dashboard can also be customized. Eliminate things you don't use through a simple dropdown and checkboxes. Then rearrange the "modules" by simple drag-and-drop.

1.3.4 The Back End

The back end of WordPress, hereinafter known as the Admin area, is the part of WordPress that is seen only by you, your co-authors, and your site administrators. You view this area directly through a web browser and it is used to create and control all of the content and otherwise manage the site. This is essentially a secret hidden area which normal visitors will never see and likely don't care about.

1.3.5 The Front End

The end result of these various WordPress components is the part of your site that visitors actually see and care about: the front end. The front end of your WordPress-powered site consists of all your site's publicly available web pages. Posts, Pages, Archives, everything.

So let's put it together and see how the front end is generated. First, the content you create in the Admin area is stored in the database. Then, the core files interact with the database to render the website for your visitors. The front end is where WordPress brings the magic together and makes it happen.

1.4.1 Tools of the Trade

You are going to need an internet connection. Shocking, we know. What else?

1.4.2 A Domain Name

Since we are working with the self-hosted version of WordPress, we are going to need an environment to work. That's what "self" means – bring your own environment. The first step is getting a domain name (digwp.com = a domain name). If you've never gone through this process before, don't worry it's really not too big of a deal, despite the often horrendous user-interface of many of the major retailers. GoDaddy.com is a popular choice for purchasing domains.

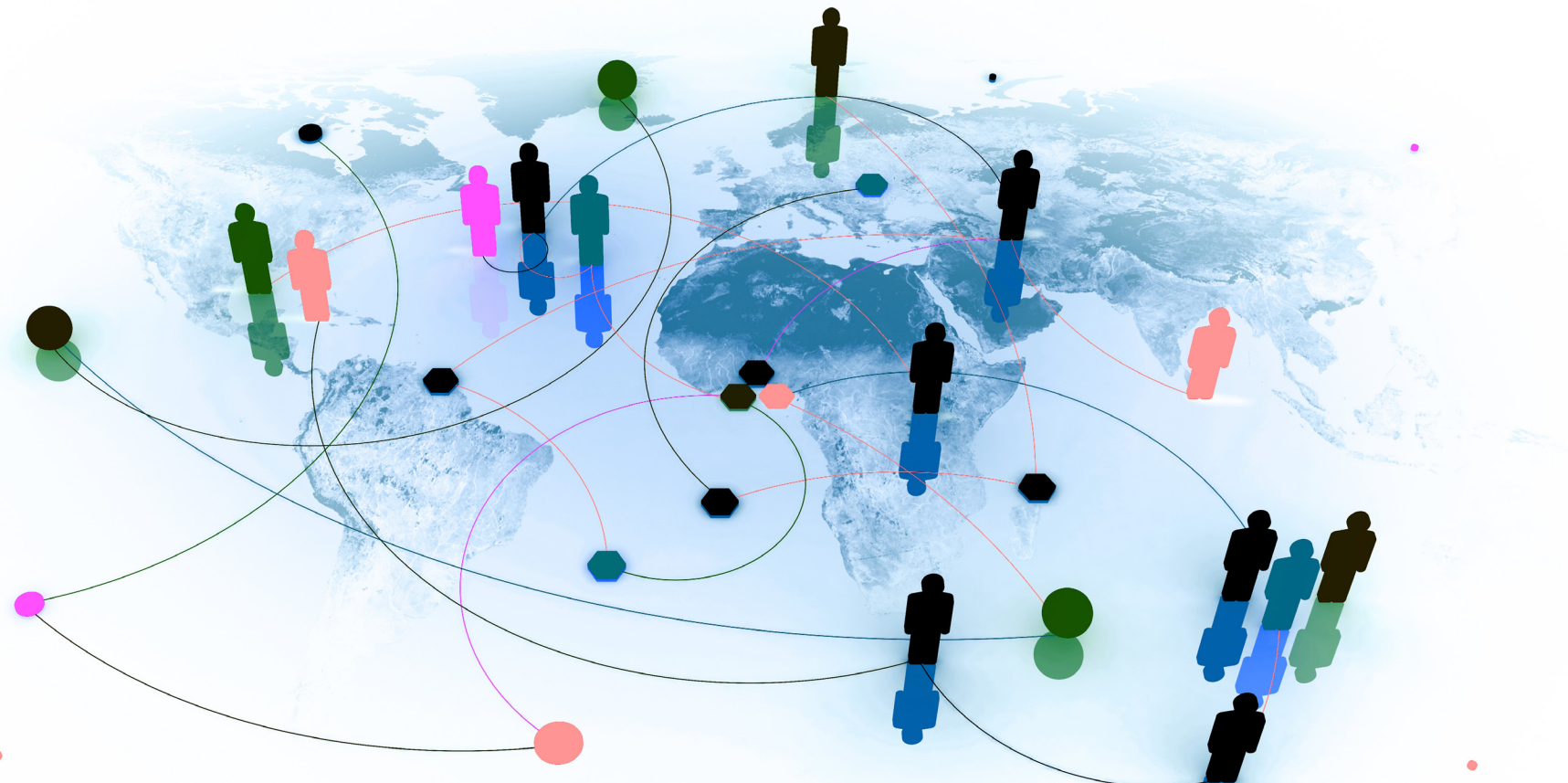
Real Estate

Owning your own domain name is like owning your own house. You don't have a landlord telling you that you can't knock down that wall or you can only put up posters with poster putty. With your own site, you can do whatever you want.

1.4.3 Web Host / Server

Owning a domain name is half of the equation. Now you need a web server to point that domain toward. The web server will then do its thing and serve up your website. You don't need to buy your domain and hosting at the same place, and in fact, we advise against it. For example, a hosting company doesn't have a whole lot of incentive to offer you support in moving your hosting to a different server, should that ever become necessary.

Hosting is more expensive than domain names, but for low to medium traffic sites, even basic hosting plans are adequate. Digging Into WordPress served 150,000 pageviews per month at the time of this writing. It is hosted on a \$20/month Media Temple plan alongside many other sites, runs great, and will for a while to come.



1.4.4 Text / Code Editor

Your FrontPage / GoLive days are over. WYSIWYG editors will be of no use to you while building dynamic, WordPress-powered websites. You are better than that anyway. We are going to get our hands dirty with real code, so you need to be using a real code editor. Here is a summary of some of the better code editors currently available:

Mac	TextMate	\$59	http://digwp.com/u/231
Mac	TextWrangler	Free	http://digwp.com/u/232
PC	UltraEdit	\$49	http://digwp.com/u/233
PC	Notepad++	Free	http://digwp.com/u/234
Both	jEdit	Free	http://digwp.com/u/235
Mac	Smultron	\$5	http://digwp.com/u/592
Linux	Geany	Free	http://digwp.com/u/593

1.4.5 FTP Program

To connect to your web server and transfer files back and forth from your computer, you'll need some FTP software. If the program transfers files, you are good to go, but some also have swell features you may be interested in. You make the call.

Mac	Transmit	\$29	http://digwp.com/u/236
Mac	Fetch	Free	http://digwp.com/u/237
PC	AutoFTP	Free	http://digwp.com/u/239
Both	FileZilla	Free	http://digwp.com/u/240
Both	FireFTP	Free	http://digwp.com/u/414
Both	Cyberduck	Free	http://digwp.com/u/589
PC	WinSCP	Free	http://digwp.com/u/590



Double Cool

Coda for the Mac is an FTP client and code editor rolled into one. It also has a built-in terminal, reference manuals, code sharing via Bonjour, and subversion support.

The code editor may not be as robust as some of the others, but the combo functionality is pretty sweet.

<http://digwp.com/u/18>

Chris' OS X dock



- ☒ Computer with Internet
- ☒ Domain Name
- ☒ Web Hosting
- ☒ Code Editor
- ☒ FTP Software
- ☐ Mad Skillz

...we're getting there!

John Boardly's "I Love Typography" is a beautiful WordPress-powered site.

<http://ilovetypography.com/>

The screenshot shows a web browser window with the address bar displaying "http://ilovetypography.com/". The page has a navigation bar with links: HOME, SEARCH, TAGS, CONTACT, FREE, ABOUT, FONT GAME, TYPENUTS, SHOP, WLT, and T. The main heading is "i love typography" in a large, stylized font. Below it, the date "NOV 6 2009" and "[12 COMMENTS]" are shown. The article title is "Type: A Visual History of Typefaces & Graphic Styles", reviewed by James Puckett. The text of the review discusses the book's content, noting its focus on modern typography and its critical reception. On the right side, there are sections for "Popular articles" and "Recent articles".

fonts, typefaces and all things typographical — I love Typography (ILT)

HOME SEARCH TAGS CONTACT FREE ABOUT FONT GAME TYPENUTS SHOP WLT T

i love typography

NOV 6 2009 [12 COMMENTS]

Type: A Visual History of Typefaces & Graphic Styles

REVIEWED BY JAMES PUCKETT

I was excited when Taschen announced the first volume of *Type: A Visual History of Typefaces and Graphic Styles*, described as “*This exuberant selection of typographic fonts and styles traces the modern evolution of the printed letter*”*. Such language, including the title, is disingenuous, because this book is not a history. *Type* does contain a short essay by coauthor/coeditor Cees W. de Jong about type history, but it is poorly written and riddled with inaccuracies. Similarly bad are the captions that introduce each specimen. Many are obvious or inane statements such as “*It was an honor to have one’s name set on such a lovely publication.*” (p. 88), “*This printer from Amsterdam evidently had a lot of customers in the countryside.*” (p. 100) and “*Back in time! Two lines, two different typefaces in a fantasy world.*” (p. 243). Overall the writing feels like a draft rushed to press; it should have been fact checked and edited by a historian.

Popular articles

- How to make a font
- Best ‘fonts’ of 2008
- Type history series
- Web typography guide
- On choosing type
- Identify that font
- Who shot the serif?
- Sunday type: napkin type
- iFont, iPhone
- Web fonts — where are we?



Lost in the cloud? Unfluff your web hosting. (mt) Media Temple.

ads via THE DECK

Recent articles

- Type Camp
- Karbon Type

A complex system that works is invariably found to have evolved from a simple system that worked. The inverse proposition also appears to be true: A complex system designed from scratch never works and cannot be made to work. You have to start over, beginning with a working simple system.

— JOHN GALL

2

Setting Up WordPress

2.1.1 The Famous Five Minute Install

Installing WordPress is famously easy. Upload files, create database, update the wp-config.php file, and go through the online installer which is only a handful of very simple questions. The more times you do it, the faster you get at it. Five minutes, ha! You'll be down to two-and-a-half in no time.

The "How"

We didn't want to waste a bunch of pages explaining "how" to install WordPress. It's not complicated, and is covered in detail at the Codex:

<http://digwp.com/u/241>

2.1.2 Where To Install?

When you download the core files from WordPress.org, you end up with a .zip file sitting there on your computer. Unzip it, and you have a folder called "wordpress" that is full of files. One option is to upload the contents of that folder right to the root directory of your website and start the installation process. We suggest a slightly different approach.

Instead, rename that folder something strange and obscure, like "blackmothsuperrainbow" and upload that folder to the root directory of your site. Then you say, "But wait! I don't want my site's URL to be `http://mydomain.com/blackmothsuperrainbow/!`" Of course not, good sir, that would be strange and obscure. You definitely want WordPress to control the root of your site. In order to do that, just copy the index.php file and the .htaccess file from the blackmothsuperrainbow folder and paste into the root directory. If an .htaccess file doesn't exist, go ahead and create one — we'll be using it later on. Then open the index.php file and change this:

*Awwwwww.
Isn't it cute?*



First things first

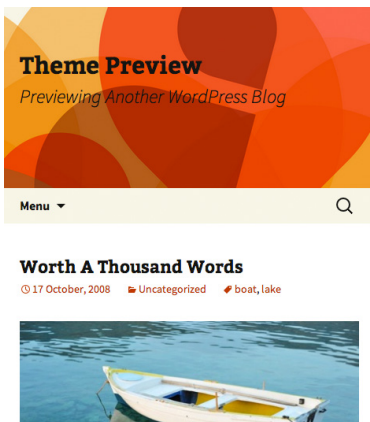
*If you are moving the WordPress core files **after** installation, change the Admin settings first, then copy/paste the index and htaccess files.*

For more in-depth information about installing WordPress in a subdirectory, check out these helpful resources:

<http://digwp.com/u/522>

<http://digwp.com/u/523>

The “default” WP theme used to be Kubrick, which was pretty gnarly. Getting rid of that was priority #1. WordPress now ships with the “Twenty Thirteen” theme, which is quite colorful.



```
require('./wp-blog-header.php');
```

...to this:

```
require('./blackmothsuperrainbow/wp-blog-header.php');
```

You'll now have to log in at <http://mydomain.com/blackmothsuperrainbow/wp-admin/>, but WordPress will be in control of the root just as if that were its actual location. Once you have installed WordPress and logged in to the Admin area, go to **Settings > General** and ensure that the settings for your WordPress address (URL) points to <http://mydomain.com/blackmothsuperrainbow/> and Blog address (URL) points to <http://mydomain.com/>. Of course, when choosing the name of your installation directory, you should use your own strange and obscure word.

Why go through these extra steps just to install WordPress? The benefits are twofold and significant:

- 1. Security through obscurity.** Any evil bots scanning and probing your site looking for possible WordPress exploits probably won't even be able to find your WordPress files.
- 2. It keeps your root directory clean.** Nothing worse than a messy root directory. Except for maybe a hacked site.

2.1.3 Checking Default Performance and Proper Functionality

After you have completed the installation and are looking at the Dashboard in the WordPress Admin area, you should take a little time to click around and make sure things appear and behave as expected. If you are already looking at the Dashboard and things seem normal there, click your site title in the upper left and visit the front end of your site. Does it load up and look like a website? When you click links, do they work and take you to the appropriate places?

Ninety-nine percent of the time, everything is going to be fine after a fresh WordPress installation. But even so, now would be the absolute best time to verify that everything is running properly, smoothly, and as expected. You want to check everything out now, because fixing things at this point will take much less work than later on in the game. Once we have verified that, yes, WordPress is operating beautifully, it's time to dig in a little deeper.

2.2.1 OK, I'm In. Now What?

First of all, you should probably crack a beer. You've successfully installed an incredibly powerful publishing platform and are well on your way to creating a killer website.

2.2.2 Just Publish Something!

We have the whole rest of this book to prod and poke at settings and alter code and nitpick the details. But none of that has any context unless you get your feet wet a little bit and start getting a feel for *how* WordPress works and how easy it makes publishing content. It's like learning to play the guitar. You can force yourself to play scales and learn chord voicings all day, but you'll be bored to tears and the information won't stick as well as it would if you have some context (a song) to attach it to. More fun **and** more effective, what a concept.

So why don't you click that "New Post" button right up at the top of your Dashboard. Then type yourself in a title in that top box, maybe write a few sentences about your cat, then hit that big blue "Publish" button.

2.2.3 Go Look At It!

Now go check your homepage and see the fruits of your last few seconds of labor. Brand new content, sitting right there for your next visitor to read. **BAM!** Feel the power. Does it feel good? We thought so. Now we need a plan.

Shock teenage gangsters

Someone submitted a code snippet to us with this title. It was just a simple .htaccess redirect of the wp-config.php file (only for use on NON-WordPress sites), but that name was too awesome not to publish. Check it:

<http://digwp.com/u/422>

2.2.4 The Plan

As we surf through the rest of this chapter, what we are really doing is developing a “plan” for your site. As you read through the following sections, envision how each setting, feature, or option would relate to the site you are building right now. If you aren’t building anything right now, perhaps think of a fictional site so you have some context. I like to think I’m building a site called “Aunt Bea’s Pie Site,” a site dedicated to Aunt Bea and her amazing pies.

Throughout the book we will also be peppering you with plugins that do this and that and may be of value to your site. There are *some* plugins though, that are universally useful to *any* WordPress site. We will cover these at the end of this chapter.

2.3.1 Permalinks: Your URL Structure

The default, out-of-the-box setting for WordPress permalinks is a bit gnarly. They look like this:

```
http://mydomain.com/?p=12
```

Why is that? Why doesn’t WordPress come with a better default setting? Well it’s because this setting doesn’t require any special server files or setup at all. The “?p=” part of the URL references Post and Page IDs (like little secret codes that are unique to every Post or Page). The value after the equal sign is the value for that particular parameter. So the link above tells WordPress to “retrieve and display the Post or Page that has the ID value of 12.” And, like man’s best friend, WordPress will obey.

As practical as these default URLs are, you would rather that your URLs looked more readable, like “http://mydomain.com/super-big-contest/.” With a URL like that, the server is by default going to look for a file or subdirectory named “super-big-contest” and take you there. But, with a dynamic platform like WordPress, that directory doesn’t actually exist! WordPress doesn’t actually create a physical directory structure for all of your posts and pages, it just “fakes it” with a little magic from a special little file called `.htaccess`.

.htaccess made easy

For a complete guide to .htaccess, check out Jeff’s new book, .htaccess made easy.

<http://htaccessbook.com/>

2.3.2 HTAccess

The full name for this file is literally “.htaccess”. It begins with a period and has no file extension. Weird, we know. Some operating systems treat files like this differently. For example, in Mac OS X, files that begin with a period are “hidden” from view. A default download of WordPress does not include this file. Yet, it is this file and its contents which make all special permalink handling possible in WordPress.

The .htaccess file should exist at the root directory of your WordPress install, but don’t run off and create one just yet. In many server environments, WordPress can do this automatically for you, which is the best way to go (less error-prone). Simply go into your **Settings > Permalinks** area and choose any setting other than the default and click “Save Changes.” If you received the following response message, you are all set:

Permalink structure updated.

This means WordPress was able to create the .htaccess file and add to it the proper code. If you did *not* get this message, you will instead get a box of code

Apache

.htaccess files are only applicable to Apache web servers. If you are running shared hosting that you purchased yourself, there is a good chance it’s Apache.

If you aren’t using Apache, you can still get somewhat decent permalinks. See the PATHINFO section here: <http://digwp.com/u/19>

Short URLs

A “short URL” is just like it sounds, a URL with very few characters. These have become popular because of services like Twitter which have a strict 140-character limit on messages, including URLs. Services like Bit.ly can create these for you and they look like this: <http://bit.ly/11q4xv>

There is always controversy with short URLs, despite their obvious usefulness. The URLs themselves provide no useful information to users. Worse, if the service were to close, there would be a ton of non-resolving URLs out in the wild. As such, rolling your own short URLs is getting more and more popular.

One cool thing about WordPress, is that the default URL structure is pretty short all by itself:

<http://digwp.com/?p=212>

You can get this link by just clicking the Get Shortlink button from the post editor.

DiW v3.0 Book News

Permalink: <http://digwp.com/2010/07/v3-book-news/>

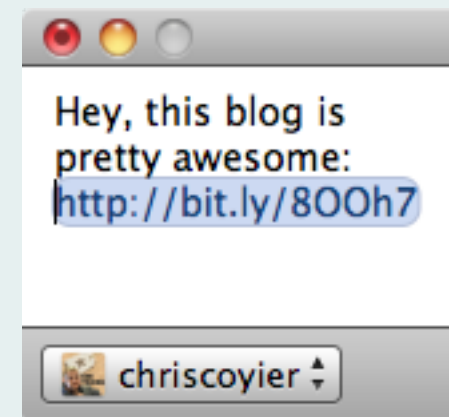
Edit

View Post

Get Shortlink

This book uses the Short URL plugin. Here is a strangely meta short URL to that:

<http://digwp.com/u/460>



Pop up tweeting box from the Mac app Tweetie.

%postname% woes

Using ONLY the postname in the permalink structure apparently has some performance issues:

<http://digwp.com/u/463>

Although to be honest, that's what CSS-Tricks.com uses and it has never seemed to be a big deal.

Update: After crashing the CSS-Tricks site, the performance issues with post-name permalinks is now fixed in the latest version of WP:

<http://digwp.com/u/588>

Read more about the changes in section 3.5.6 of the supplemental book.

Obscure Bug

If you use %category% to start your permalinks, pagination of those actual category pages may not work correctly. There is a plugin fix:

<http://digwp.com/u/33>

that literally tells you to update the .htaccess file manually. If this is the case, you will actually need to do the grunt work yourself. If so, don't panic. Simply create a new .htaccess file in the root directory of your WordPress install, and then insert the HTAccess code that WordPress provides. It's going to look something like this:

```
# BEGIN WordPress
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteBase /
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteRule . /index.php [L]
</IfModule>

# END WordPress
```

2.3.3 Which Style of Permalinks?

WordPress enables you to choose your own permalink structure. Although the format you pick for your permalinks is largely a matter of personal style, you should consider carefully the context of your site when making your decision. For example, is your site clearly a blog with frequent postings? Perhaps a "/%year%/%monthnum%/%day%/%postname%/" structure is best for you. Let's say you are planning to have a site full of record reviews. The date may not matter as much, so picking a structure more like "/%category%/%postname%/" makes more sense. Planning on very few posts with "timeless" content? Maybe "/%postname%/" is sufficient all by itself.

Examples of different permalink structures

At DigWP.com, we use a URL structure with partially dated permalinks:

<http://digwp.com/2009/06/redirect-category-search-and-tag-urls/>

At QuotesOnDesign.com, the date is less important and the permalink is simply the author of the quote:

<http://quotesondesign.com/felix-sockwell/>

At EnvisionCad.com, the posts are categorized into just a few major categories, so permalinks like this make the most sense:

<http://envisioncad.com/training-dates/residential-grading/>

2.3.4 Pick One and Stick With It

Once you have decided on an optimal permalink structure, do your site a favor and stick with it for the life of your site. People use these links to link directly to your site, and up and changing them down the line isn't a good idea. While modern versions of WordPress are typically smart enough to get modified URLs redirected to the right place, it doesn't look nice in people's browsers and isn't doing any favors for your search engine optimization (SEO) efforts.

2.3.5 SEO Consideration: Mind Your Post "Slugs"

As it is called, the "slug" of a WordPress URL is a special string of characters that represents each post in a URL-friendly way. To illustrate, if you write a post called "57 Ways to Cook a Gizzard, Plus One Way You should Definitely not Cook a Gizzard," the slug that WordPress will auto-generate for your post will look like this:

57-ways-to-cook-a-gizzard-plus-one-way-you-should-definitely-not-cook-a-gizzard

That is going to be one heck of a URL if you use `/%postname%/` in your permalink structure. Most research suggests that incredibly long URLs like this are undesirable for SEO. Fortunately, you aren't stuck with your slugs! After your post has been saved at least once, you'll see a small area right underneath the box where you enter the post title in the post-editing screen. There, you can change the post slug

Optimizing Permalinks

For more information on how to create and optimize your site's permalinks, check out chapter 8.3.1, Optimizing Permalink Structure and also this comprehensive overview on our own blog:

<http://digwp.com/u/461>

Old Slugs

There are plugins out there that help handle redirecting old links to new links should the slug of the page change or the permalink structure change. That's awesome, but just not changing links is even better.

Cool URLs Don't Change

Don't believe us? Here it is straight from the horse's mouth:

<http://digwp.com/u/383>

But really, we're sure your mother is very good looking

to anything you would like, but remember to avoid any strange characters. Stick with lowercase letters, numbers, and dashes or underscores. Here is a much better “slug” for our example:

Good: <http://gizzards.com/how-to-cook-a-gizzard>

Bad: <http://bad-jokes.com/really-dumb-jokes/10/2008/your-momma-is-so-fat-she-sat-on-a-dollar-and-made-four-quarters>

But hey, even at its worst, WordPress isn't that bad. It certainly could be worse.

Really Bad: <http://www.surfboardsforyou.com/boards/filename.php?id=F98ZF4&productId=39222&section=wicked&brand=582&template=532>

Takeaway tips for creating optimal permalinks:

- Short is good
- Keep the overall URL structure relevant to the site
- Make the slug a memorable representation of the post

2.4.1 Categories and Tags

WordPress Posts and Pages will be the new home for loads and loads of your content. We already know that each of them has a unique ID, slug, and date stamp. That helps a bit in terms of organization and archiving, but WordPress has better methods in store for us. There are no pre-set categories or tags in WordPress. For example, the default “Hello, World!” post in WordPress is by default categorized as “Uncategorized” and has no tags. Thus, you'll need to create your own set of categories as you begin working with WordPress and as needed anytime thereafter.

In the old days, there was no way to “mass” assign or remove categories so adding a new category that old content fit into was a matter of going back and editing lots of posts one-by-one. Nowadays, you can click the checkboxes on the Edit Posts screen, check multiple boxes, and select **Edit > Apply** and get a screen for mass editing. Still, we generally believe you should put some thought into your category

Always There

Keep in mind that, even when you can't see them or don't use them in your permalink structure, IDs, dates, and other post- and page-specific information is always associated with your posts and pages from within the database.

plan early on, just to help make sure things don't get out of hand later on.

In addition to categories, each post may also be assigned any number of *Tags*. Tags are used to further categorize content at a more specific level. A real world example will serve us well here, so let's take a look at a Post on the DigWP.com website:

Title: Custom WordPress Title Tags

As you may have guessed, the content of this post is about how to create great titles in WordPress (see chapter 3.3.4). You might think we'd categorize this as "WordPress," but in fact our entire site is about WordPress so that would be quite pointless. Instead, we have categorized this post as "PHP," since it specifically deals with PHP functions to accomplish what we are writing about. "PHP" is the primary subject of this article, hence its designated category, but it isn't the *only* subject that the post talks about. Thus, we specify several Tags to indicate that content also contains information about "headers," "titles," and "tags." Indeed, these are the Tags chosen for that article.

If that example is too specific and self-referential, think of the example of a photograph of a tree. You would categorize this as "Photograph," or if your site was all photographs, "Tree." You then might tag the photo as "fall," "leaves," "sunset," and "beautiful" – assuming those things accurately described the photos.

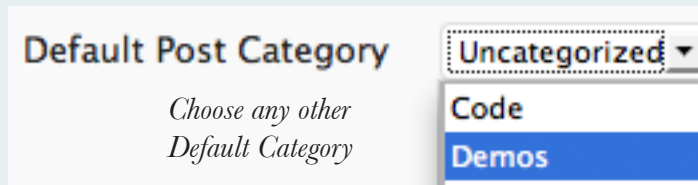
Remember that the purpose for all of this categorizing and tagging is ultimately to assist visitors (and yourself) in navigating your site and what is sure to be a huge amount of content. The larger your site grows, the more difficult it becomes to organize your content such that it is easily findable. Categories and tags offer your visitors a way to navigate your content in a conceptually logical way. For example, if the viewer

Eliminating "Uncategorized"

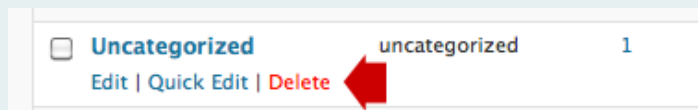
WordPress comes with a single post already published on your site when you install it: "Hello, World!" Because all posts need to have at least one category, it is assigned to the default category, "Uncategorized."

Uncategorized is rather unsightly, but thanks to a recent WordPress update, it's easy to remove.

The trick is assigning a *new default category*. Just log into Admin, go to your **Writing Settings**, and select any other default, like so:



After saving your settings, head on over to your **Category Settings**, where you should see the "Delete" link now appearing beneath the Uncategorized category.



And BAM, goodbye Uncategorized. You could also just rename it and use it for something, but it's nice to be able to finally delete it.

☐ Uncategorized	uncategorized	1
☐ WordPress	wordpress	0

Old WordPress: No Delete Checkbox for Uncategorized

Function Differences

While it's true that tags and categories are largely interchangeable, there are some functions which use them in different ways. For example, it is easier to exclude a category from the loop with WP_Query than it is to exclude a tag:

```
new WP_Query('cat=-3');
```

See chapter 4 for more about working with custom loops.

of our “Custom WordPress Title Tags” post was interested in reading more articles about PHP, they could click on the link for the “PHP” category to see a list of more posts on the topic.

2.4.2 They are Basically the Same

Categories and tags are treated slightly different in the admin panels. Tags you type in one by one (or choose from the tag cloud) whereas Categories you pick from the list of checkboxes. Don't be fooled though, on a purely functional basis Categories and Tags have no significant difference. The URLs that point to their respective archives, however, are slightly different:

<http://digwp.com/category/php/>

<http://digwp.com/tag/header/>

Each of these URLs commonly use the same theme file (archive.php) to display a list or summary of each of their included posts. These URL structures are able to be customized (go to **Settings > Permalinks**), but they will by default use the same archive.php theme file to display the posts that belong to the specific Category or Tag. The way in which these posts are displayed on the front end depends on the actual contents of your theme's archive.php file. We'll elaborate further on this concept in the next chapter.

2.4.3 Use Only One Category Per Post

It is generally considered best practice to only categorize each of your Posts with *one* category. This will serve you in two ways. First of all it is less confusing for people browsing your content by category, as they won't find any cross-over and potentially confusing situations where they aren't sure if an article they found in two places is really the same or not. The other is potential duplicate content issues, where search engines may find your post in several different places and think you are trying to game the system by adding lots of similar content all over.

Another advantage to using single categories is that you may wish to style posts in different categories in different ways. The most sure-fire way to accomplish this is to make sure Posts only have one so you don't have to deal with conflicts.

Duplicate Content

We show you how to control duplicate content in chapter 8.2.1.

2.4.4 Use Multiple Tags Per Post

The default wisdom for tags is to use them to identify sub-topics or minor points of interest inside the Post. So let's say you were publishing a Post that was a review of a 1969 Martin Acoustic Guitar. If your site was based on music in general, this Post might be categorized as "Instrument Review," but with tagging, we can get more specific. We might tag this "martin", "1969", and "guitar". Then later you write a review of The Who's first rock opera, *Tommy*. This might be categorized as "Album Review", and then tagged as "the who", "rock opera", and... "1969". So these two posts share the same tag, "1969".

Now we are starting to build some fun navigational possibilities for our users. If we build some Tag-based navigation, people could click the 1969 tag and see all the Posts of things related to that year. This will get more and more interesting as you continue to publish and tag, and may even open up some connections that you didn't think about.

2.4.5 Don't Go Overboard!

The whole point of using Categories and Tags is to assist human beings in navigating your site in intuitive ways. If you have 350 different categories on your site and use all of them lightly, this is well beyond the point of useful scanning and browsing. Our advice is to keep the number of your categories around 15 or less and that of your tags at around 100 or less.

No Categories

CSS-Tricks.com doesn't use any categories or tags at all. There are dated archives, but otherwise relies heavily on search to allow people to find things they are looking for. Nobody seems to mind.

ACTION!

PopCritics.com is a very awesome example of custom taxonomies in action. It's a bit like IMDb in that everything is cross-referenced... only the design is cleaner and the URLs are nicer.



2.4.6 You Don't Need to Use Them At All

Keep in mind that you don't need to categorize or tag your Posts at all. It is very common practice and can be very useful, but it is not required. This is a classic example of considering your audience over the technology. If you think it would mostly be clutter and doesn't make sense for your site, just don't bother categorizing or tagging at all. You can easily remove any referencing links that might appear in your themes.

2.4.7 Custom Taxonomies

Remember how we just said categories and tags are basically the same? They are. They are two different "taxonomies," that is, ways to classify a post. It is a somewhat of an advanced technique, but WordPress allows you to create additional taxonomies for classification purposes.

A perfect example is a site that was built to review movies. Movie reviews are classifiable in so many ways: by Date, Actors, Genre, Director, Producer, and the list could go on and on. Categories and Tags alone are insufficient to create the ideal navigational structure for this site. What we could do is create new taxonomies for all these different classifications. Behaviorally, and interface-wise, they behave just like tags.

Now take a practical example, a movie review for "Sleepy Hollow." Here is a summary:

Post Title: Review of Sleepy Hollow

Category: Review

Tags: Ichabod Crane, New York, Gothic

Actors: Johnny Depp, Christina Ricci

Producer: Scott Rudin, Adam Schroeder

Director: Tim Burton

Genre: Horror/Suspense

Permalink: <http://mymoviereviews.com/review/sleepy-hollow/>

The idea is that you can attach all this data to the movie review in the same normal, natural way as you would attach ordinary tags. In order to do this, you need to “register” some new taxonomies, which you can do by adding some code to your `functions.php` file.

```
function create_my_taxonomies() {  
    register_taxonomy('actors', 'post', array(  
        'hierarchical' => false, 'label' => 'Actors',  
        'query_var' => true, 'rewrite' => true));  
    register_taxonomy('producers', 'post', array(  
        'hierarchical' => false, 'label' => 'Producers',  
        'query_var' => true, 'rewrite' => true));  
}  
add_action('init', 'create_my_taxonomies', 0);
```

With this code in place, now you’ll see some new tagging boxes show up to the right of the content box when creating a new Post in the Admin area. You use them in the exact same way as tags.

What we get now, automatically, are new URLs that go along with our new taxonomy:

<http://mymoviereviews.com/actor/johnny-depp/>
<http://mymoviereviews.com/actor/christina-ricci/>
<http://mymoviereviews.com/producer/scott-rudin>

Post Tags

Separate tags with commas.

[Choose from the most used tags in Post Tags](#)

Actors

Separate tags with commas.

[Choose from the most used tags in Actors](#)

Producers

Separate tags with commas.

[Choose from the most used tags in Producers](#)

NOTE

Custom taxonomies are only available in WordPress 2.8 and above and significantly improved in 2.9 and 3.0

We discuss WordPress Taxonomies further in Chapters 5.5.4 and 3.2.5 in the supplemental book.

<http://mymoviereviews.com/producer/adam-schroeder>

<http://mymoviereviews.com/director/tim-burton/>

<http://mymoviereviews.com/genre/horror-suspense/>

These URLs would use the `archive.php` file to display all Posts that are tagged in that way. In order to create a truly cross-referenced system like this (hey, we could build our own IMDb!) we'll need to know how to display this custom taxonomy information in our themes. It's a little different than using regular tags.

Display a custom taxonomy tag cloud in your theme

If you would like to display a “tag cloud” containing all of the items in a specific, custom taxonomy, insert the following code anywhere in your theme files:

```
<?php wp_tag_cloud(array('taxonomy' => 'people', 'number' => 45)); ?>
```

Remember to edit the “people” term to match the name of the desired taxonomy. You may also want to edit the “number” parameter, which specifies how many tags are to be shown. To display **all** tags from the “people” taxonomy, use “0”.

Query for posts using a particular custom tag

To display only posts that have a specific tag from one of your custom taxonomies, place the following code before the loop in one of your theme files:

```
<?php query_posts(array('people' => 'will-smith', 'showposts' => 10)); ?>
```

You display posts belonging to any tag from any taxonomy by simply editing the taxonomy name, “people”, and the tag name, “will-smith”. Lastly, you can choose to display any number of posts by editing the value “10”. Remember that this code must be placed before the loop in order to work. We'll be getting to the loop in chapter 4.

List values of custom taxonomy on a per-Post basis

This code is useful for displaying all of the terms associated with a particular taxonomy for a given post. Simply place the following snippet where you would like to display the comma-separated series of tag links:

```
<?php echo get_the_term_list($post->ID, 'people', 'People: ', ', ', ', ' '); ?>
```

For each of your posts, the output of this particular example will display all of the tags associated with the “people” taxonomy. You can change this up to any taxonomy you want by editing the first instance of the term “people” in the code. The last three parameters tell WordPress what to place before the tag links, between each link, and after the tag links, respectively.

2.5.1 Users and Administrators

If you have installed WordPress, then you have set up at least one user. Each user is basically an “account” on WordPress. When you go through WordPress installation, you pick your own username and password. That username and password is one user, but you are not limited to that. You can manually add new accounts, assign privileges, and even enable visitors to register as users themselves.

Even if your theme doesn't display author names on the site itself, author names are still used in the RSS feed. Using a properly displayed name is a nice touch when reading through RSS. If you don't want to use your real name, even something like Site Manager is nicer looking than the default “admin.”

★ Global Custom Fields »

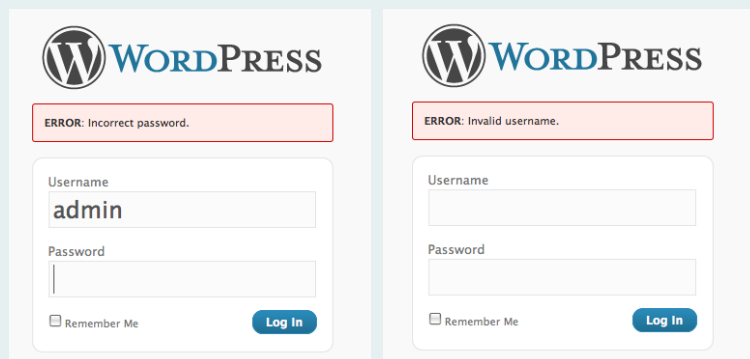
by Chris Coyier

😊 [1 person liked this](#)

UPDATE: Make sure to take out “[Take Two](#)” on this cor

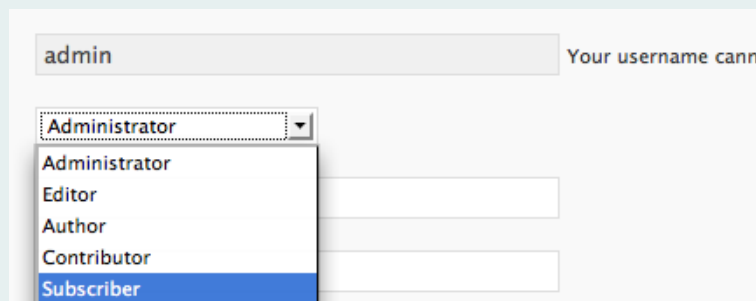
Custom fields allow us to attach data to Posts or Pages. Custom fields are an amazingly flexible and single-handedly allow WordPress Custom Fields used on single Posts can be limiting in some circumsta

Neutering the Admin Account



Hey, thanks!

Too much information?



WordPress tries to be as helpful as possible on its login screen. Like any good web application, it tells you when you have an incorrect password. WordPress does something else though, something that many people feel gives away too much information. It tells you if the username you are attempting to use exists or not.

The problem here is that anyone can access your login screen, try to log in with the "admin" username, and be awarded with the knowledge that the user does indeed exist, even if they get the password wrong. Prior to WordPress 3.0, "admin" was the default username and had full privileges so breaking into that account would be ideal for them.

We suggest leaving an "admin" account active, but neutering its privileges. That way, you can keep the baddies guessing, and even if they do break in, they get nothing.

If your account is currently "admin", create a new user with a new username, then delete the current "admin" account being careful to attribute all posts to the new account. Then create the "admin" account again, only make it a "Subscriber" level user, which has no add/edit/delete privileges.

Extremely Secure Passwords

Your Login page is a public-facing portal to your Admin area. Obviously, if your login was somehow compromised, an intruder could do serious harm to your site, damaging themes, deleting content, and worse. A brute-force password hack is probably the rarest of ways your site could be compromised, but you should still have an extremely secure password

- Don't use something obvious like "password" or "1234"
- Use a combination of letters and numbers
- Shoot for 8 characters or longer
- Don't use the same password you use for anything else
— This one is just *too* important!



2.5.2 Add a New Account for Yourself

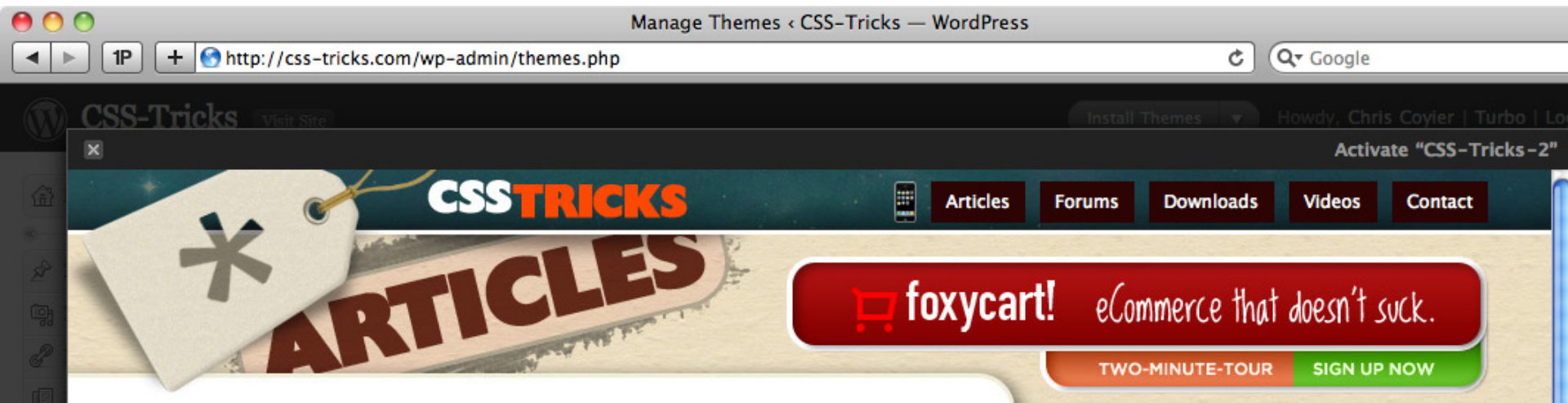
One thing that you cannot ever change, once an account is created, is the username. That means that the “admin” account will have a user name of “admin” forever. We recommend not using this account as your regular account. Instead, set up a new account for yourself right away, using a username that is more memorable and specific to yourself, but of course with an extremely secure password.

To do this, go to **Users > Add New**, fill out all the required fields, and be sure to choose “Administrator” as your role. Then log out and back in again with your brand new account. Now go back to **Users > Authors & Users**, hover over your existing account, and click “Edit.” You will now enjoy some additional options that weren’t available when you initially created the account.

With your new account, you can cosmetically change the look of the Admin area by selecting a new color scheme. More importantly, you can change public-facing details about your profile. We suggest changing your “Display Name” to something more sightly, like your real name, for example. You can also edit your biographical information, preferred URL, and other personal details. This information may then be displayed on your web pages by calling the information from your theme files.

Old Themes

As you redesign your site over the years, you should leave your old themes in the wp-content folder. It's kinda fun sometimes to go throwback. Maybe your blog could dress up as a previous version of itself for Halloween.



2.6.1 Choosing the Perfect Theme

With everything that you will learn about WordPress from this book and elsewhere, it is our hope that you will develop your own themes. Creating your own theme enables you to get everything looking and working exactly how you want it, right down to the last detail. But certainly, there is no shame in using a pre-made theme as you begin your journey with WordPress. After all, many top WordPress developers (including us!) used pre-fabbed themes as a starting point. Using someone else's theme is an excellent way to dig into the process of customizing and eventually building your own.

2.6.2 Where to Find Awesome Themes

"There are two kinds of themes in this world," my daddy always told me: "free themes, and paid themes." Paid themes often call themselves "premium" themes. In general, paid themes are going to have nicer designs, be coded a little better, and may offer fancy theme options. But then he also warned me: "there are some amazing, high-quality *free* themes, and there are some downright-bad *paid* themes." We can give you some general things to look for, but you are just going to have to use your best horse-sense when making the final call.

Free theme resources

- **WordPress.org Theme Library** - <http://digwp.com/u/20>
Straight from the motherland. Themes ahoy!
- **The Mighty Google** - <http://digwp.com/u/21>
Searching Google will work long after this book has been recycled.
- **Digging Into WordPress** - <http://digwp.com/u/384>
Besides the ones that come with this book, we offer some free themes in our Theme Clubhouse.
- **Smashing Magazine** - <http://digwp.com/u/22>
You might need to use their search form, but they offer a number of nice high-quality free themes.

Places to buy premium themes

- **ThemeForest** - <http://digwp.com/u/385>
- **WooThemes** - <http://digwp.com/u/386>
- **ElegantThemes** - <http://digwp.com/u/387>
- **UpThemes** - <http://digwp.com/u/561>

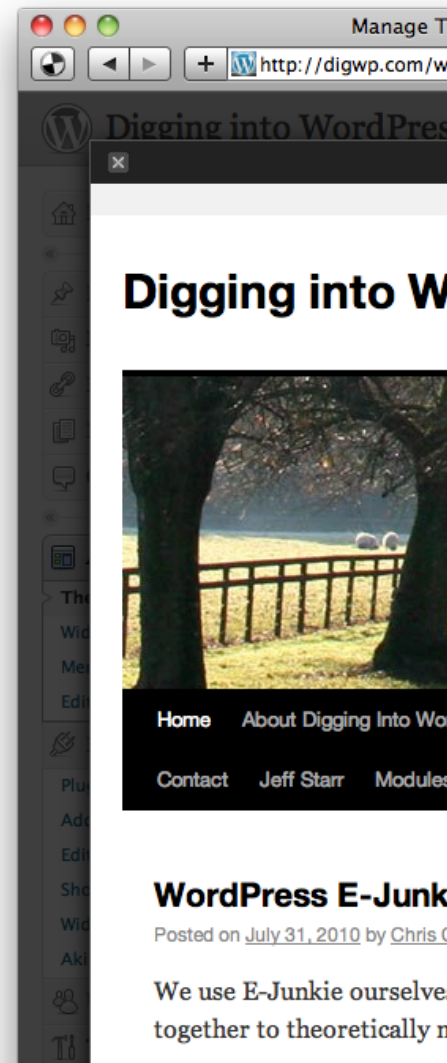


2.6.3 Previewing Themes

A lot of themes you find around the internet will be accompanied by a demo, enabling you to see how the theme looks and works before you commit to it. But even if the theme *doesn't* have a demo, you can demo it yourself, on your own blog. Upload it to your `/wp-content/themes` folder, and navigate to the "Appearance" page of your Admin area. There, you find thumbnails and descriptions of your site's currently available themes. Locate the theme that you would like to preview from among the crowd and click on its thumbnail. A popup window will then show you what your site will look like when running that particular theme. You can even click around and check things out without actually activating it. Once you are satisfied and would like to use the theme, click on the "Activate" link in the upper right-hand corner of the screen and you're all set.

2.6.4 Key Things to Look For in a Theme

When choosing that perfect theme for your site, you want to focus on how it looks and how it works. Does it rock your browser's very existence? Does it deliver your content on a silver platter for your visitors? Does it make you want to scream in ecstasy and dance the jig? If so, then you know it's the right theme for you. There is no reason to settle for anything less than absolutely perfect, especially given the vast menu of awesome themes available to you. If you find something *close* to perfect that could use a little tweaking, remember that it is much easier to change things like color and font size than the underlying structure and functionality.



That said, here are some key things to look for when searching for the ideal theme:

Navigation

Take a look at what the navigation is like on the theme. Then think about how you picture the navigation working best on your site. Do categories need to be prominently displayed? Are pages more important? Do you need a dropdown menu system? Is there room for you to build your own navigation if needed? Does it support the WordPress 3.0 menu system?

Theme options

Some themes come equipped with *theme options*, literally an extra area of settings in the Admin area for customizing the theme. These options can range from simple, like altering colors, to complex, like integrating social media into your theme. Sometimes these theme options can be very compelling, so look around to see if anything catches your eye.

Widget ready?

When developing a theme, a designer may establish certain areas as “widget-ready.” A widget-ready section in your theme enables you to quickly and easily customize its appearance and functionality. A commonly seen widget-ready area is the theme’s sidebar. In a widget-enabled theme, there is a special place in the Admin area where you can configure widgets without messing with any code. For example, you can drop in a mini-calendar, a chat feature, or some administrative links. Within the comfort of your Admin area, you can specify options and even drag the widgets around to adjust the order in which they appear on your web pages. If you can picture yourself benefitting from widgets, you should ensure that your theme is widget-ready.

Extra functionality

When it comes to functionality, the sky is the limit when it comes to WordPress themes. Some themes really go nuts with functionality that extends far beyond WordPress. For example, a theme may be built to integrate a photo-sharing service

such as Flickr, a statistical application such as Google Analytics, or even a database interface such as phpMyAdmin.

Frameworks

WordPress theme “frameworks” are ever-growing in popularity. These frameworks can add a little extra to the learning curve of WordPress, but once you are comfortable with one, they can greatly facilitate the theme-building process by providing all of the core features and options generally used within the theme. This isn’t the time or place to go into detail, but you may want to look into some of the more popular frameworks:

- **Thematic** (free) <http://digwp.com/u/392>
- **Hybrid** (free, with optional paid support) <http://digwp.com/u/393>
- **Thesis** (\$87) <http://digwp.com/u/394>

Comes with source files?

It is likely that, even if you find a theme that you really like, you’ll want to be doing some customization. If that involves modifying the theme’s images, it’s really nice if the theme includes the source files from which it was created. These could be Photoshop/Fireworks files, vector resources, icons, full-resolution images, etc.

Of course we hope that you develop the chops to build your own themes, but in a pinch, services like WPCoder are great for turning designs into real themes.

<http://wpcoder.com>



Linkage

Flickr:

<http://digwp.com/u/389>

Google Analytics:

<http://digwp.com/u/390>

phpMyAdmin:

<http://digwp.com/u/391>

GPL

The fine folks at Automattic have stated under no uncertain terms that WordPress themes should be licensed under the GPL, the same licence as WordPress is under.

<http://digwp.com/u/562>

This idea of “art directing” articles online has been popularized by Jason. Check out his blog for some jaw-dropping examples of beautiful art direction in blog posts.

<http://digwp.com/u/242>

His blog isn’t powered by WordPress, but interestingly enough, Jason designed both the WordPress Admin area and the WordPress.org website.

You can do your own art direction of individual posts by being able to add custom CSS to specific posts. Check out:

<http://digwp.com/u/464>

To the right you can see two plugins in the list, one active, one inactive.

2.7.1 Getting Started with Plugins

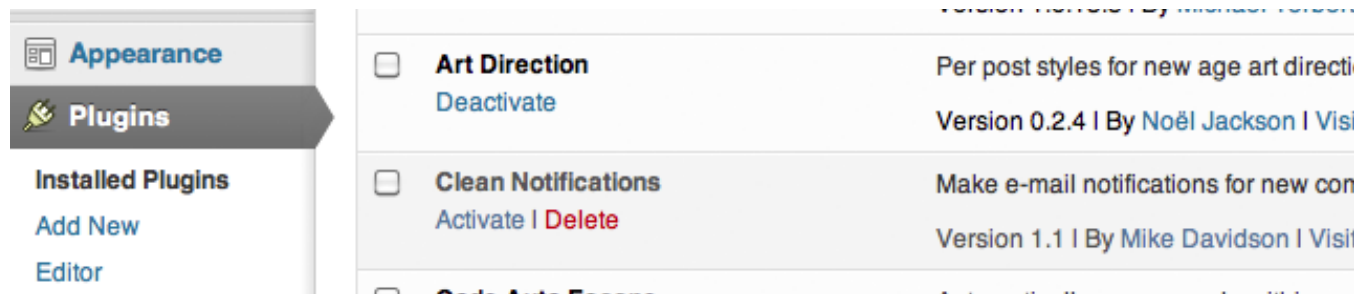
Part of the genius and magic of WordPress is the plugin system. These plugins extend and enhance what WordPress is able to do in very specific ways. Anyone is able to write a plugin for WordPress, and even include it in the official plugin library, otherwise known as the WordPress Plugin Repository <http://digwp.com/u/396>. Let’s explore some essential information for getting started with plugins.

2.7.2 Installing and Activating Plugins

Among the WordPress files on your server, there is a special directory, `/wp-content/plugins/`, that contains all of your site’s plugins. The tried-and-true method for installing a plugin is to download it to your computer, and then upload it to your server into the `plugins` directory. After that, the plugin will appear in the Admin area on the Plugins page. By default, new plugins are inactive, so you will need to manually activate them before they take any effect.

You can, at any time, deactivate a plugin in this same way. Do be aware, however, that plugins have serious power. Any time you activate or deactivate a plugin, you should do some thorough investigating of your site to make sure everything is looking and functioning as expected.

Plugins can also be searched for and installed directly from the Admin area of your site. Just go to **Plugins > Add New**. The plugins available here are exactly the same as those available at the WordPress Plugin Repository. In order to take advantage



of this direct web installation, your plugins directory must be “writeable” by the server. In a perfect world, giving write permissions to a directory would be absolutely safe, but in the hostile environment of today’s Web, you should definitely consider carefully whether or not such permission is truly necessary.

The bonus of downloading plugins from the WordPress.org directory is that you can be sure that the plugin isn’t malicious in any way. There are certainly ways you can get yourself into trouble with plugins, but plugins obtained from the Repository are unlikely to damage your site or harass your visitors. There are plugins “out in the wild” available for download as well, but there are no guarantees as to what you will get, so be very conscious of the source when installing such plugins.

2.7.3 Difference Between Disabling and Uninstalling

Disabling a once-active plugin prevents it from functioning, but does not physically remove the plugin from your `plugins` directory. You could have a thousand disabled plugins doing nothing except for taking up space in your plugin folder. By actually uninstalling a plugin, you remove all files associated with it, and if possible also reverse any changes that the plugin might have made to the database.

In the process of installation and operation, many plugins will automatically insert content into your WordPress database. Such plugins may add new tables or fields, modify existing data, and store information required for usage. Once made, these types of changes will persist even after the actual plugin files are deleted from your server.

Well-built plugins will provide a complete uninstall option that does the work of cleaning up its database changes for you. Plugins that do not provide such convenience must be cleaned up manually. If this is the case for a plugin that you would like to completely uninstall, make sure that you really know what you are doing before making any changes to your database. And don’t forget to make a backup just in case something goes awry.

File Permissions

Refer to chapter 9.1.3 to learn more about setting secure file permissions for WordPress.

Function Exists?

When you deactivate a plugin, you run the risk of a PHP function being present in your theme that doesn't exist. Essentially a disaster that will surely wreck your theme. Before calling plugin-specific functions in your theme, use a conditional to ensure it exists:

```
<?php
if (function_exists('get_poll')) { get_poll(); }
?>
```


2.7.4 Recommended Plugins

The nature of plugins is that they provide WordPress with *supplemental functionality* that may not be needed by every site. Rather than try to squeeze a million features into the WordPress core, application-specific functionality is left to the awesome developers within the thriving WordPress community. Developers see a need (or an opportunity), create a plugin, and release it to users. If the plugin is popular enough, and makes sense to integrate into the WordPress core, the wizards behind the curtain will see that it happens.

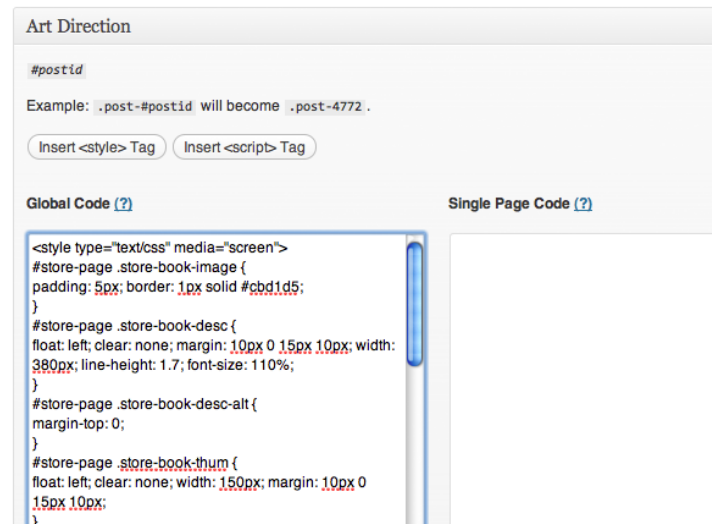
Even so, there remain a number of top-notch plugins that, for whatever reason, have yet to be swallowed up by the core. Here are some of the best that we find useful for virtually *any* type of WordPress-powered site.

VaultPress <http://vaultpress.com>

VaultPress is a plugin and a paid service from Automattic, the creators of WordPress. Once set up, your entire blog is backed up to “the cloud” including all files on the server (WordPress core, themes, plugins, images, etc.) and the database. They offer a premium-level service that includes scanning your files for possible security issues.

Art Direction <http://digwp.com/u/24>

This plugin allows you to insert extra code (typically CSS or JavaScript, but could be anything) into specific Posts/Pages. The custom code can be inserted anywhere the Post appears, or only when viewing that Post alone (single view) Who says every one of your Posts has to have the same styling? Nobody, that’s who. Does every article in a magazine look exactly the same? No, not only because that would be boring but



because each article is unique and should be designed as such. Having complete stylistic and functional control over every Post and Page of your site is very powerful and opens up some awesome design possibilities.

FeedBurner FeedSmith <http://digwp.com/u/26>

The point of using FeedBurner is to get some statistics on how many people subscribe to your site. But what point are statistics unless they are accurate? This plugin will redirect anyone trying to access your WordPress feed directly to your FeedBurner feed address. Set-it-and-forget-it.

W3 Total Cache <http://digwp.com/u/424>

Boosts the performance of your site (i.e., how fast your page loads) by combining a variety of techniques: file caching, database query caching, minifying/compressing/combining files, CDN integration, and more.

WP-DBManager <http://digwp.com/u/111>

There is nothing more important and vital to your WordPress-powered site than the mysterious database that lives on your server. If your entire server was destroyed, but you had a recent backup of your database, you would be OK.

WP-DBManager makes sure you're covered with robust database management from within the WordPress Admin area. WP-DBManager makes it easy to automatically backup, optimize, repair, and even send backups via email.

Posts Per Page <http://digwp.com/u/112>

There is only one setting in WordPress to display how many Posts to show on a page (located under **Settings > Reading**). But what if it made sense to display only one post at a time on your blog's homepage? That would mean that your search page would also display only one post, which is dumb.

FeedBurner

Refer to chapter 6.4.1 to learn more about setting up and using FeedBurner to deliver your feeds.

CDN?

“CDN” stands for Content Delivery Network. These are services which serve up files faster than a typical web server can, and typically used in conjunction with a main webserver to speed up how fast a page loads. An example would be Amazon S3:

<http://digwp.com/u/425>

Custom Post Limits

Another good plugin for customizing the number of posts per page is Custom Post Limits:

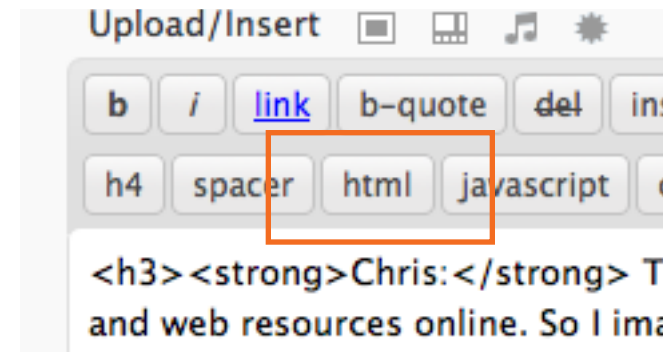
<http://digwp.com/u/594>

The **Posts Per Page** plugin allows you more *fine-grained control* over how many Posts are displayed for each type of page, including search pages, category pages, archive pages, and everything else.

Post Editor Buttons <http://digwp.com/u/113>

There is a user-setting for turning off the visual editor. When you do that, instead of the rich-text editor you see when creating posts, you just get a few buttons and see the raw HTML in the content box. The full control over formatting that this editing mode provides is nice, but the buttons you get are fairly limited.

The good news is that the **Post Editor Buttons** plugin allows you to create your own buttons on the fly, which potentially could be useful for any type of site. Below, we see a number of custom buttons added: “h3”, “h4”, as well as buttons such as “html”, which wraps the selected text in their respective tags.



WordPress SEO by Yoast <http://digwp.com/u/599>

The only SEO plugin you need to further optimize WordPress for the search engines. One stop does it all: on-page analysis, feed optimization, breadcrumb navigation, and easy import from other SEO plugins like **All in One SEO** and **Headspace2 SEO**. It's MultiSite-compatible, has tons of extra features, and even takes care of your XML sitemaps! Visit the plugin page at the Codex for full details.

Google XML Sitemaps <http://digwp.com/u/23>

If you're not using the WordPress SEO plugin to create your XML sitemaps, this plugin will do the job beautifully. Simply install, configure, and presto! — instant XML-sitemap functionality for your entire site.

With the **Google XML Sitemaps** plugin, your sitemap will be Google-compliant, and updated every time you edit or create a post. Then, whenever your sitemap is updated, major search engines like Google, Bing, and Yahoo are instantly notified. Best of all, everything happens quietly and automatically behind the scenes, so you can focus on creating fresh content for your visitors.

Clean Notifications <http://digwp.com/u/25>

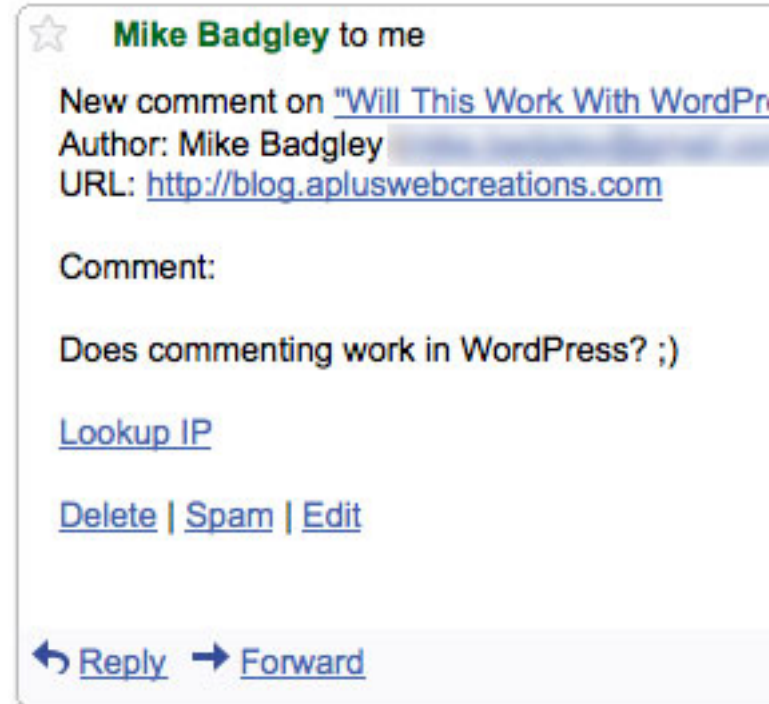
The default comment notification email from WordPress is kind of fugly. It's plain text, and contains a whole bunch of links. Thankfully, the **Clean Notifications** plugin utilizes some very basic HTML to help the emails look much more readable and user-friendly (see screenshot at right).

Subscribe To Comments <http://digwp.com/u/28>

This plugin makes it easy to stay current with the conversation by clicking a checkbox next to the comment form. After checking the box and submitting a comment, you will receive an email whenever a new reply is posted on the thread. Best of all, you can unsubscribe at any time by clicking a link in any one of the email notifications.

What's next?

Now that we have WordPress installed, configured, set up, plugged in and ready to go, it's time to dig into the heart of your WordPress-powered site: the theme.



Life is conversational. Web design should be the same way. On the web, you're talking to someone you've probably never met - so it's important to be clear and precise. Thus, well structured navigation and content organization goes hand in hand with having a good conversation.

– **CHIKEZIE EJIASI**

3

Anatomy of a WordPress Theme

3.1.1 Understanding Theme Files

It is time for us to start taking a close look at how themes are built and how they work. If you have ever themed any kind of application before, you will appreciate how straightforward and intuitive WordPress theming actually is (with plenty of power when you need it). If you have never themed any application before, never fear, it's easy.

3.1.2 Every Theme is Different

Of course, the look of all themes is different. But if you were to download five different WordPress themes and open the folders side by side, you'll see a slightly different sets of files as well. There are a couple of required files and a number of files you will likely find in all themes, but beyond that the door is pretty wide open. For example, some themes might come with a special archives page because that theme is built to showcase archives in a unique way. Another theme might be missing `search.php`, because its `index.php` is built to handle search the functionality.

3.1.3 Commonly Used Theme Files

In the chart on the next page, notice how we have labeled each of the theme files. Two of them, `index.php` and `style.css` are labeled as "CORE". This means that they are absolutely essential to your theme. In fact, WordPress will not recognize any theme if these two files are not within the theme folder. Technically, you

	STANDARD (used in most themes)		SPECIAL (optional additions)		CORE (required)		JUNK (legacy, don't use)

404.php	Error page, served up when someone goes to a URL on your site that doesn't exist
archive.php	Page that displays posts in one particular day, month, year, category, tag, or author
archives.php	Page template that includes search form, category list, and monthly archives (requires page using it)
comments-popup.php	If you enable popup comments (obscure function), the comments link will use this template
comments.php	This file delivers all the comments, pingbacks, trackbacks, and the comment form when called
footer.php	Included at the bottom of every page. Closes off all sections. (Copyright, analytics, etc)
front-page.php	Displays content for the site's front page, aka home page
functions.php	File to include special behavior for your theme.
header.php	Included at the top of every page. (DOCTYPE, head section, navigation, etc)
home.php	Displays content for the site's front page if front-page.php is not available
image.php	If you wish to have unique pages for each of the images on your site (for credits, copyright...)
index.php	This is typically the "homepage" of your blog, but also the default should any other views be missing
links.php	Special page template for a home for your blogroll
loop.php	Common in newer themes, an optional file to house your custom, multiple, or regular loops
page.php	Template for Pages, the WordPress version of static-style/non-blog content
rtl.css	A special CSS file for your optional inclusion to accommodate "right to left" languages
screenshot.png	This is the image thumbnail of your theme, for help distinguishing it in the Appearance picker
search.php	The search results page template
sidebar.php	Included on pages where/when/if you want a sidebar
single.php	This file is displays a single Post in full (the Posts permalink), typically with comments
style.css	The styling information for your theme, required for your theme to work, even if you don't use it

could build a theme with only these two files. And a simple theme it would be! That might be just what you need for some special applications of WordPress, but in general, you are probably using WordPress because you want a bit more functionality than that would offer.

Most themes will include both the CORE and "STANDARD" files. The STANDARD files cover everything both you and your visitors will expect from a blog. Things like permalinked posts and pages, error catching, commenting, and organized archives.

Some of these files are marked as "SPECIAL", in that they offer something above and beyond the basics. For example, the `image.php` file. If you choose to use the WordPress default media library to manage the files you post to your site (images, movies, etc.), you can insert them into your posts with a link to a special page on your site controlled by the `image.php` file. This can be useful. You can include special information on this page like copyright information, author information, usage rights, etc. Not all sites would want or need this, hence its designation as SPECIAL.

One of the files listed, `comments-popup.php`, is marked as "JUNK" because it's no longer used but still found in many themes. The file is just weird; we could tell you all about it, but it's not worth the ink (really).

More to the list

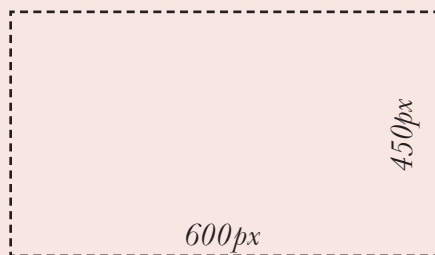
The chart on the opposite page isn't a full list of all template files, just some common ones. See page 57 for more. You are also free to create as many of your own custom theme files in here as you like, that can act as page templates. Check out this page at the WP Codex for a closer look at theme templates:

<http://digwp.com/u/634>

Brand Your Theme

Adding a screenshot and description to your theme is a great way to give it an distinct identity. Here's how:

1 Create a file named `screenshot.png` and put it in your theme folder.



2 Put this description at the top of your `style.css` file

```
/*
Theme Name: Theme Name
Theme URI: http://your-website.com/cool-theme/
Description: Totally awesome WordPress theme by
<a href="http://your-website.com/">Yours Truly</a>
Version: 1 (WP2.8.4)
Author: Your Name
Author URI: http://your-website.com/
Tags: super, awesome, cool, sweet, potato nuggets
*/
```


3.1.4 How Theme Files Work Together

These files are not stand-alone templates. They interact and call upon each other to get the job done. For example, `index.php` alone will call and insert `header.php` at the top of it, `sidebar.php` in the middle of it, and `footer.php` at the bottom of it. Then, the `sidebar.php` file might have a function to call in `searchform.php`. Likewise, the `header.php` file, which includes the `<head>` section, will call upon the `style.css` file.

It is this modular, dynamic approach that gives WordPress theme building a lot of its power. For those folks coming from a background of building static sites, the nature of using templates is probably already quite appealing. Imagine wanting to add a navigational item to the site's main menu bar, which likely lives in the `header.php` file. One change, and the new navigational item is reflected on all pages of the site. Going further, the menu bar itself is likely generated from a built-in WordPress function. As soon as you publish a new page from the Admin area of WordPress, the menu-bar function will recognize the new page and automatically append it to the sitewide menu bar. This is powerful stuff that makes site modifications, updates, and management very easy.

3.2.1 Understanding Different Page Views

There are really only a handful of different types of page views:

- **The Home Page** - usually at the root URL of your domain
- **Single Posts** - displays one post at a time, usually in its entirety
- **Static Pages** - pages that are outside the flow of normal posts
- **Custom Pages** - static pages that have been customized
- **Search Results** - displays a list or summary of posts matching a search
- **Archive** - shows series of posts for categories, tags, dates, and authors

3.2.2 Page Views are for Pages

We already learned about Pages and how they are most commonly used for “static” style content. You cannot categorize or tag a Page, they exist outside the chronological flow of posts, and they don’t appear in the RSS feed like Posts do. As such, the theme template used to display Pages is generally different than that used to display Posts. For example, it may lack the functionality to display things such as dates, author names, and comments. Instead, it might include functionality to display the breadcrumb trail of its hierarchy of parent pages (see chapter 5.5.6).

3.2.3 Single Views are for Posts

The `single.php` file is responsible for displaying a single Post. There may be parts of the `single.php` template file for displaying categorization and other “meta” information about the post, as well as the functionality required for displaying the comments area and comment form. Perhaps you want your single posts to be a bit wider and less cluttered? The `single.php` file is where you might omit calling the sidebar and adjust your CSS accordingly.



PAGE

Regular Title

No comments
This content isn't really meant for public discussion.

Unique sidebars
The sidebar needs on this page are different than elsewhere on the site. WordPress can accommodate.

Nav Highlighting
About page = About highlighted in navigation

POST

Extra Blog Header
Blog posts have “blog” header in addition to title and meta about this post.

Comments
This content is meant for public discussion. (not visible in screenshot, but there!)

Unique sidebars
Blog area has blog-related ancillary content, like categories, subscription info, and popular content.

Nav Highlighting
Any blog page = Blog highlighted in navigation



3.2.4 The Many Faces of Archive Views

There are many types of archives, and this one file, `archive.php`, is often in charge of displaying them all. When viewing a particular category, tag, author, or date-based archive, WordPress will generate the markup and display the content according to the code contained in the `archive.php` file.

Look at all the archive links at the Digging Into WordPress site. Every one of those subsequent pages is handled by the `archive.php` file



3.2.5 How WordPress Decides Which File to Use for Rendering the View

All this talk about different page views is begging the question, “how does WordPress figure out which template file to use?” You might assume that it is hard-wired into WordPress, but as we’ve learned, most of the files in a theme are optional. If your theme doesn’t have an `archive.php` file, does WordPress just display a blank page? Absolutely not, it moves down its hierarchy of template files to find the next most appropriate file to use. Ultimately, all paths in the WordPress templating world end at the `index.php` file. No wonder this is such an important and required file!

Just as we move down the hierarchy toward `index.php`, we can travel in the other direction and create template files that are very specific. For example, if we wish to have a unique template when viewing category #456 of our blog, we can create a file called `category-456.php`, and WordPress will automatically use it. Let’s take a look at the hierarchy flowchart.

WHICH TEMPLATE FILE WILL WORDPRESS USE?

T H E T E M P L A T E H I E R A R C H Y

PAGE TYPE tries first > tries next > tries last

404 404.php > index.php

SEARCH search.php > index.php

TAXONOMY taxonomy-{tax}-{term}.php > taxonomy-{tax}.php > taxonomy.php > archive.php > index.php

HOME home.php > index.php

ATTACHMENT {mime-type}.php > attachment.php > single.php > index.php

SINGLE single-{post-type}.php > single.php > index.php

PAGE {custom-template}.php > page-{slug}.php > page-{id}.php > page.php > index.php

CATEGORY category-{slug}.php > category-{id}.php > category.php > archive.php > index.php

TAG tag-{slug}.php > tag-{id}.php > tag.php > archive.php > index.php

AUTHOR author-{author-nicename}.php > author-{author-id}.php > author.php > archive.php > index.php

DATE date.php > archive.php > index.php

ARCHIVE archive.php > index.php

3.3.1 Kicking It Off with the Header

If you had never seen the files in a WordPress theme before, you could probably guess which file is responsible for the top of pages. It's everybody's favorite theme file: `header.php`!

3.3.2 The DOCTYPE and HTML Attributes

In 99.999% of all themes, the header file is the *first* file that is called when WordPress is rendering any type of web page. As such, its contents begin with the same code that *all* web pages begin with, namely, the DOCTYPE. This isn't the time or place to talk about why to choose one DOCTYPE over another (there are plenty available to choose from), but suffice it to say that XHTML 1.0 Strict is a very common DOCTYPE choice these days. Here's how it looks in the source code:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Directly after any site's DOCTYPE element is the opening HTML tag, which has a number of attributes that work with the DOCTYPE to prepare the browser for what to expect from the source code. Two commonly seen attributes for the `<html>` tag include language attributes and XML namespace declarations. At this point, WordPress jumps in to help define the page's language attributes:

```
<html xmlns="http://www.w3.org/1999/xhtml" <?php language_attributes(); ?>>
```

At the time of the writing of this book, HTML5 is really starting to get popular. The DOCTYPE for this upcoming version of HTML is deliciously simple:

```
<!DOCTYPE html>
```

It just doesn't get much better than that. Needless to say, we're looking forward to the day when HTML5 is completely implemented.

3.3.3 META Elements

After the opening `<html>` tag, we move into the `<head>`, which is also common to all web pages and provides all sorts of information the browser needs to display the page as intended. Within the `<head>` section, we begin with some choice `<meta>` tags, which can be thought of as “information about information.” In this case, the HTML is the information, and so meta tags describe that information. To let the browser know the content type and language used, WordPress helps us with some super-handly template tags:

```
<meta http-equiv="Content-Type" content="<?php bloginfo('html_type'); ?>;  
      charset=<?php bloginfo('charset'); ?>" />  
<meta charset="<?php bloginfo('charset'); ?>">
```

Simplified HTML5

The bottom example is the simplified HTML5 version of declaring a character set.

Other important meta tags include “description” (very important) and “keywords” (less important). But because the description and keywords for any given page on your site depend on the actual content of that page, it is best to dynamically generate these tags rather than include them directly here. See page 48 for information about Yoast’s WP SEO plugin to handle meta tags automatically.

3.3.4 The Title

The `<head>` is also where the `<title>` for the page is declared, which is an incredibly important line in any HTML code. It is literally what is shown at the top of the browser window, what is saved as the default title of bookmarks (both saving locally and socially), and is used for the title link in search-engine listings. Again, we are in the tough position where this bit of code is written only once, right here, and is used for every single page on the entire site. So how do you craft it so that the title is optimal on every possible page? Glad you asked.

Here is an excellent function that enables top-notch, attractive-looking and descriptive titles for every possible type of web page. Simply use this code as the `<title>` element in your theme’s header .php file and you’re good to go:

Perfect Title Tags

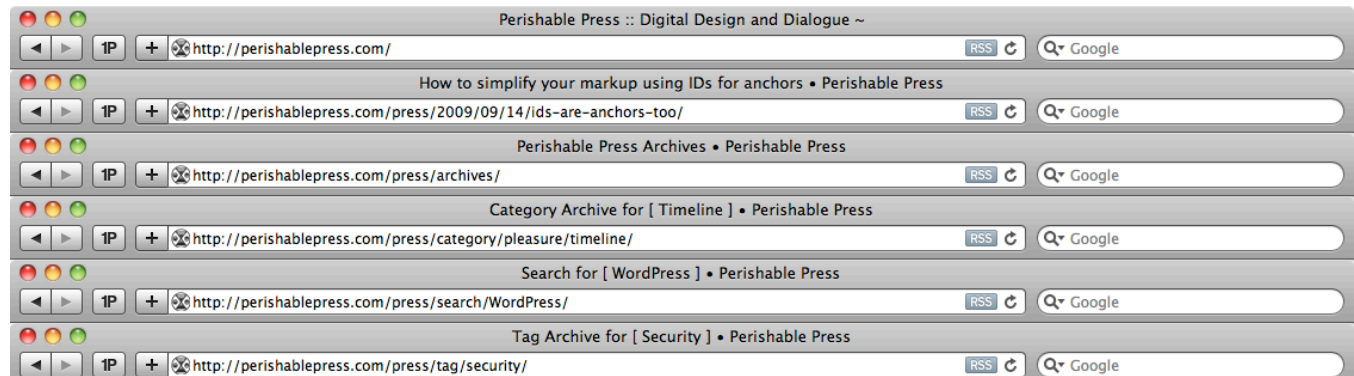
For the full scoop on creating perfect title tags for your WordPress-powered site, check out these two articles:

<http://digwp.com/u/397>

<http://digwp.com/u/398>

```
<title>
    <?php if (function_exists('is_tag') && is_tag()) {
        single_tag_title('Tag Archive for &quot;'); echo '&quot; - ';
    } elseif (is_archive()) {
        wp_title(''); echo ' Archive - ';
    } elseif (is_search()) {
        echo 'Search for &quot;'.wp_specialchars($s).'&quot; - ';
    } elseif (!(is_404()) && (is_single()) || (is_page())) {
        wp_title(''); echo ' - ';
    } elseif (is_404()) {
        echo 'Not Found - ';
    }
    if (is_home()) {
        bloginfo('name'); echo ' - '; bloginfo('description');
    } else {
        bloginfo('name');
    }
    if ($paged > 1) {
        echo ' - page '. $paged;
    } ?>
</title>
```

Those sure would bookmark nicely, wouldn't they?



The All-In-One SEO Plugin that we mentioned earlier can also be put in charge of handling page titles. The advantage is that it keeps this area of the theme cleaner and does provide what is generally considered the best page title format for SEO. The disadvantage being that it isn't very customizable or nearly as configurable as doing it yourself.

3.3.5 Link Elements

The `<head>` is also the place to link to external resources like CSS and JavaScript files. Since your theme requires the presence of a `style.css` file in the root directory of your theme, you might as well use it. Including it is as simple as this:

```
<link rel='stylesheet' href='<?php bloginfo("stylesheet_url"); ?>'
      type='text/css' media='screen' />
```

The parameterized function, `bloginfo("stylesheet_url")`, literally returns the exact URL of the stylesheet. No reason to hard-code anything here. And in fact, the `bloginfo()` function can return all sorts of useful information, which we'll dig into shortly.

Parameterized is a fun word, isn't it?

On the other hand, including JavaScript files in your theme is slightly trickier, especially if you want to do it the *right* way (you do). Let's say you want to include the popular JavaScript library jQuery on your page, and also a custom script of your own that makes use of jQuery. Because jQuery is such a popular library, it is used fairly commonly by other plugins, and in fact by the WordPress Admin area itself. As such, WordPress literally ships with a copy of jQuery you can link to. To do so, simply call this function in your head area or `functions.php` file:

```
<?php wp_enqueue_script('jquery'); ?>
```

Doing it this way has a few distinct advantages.

- 1. It's easy.** It creates a link to a file you know is there and you know works.
- 2. It lets WordPress know** that the requested file is successfully loaded.

If you go off and download your own copy of jQuery and link to that, WordPress has no idea that you've done this. Then if you start using a plugin that utilizes jQuery, it will go off and load another copy, which will cause all sorts of havoc. Conversely, if you enqueue the file instead, the plugin will recognize the fact it already exists and not load a duplicate copy. *Hurrah!*

On the other hand, when you load your *own* script, you don't really need to enqueue it because it is already totally unique and not included in WordPress. You can load your own script on the page like this:

```
<script type="text/javascript"
      src="<?php bloginfo('template_url'); ?>/js/myscript.js"></script>
```

As you can see, we are using another `bloginfo` function here, only this time it outputs the URL path to the active theme directory, not to any particular file.

Now, let's say on your archives pages that you have a whole bunch of special CSS that isn't used anywhere else on the site and a custom script that is unique to your archives pages. You can use some special WordPress logic to detect if the archives pages are the ones being viewed, and load the files only in that situation:

```
<?php if (is_page_template('page-archives.php')) { ?>
    <link rel="stylesheet" href="<?php bloginfo('template_url'); ?>/css/
archives.css" type="text/css" media="screen" />
    <script type="text/javascript" src="<?php bloginfo('template_url'); ?>/
js/archives.js"></script>
<?php } ?>
```

That will take effect if you are using a special page template for your archives that is literally named `"page-archives.php"`. If instead you happen to know the ID of the page (available in the Admin area, see note on next page), that could be written like this:

```
<?php if (is_page("5")) { ?>
    <link rel="stylesheet" href="<?php bloginfo('template_url'); ?>/css/
archives.css" type="text/css" media="screen" />
    <script type="text/javascript" src="<?php bloginfo('template_url'); ?>/
js/archives.js"></script>
<?php } ?>
```

...where "5" in the first line is the page ID. Feel free to use PHP's "or" operators here to cover multiple pages.

Putting all of that together, our code looks something like this:

```
<?php wp_enqueue_script('jquery'); ?>
<?php wp_head(); ?>
<script type="text/javascript" src="<?php bloginfo('template_url'); ?>/js/
myscript.js"></script>
<?php if (is_page("5")) { ?>
    <link rel="stylesheet" href="<?php bloginfo('template_url'); ?>/css/
archives.css" type="text/css" media="screen" />
    <script type="text/javascript" src="<?php bloginfo('template_url'); ?>/
js/archives.js"></script>
<?php } ?>
```

Hey! What's up with that wp_head() thing? Glad you asked...

What is My Page ID?

Determining the ID of your posts and pages is not as easy as it used to be. In previous versions of WordPress, the ID was conveniently displayed right next to the post or page in the Admin area.

In newer versions of WordPress, ID information has been removed, and is only accessible by hovering over the post/page link in the Admin's Edit Posts or Edit Pages screens.

Thus, to get your ID, hover over its link in the Admin and look at your browser's Status Bar to see the information. It will be appended to the URL as the last parameter value.

3.3.6 The wp_head() Function

A must for any theme, the `wp_head()` function simply tells WordPress “Right here, this is inside the `<head>`.” It is kind of a generic function that is used as a “hook” on which the WordPress core, plugins, and custom functions may attach things.

For example, if you have the XML-RPC functionality of your blog enabled (Settings > Writing), it requires a special `<link>` element to be added into the `<head>`. If it is present within your theme, the `wp_head` function will be used by WordPress to include the required XML-RPC element to the `<head>`.

Similarly, in the previous section, the code uses the `wp_enqueue_script` function. All by itself, that function doesn’t have any effect. But when the `wp_head` tag is also present, it serves as a hook that serves as the location at which the `wp_enqueue_script` function will load the script.

Plugins also use the `wp_head` function to load their own scripts and CSS files. Sometimes they even insert inline CSS and JavaScript, which is a bit annoying and makes for a messy “View Source” experience.

3.3.7 Template Tags

Now is a good time to mention that there is a WordPress function for pulling out a variety of information about your blog. This information is useful on a regular basis when creating themes. Here is the function...

```
<?php bloginfo('template_url'); ?>
```

...and here is the different types of data that you can get from it:

```
admin_email = jeff@digwp.com  
atom_url = http://digwp.com/home/feed/atom  
charset = UTF-8
```

```
comments_atom_url = http://digwp.com/home/comments/feed/atom
comments_rss2_url = http://digwp.com/home/comments/feed
description = Take Your WordPress Skills to the Next Level!
url = http://digwp.com/home
html_type = text/html
language = en-US
name = Digging Into WordPress
pingback_url = http://example/home/wp/xmlrpc.php
rdf_url = http://digwp.com/home/feed/rdf
rss2_url = http://digwp.com/home/feed
rss_url = http://digwp.com/home/feed/rss
siteurl = http://digwp.com/home
stylesheet_directory = http://digwp.com/home/wp/wp-content/themes/largo
stylesheet_url = http://digwp.com/home/wp/wp-content/themes/largo/style.css
template_directory = http://digwp.com/home/wp/wp-content/themes/largo
template_url = http://digwp.com/home/wp/wp-content/themes/largo
text_direction = ltr
version = 2.8.5
wpurl = http://digwp.com/home/wp
```

If you were looking closely, you may have noticed we have already used this function earlier in our example showing how to include a stylesheet:

```
<link rel="stylesheet" href="<?php bloginfo('template_url'); ?>/css/archives.css" type="text/css" media="screen" />
```

This is how you can generate a URL from inside your theme folder without having to hard-code anything or worry about relative file paths. Hard-coding is problematic (what if you change the name of the theme?). Relative file paths are problematic too, because the URL structure of a site can change and go many levels deep, the only reliable way to do it is to start with the root ("/"), which would then require the theme's folder name anyway.

Global Custom Fields

Another way to look at the `bloginfo()` function (see 3.3.7) is as a “Global Custom Field.” That is, a value that you can access from anywhere that returns a value you can use. Posts and Pages can have custom fields as well, but they are localized to that Post or Page and thus not very “Global.” Creating your own global custom fields could potentially be very useful. For example, let’s say you use the Amazon Affiliate Program to help your site earn money. This affiliate code is baked into all sorts of data that you can get from Amazon, like URLs for linking to products and their widgets. As with everything, you could hard-code this affiliate code everywhere it needs to be, but that isn’t a very efficient technique. If this code were to change some day (you never know), you are stuck updating a lot of code. Instead, let’s do it right by literally creating a custom settings area in the Admin for creating our own global custom fields.

Add this to your `functions.php` file:

```
<?php add_action('admin_menu', 'add_gcf_interface');

function add_gcf_interface() {
    add_options_page('Global Custom Fields', 'Global Custom Fields', '8', 'functions',
        'editglobalcustomfields');
}

function editglobalcustomfields() { ?>
    <div class="wrap">
        <h2>Global Custom Fields</h2>
        <form method="post" action="options.php">
            <?php wp_nonce_field('update-options') ?>
            <p><strong>Amazon ID:</strong><br />
                <input type="text" name="amazonid" size="45"
                    value="<?php echo get_option('amazonid'); ?>" />
            </p>
            <p><input type="submit" name="Submit" value="Update Options" /></p>
            <input type="hidden" name="action" value="update" />
            <input type="hidden" name="page_options" value="amazonid" />
        </form>
    </div>
    <?php } ?>
```

You can now display this value anywhere in your theme with the `get_option()` template tag:

```
<?php echo get_option('amazonid'); ?>
```

GCF @ DigWP.com

*A bit more about Global
Custom Fields at DigWP.com:*

<http://digwp.com/u/586>

<http://digwp.com/u/587>

3.4.1 The WordPress Loop

The loop is the *one thing* that is *absolutely core* to understanding how WordPress works. In its most basic, generalized form, the loop looks like this:

```
<?php
// The Loop
if (have_posts()) : while (have_posts()) : the_post();
...
endwhile; else:
...
endif;
?>
```

As veteran developers know, a “while” loop is a standard concept in any programming language, and its use here is just standard PHP. First the loop makes sure that there are some posts to display (the “if” statement). If that is true, it begins the loop. Then, the function “the_post()” sets the stage for WordPress to use inner-loop functions, which we will explore soon. Once the_post() has been iterated the specified number of times, “have_posts()” turns to false and the loop stops.

Yikes! That is sounding pretty abstract. Perhaps we better break things down so we don’t lose each other.



*Bad analogy!
Bad analogy!*

3.4.2 The Loop in Plain English

Are there any posts published? Sorry, just had to ask, the rest of this code will go funky if there aren't any.

Begin the loop. This will cycle through the number of Posts you have set to display (under **Settings > Reading**).

A header tag with an anchor link inside it. The text will be the title of the Post, and the link will be the permalink to the single Post page.

A custom field that is attached to this Post is pulled out and displayed. In this case, the key of "PostThumb", which returns an "" tag symbolizing this Post.

"Meta" information about the Post is displayed: the Month Day, Year the Post was published and the display name of the Author who wrote it.

The full content of the Post is displayed.

More meta information about the post is displayed: all the tags and categories given to this Post and the number of comments, which is a link to the commenting area.

End of the loop

If there are older or newer posts available, display links to them.

No posts? (a failsafe)

Better tell the people.

All done.

```
<?php if (have_posts()) : ?>

<?php while (have_posts()) : the_post(); ?>

    <div class="post" id="post-<?php the_ID(); ?>">

        <h2><a href="<?php the_permalink(); ?>"
            rel="bookmark" title="Permanent Link to <?php the_
            title_attribute(); ?>"><?php the_title(); ?></a></h2>

        <?php echo get_post_meta($post->ID, 'PostThumb',
            true); ?>

        <p class="meta">

            <span>Posted on</span> <?php the_time('F jS,
            Y'); ?> <span>by</span> <?php the_author(); ?>

        </p>

        <?php the_content('Read Full Article'); ?>

        <p><?php the_tags('Tags: ', ' ', ' ', ' <br />'); ?>
        Posted in <?php the_category(' ', ' '); ?>
        <?php comments_popup_link('No Comments;',
            '1 Comment', '% Comments'); ?></p>

    </div>

    <?php endwhile; ?>

    <?php next_posts_link('Older Entries'); ?>

    <?php previous_posts_link('Newer Entries'); ?>

    <?php else : ?>

        <h2>Nothing Found</h2>

    <?php endif; ?>
```

3.4.3 The Loop Just Knows...

As mentioned, the loop is simply a dressed-up “while” loop. While there are posts available in the database, display the posts. In theory, it’s simple and utilitarian. But what might remain confusing is just how this while loops knows exactly what to loop. While... what? Well, without you having to tell it, the basic loop function already knows what its query is going to be! To see for yourself what the query string is, you can echo it to the web page by adding this little snippet directly before the loop:

```
<?php global $query_string; echo $query_string; ?>
```

If we were to place this snippet above our `single.php` loop at the Digging Into WordPress site, the following information would be displayed on any single page:

```
year=2011&monthnum=02&name=version-3-update
```

In plain English, that reads: “The date is February 2011 and the post name is *Version 3 Update*.” Likewise, if we echo that `$query_string` variable from our `archive.php` file, and then visit the “JavaScript” category archive, we see this:

```
posts_per_page=10&what_to_show=posts&orderby=date&order=DESC&category_
name=javascript
```

In plain English: “Show ten Posts from the Javascript category in descending chronological order.”

Note that we did nothing *manually* to change this query string, but merely by loading a different type of page (an archive view), WordPress provides the proper query to make that loop do the right thing. Don’t worry if this sounds confusingly technical. It doesn’t really matter. The point is that The Loop *just knows* what to loop through for the type of page you are building and displaying.

loop.php

Many WP themes include a `loop.php` file, which helps reduce repetitive code in other theme files. Explore!

3.4.4 Some Common “Loop Only” Functions

While (get it?!) you are inside the loop, you have access to a number of functions that aren’t available elsewhere. These are functions that return things specific to individual posts. So it’s not that these functions are limited per se, but they just wouldn’t make any sense otherwise. A great example:

```
<?php the_title(); ?>
```

This function displays the title of the current Post. Remember that we are in a loop, so if that loop runs five times, this function will display five different items, namely, the title for each of our five posts. Here is a number of common and highly useful “inside-loop-only” functions:

- `the_permalink()` - displays the permalink URL for each post
- `the_ID()` - displays the ID of each post
- `the_author()` - displays the name of the author for each post
- `the_category()` - displays the category/categories to which each post belongs

While you are inside the loop, you also have access to a bunch of preset variables that are populated after `the_post()` is executed. These variables exist in the object `$post`. Much of this object data is used by functions that use it in more elaborate ways, but the `$post` object provides “raw” data that is sometimes incredibly useful.

Get more info

Tip: if you know the name of the variable(s) you’re after, you can use any of the following PHP functions to see what’s available:

```
print_r()  
var_dump()  
get_defined_vars()
```

- `$post->ID` - returns the ID of post; useful for other functions that need an ID.
- `$post->post_content` - the actual post content, including all markup; useful when you need to process the content of a post before outputting it.
- `$post->post_modified` - returns the date/time the post was last updated.
- `$post->post_name` - returns the slug of the post.

In addition to these, there are many more. Check out <http://digwp.com/u/635> and <http://digwp.com/u/636> for reference.

3.4.5 Some Common “Outside Loop” Functions

Some functions are built to return more global and/or generic information that doesn’t have anything to do with any one particular Post. As such, they are meant to be used in templates *outside* of the loop.

Here is a number of common and frequently used “outside-loop-only” functions:

- `wp_list_pages()` - displays a list of links to your static pages
- `next_posts_link()` - displays a link to older posts in archive views
- `wp_tag_cloud()` - displays a tag cloud of all your tags
- `get_permalink()` - returns the permalink of a post for use in PHP

Of course this is just a tiny sampling of all the functions available. The point we are trying to drive home here is that some functions are dependent on being inside the loop to function properly, and some do not.

[Get more info](#)

We go into more depth on the WordPress Loop in chapter 4, including custom loops, multiple loops, and more.

3.5.1 Comments

Comments may be one of the reasons you are using WordPress to begin with. It is a powerful system with lots of options and management possibilities. We are going to go much more in-depth into comments in chapter 7, but comments are definitely part of the anatomy of a theme, so let’s get familiar now.

3.5.2 The comments.php File

In general, WordPress themes neatly compartmentalize the commenting functionality into a single file, `comments.php`, which is responsible for the following:

- All the logic necessary for displaying the appropriate data

- Displaying all the current comments
- Displaying the comment form

To accomplish these things, the `comments.php` file requires a complex system of logic. It goes something like this:

Is the post protected? (password required)

- Yes - display message and stop
- No - continue

Are there any comments?

- Yes - display them and continue
- No - don't display anything and continue

Are comments open?

- Yes - continue
- No - display message and stop

Is registration required?

- Yes - display message and stop
- No - display comment form

And that is just the basic functional flow. Additional logic is used within the various functions to accommodate different conditions and different settings. For example, is the comment form being displayed in the default generic state, or is it an existing comment that is being replied to?

```
<?php comment_form_title('Leave a Comment', 'Leave a Reply to %s'); ?>
```

3.5.3 Selective Inclusion for Different Views

The beauty of having all of this magic tucked away into a single file is that you can add the complete commenting functionality to different page views (i.e., different theme template files) with a single function:

```
<?php comments_template(); ?>
```

For example, your `single.php` file will likely have this function below the loop and all the content stuff. Your `page.php` might be incredibly similar to your `single.php` file, but may omit this function because you don't wish to have commenting on your static content. What if the day comes along though where you wish to include comments on Pages? You can simply create a new page template and include the `comments_template()` function. *Easy.*

So, do you need to create a new special page template for every little trivial change like this? Well, no, you don't have to. How about another common example. Say you want *some* of your Pages to display your *regular* sidebar, and hide it from some of your other Pages. Apart from this difference, all of your pages will be exactly the same. You *could* create a special page template called something like `page-nosidebar.php` and omit the `<?php get_sidebar(); ?>` template tag. But that's a lot of repeated code for such a trivial change.

A better approach is to use a custom field to designate if you want to have a sidebar or not. Then in the template, look for that custom field and behave accordingly. Here is how this could look in your `index.php` file:

```
<?php // conditional sidebar display
if (!get_post_meta($post->ID, "noSidebar", true)) {
    get_sidebar();
} ?>
```

You could use this exact same technique for anything that you wish conditionally to include or not include on a page template, without having to create *multiple* templates.

Name	Value
noSidebar	TRUE

Buttons: Cancel, Add Custom Field

Screenshot showing how to set the custom field for this example

The screencasts on CSS-Tricks are Pages, not Posts. They have their own special template which, among other things, includes user comments.



#72: Building a Website (2 of 3): HTML/CSS Conversion

Posted on: 9/23/2009 By: Chris Coyler

In part 2 of this series, we begin the HTML/CSS conversion of the Photoshop mockup we created in part one. We start with a very skeletal project framework. Then we take a look at the Photoshop file layer organization. Then we start from the bottom up, creating the pieces we need from the Photoshop file and writing the HTML and CSS we need to get the job done. Much of the work isn't actually "string" the Photoshop file, but looking closely at it and trying to mimic what is done there with correct markup and CSS techniques.

Running time: 01:11:10

Direct Download: [High Quality, Quicktime .MOV Format \(AppleTV Ready\)](#)

Links from Video:

- [Skins.Yong.Bike](#)
- [Query.Prospect](#)
- [Part 1: Photoshop Mockup](#)
- [Part 2: HTML/CSS Conversion](#)
- [Part 3: WordPress Theming](#)



[click here for full screen](#)

Responses

- Tom Rogers** says: 09/23/2009 at 7:48 am (UTC) [Reply](#)
Have been looking forward to this all week and it didn't disappoint.
- Forrest** says: 09/23/2009 at 8:18 am (UTC) [Reply](#)
Thanks for taking the time to share. Always appreciate it.
- Sumeet Chawla** says: 09/23/2009 at 8:18 am (UTC) [Reply](#)
Ah... I was waiting for this since you released the part one of this series. I'll use to check the site every 22 hours lol...
Wow, you added a comment edit thingy... what plug-in did u use for it? Anyway, gonna sit back, get a glass of pepsi and enjoy this screencast now :)
- Brandon** says: 09/23/2009 at 11:28 am (UTC) [Reply](#)
Chris, I saw your tweet that you were having trouble with (show) last night... glad to see that you got it working and up today! 1 step closer to getting my blog and future client blogs workin' well because of you!
You really don't need to wait till next week... hint hint... I'm ready to watch the 3 part right now..!
- Brandon** says: 09/23/2009 at 11:28 am (UTC) [Reply](#)
Chris, saw your tweet last night that you were having some trouble with (show)... lame... I'm super glad you got it working and up today! Thanks to you I'm now 1 step closer to becoming a wordpress ninja! :>) That said... you really don't have to wait till next week to get the part 3 out - hint hint - I'm ready to watch it right now...
- Nutt** says: 09/23/2009 at 11:37 am (UTC) [Reply](#)
These videos have been a huge help in understanding how css works. Anyways can't wait for #3 and the conversion to wordpress. Keep up the awesome work.
- Carlos G.** says: 09/23/2009 at 12:19 pm (UTC) [Reply](#)
Thanks Chris...Very helpful

3.6.1 The Sidebar

Sometimes One,
Sometimes Two

The Fuel Network blogs have two sidebars on the right of their main content area when viewing their homepage. When viewing an individual post, the “middle” sidebar is gone, leaving a wider main content area.

Sidebars are such ubiquitous design elements that special functionality is built right into WordPress for handling them. In any of your theme files where you wish to display a sidebar, simply place the following template tag where you would like it to appear:

```
<?php get_sidebar(); ?>
```

That function will go get the file “sidebar.php” and place its content where this function was called. Working with *multiple* sidebars?

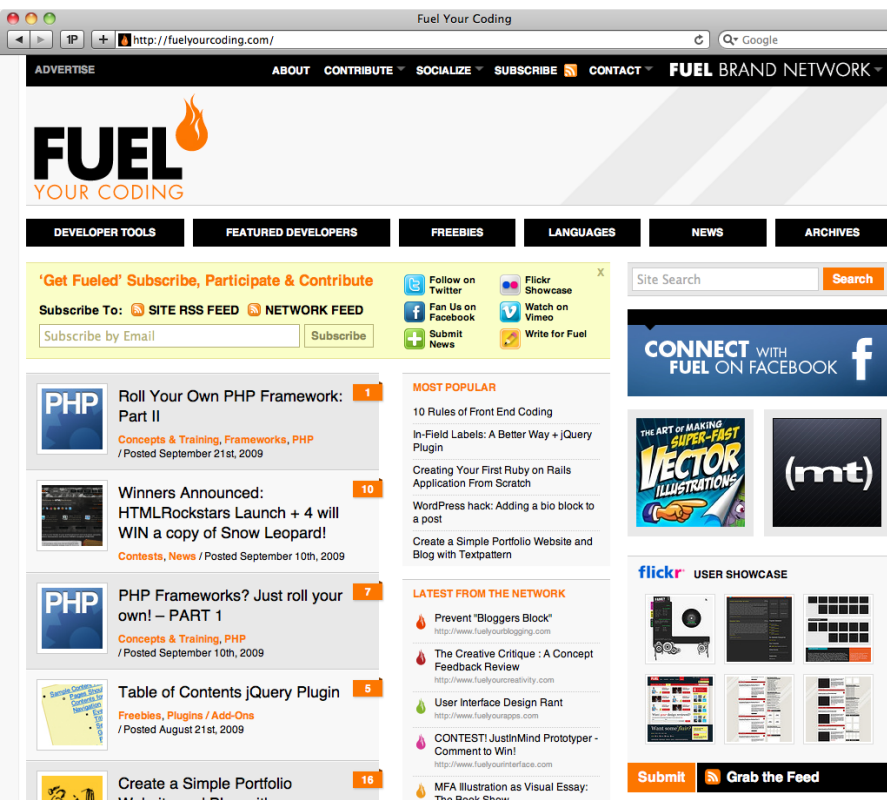
```
<?php get_sidebar('secondary'); ?>
```

That function will retrieve the file called “sidebar-secondary.php” and load that.

Despite all this special functionality, sidebars are by no means required. Don’t want one? Don’t need one? No problem. Just don’t put a sidebar.php file in your theme and don’t call the function.

3.6.2 Purpose and Placement

Sidebars are for “stuff.” Websites are full of stuff. There is the main content, of course. Navigation, sure. But then there is all kinds of other stuff. We don’t want this stuff in the footer because that’s way down there all lonely at the bottom of the page. So we put it on the side instead. You know what I’m talking about. Stuff = alternate navigation, ancillary content, small forms, descriptive text, advertising, blogrolls, pictures of cats... stuff.



Yet despite all of the stuff they contain, sidebars are typically much narrower than the main content area. Sidebars are generally placed to the left or right of the main content (e.g., your posts), but may actually appear anywhere. It all depends on the structure and styling of your theme.

3.6.3 Popular Sidebar Functions

The particular requirements for the site you are building should dictate what you put in a sidebar. Just throwing that out there, because you should never make design decisions based on something like, “this other blog I saw had a tag cloud so my blog should have a tag cloud.” Your blog should have a tag cloud if you use a lot of tags to classify your Posts and you think your audience will benefit from being able to navigate your site in that way. That being said, there are a number of popular functions that can be useful in the sidebar setting. Let’s take a look:

- **List recent posts**

Perhaps your homepage design displays only one Post at a time. Or perhaps it lists several, but you want to give people access to the latest ten Posts. There is a special function for displaying such a configuration. In typical WordPress fashion, it is a custom function that accepts a number of parameters that can be useful in lots of situations.

```
<?php wp_get_archives(array(
    'type' => 'postbypost',      // or daily, weekly, monthly, yearly
    'limit' => 10,                // maximum number shown
    'format' => 'html',          // or select (dropdown), link, or custom
    'show_post_count' => false,  // show number of posts per link
    'echo' => 1                  // display results or return array
)); ?>
```

This will output a list of linked Post titles according to the specified parameters. You can’t display excerpts or any other information from the Post, however; for that you’d need to run a custom loop.



Example from:

<http://webdesignerwall.com>

• Display a tag cloud

If you decide to tag the content on your blog, a “tag cloud” may be the ideal way to offer users navigation of that particular taxonomy.

```
<?php wp_tag_cloud(array(
    'smallest' => 10,          // size of least used tag
    'largest' => 18,           // size of most used tag
    'unit' => 'px',            // unit for sizing
    'orderby' => 'name',       // alphabetical
    'order' => 'ASC',          // starting at A
    'exclude' => 6              // ID of tag to exclude from list
)); ?>
```

These are just a few example parameters, see the Codex at <http://digwp.com/u/30>.

• List of categories

Categories can sometimes act as the main navigation for your site. Or, they can be a secondary form of navigation to display in a sidebar. Either way, displaying a list of them dynamically is pretty easy.

```
<?php wp_list_categories(array(
    'orderby' => 'name',        // alphabetical
    'order' => 'ASC',           // starting at A
    'show_count' => 0,          // do NOT show # of posts per cat
    'title_li' => __('Categories'), // include title list item
    'exclude' => 12,             // ID of category to exclude
    'depth' => 0                 // levels deep to go down cat tree
)); ?>
```

These are just a few example parameters, see the Codex at <http://digwp.com/u/31>.

• Show the blogroll

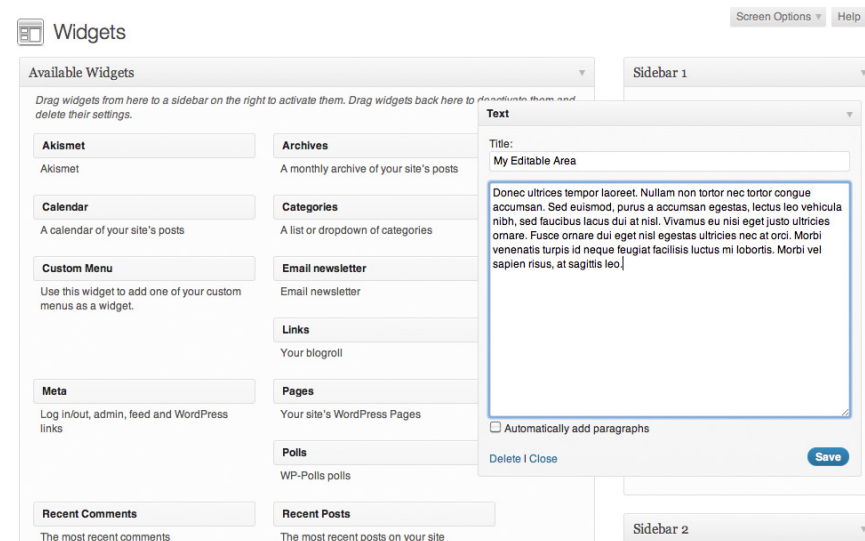
In the Admin area, there is an entire area just for “links” (located in the same area with Posts and Pages). This area was once commonly referred to as the “Blogroll” but that term has gone a bit out of fashion. Regardless of what it’s called, the list of links managed in this area may be easily displayed with this:

```
<?php wp_list_bookmarks(array(
    'orderby' => 'name',           // alphabetical
    'order' => 'ASC',              // starting at A
    'limit' => -1,                 // unlimited, show ALL bookmarks
    'title_li' => __('Bookmarks'), // include list item title
    'title_before' => '<h2>',      // tag before title
    'title_after' => '</h2>',      // tag after title
)); ?>
```

These are just a few example parameters, see the Codex at <http://digwp.com/u/32>.

• Editable text area

One of the things you may wish to include in a sidebar is an area of text that you can edit from the back end. There are a couple of different ways to do this. One way would be to use a “global custom field” (see page 66). Another way would be to use widgets. Widgets are WordPress’ way of allowing management of regions through the Admin, rather than having to edit theme files. One of the many different types of widgets is a generic “text” widget. If you have a “widgetized” sidebar, you can just drag this over into that area, enter your info, and save it.



Links Manager

As of WP 3.5, the Links Manager (aka blogroll) is no longer included with new installations. To restore its functionality, use a plugin:

<http://digwp.com/u/639>

3.6.4 Widgets, Widgets, Widgets

“Widgetizing” a sidebar, or any other region for which you wish to have manageable areas, is pretty easy. And because widgets are standardized, plugins can make use of them and deliver special widgets that you can control directly from within the Admin area. We’ll look more into widgetization in section 4.8.1, but for now, this is the code that you would place into your theme file:

```
<div id="sidebar">
  <ul>
    <?php if (!function_exists('dynamic_sidebar') || !dynamic_sidebar()) : ?>
      <li><!-- stuff shown here in case no widgets active --></li>
    <?php endif; ?>
  </ul>
</div>
```

Now in your `functions.php` file, you “register” the sidebar with a custom function:

```
if (function_exists('register_sidebar')) {
    register_sidebar(array(
        'before_widget' => '<li id="%1$s" class="widget %2$s">',
        'after_widget'  => '</li>',
        'before_title'  => '<h2 class="widgettitle">',
        'after_title'   => '</h2>',
    ));
}
```

More Sidebar Ideas

For more great techniques and ideas for crafting the perfect sidebar, check out chapter 4.4.1, “Side Content and Useful Menu Items.”

Now every widget you add will appear inside list tags with corresponding `<h2>` headings, which should fit well into the surrounding markup. Each widget will also have a unique ID and common class name (for potential CSS styling).

3.7.1 The Search Form

WordPress provides built-in search functionality, and you should definitely share it with your visitors. It is an expected feature of most websites, providing a useful way for visitors to locate their favorite content.

3.7.2 Why is This a Separate File?

It is very common for a WordPress theme to have a file called `searchform.php`. This is because the search form may be used in multiple theme files and in different locations, so placing it in its own file helps to keep things modular. Just like the sidebar, which you can include at your leisure with the `get_sidebar()` function, you can include your `searchform.php` in any other template file by calling this function:

```
<?php get_search_form(); ?>
```

This function accepts no other parameters, but of course if you had a good reason to rename `searchform.php` or keep it anywhere other than the root directory of your theme, you could just use the standard include code instead:

```
<?php include(TEMPLATEPATH . '/inc/special-search-form.php'); ?>
```

Where might you use this?

- On the 404 page (`404.php`)
- In the “else” part of The Loop
- In the sidebar

3.7.3 Alternatives to WordPress Search

The fact of the matter is that the WordPress built-in search kind of sucks. It lists things in chronological order based on whether or not it found any of your search

Search Everything

Instead of just looking at post titles and content to locate matching search terms, wouldn't it be neat if WordPress searched through everything in your database?

The Search Everything plugin does exactly that, telling WordPress to search through comments, drafts, attachments, and just about everything else.

<http://digwp.com/u/400>

terms. When searching for your query, WordPress looks in the titles and content of your posts and pages. If the search terms aren't located there, WordPress will tell you that nothing can be found.

To make things worse, there is no *advanced* search functionality, meaning you have very little control as a user *or* as a theme developer as to how results are refined, returned, and displayed. There are ways of hacking together a decent WordPress search system, but it takes quite a bit of meddling.

A much easier way to improve the WordPress' default search functionality is either to replace it entirely with Google or install a plugin that will beef things up for you.

Google Search <http://digwp.com/u/53>

As it happens, an alternative to WordPress search is to just use Google search instead. Not just generic Google full-web search, but rather a free service called the Google Custom Search Engine, with which you can integrate customized, site-specific search engines into any site. It's easy and super-awesome.

Search API Plugin <http://digwp.com/u/54>

This plugin provides a far more powerful searching system than the default WordPress search. With the Search API plugin, you can search different types of content, search within particular categories, use search operators like AND, OR, and NOT, and even search using a more comprehensive "fulltext" MySQL search. As if that weren't cool enough, this plugin also integrates with Google Custom Search.



Each of the different search choices on CSS-Tricks activates a different Google Custom Search Engine.

3.8.1 The Footer

Just like headers and sidebars, footers are one of those ubiquitous design elements. They are so commonly used that WordPress again has a special template tag for including them into your theme:

```
<?php get_footer(); ?>
```

This function will accept only one parameter, a string, which works like the sidebar function. Used without a parameter it will fetch the `footer.php` file and insert it. When used with a parameter like so...

```
<?php get_footer("mini"); ?>
```

```
<?php get_footer("mega"); ?>
```

...the `get_footer()` template tag will retrieve the theme files `"footer-mini.php"` and `"footer-mega.php,"` respectively.

3.8.2 The `wp_footer()` Hook

Remember the `wp_head()` hook? Well, the `wp_footer()` hook is exactly like that, only used down in the footer instead. It tells WordPress, "the footer is right here." All by itself, it doesn't do anything, it's just a generic hook that can be used to which scripts and other functionality may be attached.

For example, it is fairly common practice to load HTML-dependent JavaScript files from within the footer instead of the header. The location of the `wp_footer()` hook within your `footer.php` file will determine where the JavaScript is displayed in the source code. Thus, a good place for this hook is just before the closing `<body>` tag.

```
<?php wp_footer(); ?></body>
```

Mini footer



JeffCampana.com

Just a thin black bar with a copyright and a small illustration. Even that small bit of text could be kept dynamic:

```
&copy; <?php  
echo date("Y");  
bloginfo('name'); ?>
```

Mega footer



YoDiv.com

Enormous section of content displayed with a clever “underground” design. This could have been accomplished any number of ways, but probably most practically by using the Links area in the Admin and using wp_list_bookmarks() to generate the different categories of links.

3.8.3 Mini Footers / Mega Footers

Like your sidebar, the design of your footer should serve the needs of your site and its audience. If you don't need the room in your footer, no need to junk it up with unnecessary stuff. At the same time you shouldn't be afraid to do something interesting or innovative if you have appropriate content and the desire to do so.

3.9.1 Theme Functions

WordPress themes can include a special file, `functions.php`, which gives you a lot of control and power over things that happen in your theme. Think of it like a file that can do anything a plugin can do, *without* needing a plugin. Neat eh? This allows for a lot of cool theme functionality that was not possible in the days before this file existed. The `functions.php` file is also beneficial for themes that need to be "bundled" with certain plugins.

3.9.2 Functions are for Specific Themes

Because you can accomplish similar things with plugins as you can with custom functions (i.e., the `functions.php` file), some rational decisions should be made about where to do what.

Because the functions contained within the `functions.php` file reside within the theme folder itself, the code inside it depends on that particular theme being active in order to run. Plugins on the other hand, remain operational (or inoperational, as the case may be) regardless of which theme is active. Thus, don't put anything in `functions.php` that is required by multiple themes. Likewise, don't do anything with a plugin that is only relevant to the current theme.

Examples:

Hide WordPress upgrade notification bar	Use a plugin
Add button to Post Editor	Use a plugin
Load jQuery from Google	Use functions.php
Replace default Gravatar image	Use functions.php

In the top two examples, the desired result is related to the Admin area and so has nothing to do with the theme itself. You should use a plugin here, because they are theme-independent and won't stop working when you change themes.

In the bottom two examples, those two things are specific to a theme, and thus should happen in the `functions.php` for that theme.

3.9.3 Advantage Over Core Hacks

In the dark days before `functions.php`, modifying and customizing WordPress functionality was often a matter of altering "core" files. There are a number of problems with this. First and foremost, you could break something in a pretty substantial way and have no clear path to restored functionality. Equally as important, upgrading WordPress means upgrading core files. As in, overwriting them. You would lose your alteration entirely unless you documented exactly what you did; and even then you would have to trudge through and manually re-implement your changes with every new version of WordPress (and there are plenty, trust us). As you do this, you will also need to account for any changes in the core files and adapt your hack accordingly. A real pain in the keyboard, if you know what we mean. The worst part of this nasty practice is that, after going through a few upgrades, you will probably throw in the towel and stop upgrading

to new versions. This of course is bad because it leaves you open to potential vulnerabilities and missing out on the latest and greatest cool features.

Fortunately hacking the WordPress core is rarely needed these days because there are so many hooks available to latch onto and literally rewrite content or append new functionality.

Up next...

Now that we are familiar with the anatomy and basic functionality of a WordPress theme, let's dig deeper into WordPress and learn as much as we can about theme design and development. Strap yourself in, it's going to be a wild ride!

If You Must...

As bad a practice as it is to hack the WordPress core, there may be situations in which doing so is the only way to accomplish your goals (WordPress isn't all-powerful, after all). If you find yourself contemplating a little core hacking, be sure to check out chapter 5.4.3 – Hacking the WordPress Core – for some helpful tips.

It may not look like a typical WordPress site, but it is! All the quotes seen in this book are from Quotes on Design.

<http://quotesondesign.com>



My rates are as follows:

\$50/hour

\$75/hour if you watch

\$100/hour if you help

– **CLASSIC AUTOBODY SIGN**

4

Theme Design and Development

4.1.1 Customizing the Loop

As we mentioned in the previous chapter, if there is one thing that you need to understand to be a true badass theme developer, it is how the loop works and what you can do with it. With this as our mantra, let's spend the first part of this chapter looking at **three ways** to make the WordPress Loop do whatever you want.

Here is a quick summary of *three ways* to customize loop functionality:

query_posts() - <http://digwp.com/u/506>

If you are working with a *single loop* and just want to modify the type of posts that are returned, use `query_posts()`. It's perfect for limiting the number of posts, excluding posts from a certain category or tag, and so on.

WP_Query() - <http://digwp.com/u/505>

For a powerful way to customize *multiple loops*, use `WP_Query()`. By setting up additional instances of `WP_Query()` in your theme template, you can create any number of multiple loops, and customize the output of each.

get_posts() - <http://digwp.com/u/504>

Last but not least, use the simple `get_posts()` function to easily create additional loops anywhere in your theme. `get_posts()` accepts the same parameters as `query_posts()`, and is perfect for adding custom loops to your sidebar, footer, or anywhere else in your theme.

Let's walk through some examples to illustrate these three techniques.

3 Ways to Customize

There are three main ways to customize & modify the Loop:

`query_posts()`

Place before the loop to do stuff like limit the number of posts and exclude categories.

`WP_Query()`

Powerful class that handles request information for WP. Used to create any number and type of multiple loops.

`get_posts()`

A simple function for creating and customizing additional loops. Accepts the same parameters as `query_posts()`.

There is no loop

Another great tool for customizing the main loop (i.e., not a secondary loop) is the `pre_get_posts()` hook, which is called after the query variable object is created, but before the actual query is run. More info and examples:

<http://digwp.com/u/637>

4.1.2 Customizing the Loop with `query_posts`

The default WordPress loop typically looks something like this:

```
<?php // The WordPress Loop
if (have_posts()) : while (have_posts()) : the_post();
...
endwhile; else:
...
endif;
?>
```

The post information displayed by this loop depends on numerous factors, including the requested URL and corresponding theme template used to generate the page. For example, when someone visits your home page, the default loop will display posts from any category up to the number of posts specified in the Admin.

But let's say that instead of showing posts from *any* category, you want to exclude your "Asides" category and maybe display those posts on a separate page somewhere. This is a perfect job for the `query_posts()` function, which easily excludes all posts from the Asides category, which we'll say has a `tag_ID` of "9":

```
<?php // The WordPress Loop - customized with query_posts
query_posts('cat=-9'); // exclude Asides category (tag_ID = 9)
if (have_posts()) : while (have_posts()) : the_post();
...
endwhile; else:
...
endif;
?>
```

Those three dots

What are those three dots that appear between the lines of code? That's just shorthand notation for "whatever you want" – basically any PHP template tags and/or markup needed to structure your post content.

That does the job of excluding all posts from the Asides category, but we need to ensure that the original query is available to the loop. We also want to reset the query once we are done with it. With these new modifications, we get this:

```
<?php // The WordPress Loop - customized with query_posts
global $query_string; // grab the global query information
$posts = query_posts($query_string.'&cat=-9'); // exclude Asides category
if (have_posts()) : while (have_posts()) : the_post();
...
endwhile; else:
...
endif;
wp_reset_query(); // reset the query
?>
```

Using this basic template for customizing loops, it is easy to modify just about anything. For example, with a few additional parameters, we can limit to three posts, reverse post order, and exclude more categories:

```
$posts = query_posts($query_string.'&cat=-7,-8,-9&posts_per_
page=3&order=ASC');
```

The key to using `query_posts` successfully is keeping the original query via the `$query_string` variable. Once it's included, you can add as many parameters as needed to achieve the desired loop functionality (see sidebar note for details).

Yes, it's easy and powerful, but not without drawbacks. `query_posts` uses additional database queries and may affect conditional template tags such as `is_page()` and `is_single()`. Fortunately, in those cases where `query_posts` just won't suffice, we can use either `WP_Query` or the simpler `get_posts` function to make it happen.

Be careful..

Note that the `query_posts` method of customizing loops is no longer recommended. It still works as described, but as discussed there are some drawbacks to consider. In general, use either of the other methods (i.e., `WP_Query` or `get_posts`) to create or modify additional loops as needed. To modify the main loop, use the `pre_get_posts()` hook:

<http://digwp.com/u/637>

`wp_reset_query()`

The `wp_reset_query` tag restores the original query for use by subsequent loops and template tags. It's only required when using `query_posts` to modify the original query, so not needed for the default loop.

More Parameters

As of WordPress 2.6, both `get_posts` and `query_posts` accept the same parameters as the `WP_Query` class. Check the WordPress Codex for a complete list (there are many):

<http://digwp.com/u/507>

4.1.3 Customizing the Loop with WP_Query

For complete control over the customization of *any number of loops*, WP_Query is the way to go. When used to modify a default loop, it looks similar to query_posts:

Same Parameters

All three methods of setting up and customizing multiple loops accept the same parameters, which are based on those of the WP_Query class. As both query_posts and get_posts use WP_Query, they accept the same parameters. So regardless of which method you use to customize your loops, the parameter syntax is the same.

```
<?php // The WordPress Loop - customized with WP_Query
$custom_query = new WP_Query('cat=-9'); // exclude Asides category
while($custom_query->have_posts()) : $custom_query->the_post();

...
endwhile;

wp_reset_postdata(); // reset the query
?>
```

It also accepts the same parameters as query_posts, so modifying stuff like number of posts, included/excluded categories, and post order looks quite similar. As seen in the following examples, WP_Query makes it easy to customize the loop by simply changing the parameter:

```
$custom_query = new WP_Query('cat=-7,-8,-9'); // exclude any categories
$custom_query = new WP_Query('posts_per_page=3'); // limit number of posts
$custom_query = new WP_Query('order=ASC'); // reverse the post order
```

As expected, we can combine parameters with WP_Query using the same syntax as both query_posts and get_posts. Here's the equivalent of our query_posts example:

```
$custom_query = new WP_Query('cat=-7,-8,-9&posts_per_page=3&order=ASC');
```

Notice, however, that with WP_Query, we don't need the \$query_string variable.

In addition to using WP_Query to customize the *default loop*, we can also use it to create and customize *multiple loops*. Here is a basic example:

Order doesn't matter

In case you were wondering, multiple parameters may be included in any order. The only exception is for the \$query_string variable included when using query_posts(). See chapter 4.1.2 for details.

```

<?php // Loop 1
$first_query = new WP_Query('cat=-1'); // exclude category
while($first_query->have_posts()) : $first_query->the_post();
...
endwhile;
wp_reset_postdata(); // reset the query

// Loop 2
$second_query = new WP_Query('cat=-2'); // exclude category
while($second_query->have_posts()) : $second_query->the_post();
...
endwhile;
wp_reset_postdata(); // reset the query

// Loop 3
$third_query = new WP_Query('cat=-3'); // exclude category
while($third_query->have_posts()) : $third_query->the_post();
...
endwhile;
wp_reset_postdata(); // reset the query
?>

```

Each of these additional loops may be placed anywhere in your theme template – no need to line them up sequentially. For example, one loop may be placed in your sidebar, another in your footer, and so on. And with the output of each loop easily modified using any of the available parameters, any loop configuration is possible.

As you can see, using WP_Query is a good choice for really customizing some multiple-loop functionality, but for most cases, there is a better way to do it.

Mix & Match

If necessary, it is possible to use any combination of query_posts, get_posts, and WP_Query for your customized multiple loops. When possible, however, aim for consistency when setting up your loops.

Paging with WP_Query

With the default loop, queried posts are paginated so that visitors can view Previous Posts and Newer Posts. To keep this paging functionality when customizing loops with WP_Query, append “&paged=’.\$paged” to the parameter string, like so:

```

new WP_Query(
'cat=-9&paged=’.$paged
);

```

New in WP 3.3

Version 3.3 includes a new is_main_query() function that lets you modify the main WP_Query object only, perfect for customizing multiple loops. Learn more:

<http://digwp.com/u/580>

4.1.4 Customizing the Loop with get_posts

The easiest, safest way to create multiple loops in your theme is to use `get_posts`. Anywhere you need to create an additional, custom loop, just add the following:

No Pagination

Note that post pagination is not possible with `get_posts`.

Default loops and those created and modified via `query_posts` and `WP_Query` support post-navigation using `next_posts_link` and `previous_posts_link` tags. These tags don't work with `get_posts`, which displays a simple 'static' set of posts that match your criteria.

So, if you need to navigate to the previous/next set of posts, use `WP_Query` for your multiple custom loops.

```
<?php // additional loop via get_posts
global $post;
$args = array('category' => -9); // exclude Asides category
$custom_posts = get_posts($args);
foreach($custom_posts as $post) : setup_postdata($post);
...
endforeach;
?>
```

This code creates a loop of all posts *except* those in the excluded Asides category. Of course, excluding a category is just *one* way to customize your additional loops. By using any of the same parameters accepted by `WP_Query` and `query_posts`, it is possible to create loops that display just about anything you want.

Notice, however, that `get_posts` requires the use of an array for the parameters. The format for multiple parameters looks like this (using our previous example):

```
$args = array('category'=>-7,-8,-9, 'numberposts'=>3, 'order'=>'ASC');
```

Also notice that we're using `numberposts` instead of `posts_per_page` to limit the number of posts. According to the WP Codex, `posts_per_page` *should* work with `get_posts`, but if it doesn't just go with `numberposts`. There is also a `showposts` parameter that also seems to work fine with `get_posts`.

The bottom line for customizing default loops and creating multiple loops:

- To modify the default loop, use `query_posts()`
- To modify loops and/or create multiple loops, use `WP_Query()`
- To create static, additional loops, use `get_posts()`

4.1.5 The Loop Doesn't Care About Markup

It's true. Here is a pretty "normal" loop, enclosed in a single <div>:

```
<?php if (have_posts()) : while (have_posts()) : the_post(); ?>
    <div <?php post_class(); ?> id="post-<?php the_ID(); ?>">
        <h2><a href="<?php the_permalink(); ?>"><?php the_title(); ?></a></h2>
        <p class="meta">Posted on <?php the_time('F jS, Y'); ?></p>
        <?php the_content('Read More'); ?>
        <p><?php the_tags('Tags: ', ' ', ' ', '<br />'); ?>
        Posted in <?php the_category(', '); ?></p>
    </div>
<?php endwhile; ?>
    <div class="next-posts"><?php next_posts_link('&laquo; Older Entries') ?></div>
    <div class="prev-posts"><?php previous_posts_link('Newer Entries &raquo;') ?></div>
<?php endif; ?>
```

And here is the same exact loop, only marked up as a simple ordered list:

```
<?php if (have_posts()) : ?>
    <ol>
        <?php while (have_posts()) : the_post(); ?>
            <li <?php post_class(); ?> id="post-<?php the_ID(); ?>">
                <h2><a href="<?php the_permalink(); ?>"><?php the_title(); ?></a></h2>
                <p class="meta">Posted on <?php the_time('F jS, Y'); ?></p>
                <?php the_content('Read More'); ?>
                <p><?php the_tags('Tags: ', ' ', ' ', '<br />'); ?>
                Posted in <?php the_category(', '); ?></p>
            </li>
        <?php endwhile; ?>
    </ol>
<?php endif; ?>
```


And here it is again as a definition list:

```
<?php if (have_posts()) : ?>
    <dl>
        <?php while (have_posts()) : the_post(); ?>
            <dt <?php post_class(); ?> id="post-<?php the_ID(); ?>">
                <a href="<?php the_permalink(); ?>"><?php the_title(); ?></a>
            </dt>
            <dd>
                <?php the_excerpt(); ?>
            </dd>
        <?php endwhile; ?>
    </dl>
<?php endif; ?>
```

Excerpt Length

By default, the_excerpt() displays only the first 55 words of the post. So when using, you'll get post excerpts for any post greater than 55 words in length. Anything less and the_excerpt will show the entire post.

Notice that not only is the markup different in each of these examples, but the functions we use are different as well. For example, the definition list example uses the_excerpt() instead of the_content(), which only displays a portion of the entire post (see side note). This might be more appropriate for, say, a get_posts loop in the sidebar that displays recent posts. The take-home message here is that the WordPress Loop is a flexible, powerful thing — you can use any markup and template tags and arrange them however you wish.

4.1.6 The Power of WP_Query

A lot of the magic ahead is accomplished by using the WP_Query function, which is definitely worth getting to know! As discussed in chapter 4.1.3, a new instance of WP_Query enables you to customize the perfect loop with a few simple parameters.

As we learned in chapter 3.4.3, every execution of the loop is based on a default query that changes according to the type of page being viewed, site settings, and so on. This default query may be replaced or modified with the WP_Query class.

In many cases, you might want to preserve the original query and adjust only certain parameters. To do this, use `WP_Query` to setup the following loop structure:

```
<?php // Customized Loop via WP_Query
    $custom_query = new WP_Query('cat=10&showposts=2&paged='.$paged);
    while($custom_query->have_posts()) : $custom_query->the_post();
        ...
    endwhile;
    next_posts_link('&laquo; Older Entries');
    previous_posts_link('Newer Entries &raquo;');
    wp_reset_postdata(); // reset the query
?>
```

As in previous examples, by changing the parameters in your new `WP_Query`, you can easily customize the loop to display virtually any set of posts. Let's look through some more tasty "recipes" for common things you might want to do with the loop.

4.1.7 Displaying Different Numbers of Posts

There is a global setting in the Admin area for setting the number of Posts to display. In chapter 2.7.4, we recommend the *Posts Per Page* plugin for more fine-grained control of this setting, but you can also control it directly through the loop itself. For example, here we are overriding the default number of posts to display:

```
$custom_query = new WP_Query('showposts=6&paged='.$paged);
```

This custom query will limit the number of posts to six. By changing the "6", you may display any number of posts you wish. To display all of your posts, set the value of the `showposts` parameter to "-1".

4.1.8 Excluding Specific Categories

One of the most common customization requests for WordPress is, “how do I prevent posts in a certain category from displaying on my home page?” Say you have a category about hamsters that, for whatever reason, you want to omit from the post loop on your home page. You still want hamsters content in your feeds, just not on your website. How would we do something like this? There are multiple ways to handle this request, but the *simplest* solution is to be found in... yep, you guessed it, the `WP_Query` class:

```
$custom_query = new WP_Query('cat=-3&paged='.$paged); // no hamsters
```

In general, a dash (or minus sign, “-”) appearing before a parameter value signifies *exclusion*. Thus, here we are excluding category number three, which happens to be our hamsters category. When this query is used for our custom `WP_Query` loop, posts about hamsters will *not* be displayed.

Other possible ways to exclude a category include using PHP’s “continue” function to advance the loop prematurely, hiding content with CSS or JavaScript, hacking the core, and using a plugin. Even so, `WP_Query` is the cleanest and most flexible.

NOT

...on the Home Page.



4.1.9 Changing the Sort Order

WordPress is sort of a LIFO application by default — “Last In First Out”. It’s a smart default, because most blog-like sites like to show off their newest content first. This gives people a reason to return since they know the new stuff will be displayed up front and center. However, that might not be the ideal configuration for every site.

Say you were using WordPress to write and present a novel. You were writing it and releasing it chronologically chapter by chapter. So when visitors came to your site, you want to show them chapter One, but that is actually your oldest post, so it will be buried beneath the newer chapters. No problem, just reverse the sort order:

```
$custom_query = new WP_Query('orderby=date&order=ASC&paged='.$paged);
```

4.1.10 Show Specific Pages, Embed a Page within a Page

Another interesting and useful loop trick is to use a new WP_Query class to display only one specific Page. This could be useful for a variety of reasons, including a situation where you needed to embed the contents of one Page within another Page. Here is an example of a query that will display the About page:

```
$custom_query = new WP_Query('pagename=about&paged='.$paged);
```

`pagename=about`

Note that the pagename value used here refers to the actual slug for the page. See chapter 2.3.5 for help with slugs.

4.1.11 Using Multiple Loops: An Example

There is nothing holding you to using only one loop. You can have as many loops as you'd like. In fact, it's fairly common to have multiple loops going in a theme. A simple example would be a homepage that shows the full content of the newest three posts. And then in the sidebar, there is second loop which displays the excerpt of the *next* seven posts. Nothing wrong with it, works just fine.

Let's look at a more complex example. Let's say we have some fancy four-column theme. We have a left sidebar where we want to show a list of excerpts from the "events" category. Then we have a main column, split into two, where we display the ten most recent posts in two columns of five each. Then in the right sidebar, we show another ten posts, beginning after the posts featured in the main column.

For this setup, we are going to include four loops within the theme file(s) required to generate our fancy four-column layout. For the sake of simplicity, we will say

1 - Left Sidebar

```
new WP_Query(
'cat=7&showposts=3'
);
```

*Give me three posts
from category seven*

2 - Left Column

```
new WP_Query(
'cat=-7&showposts=5'
);
```

*Give me five posts **not**
from category seven*

3 - Right Column

```
new WP_Query(
'cat=-7
&showposts=5
&offset=5');
```

*Five more posts **not** from
category seven (skip five)*

4 - Right Sidebar

```
new WP_Query(
'cat=-7
&showposts=10
&offset=10');
```

*Ten more posts **not** from
category seven (skip ten)*

that all of these loops are contained within our `index.php` file, even though in practice the loops could be located in any combination of theme files.

Now that our four loops are in place, we need to ensure they are going to deliver our carefully planned display of posts. If we were to simply plop down four default WordPress loops, our web page would display four columns, each containing the exact same posts. So, to massage our loops into shape, we turn again to the powerful `WP_Query` class.

As shown in the previous diagram, we add two or three (depending on the loop) parameters to each of our four `WP_Query` loops. These parameters result in the following behavior:

- **The “cat” parameter**

- Loop 1 - display posts *only* from category seven
- Loop 2 - display posts *not* from category seven
- Loop 3 - display posts *not* from category seven
- Loop 4 - display posts *not* from category seven

- **The “showposts” parameter**

- Loop 1 - display *only* three posts
- Loop 2 - display *only* five posts
- Loop 3 - display *only* five posts
- Loop 4 - display *only* ten posts

- **The “offset” parameter**

- Loop 1 - (not used)
- Loop 2 - (not used)
- Loop 3 - skip the first five posts
- Loop 4 - skip the first ten posts

Multiple & Custom Loops

A favorite topic at PerishablePress.com is the WordPress Loop. If you are looking for more in-depth information on creating and using multiple & custom loops, scan through the library of articles in the WordPress “loop” archive:

<http://digwp.com/u/402>

At this point, the loops have each been modified to display the desired posts. The only other thing to consider before calling it a night is whether or not we want to display post navigation links for one of our loops. In a default single-loop setup, we can use the `posts_nav_link()` template tag to display post navigation links on

our home, index, category, and archive pages. Such navigation links would enable visitors to check out previous or more recent posts, depending on page view.

For this example, we'll go ahead and paginate the the second loop so that visitors can navigate through your posts as displayed in the Main Left column.

With this plan, our multiple loop configuration looks like this:

```
// Loop 1 - Left Sidebar
    $first_query = new WP_Query('cat=7&showposts=3');
    while($first_query->have_posts()) : $first_query->the_post();
        the_excerpt();
    endwhile;
    wp_reset_postdata();

// Loop 2 - Left Column
    $second_query = new WP_Query('cat=-7&showposts=5&paged='.$paged);
    while($second_query->have_posts()) : $second_query->the_post();
        the_content();
    endwhile;
    next_posts_link(); previous_posts_link();
    wp_reset_postdata();

// Loop 3 - Right Column
    $third_query = new WP_Query('cat=-7&showposts=5&offset=5');
    while($third_query->have_posts()) : $third_query->the_post();
        the_content();
    endwhile;
    wp_reset_postdata();

// Loop 4 - Right Sidebar
    $fourth_query = new WP_Query('cat=-7&showposts=10&offset=10');
    while($fourth_query->have_posts()) : $fourth_query->the_post();
        the_content();
    endwhile;
    wp_reset_postdata();
```

WP Page Navigation

For a definitive guide to WordPress page navigation, check out Digging Into WordPress:

<http://digwp.com/u/401>

Loop 1 - Left Sidebar

Display three posts from category seven in the left sidebar. We display excerpts here with the_excerpt instead of the_content.

Loop 2 - Left Column

Display five posts not from category seven, and include the \$paged parameter to paginate the results. After the loop, we enable post navigation using the next_posts_link & previous_posts_link template tags.

Loop 3 - Right Column

Display five more posts not from category seven. We also use an offset parameter to avoid post duplication.

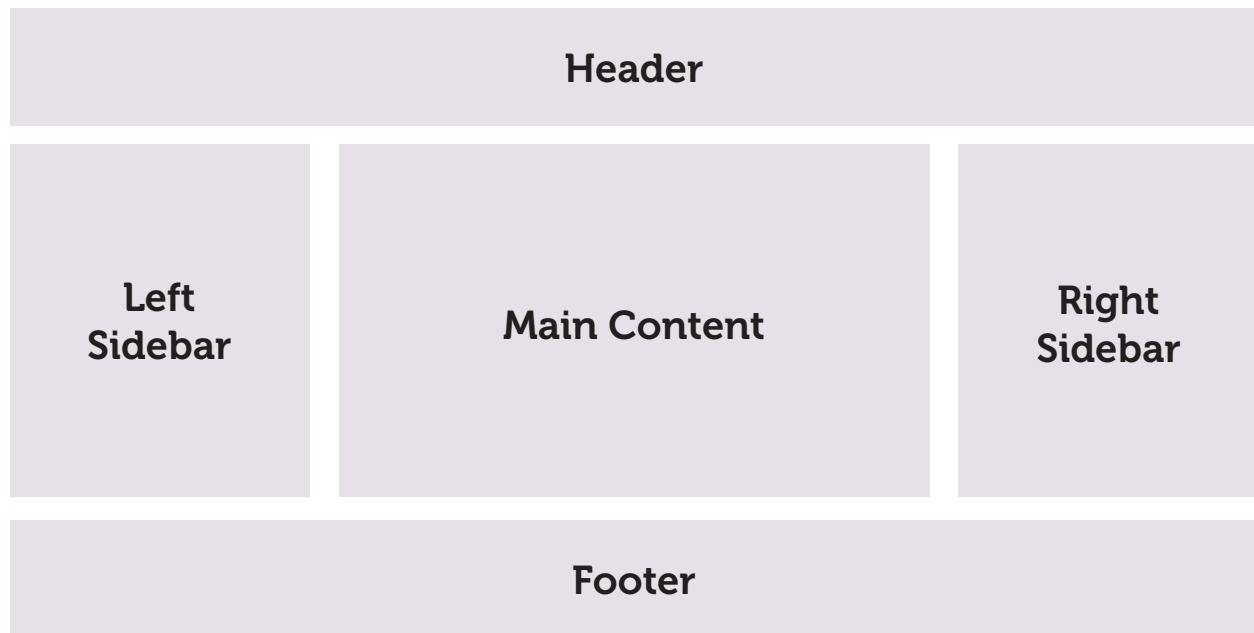
Loop 4 - Right Sidebar

Display ten more posts not from category seven. Again, the offset parameter is used to avoid duplication of posts.

Now *that* is a thing of beauty! And you can use this example as a template when setting up your own custom loops. Of course, you should add some HTML and template tags to flesh things out and structure the post content. If the goal is to create multiple, custom loops with pagination, this template will get you started.

4.2.1 Sidebars and Footers

Sidebars and footers are such ubiquitous design elements that there are special functions for calling them, just like the header. Let's take an example like this:



Ah, notice the two sidebars? That is an excellent way of presenting lots of information to your visitors. Here's how it's done...

4.2.2 Multiple Sidebars

The code for creating a layout like the one shown above looks like this:

```

<?php get_header(); ?>
<?php get_sidebar(); ?> ← first sidebar appears here
    <div id="main-content">
        // Loop
    </div>
<?php get_sidebar("secondary"); ?> ← second sidebar appears here
<?php get_footer(); ?>

```

The two `get_sidebar()` template tags seen in this code should be placed in all of the major template files that are responsible for a full page layout (`index.php`, `archive.php`, `single.php`, `page.php`). You would then customize your two sidebar templates, `"sidebar.php"` and `"sidebar-secondary.php"`, with any markup and template tags that you would like. Here is a very basic example showing the HTML that might be generated from our dual-sidebar code:

```

<div id="sidebar">
    <!-- place markup and template tags here for your first sidebar -->
</div>
<div id="main-content">
    <!-- place markup and template tags here for your post loop -->
</div>
<div id="sidebar-secondary">
    <!-- place markup and template tags here for your second sidebar -->
</div>
<div id="footer">
    <!-- place markup and template tags here for your footer -->
</div>

```

Notice we didn't actually call the right sidebar "right", we called it "secondary". If someday it was decided that the sidebars should be flipped, "right" would cease to make sense.

That's what semantics means in web design – describing without specifying

http://net.tutsplus.com

```
<ul>
  <li><a href="/">Home</a></li>
  wp_list_categories();
</ul>
```

List of categories
as main nav

Breadcrumbs

```
while (have_posts())
```

Loop of most
recent content

```
wp_list_pages("include=7");
```

Static
Page

```
bloginfo('rss_url');
```

RSS Feed
Link

```
get_search_form();
```

Prominent
Search

The screenshot shows the net.tutsplus.com website. Red lines connect annotations to specific features: 'List of categories as main nav' points to the top navigation bar; 'Breadcrumbs' points to the 'HOME \ ALL POSTS' link; 'Static Page' points to the 'HOME' link in the top navigation; 'RSS Feed Link' points to the RSS icon in the top right; 'Prominent Search' points to the search bar; and 'Loop of most recent content' points to the main article area.

net tuts+ TM

tuts+ NETWORK ENGLISH SEARCH NETTUTS

HOME TUTORIALS ARTICLES FREEBIES VIDEOS ABOUT

SUBSCRIBE BY RSS EMAIL OTHER

HOME \ ALL POSTS

Voting Begins: The First 5 Screencast Contestants

16

Oct 5th in General by Jeffrey Way

Each day this week, we'll post five of the top twenty-five videos from the Nettuts+/Screenr competition. Your job, if you choose to accept it, is to review these short web development video tips and choose your favorite one in the poll below. How should you judge? Quite simply – there are no rules! Pick the one that helped you the most. The most chosen video each day will earn a spot in the top five at the end of the week!

Today, you'll learn everything from setting up your first Django project, to simplifying the beginnings of a jQuery plugin, to applying guides in Photoshop. Each video is less than or equal to five minutes.

[Continue Reading](#)

Simple Techniques to Lock Down your Website

46

Oct 2nd in PHP by Dustin Blake

Plus
Source Files, Bonus Tutorials and More for \$9 p/month
[LEARN MORE & SIGNUP](#) | [LOGIN HERE](#)

Create >>
FREE STUNNING FLASH websites

TOP 25 WEB HOSTS
TOPHOSTS.COM
The Complete Web Hosting Resource Since 1997

P2H.COM
DESIGN TO XHTML
BRILLIANTLY AFFORDABLE

W3 <MARKUP/> CSS & XHTML

DreamTemplate
AMAZING WEB TEMPLATES
5,000+ DESIGNS

This gives us the structure we need to create our three-column layout using CSS:

```
#sidebar { width: 200px; float: left; }  
#main-content { width: 500px; float: left; }  
#sidebar-secondary { width: 200px; float: right; }  
#footer { clear: both; }
```

4.3.1 Menus, Archive Lists & Tag Clouds

There are all kinds of ways to create dynamic navigation from within WordPress. This is a great thing because good navigation is key to the success of any website. When we create Pages and Posts in WordPress, at the same time we are creating that content we are creating ways in which to navigate to that content. Let's look at two examples.

A category-rich blog

Look at the popular web development blog **Nettuts+**. They have a fairly traditional blog structure, where Posts are published from day to day in a chronological format. The homepage features new Posts of all types, starting with the most recent. On a blog like this, it is likely visitors are coming to do one of two things: See what's new or find something they are looking for. The site does a good job at both, displaying a prominent search bar, and featuring new content first. For that latter group, search might not be the only thing they need. What if they aren't looking for something specific enough for search, or they can't think of a search term they would need to find it. In this case, breaking down the site into categories is perfect, because that gives that visitor an option to drill down into the site and likely get closer to what they are looking for. For example, if they were looking for an icon set to use, their intuition might lead them to click on the Freebies part of the navigation. You can see some of these key layout elements, along with the WordPress code that goes with them, on the adjacent page.

Hierarchical content

Of course we know that WordPress isn't limited to a blog structure. Equally viable is using the Page system to create *hierarchies* of content. Take for example the Snippets section on CSS-Tricks. The homepage for the Code Snippets is the proud parent of six Child Pages, which each serve as a category for a particular coding language. Then, each of the six Child Pages is the parent of many *more* Child Pages, which together comprise the growing army of code snippets.

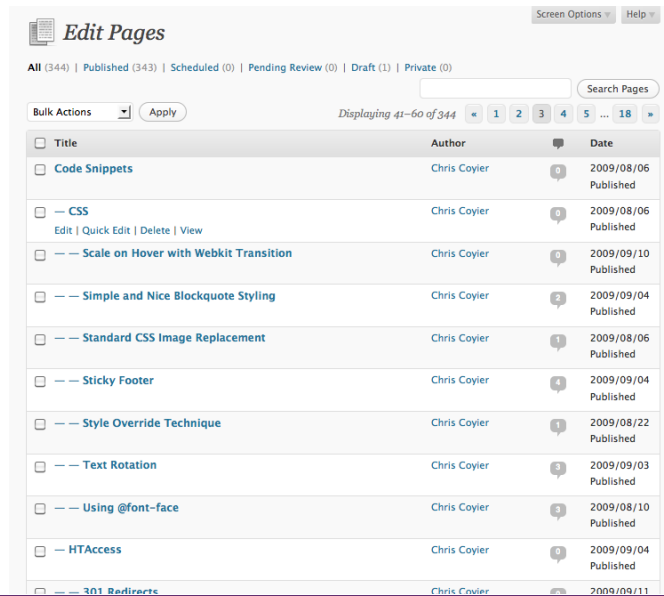
wp_list_pages

For a great collection of delicious recipes for Page menus and Page listings, check out this article at Digging Into WordPress:

<http://digwp.com/u/403>

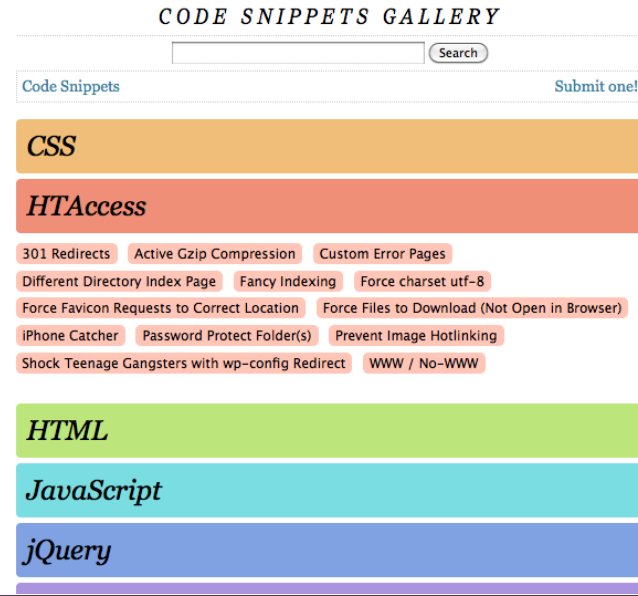
This is a whole hierarchy of content that presents all kinds of opportunities for menu creation. In this Snippets example, the Snippets homepage has a unique page template which uses `wp_list_pages` to output all of its own Child Pages. By default, that function lists not only Child Pages, but also the entire hierarchy of pages beneath it. In other words, Child Pages of Child Pages, also known as grandchildren. In the markup, nested lists represent the Hierarchical relationship of these pages.

In the CSS, the different levels of nested lists are targeted and styled differently to emphasize their hierarchy and provide intuitive navigation of the individual snippets. A little jQuery is also in effect, which makes the list act like an accordion, where each group can be opened and closed for browsing.



On the left, the Pages are set up in a hierarchy in the Admin area of WordPress.

On the right, that page structure is output with `wp_list_pages()` and styled with CSS and jQuery.



4.3.2 Page-Specific Menu Styles

In the Snippets section on CSS-Tricks (see previous section), we have a three-layer deep hierarchy of Pages. Each different level of the hierarchy has different navigational menu needs. Here is a closer look into the page-specific menu styles used for CSS-Tricks' Code Snippets.

- **Parent Page = Snippets Home Page**

The parent page uses a special template file called "page-snippet-cat.php." This unique page template allows us to do anything we want with this page, but ultimately it won't be all that different than the rest of the pages on the site. For example, the custom Snippets Page includes the exact same header, sidebar and footer as every other page on the site. The difference between the custom page and other pages is a special header section that displays all child pages with this:

```
wp_list_pages('title_li=&exclude=3285,3473&child_of='.$post->ID);
```

This function generates all of the markup required to display a nested-list menu of every single snippet posted to the site. In the parameters of this function, the title is eliminated, a few pages are excluded (e.g., the Submit Snippet page doesn't need to be shown), and the ID of the current page is specified to ensure that the menu displays *only* children of the Snippets page.

- **Children Pages = Code Languages**

Conveniently, all six children pages use the *exact* same page template as the parent page, which keeps things nice and compact. Anytime you can make a template flexible enough to handle many pages, you should. Because our `wp_list_pages` function lists pages that are the child pages of the current page, it works just fine here as well, and already includes the special Snippets section header.

With both the parent and children pages of this hierarchy, we need some special CSS and JavaScript. The CSS will apply some special styling and the JavaScript will produce the "accordion" display effect. Why can't we include this CSS code in our site's CSS file and the JavaScript code in our site's JavaScript file? Well, we

could... but 95% of the pages on this site don't need them, so it's a waste of bandwidth 95% of the time. Better to go with the extra file request only when needed. The following code is used in the <head> section of header.php:

```
<?php if (is_page_template("page-snippet-cat.php")) { ?>
    <link rel="stylesheet" type="text/css" href="<?php bloginfo('stylesheet_
directory'); ?>/css/snippets.css" />
    <script type='text/javascript' src='<?php
bloginfo('stylesheet_directory'); ?>/js/snippets.js'></script>
<?php } ?>
```

- **Grandchildren = Individual Snippets Pages**

The actual code snippets will have their own unique page template. At this point we no longer need to list child pages (the grandchildren aren't parents), so the `wp_list_pages` function is gone. Now that we are two generations down the hierarchy though, it makes sense to offer the user a navigational trail back up the directory tree. For this, we can use the excellent **NavXT** plugin, available at <http://digwp.com/u/114>, which makes outputting a breadcrumb trail as simple as:

```
<?php
    if (function_exists('bcn_display')) {
        bcn_display();
    }
?>
```

So this was all a rather specific example, but the main point is that when you use a unique page template to create a page, the sky is the limit for what kind of navigation/menu you want to offer. Think of the needs of your user when they have reached that page and try to accommodate.

4.3.3 Create the Perfect Archives Page

Of course “perfect” is a very relative term and the perfect page for your site depends on your site’s particular content and purpose. But if you want to create an effective Archives Page, here is a pretty solid approach.

Create a new page template just for archives, something like “archives.php,” and include a good variety of template tags to produce archive links to all of your content. Here are some useful tags to use on your Archives Page:

- **List of all posts organized by year**

```
<?php wp_get_archives('type=yearly'); ?>
```

- **List of all posts organized by month**

```
<?php wp_get_archives('type=monthly'); ?>
```

- **List of all posts organized by day**

```
<?php wp_get_archives('type=daily'); ?>
```

- **List of all authors**

```
<?php wp_list_authors(); ?>
```

- **List of all categories**

```
<?php wp_list_categories('title_li='); ?>
```

- **List of all tags**

```
<?php wp_tag_cloud(); ?>
```

- **List of all pages**

```
<?php wp_list_pages('title_li='); ?>
```

Putting these template tags together, the page template for our Archives would look like this:

Archive vs. Archives

In WordPress, “Archive” refers to page views including categories, tags, dates, and so on.

The term “Archives” (note the “s”), on the other hand, refers to a page that is used to display organized links to all of your site’s content.

Random Bonus Info

If you ever see code in WordPress like this:

```
__("string");
```

Where the string could be anything, that is actually a function for “localization” in WordPress. If a different localization is active, it will search for the translation for that word and return that, otherwise return what is provided.

Don't type it out..

Just a reminder that all code examples presented in the printed edition of the book are also included in the PDF for easy copy-&-paste action.

```
<?php /* Template Name: Archives */ ?>
<?php get_header(); ?>
<div id="content">
    <h2><?php the_title(); ?></h2>
    <div class="col">
        <h3>Archives by Year:</h3>
        <ul><?php wp_get_archives('type=yearly'); ?></ul>
        <h3>Archives by Month:</h3>
        <ul><?php wp_get_archives('type=monthly'); ?></ul>
        <h3>Archives by Day:</h3>
        <ul><?php wp_get_archives('type=daily'); ?></ul>
        <h3>Archives by Category:</h3>
        <ul><?php wp_list_categories('title_li='); ?></ul>
        <h3>Archives by Author:</h3>
        <ul><?php wp_list_authors(); ?></ul>
    </div>
    <div class="col">
        <h3>All Pages:</h3>
        <ul><?php wp_list_pages('title_li='); ?></ul>
        <h3>Archives by Tag:</h3>
        <?php wp_tag_cloud(); ?>
    </div>
</div>
<?php get_sidebar(); ?>
<?php get_footer(); ?>
```

4.3.4 Impress Your Visitors with a Tag Cloud

One of the *coolest* things about tags is that you can display them as a giant “cloud” of links. Tag clouds are awesome *not* because they are great navigational tools, but rather because they give the visitor a *glimpse* of the “big picture” of what your site is all about. In WordPress, tag clouds are *easy* to display and *highly* configurable, enabling *anyone* to customize the *perfect* tag cloud for their site. We could go on and on for days just *talking* about tag clouds, but instead we’ll just show you the code needed to make your own:

```
<?php wp_tag_cloud(array(
    'smallest' => 10,      // size of least used tag
    'largest' => 18,       // size of most used tag
    'unit' => 'px',        // unit for sizing
    'orderby' => 'name',   // alphabetical
    'order' => 'ASC',      // starting at A
    'exclude' => 6         // ID of tag to exclude from list
)); ?>
```

At this point in the game, the parameters used in the `wp_tag_cloud` template tag should be pretty straightforward. Just use the comments in the code to set a desired value for each of the parameters and you are off to tag-cloud heaven.

Pimp your tag cloud

For more information on the parameters available to the `wp_tag_cloud`, check out the WordPress Codex:

<http://digwp.com/u/404>

4.4.1 Side Content & Useful Menu Items

One of the *funnest* things to build is your site's sidebar. Let's look at some things to include in your sidebar that might be useful and appealing to your visitors.

4.4.2 Displaying Recent Comments

There are *three possibilities* here. Let's go from easiest to hardest.

- **Widget** - If the sidebar is widgetized, simply drag the "Recent Comments" widget (built-in) into the sidebar zone. Give it a title, specify the number you'd like to show, and save it. Done.
- **Plugin** - The **Get Recent Comments** plugin at <http://digwp.com/u/115> provides a simple function, customizable through the Admin, for displaying recent comments. It is built to be a widget, but as we just learned that functionality is obsoleted now by WordPress' own widget.
- **Custom Function** - Because comments are stored in our database, getting that data ourselves is possible. In a nutshell, we create a new function in our

functions.php file and craft an SQL query that would gather all of the most recently approved comments and return them to us. Then we'd loop through those returned results and display them one by one (code and more information available from **WPLancer** at <http://digwp.com/u/116>). Here it is:

```
<?php // display recent comments
function dp_recent_comments($no_comments = 10, $comment_len = 35) {
    global $wpdb;
    $request = "SELECT * FROM $wpdb->comments";
    $request .= " JOIN $wpdb->posts ON ID = comment_post_ID";
    $request .= " WHERE comment_approved = '1' AND post_status = 'publish'
                AND post_password = ''";
    $request .= " ORDER BY comment_date DESC LIMIT $no_comments";
    $comments = $wpdb->get_results($request);
    if ($comments) {
        foreach ($comments as $comment) {
            ob_start(); ?>
            <li>
                <a href="<?php echo get_permalink( $comment->comment_post_ID
) . '#comment-' . $comment->comment_ID; ?>"><?php echo dp_get_author($comment); ?></a>
                <?php echo strip_tags(substr(apply_filters('get_comment_
text', $comment->comment_content), 0, $comment_len)); ?>
            </li>
            <?php
            ob_end_flush();
        }
    } else {
        echo '<li>No Comments</li>';
    }
}

function dp_get_author($comment) {
    $author = "";
    if ( empty($comment->comment_author) )
        $author = 'Anonymous';
    else
        $author = $comment->comment_author;
    return $author;
} ?>
```

To use this function, simply call it from anywhere in your theme:

```
<?php dp_recent_comments(6); ?>
```

4.4.3 Displaying Recent Posts

Let's do this again from easiest to hardest.

- **Widget** - The easiest possible way to do this is through a widgetized sidebar. Just like recent comments, there is a built-in "Recent Posts" widget which will do the trick. Just drag it into your sidebar zone and save it.
- **Custom Function** - We don't have to write a custom function this time, we can just use the `wp_get_archives` function, which we already looked at a little. A simple function like this will display linked titles to recent posts:

```
<?php wp_get_archives('type=postbypost&limit=5'); ?>
```

- **Custom Loop** - Finally we could go totally home-brew and just write a custom loop to display recent posts. This offers by far the most control as we can do anything inside this loop, including accessing the full content or custom fields.

```
<?php // display recent posts
$home_brew = new WP_Query('showposts=10&orderby=date');
while($home_brew->have_posts()) : $home_brew->the_post();
    // output custom stuff here! Post title, content, custom fields..
endwhile;
wp_reset_postdata();
?>
```

Resetting

To ensure the original query is around after execution of our custom loop, we reset it with this template tag:

```
wp_reset_postdata();
```

This custom loop is easily modified with additional parameters such as offset and order. To change the number of recent posts, just change "10" to whatever.

4.4.4 Listing Popular Posts

Popularity is more loosely defined than something like “recent.” How do you define popularity? Page views? Number of comments? Back links? If you want to accommodate all the above, try the **WordPress Popular Posts** plugin, which is available at <http://digwp.com/u/117>.

This plugin logs relevant data and then makes outputting a list of popular posts as trivial as using this template tag anywhere in your theme:

Function exists?

Before using a function that was created via plugin, it is best practice to use `function_exists` before calling it. If the function doesn't exist (i.e., the plugin isn't installed or is deactivated), the function won't be called and a fatal error will be avoided. This is much better than calling a nonexistent function since PHP will halt rendering and will likely destroy your theme.

```
<?php if (function_exists('get_mostpopular')) get_mostpopular(); ?>
```

If you wanted to gauge popularity only based on the number of comments, you could achieve this with your own database query (thanks to the **Bin Blog** for the idea – <http://digwp.com/u/118>):

```
<ul class="popular-posts">
    <?php $popular_posts = $wpdb->get_results("SELECT id,post_title FROM
        {$wpdb->prefix}posts ORDER BY comment_count DESC LIMIT 0,10");
        foreach($popular_posts as $post) {
            print "<li><a href='". get_permalink($post->id) ."'>".
                $post->post_title."</a></li>\n";
        } ?>
</ul>
```

4.4.5 Listing Recently Modified Posts

You might think we'd have to go database fishing or use a plugin for this. We certainly could, but the `WP_Query` function supports an `orderby` parameter which can get the job done for us easily:

```

<?php // display recently modified posts
$home_brew = new WP_Query('showposts=5&orderby=modified');
while($home_brew->have_posts()) : $home_brew->the_post();
    // output custom stuff here! Post title, content, custom fields..
endwhile;
wp_reset_postdata();
?>

```

4.4.6 Listing Random Posts

Again the WP_Query function has our back, allowing the orderby parameter to accept "rand" to display a series of random posts:

```

<?php // display random posts
$home_brew = new WP_Query('showposts=3&orderby=rand');
while($home_brew->have_posts()) : $home_brew->the_post();
    // output custom stuff here! Post title, content, custom fields..
endwhile;
wp_reset_postdata();
?>

```

With Ajax

A neat idea with this technique is to make a page template that displays just one post and nothing else. Then use Ajax to call that URL and display it (for example, in the sidebar). We cover (and use) this technique on the Digging Into WordPress website, including a "Get Another!" button:

<http://digwp.com/u/411>

4.4.7 Import and Display Twitter

Integrating your recent tweets from Twitter can be a fun way to communicate with your web visitors and keep content on your site fresh. Twitter has a robust API system for getting at and using that data. But Twitter is not an infallible service, and in fact, slowness and downtime is a pretty common occurrence for them. Because of this, when using their API to get stuff and display it on your own pages, it should be done in such a way that won't affect the loading of the page, and won't look awful in the case of API failure.

This Twitter API communication can be done entirely through JavaScript, which is our preferred and recommended way for a few reasons:

- Connection to the Twitter API happens on the client side and keeps server load down
- If done right, doesn't affect page load time
- Data can be processed and appended to the page only upon success

So let's get it done. We are using jQuery in this book (because it's awesome) so let's keep going down that route.

Step 1: Enqueue jQuery

Put this PHP code into your `functions.php` file. This will load the jQuery library onto your page by inserting a link in the `<head>` section where you call `wp_head`.

```
if(!is_admin()) {  
    wp_deregister_script('jquery');  
    wp_register_script('jquery', ("http://ajax.googleapis.com/ajax/libs/  
jquery/1.3.2/jquery.min.js"), false, '1.3.2');  
    wp_enqueue_script('jquery');  
}
```

Facebook & Friends

Twitter is great, but there are other social-media services that you may want to integrate with WordPress. Here are some good articles to get you started:

How To Integrate Facebook With WordPress

<http://digwp.com/u/600>

How To Integrate Facebook, Twitter And Google+

<http://digwp.com/u/601>

Step 2: Load your custom script

You'll need to load a JavaScript file for this, so if you have one already going for your site, you can use that, otherwise load in a new one.

```
<script type="text/javascript" src="<?php bloginfo('template_url'); ?>/js/twitter.js">  
</script>
```

Step 3: The jQuery plugin

Next, we want to create a jQuery plugin:

```
(function($){  
  $.fn.lastTwitterMessage = function(username){  
    var $base = this;  
    if(!username || username == "") return this;  
    var url = "http://twitter.com/statuses/user_  
    timeline.json?callback=?";  
    $.getJSON(url,{count:10,screen_name:username  
    },  
    function(data){  
      if(data && data.length >= 1){  
        try{  
          var item = null;  
          for(var i = 0; i < data.length; i++){  
            if(/^@/i.test(data[i].text)) continue;  
            item = data[i]; break;  
          }  
          if(!item) return;  
          var $tweet = $("<p></p> ").text(item.text);  
          $tweet.html(  
            $tweet.html()  
              .replace(/((ftp|http|https):\\\/\\\/(\w+:{0,1}\w*@)?(\S+)(:[0-9]+)?(\/|\/([\w#!\:?.+=&%!\-\/]))?)/gi,  
                '<a href="$1">$1</a>')  
              .replace(/(^|\s)#(\w+)/g, '$1<a href="http://search.twitter.com/search?q=%23$2">#$2</a>')  
              .replace(/(^|\s)@(\w+)/g, '$1<a href="http://twitter.com/$2">@$2</a>')  
            )  
          $tweet.append(" &nbsp;<a href='http://twitter.com/' + username + "'>(&#8734;)</a>&nbsp;<span>");  
          $base.empty().append($tweet).show();  
        } catch(e) { };  
      }  
    });  
    return this; // Don't break the chain  
  };  
})(jQuery);
```

Been getting nailed with WordPress spam across the board that use "http://link" as the URL. (∞)



In the footer of CSS-Tricks, a speech bubble is displayed above Chris showing his latest tweet. With this jQuery method, should the Twitter service be unavailable, the speech bubble isn't displayed.

That plugin, when we call it, does all the heavy lifting of communicating with Twitter and pulling the latest tweet. It even does fancy stuff like turning URLs into real links, hash tags into search links, and @replies into profile links.

Step 4: Calling the plugin

You could load in another JavaScript file just for this, or just append this beneath the code you just added. We need a DOM-ready statement and then create an element to load the plugin on.

```
$(function() {  
    $("<div id='tweet'></div>").hide().appendTo("#footer")  
    .lastTwitterMessage('chriscoyier');  
});
```

That code is the magic. It waits for the page to be ready to be manipulated (DOM-ready), creates a new (hidden) element, appends it to the page (into the footer), and then calls the plugin on it. Should the plugin be successful in its duty, the new element will show up on the page, if not, it will remain hidden.

Wha?

What the heck is Delicious you ask? Delicious is a very popular social bookmarking site. You save links with annotation to your account online (so you'll never lose them). And because others do the same, Delicious is able to know and share what are the most popular links going around at any given moment.

<http://delicious.com>

4.4.8 Import and Display Delicious

There is, unsurprisingly, a number of ways to get this done. Delicious has APIs we could wrangle with (JSON or XML). Delicious serves up RSS feeds we could parse (see the next section). Delicious has JavaScript widgets that we could harness: <http://digwp.com/u/228>. There are also quite a number of plugins for WordPress that specially deal with Delicious: <http://digwp.com/u/229>.

There is one technique that stands out above the rest though, and that is a plugin which imports data from your Delicious account and creates Posts from it. It's called **Postalicious**: <http://digwp.com/u/230>. It's been a few years since it has been updated but it still works swimmingly with the current version of WordPress.

Postalicious is able to check Delicious every hour and pull in new links. You can set it to create drafts (recommended) or auto-publish posts. Simply choose a category, and Postalicious will automatically create the posts, the title, and all of the HTML formatting!

It should be noted that Delicious actually has some built-in (albeit rather hidden) functionality right in the web app for communicating with a WordPress blog. It isn't very user-friendly and is difficult to customize, so we don't recommend it.

4.4.9 Import & Display Other Feed Content

An interesting fact about WordPress that you may not know is that it includes a built-in RSS feed parser. This makes the job of fetching recent content from other sites for display on your own pretty darn easy. You'll need to include the `feed.php` file that ships with WordPress on any page you want to do feed parsing, but after that you are free to set up new SimplePie objects and do all the feed parsin' you desire. Here is a general example showing how to import and display our feed:

```
<?php // Import & Display Feed
if(function_exists('fetch_feed')) {
    $feed = fetch_feed('http://digwp.com/feed/');
    if (!is_wp_error($feed)) :
        $feed->set_cache_duration(1200); // set cache in seconds
        $feed->init(); // initialize the process
        $limit = $feed->get_item_quantity(7); // specify number of items
        $items = $feed->get_items(0, $limit); // create an array of items
    endif;
}
if ($items) { ?>
```

Quick Backstory

WordPress used to use MagpieRSS to do its feed-parsing. The project was discontinued and one of its competitors SimplePie was gaining traction in a big way. WordPress switched over to SimplePie, and now SimplePie development has ended.

Someone may pick up SimplePie and run with it, but it seems like inclusion into WordPress is the kiss of death for any feed parser.

Import & Display Feeds

For a complete tutorial on importing and displaying feeds in WordPress, check out:

<http://digwp.com/u/508>

Feed Import for pre-2.8

For the scoop on importing feeds in older (pre-2.8) versions of WordPress, check out this in-depth article:

<http://digwp.com/u/405>

September 19, 2010

Tire exhibit changes site

Just don't look for the every-
 year International Tire
 Exhibition and Conference in
 Akron, the self-proclaimed center
 of the universe for all things tire
 related.

September 19, 2010

Inochem seeks government support for Potash bid: paper

Chinese chemical conglomerate
 Inochem Group has formally
 asked the government to back a
 bid for Canada's Potash Corp , a
 local newspaper reported on
 Sunday.

Imported Feed Example

*Above is a screenshot of a
 site displaying content from
 an external RSS feed. It
 is displayed in a way that
 matches the design of the site.*

```
<ul>

<?php foreach ($items as $item) { ?>
<li>

    <div>

        <strong>
            <a href="<?php echo $item->get_permalink(); ?>">
                <?php echo $item->get_title(); ?></a>
        <strong>
            <em><?php echo $item->get_author(); ?></em>
        </div>
        <div><?php echo $item->get_description(); ?></div>
        <div><?php echo $item->get_date('j F Y'); ?></div>
    </li>
<?php } ?>
</ul>

<?php } ?>
```

Just place that code in the sidebar.php or any theme template file and enjoy feed content displayed anywhere on your site and anywhere on the page.

The first part of the script does the processing, enabling you to specify the feed, set the cache duration, and specify the number of posts.

The second part of the script provides the markup and PHP, displaying the results as an unordered list. Each post item includes a title link, author name, post content, and post date.

Of course, everything is completely customizable. For a complete list of functions, check out the SimplePie wiki at <http://digwp.com/u/140>.

4.5.1 Creating and Using Child Themes

WordPress supports an interesting theme-development technique known as child themes. The idea is that you can declare a base theme (or a “parent” theme as makes more sense with this metaphor) and override the styling of that theme without touching the base theme at all.

Why bother with this? Why not just duplicate the theme and make changes? Well there is plenty of debate around this. Some folks think child themes are a waste of time and others wholeheartedly endorse them. There are some things to think about to help you reach your own conclusion. Here are a few...

Does your theme release updates that you like to stay up to date with? Theme frameworks like to do this. If you have altered your theme, it becomes a big pain to update it with a fresh release from the author. If you are using a child theme, it's not a big deal, just replace the parent and your alterations are intact. Are you a theme author that wishes to release multiple variations on a theme? Child themes may be the way to go.

With child themes, you can offer up a theme and users can just activate the one they like the best, without any manual alterations by them or duplicate code by you. Doing so is surprisingly easy, and is very effective at maintaining efficient code while staying current with updates.

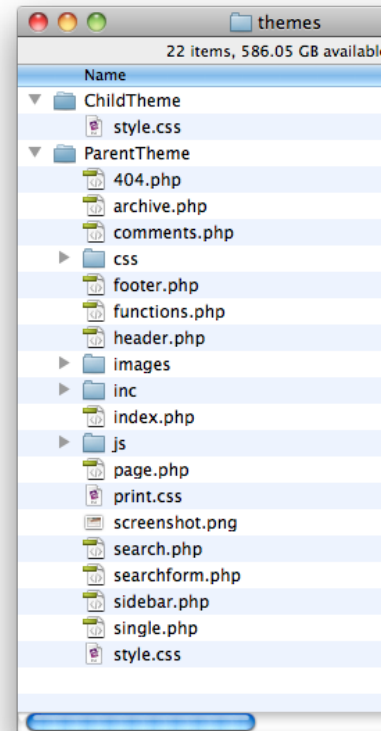
Creating a child theme is remarkably simple. Just create a folder (named as you wish) in your `wp-content/themes` directory, and within that folder place a single `style.css` file.

The CSS file will have some specially formatted comments at the top of it which will identify it, and most importantly, identify the parent theme which it intends to use. Here is an example:

```
/*  
Theme Name: My Cool Child Theme  
Theme URI: http://digwp.com/  
Description: Child Theme for Digging Into WordPress  
Author: Chris Coyier  
Author URI: http://chriscoyier.net/  
Template: DigWP  
Version: 2.0  
*/  
  
@import url("../digWP/style.css");
```

The most important line there is the “Template: DigWP” line, which references the folder of the parent theme. The last line imports the stylesheet from that parent theme. That is optional, but very common, as the whole point is to start with the parent theme as a base and overwrite/add to it. Anything you write in the CSS file below this will overwrite anything from the parent theme.

For example, if the link color is red in the parent (i.e., a { color: red; }), but you declare it green (i.e., a { color: green; }), *your* declaration in the child theme will be applied even though it uses the exact same selector, because it is declared after the parent style. No need for any fancy !important rules or more specific selectors.



4.6.1 Styling Your Theme

In any WordPress theme, the `style.css` is required. In fact, WordPress won't even recognize a folder in your themes folder unless it includes this file. It has to be there, but you aren't necessarily required to use it. You don't even have to call this CSS file from your theme if you don't want to, but since it's required and definitely standard practice, you might as well use it.

4.6.2 Different Inclusion Methods

When it comes to including CSS stylesheets for your WordPress-powered site, there are plenty of options available to you. Let's take a look.

The not-so-dynamic method

```
<link rel="stylesheet" href="/wp-content/themes/Your-Theme/style.css" type="text/css"
media="screen, projection" />
```

The super-direct method

```
<link rel="stylesheet" href="<?php bloginfo('stylesheet_url'); ?>" type="text/css"
media="screen, projection" />
```

The dynamic method

```
<link rel="stylesheet" href="<?php bloginfo('stylesheet_directory'); ?>/style.css" type="text/
css" media="screen, projection" />
```

IE-specific stylesheets

```
<!--[if IE 6]>
    <link rel="stylesheet" type="text/css" href= "<?php bloginfo('stylesheet_directory'); ?>/
css/style-ie6.css" />
    <script type='text/javascript' src='<?php bloginfo('template_url'); ?>/js/DD_
belatedPNG_0.0.8a-min.js'></script>
<![endif]-->
```

DD_belatedPNG

That example JavaScript file right over there is actually a really useful script for getting alpha-transparent PNG files to work in IE 6.

<http://digwp.com/u/412>

Linking to multiple CSS files

```
<link rel="stylesheet" href="<?php bloginfo('stylesheet_directory'); ?>/style.css" type="text/
css" media="screen, projection" />
<link rel="stylesheet" href="<?php bloginfo('stylesheet_directory'); ?>/forms.css" type="text/
css" media="screen, projection" />
```

@import

One somewhat common technique is to use the dynamic method to call your theme's `style.css` file at the root of the theme, but then have that file be empty save for a few lines that load in additional stylesheets. This is perfectly acceptable, but do note that this method is just as taxing (or more) than loading multiple stylesheets directly in the `<head>`. The cleanliness of this though, is appealing.

```
@import "style/css/reset.css";  
@import "style/css/typography.css";  
@import "style/css/layout.css";
```

Conditionally loading extra CSS files

```
<?php if (is_page_template("page-snippet.php")){ ?>  
    <link rel="stylesheet" type="text/css"  
        href="<?php bloginfo('stylesheet_directory'); ?>/css/snippets.css" />  
<?php } ?>
```

4.6.3 To Reset or Not To Reset?

While we are on the subject of CSS, but not on the subject of actually writing any particular theme, we may as well talk about “resetting” stylesheets. The theory of a reset stylesheet is that it removes the various default styling applied to elements by browsers. These default CSS styles can be a bit unpredictable.

For example, the amount of default padding applied to the `<body>` element is different in Firefox than it is in Internet Explorer. Likewise for other browsers – they each apply their own set of CSS rules to various elements. In order to rein in these differences, we use **reset stylesheets** to eliminate certain default styles and start fresh. If we want some padding on our `<body>`, we can then apply it according to our specific needs and have that value be consistent across all browsers.

Popular CSS Resets

- **Eric Meyer's Reset Reloaded** – From the man himself: <http://digwp.com/u/142>
- **YUI (Yahoo User Interface) Reset CSS** – <http://digwp.com/u/143>
- **Star Selector Reset** – Looks like this: `* { margin: 0; padding: 0; }`

Reset stylesheets are not specific to WordPress, but rather a common consideration for all websites. But since we are WordPress folks, this is how we might include a reset stylesheet in our theme:

```
<link rel="stylesheet" href="<?php bloginfo('stylesheet_directory'); ?>/
reset.css" type="text/css" media="screen, projection" />

<link rel="stylesheet" href="<?php bloginfo('stylesheet_directory'); ?>/
style.css" type="text/css" media="screen, projection" />
```

Notice we included it *before* the main `style.css` file. Because we ultimately reset the reset-styles with our own values, the reset will need to go first. Critics of reset stylesheets will say that, because we ultimately reset the reset, it's unnecessary. Proponents say that the reset is there to catch things we normally wouldn't write into our own stylesheet but may pop up in the lifespan of the site.

4.6.4 Basic CSS Optimization

Now that we've just shown you how you might include a reset stylesheet in your theme, we'll tell you that might not be the best way to do it. **Ha!** You really gotta watch us!

The problem with including a reset stylesheet isn't the reset stylesheet itself, but rather because it's a separate file, meaning *yet another* thing the browser needs to download. In geek speak, it's another page resource, or another HTTP request. The more HTTP requests, the slower your page will download and render.

If speed is of the utmost concern, you may want to include the reset at the top

CSS Reset Library

There are many, many different pre-made CSS resets available to you. For a fairly comprehensive collection of some of the best, check out:

<http://digwp.com/u/406>

Online CSS Reset Tool

CSSresetr.com is an online tool and CSS Reset Library that enables you to easily apply any reset to any type of content. You can even upload your own HTML templates.

<http://digwp.com/u/509>

of your CSS file instead of as a separate file. In this case, that makes perfect sense because the reset will be loaded on all pages of the site. But what about CSS files that you load conditionally? For example, perhaps you have a unique homepage with its own unique set of CSS rules. Loading a `homepage.css` file in that circumstance makes perfect sense, because otherwise every other page on the site will be loading that CSS file despite it not needing it. So even though it's a separate HTTP request, it probably makes the most sense.

WordPress Optimization

We explain how to minimize the number of HTTP requests for your site in more detail, and explore many other optimization techniques in chapter 9.5.1.

Reducing the number of CSS files used isn't the only way to optimize CSS though, you might also employ techniques to reduce the file size of the CSS itself. When we write CSS, we write it so it is best readable for us. That means spacing declarations, tabbing new lines, and including comments such that reading and maintaining the file works best for us. But all of the spacing, tabbing, new lines, and comments adds extra weight to the CSS file, and thus produces a larger file for visitors to download.

Nicely formatted CSS is for *you*, not for the browser, and not for your visitors. So there is no particular need to serve up that nicely formatted version. You may want to use a tool like the **CSS Optimiser** – <http://digwp.com/u/144> – to strip all of that extra stuff away from the CSS file and keep the file size as small as possible. Of course, as soon as you do this, the file becomes nearly impossible to maintain, so be sure to keep an always-up-to-date copy of the original stylesheet. Then make changes to the original, optimize it, and serve up the optimized version live.

Here is a good example of what an optimized stylesheet might look like:

```
#mainNav{position:absolute;bottom:8px;left:0;width:780px;list-style:none;z-index:100}#mainNav
li{display:inline}#mainNav li a{-webkit-transition:opacity .15s ease-in-out;display:block
;float:left;width:144px;height:50px;background:url(images/nav.png) no-repeat;text-indent:-
9999px;margin:0 10px 0 0;opacity:0.5;filter:alpha(opacity=50)}#mainNav li.forums a{background:-
144px 0}#mainNav li.videos a{background:-288px 0}#mainNav li.downloads a{background:-432px
0}#mainNav li.snippets a{background:-576px 0}
```

Not very readable eh? But that chunk above had a 43% file-size savings compared to the original CSS from whence it came.

4.7.1 Using Multiple Themes

Of course, only one theme can be displayed to any one person at a time. But that doesn't mean you can't serve up different themes to different people. Why would you want to do such a thing?

- Give users the choice of themes
- Serve a different theme to yourself, admins, or other registered users
- Have users arriving in different circumstances (e.g. mobile devices) see a different theme

How we accomplish this witchcraft depends on which of these scenarios we are trying to accommodate. Let's look at each one...

Giving Users Choice

It might be a fun feature to allow your users to choose a custom theme for viewing your site. This is probably not practical for a large number of sites, but if done with class it can be very cool! For example, Perishable Press offers theme switching right in the sidebar (or footer, depending on theme). At **DigWP.com**, we have a "Theme Clubhouse" – <http://digwp.com/u/407> – where we show off our free and exclusive themes using a single installation of WordPress.

This technique is made possible through the excellent **Theme Switcher** plugin, available at the *WordPress Plugin Directory*: <http://digwp.com/u/408>

For an extensive example of live theme-switching, check out **Perishable Press**, where Jeff offers visitors the choice of more than 20 different themes: <http://digwp.com/u/510>

Important: Make sure that you incorporate a way for visitors who have just switched themes a way to switch back to the default theme. Also interesting are the possibilities afforded by combining theme-switching with child themes.

Child Themes?

Tip: Refer back to chapter 4.5.1 for more information on child themes.

Different theme for administrators

The most common use for the theme-switching technique is theme development. As any good designer knows, to do the best work possible you need to design against real content, not a bunch of *lorem-ipsum* filler. If you could go commando and develop against a live theme, but only *you* saw that theme while everyone else was seeing the *old* theme, that would be an *ideal* way to develop. That is exactly what is possible with the **Theme Switch** plugin – <http://digwp.com/u/409> (not to be confused with the aforementioned Theme Switcher plugin – yeeesh).

Upload and activate this plugin, and you'll have an options panel for selecting the theme to display based on the user's role. For example, you could show only admin-level users a particular theme. This goes for all user roles, so you could theoretically show a different theme to all registered users. Or perhaps if you are working on a new design, open that up beyond admins to show your author-level users the design as well.

Different theme for mobile viewers

Another reason you might want to serve up an alternate theme is based on the circumstances by which the user has arrived at your site. A good example of this is delivering a special theme for users arriving via mobile device.

As with just about everything else you need to do with WordPress, there are plugins *specifically* designed to accommodate your site's mobile visitors. The **WPtouch** plugin – <http://digwp.com/u/410> – detects for high-capability mobile devices (e.g., the iPhone, iPod Touch, Android, and Blackberry) and serves up a very nice and fully functional theme for those users.

If you would rather roll your own theme, one possibility is to do your own mobile device sniffing and redirect those users to another URL. Here is a JavaScript way to go about it that you could put in your site's <head>. Just make sure that the script is not also located on the URL to which it redirects, otherwise your visitors will be caught in an infinite loop!

Mobify <http://mobify.me>

Another option for creating a special theme for mobile viewers is a free service called Mobify. Mobify doesn't actually create WordPress themes, but the end result is essentially the same. They process the content from your site, and then apply new CSS styles (created and controlled by you) to create the mobile view. This means that the mobile view is just

as dynamic as the regular site. Mobify will serve this content from the Mobify domain, or just as easily, through a subdomain located on your own site (e.g., m.alistapart.com) with a simple CNAME change on your domain registry.



```
<script type="text/javascript">
    if (screen.width <= 699) { document.location = "mobile.html"; }
</script>
```

This technique could be combined with the Theme Switcher plugin (see previous section on *Giving Users Choice*), as that plugin allows for special URLs which can switch the theme. The possibilities are literally endless!

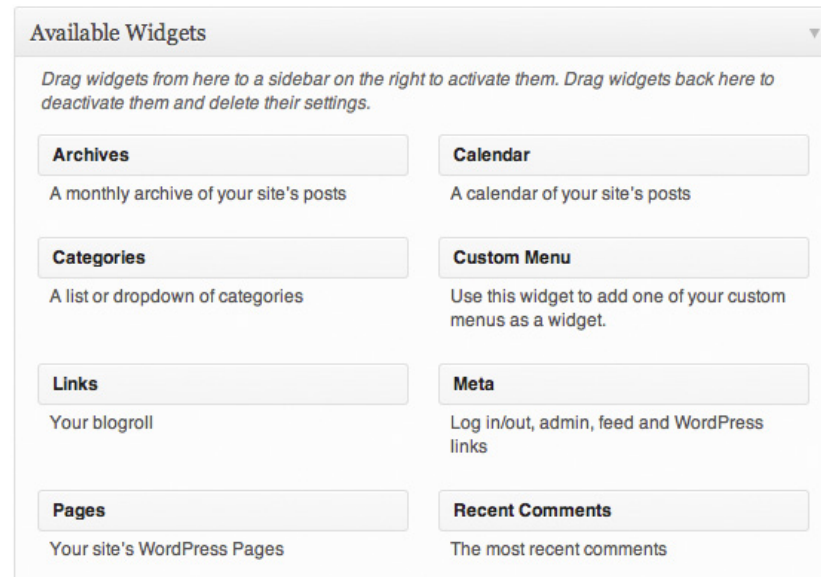
Responsive Design

...is everywhere, including the revamped WordPress Admin UI. As mentioned in chapter 3.5.6 of the supplemental book, version 3.3 brings better responsiveness for smaller screens like the iPad.

4.8.1 Widgetizing

Many themes you'll find around abouts the interweb are already "widgetized." That is, they contain the code necessary to let the WordPress Admin know about areas that are ready to accept widgets and be managed by the **Appearance > Widgets** settings area in the Admin.

Theme authors who intend their work to be used by a wide audience almost always include widgetized areas, but it's not a guarantee. If you are building your own theme, it's totally up to you if you wish to include this.



As we have seen earlier in this chapter, widgets can do some pretty cool stuff so at least you should probably consider it. You can have static stuff in your sidebar along with a widgetized area too, so the commitment here is pretty low. Here's how to do it:

Step 1: Declare your widgetized area

The most common place for widgets is in the sidebar. So in the `sidebar.php` file:

```
<!-- Static content could go above widgetized area. -->

<?php if (!function_exists('dynamic_sidebar') ||
        !dynamic_sidebar('Sidebar Widgets')) :

?>
    Content in here will ONLY show if there are no widgets active,
    or if the version of WordPress running doesn't support widgets.
    Otherwise, this area is where the widgets go, set via the Admin

<?php endif; ?>

<!-- Static content could go below widgetized area. -->
```

Step 2: Activate your widgetized area

In the `functions.php` file, use this function to activate the widgetized area:

```
// activate widget area
if (function_exists('register_sidebar')) {
    register_sidebar(array(
        'name'=>'Sidebar Widgets',
        'before_widget' => '<div id="%1$s" class="widget %2$s">',
```

```

        'after_widget' => '</div>',
        'before_title' => '<h4 class="widgettitle">',
        'after_title' => '</h4>',
    ));
}

```

You'll notice in this code that you reference the widgetized area by name and then give it some HTML parameters for before/after the widget and before/after the title. This gives you the chance to mimic your theme's setup and gives you the hooks you need to style the widgets as needed. In the HTML, notice that each widget also contains a class and ID value, which gives you the opportunity to style all the widgets as a group or target individual widgets for styling.

Step 3: Managing your widgets

There are all sorts of potentially useful widgets built right into WordPress. Simply drag them from the "Available Widgets" section over into the widget area to the right. The area should be showing and properly labeled now. Once dragged over, widgets typically have options that you can set, then save. Note that many plugins operate by adding to this list of available widgets.

If the beauty of widgets still eludes you, consider the control it offers someone managing a site who is somewhat comfortable with WordPress but not at all comfortable with editing code. A widgetized area allows them to rearrange things, edit text, change titles, change the number of things shown, and more, directly through the Admin itself, without having to change one piece of code.

Coming up...

With a good understanding of the WordPress theme system under our belts, we forge ahead and dig into the many ways that WordPress can be extended...

Any sufficiently advanced technology is
indistinguishable from magic.

– **ARTHUR C. CLARKE**

5

Extending the Functionality of WordPress

5.1.1 Extensibility

Out of the box, WordPress is a powerful platform for dynamic websites that may be used to build a wide variety of projects. Especially for blog-style sites, WordPress provides everything you need to establish yourself on the Web with a solid, fully functional blog featuring everything from a beautiful, easy-to-use Admin interface to a robust posting and commenting system that makes the process of posting content and facilitating discussion a real breeze. Even so, one of the reasons why WordPress has achieved its enormous popularity involves its vast *extensibility*.

5.1.2 Extending WordPress with Plugins

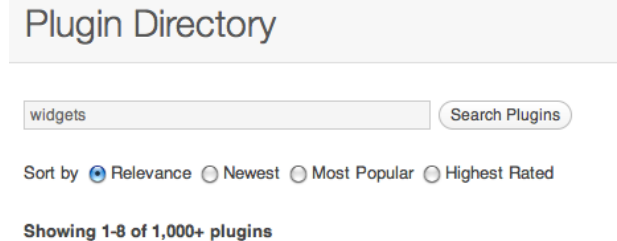
From themes and plugins to custom functions, scripts and even core hacks, the degree to which WordPress enables users to customize its implementation is astounding. In fact, with a few choice plugins and some of WordPress' built-in features, it is possible to transform WordPress from a powerful blogging engine into a highly customized Content Management System (CMS). In this chapter we examine some of best techniques and important aspects for extending the functionality of WordPress.

5.1.3 A Plugin for (Almost) Everything

The easiest way to integrate additional functionality into WordPress involves taking advantage of the thousands of plugins available at the WordPress Plugins Directory

and around the Web. Even when you exclude the many plugins not included in the official directory, there are still more than 21,400 plugins available at the time of this writing (more than twice the number since the last update of this book). With that many plugins available to you, the chances that you will find one that suits your specific needs is very good indeed. Here are some of the more popular categories of plugins featured at the Plugin Directory (as of WP 3.1):

- **widgets** – 3,211 plugins
- **Post & posts** – 2,116 plugins
- **administration (admin)** – 1,623 plugins
- **sidebar** – 1,387 plugins
- **comments** – 1,109 plugins
- **images** – 1,051 plugins
- **links** – 845 plugins



And many, many more. Out of the box, WordPress includes two plugins, one incredibly useful and the other relatively useless. The first of these default plugins is Akismet, which is one of the best anti-spam plugins currently available for WordPress. Akismet protects your comments by comparing them against their constantly evolving database. Any comments that look like spam are blocked and sent quietly to the spam bin. Users can then review the comments flagged as spam or else let Akismet delete them automatically after 15 days.

Unfortunately, the other plugin that is included with WordPress, Hello Dolly, is not nearly as useful. This plugin does nothing more than display a random lyric from Jerry Herman's song, "Hello Dolly," in the upper-right hand corner of the WordPress Admin screen. Hello Dolly was the very first plugin made for WordPress, and will probably be included with WordPress until the end of time. Unless you're really sentimental or happen to love Louis Armstrong, this plugin is essentially worthless and should be deleted as part of your configuration and set-up routine.

Of the thousands of plugins available for WordPress, there are a handful of plugins

Goodbye Dolly?

In a poll at the DiW site, WordPress users voted whether or not the Hello Dolly plugin should be included with WordPress. The results? Of over 1200 people who took the poll, more than 1000 of them (78%) voted that the plugin should not be included with WordPress.

<http://digwp.com/u/419>

that are installed on a large majority of sites and are considered by many to be “absolutely essential” for any WordPress installation. These plugins greatly increase the power of WordPress in several important areas.

As you set off to load up on plugins, keep in mind that around 99% of WordPress plugins are created by independent, third-party developers and are not always tested or optimized for maximum performance. Many are knock-offs, spin-offs, or just not needed because they provide functionality that is easily achieved by simpler methods. In fact, chances are high that you may not even *need* a plugin to achieve your goals. So before reaching for that easy plugin fix, ask yourself if installing *another* plugin is indeed the best solution.

5.1.4 Do You *Need* a Plugin?

So you want to add some awesome new feature to your WordPress site. Before installing another plugin, ask yourself if there is a simpler way to do it. The beauty of WordPress is that there are so many ways of doing things, especially when it comes to customizing your theme design. For example, to display the date and time of the most recent post modification, you *could* install a plugin to do the job, or you could simply add this slice of code to the desired location in your theme:

```
<p>Updated on <?php $x = get_the_time('U'); $m = get_the_modified_time('U');  
if ($m != $x) { the_modified_time('F d, Y'); } ?></p>
```

When it comes to enhancing your theme with stuff like category icons, random images, and popular posts, there is seldom need to install another plugin to make it happen. Before installing that next plugin, do a quick search for a script, widget, or custom function that will do the job. Taking a few seconds to check the alternatives may save you from having to maintain another unnecessary plugin.

The idea here is to keep the number of plugins to a minimum. By doing so, you eliminate extraneous script processing and help to ensure optimal site performance. Minimizing the number of plugins used for your site also improves the likelihood of smooth upgrades. The fewer third-party plugins you have

Essential Plugins

See chapter 2.7.4 for our list of essential plugins for just about any WordPress site.

Post-Modified Date

What does this code snippet do? It displays the date/time that the post was last modified. Just place into the loop (perhaps along with other metadata) and the “post-modified” date and time will be displayed. For more code snippets that replace plugins, check out this article at Perishable Press:

<http://digwp.com/u/415>

running, the less opportunity there is for something to go wrong while upgrading to the latest version of WordPress. Likewise with plugin updates: the chances of conflicts decrease with the number of plugins installed. It's all about facilitating upgrades, avoiding conflicts, and fostering maintainability.

Even so, there are many situations where plugins provide the perfect solution:

- Complex scripting needs or when a suitable alternative is not readily available
- Functionality that affects the core functionality of WordPress, such as caching
- Extensive functional enhancement, such as for database backups & sitemaps

Useful Plugins for Theme Developers

Here are three plugins that we have found to be virtually indispensable for theme development:

Custom Query String Reloaded

<http://digwp.com/u/511>

Specify the number of posts to display for different page views. Show 10 posts on your home page, 20 posts for your archive pages, and 50 posts for your search results. Without CQS, all of these page view simply display the same number of posts (as specified in the Admin area under **Settings > Reading**).

Condition		Show
_search	20	posts
SS	5	posts

The Excerpt Reloaded

<http://digwp.com/u/131>

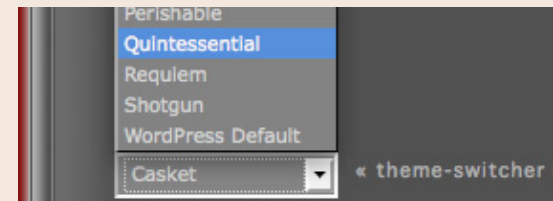
the_excerpt_reloaded enables you to completely customize the excerpts that are displayed on your site. Provides control over excerpt size, type, format, allowed markup elements, and much more. This level of control is perfect for displaying excerpts in multiple or customized loops. See **chapter 4** for more information about customizing multiple loops.

```
<div class="sides"><a href="<?php the_time('F d, Y @ g:i A')>"><?php the_excerpt_reloaded(77, 'v class="excerpt">
```

Theme Switcher

<http://digwp.com/u/132>

Perhaps the most underrated plugin of them all, Theme Switcher enables multiple themes on your site. This enables users to choose alternate themes, but it is also useful for developing themes behind the scenes. When developing a new theme, no need for fancy redirects or “under construction” messages. Simply install the plugin and get to work.



- Functionality that requires significant configuration of options and settings
- Enhancing administrative functionality with modified or additional admin areas
- You are uncomfortable editing code and would prefer doing things the easy way

When it comes to these types of scenarios, choosing a plugin is most likely your best bet. If so, do your research, find the best plugin for your needs, and give it a go. As mentioned previously, there are thousands of free, easy-to-use plugins available for immediate use. So regardless of your goals, there should be a plugin that suits your needs perfectly.

5.1.5 Choosing the Perfect Plugin

The key to finding the perfect plugin is research. Taking the time to investigate potential plugins ensures smooth implementation and prevents unnecessary headaches in the future. Of course, many of the more popular plugins (such as Akismet, XML Sitemaps, Database Manager) may be on your list before you even install WordPress, but even so, you may be surprised at the growing number of alternatives. Here are some tips to help you choose the perfect plugin:

- 1. Determine the need.** What do you want the plugin to do for you?
- 2. Check the WordPress Codex.** See if there are any suitable matches at the WP Plugin Directory. That should be your first stop for plugin shopping.
- 3. Regardless of what you find** at the Codex, search the Web for alternatives. Many developers deliberately choose *not* to list their plugins at the Codex. There are some great plugins that are only available directly from the author.
- 4. Check the compatibility** of the plugin before installation. Make sure that it works with your version of WordPress.
- 5. Check the support** of the plugin. Is there a way of getting help if you need it? Is there a forum? Does the author appear to be responsive?

- 6. Determine how frequently the plugin is updated.** This may provide clues as to the level of commitment that may be expected from the plugin author. There is nothing worse than relying on a plugin that fails to evolve with new versions of WordPress.
- 7. Is the plugin well-documented?** Does the documentation explain everything adequately? Is there a change log or history of changes?
- 8. Search the Web for specific issues** related to the plugin. Dig for the stuff that isn't mentioned on the plugin page. Search for phrases such as "problem with name-of-plugin," "name-of-plugin issues," or even "name-of-plugin sucks." Also research the performance of the plugin so it doesn't slow you down.
- 9. Check the files.** Once you have found that perfect plugin, check its files carefully. Look for anything that seems out of place. Are there extraneous files? Is documentation included? Examples?
- 10. Check the code.** If you understand PHP, (X)HTML, CSS, and/or JavaScript, take a good look at the code and do your best to see if everything is legit. Keep an eye open for anything that cries foul, such as spam links, unreasonable licenses, and so on.

While you may not need to perform all of these steps for every plugin you use, keeping these things in mind will help you to choose a perfect collection of plugins for your site. Again, the key to maximizing your experience with a plugin involves taking the time to research and understand its purpose and functionality.

5.2.1 Plugin Usage and Maintenance

Contrary to popular belief, plugins are not simply set-it-and-forget fixes for WordPress. As convenient and easy as many plugins happen to be, most still require initial configuration, periodic maintenance, regular updates, and occasional troubleshooting and tweaking. In this section, we examine some helpful tips for optimizing plugin usage and maintenance.

5.2.2 Sequential Installation

One of the best pieces of advice that we can give as you begin adding new plugins to your WordPress site is to do so *sequentially*, one at a time. Installing your plugins one at a time gives you the opportunity to test your site for proper functionality. This enables you to know immediately if the plugin is compatible with your site or not. Installing 20 plugins all at once only to discover afterwards that your site is broken requires you to go back through each plugin, one-by-one, to determine the source of the issue. Thus, you will save time, stress, and a big headache by meticulously installing your plugins in an organized, sequential fashion.

5.2.3 Keep Plugins Up-To-Date

One of the best ways to ensure a smooth experience with your plugins is to keep them up-to-date. The easiest way to stay current is to keep an eye on your WordPress Admin area. Whenever an update for a plugin (or for WordPress itself) is available, you will see a notice displayed on your various Admin pages. Once you see that an update is available, there are essentially two ways to upgrade:

Automatic upgrades

Since WordPress 2.5, site administrators may upgrade their plugins automatically by clicking on the plugin's upgrade link on the Plugin Admin page. WordPress 3.0 and beyond even has *bulk updating* to do everything at once. No more excuses for not keeping your plugins up-to-date.

Manual upgrades

Manually upgrading your plugins provides a much greater degree of control over the entire process than possible with automatic methods. Maintaining control over the upgrade process is important for plugins that you may have customized, and also for expedient diagnosis and resolution of any issues that may arise.

Shady plugins

From time to time reports surface of plugins that contain malicious code. Most plugins are completely fine, but always keep an eye out for suspicious-looking code. If you are not sufficiently familiar with code to know the bad stuff when you see it, take some time to research the plugin you are about install. 99% of the time you're cool, but it's better safe than sorry.

New versions of plugins often include new files, options settings, or changes to existing settings that may interfere with normal site performance. Further, plugin updates sometimes require complete un-installation of previous versions or even additional steps in order for proper installation to occur. Manually upgrading your plugins eliminates potential problems by giving you full control.

5.2.4 Subscribe to Plugin Comment Threads

A great way to stay current with news relating to your plugins is to subscribe to any relevant plugin feeds. Good candidates include comment and forum threads, plugin-specific post feeds, and feeds from sites that primarily cover WordPress plugins. You may want to create a folder in your feed aggregator called “My WordPress Plugins” and review the results periodically.

5.2.5 Getting Help with Plugins

As you embark on the process of installing and configuring plugins, it is inevitable that you will encounter issues and conflicts that may be beyond your expertise to resolve. Should this happen, the first place to go for help is to the plugin’s developer or author. Most often, developers will leave a comment thread open for their plugin pages, or else provide some sort of an official forum for handling plugin-related issues.

If the problem is not resolved using these methods, don’t hesitate to contact the plugin author directly. They created the plugin and thus should be more than happy to help people with its implementation and use.

Other places to go for help include the WordPress.org site, where you may register and ask WordPress-related questions in the forum. If you take this route, make darn sure you have searched the Web and the WordPress forum as carefully as possible. Many of the forum moderators have little patience for users who don’t bother doing their homework *before* posting a question.

Private Party

Automatic upgrades works only for plugins hosted at the official Plugin Directory. Another good reason to list your plugin there.

It’s free for a reason

It is important to remember that free help is just that – free. Many WP newbies make the mistake of expecting immediate and perfect solutions when asking for help. This just isn’t the way it works. WordPress heads enjoy helping people learn, but make sure you take the time to research the issue to the best of your ability before asking for assistance. Be clear, polite, and remember to show appreciation to those who take the time to help you. It’s all about karma ;)

5.2.6 Diagnosing Plugin Conflicts

Diagnosing plugin conflicts is often easier than it may seem. Assuming that you are installing your plugins *sequentially* (see above), problems that arise upon installing or updating a plugin are easily spotted. Once you know which plugin is causing the issue, you are in a better position to seek and find a solution. Many times, a plugin that fails to work or causes errors is incompatible with another plugin. To determine if this is the case, leave the new/upgraded plugin activated and sequentially disable each of your plugins. After each plugin has been disabled, check for resolution of the issue in question and continue the process until it is resolved. If, after disabling all other plugins, your new/upgraded plugin still is associated with issues, it may be incompatible with WordPress itself.

5.2.7 Disabling and Uninstalling Plugins

As you work with plugins, keep in mind that there is a difference between *disabling* a plugin and *uninstalling* it. In general, disabling a plugin means that the plugin is inactive yet still present in the plugins directory. More importantly, any settings for disabled plugins are still present in the database. On the other hand, when a plugin is uninstalled, it is no longer present in the plugins directory and any related database settings have likely been deleted.

It is important to keep in mind that many plugins add information to your WordPress database. Upon initial activation, plugins may modify or add information to various tables, most typically the “options” table. New database tables may be added and populated with data as well. Thus, as you go about trying out new plugins, it is important to be aware of any changes made to your database. A well-designed plugin will provide an uninstall feature that will clean-up after itself and remove all traces of its settings from the database.

There may also be situations where you need to quickly disable one or more of your plugins due to a conflict, troubleshooting, and so forth. While disabling plugins is usually handled from within the Admin area, there may be situations where this is not possible. There are several ways to disable plugins, but the quickest and easiest method is simply to rename either the plugin (to disable

Know thy files

If you are automatically updating your plugins and/or core files through the Admin (or any other method), it is wise to remember that the files on the server will be newer than the ones on your local machine. This may sound obvious, but much confusion and many errors may be avoided by not overwriting updated files with older ones. A good way to prevent this is to either use some sort of a version control system (such as Subversion), or else play it safe and go with the manual-update method.

individual plugins) or the entire wp-content/plugins directory (to disable all plugins). Renaming the plugin folder to, say, "plugins_inactive", will effectively disable (not uninstall) all of your plugins. Once you are ready to reactivate any or all of your plugins, simply rename the directory back to "plugins" and you are good to go. All of the options will be preserved, but you will need to reactivate each plugin manually.

phpMyAdmin for WP

WP-phpMyAdmin is a plugin that enables you to work with phpMyAdmin from the comfort of the WordPress Admin.

<http://digwp.com/u/512>

If you enjoy access to a database interface application such as the excellent phpMyAdmin, there are many ways to interact with and modify the database directly by simply executing various SQL commands. For example, after making a backup of your database, execution of the following SQL command will enable you to easily disable any or all of your plugins:

```
SELECT * FROM wp_options WHERE option_name = 'active_plugins';
```

Change your DB Prefix

See chapter 9.1.7 for ways to secure your database. Changing the default database prefix is one way to help prevent common attacks.

Note that you may need to edit the default WordPress table prefix, "wp_", if you are using something different.

Once the active_plugins column appears, click to edit it. You will see something similar to the following, depending on the number and type of plugins you have installed:

```
a:31:{i:0;s:13:"AddMySite.php";i:1;s:19:"akismet/akismet.php";i:2;s:23:"all_in_one_seo_pack.php";i:3;s:16:"authenticate.php";i:4;s:28:"breadcrumb-navigation-xt.php";i:5;s:18:"codeautoescape.php";i:6;s:37:"contact-coldform/contact_coldform.php";i:7;s:32:"custom-query-string-reloaded.php";i:8;s:30:"customizable-post-listings.php";i:9;s:33:"dd-sitemap-gen/dd-sitemap-gen.php";i:10;s:20:"download-counter.php";i:11;s:13:"feedcount.php";i:13;s:15:"get-weather.php";i:14;s:36:"google-sitemap-generator/sitemap.php";i:15;s:13:"gravatars.php";i:16;s:19:"kill-admin-nags.php";i:17;s:18:"landingsites13.php";i:18;s:30:"nofollow-free/nofollowfree.php";i:19;s:17:"ol_feedburner.php";i:20;s:16:"plugins-used.php";i:21;s:22:"popularity-contest.php";i:22;s:39:"search-everything/search_everything.php";i:23;s:27:"simple-tags/simple-tags.php";i:24;s:26:"simple_recent_comments.php";i:25;s:18:"simple_twitter.php";i:26;s:25:"subscribe-to-comments.php";i:27;s:24:"the-excerpt-reloaded.php";i:28;s:18:"theme-switcher.php";i:29;s:9:"top10.php";i:30;s:16:"wp-db-backup.php";}
```

Remember to Backup

Reminder to make a current backup before making any changes to your database.

That entire array of code represents every active plugin on your site. Thus, to quickly disable all plugins without using the WP Admin area, highlight the entire block of code, cut it out, and paste it into a safe, offline text file. After removing

the code, click the button to save your changes and that's it. All WordPress plugins are now deactivated (yet still installed, and with all plugin options intact). This obviously is a huge time-saver that really comes in handy during those mission-critical, time-sensitive situations where every second counts.

Once you are ready to re-activate your entire set of plugins, simply cut/copy & paste the preserved code back into the "active_plugins" field. Click save and done. Again, don't forget to backup your database before making any changes.

Or, instead of disabling the entire collection, you may selectively disable any of your plugins by locating and removing its name from within the list. Here is the general pattern once you format the code a bit:

```
a:31:{
  i:0;s:13:"AddMySite.php";
  i:1;s:19:"akismet/akismet.php";
  i:2;s:23:"all_in_one_seo_pack.php";
  i:3;s:16:"authenticate.php";
  i:4;s:28:"breadcrumb-navigation-xt.php";
  .
  .
  .
}
```

So, to deactivate any plugin, simply remove its respective line from the list. Alternately, here is a simple SQL query to disable all active plugins:

```
UPDATE wp_options SET option_value = ''
WHERE option_name = 'active_plugins';
```

This query will clear the active_plugins field of all active plugins, effectively disabling (without uninstalling or modifying) the entire set. Perfect if you plan on re-enabling each plugin individually, say, after resolving some heinous server error. Whereas the previous technique makes it easy to re-enable all plugins en masse, this query is perfect for simply "nuking" all active plugins with no remorse.

Quickly Disable Plugins

In the blink of an eye, you can easily enable/disable all plugins via the database:

<http://digwp.com/u/513>

5.2.8 Share Your Experience with Others

Finally, as you become familiar with and begin using WordPress plugins, it is important to share your experience – either positive or negative – with the community. If the topic happens to fit in at your site, throw down a few posts describing any significant discoveries that you may have encountered with your plugins.

You should also share important information with the plugin authors themselves and, if the issue is not security related, perhaps even post the information on a relevant forum, such as the one provided at [WordPress.org](https://wordpress.org).

The idea here is to give back to the community by sharing your insights and experience in order to help users and developers better understand the plugins they are working with.

5.3.1 Extending WordPress with Custom Functions

Just because you *can*, doesn't mean you *should*. As discussed in the previous sections, there are many ways to extend the functionality of WordPress, especially if you are familiar with and comfortable working with a bit of code.

There are many situations where your development and design efforts are better served with WordPress' amazingly convenient theme-specific script functionality, also referred to as the theme's "functions.php" file.

The purpose of the `functions.php` file is to provide developers and designers a way to expand the functionality of WordPress on a per-theme basis. Each theme may use a `functions.php` file to include any number of PHP functions and scripts. This additional code is then processed when the theme is active, thereby extending the functionality of WordPress as it relates to that particular theme.

5.3.2 Plugins vs. Theme Functions (functions.php)

While both plugins and theme functions (i.e., scripts contained within the `functions.php` file) may be used to extend the functionality of WordPress, there are some key differences between the two:

- **Locality** - Plugins usually operate on a sitewide basis. Theme functions operate only when the theme is active
- **Updates** - Plugins are usually updated periodically. Most theme functions typically are *not* updated (depending on the theme – some themes are updated regularly and may include changes to the `functions.php` file).
- **Admin** - Plugins are always shown in the Admin Plugins page. Theme functions may or may not appear in the Admin Plugins page, depending on the function or script.

So when should you use a `functions.php` file instead of a plugin? While the answer depends on many factors, here are a few general guidelines:

- Theme-specific functions should be placed in a `functions.php` file
- Custom theme functionality should be placed in a `functions.php` file
- Smaller scripts and functions should be placed in a `functions.php` file
- Sitewide functions should be implemented as a plugin (especially when using multiple themes)
- Functionality requiring an Admin interface regardless of theme should be implemented as a plugin
- Fundamental changes in functionality should be implemented as a plugin

5.3.3 Examples of Useful Theme Functions

Way back in the days of WordPress 1.5, many developers had no idea that theme-specific functionality was even possible. Many functional moderations were made

FYI

Version: 1.5.0

Author: [Shaon](#)

Last Updated: 20 hours ago

Requires WordPress Version:
2.0.2 or higher

Compatible up to: 3.2.0

Downloaded: 15,200 times

[WordPress.org Plugin Page »](#)

[Plugin Homepage »](#)

The plugin directory allows you to see if the plugin version and your WordPress version are going to work together, according to reports by real users.

Stop Nagging! – How to Disable the Update Nag

In the Admin area, WordPress will remind you when plugin or core updates are available. The plugin reminders appear only on the Plugins page, but the update-WordPress reminders appear on every page. To stop the nagging, upload & activate this plugin:

```
<?php /*
Plugin Name: StopNag
Description: Stops the nagging.
Version: 1.0
Author: Jeff Starr */
add_action('admin_menu', create_function('$a', "remove_action('load-plugins.php', 'wp_update_plugins');"));
add_filter('pre_option_update_plugins', create_function('$a', "return null;"));
add_action('init', create_function('$a', "remove_action('init', 'wp_version_check');"));
add_filter('pre_option_update_core', create_function('$a', "return null;"));
?>
```

to the core of WordPress itself – something that you should *never* do, unless you positively, absolutely have no other choice. Most of the time, the functions.php file enables you to customize and make changes without modifying core files. Let's look at a few examples of how to use the functions.php file.

5.3.4 Example #1: Easy Admin Buttons for Comments

In addition to managing comments through the WordPress Comments Admin area, it is also helpful to have some easy admin buttons located next to the comments as they appear on your blog. Here is an example of what we're talking about:

Trackbacks / Pingbacks

1. [The Power of HTML 5 and CSS 3 • Perishable Press « Netcrema - creme de la social news via digg + delicious + stumbleupon + reddit](#) • [\[edit \]](#) [\[delete \]](#) [\[spam \]](#) *EASY FRONT-END ADMIN*
2. [Web fonts, HTML 5 roundup – Jeffrey Zeldman Presents The Daily Report](#) • [\[edit \]](#) [\[delete \]](#) [\[spam \]](#)
3. [popurls.com // popular today](#) • [\[edit \]](#) [\[delete \]](#) [\[spam \]](#)



Here is an easy way to add convenient “spam” and “delete” links next to each comment whenever you are logged in as Administrator. Having access to spam and delete buttons next to each comment makes it super-easy to clean up missed spam, scummy trackbacks, and other garbage. To add this functionality, we take advantage of the `functions.php` file by adding the following script:

```
// Display Spam & Delete links when logged in
function spam_delete_links($id) {
    global $id;
    if (current_user_can('edit_post')) {
        echo ' | <a href="'.get_bloginfo('wpurl').'/wp-admin/comment.php?
action=c&c='.$id.'">Delete</a>';
        echo ' | <a href="'.get_bloginfo('wpurl').'/wp-admin/comment.php?
action=c&dt=spam&c='.$id.'">Spam</a>';
    }
}
```

Once the function is in place, call it by adding the following code to the desired location in your `comments.php` file (for example next to each comment):

```
<?php spam_delete_links(get_comment_ID()); ?>
```

And that’s all there is to it! Your comments area should now feature quick and easy “spam” and “delete” links next to each individual comment.

Mind your `<?php?>`

In this example, we provide two chunks of code. The first does not include `<?php?>` opening/closing brackets, but the other example does. So what up? Your functions file is mostly PHP code and probably already has the opening/closing brackets. Conversely, theme template files usually contain lots of HTML markup with snippets of PHP mixed in. Feel free to add/remove the brackets as needed.

5.3.5 Example #2: Sitewide Shortcode Functionality

WordPress shortcodes are shortcuts for frequently used content such as links, images, and titles. For example, instead of writing out the entire link to your homepage, shortcodes enable you to simply write “[home]” wherever you want the link to appear. Shortcodes may be as advanced as needed to do just about any task.

Shortcode API

Shortcode functionality is made possible thanks to a set of WP functions collectively referred to as the Shortcode API (introduced in version 2.5). Learn more at the Codex:

<http://digwp.com/u/514>

As expected, the `functions.php` file is the easiest way to implement shortcodes. For this example, the following function will convert the `[home]` shortcode into a link to your site's homepage:

```
<?php // shortcode for homepage link
function myHomePage() {
    return '<a href="http://domain.tld/"
           title="My Website Homepage">My Homepage</a>';
}

add_shortcode('home', 'myHomePage');
?>
```

Place this function in your theme's `functions.php` file, and then call the function by using the shortcode. Simply write "`[home]`" anywhere in your blog post and the `myHomePage` function will display the link to your homepage. Better than magic.

5.3.6 Example #3: Moving Plugins to `functions.php`

The `functions.php` file is so awesome that you can even put *entire plugins* into it. This is a great way to isolate a plugin's functionality to a specific theme. By default, plugin functionality affects *all* themes, however when included via `functions.php` file, a plugin will only affect the theme to which it belongs. In our experience, the *easiest* plugins to transfer are those consisting of only one file. Other plugins may require additional maneuvers to work properly when relocated to `functions.php`.

5.3.7 Example #4: Creating Plugins from Functions

Moving a plugin to your `functions.php` file is as easy as creating a plugin from a custom function. For either of the previous example functions, we would simply copy & paste the entire function into an empty PHP file (named whatever).

Then, before uploading your file to the `/plugins/` directory, specify the required information at the beginning of the plugin file. Here is an example of a valid plugin header that you can use as a template for creating your own plugins:

```
<?php
/*
Plugin Name: My Awesome New Plugin
Plugin URI: http://example.com/plugins/awesome-new-plugin/
Description: Provides awesome new functionality for WordPress.
Version: 1.0
Author: The Awesome Plugin Maker
Author URI: http://example.com/
*/
// Place awesome function here
?>
```

Generally we prefer to keep simple functions in the `functions.php` file, but implementing them via plugin has benefits, such as working across all themes and enabling easy activation and deactivation via the Admin.

These examples are just the tip of the iceberg when it comes to the wide range of functionality that may be implemented via your theme's `functions.php` file.

Beyond plugins and theme functions, there are many other ways to extend the functionality of WordPress. Let's take a look.

5.4.1 Other Ways to Extend & Customize

So far we have explored the two most common ways of extending WordPress functionality, namely, plugins and the `functions.php` file. In addition to these methods, you may also implement custom functionality directly within your theme template files. This is a flexible, convenient way to customize your theme.

Easy Shortcode Permalinks

When you are building a theme, and the circumstance comes up where you need to create a link to a specific page hard-baked right into the theme, there is a function you should be using.

Not great

```
<a href="/contact/">Contact</a>
```

Much better

```
<a href="<?php echo get_permalink(12); ?>">Contact</a>
```

That “12” would be the ID of the Post or Page. Why is this better?

- If the slug ever changes, you are still cool.
- If the site moves from a sub directory (like if you were developing and then moving) to a top level domain or vice versa, you are still cool.

Doing it this way is a permanent reference to that Post or Page that will always be correct. This works great when we are working within our theme files, but what about when we are working within WordPress and actually writing Posts and Pages?

By default, we can’t run PHP within the content of our Posts and Pages*, so we can’t use the `get_permalink` function. What we can do, is create a shortcode with just about the same functionality.

** If you need to run PHP inside Post content, check out this plugin: <http://digwp.com/u/465>*

```
// easy shortcode permalinks
function permalink_thingy($atts) {
    extract(shortcode_atts(array(
        'id' => 1,
        'text' => "" // default value if none supplied
    ), $atts));

    if ($text) {
        $url = get_permalink($id);
        return "<a href='$url'>$text</a>";
    } else {
        return get_permalink($id);
    }
}
add_shortcode('permalink', 'permalink_thingy');
```

This shortcode can be used in two ways:

Basic

```
<a href="[permalink id=49]">Using without providing text</a>
```

Provide only the ID parameter and it only returns a URL. This way you can use that URL however you want. For example, if you needed to add a special class name to the link or something (but only occasionally).

Providing text

```
[permalink id=49 text='providing text']
```

This way returns a fully formatted anchor link back, using the text you pass.

5.4.2 Functions Within Theme Files

As discussed in the Themes chapter of this book, theme template files contain numerous template tags, PHP scripts, and (X)HTML markup. Within these files, designers and developers may place just about any type of custom functionality.

For example, the previous `functions.php` example for “Easy Admin Buttons” could be placed directly within the theme file instead. Within the comment loop, we could add the script as follows:

```
<p><?php comment_author_link(); ?><p>
<p><?php if (current_user_can('edit_post')) {
    echo '<a href="'.get_bloginfo('wpurl').'/wp-admin/comment.php
?action=cde&c='.comment_ID().'">Delete</a>';
    echo ' | <a href="'.get_bloginfo('wpurl').'/wp-admin/comment.php
?action=cde&dt=spam&c='.comment_ID().'">Spam</a>';
} ?></p>
<?php comment_text(); ?>
```

This code will output the following markup, which will vary depending on the actual content of your site:

```
<p><a href='http://domain.tld/' rel='external nofollow'>Author</a></p>
<p><a href="http://domain.tld/wp-admin/comment.php
?action=cde&c=123">Delete</a> | <a href="http://domain.tld/wp-admin/
comment.php?action=cde&dt=spam&c=123">Spam</a></p>
<p>Hello, thanks for this article, it is exactly what I needed!</p>
```

The key to placing functions directly in your theme template files involves changing the code from a function to a script that will execute upon page load. To illustrate, compare this implementation of Easy Admin Buttons with that in chapter 5.3.4.

This is just one example of a custom function that works well when integrated directly into a theme template file. Other good examples include just about any sort of custom loop functionality. Modifying loop functionality directly in theme files is often easier than trying to abstract functionality via plugin or `functions.php`.

5.4.3 Hacking the WordPress Core

Last and certainly *least*, we arrive at the “forbidden” method of extending WordPress, otherwise known as modifying default WordPress functionality, aka “hacking the WordPress core.” Hacking the WordPress core is nothing short of a heinous crime as measured by today’s standards, but it remains an effective way to achieve functionality not available through other, more legitimate methods.

While some WordPress heads will argue that you should absolutely, positively *never* hack a core WordPress file on *any* occasion or for *any* reason whatsoever and forever and ever and ever, the truth is that there are some cases where getting

Super-Easy Featured Images (Post-Thumbnails)

Surfing the Web, you will notice that many blogs include a featured image for each post. For example, at **DigWP.com**, we attach a thumbnail to every post to help improve the user-experience for visitors. Before WP 2.9, including post-thumbnails required the use of custom-fields, but now the functionality is *built-in*. To get started, enable featured images with this code added to your theme’s `functions.php` file:

```
<?php if (function_exists('add_theme_support')) { add_theme_support('post-thumbnails'); } ?>
```

With that code in place, go to write or edit a post as usual in the Admin, and click on “Set Thumbnail” in the “Post Thumbnail” panel. From there, select and tag your image from within the Media Library. Then, to display post thumbnails in your theme, add the following tag anywhere within the loop:

```
<?php the_post_thumbnail(); ?>
```

“Post-thumbnails” are now called “Featured Images,” but the basic functionality is the same. Check out chapter 2.2.6 in the supplemental book for more info.

the job done is more important than following ideals and best practices. When functional goals cannot be met with existing plugins and scripts, a solution may be found in the editing of the WordPress core.

“What!?” I hear some purists freaking out right now, screaming loudly and shaking their fists in opposition, “this is blasphemy against the WordPress gods!” But if you think about it, there really is nothing sacred about the WordPress core. The usual arguments against editing core files revolve mostly around the concept of maintainability. WordPress is continually updated with patches, fixes, and new versions. Thus, having a bunch of core edits to worry about reduces the ease with which these updates are applied to your site. Or so the argument goes.

Regardless of the debate, there may be situations where you have no choice but to hack a few lines of core code to do the job. So, rather than pretend that people don’t do it, here are some tips for when you find yourself doing the “evil” deed:

- Don’t hack the core unless you have absolutely no other option
- Do your research and understand *exactly* what you are doing
- Create a “read-me-before-updating” text file that details the changes

Bottom line: Before hacking the core, do your research, know what you are doing, and make absolutely certain that no other options exist. Also, take good notes and refer to them before every future WordPress upgrade. You never know, you may actually *learn* something new from digging around under the hood!

5.5.1 WordPress as a Content Management System (CMS)

As we have seen, the possibilities for extending WordPress functionality are virtually endless. What began as a humble blogging platform called b2/cafeblog way back in 2001 is now robust and flexible enough to serve as a highly customizable Content Management System (CMS).

Update: WordPress 3.0

Check out chapter 3 in the supplemental book to learn about all of the new CMS-related functionality that is now included with WordPress.

One of the main distinctions between a CMS and a simple blog involves the flexibility of the underlying software and its ability to create and manage multiple users and various types of digital media, which may include everything from text and images to audio and video content. With each new update, WordPress gets better at doing these things and much more.

Out of the box, WordPress provides many features that help any number of users publish and manage a wide variety of digital content, and with the addition of a few key plugins, transforming WordPress into a fully functional CMS is a breeze.

In this section, we'll first examine the CMS functionality that is built into WordPress, and then explore some key plugins that take WordPress' CMS capabilities to the next level.

5.5.2 Working With Custom Fields

Post Thumbnails

As of WordPress 2.9, post thumbnails are easier than ever. Instead of using custom fields, you can now display post thumbnails with a simple template tag. See the previous popout box, and also chapter 2.2.6 in the supplemental book for more information.

Perhaps the most powerful CMS-enabling feature included with WordPress is found in its robust custom-fields functionality. Custom fields enable users to associate additional custom content and information with each post. The result is massive flexibility in terms of content organization, design, and management.

To better understand how custom fields work, let's consider a scenario where you would like to associate thumbnail images with your posts. You *could* include the image information along with the post text.

But this setup would force you to display the thumbnail along with the post, in the exact order that it appears in the post content. If you wanted to display the thumbnail outside of the post, say in the sidebar, how would you go about it? Easy. Custom Fields to the rescue.

By setting a custom field key named "thumbnail" with a corresponding value of "http://domain.tld/images/thumbnail-01.jpg", we have associated the thumbnail with our post in such a way that will enable us to display it anywhere on the page.

With the custom fields added to our posts, we are now ready to tap into some core WordPress functionality and display our custom post-images in the desired fashion. The function we will be using to retrieve the custom images is `get_post_meta`, which is a native WordPress function designed to retrieve specific custom-field key values. The `get_post_meta` function takes the following three parameters:

\$post_id – defines the post associated with custom-field data

\$key – defines the key of the desired custom-field meta value

\$single – specifies whether to return the data as an array or string

Plugged into `get_post_meta`, these parameters look like this:

```
<?php get_post_meta($post_id, '$key', $single); ?>
```

To specify the ID parameter, `$post_id`, for each post, we use `"$post->ID"`, which requires the function to be placed within the loop. For the `$key` parameter, we will use the name of the target key, which in this case is `"thumbnail"`. And finally, because we want the key value returned as a string, we use `"true"` for the `$single` parameter. At this point our `get_post_meta` function looks like this:

```
<?php get_post_meta($post->ID, 'thumbnail', true); ?>
```

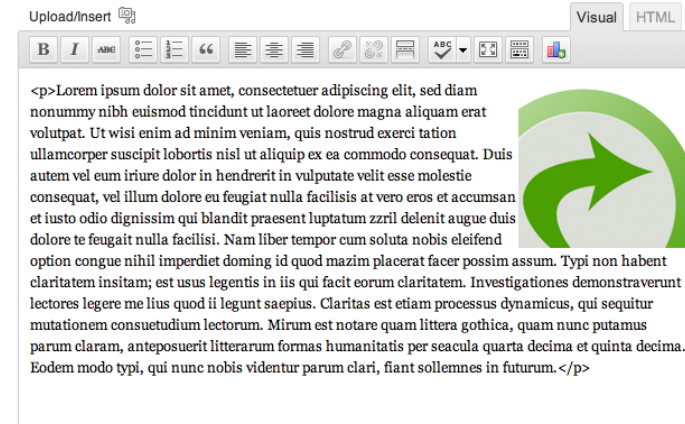
And we are almost there. As is, this code will simply return the custom-field value without printing it to the web page. So, we make one final edit to `"echo"` the data to the web page:

```
<?php echo get_post_meta($post->ID, 'thumbnail', true); ?>
```

Used anywhere in the loop, this tag will output each post's `"thumbnail"` custom-

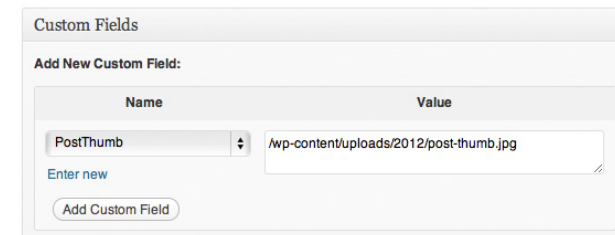
Bad

Items such as post thumbnails don't belong in post content.



Good

Using custom fields for stuff like post thumbnails makes it easy to display the information apart from the main post content.



Add some style

If you're trying this demo and your post thumbnail isn't appearing quite exactly as it is in the screenshot, try adding a few CSS styles to your theme's style.css file:

```
.postThumb {  
    float: left; clear: none;  
    margin-right: 10px;  
}
```

Screenshot showing post thumbnails moved to the sidebar via second loop.

Recent Posts



field value, which at this point is simply the URL of the thumbnail image. By enclosing the `get_post_meta` tag with a little markup, we can easily transform each URL into an actual thumbnail image that links to its corresponding post:

```
<a class="postThumb" href="<?php the_permalink() ?>">  
    " />  
</a>
```

With that code placed in the loop somewhere next to the `the_content()`, each post will display a thumbnail image that links to the single view of the post itself. See screenshot for example.

Thumbnails are just one example of how post data may be stored in custom fields and then displayed virtually anywhere in your theme.

To demonstrate this flexibility, let's now move our thumbnails to the sidebar. Using custom fields for stuff like images and media content makes doing stuff like this easily possible.

The trick here is knowing that `get_post_meta()` requires the loop in order to work. To display our thumbnails in the sidebar, we need to create a *second loop* in our sidebar.php file.

As seen in chapter 4.1.4, `get_posts()` is the perfect candidate for adding extra loops anywhere in your theme. So, after including our post-thumbnail snippet from above, our sidebar loop looks like this:

Catching a Train



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec odio. Quisque volutpat mattis eros. Nullam malesuada erat ut turpis. Suspendisse urna nibh, viverra non, semper suscipit, posuere a, pede. Donec nec justo eget felis.

Tripping in Times Square



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec odio. Quisque volutpat mattis eros. Nullam malesuada erat ut turpis. Suspendisse urna nibh, viverra non, semper suscipit, posuere a, pede. Donec nec justo eget felis.

Empire State Building at Night



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec odio. Quisque volutpat mattis eros. Nullam malesuada erat ut turpis. Suspendisse urna nibh, viverra non, semper suscipit, posuere a, pede. Donec nec justo eget felis.

Chilling in Central Park



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec odio. Quisque volutpat mattis eros. Nullam malesuada erat ut turpis. Suspendisse urna nibh, viverra non, semper suscipit, posuere a, pede. Donec nec justo eget felis.

The New York Skyline



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec odio. Quisque volutpat mattis eros. Nullam malesuada erat ut turpis. Suspendisse urna nibh, viverra non, semper suscipit, posuere a, pede. Donec nec justo eget felis.

Screenshot showing post thumbnails displayed next to respective post content.

```

<?php // second loop for post thumbnails
global $post; $args = array('showposts=10&offset=0');
$custom_posts = get_posts($args);
foreach($custom_posts as $post) : setup_postdata($post); ?>

<a class="postThumb" href="<?php the_permalink() ?>">
    " />
</a>

<?php endforeach; // end second loop ?>

```

Notice the two parameters we are passing to `get_posts`: `showposts` specifies the number of posts to display and `offset` lets us skip any number of posts.

This example shows how custom-field functionality greatly facilitates the use of WordPress as a CMS. Instead of posting static slabs of content, use custom fields to articulate stuff like thumbnails, media links, taglines, subtitles, author info, and anything else that you want to display apart from `the_content()`.

Takeaway: Custom fields rock hard by making post content more flexible.

5.5.3 Users, Roles and Permissions

WordPress provides excellent support for multiple users. Any WordPress-powered blog is capable of supporting a wide range of different users, each capable of performing a different set of roles.

The term “users” refers to anyone registered with your site. What actually defines a user, however, are the different things that they are permitted to do. The things that users can do are called “roles,” which by default include the following:

get_posts()

In the first part of this code, we are setting up the variables and logic required for a second `get_posts` loop.

get_post_meta()

In the next part of this code, we display the linked images using the `get_post_meta` tag and some markup.

End the loop.

Lastly, we end the `get_posts` loop with `endforeach`.

Custom-Fields Tutorials

For more in-depth information on working with custom fields, check out these two tutorials:

<http://digwp.com/u/515>

<http://digwp.com/u/516>

End the loop.

An entire book could be written about WordPress' Roles and Capabilities. It's powerful, flexible, and explained in gory detail at the WP Codex:

<http://digwp.com/u/517>

True Subscribers

Even though subscribers don't have any real admin privileges, they are still logged in and thus the function `is_user_logged_in()` will return `TRUE`. This could be a way to give registered users extra content.

Got Subscribers?

In the General Settings, you choose whether or not to allow open registration for your site. If you enable people to register, you can also set the Default Role for New Users. Recommended to leave this set at the Subscriber level unless you know what you're doing

- **Super Admins** - Super Admins are all-powerful users with complete control over everything, **but** they only exist when using Multisite for a site network.
- **Administrators** - Think of this as “super-admin” for regular, singular installations of WordPress, providing full access and privileges to everything other than the Multisite stuff (which won't exist unless you enable it).
- **Editors** - Editors may publish and edit posts and manage the posts of others.
- **Authors** - Authors may publish and edit their own posts, but not the posts of others.
- **Contributors** - Contributors may write and manage their posts, but not publish them.
- **Subscribers** - Subscribers are visitors who have registered with your site. Their privileges are no different than those of the common visitor, but subscribers have shown extra interest in your site by providing their information while registering for your site.

Each of these roles may be assigned any number of specific permissions, or “capabilities.” When applied to a specific role, capabilities enable all users of that particular role to do things like post content, edit posts, moderate comments, and so on. There is no limit to which capabilities may be enabled for a particular role. For example, you could give subscribers more capabilities than administrators.

The entire user-management system is extremely flexible, enabling you to customize and configure your WordPress installation for even the most complex CMS applications. The multiple-user functionality is an essential component of WordPress' CMS capabilities, and may be enhanced further with a variety of plugins, which we will explore a little later in the chapter.

5.5.4 Categorizing, Tagging, and Custom Taxonomies

WordPress also provides extensive organizational capabilities through the use of a highly flexible system of categorizing and tagging. In a nutshell, categories may be

used to define content that is broadly associated, such as “movies,” “music,” and “food,” while tags are used to associate finer distinctions, such as “tacos,” “pizza,” and “cheeseburgers.”

Posts may belong to any number of categories, and then tagged along more specific characteristics. For example, a post categorized under “movies” may be tagged with “sci-fi,” “space,” “future,” and “aliens.” This level of organization provides much more power over the structure of your site.

In WordPress 2.3, WordPress implemented a new “Taxonomy API” that changed the way content is categorized and organized. Since then, users have been able to create custom taxonomies to organize their content as effectively as possible. The concept is a little difficult to grasp at first, but if you think of taxonomies as “groups of tags” then you’ve pretty much got it. By default, WordPress organizes your content with the following three taxonomies:

- **category** - used for classifying posts into categories
- **post_tag** - used for tagging posts with various tags
- **link_category** - used to classify links into categories

In general, the tags and categories that belong to a taxonomy are called “terms.” Thus, your post tags are all terms of the “post_tag” taxonomy, your categories are all terms of the “category” taxonomy, and so on.

That’s great, but how does all of this new taxonomy mumbo-jumbo help us use WordPress as a CMS? Basically, taxonomies provide yet another level of classification and organization for your content. Once you create a new taxonomy, you can add as many terms (tags or categories) as desired. Let’s look at an example.

For a **Web-development tutorial site**, you could create taxonomies for “topics,” “languages,” and “applications.” Then, each of these taxonomies would be given a series of terms that applies to each post. So for example, a post on Ajax-powered contact forms could be classified with something like this:

Custom Taxonomies

We cover the “how” of custom taxonomies in chapter 2.4.7.

Update: also check out chapter 3.2.5 in the supplemental book for more information on the new custom-taxonomy functionality included in WordPress 3.0.

Taxonomies @ Codex

For more information on WordPress custom taxonomies, check out the official Codex:

<http://digwp.com/u/518>

Custom Post Types

Update: for even more flexibility in creating, organizing, and displaying custom-types of content, check out WordPress' new Custom Post-Types functionality. Learn more in chapters 2.2.9 and 3.2.7 in the supplemental bonus book.

- **Topics** - ajax, forms, wordpress
- **Languages** - javascript, php, sql
- **Applications** - blogs, e-commerce

This hierarchy enables you to create advanced menus and tag lists in your theme design. Clearly, the ability to create taxonomies opens the doors to new possibilities, enabling you to structure your site in elaborate, complex ways.

5.5.5 Page Templates

Page Templates enable you to apply custom design and functionality to your different WordPress pages. This feature enables you to craft special pages that will process and display content according to your specific needs. Consider the following page types and how Page Templates are used with each:

- **Search Page** - display results in title-only list format and include a search-term heading
- **Archives Page** - organize vast amounts of archived content with template tags and multiple loops
- **Tag Archives Page** - display a tag cloud along with popular and recently updated tags

Implementing custom Page Templates is easy. The default Page template in your theme is called "page.php". If no custom page templates have been created, all of your WordPress Pages will be displayed according to the default page.php template. To create and use a new type of page template – for example, a template with no sidebar and no title – simply create a file within your theme directory named "page-nosidebar-notitle.php" and place the following code at the top:

```
<?php /*  
    Template Name: No Sidebar No Title  
*/ ?>
```

✓ Default Template

Additional Posts

Archives

Archives Getter

Articles

Auto Complete

Book – Errata

Book Area

Comment Graph

Custom Login Demo

Downloads

No Sidebar

Random Posts

Store

Testing

Thank you

Thumb Archives – Grid

Thumb Archives – Horz

After uploading, navigate to the Search Page in the WordPress Admin (under **Pages > All Pages**) and select your custom Page Template as seen in the screenshot to the right.

Your new custom Search Page may now include any code you desire – custom loops, search listings, and so on. This same technique for creating a custom Search Page may be used to create any number of unique page templates, thereby facilitating a more flexible system for managing your online content.

5.5.6 Page, Category, and Tag Hierarchies

Another awesome CMS-enabling feature is the ability of users to create page, category, and even tag hierarchies (via taxonomies), whereby any number and level of “child” items may be created for any given “parent” item. You may create any number of first-level, “child” items, as well as any number of next-level items.

For example, a page named “Contact” that is used for a sports website may feature a parent-child page hierarchy that organizes contact data into logical sections:

Contact	[first-level page]
Swim Team	[second-level page]
Mary Poppins	[third-level page]
Tom Thumb	[third-level page]
Jack Horner	[third-level page]
Soccer Team	[second-level page]
Nick Mason	[third-level page]
Rick Wright	[third-level page]
David Gilmour	[third-level page]
Roger Waters	[third-level page]
Chess Team	[second-level page]
Anakin Skywalker	[third-level page]
Obi-Wan Kenobi	[third-level page]
Chewbacca	[third-level page]

More than just Pages

As discussed in the previous section, this same parent-child logic is also available for the creation of subordinate categories and tags.

This parent-child functionality provides extremely flexible organizational capabilities to WordPress users and designers, enabling efficient management for even thousands of posts and pages.

5.5.7 Dynamic Menus

Easy Custom Menus

Update: with version 3.0, WordPress now provides complete control over the creation and display of your menus, all from within the comfort of the WP Admin. See chapter 3.2.6 for details.

Every site needs a solid, well-structured navigational system, and for CMS-type sites this is especially true. Fortunately, WordPress provides a number of powerful functions that help users create just about every type of menu imaginable.

wp_list_pages()

Automatically lists pages in just about every possible configuration imaginable. You can control which pages are shown, limit the appearance of child pages, as well as format page titles, customize links, specify sort order, and much more.

wp_get_archives()

Automatically lists yearly, monthly, and even daily archives for your blog. Provides parameters to limit maximum number of items displayed, display associated post count, and control the formatting of the list itself.

wp_list_categories()

Displays a list of categories that is completely customizable. There are parameters for everything from excluding specific categories and specifying sort order to inclusion of feed links control over category depth. Powerful stuff!

wp_tag_cloud()

Displays a collection of links, known as a “cloud,” for each tag used on your blog. The relative size of each tag may be set based on popularity, or it may be kept static for a more uniform-looking cloud.

Tag Clouds

See chapter 4.3.4 for more information about tag clouds.

More Nav Plugins

For more great navigational plugins, the WordPress Codex is a great place to start:

<http://digwp.com/u/122>

WordPress’ built-in navigational functionality is usually plenty sufficient for setting up proper navigation on your site, but there is always room for improvement. Here are a few of the many freely available plugins that help improve and enhance WordPress’ default navigation options:

- **WP-PageNavi** <http://digwp.com/u/133>
Transforms archived page navigation into a row of links to all of your pages, enabling users (and search engines) to access any archive page with a click.
- **Breadcrumb NavXT** <http://digwp.com/u/134>
Generates locational breadcrumb trails to help visitors where they are in the grand scheme of things. The breadcrumb trails are highly customizable.
- **Wordpress Navigation List Plugin NAVT** <http://digwp.com/u/136>
Enables complete control over the creation, styling and contents of your site's navigation. Easy drag-n-drop interface makes menu customization a breeze.
- **WP-dTree** <http://digwp.com/u/137>
Generates navigation trees for your posts, pages, links and categories. Uses WordPress' built-in Scriptaculous library for awesome display effects.
- **Sub Pages widget** <http://digwp.com/u/123>
Displays the pages which are children from the current root page.

5.6.1 Extending CMS Functionality

The good news is that there are many ways to extend WordPress' built-in CMS functionality. The bad news is that we've relocated the information to the book's companion site, **DigWP.com**: <http://digwp.com/u/568>. Check it out! :)

But wait, there's more...

Truly, there is no end to the ways that WordPress can be extended. In this chapter, we have seen how to extend WordPress with plugins, theme files, custom functions, and the advanced functionality available in WordPress itself.

Even with all of these possibilities, there is yet another powerful way to extend your site's potential. WordPress' powerful content-syndication system enables you to share your content with the farthest reaches of the Internet...

Blog > Code > Breadcrumb NavXT

Blog > WordPress > Archive by ca

Blog > Articles by: mtekk

Blog > 2008 > September > 04

Blog > WordPress > Plug-ins > Blog

Breadcrumbs Rock

For large or complex sites, breadcrumb navigation can be a hugely beneficial part of your site's navigational system. They help users orient themselves within the site while enabling quick and easy links to other parts of the site.

The Tao of WordPress

Want to go further with different ways to extend and customize WordPress? Check out "The Tao of WordPress", a complete guide for beginners, admins, and users:

<http://wp-tao.com/>

When I am working on a problem, I never think about beauty. I only think about how to solve the problem. But when I have finished, if the solution isn't beautiful, I know it is wrong.

– **RICHARD BUCKMINSTER FULLER**

6

Working with RSS Feeds

6.1.1 Working with RSS Feeds

WordPress is awesome because you can just start posting content and have it appear instantly on your website. As you publish your blog posts, WordPress generates a variety of feed formats that contain the latest posts for your site.

Out of the box, WordPress creates feeds for just about every type of page-view: category and tag archives, date archives, comment threads, and so on. For typical blogs, most visitors don't bother with anything less than a site's full, main-content feed and maybe the complete, all-comments feed. People grab these two main feeds to stay current with just about everything happening at your site. They add your feeds to their feed-reader of choice, and then enjoy all of your site's feed-delivered content at their convenience.

As dynamic-publishing and content-distribution tools, feeds have revolutionized the way content is delivered and consumed on the Web. Instead of people having to visit your site to read your content, they can simply open their feed reader and browse all of the latest articles from all of their favorite sites. The ease of this technology makes it easier to stay current with a much larger volume of information. It's a good thing, and WordPress helps you be a part of the action.

6.1.2 The Pros and Cons of Delivering Feeds

Anyone may visit your site. Likewise, anyone may subscribe to your feed. This easy access and user-convenience is a double-edged sword, however, especially for sites

Don't use relative URLs in your posts!

Relative links look like this: `"/path/to/file.html"`, and may function properly from within your posts, but they will break when included in your feed. Because feeds are distributed on other domains, relative URLs for links and images will assume an incorrect base URL. If you are unable to avoid relative URLs on WordPress blogs, use a plugin such as **URL Absolutifier** <http://digwp.com/u/84>, which filters the feed content and converts any relative URLs to their absolute counterparts.

that depend on advertising revenue for sustainability and profit. Many people will not bother clicking through to your site if they are getting all of your content directly from your feed. Here are some other possibilities:

- Loss of site traffic, fewer click-throughs, visits, et al.
- Loss of brand identity because of feed-reader uniformity
- Stolen content due to evil spammers and scrapers

Fortunately there are some excellent solutions for dealing with these issues. But even with the potential downsides, delivering feed content is a great way to reach out and build an audience.

But before getting too far ahead of ourselves, let's back up a bit and review the different types of feeds available for WordPress-powered sites.

6.2.1 Different Types of WordPress Feeds

WordPress provides users with a wide variety of different feeds. There are feeds for just about everything: categories, tags, archives, and much more. WordPress also enables you to create custom feed content to do stuff like include or exclude specific tags or categories. And to make things even more flexible, WordPress provides all of your feeds in multiple formats: RSS, RDF, RSS2, and Atom. Sounds overwhelming at first, but relax – we show you everything you need to know.

6.2.2 Posts Feed

Your site's Post feed or main-content feed is the primary source of content for subscribers. Unless specified otherwise with a plugin or custom script, each

Choose Your Feed Reader!

Quick review of some popular feed-reader choices

There are many feed readers available to choose from, including online services, mobile applications, and even desktop software. While all of these different readers do essentially the same thing – aggregate and display your favorite feeds – there are some key differences that you should consider. Let's check out some of the more popular choices.

Google Reader <http://digwp.com/u/160>

Millions of people use Google Reader. It is super-easy, fast, and provides many options, including the ability to “star” favorites, “like” posts, and even “share” feed items on your own website. Highly recommended.

Bloglines <http://digwp.com/u/162>

Free and easy online feed aggregation service for searching, subscribing, creating, and sharing news feeds, blogs, and rich web content. Many features and fully customizable, including full support for mobile browsers and multiple languages.

NewsGator <http://digwp.com/u/163>

Award-winning desktop-based RSS reader that comes in a variety of flavors: Windows, Mac, and iPhone.

Thunderbird <http://digwp.com/u/164>

The powerful open-source email client is also an extremely powerful feed reader. Supports just about any feed format you can throw at it, and makes subscribing, managing, and reading feeds a real breeze.

Firefox <http://digwp.com/u/165>

The world's best browser makes it super-easy to stay current with all of your favorite sites. Using Firefox's Live Bookmarks feature, you can automatically keep track of your feeds and know instantly when any of them are updated.

Of course, this list is just the tip of the iceberg when it comes to feed readers. A quick search on your favorite search engine will provide many more great options.

What is that domain?

*In most of our examples, we use `domain.tld` for any URLs that need included. “domain” refers to your domain-name, and “tld” refers to “top-level domain”, which is what your `.com`, `.net`, and `.so-on` are called. So whenever you see “`domain.tld`”, just think “*your-website.whatever*”.*

published post will appear in your site’s Posts feed. Only *post content* appears in the main-posts feed. It is available from your site at the following URL:

`http://domain.tld/feed/`

Most sites serve their main content feed as their site’s primary feed. By default, it contains everything that is posted on your site, thereby making it easy for subscribers to stay current with all of your posted content.

6.2.3 Comments Feed

Your site’s Comments feed includes a chronologically ordered display of **all comments** left on your site. It is available from your site at the following URL:

`http://domain.tld/comments/feed/`

Many sites also provide visitors with a link to *subscribe* to their comments feed. Typically, you will see links and/or icons for both the “main posts feed” and the “main comments feed” appearing next to one another. Providing these two feeds makes it easy for your readers to stay current with all of *your content* and all of *your visitor’s comments*.

6.2.4 Individual Post Comments Feed

For *each post* on your site that is open to comments, there is a corresponding feed that people may subscribe to in order to follow the conversation. Each Post-Comments feed includes *all comments* for that particular post. Here’s an example:

`http://domain.tld/2009/10/billy-mays-fan-club/feed/`

Post-Comment feeds are useful, but in our experience people prefer to subscribe to comment updates via email. See the popout in chapter 6.3.1 for more information.

6.2.5 Category and Tag Feeds

Each individual Category and Tag archive on your site features its own feed. Each of these feeds includes all of the post content for its respective Category or Tag. Here is the general format for each type:

`http://domain.tld/category/football/feed/`

`http://domain.tld/tag/football/feed/`

Category and tag feeds are extremely useful for providing topic-focused content to your subscribers. Sites that cover more than one particular niche may provide a feed for each topic, which may be more useful to readers than receiving news that is of no interest. See the sidebar note for other ideas.

What feedest thou?

Another good example for a category-specific feed is seen in the case of “Side Blogs”, “Asides”, “Mini Updates”, or whatever they’re called these days. Many bloggers keep a category set apart for posting brief thoughts, small snippets of news, links, and so on. A category-specific feed is perfect for enabling your subscribers to stay current with your “side” ramblings.

6.2.6 Other Feed Types

There are also many other types of feeds that are automatically generated from your WordPress-powered site. There are feeds for each different author, chronological archives, and even pages. Many of these types of feeds are rarely seen in the wild, but they do exist and are available should you decide to use them.

- **Author feed** - `http://domain.tld/author/blake/feed/`
- **Yearly archives** - `http://domain.tld/2009/feed/`
- **Monthly archives** `http://domain.tld/2009/10/feed/`
- **Daily archives** - `http://domain.tld/2009/10/30/feed/`
- **Page feed** - `http://domain.tld/about/feed/`

In these examples, replace “domain.tld” with your domain name, and then select an author name and/or specific date(s). Experiment, learn, and have fun!

List Category Feed Links with a Nice Feed Icon

Here is a nice way to provide your visitors with a list of all your category feeds. All we need is the `wp_list_categories()` tag and a few parameters:

```
<?php wp_list_categories('feed_image=http://digwp.com/feed-icon.png&depth=1'); ?>
```

This will output a nice list of all your parent-level categories along with a feed link icon next to each. Remember to use your own path for the feed image! For complete info on customizing this highly flexible tag, see the Codex: <http://digwp.com/u/86>

6.3.1 Feed Configurations & Formats

Just as there are many different types of feeds, there are also many different types of feed *formats*. All of your feeds are generated in a variety of flavors:

Page Feeds

Pages do create feeds, but unless you have comments enabled on that page, it usually doesn't make sense to offer visitors a feed link for a page.

- **RSS 2.0**

Owned by Berkman Center, the RSS 2.0 format is extensible via modules and is recommended for general-purpose, metadata-rich syndication.

- **RSS 1.0 / RDF**

Owned by the RSS-DEV Working Group, the RSS 1.0 format is based on RDF, extensible via modules, and recommended for RDF-based applications.

- **RSS 0.92**

Owned by UserLand, the RSS 0.92 format allows for richer metadata than its 0.91 predecessor. This version is now obsoleted by version 2.0.

- **Atom**

Created by leading service providers, tool vendors and independent developers, Atom is an XML-based document format designed to be a universal publishing standard for personal content and weblogs.

Several years ago, choosing the best format to provide on your site was very important. Different feed readers accepted only certain formats, and the war was raging to see which format would finally win out. Fortunately, we no longer need to worry about which format to use because all of the most popular feed readers

provide support for virtually all different feed formats. Many of the sites that we visit and interact with continue to deliver their feeds in RSS-2.0 format. The Atom format also has a loyal following, but is nowhere near as popular as the RSS 2.0 format.

With all of these different formats, how does one distinguish between them? The answer is found in the feed's URL structure. Consider these examples of different formats for the main feed:

- **RSS 2.0 format** - `http://domain.tld/feed/`
- **RSS 2.0 format** - `http://domain.tld/feed/rss2/`
- **RSS 0.92 format** - `http://domain.tld/feed/rss/`
- **RDF/RSS 1.0 format** - `http://domain.tld/feed/rdf/`
- **Atom format** - `http://domain.tld/feed/atom/`

This same basic pattern applies to all different types of feeds. Unless a particular format is appended to the end of the feed URL, the format is RSS 2.0. Otherwise, the format is determined by the particular `"/format/"` appended to the feed URL. For example, here are the various feed formats available for a hypothetical "Coffee" category:

- `http://domain.tld/category/coffee/feed/`
- `http://domain.tld/category/coffee/feed/rss2/`
- `http://domain.tld/category/coffee/feed/rss/`
- `http://domain.tld/category/coffee/feed/rdf/`
- `http://domain.tld/category/coffee/feed/atom/`

Now that we are familiar with the myriad feeds provided by WordPress, let's dig into the configuration and optimization of your site's feeds. The first thing you need to decide is whether to deliver "full feeds" or "partial feeds."

Subscribe to Comments!

Enable your visitors to stay current via email

The easiest way to stay current with comment threads is to "subscribe to comments via email." This functionality isn't included with WordPress by default, but is easily added with Mark Jaquith's excellent plugin, [Subscribe to Comments: `http://digwp.com/u/85`](http://digwp.com/u/85)

Once installed, `Subscribe to Comments` provides a checkbox next to your comments that enables users to receive email notifications of any updates to that particular comment thread.

Once subscribed, users will receive a simple plain-text email notifying them of the new comment, as well as links to manage their email subscriptions for that particular site. Unsubscribing at any time is as easy as clicking a few links.

Feed URL Canonicalization

Perhaps you've noticed that, for every feed on your site, there are **two** versions of the feed in the popular RSS 2.0 format.

The **first RSS-2.0 format** is available when no specific format name is appended to the feed URL.

The **second** RSS-2.0 format is available when the `"/rss2/"` format name is appended to the feed URL.

To simplify this duplicity, you may want to "canonicalize" your RSS-2.0 feed.

Canonicalization is the process of ensuring that each feed is available only at a specific URL.

In general, canonicalization is good for SEO, statistics, usability, and more. Here is a quick `.htaccess` trick for ensuring that your RSS-2.0 feed is always delivered via the `"http://domain.tld/feed/"` URL:

```
RedirectMatch 301 ^/rss2$ http://  
domain.tld/feed/
```

Just add that rule to your site's web-accessible root `.htaccess` file and test the results. Viola! Feed canonicalization in seconds.

Of course, there are many more canonicalization tricks at your disposal, as we discuss further in chapter 8.

6.3.2 Full Feeds

Within the WordPress Admin (under **Settings > Reading**), there is an option to generate "full" feeds or "partial" feeds. By setting your feeds to deliver *partial* content, each post item in your feed is truncated according to either a specific number of characters, the presence of a post excerpt, or the location of the "read-more" (`"<!--more-->"`) tag. In short, partial feeds show *partial* content.

On the other hand, configuring your site to deliver *full feeds* means that you want your feed to include the *entire content* for each published post. While this certainly makes your feed more attractive to potential subscribers, it also invites greater opportunity for abuse. Many unscrupulous "scraper" sites on the Web make money by subscribing to full feeds, displaying them on their own sites, and placing advertisements next to them. Many other scraper sites simply steal your content outright to pass off as their own work.

As mentioned, there is another downside to providing full-content feeds. People are less likely to click through to your site to read the article when they can enjoy the entire thing in the comfort of their own feed reader. Fewer click-throughs means less traffic. Less traffic, in general, is not a good thing, especially if you rely on visitors for advertising revenue. Perhaps worse, not clicking through to your site, visitors are seeing less of you and your unique brand.

Despite these potential downsides, many bloggers swear by the "full-feed" format (including us – DigWP.com delivers a *full* feed). The reasoning behind this is that feeds are all about *ease of distribution* and *open sharing of content*. As soon as you begin to restrict this core functionality, you detract from the benefits feeds provide while appearing manipulative and miserly. In other words, providing full feeds is generally seen as a very cool thing to do; while partial feeds on the other hand ...not so much.

6.3.3 Partial Feeds

While many people live and die by the full-feed method, there remain unavoidable issues involved with doing so. Fortunately, many, if not all, of the previously discussed difficulties are easily resolved by simply providing only partial feeds to your visitors. Partial feeds typically feature only the first few sentences of each post, or else display any explicitly defined excerpts included with each post.

The partial-feed format requires readers to actually click-through and visit your site if they decide the post is something they would like to read in its entirety. Thus, in visiting your site, traffic levels rise and revenue may proportionally increase.

Of course the other major benefit to only serving partial feeds is that scrapers and content thieves will have no real use for your content. They don't waste time displaying partial feeds on their illicit sites, and the scrapers are generally *too lazy* to do anything that can't be automated.

The bottom line when it comes to displaying full versus partial feeds is that it all depends on your specific needs. If you are more concerned about ad revenue than being seen as someone who "gets it," then partial feeds are most likely the way to go. Conversely, if your goal is to produce and share your content with as many like-minded subscribers as possible, then you don't want to short-change them by providing anything less than the "full"-meal deal!

6.3.4 Number of Posts

While configuring your feed for either full or partial post display, you should also consider the number of posts included in your feed. In the WordPress Admin, under **Settings > Reading**, you will see two fields, "Blog pages show at most..." and "Syndication feeds show the most recent...". These options enable you to specify how many posts should appear on your home page and in your feed, respectively.

Enabling Full Feeds

Configuring full feeds is easy! Simply go to Settings > Reading in the Admin area, click on the Full-text option and save the changes.

For each article in a feed, show ☒ Full text ☐ Summary

Partial Feeds

Configuring partial feeds is just as easy. Simply go to Settings > Reading, click on the Summary option and save the changes.

For each article in a feed, show ☐ Full text ☒ Summary

Truncated Feeds?

Older versions of WordPress had a problem where feeds would be truncated at the <!--more--> tag even when the full feeds option was checked. If this is happening to you, 1) upgrade WordPress, or if you can't, 2) use this plugin:

<http://digwp.com/u/87>

Here you may set any number of posts for your feed – this may be different than the number set for your archive posts.

Syndication feeds show the most recent

10 posts

Note that when it comes to your feeds, the number of posts specified in the WordPress Admin applies to every feed produced by your site, including comment feeds!

6.3.5 WordPress Feed Formats

WordPress Post-feed formats for permalink-enabled URLs

If you have permalinks enabled on your site, your Posts feed is accessible via the following URLs:

<code>http://domain.tld/feed/</code>	- RSS 2.0 format
<code>http://domain.tld/feed/rss2/</code>	- RSS 2.0 format
<code>http://domain.tld/feed/rss/</code>	- RSS 0.92 format
<code>http://domain.tld/feed/rdf/</code>	- RDF/RSS 1.0 format
<code>http://domain.tld/feed/atom/</code>	- Atom format

WordPress Post-feed formats for default URLs (non-permalink)

By default, your Posts feed is accessible via the following file-based URLs (even when you have permalinks enabled):

<code>http://domain.tld/wp-rss2.php</code>	- RSS 2.0 format
<code>http://domain.tld/wp-rss.php</code>	- RSS 0.92 format
<code>http://domain.tld/wp-rdf.php</code>	- RDF/RSS 1.0 format
<code>http://domain.tld/wp-atom.php</code>	- Atom format

Finding Balance

When choosing the number of posts to display on your home and archive pages, try to find the right balance between providing plenty of content and delivering speedy web pages. Too much content can slow things down for users with slow connections, but not enough content can leave your site looking empty.

If you host your own RSS feeds (rather than use something like FeedBurner) also consider balance. The more articles you show in your feed, the more bandwidth intensive serving that feed is.

WordPress Post-feed formats via query string (non-permalink)

Alternately, your Posts feed is also available at the following query-string-based URLs (even when you have permalinks enabled):

```
http://domain.tld/?feed=rss2 - RSS 2.0 format
http://domain.tld/?feed=rss - RSS 0.92 format
http://domain.tld/?feed=rdf - RDF/RSS 1.0 format
http://domain.tld/?feed=atom - Atom format
```

Display default Post-feed URLs

To determine/display the default posts feed URL for your blog's main content, place any or all of these template tags into a useful location in one of your theme files:

```
<?php bloginfo('rss2_url'); ?> - RSS 2.0 format
<?php bloginfo('rss_url'); ?> - RSS 0.92 format
<?php bloginfo('rdf_url'); ?> - RDF/RSS 1.0 format
<?php bloginfo('atom_url'); ?> - Atom format
```

WordPress main comments feed

Your blog's main comments feed is available only in RSS 2.0 format, but there are several URL options from which to choose:

```
http://domain.tld/comments/feed/ - Permalink format
http://domain.tld/wp-commentsrss2.php - Default file-based format
http://domain.tld/?feed=commentsrss2 - Query-string format
```

Display main comments feed URL

To display the default URL for your main comments feed, add this template tag to

your theme file and load the page in your browser:

```
<?php bloginfo('comments_rss2_url'); ?>
```

Post-specific comment feeds

By default, every post also delivers its own feed featuring all of its comments. To display feed URLs for individual, post-specific comment feeds, place this template tag anywhere in the main post loop or comment loop:

```
<?php comments_rss_link('Subscribe to comments on this post via RSS'); ?>
```

Alternately, to display the comment-feed URL for any specific post, simply append either of the following to the original post URL:

```
feed/          - Permalink format  
?feed=rss2     - Default format
```

What is my Feed URL?

For a complete online reference of all available WordPress feed formats, types, and possibilities, check out this article:

<http://digwp.com/u/380>

Here is an example of each method for a generalized post URL:

```
http://domain.tld/individual-post/feed/      - Permalink format  
http://domain.tld/individual-post/?feed=rss2 - Default format
```

When using default (non-permalink) URLs, the post-specific comment feeds are available via the following format:

```
http://domain.tld/?feed=rss2&p=123
```

...where "p=123" references the post ID.

Category Feeds

To display individual category feed URLs, use either of the following formats:

`http://domain.tld/category/categoryname/feed/` - Permalink format
`http://domain.tld/wp-rss2.php?cat=33` - Default format

Whew! As you can see, WordPress provides a *ton* of feed formats and configuration options. And these are just the standard formats available by default. Later in chapter 6.6.5, we'll show you how to rig up just about any feed imaginable.

6.4.1 Using FeedBurner For Feed Delivery

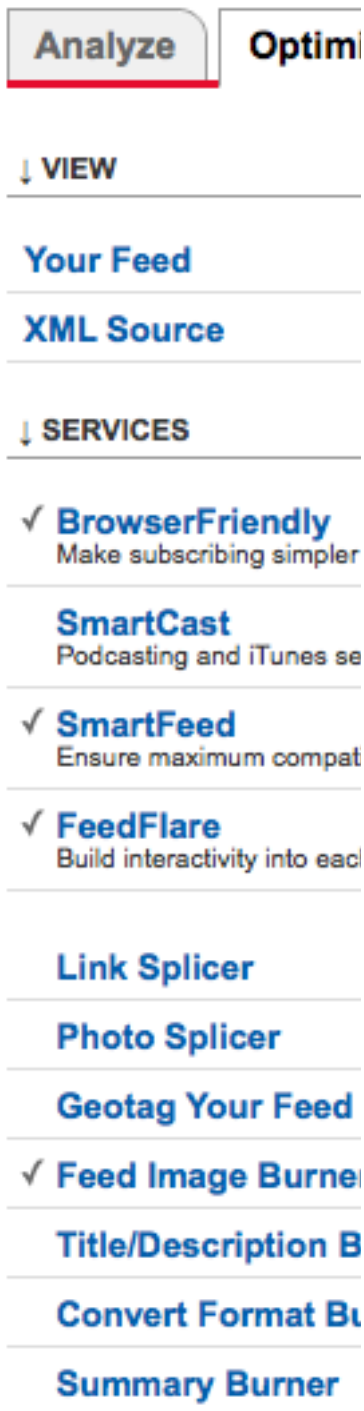
There are many ways to track your feed statistics. Perhaps the most popular feed-tracking method is the free feed-delivery service provided by **FeedBurner** <http://digwp.com/u/166>. Millions of site owners and bloggers redirect their feeds to FeedBurner, which in turn delivers the feeds to subscribers while gathering statistics on visits, click-throughs, and much more.

We use and recommend FeedBurner for all of our feed-delivery needs, so we want to take a moment to discuss why it's cool, how to use it, and some of the awesome things that it can do.

6.4.2 Benefits of Using FeedBurner

Perhaps the *best* reason to use FeedBurner is its immense popularity. With millions of users, the chance of FeedBurner closing shop is next to nil. Further, FeedBurner belongs to Google since 2007, and it is unlikely that anyone will be snatching it from their able hands. To be sure, Google has dropped the ball a few times for their FeedBurner service, but the issues are usually resolved within a bearable amount of time (it could be better), and with Google, you know that there are always people around to fix things when they break.

If that wasn't quite convincing enough, let us just emphasize the vast amount of data that is collected by FeedBurner. Everything that you could want to know



about your feeds is available to you from an elaborate user-interface in your FeedBurner account's Admin area. Here is a rundown of some of the best features that are available when you host your feed(s) at FeedBurner:

- **Awesome analytical tools** - Comprehensive subscriber statistics, click-throughs, and many other key statistics for your feed
- **Integrated AdSense advertising** - Google makes it easy to manage and monetize your feeds with AdSense
- **Universal feed delivery** - FeedBurner's SmartFeed formats your feed to work perfectly with virtually any feed reader
- **Tons of optimization options** - FeedFlares, Link Splicing, Geotagging, and many more savvy optimization tools
- **Great tools to publicize your feed** - Headline animator, Email subscriptions, feed branding, and everything else you need to help spread the word
- **Easy exportation of data** - You can export your feed stats via Excel or CSV file for easy analysis with your favorite spreadsheet or statistical software

Even better, FeedBurner is simple to set up and use, tracks everything automatically, is easy to configure, and costs absolutely nothing. Sure there are periods of downtime, but so far these have been few and far between, with consistent and reliable FeedBurner service being the norm.

6.4.3 Setting Up & Configuring a FeedBurner Account

To "burn" your feed using the FeedBurner service, you must do the following:

- Register for a Google account (a single account may host multiple feeds)
- Enter the URL of the feed that you would like to burn
- Pick a FeedBurner feed URL for each of your feeds
- Redirect your original feed URL to its corresponding FeedBurner counterpart

For the last step – redirecting your feed to FeedBurner, there are several options, including redirecting via plugin, HTAccess (or Apache configuration file), or PHP. Let's take a look at each of these methods.

6.4.4 Redirecting to FeedBurner via Plugin

Two great options for redirecting your feeds to FeedBurner are the **Feedsmith** plugin – <http://digwp.com/u/167> (direct download link, zip file) – and the **FD FeedBurner** plugin – <http://digwp.com/u/168>. Either of these plugins do a good job at automatically redirecting your site's main feed and, optionally, your main comments feed. Simply install either plugin, navigate to the plugin options page, and enter your FeedBurner feed URL(s) in the fields provided. Once done, save the options and check that everything is working as expected.

Set Up Your FeedBurner Feed

This plugin makes it easy to redirect 100% of traffic for your feeds to a FeedBurner feed and apply a variety of features you choose to improve and enhance your

1. To get started, [create a FeedBurner feed for Digging into WordPress](#). This
2. Once you have created your FeedBurner feed, enter its address into the fi

<http://feeds2.feedburner.com/DiggingIntoWordpress>

As Easy as it Gets

Installing and using the Feedsmith plugin is a no-brainer: simply enter your feed URL(s) in the plugin options section and click save. Nothing more to do – everything else happens automatically behind the scenes.

6.4.5 Redirecting to FeedBurner via HTAccess

One of the limitations of using a plugin to redirect your site's feeds to FeedBurner is the lack of customization and configurational control over which feeds are redirected. By default, either of the currently available FeedBurner-redirect plugins will redirect *all* of your WordPress feeds to your FeedBurner URL. For multi-topic or multi-author blogs, this may prevent you from delivering topic-specific feeds to your readers.

For example, what if you wanted to redirect your individual category feeds to their corresponding FeedBurner counterparts? With these plugins, this is not possible. Fortunately, we can use Apache's powerful HTAccess directives to get the job done quickly and easily. For this example, let's assume the following hypothetical collection of WordPress/FeedBurner feed URLs:

- **Main feed:** `http://domain.tld/feed/`
=redirects to=> `http://feeds.feedburner.com/main-feed`
- **Comments feed:** `http://domain.tld/comments/feed/`
=redirects to=> `http://feeds.feedburner.com/comments-feed`
- **"Business" category feed:** `http://domain.tld/category/business/feed/`
=redirects to=> `http://feeds.feedburner.com/business-feed`
- **"Pleasure" category feed:** `http://domain.tld/category/pleasure/feed/`
=redirects to=> `http://feeds.feedburner.com/pleasure-feed`
- **"Nonsense" category feed:** `http://domain.tld/category/nonsense/feed/`
=redirects to=> `http://feeds.feedburner.com/nonsense-feed`

Currently, there is not a plugin that will handle this scenario, but by placing the following Apache directives into our site's root `.htaccess` file (or `httpd.conf` file) we can get the job done quite easily:

```
RewriteCond %{HTTP_USER_AGENT} !^.*(FeedBurner|FeedValidator) [NC]
RewriteRule ^feed/?.*$ http://feeds.feedburner.com/main-feed [L,NC,R=302]
RewriteCond %{HTTP_USER_AGENT} !^.*(FeedBurner|FeedValidator) [NC]
RewriteRule ^comments/?.*$ http://feeds.feedburner.com/comments-feed [L,NC,R=302]
RewriteCond %{HTTP_USER_AGENT} !^.*(FeedBurner|FeedValidator) [NC]
RewriteRule ^business/feed/?.*$ http://feeds.feedburner.com/business-feed [L,NC,R=302]
RewriteCond %{HTTP_USER_AGENT} !^.*(FeedBurner|FeedValidator) [NC]
RewriteRule ^pleasure/feed/?.*$ http://feeds.feedburner.com/pleasure-feed [L,NC,R=302]
RewriteCond %{HTTP_USER_AGENT} !^.*(FeedBurner|FeedValidator) [NC]
RewriteRule ^nonsense/feed/?.*$ http://feeds.feedburner.com/nonsense-feed [L,NC,R=302]
```

Notice that the first two RewriteRule directives redirect the main feed and comments feed to their respective FeedBurner URLs. After that, the next three rewrite rules redirect each of the three category URLs to its respective FeedBurner URL. For each of these lines, edit both the category name and its associated FeedBurner feed URL according to your actual values.

Including additional categories is as easy as adding a new set of RewriteCond / RewriteRule directives. Just copy, paste, edit accordingly, and done! Likewise, to remove any unnecessary category feeds, simply comment out or delete them.

For example, if you want to redirect only your site's main and comments feeds, the code would look like this:

```
<IfModule mod_rewrite.c>
RewriteCond %{HTTP_USER_AGENT} !^(FeedBurner|FeedValidator) [NC]
RewriteRule ^feed/?.*$ http://feeds.feedburner.com/main-feed [L,NC,R=302]
RewriteCond %{HTTP_USER_AGENT} !^(FeedBurner|FeedValidator) [NC]
RewriteRule ^comments/?.*$ http://feeds.feedburner.com/comments-feed [L,NC,R=302]
</IfModule>
```

With that code in place, your server will redirect all requests for your blog's **Posts Feed** and **Comments Feed** to their respectively associated FeedBurner URLs. All other types of feeds (tag feeds, individual post-comment feeds, author feeds, etc.) will not be affected and will appear as normal when requested. And the best part? With this HTAccess method, there is no need to install an additional plugin – this method takes care of everything you need to redirect your feeds properly.

6.4.6 Redirecting to FeedBurner via PHP

If the thought of using HTAccess to redirect your feeds makes your skin crawl, don't sweat it – here is an alternate version for redirecting your feeds to FeedBurner using good 'ol-fashioned PHP.

Look Ma, No Plugin!

To redirect your feeds to FeedBurner without using a plugin, copy the code featured on this page and paste it into your site's root .htaccess file. This snippet of code gets the job done without any PHP processing from the server.

With Great Power...

If we haven't mentioned it already, let us take a moment to remind you that working with .htaccess is powerful stuff. Make sure you make a backup of any files that you're working with, and always test vigorously. With .htaccess, a single mistyped character will result in a server error. So be careful when using!

Credit

Thanks to Justin Tadlock for sharing this PHP-redirection technique with the WordPress Community:

<http://digwp.com/u/88>

1. Add the following code to your active theme's functions.php file:

```
function custom_feed_link($output, $feed) {
    $feed_url = 'http://feeds.feedburner.com/your-feedburner-feed';
    $feed_array = array(
        'rss' => $feed_url,
        'rss2' => $feed_url,
        'atom' => $feed_url,
        'rdf' => $feed_url,
        'comments_rss2' => ''
    );
    $feed_array[$feed] = $feed_url;
    $output = $feed_array[$feed];
    return $output;
}
add_filter('feed_link', 'custom_feed_link', 1, 2);
```

2. Edit the \$feed_url with the URL of your FeedBurner feed.

Once in place, this code will redirect all of your site's main feeds – in all formats – to your corresponding FeedBurner counterpart. Additionally, you may also redirect other types of feeds. For example, here is the code used (for functions.php) to redirect your category, author, tag, and search feeds:

The <link>

While this code doesn't redirect feeds in the sense of literally changing the URL being accessed (the other methods do), it does change the <link> element which is what tells browsers and feed reading applications where the RSS feed lives.

```
// redirect various feeds
function other_feed_links($link) {
    $link = 'http://feeds.feedburner.com/your-feedburner-feed';
    return $link;
}
add_filter('category_feed_link', 'other_feed_links');
add_filter('author_feed_link', 'other_feed_links');
add_filter('tag_feed_link', 'other_feed_links');
add_filter('search_feed_link', 'other_feed_links');
```

This code may be customized to fit your needs. For example, if you only need to redirect your category feeds, simply delete the three other `add_filter` lines in the code. Likewise, if you need to add additional feed types, simply replicate one of the `add_filter` lines and specify the feed type in the first PHP parameter. Whatever you do, remember to test, test, test that everything is working properly.

6.5.1 Tracking & Displaying Feed Statistics

Once you get your feed established and properly configured, you will want to begin offering it to your visitors. There are many ways to do this, including text links, image links, and even FeedBurner badges that display the total number of subscribers for your feed. Along with the number of subscribers for your site, the free FeedBurner service also keeps track of many other types of statistical data. From subscriber count and click-throughs to user reach and page hits, FeedBurner is your one-stop resource for in-depth feed analysis.

6.5.2 Types of Statistics Provided by FeedBurner

As mentioned, FeedBurner provides some great tools for managing, optimizing, and tracking your feeds. FeedBurner statistics include:

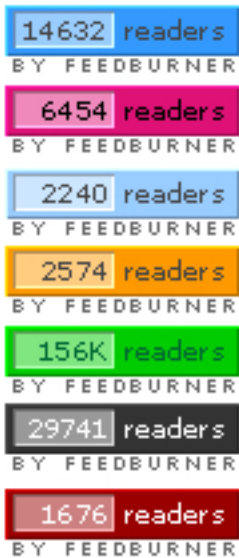
- **Subscriber counts** – daily totals and chronological history
- **Feed reader applications** that are used to access your feed
- **Uncommon uses**, including re-syndication
- **Reach** – the unique number of people who view or click your feed content
- **Item views** – tracking of access to specified feed items
- **Item link clicks** optimized for complete item use or search engine ranking
- **Item enclosure downloads**, including podcast downloads

As you can see, FeedBurner tracks just about everything you need to know. Many of these statistics are included automatically with your account, but others are “opt-in” and require you to activate them. Fortunately, FeedBurner makes managing, using, and exporting your feed stats as pain-free as possible, with everything fully accessible and configurable from within your account area.

6.5.3 Displaying FeedBurner Statistics

By far, one of the most popular uses of FeedBurner statistics is the public display of a feed’s subscriber count. There are many ways to do it, including those little rectangular badges frequently seen in the sidebars of blogs and sites around the Web. These free “chicklets,” as they’re called, are delivered by FeedBurner and configurable via the FeedBurner Admin area.

Depending on your site’s design, however, you may prefer to display your subscriber count in a way that better suits your specific design. Fortunately, there are alternate, “text-only” ways of displaying your subscriber count, enabling you to markup and style the numbers to fit your design. Here are two ways to do it.



FeedBurner Chicklets

You can choose animated and non-animated chicklets, as well as customize the colors used.

Method 1: Use PHP to display FeedBurner subscriber count

Using FeedBurner’s **Awareness API** and a bit of PHP, it is possible to display your FeedBurner subscriber count as plain text. Here’s how to do it in three easy steps:

1. Open your `sidebar.php` theme file and add the following code to the location where you would like to show off your subscriber count:

```
<?php // display FeedBurner count as plain text
$feed = "https://feedburner.google.com/api/awareness/1.0/GetFeedData?uri=digwp";
$curl = curl_init();
curl_setopt($curl, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($curl, CURLOPT_URL, $feed);
$feed = curl_exec($curl);
```

```
curl_close($curl);  
$xml = new SimpleXMLElement($feed);  
$feedburner = $xml->feed->entry['circulation'];  
echo $feedburner;  
?>
```

2. Replace the term “digwp” in the first line with the username of your FeedBurner account.
3. Done! Markup and style the plain-text subscriber-count output as desired and enjoy the design-savvy results.

Method 2: Use a plugin!

If you would rather just jump in and use a plugin to display your FeedBurner stats, Francesco Mapelli’s excellent **Feed Count** plugin (see side note) does the job very well. The plugin is a snap and provides everything needed for customized text-display of your subscriber count. With the plugin’s text output of your feed stats, it’s possible to echo virtually any message or markup to embellish your data.

6.5.4 Alternatives to FeedBurner

During the writing of this book, all FeedBurner accounts were transferred to Google. During this process, an enormous percentage of FeedBurner users experienced a sudden, drastic decrease in the number of reported subscribers. This was very upsetting to say the least, and led many people to begin exploring alternatives to the now-questionable FeedBurner service.

“Are there any alternatives to FeedBurner?” We get this question every time FeedBurner drops the ball. Until recently, FeedBurner was pretty much *it* for delivering feeds and keeping track of statistics. This is one of the reasons why so many sites use FeedBurner – the competition has been scarce. Fortunately, some promising FeedBurner alternatives are finally gaining some traction:

Feed Count Plugin

Unfortunately, at the time of this writing, the Feed Count web page was marked as an “attack site.” So, until the author resolves the issue, we have made the plugin available through our site at this URL:

<http://digwp.com/u/378>

To the Streets!

The FeedBurner problems made a lot of folks angry and was documented by TechCrunch here:

<http://digwp.com/u/90>

- **FeedBlitz** <http://digwp.com/u/91>
FeedBlitz has been around as long as I can remember and should have been in the feed-delivery/tracking game from the beginning. Even so, better late than never with their new RSS delivery service.
- **RapidFeeds** <http://digwp.com/u/92>
RapidFeeds publishes and monitors your RSS feeds for free. Looks relatively new but very promising.
- **FeedStats** <http://digwp.com/u/93>
FeedStats is designed to track the total number of people who are subscribing to your various feeds. The total number of subscribers takes into account different feed formats, such as RDF, RSS, and Atom.
- **Feed Statistics** <http://digwp.com/u/94>
In addition to tracking your daily subscriber count, the Feed Statistics plugin also monitors and reports information about which feed readers are being used to access your feeds, as well as data on Post Views, Top Feeds, and Click-throughs – all within the convenience of the WordPress Admin.

6.6.1 Customizing Feeds

Once your feed is up and running, there are many ways to customize its appearance and functionality. If you are using FeedBurner, simply log into your account and choose your settings. Not using FeedBurner? No sweat, it's easy to customize your WordPress feeds with better formatting, custom images, feed comments, and much more. Let's dig in and check it out...

6.6.2 Formatting Feed Images

Images, as in `` elements, are by default “inline” elements. They have a width and a height, but they don't break lines and kind of just go with the flow of the text around it.

Prevent FeedBurner Errors with a Fallback Mechanism

As awesome as it is, the Feed Count plugin can't work when the FeedBurner data is unavailable. Despite its best intentions, FeedBurner occasionally returns inaccurate data for the subscriber count. For those of us who care about the accuracy of our publicly displayed feed statistics, displaying information like this on your site is simply unacceptable:

Join N/A awesome subscribers!

...or perhaps even worse:

Join 0 awesome subscribers!

Ugh. Clearly *not* the way to leave a good impression and encourage visitors to subscribe to your feed. Fortunately, we have an excellent fallback mechanism to ensure that all visitors see the correct subscriber information. Begin by placing the following script in your active theme's `functions.php` file:

```
<?php
function feedcount_fallback() {
    if(function_exists('fc_feedcount')) {
        ob_start();
        fc_feedcount();
        $count = ob_get_contents();
        ob_end_clean();
        if($count == 'N/A' || $count == '0') {
            echo 'many other';
        } else {
            echo $count;
        }
    } else {
        echo 'many other';
    }
}
?>
```

Once that code is in place, replace the default Feed Count function call with this in your theme's template file:

```
<p>Join <?php if (function_exists('feedcount_fallback'))
feedcount_fallback(); ?> awesome subscribers!</p>
```

Once in place, that code will output one of the following messages, depending on availability of accurate FeedBurner subscriber-count data:

When the subscriber count is correct:

"Join 243 awesome subscribers!"

And when the subscriber count is either "N/A" or "0":

"Join many other awesome subscribers!"

How does this script work? Basically, we are using PHP's output buffering functionality to capture the output value of the `fc_feedcount()` function and compare it to the two possible error values ("N/A" and "0").

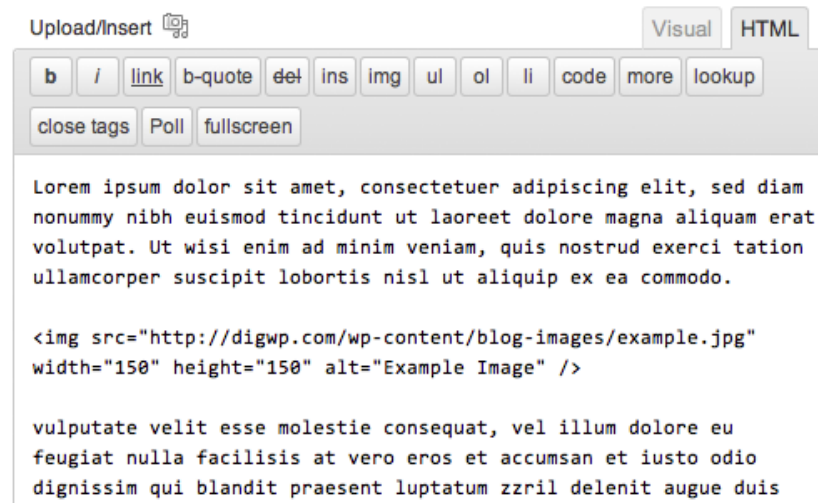
When the error values are detected, the alternate, fallback message is displayed; otherwise, the function displays the correct count value.

Plus, as an added bonus, the function covers your bases by outputting the fallback message in the event that the Feed Count plugin itself should fail. This method ensures proper subscriber count display without relying on the availability of JavaScript, the accuracy of FeedBurner data, or even the functionality of the Feed Count plugin!

More information at Perishable Press: <http://digwp.com/u/89>

In your theme, the CSS might be changing this default inline behavior. It's possible that images are styled as block-level elements in your stylesheet, which forces line-breaks before and after. Or, the images could have special classes applied to them which float them left or right, causing surrounding text to wrap. But unfortunately, your CSS doesn't mean anything when the Post is being read through an RSS reader. Those images will revert to their old inline-level selves.

One classic way of avoiding this problem is making sure you have an extra line break above and below the image tag when creating/editing your Post. Here is a screenshot demonstrating this technique:



The empty lines above and below the tag will ensure that the top paragraph, image, and bottom paragraph will all be automatically wrapped in <p> tags (without implicitly using them).

One of the odd things that WordPress does is automatically apply `<br />` tags into posts where a single return is used, and wrapping `<p></p>` tags to blocks of content separated by double line-breaks.

The end result is that this image tag separated from

BlackBerry App World More Expensive Than Android App Stores (Report)

by Robin Wauters on October 6, 2009

43



We've covered a couple of [Distimo](#) reports because they provide us with some valuable data. In an in-depth analysis of publicly available data, Distimo has added RIM's [BlackBerry App World](#) to the mix, which gives us even more data points to analyze app pricing and whatnot.

In [the latest report](#) (September 2009), Distimo notes that software programs for Blackberry devices are considerably more expensive than comparable apps for competing devices/platforms.

In fact, the average price for apps is *more than three times higher* than apps in the App Store and Google's Android Market, which is sort of unbelievable. There's not a single category where an app is lower than its equivalent on the latter two application storefronts. For serious, business-related tools are definitely much more expensive. (chart after)

The nicely floated image in this TechCrunch article reverts to a non-floated inline image in the RSS feed and is rather awkward.

TechCrunch

TechCrunch is a group-edited blog that profiles the companies, events defining and transforming the new web.

[BlackBerry App World More Expensive Than iPhone App Stores \(Report\)](#)

October 6, 2009 7:33 AM



We've covered a couple of [Distimo](#) reports in the past because they provide us with some valuable data. In an in-depth analysis of publicly available data. Now the company has added RIM's [BlackBerry App World](#) store to the fray, which gives us even more data points to analyze app pricing and whatnot. In [the latest report](#) (September 2009), Distimo notes that software programs for Blackberry devices are considerably more expensive than comparable apps for competing devices/platforms. In fact, the average price for apps is *higher* than apps in the App Store and Android Market, which is sort of unbelievable. There's not a single category where the price is lower than its equivalent on the latter two application storefronts. For serious, business-related tools are definitely much more expensive. (chart after)

other content by double line-breaks on either side will now be wrapped in paragraph tags, which are by default block-level elements. This effectively makes your image a block level element, and will prevent any weird text wrapping.

Another option is to use inline styling on the images, for example:

```

```

This works on the site *and* through RSS feeds, with the major shortcoming that should you ever need to update image styles, you'll have to return to this post and do it manually (rather than with a sweepingly simple CSS change).

6.6.3 Adding a Custom Feed Image

Out of the box, WordPress feeds are very plain. Apart from any images that your posts or comments may include, your feeds are delivered without any logos or icons. A great way to enhance your site's brand is to add a custom image to your feeds. Here's how to do it:

1. Add the following code to your theme's `functions.php` file:

```
// custom feed image
function mwp_feedImage() {
    echo "<image>
        <title>Digging Into WordPress</title>
        <url>http://digwp.com/images/feed-icon.png</url>
        <link>http://digwp.com/</link>
        <width>150</width>
        <height>150</height>
        <description>Digging Into WordPress</description>
    </image>";
}

add_action('rss2_head', 'mwp_feedImage');
```


Inline Stylin' for RSS

It is a classic CSS sin to use inline styling. It completely defeats the purpose of using CSS, which is to separate content from design. If we wanted to call special attention to a box on the page, it would be smart for us to do something like:

```
<div class="callout">
  ... content ...
</div>
```

When we publish our content with RSS as WordPress does, people won't be viewing our content on our website, but in whatever they use to read RSS content. This could literally be anywhere, but a classic example would be a reader reading through Google Reader.

In Google Reader, our special class of "callout" has no meaning whatsoever. On our own website, callout might mean it has

some extra padding and a yellow background, but in Google Reader, it will have no effect at all.

If we want to make that extra padding and yellow background come through over RSS, we can force it by using inline styling.

```
<div style="padding: 15px; background: #fff3c4;
  margin: 0 0 10px 0;">
  ... content ...
</div>
```

Just remember that inline styling is generally regarded as evil for a reason... if you ever needed to change the styling, you'd have to go back into every single Post that used it, instead of being able to just change the CSS for "callout."

 **Edit Post**

New Course: Yiddish-American Popular

Permalink: <http://jewishstudies.wisc.edu/new-course-yid...opular-culture/>

Upload/Insert    

b **i** **link** **b-quote** **del** **ins** **img** **ul** **ol** **li** **code** **mc**

close tags **Image Browser**

```
<div style="padding: 15px; background: #fff3c4; margin: 0 0 10px 0">
<strong>Course Description</strong>
East European Jews who arrived in America at the end of the 19th c
a nascent and vibrant popular culture came in contact with a simult
emerging -- and equally influential -- American equivalent. Using
media such as 78 rpm recordings, cartoons, films and radio broadc
class will examine how these two cultures interacted, influenced and
each other.
```

Inline styling in the Post

Result in RSS reader

★ **New Course: Yiddish-American Popular Culture** 10:02 AM (24 minutes ago)

from **Jewish Studies** by [chriscoyle](#)



As co-sponsors, the Center would like to announce the addition of an exciting interdisciplinary arts course offering next spring. The UW-Madison Arts Institute sponsors an interdisciplinary arts residency each semester, giving students the unique opportunity to work closely with a professional artist or arts scholar. These courses are for-credit and offered through multiple departments.

Our spring 2009 artist in residence is Henry Sapoznik, an award winning author, record, and radio producer and performer of traditional Yiddish and American music. He will be teaching a 3-credit course on "Yiddish-American Popular Culture, 1890-1950," on Tuesdays and Thursdays 12:00-3:50PM in 1227 Engineering Hall. The course has no prerequisites, and will be available through the following departments: Jewish Studies 510-450, Art 448-004, Music 497-001, Folklore 530-004, Theatre and Drama 469-001

Course Description

East European Jews who arrived in America at the end of the 19th century with a nascent and vibrant popular culture came in contact with a simultaneously emerging -- and equally influential -- American equivalent. Using period media such as 78 rpm recordings, cartoons, films and radio broadcasts the class will examine how these two cultures interacted, influenced and portrayed each other.



Henry Sapoznik is an award winning author, record and radio producer and performer of traditional Yiddish and American music. A pioneering scholar and performer of klezmer music, Sapoznik founded both the Max and Frieda Weinstein Archives of Recorded Sound at the YIVO Institute for Jewish Research (1982) and KlezKamp: The Yiddish Folk Arts Program (1985). He is the recipient of the Peabody Award for Excellence in Broadcast Journalism (2002) for the 10 part series "The Yiddish Radio project" and the 2000 ASCAP Deems Taylor Award for Excellence in Music Scholarship for his book

2. Edit each of the different XML elements to reflect your site's information.
3. That's it! Save, upload and check out your new custom-branded RSS feed!

Add custom image to Atom feed

1. Add this code to your functions.php:

```
function mwp_atomImage() {  
    echo "  
        <icon>http://digwp.com/images/favicon.ico</icon>  
        <logo>http://digwp.com/images/feed-icon.png</logo>  
    ";  
}  
add_action('atom_head', 'mwp_atomImage');
```

2. Edit each of the different XML elements to reflect your site's information.
3. All done! Save, upload, and check out your new custom-branded Atom feed!

Alternately, FeedBurner users may add a feed image to any of their feeds by using the "Feed Image Burner" feature available in their account admin area.

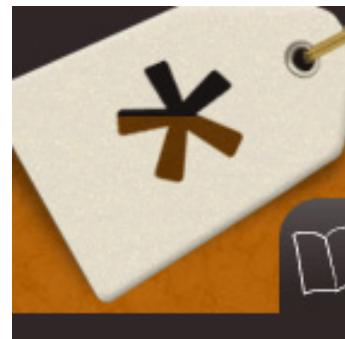
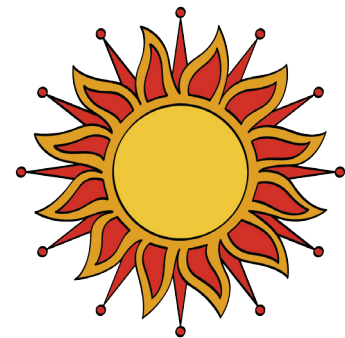
6.6.4 Include Comments in Feeds

While WordPress excludes post comments from feeds by default, certain sites may benefit from including them. The easiest way to do so is to mashup a quick feed pipe at the free Yahoo! Pipes service <http://digwp.com/u/169>. Sounds complicated, but it's really simple. Here's how to do it in ten easy steps:

1. Sign into Yahoo! Pipes with your Yahoo! ID.
2. From the "Sources" menu in the left sidebar, drag an instance of the "Fetch Feed" badge to the working area.

Branding is Key

*Presenting a strong brand identity
is a key part of your site's success.
Customizing your feeds with your logo is
an excellent branding opportunity.*



Merging Feeds

As seen in this section, Yahoo! Pipes is an excellent way to merge multiple feeds into a single feed, but it's not the only mashup service that does so. Another good service is RSS Mix, which you can check out at <http://digwp.com/u/172>

3. Add your Post Feed and Comments Feed to the "Fetch Feed" badge. Click the plus "+" sign to get another input field.
4. From the "Operators" menu in the left sidebar, drag an instance of the "Sort" badge to the working area.
5. In the "Sort" badge, select the "item.pubDate" option and sort in descending order.
6. You should have three badges at this point: "Fetch Feed," "Sort," and "Pipe Output," which was added to the working area automatically.

Fetch Feed

- + URL
- ? urner.com/DiggingIntoWordpress
- http://digwp.com/comments/feed/

Sort

- + Sort by
- item.pubDate in ascending order

Pipe Output

Time taken: 0.132856s [Refresh](#)

- ▶ WordPress Debug Theme
- ▶ Ideas for Plugins I'm Too Lazy To Write
- ▶ Comment on New Poll: How Many Plugins do You Use?
- ▶ Comment on New Poll: How Many Plugins do You Use?
- ▶ Comment on Plugin Wishes Come True by Latz
- ▶ Poll Results: Goodbye Dolly

Annotations:

- Add your Posts Feed URL and Comments Feed URL to the Fetch Feed Badge
- Specify sort order as "item.pubDate" in the Sort Badge
- Click on the Pipe Output Badge to see the output results

7. Connect the three badges by connecting the circles on the upper and lower edges (see screenshot on previous page).
8. Click the “Refresh” button in the debugger window below the working area to see the output of the piped feed.
9. To save your feed, click “Save” in the top-right area of the screen and name your new pipe. After saving is complete, click “Run Pipe” at the top of the screen. A new window will then open with configuration options and so forth.
10. Finally, click “Publish” to make it official. The URL of your RSS-formatted feed is available by clicking the “Get as RSS” link next to the feed icon.

That’s all there is to it! You now have a shiny new feed that contains your posts and comments in chronological order.

Of course, there is much more you can do with the incredibly awesome Yahoo! Pipes service, so feel free to experiment and have some fun. By the way, you can check out the DiW Posts and Comments feed used for this tutorial at <http://digwp.com/u/170>.

6.6.5 Creating Custom Feeds

There are many situations in which the default WordPress feeds simply won’t do. For example, you may have a category or group of categories that you would like to exclude from your main feed. Fortunately, WordPress provides plenty of special feed parameters that enable you to carefully craft the perfect custom feed.

Exclude categories, authors, and more from feeds

Here is the general format for excluding content such as specific categories, authors and so on:

DiW Posts and Comments

Click to add description

Pipe Web Address: <http://pipes.yahoo.com/pipes/pip>

☆ Edit Source Delete Publish Clone

Get as a Badge MY YAHOO!

List

WordPress Debug Theme

Sometimes you need to see what’s wrong with a hacks around for a while to do that, but finally sta quite simple for now, as it only does a few things

Ideas for Plugins I’m Too Lazy To Write

... or slightly more accurately, that I don’t know h to have a title and a subtitle for every single Post alter the Admin screen for writing posts to insert

Comment on New Poll: How Many Plugins

6 and 7 plugins respectively. Sometimes I dread

Comment on New Poll: How Many Plugins

71 plugins?!?! Wow. I have trouble keeping track

Comment on Plugin Wishes Come True b

Another issue that came to my mind is that the s templates for feeds, it has to be done using a filter (http://x.elektroelch.de/f). I will translate it tomorrow ever can contact him).

`http://digwp.com/feed?cat=-n`

Just change “n” to the category ID that you want to exclude. After doing so, your RSS feed will include all posts *except* for those from the specified category. Likewise, we can exclude **multiple categories** like so:

`http://digwp.com/feed?cat=-x,-y,-z`

Here we would replace “x”, “y”, and “z” with the three categories of our choice. Similarly, additional categories may be added by simply adding an additional “,-n” to the end of the query string.

Subscribe to Your Own!

During the process of setting up a new feed, it is always a good idea to grab a subscription for yourself. Adding your feed to a few different feed readers is a great way to keep an eye on your feed and ensure that it is looking good.

Likewise, we may exclude specific authors, dates, and much more by simply replacing the “cat” parameter and its corresponding value(s) with any of these:

author=-n - excludes all posts posted from the Author with ID of “n”

year=-2009 - excludes all posts published in the specified four-digit year

monthnum=-11 - excludes all posts published in the specified month number

day=-22 - excludes all posts published on the specified day

paged=-33 - excludes all posts contained on the specified archive page

s=-wordpress - excludes all posts containing the search term “wordpress”

category_name=-wordpress - excludes all posts in the “wordpress” category

You may also combine these parameters to generate custom-fit WordPress feeds. Here are some examples:

`http://digwp.com/feed?cat=-11&s=-apple`

This would generate a feed that excludes category #11 as well as any posts containing the term “apple.”

`http://digwp.com/feed?cat=-11&year=-2008&author=-3`

This would generate a feed that excludes category #11, excludes posts from 2008, and excludes posts written by author #3. As you can see, the pattern is simple:

```
.../feed?parameter01=value01&parameter02=value02&parameter03=value03
```

As you can see, each parameter-value pair is concatenated via the “&” symbol. Hopefully you won’t need anything that specific, but it’s there if needed.

Excluding categories via functions.php

Apart from URL-modification or using a plugin, it is also possible to exclude categories and other parameters from your feeds by placing the following script in your theme’s functions.php file:

```
// exclude categories from feed
function excludeCatsfromFeed($query) {
    if ($query->is_feed) {
        $query->set('cat','-x,-y,-z'); // excluded categories
    }
    return $query;
}
add_filter('pre_get_posts','excludeCatsfromFeed');
```

Once in place, go ahead and edit the “x”, “y”, and “z” to match the IDs of the categories you wish to exclude. You may also add or remove categories, or even exclude other items by replacing the “cat” parameter in the third line with any of the available query parameters discussed previously.

This function makes it easy to deliver customized feeds in an extremely flexible fashion. Once in place, any posts belonging to the specified categories will no longer be displayed.

Excluding the Easy Way

If modifying your feed URLs is not your preferred way of excluding categories from feeds, you may want to check out the Ultimate Category Excluder plugin from Michael Clark:

<http://digwp.com/u/171>

Custom feed including only specific categories, authors, etc.

Including specific content in your feeds is as simple as removing the minus sign "-" from the various formats presented above. All of the same parameters and their corresponding values may be used. Here are some copy/paste code examples to clarify the technique:

Include posts from categories "x", "y", and "z" only

```
http://digwp.com/feed?cat=x+y+z
```

Includes the term "apple" from posts in category 11 only

```
http://digwp.com/feed?cat=11&s=apple
```

Include posts from category 11, the year 2008, and from author 3

```
http://digwp.com/feed?cat=11&year=2008&author=3
```

Generate a feed consisting of multiple tags

```
http://digwp.com/?feed=rss&tag=tag1,tag2,tag3
```

...and so on. Many combinations of these parameters are possible, enabling you to configure the perfect feed for your readers.

Including categories via functions.php

For those of you not comfortable with the idea of messing around with your feed's URL, it is also possible to include categories and other parameters in your feeds by placing the following script in your theme's `functions.php` file:

```
// include categories in feed
function includedCatsFeed($query) {
    if ($query->is_feed) {
        $query->set('cat','x,y,z'); // included categories
    }
    return $query;
}
add_filter('pre_get_posts','includedCatsFeed');
```

Once in place, go ahead and edit the “x”, “y”, and “z” to match the IDs of the categories you wish to include. You may also add or remove categories, or even exclude other items by replacing the “cat” parameter in the third line with any of the available query parameters discussed previously.

6.6.6 More Feed Customization Tricks

Just in case all of those customization tricks weren’t enough, here are a few more that you may find helpful:

Alphabetize feed posts in ascending order by title

<http://digwp.com/feed?orderby=title&order=asc>

Feed for a specific page

<http://digwp.com/page/feed>

Alternate format for specific page

<http://digwp.com/feed?pagename=page>

One for the Road..

Alternate feed format for a specific page:

http://digwp.com/feed?page_id=n

6.6.7 Styling Feeds

Although certain “purists” will tell you to never style your feeds, many people choose to do so to improve upon the ultra-plain default style that most feed readers provide. By adding a few custom CSS styles to your feeds, it is possible to improve your site’s image, increase brand awareness, and facilitate usability.

The easiest way to add some CSS pizzazz to your feed is to add inline styles via the impressive Feedstyler plugin. The Feedstyler plugin is perfect for WordPress users who are comfortable with CSS and wish to add custom styles to their WordPress feeds. Feedstyler retains class and ID attributes in your markup while enabling you to specify alternate feed styles. In addition to keeping your site and feed styles separate, Feedstyler also accepts nested CSS declarations, accepts multiple class names, and works on all feed readers that do not remove inline styles from the XML markup. The plugin has certain limitations in terms of what it can and can not do, but overall it’s a great way to style your feeds. Get it at: <http://digwp.com/u/173>

6.6.8 Removing the WordPress Version Number

By default, WordPress provides your WordPress version in your feeds. If you peek under the hood and look at the source code for any of your feeds, near the top of the file, you’ll see that WordPress includes its version number via <generator> tags:

```
<generator>http://wordpress.org/?v=2.7</generator>
```

While this is useful information to legitimate sources, it serves as a slight security risk by enabling attackers to target any specific security holes that may be present in your particular version of WordPress. Thus, it is a good idea to remove this tag from your feeds and your theme’s header file as well. To do so, include the following simple function in your theme’s functions.php file:

```
function killVersion() { return ''; }  
add_filter('the_generator', 'killVersion');
```

This will replace your WordPress version with literally nothing whenever its generating function is called. To also remove the version information in the header, we take a slightly different approach with this line in the `functions.php` file:

```
<?php remove_action('wp_head', 'wp_generator'); ?>
```

This simple yet effective code prevents the `wp_generator` function from appearing in your theme's `header.php` file.

6.6.9 Disable and Redirect Unwanted Feed Formats

As discussed, WordPress generates your various feeds in a variety of formats, including RDF, RSS 2.0, and Atom. This is great if you're using them, but if not, you may want to disable and redirect those alternate formats for the sake of clarity, uniformity, and statistical analysis.

Using the following code in your theme's `functions.php` file, any requests for the specified feeds will be redirected to the URL of your choice:

```
function redirectFeed() {  
    wp_die();  
    header("Location: http://digwp.com/feed/");  
    exit();  
}
```

Edit the URL in the third line with something useful, like your home page, a "Subscribe" page, or even your active Posts Feed. Once this is done, specify which alternate feed formats should be redirected by including any of the following lines directly beneath the previous function:

```
add_action('do_feed', 'redirectFeed', 1);    // all feeds  
add_action('do_feed_rdf', 'redirectFeed', 1); // RDF (RSS 0.92) feed
```


Clean Up Your Head

You can remove any/all of the content that is automatically displayed in your WordPress <head> section by applying any/all of the following directives to your theme's functions.php file:

```
// clean up unwanted items from <head>
remove_action('wp_head', 'rsd_link');
remove_action('wp_head', 'wp_generator');
remove_action('wp_head', 'feed_links', 2);
remove_action('wp_head', 'index_rel_link');
remove_action('wp_head', 'wlwmanifest_link');
remove_action('wp_head', 'feed_links_extra', 3);
remove_action('wp_head', 'start_post_rel_link', 10, 0 );
remove_action('wp_head', 'parent_post_rel_link', 10, 0);
remove_action('wp_head', 'adjacent_posts_rel_link_wp_head', 10, 0);
```

This latest batch works with WP 3.3+, but chances are high that some or all of these directives will stop working at some point going forward. For some reason, the functionality we're dealing with here changes almost constantly.

```
add_action('do_feed_rss', 'redirectFeed', 1); // RSS 1.0 feed
add_action('do_feed_rss2', 'redirectFeed', 1); // RSS 2.0 feed
add_action('do_feed_atom', 'redirectFeed', 1); // Atom feed
```

6.6.10 Insert Custom Content into Feeds

One of the easiest things that you can do with WordPress is add custom content to your posts. The custom content can be anything – text, markup, or even scripts – and may be set to appear in blog posts, feed posts, or both. You can easily set content to display before or after your post content, or in both places, if desired.

Good examples of how this sort of functionality is used include the addition of copyright notices displayed in the feed footer, and advertisements appearing before and/or after each feed item. The possibilities are endless.

To add some custom content to your own feed, place the following code snippet to your active theme's `functions.php` file:

```
function insertContent($content) {  
    $content = $content . '<p>Place your custom content here!</p>';  
    return $content;  
}  
add_filter('the_excerpt_rss', 'insertContent');  
add_filter('the_content_rss', 'insertContent');
```

Simply edit the second line of the function to include the desired code, markup, or text content. This function works by appending the specified custom content to the post content. Then, to ensure that the added content is only included within your feeds, we are using WordPress' `add_filter()` function to execute the code only for full and excerpted feed content. Further, by slightly tweaking the code, you can insert custom content to appear *before* your regular feed content:

```
function insertContent($content) {  
    $content = '<p>Place your custom content here!</p>' . $content;  
    return $content;  
}  
add_filter('the_excerpt_rss', 'insertContent');  
add_filter('the_content_rss', 'insertContent');
```

Easy Ad Inserts

Using the code in this section, inserting advertisements into your feeds is a snap.

And, as you can imagine, we can follow this line of thinking to include custom content **both before and after** your feed post content:

```
function insertContent($content) {  
    $content = '  
        <p>This content appears before the feed post.</p>' . $content . '  
        <p>This content appears after the feed post.</p>';
```

```
        return $content;
    }
    add_filter('the_excerpt_rss', 'insertContent');
    add_filter('the_content_rss', 'insertContent');
```

6.6.11 Importing and Displaying External Feeds

This idea takes the blogroll one step further. Feed technology makes it easy to import and display virtually any feed anywhere on your site. For example, you may want to upgrade that tired old blogroll with a sidebar section full of your friends' recent posts. Here are some choice plugins to make it happen.

Import without a plugin

To display external feeds without using a plugin, we can use WordPress' built-in RSS import functionality to get the job done. See chapter 4.4.9 for more information.

- **Parteibuch Aggregator** <http://digwp.com/u/177>

Displays aggregated feeds in WordPress pages and sidebars. Share the feeds you read with the visitors of your blog. It is designed to be able to cope with hundreds of feeds, thousands of aggregated feed items and a lot of traffic.

- **InlineRSS WordPress Plugin** <http://digwp.com/u/178>

Enables the importation and display of virtually any RSS feed. This plugin will also display the titles of multimedia content and provide links to it.

- **FeedWordPress** <http://digwp.com/u/179>

A completely customizable feed-import plugin that works with your blogroll to import and display your favorite feeds.

6.6.12 Buffer Period After Posting

Thanks to a tip from WPengineer <http://digwp.com/u/181>, we now enjoy a nice way to provide a little time buffer between the moment of posting and the moment the content appears in our feeds. Before learning about this method, there was no way to prevent mistakes from immediately appearing in feeds that were noticed *after* hitting the "Publish" button. Once the mistake is in the feed, it is beamed out all over the place, leaving readers confused and you looking sloppy. Not cool man.

Awesome Plugins for Custom Feed Content

Looking for a great way to spice up your blog feeds? Here are some of the best “feed-footer” plugins available for enhancing your feeds with custom content, links, and much more!

- **Better Feed WordPress Plugin** <http://digwp.com/u/174>

A great plugin that enables flexible customization of your feed footer. Add a copyright notice, custom markup, “read-more” links, number of responses, related posts, and much more. We use this plugin at the Digging Into WordPress companion site <http://digwp.com/> to help promote the book. Highly recommended.

- **RSS Footer** <http://digwp.com/u/175>

This very simple plugin from WordPress guru Joost de Valk enables you to add an extra line of content to articles in your feed, defaulting to “Post from: ” and then a link back to your blog, with your blog’s name as its anchor text, which you can edit from the plugin’s Admin Options page.

- **Authenticate** <http://digwp.com/u/176>

Authenticate is a highly flexible content-addition plugin that enables universal and/or targeted inclusion of custom content for both feeds and posts. Ideal for adding copyright information, distribution policy, thank-you messages, custom links, special offers & more. Custom content may be added to any location within posts or feeds – before, after, or even within post content.

Fortunately, we can avoid this embarrassing problem by placing the following code into our theme’s functions.php file:

```
// buffer period after posting
function publish_later_on_feed($where) {
    global $wpdb;
    if (is_feed()) {
        $now = gmdate('Y-m-d H:i:s'); // timestamp in WP-format
        $wait = '5'; // integer value for wait + device
        $device = 'MINUTE'; // MINUTE, HOUR, DAY, WEEK, MONTH, YEAR
        $where .= " AND TIMESTAMPDIFF($device, $wpdb->posts.post_date_gmt, '$now') > $wait ";
    }
    return $where;
}
add_filter('posts_where', 'publish_later_on_feed');
```

Publish Later

This function takes advantage of the \$where variable by comparing the query value to the SQL function timestampdiff(). After the specified time interval has elapsed, the post will be published to your feed. Note that this function does not affect normal post publishing on your site. Edit the \$wait variable in the function to customize the buffer period.

As is, this code will provide a 5-minute buffer period for you to catch any errors and make any last-minute changes. Five minutes later, your feed will be generated as normal. Within the code there is info for changing the buffer period.

6.6.13 Protecting Feed Content

The flexibility and mobility of feeds also makes them prone to abuse. When normal visitors and subscribers get ahold of your feed, they usually add it to their feed reader of choice and enjoy the convenience of automatically updated, localized, and consolidated content. Conversely, when criminals get ahold of your feeds, they are free to steal and abuse them. As mentioned earlier, so-called “scraper” sites steal feeds and display them on automated blogs in order to make money from cheap advertisements. Yes it sucks, but there are some things you can do to stop it.

Feed Footer Plugins

A great way to deter scrapers while customizing the functionality of your feeds is to use one of the various “feed-footer” plugins. In general, these plugins work automatically in the background to add specific information to your feed posts. Many types of custom information may be added, including copyright statements, digital fingerprints, post links, author information, and much more.

- **Better Feed WordPress Plugin** <http://digwp.com/u/174>
Ozh’s Better Feed plugin enables you to customize your feeds with copyright info, “Read-More” links, comment counts, related posts, and tons more.
- **Owen Winkler’s AntiLeech WordPress Plugin** <http://digwp.com/u/182>
Deters and protects against content thieves by including a small “AntiLeech” graphic in your feed’s output. The graphic transmits user-agent data back to your site for easy blocking. Great feed protection with tons of features.
- **Digital Fingerprint Plugin** <http://digwp.com/u/183>
Places a digital fingerprint, customized and tailored by each individual user of this plugin, into blog posts. Once embedded in your post, the plugin allows you to quickly and easily search the Web for any violations.

Scrapers Suck!

To learn more about stopping content-scrapers, check out:

How to Deal with Content Scrapers:

<http://digwp.com/u/595>

How to Protect Your Site Against Content Thieves:

<http://digwp.com/u/596>

- **Angsuman's Feed Copyrighter Plugin** <http://digwp.com/u/184>

Adds a simple, unobtrusive copyright message to your feed items, thereby acting as a deterrent to unscrupulous content thieves. Quick, easy & effective.

Enable Partial Feeds

Perhaps the best way to prevent content theft is to enable the partial feed format. As discussed earlier in this chapter, enabling partial feeds discourages scrapers and other thieves by rendering your feeds useless to automated processing. Populating their scraper pages with partial posts is not as lucrative as it is with full posts. Sure, some of your legitimate users may complain a little, but this is a great way to defend against stolen content.

Link Back to Your Site

Another effective method of preventing stolen content is to include links back to your site within your posts. Many bloggers do this automatically as they write their content because it helps readers discover related content, and also facilitates the distribution of valuable link equity into other areas of your site.

Best of all, when scraper sites display your intact feeds on their sites, all of those links that you have added to your content will be pointing right back at your site. This acts as a strong signal to both visitors and search engines concerning the original source of the material.

Take Advantage

It never ceases to amaze us to see how many thieves scrape the content of popular sites like Smashing Magazine, Noupe, NetTuts, and the like. If one of your posts is fortunate enough to get a link in a post from one of these sites, don't be surprised to see gazillions of pingbacks back to your site. Sure, the linkbacks are coming from scraper sites, but it's still a good reason to include plenty of internal links in your posts.

6.7.1 Validating Feeds

In this chapter, we have learned lots of essential information concerning WordPress feeds. Now that we know what they are, where they are, and how to customize our feeds, let's wrap things up by looking at an important practice when delivering syndicated content to your readers: *validating your feeds*.

With so many different feed readers available to choose from, it is vital to ensure that your feeds are being generated and delivered as intended. The last thing you

want is to spend your time and effort writing content that is not being displayed properly because of errors. Fortunately, there are plenty of free online Feed Validators that will help you check your feeds for accuracy.

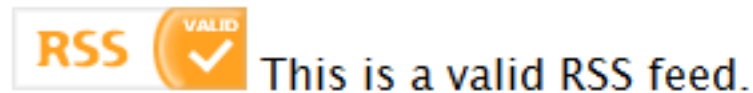
Free online feed validators

In the process of configuring your WordPress feeds, it is a good idea to check your feeds for proper validation with any of the available online feed validation services. Here a few to choose from:

- <http://feedvalidator.org/>
- <http://validator.w3.org/feed/>
- <http://www.atomenabled.org/feedvalidator/>

And of course, there are many other good validators as well. To validate your feed, visit one of these sites and enter the URL of the feed you would like to check. If all goes well, you should see something like this:

Congratulations!



Moving on...

In this chapter, we have learned a great deal about setting up, using, and customizing WordPress feeds. In the next chapter, we explore the exciting world of WordPress comments and dig into different ways of customizing, enhancing and improving the functionality of your WordPress comment system.

Chris' personal blog (next page) uses some art direction to apply styling appropriate to the content of individual Posts.

<http://chriscoyier.net/>

01/30/2010

Winter is the

Dry, crisp air. The color palette of the world wear comfortable clothes all layered up – and to. Everything seems quieter and more relaxed.

Really.

LEAVE A REPLY

Logged in as Chris Coyier. Log out »

Submit Comment

BE MY FWEND • PEOPLE I ENJOY

The ones I actually use:



POWERED BY

100% WordPress baby, with

Jesse Lynch

Jeff Campana

Leah Shea

David Walsh

Richard Felix Jr.

10/26/2009

List of Post Apocalyptic Movies

I've always been into post apocalyptic stuff. I said in my recent review of The Road that I think it's because it's such an interesting thing to fantasize about. Kind of like how you can daydream forever about what you would do if you won the lottery.

I thought it would be fun to put a list together of real post apocalyptic movies. This isn't going to be a comprehensive list. There are going to be some rules as well as somewhat arbitrary moderation by me for what makes the list. This is what I'm thinking:

- **The movie takes place in the future.** That is, a time beyond when the movie was made. So in this case, post-apocalyptic means post-apocalyptic EARTH and the HUMAN RACE.
- **But not just "the future".** There are some movies that take place in a Dystopian (Brazil) or Utopian (Time Machine, kinda) future that sort of imply that the reason was the apocalypse. But these movies aren't really about the apocalypse, they are about the future.
- **The reason can be ridiculous.** Zombies, aliens, giant space bugs from mars... bring it on.
- **But not just an excuse.** Action movies are fine, and B movies are fine, but movies that are just total pieces of shit don't get to go on the list.
- **There is some kind of vision.** The bottom line is that I'm interested to see an interpretation of the apocalypse from someone who really thought about it and had some ideas about it.
- **It's a movie.** There are some absolutely fantastic post-apocalyptic things that have been on TV: Battlestar Galactica, Jericho, Kings... But this list is for movies.
- **It could be pre-apocalyptic** as long as it's also at least partly post-apocalyptic (The Day After Tomorrow).

The Day the Earth Caught Fire

1961



Panic in Year Zero!

1962



The Earth Dies Screaming

1962



The Last Man on Earth

1964



Omega Man

Charlton Heston has this badass shelter where he keeps out the night dwelling vampire zombies ("The Family"). He meets some resistance type folks, hooks up with a hot black chick, and

1971



10/26/2009

Grooveshark on the iPhone

I met some of the guys from [Grooveshark](#) earlier this year (whatup Vishal and John) when I was down in Florida for the Front End Design Conference. We had had a good time. We hung out and talked shop at the conference, ate alligator tacos, and saw Black Mosh Super Rainbow.

Never heard of Grooveshark? Well you should. It's amazing. You can listen to any song in the world for free from your computer. With a free account, you can make playlists, favorite songs, upload your own songs, and be friends with other Grooveshark users. It's like iTunes, except your library is about 1,000 times bigger and it's free. Ammm, it has radio mode which learns your likes and dislikes so it's got the Pandora angle covered.

Now Grooveshark is going to have an iPhone app! It's pretty full featured, meaning that all of your favorites and playlists from your account come with you to the app. As well as of course the ability to search all the songs in the world and listen to them.

NOTE: The Grooveshark app is not yet approved by Apple. Hopefully, it will be soon.

FAVORITE FEATURES?

You know how on scroll-wheel iPods you can hold down the button when you find a song you want to listen to to add it to the On-The-Go playlist? This allowed you to be an in-car DJ without having a computer or plan ahead. Well I don't have a scroll-wheel iPod anymore, I have an iPhone. There is no way on the iPhone to make an On-The-Go playlist.* With the Grooveshark iPhone app, you can be playing tunes, browsing around, and add songs on the fly to your song queue. Nice.

*OK! guess there is a way to do On-The-Go on the iPhone, I just didn't know about it.

SCREENS



COMMENTS



Russell Heimlich says:

Nice that looks awesome. I'm a huge Grooveshark fan and I hope everyone finds out about them. Can't wait to see the Android app as that will be my next phone for sure.



Chris Spooner says:

Looking forward to checking this out, strangely enough it was just yesterday I searched the App store for Grooveshark, not realising an app was in the pipeline.

I'm also digging the cool post design you've put together!



Kevin M says:

Wow, love the splash screen for the app! I've been using Grooveshark for months now, so I'm pleased to see they're extending the service to the iPhone.

But, being an Android user myself I am quietly hoping that the Gehack may find its way into my market.

Think like a wise man but communicate
in the language of the people.

– **WILLIAM BUTLER YEATS**

7

Working with Comments

7.1.1 Optimizing the WordPress Comments Area

One of the most exciting things about WordPress is its highly flexible commenting system. There is so much that you can do to improve and optimize your site's comments area that the subject is worthy of an entire chapter of its own. Out of the box, WordPress provides a robust commenting and response system that enables any site to effectively integrate and expediently handle even large volumes of comments, pingbacks, and trackbacks. As good as the default comment system is, however, there is a great deal more that may be done to sharpen the appearance, functionality, and security of your WordPress-powered comment area.

7.1.2 Welcome to the WordPress Comments Area

The heart of the blogosphere is driven by its interactivity. Publishing content with dynamic technologies like WordPress make it easy for millions of people to express themselves, share news, and produce many other types of content. But the interactivity enabled by WordPress' powerful commenting and response system connects users, authors, and machines to similar stories, related posts, and associated information around the web. The WordPress comments area is powerful, and with the help of some custom techniques, a few plugins, and a bit of knowledge, is easily transformed into a highly interactive content response system. This is fun, exciting stuff that will do wonders for the usability, likability, and popularity of your site.

Pingbacks

Pingbacks require an actual link to your post that comes from some other site. They link to you, and if your theme is set up for it, your site automatically links back to them.

Trackbacks

Trackbacks are very similar to pingbacks, but don't require that the other site actually link to your post. All they need to do is tell WordPress to send a trackback to your post via the Post Edit screen. If trackbacks are enabled on your site, your post will then link back to the other site.

Overly Confusing

The end result of a pingback or trackback is the same thing, a comment on your site linking to another site that referenced you. The difference is in the technology of how it works.

7.1.3 About the WordPress Comment System

Whenever you post an article on your site, WordPress makes it possible and even easy for readers to respond to your content in various ways. Essentially, there are three different ways for your readers to respond to your post. They may either leave a comment directly, set a trackback from an article on their site, or else simply write about your post on their blog. As the response process unfolds on your site, there are many different aspects involved. Here are the different parts of a typical response area:

7.1.4 Comments, Pingbacks, and Trackbacks

In WordPress, three different comment types are possible. The most familiar type of comment goes like this: a visitor arrives at your site, reads your post, and has something to say about it. They find a form, usually located after the article. They type in their name, email address, website URL, and then submit their comment. That's just called a "comment," and it will show up along with any other comments in your site's comment area.

The second type of comment is called a "pingback," and is actually more of a response than a comment. Let's say that you write a post about bacon and the Food Network decides to tell their readers about your sizzling article. They publish a post on their site that includes a link to your article so their readers can go there and read about bacon. In this scenario, WordPress would detect that incoming link from the Food Network and reciprocate by automatically posting a link *back* to the Food Network site. This return link usually displays the title of the linking post as its anchor text.

The third type of response that WordPress enables is called a "trackback," which is also much more of a response than an actual comment. To illustrate, let's say that the Food Network mentions your bacon article but does not actually include a link to your site. If they create a trackback to your post (via the Post Edit screen in the WordPress Admin), your post will link back to the Food Network article. Aside from the differing underlying technology, pingbacks and trackbacks function nearly identically. The main difference is that an actual link is not required for trackbacks.

There is a great deal of fancy technology behind the functionality of pingbacks and trackbacks, the scope of which is beyond this book. What is most important to know for now is the different types of responses and how they are made:

- **Comments** - left directly on your site by your visitors
- **Pingbacks** - a link from one blog post that is returned by another
- **Trackbacks** - a response (not necessarily a link) from one blog post that is linked back from another

Comment or Response?

Technically, pingbacks and trackbacks are not comments. Yet many sites still lump together all types of responses as “comments,” as in, “This post has 15 comments.” More accurately, we would say, “This post has 15 responses.”

7.1.5 Anatomy of the WordPress Comment Area

There are four fields and a button that make up your garden-variety WordPress comment form: Name, Email, Website, Comment, and Submit. Seems easy enough, but the HTML markup for such forms is some of the most verbose and strange HTML imaginable. Throw in a host of special WordPress functions to enhance the comment form’s functionality, and you have yourself a fairly complex little chunk of code.

The WordPress gods were aware of this complexity, and have taken steps to make things as easy as possible. For example, all of the code required for the WordPress comment system is contained within a single theme template file named “comments.php.” Thus, to enable comments, we need simply to include this file using the following template tag:

```
<?php comments_template(); ?>
```

That tag will most commonly be found in the “single.php” file, but could also

Leave a Comment

Name *

E-mail *

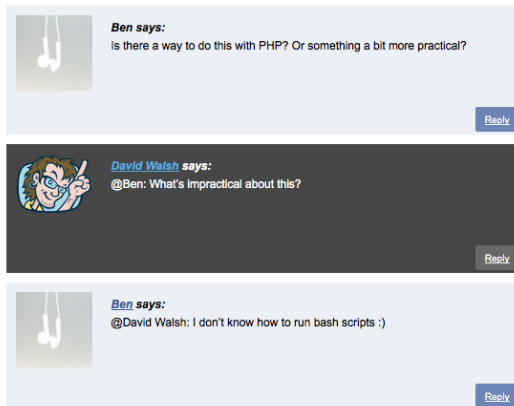
Website

☐ Notify me of followup comments via e-mail

be included in other types of template files. For example, at the CSS-Tricks site, each video screencast is displayed with a custom Page template. The regular Page template is used for things like the Contact page and thus doesn't include comments, but the video screencast template does, because each screencast is open and welcome to viewer feedback.

Now that we know where comments are used and how they are included, let's go through the anatomy of a typical WordPress Comment Area:

- **The comment display area**, or comment thread, may be configured to display comments in chronological order (oldest comment displayed first) or in reverse chronological order (newest comment displayed first) via a setting in the Admin Area (**Settings > Discussion**).



A typical comments area should display older comments first. Comments left on a post are essentially a discussion, and discussion only makes sense if displayed in chronological order.

Flipping the order and displaying newer comments first has potential uses, however, just make sure the purpose of the comment area isn't discussion.

For example, you can fairly easily create a community news section with a comment area. Just turn the label 'name' into 'title', and 'comment' into 'description' and flip the order to display most recent comments first. Presto-chango – a user contributed links area, with it's own RSS feed, since every Page or Post has its own unique comment feed (see chapter 6.2.4).

- **Nested comments** were integrated into WordPress version 2.7 and enable your readers to respond directly to specific comments. Such comments are then “nested” within the area beneath the original comment. This is the best way to enable true back-and-forth discussions in comment threads. Fortunately, as discussed later in the chapter, WordPress makes it fairly easy.



- **The comments display area** displays three types of responses: **comments, trackbacks, and pingbacks**. These may appear together or else segregated into distinct areas.

Smashing Magazine uses tabs to toggle between comments and trackbacks/pingbacks.

- **Gravatars!** These are small icons that are associated with each comment author. By default, WordPress can be configured to display Gravatars, and can even assign a random Gravatar for commentators that do not have them.

*Gravatars are connected to the author of the comment by their otherwise-hidden email address. In this screenshot, the bottom user does not have a Gravatar, either because they did not enter an email (wasn't required), or because that email didn't have an associated Gravatar. To avoid a missing icon, a default graphic is shown instead
Screenshot: sixrevisions.com.*

- **Comment metadata** includes things such as the date and time the comment was left, the name and URL of the commentator, and even the sitewide and/or local comment number.

If you use the standard `wp_list_comments` function to display comments, eliminating comment metadata may come down to using `display: none;` in your CSS file. Otherwise, a custom loop is in order.

Comments

Trackbacks/Pings

- 

Spooler

(September 28th, 2009, 4:27 am)

There are some disputable issues in the article. But it also has useful information. Read and squeeze! =)



Jim

September 27th, 2009

I'm surprised to see MarsEdit for the Mac left off the list. It's by far the best. That being said, I agree with Andy, using the built-in input method of Drupal is the best. It offers the best end-to-end experience, and you can see/expect.



Paul L'Acosta

September 27th, 2009

This is a great list John. Wish I'd have found it before as it's a horrible one-in-all blog editor. Too bad the ones for Mac don't support custom themes as pointed out. Guess I'd have to stick with ecto's ugly UI for now.



Valdecir Carvalho

September 27th, 2009

I've used Blogdesk for a long time, and then Windows Live Writer. I've used such applications, because "to me" they have such a bug with WordPress posts that look strange.

Valdecir

LEAVE COMMENT

 Name*

 Email*

 Website

Remember:

- Don't be a jerk.
- Wrap all code in `<code>` tags. (single or multiple lines)
- You may use regular HTML stuff like `<a href="">`, `<em>`, and `<strong>`

The comment form on CSS-Tricks offers some reminders about what people can and can't do in the comment area.

Submit Comment

- The **comment form** is the where the action happens. To leave a comment on your post, readers must fill out admin-specified fields (such as the name and email-address fields) of your comment form.
- The **"Name"** field of your comment form is where a commentator may specify their name. This field is usually "required," meaning that the comment will not be processed unless this field contains a value.
- The **"Email"** field of your comment form is where a commentator may specify their email address. This field is usually "required," meaning that the comment will not be processed unless this field contains a value.
- The **"URL"** field of your comment form is where a commentator may specify their website address. This field is usually "optional" because not everyone has the luxury of having their own website.
- The **"Comment"** field of your comment form is where the commentator writes the comment. This text field is obviously required because there would be nothing to post without it.
- The **"Submit"** button is what the commentator must click in order for the comment to be processed by the server and ultimately displayed (or not, depending on your comments policy) in the comment thread.

Now that we have reviewed the basics, let's dive into some great ways to improve the functionality, usefulness and user-friendliness of your response area.

7.2.1 Syndicating WordPress Comments

So you posted something that really has your readers buzzing – you are getting tons of comments and even a good number of trackbacks and pingbacks. As the conversation evolves, make it easy for your readers and commentators to stay current with the response thread by syndicating the discussion in feed format. By default, WordPress does this automatically by way of variously formatted comment feeds. Many themes provide links to these built-in feeds for every post on the site.

If your theme is not equipped with such links, you may add them quite easily by using one (or more) of the URL formats as described in the next couple of sections.



7.2.2 WordPress Main Comments Feed

By default, WordPress creates an RSS feed of all the comments on your site automatically. This feed is available at any of the following three URLs:

`http://domain.tld/comments/feed/` - **Permalink format**

`http://domain.tld/wp-commentsrss2.php` - **Default format**

`http://domain.tld/?feed=commentsrss2` - **Query-string format**

Why have a feed for all of your comments? That's a bit like asking why the sky is blue. It's not obvious, but it's mostly because WordPress is just trying to be cool and syndicate the information for anyone who wants to consume it. Perhaps you, as the

Feed Formats

For a more in-depth look at the different types of feeds and feed formats provided by WordPress, check out chapter 6.3.1.

site admin, prefer to read new comments on your site via your RSS reader rather than have them come to you via email. Perhaps your super-fans want to make sure they keep up on every last comment posted on your site. The feed is simply there to serve any need that may arise. Most likely, however, your readers would rather tune in to conversations on specific posts. That's where *post-specific* comment feeds come into play.

7.2.3 Post-Specific Comment Feeds

Sucking on a firehose of every comment a site generates is usually counterproductive. Fortunately WordPress also produces RSS feeds specifically for each Post. The URL structure is simple and easy to remember. Just append `"/feed/"` onto the end of any Post's permalink. Here is an example:

- **The Post** - `http://domain.tld/alligator-tacos/`
- **Post Comment Feed** - `http://domain.tld/alligator-tacos/feed/`

Deliciously simple. Such comment feeds make it easy for savvy visitors to subscribe to the post to stay current with the conversation. In fact, many themes include prominent links to their Post Comment Feed for easy access. The nice thing about individual post feeds is that they are available to anyone at anytime, even if the subscriber has not commented on the post.

Likewise, when using default (non-permalink) URLs, the post-specific comment feeds are available for each post via the following format:

`http://domain.tld/?feed=rss2&p=123`

...where `"p="` references the post ID.

A good place to display a feed button or link for your Post Comment Feeds is immediately before or after the comment form of each post. By making the comment feed readily available next to the comment form, commentators will be reminded to grab the feed for future updates.

This entry was posted on Wednesday, September 23rd, 2009 at 5:32 am and is filed under [JavaScript](#). You can follow any responses to this entry through the [RSS 2.0](#) feed. You can skip to the end and leave a response. Pinging is currently not allowed.
[Edit this entry.](#)

In addition to providing users a comments feed, you may also want to enable them to subscribe to comments via email. Often, receiving email notification of new comments is way easier than the process of subscribing (and eventually unsubscribing) to a new feed. Currently, WordPress does not provide this functionality by default, but until it does you can use Mark Jaquith's **Subscribe to Comments** plugin, which is readily available at <http://digwp.com/u/34>.

As a visitor leaving a comment, you have a checkbox giving you the ability to subscribe to subsequent comments on a post via email.

As an author, you are notified that you are already the author of the post (hence automatically receive new comment notifications) and are also given a link to manage subscriptions for the site.

The Subscribe to Comments plugin is easily configurable via the WordPress Admin, and provides a complete set of configurational options. After the plugin is installed, a common place to display the subscription option is directly beneath each post's comment form (see screenshot on this page).

7.3.1 Formatting the Comments Area

To code or not to code? Remember we said earlier how the Comments area can be a bit complex and the WordPress gods have taken some steps to make it easier for us? One of the big things we were talking about was a very specific WordPress function. That function is this:

```
<?php wp_list_comments(); ?>
```

The image contains two screenshots of the WordPress comment form. The top screenshot shows the form from a visitor's perspective. It includes input fields for 'Name (required)', 'Email (required)', and 'Website'. Below these fields is a checkbox labeled 'Subscribe to responses via Email' and a 'Submit' button. A small text box above the comment area states: 'YOU MAY USE <a>, , <code>, <pre> TAGS. PLEASE WRAP ALL CODE IN <code> TAGS.' The bottom screenshot shows the form from an author's perspective. It includes the same input fields, but the 'Subscribe to responses via Email' checkbox is checked. Below the comment area, there is a message: 'You are the author of this entry. Manage subscriptions.' and a 'Submit' button. The same text box about using tags is present above the comment area.

It looks so simple and innocent right there on the page, doesn't it? The fact is, `wp_list_comments` is a powerhouse of functionality, able to drastically simplify comment display in your theme. This simplicity, however, comes at the cost of flexibility. The `wp_list_comments` function literally creates and spits out all of the required HTML for all of the comments on any Post.

```
<ol class="commentlist">
    <?php wp_list_comments(); ?>
</ol>
```



Because this function generates all of the HTML automatically, you need to work with what it gives you. This can be a problem. For a theme designer, having HTML that you can't get your hands on can be frustrating. Don't like the particular class names that are used? Too bad. Don't want Gravatars? Too bad, you'll have to hide them with CSS. So at this point you'll have to decide which route you want to take:

`wp_list_comments`

- Easy to use out of the box
- Built-in threaded comments
- Built-in comment pagination
- Default CSS classes for many comment properties

```
<ol class="commentlist">
    <li class="comment even thread-even" id="div-comment-2083">
        <div class="comment-author">
            
            <cite class="fn"><a href="#">
        </div>
        <div class="comment-meta">
            <p>This just happened to
        </p>
        <div class="reply">
            <a rel='nofollow' class="reply" href="#">
        </div>
        </div>
    <ul class='children'>
        <li class="comment byuser" id="div-comment-2084">
            <div class="comment-author">
                
                <cite class="fn"><a href="#">
            </div>
            <div class="comment-meta">
                <p>That's the who
            </p>
            <div class="reply">
                <a rel='nofollow' class="reply" href="#">
            </div>
            </div>
        </li>
    </ul>
</li>
</ol>
```

Custom loop

- Full control, not as easy to implement
- Standardized and/or custom CSS classes for just about anything
- Plugin support for extra functionality like threading and pagination

Which method you decide to use depends on many factors. In general, we advise you to use the `wp_list_comments` function to display your comments. In those cases where more control over structure, presentation, and functionality is desired, using a custom loop is certainly going to be easier than wrestling with `wp_list_comments`. In addition to these two methods, there is also the “old-school” method of displaying comments: the “foreach” loop, which opens the door to more possibilities.

In the next section of our “Optimizing Comments” chapter, we’ll explore all three of these comment-display methods and then dig into some juicy techniques for formatting and enhancing your WordPress Comments Area.

7.3.2 Using `wp_list_comments()` or a Custom Loop?

Before WordPress version 2.7, the comments loop that was used to query the database and display your comments was a bit convoluted, but also well-established and well-understood. It basically consisted of the following code:

```
<?php foreach ($comments as $comment) : ?>
    <?php comment_author_link(); ?>
    <?php comment_text() ?>
<?php endforeach; ?>
```

Then, in WordPress version 2.7, comments were enhanced with optional threading and paging features. These new features are activated via the WordPress Admin

and require the `wp_list_comments()` template tag to operate. Thus, in 2.7 and better, we replace the `foreach` method with, yep, you guessed it, `wp_list_comments`:

```
<?php wp_list_comments(); ?>
```

Yep, that's all you need now to take advantage of WordPress' default, out-of-the-box, comment functionality, which now includes optional comment paging and threading. The old `foreach` loop method still works, but you will need a plugin for threaded and/or paged comments.

If you're thinking that you need to use the old comments loop to fully customize your comments display, think again. You *could* use the old loop setup, but you would miss out on all of the cool new features. Instead, WordPress makes it possible to create your own customized comments function and call it via the new `wp_list_comments()` template tag. Let's explore these different comment-display methods and check out the possibilities.

The new comments loop

The `wp_list_comments` function doesn't actually *look* like a loop, but it does the same thing as the old *foreach* loop by cycling through your comments and generating the required markup. Looking at the HTML output of `wp_list_comments`, we get something like this:

```
<ol class="commentlist">
  <li class="comment even thread-even depth-1" id="comment-1">
    <div id="div-comment-1" class="comment-body">
      <div class="comment-author vcard">
        <img alt="" src='http://www.gravatar.com/avatar/' class='avatar
          avatar-32 photo avatar-default' height='32' width='32' />
        <cite class="fn">
          <a href='http://wordpress.org/' rel='external nofollow' class='url'>
```

```

    Mr WordPress</a></cite> <span class="says">says:</span>
</div>
<div class="comment-meta commentmetadata">
    <a href="http://localhost/283/2009/08/hello-world/comment-page-
1/#comment-1">August 13, 2009 at 3:08 pm</a>
</div>
    <p>Hi, this is a comment.<br />To delete a comment, just log in and view
the post's comments. There you will have the option to edit or delete
them.</p>
    <div class="reply"></div>
</div>
</li>
</ol>

```

As you can see, the default output for the new comments loop (via the `wp_list_comments` tag), is an ordered list that contains a gravatar image, several divs, lots of variable classes, author links, and the comment itself. If this fits your theme's design and functionality, then you are good to go. Nothing could be easier, and the host of included classes and markup elements should be sufficient to get things looking exactly how you want them.

With the `wp_list_comments` method of displaying your comments, you have control over the following parameters:

- **avatar_size** - an integer specifying the size, in pixels, of your avatars. The default is 32.
- **style** - a string specifying the type of markup used for your comments. May be either "div", "ol", or "ul". The default is "ul", an unordered list. Note that you will still need to wrap your comments list with your markup element of choice. So, for example, if you are using an ordered list instead of an unordered list, you would need to include the enclosing "" tags like so:

```
<ol class="commentlist">
    <?php wp_list_comments(array('style' => 'ol')); ?>
</ol>
```

- **type** - a string that specifies the type of comments to display. Either "all", "comment", "trackback", "pingback", or "pings". "pings" is both "trackback" and "pingback" together. The default is "all".
- **reply_text** - text string specifying the text to display for the comment reply link. Default is "Reply".
- **login_text** - text to display when registration is required to comment. Default is "Log in to Reply".
- **callback** - a string specifying the name of a custom function for your comments display. Specifying your own function enables you to fashion your own comment-display loop. Defaults to null.
- **Additional parameters** may be available, see the codex: <http://digwp.com/u/35>.

That's quite a bit of control, but more customization may be necessary. If you need more control over how your comments are displayed, you will need to tap into the comment loop itself, which may be included directly in the `comments.php` file or tucked away neatly into your `functions.php` file. In the next section, we'll look at how to include a custom comments loop via the `functions.php` file, then we'll remind you how to do it old-school with the classic `foreach` comments loop.

Utilizing a custom comments loop via `functions.php`

First, open your `comments.php` file and add the following code:

```
<ul class="commentlist">
    <?php wp_list_comments('type=comment&callback=mytheme_comment'); ?>
</ul>
```

Then, open your functions.php file and add the custom comments function:

```
<?php function mytheme_comment($comment, $args, $depth) {
    $GLOBALS['comment'] = $comment; ?>
    <li <?php comment_class(); ?> id="li-comment-<?php comment_ID(); ?>">
        <div id="comment-<?php comment_ID(); ?>">
            <div class="comment-author vcard">
                <?php echo get_avatar($comment,$size='48',$default='<path_to_url>' );?>
                <?php printf(__('<cite class="fn">%s</cite>
                    <span class="says">says:</span>'), get_comment_author_link()); ?>
            </div>
            <?php if ($comment->comment_approved == '0') : ?>
                <em><?php _e('Your comment is awaiting moderation.');
```


Backwards-Compatible Comment Display

If you want your theme to be backwards-compatible, just check for the presence of the `wp_list_comments()` function in your `comments.php` file:

```
<?php if (function_exists('wp_list_comments')) : ?>
    <?php wp_list_comments(); ?>
<?php else : ?>
    <?php foreach ($comments as $comment) : ?>
        <?php comment_author_link(); ?>
        <?php comment_text() ?>
    <?php endforeach; ?>
<?php endif; ?>
```

Notice that we are not closing the `<li>` element. WordPress automatically generates the closing `</li>` depending on the markup generated for any nested comments.

Once in place, the code placed into your `comments.php` file will call your custom comments-loop from your `functions.php` file and output the contents to the page. This code may look like a lot, but it is easily chopped up and customized to suit your needs.

The point here is that it is possible to get behind the scenes and legitimately customize your comments loop using the new `wp_list_comments()` template tag.

Going “old-school” with the original comments loop

This makes it even easier to customize your comments display, but threaded comments and pagination are not possible without a plugin. Simply replace the `wp_list_comments` tag with the following `foreach` comments loop and customize with whatever markup and template tags you wish:

```
<?php foreach ($comments as $comment) : ?>
    <?php comment_author_link(); ?>
    <?php comment_text() ?>
<?php endforeach; ?>
```

And that’s all there is to it.

Now that we know how to customize our comment loop, let’s dig in, have some fun, and explore some super-awesome tips and tricks for our WordPress comment areas.

7.3.3 Implementing Paged Comments

You just never know when you have some content on your site that will explode and attract hundreds of comments. What am I talking about? You guys are stars, I'm sure that's going to happen to you all the time! A Post with hundreds of comments can have pretty crazy vertical scroll going on. In general, we aren't afraid of vertical scroll on the web (unlike that nasty *horizontal* scroll), but there are limits to anything. A recent post on CSS-Tricks with 62 comments netted 17,827 pixels in height!

An excellent way to handle posts that receive millions of responses is to break the comment thread into sections and display it across multiple pages. This "Paged Comments" feature was introduced in WordPress 2.7 and is extremely easy to implement. Go to **Settings > Discussion** in the Admin area to enable it:

Other comment settings

- ☒ Comment author must fill out name and e-mail
- ☐ Users must be registered and logged in to comment
- ☒ Automatically close comments on articles older than days
- ☒ Enable threaded (nested) comments levels deep
- ☐ Break comments into pages with top level comments per page and the page displayed by default

Comments should be displayed with the comments at the top of each page

And then in your `comments.php` file, insert these two template tags wherever you would like the "Next" and "Previous" pagination buttons to appear:

```
<?php previous_comments_link(); ?>  
<?php next_comments_link(); ?>
```

That set of tags will output the next/previous links for your comments area, automatically appearing only when there is more than one page of comments. If there is only one page of comments, no links are displayed.

HEADS UP: There are some potential duplicate-content issues (an SEO problem) with paged comments. This is because there are multiple unique URLs generated for each page of comments. Each of these URLs contains the exact same content as the original post. For example:

Canonical URLs

In 2009, the major search engines adopted support for canonical meta tags. These tags enable you to specify a single, definitive URL for your web pages, and thereby avoid any duplicate content issues. Here is an example of usage:

```
<link rel="canonical"
href="http://domain.tld/
canonical/url/" />
```

For more information on the use of canonical meta tags for WordPress, see chapter 8.2.5.

```
http://yoursite.com/ghost-dad-rocked/
http://yoursite.com/ghost-dad-rocked/comments-page-1/
http://yoursite.com/ghost-dad-rocked/comments-page-2/
http://yoursite.com/ghost-dad-rocked/comments-page-3/
```

Content-wise, the only difference between these three pages is the comments. To make matters worse, each of the comments has a datestamp that is a permalink to that particular comment, which uses the paginated URL in the link. So when a search bot wanders through, they'll see lots and lots of links to the "/comments-page-1/" version of your post – most likely many more links are pointing to the comment pages than the actual post URL.

To prevent this duplicate content issue from happening, there are several possible solutions:

- Don't use paged comments and party like it's 2007
- Use meta tags to specify the canonical URL for each post
- Install the **SEO for Paged Comments** plugin – <http://digwp.com/u/36>

If your site is super-popular, or is getting to be that way, it is probably a good idea to set some upper-limit for the number of comments that appear on a page and use a plugin to set the canonical URL for the post.

If the number of comments for each post on your site is minimal, the easiest thing to do is simply forego the paged comments and go "traditional" with each post including all of its comments on the same page.

To actually change the way WordPress generates the comment pages, the SEO for Paged Comments plugin eliminates duplicate content by replacing your post

content with excerpts on all comment pages other than the post page itself. I use this plugin on several of my sites and it works beautifully.

7.3.4 Implementing Threaded Comments

Threaded comments enable people to write a comment that is a direct response to an already existing comment, creating a more conversational flow to a comment thread. This approach is almost always ideal, assuming the design of your site can support it. Implementing threaded comments begins by enabling the option in the WordPress Admin:

Other comment settings

- ☒ Comment author must fill out name and e-mail
- ☐ Users must be registered and logged in to comment
- ☒ Automatically close comments on articles older than days
- ☒ Enable threaded (nested) comments levels deep
- ☐ Break comments into pages with top level comments per page and the page displayed by default
- Comments should be displayed with the comments at the top of each page

Assuming your theme supports threaded comments, activating this option in the Admin will endow each of your comments with a “Reply” link that enables visitors to reply directly to any comment on your post.

Before the implementation of threaded comments in WordPress 2.7, any direct replies to a comment required use of the infamous “@” symbol, as in “@comment-author: blah blah blah.” Although this method of replying has become quite common around the Web, it is no longer necessary on sites running WordPress version 2.7 or better.

Now, instead of “@ing” someone to address a previous comment, you can simply click on the “Reply” link next to the comment itself. Once submitted, your comment will be displayed directly under the comment to which you were

responding. This functionality makes it possible for each comment to be the start of a new, tangential discussion, and greatly increases the usability of your comments area.

If your theme does not provide threaded-comments functionality, here is an easy way to implement them within your `comments.php` file:

1. Backup your files, as usual (just in case you need to restore original files).
2. In the WordPress Admin, check the “Enable Threaded (nested) Comments” in the **Settings > Discussion** page. Also specify how many “levels deep” you would like to nest your comments.
3. Add the following code to your `header.php` file, directly above the `wp_head()` template tag:

```
<?php if (is_singular() AND comments_open() AND (get_option('thread_
comments') == 1)) wp_enqueue_script('comment-reply'); ?>
```

4. Add the following parameter to just before the closing “`</form>`” tag of your theme’s comment form:

```
<?php comment_id_fields(); ?>
```

5. Make sure that the `<textarea>` of your theme includes an attribute of `id="comment"`.

6. Make sure that your entire comment form is enclosed within a `<div>` with an attribute of `id="respond"`.

7. Add a “cancel-comment” link just above your comment form:

```
<div><?php cancel_comment_reply_link(); ?></div>
```

8. In your `comments.php` file, replace the old loop with the new hotness:

```
<?php if (have_comments()) : ?>
<ol class="commentlist">
    <?php wp_list_comments(); ?>
```

Starter CSS

There are a lot of classes in the HTML of threaded comments that you need to account for in CSS to make the comments actually look properly threaded. It's often easiest to start with a base set of styles and adjust as needed.

<http://digwp.com/u/468>

```

</ol>
<?php else : // this is displayed if there are no comments so far ?>
    <?php if ('open' == $post->comment_status) : ?>
        <!-- if comments are open, but there are no comments -->
        <p class="nocomments">Somebody say something!</p>
    <?php else : // comments are closed ?>
        <!-- if comments are closed -->
        <p class="nocomments">Comments are closed.</p>
    <?php endif; ?>
<?php endif; ?>

```

That's all there is to it, really. Of course, like all things WordPress, there are a gazillion different ways to customize things, change the design, enhance functionality, and so on. Once you have threaded comments set up, they are easily styled with the following CSS selectors:

```

.commentlist li.depth-1 {}
.commentlist li.depth-2 {}
.commentlist li.depth-3 {}      /* and so on */

```

Reply-To for Non-JavaScript Users

To accommodate those without JavaScript, replace your crusty old “Leave a Comment” text with something more dynamic by using this instead:

```

<?php comment_form_title('Leave a Reply', 'Leave a Reply to %s'); ?>

```

This tag will output “Leave a Reply” by default, but as soon as someone without JavaScript clicks to reply to a comment, the page refreshes and the output changes to “Leave a Reply to Whoever.” You can customize the text to whatever you want. The “%s” will dynamically output the name of the person being replied to.

Threaded Plugins

If your version of WordPress does not support threaded comments, upgrade. If upgrading is not on the menu, you can get threaded comments by using one of these fine plugins:

<http://digwp.com/u/186>

<http://digwp.com/u/187>

<http://digwp.com/u/188>

<http://digwp.com/u/189>

You can then target any element for a specific comment level like so:

```
.commentlist li.depth-1 div {}
```

This CSS would target any `<div>` elements within any first-level comment.

7.3.5 Separating Comments, Pingbacks, and Trackbacks

When displaying the responses to your posts, an effective way to reduce the clutter and keep things organized is to separate the comments from the pingbacks and trackbacks. This not only keeps the conversation flowing smoothly, it also emphasizes the ping/trackback responses by displaying them in a continuous list.

Even More Comments Styling

The new comment-display functionality generates a ton of context-specific classes, making it easy to style your comments however you wish. Each comment is surrounded by an element (`<li>` by default) that will include the following classes, depending on page view:

- 'comment', 'trackback', 'pingback'
- 'byuser' for any registered user
- 'comment-author-authorname' for specific registered user
- 'bypostauthor' for comments by the post author
- 'odd' and 'even' for odd and even numbered comments
- 'alt' for every other comment
- 'thread-odd', 'thread-even', 'thread-alt' for the top-level comment in a thread
- 'depth-1', 'depth-2', 'depth-3', etc. applied according to the level of comment nesting

```
<li class="comment even thread-odd thread-alt depth-1 parent" id="comment-2033">
```

This is one of the oldest tricks in the WordPress book, and there are as many different ways to approach it as there are different `comments.php` files. Here is a method for separating comments and ping/trackbacks that is as widely applicable, adaptable, and flexible as possible.

Thanks to the “type” parameter of the `wp_list_comments()` template tag, separating your comments, pings, and trackbacks is a breeze. Here are some examples to give you a general idea. Once you see the pattern, you can set things up however you like.

Separate comments from both pings and trackbacks

```
<h3>Comments</h3>
<ol class="comments">
    <?php wp_list_comments('type=comment'); ?>
</ol>
```

```
<h3>Pings/Trackbacks</h3>
<ol class="pings-trackbacks">
    <?php wp_list_comments('type=pings'); ?>
</ol>
```

Separate comments, pings, and trackbacks

```
<h3>Comments</h3>
<ol class="comments">
    <?php wp_list_comments('type=comment'); ?>
</ol>
<h3>Pings</h3>
```


Feature / Bury

This simple plugin will allow you to click links from the front or back end to "feature" or "bury" a comment. This doesn't do anything by itself, but adds those class names to the comment so you can style them appropriately.

<http://digwp.com/u/467>

```
<ol class="pingbacks">
    <?php wp_list_comments('type=pingback'); ?>
</ol>
<h3>Trackbacks</h3>
<ol class="trackbacks">
    <?php wp_list_comments('type=trackback'); ?>
</ol>
```

And finally, here is the code required to separate comments from pingbacks *and* trackbacks using the old-school, foreach loop:

```
<h3>Comments</h3>
<?php foreach ($comments as $comment) : ?>
    <?php $comment_type = get_comment_type(); ?>
    <?php if($comment_type == 'comment') { ?>
```

Display Clean Pingbacks and Trackbacks

When listing your pings using the default loop, you will see a link followed by a snippet of text from the page that pinged your post. These snippets can vary in size, and are difficult to control because you never know what to expect from them. Fortunately, we can clean up the default display of pings by removing the text snippet and listing only a nice, clean link back to the pinging site. To do this, we need to add the following callback function to your theme's `functions.php` file:

```
// clean pingbacks and trackbacks
function cleanPings($comment, $args, $depth) {
    $GLOBALS['comment'] = $comment;
    echo '<li>'.comment_author_link().'</li>';
}
```

Then, add the required callback parameter to the `wp_list_comments` function like so:

```
<?php wp_list_comments('type=pings&callback=cleanPings'); ?>
```

```

<div class="comment">
    <p><?php comment_author_link(); ?></p>
    <?php comment_text(); ?>
</div>
<?php } ?>
<?php endforeach; ?>
<h3>Pingbacks/Trackbacks</h3>
<ol>
<?php foreach ($comments as $comment) : ?>
    <?php $comment_type = get_comment_type(); ?>
    <?php if($comment_type != 'comment') { ?>
        <li><?php comment_author_link(); ?></li>
    <?php } ?>
<?php endforeach; ?>
</ol>

```

You will inevitably want to customize the markup, template tags, and other functionality, but the central technique is all there. We are segregating the previous single comment loop with two individual loops that target comments and ping/trackbacks, respectively. Within each of these two loops, we are testing for the comment type and only displaying the preferred type within each loop.

Along the way, we season the script with some modest HTML markup to keep things readable on the page. Once you establish core functionality on your site, you should customize the code according to your needs.



#34 — Jeff Starr • [[edit](#)] [[delete](#)]

Hi Nicholas, it really all depends on the right balance between small and large. For PNG transparency is required. GIF image is suitable for fewer colors.



#35 — Sunny Senevirathne • [[edit](#)] [[delete](#)]

I am trying to find a technique for displaying characters..If any of you know a better way let me know. Thanks!

Trackbacks / Pingbacks

1. [Pages tagged 'minute'](#) • [[edit](#)] [[delete](#)]
2. [Daily Links I AndySowards.com :: Prof Downloads, Math and being a Web 2.0](#)
3. [Graphics Optimization Blog » Blog Archive: blog posting by Jeff Starr](#) • [[edit](#)] [[delete](#)]
4. [magnus.de](#) • [[edit](#)] [[delete](#)] [[spam](#)]

Keep 'em Separated

As you can see, displaying comments, pingbacks, and trackbacks separately helps keep your comments area clean and organized. Otherwise, pingbacks and trackbacks get mixed up in your comment thread and spoil the conversation.

7.3.6 Eliminating Pingbacks and Trackbacks

It may be impossible to stop people from linking to your site (and *why* would you want to?), but you don't have to show their pingbacks and trackbacks on your site.

At best, pingbacks are still used around the Web, but trackbacks are slowly going extinct. Most trackbacks are spam anyway, so there is good reason to stop them from showing on your site.

Indeed, there are many sites that wish not to display their incoming pingbacks and trackbacks along with their comments, or anywhere else for that matter. If this sounds like you, there are several ways to go about it.

Only Display Pingbacks/Trackbacks Markup if They Exist

In the previous code example showing how to display separate comments, pingbacks, and trackbacks, the headings for each section will be displayed even if there are no responses. To prevent this from happening, we ensure that all ping/trackback-related code is displayed **only** if some ping/trackbacks actually exist. Here's the code to do this:

```
<?php if (!empty($comments_by_type['comment'])) : ?>
    <h3>Comments</h3>
    <ol class="comments">
        <?php wp_list_comments('type=comment'); ?>
    </ol>
<?php endif; ?>

<?php if (!empty($comments_by_type['pings'])) : ?>
    <h3>Trackbacks/Pingbacks</h3>
    <ol class="pingbacks-trackbacks">
        <?php wp_list_comments('type=pings'); ?>
    </ol>
<?php endif; ?>
```

Disable ping/trackbacks from the Admin area

In the Admin area, go to “**Settings > Discussion**” and uncheck the option for “Allow link notifications from other Weblogs.” This will disable pingbacks and trackbacks on your site for all posts going forward. This global setting may be overridden on a post-by-post basis by checking “Allow Pings” on the **Post Write/Edit** page.

Omit ping/trackbacks from the comments loop

Using the `wp_list_comments` tag to display your comments, simply set the “type” parameter as described in chapter 7.3.2. With the following code, for example, no pingbacks or trackbacks will be displayed – only comments:

```
<h3>Comments</h3>
<ol class="comments">
    <?php wp_list_comments('type=comment'); ?>
</ol>
```

Delete the wp-trackback.php file from the root directory

This is a very effective way of permanently disabling pingbacks and trackbacks. It is a totally safe thing to do, just remember to re-delete the file after upgrading WordPress.

Globally disabling via plugin

To globally disable trackbacks only, we can use Viper007Bond’s **Disable Trackbacks** plugin <http://digwp.com/u/37>. Since this is a very simple plugin, we can just add it to our active theme’s `functions.php` file (code modified/formatted for clarity):

Save the Juice

Pingbacks may represent a reciprocal linking affair, but trackbacks are generally free links to the referring site. Why waste the juice on a site that can't spare an actual link?

```
// Disable Trackbacks http://digwp.com/u/37
class DisableTrackbacks {
    // initialize plugin
    function DisableTrackbacks() {
        add_action('pings_open', array(&$this, 'pings_open'));
    }
    // if trackback, close pings
    function pings_open($open) {
        return ('1' == get_query_var('tb')) ? FALSE : $open;
    }
}

// load after all other plugins
add_action('plugins_loaded', create_function('', 'global
$DisableTrackbacks; $DisableTrackbacks = new DisableTrackbacks();'));

```

Once in place, the `DisableTrackbacks` function will effectively eliminate all trackbacks from registering with your site. To ensure that no existing trackbacks are displayed, you should also omit them from the comment loop as described in the previous section.

Note that pingbacks will still be processed even with the `Disable Trackbacks` script in place.

7.3.7 Control Comments, Pingbacks, and Trackbacks Directly with the Database

WordPress provides several ways of controlling which posts are open to comments, pingbacks, and trackbacks. In the Admin Settings Discussion area, you can disable all responses on a sitewide basis for all **future** comments. You may also control responses for each post via the “Discussion” panel in the post-management screen.

Beyond these methods, you can either use a plugin such as the Auto Moderate Comments plugin <http://digwp.com/u/38>, or else save time and effort by simply querying the database directly. Querying the database is a great way to control responses on *existing* posts. Here is a collection of quick, copy-&-paste SQL recipes for controlling comments, pingbacks, and trackbacks:

Globally enable pingbacks/trackbacks for all users

```
UPDATE wp_posts SET ping_status = 'open';
```

Globally disable pingbacks/trackbacks for all users

```
UPDATE wp_posts SET ping_status = 'closed';
```

Globally disable pingbacks/trackbacks before a certain date

For this query, specify the ping_status as either open or closed. Also, specify the date by editing the date, "2009-09-09", to suit your needs.

```
UPDATE wp_posts SET ping_status = 'closed' WHERE post_date < '2009-09-09'
AND post_status = 'publish';
```

Complete, one-step discussion management

Given the queries described above, we may fashion the following "one-step" SQL queries, perfect for complete, plugin-free discussion management:

Globally enable/disable all discussion: comments, pingbacks and trackbacks

For this query, specify the comment_status as either open, closed, or registered_only. Also, specify the ping_status as either open or closed.

```
UPDATE wp_posts SET comment_status = 'open', ping_status = 'open'
WHERE comment_status = 'closed' AND post_status = 'publish';
```

Back that DB Up!

Before running any SQL queries or doing any other work on your database, remember to make a backup copy or two. The database stores all of your information and data – it would suck royally to lose it.

Quick and Easy

These SQL queries are perfect for managing discussion a few times a year without using another plugin.

Globally enable/disable comments, pingbacks and trackbacks before a certain date

For this query, specify the `comment_status` as either `open`, `closed`, or `registered_only`. Also, specify the `ping_status` as either `open` or `closed`. Finally, specify the date by editing the `"2009-09-09"` to suit your needs.

```
UPDATE wp_posts SET comment_status = 'closed', ping_status = 'closed'
WHERE post_date < '2009-09-09' AND post_status = 'publish';
```

7.4.1 Customizing Comment Display

Even if your theme is already configured to display comments in a useful and visually appealing way, there is always something that may be improved. In this section, we'll show you some awesome ways to completely trick out the visual presentation of your WordPress comments area.

7.4.2 Numbering Comments Globally and Locally

There are basically two different ways to number your comments: globally and locally. Global numbering of comments happens automatically as comments are added to the WordPress database. The number of each comment on your site represents its position within the comments database table. Global comment numbers are called via the `comment_ID` template tag. This tag may be used to identify individual comments with their associated comment ID, like so:

```
<a href="#comment-<?php comment_ID(); ?>"
    id="comment-<?php comment_ID(); ?>">Permalink for this comment</a>
```

Identifying each comment with its global ID via the markup makes it easy for people to refer and link to any specific comment on your site. And, while you may also label each comment as it appears on the page with its corresponding global ID, you may want to use local numbering of comments instead. Why?

Because global numbering would appear as shown in the left screenshot, and local numbering would appear as shown in the right screenshot:

17 Responses:



Jean Moniatte says:

Gmail (and Google Aps) addresses this very well with the Archive and Labels features as well as the integrated Tasks application.

I haven't touched Outlook (or whatever email client) in 3 years, and never missed a bit of it.

#41713 - September 23rd, 2009 at 9:09 pm



Mike D. says:

Jean: Interesting, I'll have to check it out. Unfortunately I never really warmed up to the Gmail web interface, although I know others love it. I still feel like I want e-mail as a dedicated application and not just a web app.

#41714 - September 23rd, 2009 at 9:18 pm

*Global Comment Numbering
is nice in that every single comment has
a unique code and link, but doesn't make
much sense in this visual context.*

17 Responses:



Jean Moniatte says:

Gmail (and Google Aps) addresses this very well with the Archive and Labels features as well as the integrated Tasks application.

I haven't touched Outlook (or whatever email client) in 3 years, and never missed a bit of it.

#1 - September 23rd, 2009 at 9:09 pm



Mike D. says:

Jean: Interesting, I'll have to check it out. Unfortunately I never really warmed up to the Gmail web interface, although I know others love it. I still feel like I want e-mail as a dedicated application and not just a web app.

#2 - September 23rd, 2009 at 9:18 pm

*Local Comment Numbering
makes more sense in the context of a
conversation, and the links are still unique.*

Ahh, yes. Much better. A good rule of thumb is to keep our comments *numbered locally in the comment display area*, and *numbered globally in the source code*. Indeed, implementing locally numbered comments is straightforward. If your theme does not include such functionality, open your `comments.php` file and follow one of the following tutorials, depending on your comment-loop configuration:

For single comments loop (i.e., no separation of comments and ping/backtracks)

1. In your `comments.php` file, add the following line *above* your comment loop:

```
<?php $commentcount = 1; // number of first comment ?>
```


2. Then, *within* the comment loop, add the following line to the location where you would like to display each comment's number:

```
<p class="comment-count"><a href="#comment-<?php comment_ID(); ?>">#<?php  
echo $commentcount++; ?></a></p>
```

For double comments loop (i.e., separation of comments and ping/trackbacks)

1. In your comments.php file, add the following line *above* your comment loop:

```
<?php $commentcount = 1; // number of first comment ?>  
<?php $pingbackcount = 1; // number of first pingback ?>
```

2. Then, *within* the loop that displays comments, add the following line:

```
<p class="comment-count"><a href="#comment-<?php comment_ID(); ?>">#<?php  
echo $commentcount++; ?></a></p>
```

3. Finally, within the loop that displays ping/trackbacks, add the following line:

```
<p class="pingback-count"><a href="#comment-<?php comment_ID(); ?>">#<?php  
echo $pingbackcount++; ?></a></p>
```

And that's all there is to it. Notice that we have also implemented global comment numbers within this method. Thus, once in place, this code will display a link for each comment that shows local comment numbers on the web page and global comment numbers in the source code. Nice.

Once your comments are numbered locally, you can enjoy the best of both worlds by including something like this along with the display of each comment:

```
<a href="<?php the_permalink(); ?>#comment-<?php comment_ID(); ?>"  
id="comment-<?php comment_ID(); ?>" title="Permalink for comment #<?php  
echo $comment_count; ?>">Comment #<?php echo $comment_count; ?></a>
```

Once in place, the previous code will output information as in this screenshot:



...and the following markup in the source code:

```
<h4>
    <a href="#comment-72066" id="comment-72066" title="Posted on June 13, 2009">#10</a> -
    <a target="_blank" href="http://carlnunes.com" rel="external nofollow">carlnunes</a>
</h4>
```

Count Comments Only

To display the number of comments *only*, place the following code into your `functions.php` file:

```
function countComments($count) {
    global $wp_query;
    return count($wp_query->comments_by_type['comment']);
}
add_filter('get_comments_number', 'countComments', 0);
```

What we're doing here is filtering WordPress' `comments_number` function to output *only* the number of comments. Normally the function counts all comments, pingbacks, and trackbacks, but with this function in place, only comments will be counted.

To display the total number of comments after implementing the `countComments` function, simply place the `comments_number()` template tag wherever you would like to display the number.

10 Comments

vga cable Says:

June 4th, 2009 at 8:39 pm (Reply)

I know Bing is new search engine and there will be targeted keywords for some keywords it is showing after 100 pages compare to google, but anyway

Montana Says:

June 10th, 2009 at 2:53 pm (Reply)

Be careful with that javascript, piling up too much older/slower browsers at interpreting it. There is

Stephen Cronin Says:

June 18th, 2009 at 8:41 pm (Reply)

Montana,
Umm, no the JavaScript isn't on the website. Greasemonkey Firefox extension. If they can't live with the script.

Drew Says:

June 19th, 2009 at 12:01 am (Reply)

I've heard about Bing several weeks ago but I still want to stick with Google. Maybe, in due time Google when it comes to search engine.

Ferienwohnung Kroatien Says:

June 22nd, 2009 at 9:41 am (Reply)

You are correct! open source can rally build on the

7.4.3 Alternating Comment Styles

Another way to improve the usability and overall stylishness of your comments-display area is to alternate the appearance of every other comment, as seen in the left-hand screenshot of Stephen Cronin's site, scratch99.com.

Depending on your particular theme design, this method of distinguishing between odd and even comments may greatly facilitate comment readability on your site, thereby making it easier for readers to follow the conversation and perhaps feel motivated to participate.

The process of styling alternating comments differently involves determining which comments are *oddly numbered* and which are *evenly numbered*. In WordPress 2.7 and better, comments automatically include "odd" and "even" classes for odd- and even-numbered comments, respectively. This makes it a snap to implement your own custom CSS styles for each of these classes. Perhaps something like this:

```
.odd {  
    background: white;  
    color: black;  
}  
.even {  
    background: black;  
    color: white;  
}
```

Take it Easy...

Just because you can pimp out some serious alternating comment styles doesn't give you license to go crazy. In this screenshot (at left) from scratch99.com, Stephen Cronin alternates comment styles in subtle fashion, letting the content speak for itself.

Well, maybe not *that* extreme, but you get the idea. :)

7.4.4 Custom Styles for Authors & Members

Similar to the technique of styling alternate comments is the fine art of styling unique author and member comments. As with alternating comment styles, unique author and registered-member styles is a great way to improve readability, usability, and all-around sleekness of your WordPress comments area. And the good news is that WordPress 2.7 automagically includes CSS classes for both types of comments.

For the author of the blog post, WordPress includes a class named “bypostauthor”. For registered users who are logged in to your site, a class named “byuser” is included in that user’s comments. These classes enable you to easily apply custom CSS styles to these types of comments.

For older versions of WordPress, some additional code is required to implement custom styles for authors. The following code will check the comment author to see if it is the same as the post author. If it is, a CSS class named “comment-author” will be source code output. Thus, place the following code into the HTML element of your choice:

```
<?php if ($comment->user_id == $post->post_author) { echo '
class="comment-author"'; } ?>
```

Using a basic comments loop, here is how we could use this code:

```
<?php foreach ($comments as $comment) : ?>
    <div?php if ($comment->user_id == $post->post_author) {
        echo ' class="comment-author"'; } ?>
        <p><strong><?php comment_author_link(); ?></strong></p>
        <?php comment_text(); ?>
    </div>
<?php endforeach; ?>
```



David Link says:

Irony:

That fancy little article I

Having said that, I still

Sites should be built fle

mean these rules *alway*

Example being the very

That CSS won’t need to

the site, and is specific

inline styles make a goo



Chris Coyier says:

Ha, I’m glad someone p

yes, excellent example

The theory here was th

across like that in RSS,

where the only way!



Andrew Vit says:

Agreed in principle, but

just one-off unique styl

In order to separate sty

Once in place, this code will enable you to apply CSS styles such as the following:

```
.comment-author {  
    font-weight: bold;  
    color: red;  
}
```

The cool thing about this snippet is that it will output the “comment-author” class for *any* post author on your site. Unfortunately, because the CSS class is the same for all post authors, it is not possible to style author comments differently according to specific authors.

Fortunately, there is always more than one way to get things done with WordPress. Here is an alternate “author-highlight” method that will enable you to target specific post authors directly:

```
<?php if ($comment->comment_author_email == "chris@digwp.com")  
    echo 'class="author-chris"'; ?>
```

With that snippet in place, any comment left specifically by Chris may be styled by an echoed CSS class named “author-chris”. Likewise, to output unique class names for multiple authors, we can use something like this:

```
<?php if ($comment->comment_author_email == "chris@digwp.com") echo  
'class="author-chris"'; elseif ($comment->comment_author_email == "jeff@  
digwp.com") echo 'class="author-jeff"'; ?>
```

By adding additional “elseif()” statements, we may target as many specific authors as needed.

7.4.5 Styling Comments with Gravatars

As many of you know, gravatars are globally recognized avatars that appear next to your comments and other content on supportive sites around the Web. So instead of just leaving your name and website, you can show a thumbnail-size image of yourself to help improve branding and site-promotion.

To get your own Gravatar, simply go to the free signup page at the Gravatar website <http://digwp.com/u/190> and provide the required information, which is basically just a valid email address. After signing up, you get to upload your own image and customize your profile. Within moments, your new gravatar will begin to appear on gravatar-enabled websites.

To display gravatars on your WordPress site, log in to your Admin area and visit the **Settings > Discussion** page. There, you will be able to specify your preferences for the following:

- Whether or not you want to display gravatars
- Acceptable rating of your displayed gravatars
- The default gravatar that will be displayed

Once you have set these options, click save and open your theme's `comments.php` file. In the location where you want the gravatars to display, add the following line to your comments loop:

```
<?php if(function_exists('get_avatar')) { echo get_avatar($comment, $size = '50', $default = 'http://domain.tld/default.png'); } ?>
```

Then, specify the size of the gravatar, which may be anything up to 96 pixels (square), and also the default image to use in case the user does not have a gravatar. This code will output an image element (`<img>`) that points to the commentator's gravatar and includes the following attributes (based on our previous example):

Twitter Avatars

You can also display Twitter avatars with comments by grabbing the free script at:

<http://digwp.com/u/191>

Add the script to your `functions.php` file and display the avatars in your theme like so:

```
<?php twittar('50',  
'http://digwp.com/path/  
default.png', '#ffffff',  
'twitavatars', 1, 'G'); ?>
```

```

```

You may then customize the image with additional markup, CSS styles, or whatever you wish. It's that easy – *if* you are using WordPress 2.5 or better. If you are using an older version of WordPress, you may use the same code as above, but you will need to either install a gravatar plugin or else include the required code manually. Here are a couple of the best gravatar plugins:

- **Gravatars2** - <http://digwp.com/u/192>
- **WP-Gravatar** - <http://digwp.com/u/193>

To get gravatars working on older versions of WordPress, you may also skip the plugin and code it yourself. Open your theme's `comments.php` file and add the following to the location where you would like the gravatars to appear within the comments loop:

```
<?php // gravatars for WordPress < 2.5
$gravatar_size    = "50";
$gravatar_default = "http://domain.tld/default.png";
$gravatar_email   = get_comment_author_email();
$gravatar_url     = "http://www.gravatar.com/avatar.php?gravatar_
id=".md5($gravatar_email)."&default=".urlencode($gravatar_
default)."&size=".$gravatar_size;
echo '';
?>
```

Before using this code, remember to edit the “`$gravatar_size`” and “`$gravatar_default`” variables with the display size and default image, respectively. Once in place, this code will display a gravatar for each comment author who has a gravatar, or else it will display the specified default image.

Finally, if you are developing a theme that needs to ensure backwards-compatibility with older (pre-2.5) versions of WordPress, you may combine the default, built-in method with the backwards-compatible fallback.

To do this, we employ a simple bit of PHP logic, like so:

```
<?php // backwards-compatible gravatars
if (function_exists('get_avatar')) {
    // gravatars for WordPress 2.5 +
    echo get_avatar($comment, $size = '50', $default = 'http://domain.tld/
default.png');
} else {
    // gravatars for WordPress < 2.5
    $gravatar_size      = "50";
    $gravatar_default   = "http://domain.tld/default.png";
    $gravatar_email     = get_comment_author_email();
    $gravatar_url       = "http://www.gravatar.com/avatar.php?gravatar_
id=".md5($gravatar_email)."&default=".urlencode($gravatar_
default)."&size=".$gravatar_size;
    echo '';
} ?>
```

As with the previous two methods, place this code within the comments loop and you are good to go. You will want to edit the “\$gravatar_size” and “\$gravatar_default” variables with the display size and default image, respectively.

Gravatars in Posts!

Gravatars aren't only for comments — they also look great when used to display a thumbnail image of the post author. All you need to do is include the following code within the post loop of your single.php template file:

```
<?php $author_email = get_
the_author_email(); echo
get_avatar($author_email,
'96'); ?>
```


7.4.6 Add a “Your comment is awaiting moderation” Message

When moderating comments on your WordPress site, it is very helpful to commentators to let them know if their submitted comment is waiting in the moderation queue. By default, WordPress will display moderated comments to their authors but not to other visitors. Thus, it may be confusing for someone who leaves a comment and sees it appear in the comment thread, only to watch as it is completely ignored by everyone else. For this reason, including some sort of an “awaiting moderation” message is extremely important for improving the usability of your comments area.

The good news is that, if you are using the `wp_list_comments` method to display your comments, you don’t need to do anything – a comment moderation message is already built-in. If you are using a custom loop via `functions.php`, simply add the following snippet of code to your function:

Forget?

For more information on using either a custom loop or the `wp_list_comments` method of displaying comments, check out section 7.3.2 in this chapter.

```
<?php if ($comment->comment_approved == '0') : ?>
    <p class="moderation">Your comment is awaiting moderation.</p>
<?php endif; ?>
```

And, if you are using the old-school `foreach` comments loop, adding a comment-moderation message is just as simple:

1. Locate the comments loop, which should open and close with the following code:

```
<?php foreach ($comments as $comment) : ?>
    <!-- this is the comments loop! -->
<?php endforeach; ?>
```

2. Within the comments loop, add the following code:

```
<?php if ($comment->comment_approved == '0') : ?>
    <p class="moderation">Your comment is awaiting moderation.</p>
<?php endif; ?>
```

3. Your loop should now contain the following (along with other template tags, markup, etc.):

```
<?php foreach ($comments as $comment) : ?>
    <?php comment_text(); ?>
    <?php if ($comment->comment_approved == '0') : ?>
        <p class="moderation">Your comment is awaiting moderation.</p>
    <?php endif; ?>
<?php endforeach; ?>
```

That's all there is to it! Season to taste and enjoy your site's improved usability!

7.4.7 Moderation Links in the Theme Itself

As described in chapter 5.3.4 (Extending WordPress), adding some easy admin buttons is a great way to improve the comment-moderation process for your site. In addition to managing comments through the WordPress Comments Admin area, it is also helpful to have some easy admin buttons located next to the comments as they appear on your blog. Here is an easy way to add “spam” and “delete” links next to each comment whenever you are logged in as Administrator. Having access to spam and delete buttons next to each comment makes it super-easy to clean up missed spam, scummy trackbacks, and other garbage. I am continually scouring my old posts' comments and weeding out junk, which for me is always easier to see from the actual post pages themselves. Having quick and easy access to spam and delete buttons has made my comment management routine considerably easier.

To add this functionality, we take advantage of the `functions.php` file by adding the following script:

```
<?php // spam & delete links for all versions of WordPress
function delete_comment_link($id) {
```

```

        if (current_user_can('edit_post')) {
            echo ' | <a href="'.get_bloginfo('wpurl').'/wp-admin/comment.php?action=c&c='.$id.'">Delete</a> ';
            echo ' | <a href="'.get_bloginfo('wpurl').'/wp-admin/comment.php?action=c&dt=spam&c='.$id.'">Spam</a>';
        }
    } ?>

```

Place this function in your theme's `functions.php` file, and then call the function by adding the following code to the desired location in your `comments.php` file:

```
<?php delete_comment_link(get_comment_ID()); ?>
```

And that's all there is to it! Depending on the placement of the function call, your comments area should now feature quick and easy "spam" and "delete" buttons next to each individual comment. Even better, this improved function is version-independent, backwards-compatible, and thus will work for any version of WordPress.

7.4.8 Display Comment, Ping/Trackback Counts

One more trick before we dig into optimization. Here is a way to easily display the number of comments, pingbacks, and trackbacks for each of your posts. Open your theme's `single.php` file and add the following parameters to your `comments_template()` tag:

```
<?php comments_template('/comments.php', true); ?>
```

Then to display the counts, simply use any of the following snippets in your comments template:

```
<?php global $wp_query; // this line required for any of the following ?>
<?php // pingback and trackback count
    echo count($wp_query->comments_by_type['pings']); ?>
<?php // trackback count
    echo count($wp_query->comments_by_type['trackback']); ?>
<?php // pingback count
    echo count($wp_query->comments_by_type['pingback']); ?>
<?php // comment count
    echo count($wp_query->comments_by_type['comments']); ?>
```

Just Numbers

Each of these tags outputs a number, namely the number of pings, trackbacks, pingbacks, and comments, respectively.

7.5.1 Optimizing the Comment Form

As we improve the quality of our readers' comments, let's not forget about the comment form itself. There are many ways to improve the functionality and appearance of your comment form, including adding a preview feature, rich-text editor, and anti-spam functionality. Let's dig in!

Unstyled Forms

Even very well-marked-up forms look like crap without CSS. Whaddyagonnado?

7.5.2 Set up Comment Previews

Encourage your visitors to contribute to the conversation by helping them preview their comments before submission. By adding a comment-preview feature to your comment area, you provide readers with a powerful tool for eliminating

Drop-Dead Easy Comment Forms

WordPress 3.0 makes it easier than ever to include comment forms in your themes. Simply add the following tag to your `comments.php` file:

```
<?php comment_form(); ?>
```

Learn more about this tag at the WP Codex: <http://digwp.com/u/498>

Leave a Comment

[Click here to cancel reply.](#)

Name*

Email*

Website

Comment

errors, reducing noise, and increasing quality. And best of all, adding comment-preview to your site is as easy as installing and configuring one of these plugins:

- **Ajax Comment Preview** <http://digwp.com/u/2>
Provides an excellent Ajax-powered comment-preview plugin that works and feels great. Admin Options page lets you specify formatting options.
- **Ajax Force Comment Preview** <http://digwp.com/u/194>
This plugin takes the utility of comment previews one step further by actually “forcing” commentators to preview their comments before submission. The plugin takes advantage of Ajax to avoid the need for a page refresh. Features built-in anti-spam functionality.
- **jQuery Comment Preview WordPress Plugin** <http://digwp.com/u/40>
jQuery Comment Preview uses the jQuery library to generate built-in comment previews that do not require refreshing the page. Includes Admin Options page.
- **Live Comment Preview** <http://digwp.com/u/39>
Live Comment Preview provides a “smooth live preview as you type.” This cool plugin makes live comment previews super easy – simply activate and enjoy. No Ajax required – works completely in the browser.
- **Really Simple Live Comment Preview** <http://digwp.com/u/41>
A do-it-yourself tutorial that has received a lot of attention. The author keeps the article updated with the latest code and improvements, so if you need to implement a custom live-comment-preview solution, be sure to check it out.
- **Filosofo Comments Preview** <http://digwp.com/u/42>
This well-built plugin works great with many WordPress themes and even works with WordPress-2.7’s new threaded comments feature. Includes Admin Options page for complete control over appearance and functionality.

7.5.3 Rich-Text Editors for Comments

Out of the box, most WordPress themes provide a plain `<textarea>` for comments. This may work fine for savvy users who are familiar with a bit of HTML to add some

structure to their comments, but what about the average visitor who may not realize that such markup is possible.

Fortunately, there are many options available for enabling your visitors to style their comment with simple markup using a rich-text editor. Let's look at some of the best rich-text editors and plugins available for your comment form:

- **TinyMCEComments** <http://digwp.com/u/46>

TinyMCEComments transforms your comment textarea into a full-featured WYSIWYG rich-text editor using WordPress' bundled TinyMCE library (WordPress 2.0 and up).

- **TinyMCE** <http://digwp.com/u/47>

TinyMCE is a popular open-source rich-text editor that is easy to integrate with your site. Highly customizable with themes and plugins. International language support, multiple browser support, Ajax support, and more. One of the best.

Include Comments on Non-Single Pages

Normally, WordPress will only display your comments template file, `comments.php`, when called from your theme's `single.php` or any `page.php` file. This makes it possible to display comments on single posts and pages, but for archive views such as categories and indexes, you need to add a variable into the mix. Here's how to display a comments template in any archive or index page.

Replace your regular `comments_template()` tag – you know, the tag used to call your `comments.php` file – with this one:

```
<?php global $withcomments; $withcomments = true; comments_template("/custom-comments.php"); ?>
```

This will include the file named "`custom-comments.php`" and use its contents to display your comments. Nice!

Alternately, if you are only displaying a single post on, say, your index page, you can simply include your comments template within the loop:

```
<?php while (have_posts()) : the_post(); ?>
    <?php comments_template(); ?>
<?php endwhile; ?>
```

For more RTE choices, check out “Rich-Text Editors for 2010 and Beyond” at Six Revisions:

<http://digwp.com/u/496>

- **CKEditor** <http://digwp.com/u/48>
CKEditor (recently changed from FCKEditor) is another popular open-source rich-text editor that is easy to integrate with your site. This awesome editor includes image-uploading, layout templates, custom styles, and even Adobe AIR support. Works and looks great in all modern browsers. Four stars!
- **WidgEditor** <http://digwp.com/u/49>
WidgEditor is an easy-to-use open-source rich-text editor by Cameron Adams. This is not a plugin, but is a breeze to implement. Uses clean, easy-to-read JavaScript that degrades gracefully.
- **MarkItUp!** <http://digwp.com/u/50>
MarkItUp! is a rich-text editor built on top of jQuery. Includes Ajax live preview, keyboard-shortcut support, and weighs only 6.5kb. An excellent solution for transforming any HTML textarea into a full-featured WYSIWYG text editor.
- **Damn Small Rich Text Editor** <http://digwp.com/u/51>
Damn Small Rich Text Editor is a free, lightweight rich-text editor designed to work with jQuery and PHP. Includes image-uploading capabilities and is plugin-enabled.
- **WMD: The WYSIWYM Markdown Editor** <http://digwp.com/u/52>
“What You See Is What You Mean” uses the Markdown language to mark up comments, which then transforms into HTML when the comment is saved. Comes with a nice editor bar as well as automatic live comment previews.

7.5.4 Adding Comment Quicktags

One of the best ways to enhance the functionality of WordPress comments is to implement comment quicktags. Quicktags are JavaScript buttons included with the comment form that provide shortcuts to a variety of common markup elements. While typing their comments, users may want to use some bold or emphasized text, or maybe even drop a killer blockquote or a few lines of code.

Comment quicktags enable commentators to format their comment with the appropriate markup easily with the click of a button. It’s a great feature that makes

leaving a comment so much fun. Sound good? Here are some comment-quicktag plugins to make it happen:

- **LMB^Box Comment Quicktags** <http://digwp.com/u/43>

The LMB^Box Comment Quicktags plugin adds a Quicktag Toolbar directly above the comment form's text area. The LMB^Box Quicktag Toolbar looks and functions exactly like the toolbar used in the WordPress Admin area, and may be styled to fit your site's design. The toolbar provides quicktags for strong/bold, em/italic, code, blockquote, and links tags by default, and the set of buttons is completely configurable via its Admin Options page.

- **Comment Quicktags** <http://digwp.com/u/44>

Inserts an expandable quicktag toolbar above the WordPress comment form. You can customize the default CSS styles and add your own (X)HTML buttons, all from within a handy admin interface. Very nice.

Display Comments in Descending Order

By default, WordPress displays comments in chronological order: oldest comments first, newest comments last. To reverse this order, we can take advantage of PHP's awesome `array_reverse` function. Here's how:

1. Place the following code directly before your comment loop:

```
<?php $comments = array_reverse($comments, true); ?>
```

2. Your comment loop should now look similar to this:

```
<?php $comments = array_reverse($comments, true); ?>
<?php foreach ($comments as $comment) : ?>
    <?php comment_author_link(); ?>
    <?php comment_text() ?>
<?php endforeach; ?>
```

That's it! Your comments are displayed in reverse chronological order.

- **Comment Form Quicktags** <http://digwp.com/u/45>

This plugin inserts quicktags of the admin page to the upper part of textarea of the comment form. Provides easy tag configuration via the WordPress Admin.

- **Comment Quicktags Reloaded** <http://digwp.com/u/195>

A slightly modified version of Owen Winkler's Comment Quicktags plugin. If the original Comment Quicktags plugin is breaking your theme, try this version.

7.5.5 Comment Management and Spam Prevention

Properly managing your comments can be as time-consuming as it is important. Sites with carefully managed comments sections are miles apart from sites that don't. But it takes time, patience, and a good amount of attention to detail.

Fortunately, when it comes to monitoring, moderating, and pruning comments, there are plenty of awesome plugins to help automate, improve, and enhance the process of filtering out spam and encouraging comments. We explore some of these tools in the final three sections of this chapter.

7.6.1 Controlling Comment Spam

Ahh yes. The wonderful world of comment spam. By now, we assume that everyone is well-acquainted with the never-ending and utterly hellish spam battle. It rages constantly, with desperate Viagra and Cialis spammers seeking every opportunity to flood your otherwise pristine comments area with a truckload of their stinky garbage. *Eww...*

Fortunately, all of the *smart* people are on *our* side, and there are many talented developers who continue to fight spam on the front lines. Fortunately, because of the popularity of WordPress, we enjoy a vast arsenal of effective anti-spam techniques for our sites. Let's explore some of the best.

7.6.2 WordPress' Built-In Anti-Spam Functionality

With all of the excellent plugins available, many users don't realize that WordPress provides some powerful tools for controlling spam right out of the box. Within the Discussion page of the WordPress Admin area, there are several options that are useful for controlling comments and preventing spam. Let's take a quick look at each these options.

- **Default Article Settings** - If you don't need comments on your site, disable them. Completely disabling comments is a sure-fire, bulletproof way to eliminate *all* comment spam.
- **Users must be registered and logged in to comment** - One way to allow comments but eliminate 90% (or more) of spam is to require the user to be registered and logged-in to your site before commenting.
- **Before a comment appears** - An administrator must always approve the comment. If you have the time, this is another good way to ensure that no spam comments appear on your site.
- **Before a comment appears** - Comment author must fill out name and e-mail. This may not do much for fighting spam, but it will discourage some of the lazier folks from commenting.

No Plugins, No Spam

It's true. By carefully crafting your WordPress blog's built-in comment settings, it is entirely possible to run a virtually spam-free site with absolutely no plugins whatsoever — not even Akismet. Read more at:

<http://digwp.com/u/416>



Discussion Settings

Default article settings

- ☒ Attempt to notify any blogs linked to from the article
 - ☐ Allow link notifications from other blogs (pingbacks and trackbacks)
 - ☒ Allow people to post comments on new articles
- (These settings may be overridden for individual articles.)*

Other comment settings

- ☒ Comment author must fill out name and e-mail
- ☐ Users must be registered and logged in to comment

Blacklist Ninja

For a highly effective, custom blacklist of regular expressions for your site's Comment Moderation or Spam Blacklist, drop by Perishable Press and grab a copy:

<http://digwp.com/u/196>

- **Before a comment appears** - Comment author must have a previously approved comment. This is an effective way to prevent spam, although it does require some time on busier sites.
- **Auto-close Comments** - Spammers often target older posts. Auto-closing comments on old posts helps reduce overall spam.
- **Hold a comment in the queue if it contains "x" or more links** - As it says, this is a great way to screen any comments that have too many links. How many is up to you.
- **Comment Moderation Blacklist** - This is a regular-expression blacklist of terms that will kick suspect comments into the moderation queue. Load this puppy up with all your favorite spam words – cialis, xanax, vicodin, viagra, etc. – and gain more control over your site's comments.
- **Comment Spam Blacklist** - Similar to the Moderation Blacklist, the Spam Blacklist features a list of regular expressions that will throw the comment into the spam bin. Be careful of the words that you include in this list, because anything that matches is essentially discarded.

7.6.3 Anti-Spam Plugins for WordPress

- **Akismet** <http://digwp.com/u/95>
One of the best. Requires registration key. Easy to use. Bundled with WordPress. Excellent spam protection. 'Nuf said.
- **Peter's Custom Anti-Spam** <http://digwp.com/u/97>
CAPTCHA-based anti-spam plugin. Forces all commentators to identify a random word before comment submission. Words are displayed as images and are completely customizable.
- **reCAPTCHA Plugin** <http://digwp.com/u/99>
Displays words from old books that users must correctly interpret. Uses the popular reCAPTCHA service that is used on popular sites such as Twitter, Facebook, and StumbleUpon. Upside: use of this service helps to digitize old books. Downside: requires a key to work.

- **WP-HashCash** <http://digwp.com/u/102>
CAPTCHA-based anti-spam plugin that claims to be 100% effective. JavaScript is required to generate a secret number that is verified by the plugin script.
- **Simple Trackback Validation** <http://digwp.com/u/104>
Simple Trackback Validation provides solid protection against trackback spam. If your site suffers from relentless rounds of trackback spam, this plugin is for you.

7.7.1 Other Considerations & Techniques

Clearly, the WordPress comments area is one of the most highly configurable and flexible parts of the entire application. To round out the chapter, here are a few more considerations that will help you maximize comment potential.

7.7.2 Enhancing and Encouraging Comments

In addition to the numerous methods covered so far, there are also many ways to advance comment functionality using plugins. Here are two awesome plugins for enhancing and encouraging comments on your site.

- **Keyword Luv** <http://digwp.com/u/199>
The Keyword Luv plugin rewards your commentators by separating their name from their keywords in their website link. This enables commentators to leave their name without sacrificing the keywords they want for the link to their website. From an SEO perspective, more focused and relevant anchor text benefits their site and thus encourages additional comments.
- **CommentLuv** <http://digwp.com/u/200>
CommentLuv creates a titled link to the commentator's most recent post. This encourages comments by giving commentators a general site link and a more specifically targeted post link. CommentLuv also enables click-tracking and provides statistics at commentluv.com.

7.7.3 Nofollow Links

Nofollow links look like this:

```
<a rel="nofollow" href="http://domain.tld/"  
  title="Example of a nofollow link">This is a nofollow link</a>
```

Google + Nofollow

Confused about Google's new nofollow policy? Here are some great sources for more information:

<http://digwp.com/u/201>
<http://digwp.com/u/202>
<http://digwp.com/u/203>

Nofollow Wars

For more nofollow bloodshed, see chapter 8.4.6.

Nofollow links are flagged by supportive search engines as not being endorsed by the author. Compliant search engines operate according to convention:

- **nofollow links** will not be followed through to the next page
- **nofollow links** will not be included in the calculation of page rank (Google recently announced that they *do* include nofollow links when determining PR)
- **nofollow links** will not include the anchor text in determining the linked page's relevant keywords

Basically, a nofollow link will be ignored by Bing and Yahoo, and partially ignored by Google. SEO gurus use nofollow attributes to influence the flow of link equity (page rank) throughout their site. Several years ago, WordPress developers decided it was a good idea to include nofollow attributes on all comments and pings.

Unfortunately, not everyone is thrilled about the idea of nofollow. The argument is that visitors deserve credit when commenting on your site. By slapping comment links with nofollow attributes, you remove incentive and diminish reward. Fortunately, there are some excellent “no-nofollow” plugins available for WordPress. For a comprehensive list, check out the Dofollow Plugin Library at Perishable Press: <http://digwp.com/u/204>.

7.7.4 Integrating Twitter

Last but not least, we want to look at some of the many ways to integrate everybody's favorite microblogging service, Twitter! We love to Twitter almost as much as we love to create beautiful, modern sites with WordPress. And so naturally



we are thrilled to have such a wonderful variety of plugins enabling us to combine the two. Let's have a look at some of the best.

- **TweetSuite** <http://digwp.com/u/208>
The TweetSuite plugin integrates Twitter with a host of useful tools, including everything from server-side TweetBacks and automatic post tweets to Tweet-This buttons and widget functionality.
- **Tweet This** <http://digwp.com/u/209>
Tweet-This buttons on every post. Perfect for sharing your posts on Twitter.
- **WP Twitip ID** <http://digwp.com/u/210>
This plugin makes it possible for users to add their Twitter name along with the usual comment information, such as name and URL. This provides incentive for readers to comment while enhancing the comment-display area.
- **TwitThis** <http://digwp.com/u/213>
TwitThis makes it easy for readers to share your posts on Twitter by clicking on the automatically generated "TwitThis" links for each post. Before the URL of your post is sent to Twitter, it is shortened via URL-shortening service, TinyURL.
- **Twitter Tools** <http://digwp.com/u/214>
"Complete integration" between WordPress and Twitter, enabling you to tweet blogs, blog tweets, and much more. Many options make it easy to customize.

And so...

In this action-packed chapter, we have explored the WordPress comment system in considerable depth. With such incredible flexibility, amazing customizations, and awesome plugins available to you, transforming your WordPress Comments Area into a beautiful, well-optimized, user-friendly response system is at your fingertips.

Of course, the greatest comment system in the world is of no use if nobody can find your site. In the next chapter, we dig into the fine art of search-engine optimization and explore many useful and effective SEO techniques.

@Anywhere

Twitter has a JavaScript library for integrating Twitter features into other websites. It's a little more involved than a plugin but the possibilities are very cool, like making @usernames automatically pop up "cards" with that users information and a follow button.

<http://digwp.com/u/469>

Stupid Twitter Tricks

There's nothing stupid about the awesome collection of custom Twitter tricks now available at Perishable Press:

<http://digwp.com/u/420>

I don't know the key to success,
but the key to failure is trying to
please everybody.

– **BILL COSBY**

8

Search Engine Optimization

8.1.1 SEO Strengths and Weaknesses

Out of the box, WordPress provides great flexibility in terms of organizing and managing your blog's content. Much of this flexibility comes by way of WordPress' category and tag architecture. Each and every post created on your blog may be assigned to any number of both categories and tags.

Categories are meant to classify content according to broad definitions, while tags are used to classify content more specifically. For example, a post about your favorite movies may be categorized in the "Favorite Movies" category, while being tagged for some of the movies featured in the article: "Star Wars," "The Matrix," and "Blade Runner."

Beyond this central organizing principle, WordPress brings with it many strengths and weaknesses in terms of how content is organized and made available to both users and the search engines. Let's examine some of these SEO factors before digging into the fine art of optimizing your WordPress-powered site for the search engines.

8.1.2 Strong Focus on Content

Content, as they say, is king. The Web exists because of it. Users are searching for it. Search engines are built on it. In order to succeed on the Web, your site should be focused on delivering useful content above all else. Awesomely, one of the main goals of WordPress is to make publishing content as easy as possible.

Once WordPress is set up, getting your content online happens as fast as you can create it. On the front end, there are hundreds of top-quality themes available, each focused on organizing and presenting your content with both users and search engines in mind.

8.1.3 Built-In “nofollow” Comment Links

Perhaps not as useful as originally conceived, nofollow attributes placed on commentator links have long been thought of as an effective method of improving the SEO-quality of WordPress-powered sites. For those of you who may be unfamiliar with the whole “nofollow” thing, for now suffice it to say that nofollow attributes are placed on links to prevent search engines from following those links to their targets. Originally, this was intended to serve as a way to conserve valuable page rank, but after it was revealed that this method no longer works, nofollow commentator links may be a moot point. We’ll discuss this more in-depth later on in the chapter.

WordPress + nofollow

Check out chapter 7.7.3 for more information on nofollow, WordPress, and the search engines.

8.1.4 Duplicate Content Issues

While the organizational strengths of WordPress are great for managing content, it also comes with a price: duplicate content. Duplicate content is essentially identical content appearing in more than one place on the Web. Search engines such as Google are reported to penalize pages or sites associated with too much duplicate content. Returning to our movie example for a moment, our WordPress-powered site may suffer in the search rankings because identical copies of our movie article are appearing at each of the following URLs:

- **original article** -> <http://example.com/blog/my-favorite-movies/>
- **category view** -> <http://example.com/blog/category/favorite-movies/>
- **star-wars tag view** -> <http://example.com/blog/tag/star-wars/>
- **the-matrix tag view** -> <http://example.com/blog/tag/the-matrix/>
- **blade-runner tag view** -> <http://example.com/blog/tag/blade-runner/>

Yikes! And if that weren't bad enough, we also see the exact same post content appearing at these URLs:

- **daily archive view** -> <http://example.com/blog/2009/02/02/>
- **monthly archive view** -> <http://example.com/blog/2009/02/>
- **yearly archive view** -> <http://example.com/blog/2009/>
- **author archive view** -> <http://example.com/blog/author/your-name/>

Depending on your particular WordPress theme, this situation could be even worse. By default, all of your posts are available in identical form at each of the previous types of URLs. Definitely not good from a search-engine point of view. Especially if you are the type of blogger to make heavy use of tags, the number of duplicated posts could be staggering.

8.2.1 Controlling Duplicate Content

Fortunately, WordPress' poor handling of duplicate content is easily fixed. In fact, there are several ways of doing so. In a nutshell, we have plenty of tools and techniques at our disposal for winning the war on duplicate content:

- meta nofollow tags
- meta noindex tags
- nofollow attributes
- robots directives
- canonical meta tags
- use excerpts for posts

So what do each of these sculpting tools accomplish and how do they help us eliminate duplicate content? Let's take a look at each of them.

8.2.2 Meta noindex and nofollow Tags

Meta nofollow tags are actually inline link elements located in the <head> section of your WordPress pages. For example, in your blog's "header.php" file, you may find something like this:

```
<meta name="googlebot" content="index,archive,follow" />
<meta name="msnbot" content="all,index,follow" />
<meta name="robots" content="all,index,follow" />
```

This code tells search engines – specifically, Google, Bing, and any other compliant search engine – that the entire page should be indexed, followed, and archived. This is great for single post pages (i.e., the actual "My Favorite Movies" article posted in our example), but we can use different parameters within these elements to tell the search engines *not* to index, follow, or archive our web pages.

Ideally, most bloggers want their main article to appear in the search results. The duplicate content appearing on the other types of pages may be controlled with this set of meta tags:

Love Juice

Anyone dipping into the murky waters of SEO will inevitably discover that there are many ways to refer to the SEO value of web pages. PR, Page rank, link equity, link juice, page juice, link love, love juice, rank juice, and just about any other combination of these terms is known to refer to the same thing: the success of a web page in the search engines.

```
<meta name="googlebot" content="noindex,noarchive,follow" />
<meta name="robots" content="noindex,follow" />
<meta name="msnbot" content="noindex,follow" />
```

Here, we are telling the search engines *not* to include the page in the search engine results, while at the same time, we are telling them that it is okay to crawl the page and follow the links included on the page. This prevents the page from appearing as duplicate content while allowing link equity to be distributed throughout the linked pages. Incidentally, we may also tell the search engines to neither index nor follow anything on the page by changing our code to this:

```
<meta name="googlebot" content="noindex,noarchive,nofollow" />
```

```
<meta name="robots" content="noindex,nofollow" />
<meta name="msnbot" content="noindex,nofollow" />
```

So, given these meta tags, what is the best way to use this method to control duplicate content on your WordPress-powered site? We're glad you asked. By using a strategic set of conditional tags in the "header.php" file for your theme, it is possible to address search-engine behavior for virtually all types of pages, thereby enabling you to fine-tune the indexing and crawling of your site's content. To see how this is done, consider the following code:

```
<?php if(is_home() && (!$paged || $paged == 1) || is_single()) { ?>
    <meta name="googlebot" content="index,archive,follow,noodp" />
    <meta name="robots" content="all,index,follow" />
    <meta name="msnbot" content="all,index,follow" />
<?php } else { ?>
    <meta name="googlebot" content="noindex,noarchive,follow,noodp" />
    <meta name="robots" content="noindex,follow" />
    <meta name="msnbot" content="noindex,follow" />
<?php } ?>
```

Tell Search Engines not to Index a Specific Post

In this section we see how to disable search-engine indexing for different categories, archives, pages, and other page views, but what if we want to prevent indexing of only one specific post? There are several SEO plugins that enable this functionality, but you don't really need one to do it. All you need to do is get the ID of the post for which you would like to disable indexing. Then, open your theme's header.php file and place this snippet within the <head> section:

```
<?php if ($post->ID == 77) { echo '<meta name="robots" content="noindex,noarchive">'; }
```

Change the ID from "77" to the ID of your post and done! With this in place, compliant search engines such as Google and MSN/ Bing will neither index nor archive the specified post (ID #77 in this example).

The conditional PHP tags used in this example effectively are saying: “If the current page is the home page or the single post page, then allow the search engines to both index and follow the content of the page; otherwise, since the page is neither the home page nor the single post page, it is probably a tag page, category page, or other archive page and thus serves as duplicate content; therefore tell the search engines to follow all links but not index the content.”

Of course, it isn’t always a bad thing to have *some* peripheral pages indexed in the search engines. In addition to having their home page and single pages indexed (as in our example above), many people prefer to have either tag pages or category pages (or both!) indexed in the search engines as well. In reality, the types of pages that you want indexed are completely up to you and your personal SEO strategy.

Prevent Duplicate Content Caused by Paginated Comments

Since WordPress 2.7, comments may be paginated, such that “x” number of comments appear on each page. While this is a step in the right direction, there may be a duplicate content issue resulting from the fact that your post content will appear on every page of your paginated comments. To resolve this issue, place the following code into your `functions.php` file:

```
// prevent duplicate content for comments
function noDuplicateContentforComments() {
    global $cpage, $post;
    if($cpage > 1) {
        echo "\n".<link rel="canonical" href="'.get_permalink($post->ID).'" />.'" "\n";
    }
}
add_action('wp_head', 'noDuplicateContentforComments');
```

Further Information

For more information and techniques on paged comments and duplicate content, check out chapter 7.3.3.

This code will generate canonical `<head>` links for your all of your paginated comments. The search engines will then use this information to ensure that the original post permalink is attributed as the actual article.

So, if you would like to include both tag and category pages in the search results, you would simply modify the first line of our previous example like so:

```
<?php if(is_home() && (!$paged || $paged == 1) || is_category() || is_tag() || is_single()) { ?>
```

8.2.3 Nofollow Attributes

Another useful tool in the fight against duplicate WordPress content is the controversial “nofollow” attribute. The nofollow attribute is placed into hyperlinks like this:

```
<a href="http://domain.tld/path/target/"  
  rel="nofollow">This is a "nofollow" hyperlink</a>
```

Links containing the nofollow attribute will not be “followed” by Google, but may still be indexed in the search results if linked to from another source. Because such links are not followed, use of the nofollow attribute is an effective tool in the reduction and prevention of duplicate content.

For an example of how nofollow can be used to help eliminate duplicate content, let’s look at a typical author archive. In the author-archive page view, you will find exact replicas of your original posts (unless you are using excerpts). This duplicate content is highly avoidable by simply “nofollowing” any links in your theme that point to the author-archive page view. Here is how the nofollow link would appear in your theme files:

```
<a href="http://domain.tld/author/author-name/"  
  rel="nofollow">This link will not be followed to the  
  author archive</a>
```

Exclude Admin Pages from Search Engines

You may also replace or add other types of pages to your meta-tag strategy by using any of the following template tags:

- `is_home()`
- `is_page()`
- `is_admin()`
- `is_author()`
- `is_date()`
- `is_search()`
- `is_404()`
- `is_paged()`
- `is_category()`
- `is_tag()`
- `is_date()`

To target any date-based archive page (i.e. a monthly, yearly, daily or time-based archive) that is being displayed, use this:

- `is_archive()`

Remember, there are more than just date-based Archives. Other types of Archive pages include sequential displays of category, tag, and author pages. `is_archive()` will target all of these page types.

And of course there are many more types of these conditional tags available to WordPress. See the WordPress Codex for more information: <http://digwp.com/u/3>

Exclude Specific Pages from Search Engines

Ever wanted to keep a few specific pages out of the search engines? Here's how to do it using WordPress' excellent conditional tag functionality.

Just place your choice of these snippets into the <head> section of your header.php file and all compliant search engines (e.g., Google, MSN/Bing, Yahoo!, et al) will avoid the specified page(s) like the plague.

This menu of snippets provides many specific-case scenarios that may be easily modified to suit your needs.

Exclude a specific post

```
<?php if(is_single('17')) { // your post ID number ?>
    <meta name="googlebot" content="noindex,noarchive,follow" />
    <meta name="robots" content="noindex,follow" />
    <meta name="msnbot" content="noindex,follow" />
<?php } ?>
```

Exclude a specific page

```
<?php if(is_page('17')) { // your page ID number ?>
    <meta name="googlebot" content="noindex,noarchive,follow" />
    <meta name="robots" content="noindex,follow" />
    <meta name="msnbot" content="noindex,follow" />
<?php } ?>
```

Exclude a specific category

```
<?php if(is_category('17')) { // your category ID number ?>
    <meta name="googlebot" content="noindex,noarchive,follow" />
    <meta name="robots" content="noindex,follow" />
    <meta name="msnbot" content="noindex,follow" />
<?php } ?>
```

Exclude a specific tag

```
<?php if(is_tag('personal')) { // your tag name ?>
    <meta name="googlebot" content="noindex,noarchive,follow" />
    <meta name="robots" content="noindex,follow" />
    <meta name="msnbot" content="noindex,follow" />
<?php } ?>
```

Exclude multiple tags

```
<?php if(is_tag(array('personal','family','photos')))) { ?>
    <meta name="googlebot" content="noindex,noarchive,follow" />
    <meta name="robots" content="noindex,follow" />
    <meta name="msnbot" content="noindex,follow" />
<?php } ?>
```

Exclude posts tagged with certain tag(s)

```
<?php if(has_tag(array('personal','family','photos')))) { ?>
    <meta name="googlebot" content="noindex,noarchive,follow" />
    <meta name="robots" content="noindex,follow" />
    <meta name="msnbot" content="noindex,follow" />
<?php } ?>
```

Granted, using nofollow links to control duplicate content is not 100% foolproof. If the author-archive URL is linked to from any “followed” links, that page still may be indexed in the search engines. For pages such as the author archives that probably aren’t linked to from anywhere else, nofollow may help prevent a potential duplicate-content issue.

8.2.4 Robots.txt Directives

Another useful and often overlooked method of controlling duplicate content involves the implementation of a robots.txt file for your site. Robots.txt files are plain text files generally placed within the root directory of your domain.

`http://domain.tld/robots.txt`

Robots.txt files contain individual lines of well-established “robots” directives that serve to control the crawling and indexing of various directories and pages. Search engines such as Google and MSN that “obey” robots directives periodically read the robots.txt file before crawling your site. During the subsequent crawl of your site, any URLs forbidden in the robots.txt file will not be crawled or indexed.

Keep in mind, however, that pages prohibited via robots directives continue to consume page rank. So, duplicate content pages removed via robots.txt may still be devaluing your key pages by accepting any link equity that is passed via incoming links.

Even so, with other measures in place, taking advantage of robots.txt directives is an excellent way to provide another layer of protection against duplicate content and unwanted indexing by the search engines.

Let’s look at an example of how to make a useful robots.txt file. First, review the default directory structure of a WordPress installation in the screenshot (next page).

For a typical WordPress installation located in the root directory, there is no reason for search engines to index URLs containing any of the core WordPress files. So we begin our robots.txt file by writing:

Yahoo! Disobeys

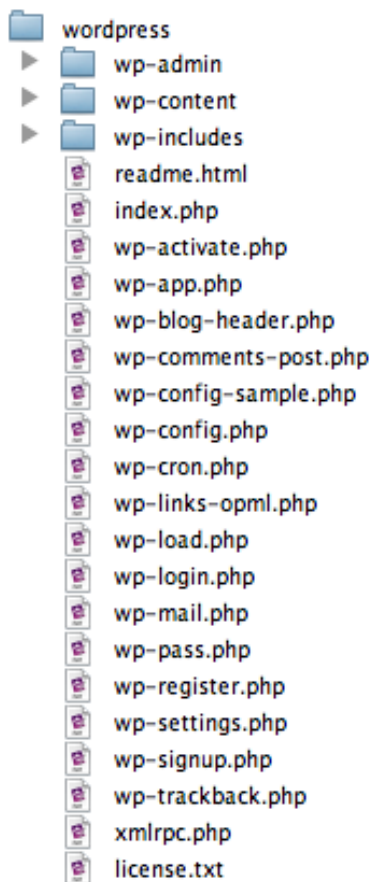
Sadly, when it comes to search engines that comply with robots.txt directives, Yahoo! falls far short:

<http://digwp.com/u/218>

Not Foolproof

Pages blocked by robots.txt directives may still appear within the index if linked to by “trusted, third-party sources.”

<http://digwp.com/u/219>



WordPress Files

Download, unzip, and you should be looking at something similar to this: WordPress' default directory structure.

Disallow: /wp-*

Disallow: *.php

These two lines tell compliant search engines to ignore any URL beginning with "http://domain.tld/wp-" or ending with ".php". Thus, all of our core WordPress files are restricted and will not be crawled by compliant search engines.

Now, consider some of the types of WordPress-generated URLs that we don't want the search engines to follow or index:

http://domain.tld/feed/	- your site's main feed
http://domain.tld/comments/feed/	- your site's comments feed
http://domain.tld/other/feeds/	- every other type of feed
http://domain.tld/post/trackback/	- every trackback URL on your site
http://domain.tld/2008/08/08/	- archive views for every day
http://domain.tld/2008/08/	- archive views for every month
http://domain.tld/2008/	- archive views for every year

Of course, there are other types of pages which we may also wish to exclude from the search engines, such as category and tag archives, but you get the idea.

To prohibit robots-compliant search engines from accessing and indexing the miscellaneous pages listed above, we add these directives to our robots.txt file:

Disallow: */feed*

Disallow: */trackback*

Disallow: /20*

Taken together, the previous two sets of robots.txt directives give us this:

```
Disallow: /wp-*  
Disallow: *.php  
Disallow: */feed*  
Disallow: */trackback*  
Disallow: /20*
```

See the pattern here? We use the “Disallow:” directive to restrict the crawling of any URL matching the specified regular-expression (regex) pattern. But use caution: regular expressions are powerful stuff, so be sure you know what you are doing before experimenting on your own.

How to “Allow” Search Engine Access

The “Allow” robots directive is designed to explicitly allow search engine access to specific files. When using wildcards to disallow entire directories, for example, the Allow directive may be used to override the setting for a specific directory or file. For example, I use the following robots.txt directives to prevent compliant search engines from accessing anything contained within my Mint statistics directory:

```
Disallow: */mint/*
```

This works great because it keeps Google et al from trespassing where they don’t belong. As it turns out, however, there is one otherwise affected URL pattern that I *want* the search engines to access, namely, my downloadable files. Using the Download Counter Pepper <http://digwp.com/u/220> to monitor my site’s downloads, my download URLs are rewritten as follows:

```
http://perishablepress.com/press/mint/pepper/orderedlist/downloads/download.zip
```

With my robots.txt directive in place, search engines will never see my downloadable goodies. This is where the awesome Allow directive comes into play. I now allow access to all of my downloads with a single robots directive:

```
Allow: */mint/pepper/orderedlist/downloads/*
```

Now, any download URL otherwise blocked via the previous Disallow directive is now explicitly allowed. Awesome.

Express Yourself

For more help with regular expressions, check out this reference from zytrax.com:

<http://digwp.com/u/221>

The regular expressions used here have been tested to work properly via Google's Webmaster Tools <http://digwp.com/u/222>, and essentially inform compliant search engines to do the following:

- Do not crawl or index any URL beginning with "http://domain.tld/wp-"
- Do not crawl or index any URL ending with ".php"
- Do not crawl or index any URL containing the character string, "/feed"
- Do not crawl or index any URL containing the character string, "/trackback"
- Do not crawl or index any URL beginning with "http://domain.tld/20"

Finally, there are a couple more things that we will need to add to our robots.txt file in order to make it complete. First, we need to specify which search engines should apply the directives, so we add this at the very beginning of the file, before our disallow rules:

User-agent: *

The wildcard operator (*) is used to target all compliant search engines, however, any specific user-agent may also be specified. For example, to apply our robots.txt directives only to Google, we would use this instead:

User-agent: Googlebot

XML Sitemap Plugin

Using a sitemap for your site is an effective way to help the search engines crawl and index your content. For an easy way to set up a sitemap for your site, check out the excellent Google XML Sitemaps:

<http://digwp.com/u/223>

With the wildcard operator, however, *everyone* is included, even Google. In addition to specifying the user-agent, we may also specify a sitemap to facilitate its use. Assuming we place our sitemap in the root of our example site, we write:

Sitemap: http://domain.tld/sitemap.xml

Combining everything, we place the following set of directives into our site's `robots.txt` file:

```
User-agent: *
Disallow: /wp-*
Disallow: *.php
Disallow: */feed*
Disallow: */trackback*
Disallow: /20*
Sitemap: http://domain.tld/sitemap.xml
```

This is a complete, well-tested set of robots directives that is optimized for WordPress-powered sites. Far more simple and equally effective as some of the other examples seen around the Web.

Just keep in mind that any `robots.txt` file will only be obeyed by *compliant* search engines, which fortunately includes the two largest, Google & Bing.

8.2.5 Canonical Meta Tags

In 2009, the major search engines (Google, Bing, Yahoo! and Ask) announced support for “canonical meta tags.” Canonical meta tags are designed to tell search engines which URL to count as the actual, original address of a web page.

For example, if you are running an e-commerce site that includes multiple URLs all pointing to the same product, such as these:

Enhancing Permalink Structure for Better Performance

When it comes to planning the best permalink strategy, consider the following quote from the WP Codex:

“For performance reasons, it is not a good idea to start your permalink structure with the category, tag, author, or postname fields. The reason is that these are text fields, and using them at the beginning of your permalink structure it takes more time for WordPress to distinguish your Post URLs from Page URLs (which always use the text ‘page slug’ as the URL), and to compensate, WordPress stores a lot of extra information in its database (so much that sites with lots of Pages have experienced difficulties). So, it is best to start your permalink structure with a numeric field, such as the year or post ID.”

In summary, although it may not matter much for smaller, low-traffic sites, it is best to keep the following in mind when choosing the format of your permalinks:

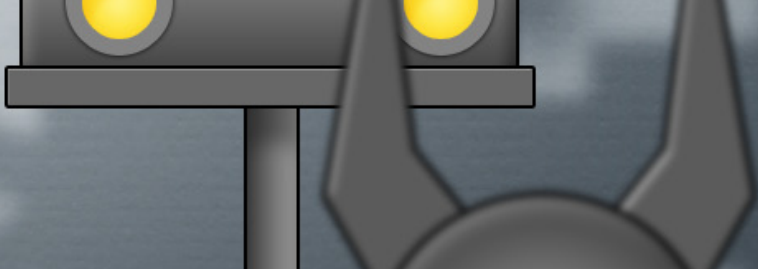
Instead of using something like this:

```
/%postname%/post_id/
/%category%/postname/
```

Get some numbers in there with something like this:

```
/%post_id%/postname/
/%year%/category/postname/
```

Much better, especially for busy, high-traffic websites.



X-Robots Meta Directives

To control indexing and caching of non-(X)HTML content types, using meta robots directives is *not* an option. An excellent example of this involves directing Google to index and cache PDF documents. The last time we checked, meta tags can't be added to PDFs, Word documents, Excel documents, text files, and other non-(X)HTML-based content. The solution, of course, is to take advantage of the relatively new HTTP header, X-Robots-Tag.

The X-Robots-Tag header takes the same parameters as used by meta robots tags. For example:

- index — index the page
- noindex — don't index the page
- follow — follow links from the page
- nosnippet — don't display descriptions or cached links
- nofollow — don't follow links from the page
- noarchive — don't cache/archive the page
- none — do nothing, ignore the page
- all — do whatever you want, default behavior

...and so on. Within ordinary meta tags, these directives make it possible to control exactly how search engines handle your (X) HTML-based web pages. And now, setting these same directives via the X-Robots-Tag header, it is possible to extend SEO-related control over virtually every other type of content as well – PDFs, Word documents, Flash, audio, and video files – you name it!

Implementing X-Robots-Tag functionality for your own files is easy. For dynamically generated content, such as PHP files, place the following code at the very top of your page:

```
// instruct supportive search engines
// to index and cache the page
<?php header('X-Robots-Tag: index,archive'); ?>
```

Of course, the actual robots parameters will vary, depending on whether or not the content should be indexed, archived, etc.

To implement X-Robots-Tag directives for non-PHP files, such as PDF, Flash, and Word documents, it is possible to set the headers via HTAccess. Customize the following HTAccess script according to your indexing needs and add it to your site's root HTAccess file or Apache configuration file:

```
# index and archive specified file types
<IfModule mod_headers.c>
  <FilesMatch "\.(doc|pdf|swf)$">
    Header set X-Robots-Tag "index,archive"
  </Files>
</IfModule>
```

There is of course much more that can be done with X-Robots-Tag. For more information, see Taking Advantage of the X-Robots Tag at Perishable Press: <http://digwp.com/4>

```
http://domain.tld/product.php?item=leopard-skin-snuggy  
http://domain.tld/product.php?item=leopard-skin-snuggy&category=designer-snuggy  
http://domain.tld/product.php?item=leopard-skin-snuggy&trackingid=123&sessionid=456789  
http://domain.tld/product.php?item=leopard-skin-snuggy&referrer=chucknorris&id=snuggling-badass
```

Then placing the following canonical meta tag in the <head> section of each of the duplicate content URLs will tell the search engines that the duplicates all refer to the original URL:

```
<link rel="canonical" href="http://domain.tld/product.php?item=leopard-skin-snuggy" />
```

With WordPress, canonical meta tags accomplish the same thing: they tell search engines which version of your pages is the correct one to index. As you can imagine, this is a powerful tool in the fight against duplicate content, and there is an excellent plugin by Joost de Valk that makes implementing canonical tags a snap: <http://digwp.com/u/185>. There are also manual techniques for setting up canonical meta tags for WordPress, but the plugin really does a great job, and is the recommended way to do it.

Important Note

Currently, canonical meta tags only serve as a “hint” to search engines as to which page should be indexed. Chances are high that they will obey your specifications, but they reserve the right to take other factors into account and make their own decisions.

8.2.6 Use Excerpts for Posts

Another effective technique for preventing duplicate content is to simply use excerpts instead of full content on all non-single page views. This way, all of those pages that would otherwise include full copies of your post content will only show an excerpt instead.

To implement excerpts, replace any existing instances of the `_content` template-tag in your *non-single* theme files with the `_excerpt` template-tag. It’s as simple as that.

```
<?php the_content(); ?> =change to=> <?php the_excerpt(); ?>
```

Many sites use this technique with great results. Especially when used in conjunction with a canonical plugin, using excerpts instead of content is perhaps the easiest, most effective way of keeping duplicate content out of the search engine results.

8.3.1 Optimizing Permalink Structure

Permalink Optimization

Additional information on optimizing permalinks and URLs can be found at DigWP.com and CSS-Tricks:

<http://digwp.com/u/499>

<http://digwp.com/u/500>

One of the great things about WordPress is its “pretty” permalinks. Permalinks refer to a particular formatting of the URL structure for a site’s web pages. By default, WordPress generates dynamic page URLs of the format `http://digwp.com/index.php?p=123`, but then makes it super-easy to transform these structures into more user-friendly format, like `http://digwp.com/post-name`. Replacing the dynamic query-string URL format with pretty permalinks is a great way to optimize your site for the search engines. In this section, we examine some best practices and tips for crafting the perfect set of permalinks.

8.3.2 Default URLs vs. “Pretty” Permalinks

More on Permalinks

To refresh your memory and learn more about setting up permalinks, flip back to chapter 2.3.1.

An important factor to consider when optimizing your WordPress-powered site involves configuring your URL permalinks. When optimizing WordPress for the search engines, the first thing you want to do is set up permalinks for your site. Here is the general structure of default WordPress URLs:

`http://domain.tld/index.php?p=123`

After a fresh install of WordPress, all of your site’s URLs are represented in this “dynamic” query-string format. Every WordPress Page and Post is represented by a sequential series of IDs. Even the URLs for feeds, category archives, tag archives, and date archives are displayed in this dynamic format. But there are numerous reasons why this format is not the best choice for your site’s URLs. Default URLs are not very user-friendly and they do not take advantage of the value that Google and other search engines place on URL keywords.

Fortunately, WordPress provides a built-in solution in the form of automatically generated permalinks. Once enabled, permalinks – also referred to as “pretty” permalinks – transform WordPress’ default URLs into a wide variety of formats, depending on your configurational preferences (see chapter 2.3.1). Here are some examples:

http://domain.tld/name-of-post/
http://domain.tld/name-of-page/
http://domain.tld/category/boots/
http://domain.tld/author/fonzi/
http://domain.tld/2008/08/08/

See? No ugly query-string parameter confusing the issue – just straightforward, keyword-rich, “static” URLs. With permalinks enabled, your posts and page URLs may include the perfect blend of keywords while retaining their user-friendliness and readability.

8.3.3 Keep Permalinks Short

After deciding to use permalinks on your site, it is important to consider the best-possible format. In the WordPress Admin, under **Settings > Permalinks**, you will find several permalink configuration options, as well as a place to specify any custom structure you wish (see screenshot at right).

The general rule of thumb for establishing an optimal permalink structure is to keep your URLs as short as possible. This reasoning is based on research that suggests that URLs based upon “flat” directory structures fare better in the search results than do those with deeply nested, overly-convoluted architecture.

Even “Static” Pages are Dynamically Generated

Posts and Pages are treated differently in WordPress. Posts are considered to be part of a timeline that flows in chronological order, whereas Pages contain content that is removed from the normal flow of posts. Perhaps because of this difference, there is a common misconception that somehow Pages are not dynamically generated from the database. But this couldn’t be further from the truth. In fact, both Posts and Pages are stored in the database and called dynamically to the web page by PHP and the Post or Page template (which may also contain content). You can create *static* web pages and then link to them like any other document, but Pages created via WordPress store their content in the database.

The screenshot shows the WordPress admin interface. On the left is a sidebar menu with options: Dashboard, Posts, Media, Links, Pages, Comments, Appearance, Plugins, Users, Tools, and Settings (highlighted). Under Settings, there are sub-links: General, Writing, Reading, Discussion, Media, Privacy, Permalinks, and Login/logout redirects. The main content area is titled 'Permalink Settings'. It explains that WordPress uses web URLs with question marks and numbers by default, but offers better aesthetics with permalinks. It lists 'Common Settings' with radio buttons for: Default (http://digwp.com/?p=123), Day and name (http://digwp.com/2012/01/12/sample-post/), Month and name (selected, http://digwp.com/2012/01/sample-post/), Numeric (http://digwp.com/archives/123), Post name (http://digwp.com/sample-post/), and Custom Structure (a text input field containing /%year%/%monthnum%/%postname%/). Below this is an 'Optional' section with text explaining custom structures for category and tag URLs, and input fields for 'Category base' and 'Tag base'.

Switching from Date-Based Permalinks to Post-Name-Only

Although there are free plugins available for changing your permalinks, we prefer to handle URL redirection with Apache/HTAccess rather than PHP because it requires fewer system resources and is executed with greater speed. One final note before we begin: the purpose of this tutorial involves removing date information from all future permalinks and redirecting all pre-existing permalinks to their restructured counterparts. Thus, if you are setting up permalinks for a new blog (or one with only a few posts), the second part of this tutorial may not be required – a simple change of permalink structure via the WP Admin (as explained below) may be all that is needed.

Part 1: Update WordPress Options

The first step in creating “post-name-only” permalinks is to update your WordPress permalink structure in the Permalinks Options page of the WordPress Admin. Using the Custom structure option, customize your permalink structure like so:

```
/%postname%/
```

After choosing the post-name-only permalink structure, save the changes and test your pages. Remember to check different types of views – home, single, archive, page, search, etc. – to ensure that your new permalinks are working as expected. Once this is done, all *future* posts will feature the dateless permalink structure. In the second part of our tutorial, we will redirect all requests for *existing* versions of your URLs to their newly configured counterparts.

Part 2: Update .htaccess file

The second step in creating “post-name-only” permalinks involves modifying your root or subdirectory htaccess file to ensure that old permalinks are redirected to, and served as, your new permalinks. Examine each of the scenarios described below, determine which method applies to your specific setup, and implement the required steps.

Option 1: Remove “year/month/day”

This method removes the “year/month/day” portion of permalinks for blogs located within the domain's root directory. So, for example, if your old permalinks looked like this:

```
http://domain.tld/2008/08/08/post-title/
```

...then the code in this section will transform them into this:

```
http://domain.tld/post-title/
```

Locate your blog's permalink htaccess rules. Then, place the following code directly after the line containing the RewriteBase directive:

```
# remove year-month-day from permalinks
RewriteRule ^([0-9]{4})/([0-9]{1,2})/([0-9]{1,2})/([^\+]+)/?$ http://domain.tld/$4/ [R=301,L]
```

Option 2: Remove “year/month”

This method removes the “year/month” portion of permalinks for blogs located within the domain's root directory. So, for example, if your old permalinks looked like this:

```
http://domain.tld/2008/08/post-title/
```

...then the code in this section will transform them into this:

```
http://domain.tld/post-title/
```

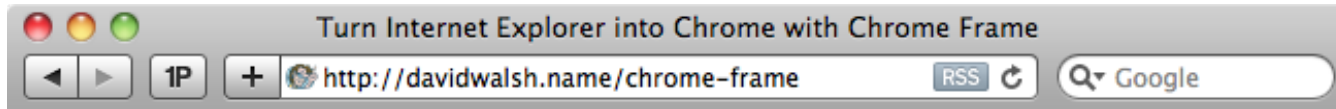
Locate your blog's permalink htaccess rules. Then, place the following code directly after the line containing the RewriteBase directive:

```
# remove year and month from permalinks
RewriteRule ^([0-9]{4})/([0-9]{1,2})/([^\+]+)/?$ http://domain.tld/$3/ [R=301,L]
```

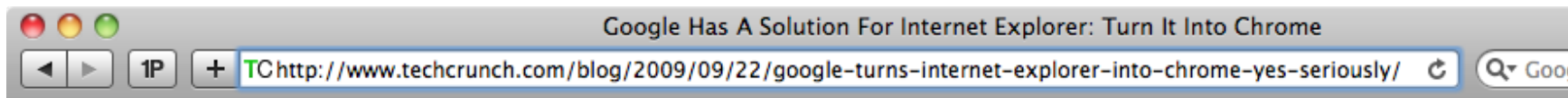
For either of these methods, remember to edit the “domain.tld” to match that of your own. No other changes are necessary. Test like crazy. After verifying that everything works as intended, sit back and enjoy your new optimized permalinks.

Here is a visual comparison of a flat directory-structure vs. a deeply nested directory structure:

Flat directory structure



Deeply nested directory structure



Thus, when it comes to your permalinks, the idea is similar: the shorter, the better. Thus, unless you have good reason for choosing otherwise, your permalinks should look more like this:

`http://domain.tld/super-awesome-post/`

...and less like this:

`http://domain.tld/2008/08/08/super-awesome-post/`

With this strategy in place, your URLs will feature a more concentrated mix of keywords while staying as “no-nonsense” and user-friendly as possible.

It's OK ...to Change Post Titles

Once you publish a post, the permalink of that web page is set. You don't want to change it, and if you do, you should ensure that a proper redirect is in place.

One thing that you definitely *can* change after publishing is the title. Many people mistakenly assume that the URL and the title are somehow interconnected and therefore can't be changed without screwing everything up.

So just an FYI, the title and URL of your pages are treated separately. That is why there are two different fields for these values: one for the title and one for the page “slug” (which serves as the permalink). So go ahead and feel free to change your post title anytime you like – it's totally fine.

Finding Duplicate Content

How much duplicate content has Google found on your site? The easiest way to find out is to spend some time searching around in Google. For example, do a “site:yoursite.com” search and examine the results.

First of all, how many results are returned? Compare that number with the total number of *unique* pages that you have. Is there a discrepancy?

Secondly, skip around the results and look for similar pages. Jump to the 10th results page and have a look. Then jump ahead five more pages and take a look. Do you see many similar titles and/or descriptions?

Lastly, check for duplicate content resulting from similar query strings by searching for “site:yoursite.com inurl:yourquerystring”. Again, examine the results. Watch for duplicate titles and similar titles with the same query string.

8.3.4 Maximize Permalink Keywords

One of the best reasons to switch to permalinks involves the ability to incorporate keywords into your URLs. Keywords make the Web go ‘round, and it is especially important to optimize your URLs accordingly. To illustrate the point, consider the difference between the following two URLs:

<http://yoursite.com/index.php?p=123>

<http://yoursite.com/search-engine-optimization/>

All else being equal, which URL do you suppose communicates more effectively the content of the page? Of course, the second URL structure, which utilizes WordPress’ permalink format and contains three keywords that search engines may use to help determine the meaning and purpose of the page. Permalinks enable you to take advantage of keyword-rich URLs that contribute to the overall SEO quality of your site.

As you write and publish your posts and pages, keep the keywords of your URLs in mind. An extremely useful feature of the WordPress **Write/Edit** screen is the ability to quickly edit permalinks. Beneath the post title there is a line that shows the current permalink for the post. To edit this, simply click on the “Edit” button to the right and change the permalink as needed.

The Difference Between `is_singular()` and `is_single()`

Permalink: http://digwp.com/2009/10/difference-between-is_singular-ar/

This feature enables you to customize your permalinks to be as terse and keyword-rich as possible. Here are some tips for optimizing your permalinks:

- The shorter the better
- Eliminate extraneous words (a, the, as, if, while, etc.)
- Include targeted keywords that correspond to the title and content of your post

When used wisely, these tips will ensure that your permalink URLs are configured for maximum results with both users and the search engines.

8.4.1 Scoring with Google

In addition to the SEO techniques discussed thus far, there are many other excellent ways to ensure a more optimized site. In this section, we will examine some of these ideas and explain how they may benefit WordPress users.

8.4.2 Content, Content, Content

As you go about optimizing your site for the search engines, keep in mind that the most important part of your site is the **content**. Make sure each page provides maximum value with the optimal amount of content. Also, when designing your theme, ensure that the content is placed as close to the top of the page as possible. Avoid placing tons of fancy scripts or other code before your content. If your theme design features a sidebar, ensure that it appears *after* your content in the source code.

A good way to view this strategy is to imagine what a search-engine spider such as the Googlebot sees when it crawls your pages. Looking at your source code, how much data must be crawled before actual content and keywords are encountered? Also, it has been reported that keeping the ratio of content to code as high as possible bestows additional SEO benefits. Food for thought!

Spider's-Eye View

A great way to evaluate how the search engines are “seeing” your web pages is to use an online SEO tool such as the Spider Test at seobook.com:

<http://digwp.com/u/224>

Simply enter your URL and the Spider Test will show you how your page looks to the search engines. Definitely helpful.

8.4.3 Detecting Duplicate Content

As discussed, duplicate content is a bad thing. Fortunately, there are many tools and techniques available for dealing with it. In addition to the previously discussed tactics (use of meta tags, nofollow attributes, robots directives, canonical tags, and excerpts), you can use an online duplicate-content checker, such as the one at Virante.com <http://digwp.com/u/225>, to test your site for many possible sources of duplicate content. For example, a recent check of this book's companion website, digwp.com, returned the following results when analyzed for duplicate content:

www vs. no www

An easy way to ensure proper redirection of your www URLs to their non-www versions is to add a quick slice of code to your root .htaccess file:

```
RewriteEngine On
RewriteBase /
RewriteCond %{HTTP_HOST}
!^domain\.tld$ [NC]
RewriteRule ^(.*)$ http://
domain.tld/$1 [R=301,L]
```

Simply change the two "domain.tld" parts to match your domain and you're all set.

For more information, or to redirect from non-www to www, check out Perishable Press:

<http://digwp.com/u/226>

- **WWW/NonWWW Header Check: FAILED**

Your site is not returning a 301 redirect from www to non-www or vice versa. This means that Google may cache both versions of your site, causing sitewide duplicate content penalties.

- **Google Cache Check: FAILED**

Google may have duplicate copies of pages on your site due to indexing both the www and non-www version of your site shows **207** pages cached, while the www version shows only **0** cached. Unless your site has subdomains, this often means that some duplicate content penalty may exist.

- **Similarity Check: SUCCESS**

You do not appear to have any pages omitted for being too similar in the top 1000 results of your site in Google.

- **Default Page Check: FAILED**

You have not standardized your default pages meaning the following versions of your url return a 200/OK Header, which may cause duplicate content issues. The following extensions work:

```
http://digwp.com/index.html
http://digwp.com/index.htm
http://digwp.com/index.asp
http://digwp.com/index.aspx
http://digwp.com/default.asp
http://digwp.com/index.php
http://digwp.com/
```

Complete Canonicalization for WordPress

As you can see here, there is much more to canonicalization than simple meta tags and default WordPress settings. Although this topic is beyond the scope of this book, you can learn more and set up complete WordPress canonicalization by visiting this post at Perishable Press: <http://digwp.com/u/227>

- **404 Check: FAILED**

You are not returning a 404 error on non-existent pages. This means Google could cache a large number of useless, duplicate pages on your site. Example: `http://digwp.com/afgahsdfjkahsdfjkasdhfjaksdhfasdjf.html`

- **PageRank Dispersion Check: SUCCESS**

You do not have a different pageranks [sic] for the non-www and www version of your domains.

As you can see, the information provided in such reports is extremely valuable. Rest assured, we will be correcting these issues just as soon as we finish the book!

8.4.4 Optimizing Heading Elements

Another important consideration involves the optimization of your pages' heading elements. Heading elements are used in page markup to denote the various headings on the page, and range in value from `<h1>` to `<h6>`. For example, a common markup scenario involves using `<h1>` elements for your site name, `<h2>` elements for each post title, `<h3>` elements for each sidebar section, and so on.

The problem with this strategy, at least from an SEO-perspective, is that search engines are reported to place slightly greater value on `<h1>` elements than the others. Thus, it is argued that, in order to better rank your individual Post Pages, you should forego the use of `<h1>` for your site's name and instead use it for your post titles. Likewise, subtitles should use `<h2>` tags, while the site name should be displayed via simple paragraph elements (`<p>`).

Yet, as with `nofollow` attributes, there are two sides to this issue. While using `<h1>` tags for post titles instead of the site name may provide an extra nudge from the search engines, it is not altogether semantically correct to do so. Without getting into a lengthy discussion about semantic markup, suffice it to say that "properly" marked-up Web documents are frequently thought of as displaying the name of the site via `<h1>` elements, with all subsequent sections following from there. Of course, this topic is hotly debated around the Web, but we thought it important enough to mention here.

They can be Different

In WordPress, the title of each post can be different on the web page and in the search results. Using a plugin such as All-in-One SEO or HeadSpace2, you can easily specify a custom title for the <title> tag

Personally, I have been known to use <h1> elements for *both* the site name and post titles. And with the advent of HTML5, documents may be structured as to include multiple heading elements of *any* type throughout the page.

8.4.5 Optimizing Title Tags

In a similar vein to heading elements, title tags are another area for potential SEO improvements. Title tags are declared via <title> elements in the <head> section of your WordPress pages and are displayed in search results, bookmarks, and in the title bar of browsers. Generally, but not necessarily, the title information included in the <head> section conveys the same information used for the Post or Page title.

[allmusic \(\(\(The Chemical Brothers > Overview \)\)\)](#)

The act with the first arena-sized sound in the electronica movement, the **Chemical Brothers** united such varying influences as Public Enemy, Cabaret Voltaire ...

[www.allmusic.com/cg/amg.dll?p=amg&sql=11...](#) - [Cached](#) - [Similar](#)

allmusic (((The Chemical Brothers > Overview)))



all

<http://www.allmusic.com/cg/amg.dll?p=amg&sql=11:avfqxqrgldke>



Google

<head>

```
<title>allmusic  ((( The Chemical Brothers > Overview  )))</title>
```

```
<meta name="keywords" content="allmusic, music reviews, new releases, artists biography">
```

```
<meta name="description" content="allmusic, music reviews, new releases, artists biography">
```

</head>

Three different uses for the title tag (from top): search-engine results, browser title bar, and source code. Of course, the title of the post may be different.

In the current version of WordPress, titles are displayed using the wp_title() template tag. By default, the wp_title() tag displays the following information, depending on page type:

- For the “Home” page — `wp_title()` displays no output
- For individual pages — `wp_title()` displays the page title
- For single post views — `wp_title()` displays the post title
- For archived post views — `wp_title()` displays no output
- For date-based archives — `wp_title()` displays the year and/or month
- For category archives — `wp_title()` displays the category title
- For author archives — `wp_title()` displays the user’s public username
- For 404 error pages — `wp_title()` displays no output
- For search results — `wp_title()` displays no output
- For tag archives — `wp_title()` displays the tag name

Very basic stuff, however, you can always beef things up by adding your site name or tagline like this:

```
<title>
    <?php wp_title('&raquo;', 'true', 'right'); ?><?php bloginfo('name'); ?>
</title>
```

Which will output the default page title followed by the name of your blog or site:

Title of Displayed Page » Blog Name

This simple implementation of the `wp_title()` tag will ensure that your site name is associated with all of your content. Even better, it will take care of those pages where no output is returned by displaying your site name.

Way back in chapter 3.3.4, we showed you how to create far-nicer page titles. While back then we were concerned about users, usability, and aesthetics, those same exact concepts are also important for SEO. Please refer to that section for the code necessary to create excellent and search-engine friendly title tags.

8.4.6 The nofollow Wars

Several years ago, Google initiated the widespread implementation of nofollow attributes for distrusted content, paid links, and crawl prioritization <http://digwp.com/u/243>. Despite Google's best efforts to gain full compliance with their recommendations, many people felt like it was a bad idea that would ultimately exacerbate the very problems it was designed to solve <http://digwp.com/u/244>.

On a more personal front, many bloggers felt that denying commentators their just rewards for leaving a comment simply wasn't fair. Since that time, there has been a growing movement of "dofollow" bloggers who defy Google by removing nofollow attributes included by default on WordPress sites.

By default, all links related to WordPress comments – namely, author and intra-comment links – are generated with nofollow attributes included. While there are many ways to remove them (see chapter 7.7.3), and thus jump on the *dofollow* bandwagon, you may want to consider the possible ramifications of doing so.

WordPress + nofollow

For a ton of excellent tips and techniques for removing, editing, and customizing your WordPress nofollow links, check out this series of posts:

<http://digwp.com/u/245>

Many well-ranked WordPress sites that feature these so-called *dofollow* comment links are included on spam and other types of *dofollow* lists across the Web. Further, it has been reported that Google actually *penalizes* sites for **not** adhering to their recommended nofollow guidelines <http://digwp.com/u/417>. Even so, many ultra-hip and user-minded individuals continue to face perpetual spam comments, loss of page rank, and potential penalties while supporting the *dofollow* cause.

As discussed in chapter 7.7.3, we now know that using nofollow to conserve page rank is no longer an effective tactic. Thanks to changes in the way Google treats nofollow links, any link equity that would have been conserved due to nofollow links will now simply vanish into nothingness.

Yet even with this new information, nofollow is still a useful tool for managing duplicate content, keeping certain pages out of the index, and protecting your site against comment links pointing to questionable sites.

8.4.7 Fixing Broken Links

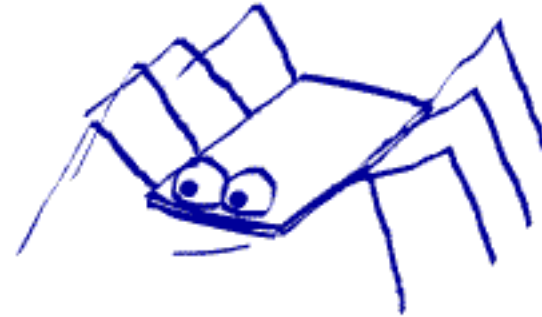
Any seasoned webmaster or blogger knows how quickly the Web changes. Dead links appear all the time, and can really decrease the inherent and perceived value of your site. If visitors and search engines are constantly running into dead links, what message are they receiving about the accuracy, currency, and relevancy of your site? Whether inbound or outbound, links need to be kept current and healthy in order to ensure the best possible presentation of your site.

When it comes to checking for dead or broken links on your site, there are a few good strategies. The most time-consuming and difficult involves keeping track of things manually. This may work for sites with small amounts of content, but for anything larger, you are going to want to automate the process as much as possible. There are online services such as the one provided by the W3C <http://digwp.com/u/246>, which checks *single* web pages, or, even better, services such as the one provided by dead-links.com <http://digwp.com/u/247>, which scans your *entire site* and reports back with complete results. Of course, there are many, many other free and paid link-checking services available on the Web, just consult your nearest search engine!

Alternately, you may wish to install a plugin to keep track of broken and changing links. If so, the **Broken Link Checker** plugin by Janis Elsts <http://digwp.com/u/248> may be just the ticket. After installing, the Broken Link Checker plugin will monitor your blog for broken links and provide a full report in the WordPress Admin. The plugin runs automatically in the background while you are logged in to the WordPress Admin. The plugin is highly configurable and definitely worth a look.

Finally, if none of the above options are for you, or if you want additional link-

Dead-Links.



Optimiz
Increas
Release
www.prwel



Start URL: <http://digwp.com/>



Maximum execution time: **45 minute**

Patience, Young Jedi

Checking your entire site for links at a link-checking service such as dead-links.com can take awhile, but the results are usually worthwhile.

checking capabilities, there are a couple of other excellent options available. If you are a **Firefox** user, you can install the free **LinkChecker** add-on by Kevin Freitas <http://digwp.com/u/249>. Once installed and enabled, Kevin's LinkChecker will check the validity of links on any page with a single click.

For offline link-checking analysis, check out **Xenu's Link Sleuth** <http://digwp.com/u/250>, which is a powerful piece of software that checks sites for broken links from your local machine. Xenu's Link Sleuth is reported to be a bit more robust in terms of the types of links that can be checked, including "normal" links, images, frames, plugins, backgrounds, local image maps, style sheets, scripts, and even Java applets. The program provides continuous, detailed reports and features plenty of awesome configurational options.

mindfeed.org XML Sitemap

URL

<http://mindfeed.org/>

<http://mindfeed.org/family-hatred/>

<http://mindfeed.org/im-loving-it/>

<http://mindfeed.org/just-nod-if-you-can-hear-me/>

<http://mindfeed.org/interpretation-is-relative/>

<http://mindfeed.org/doom-and-gloom/>

<http://mindfeed.org/give-up-dont-give-up/>

<http://mindfeed.org/blog-your-voice/>

<http://mindfeed.org/like-cyberspace-like-meatspace/>

<http://mindfeed.org/thinking-logically-about-shooting>

8.4.8 Using a Sitemap

Perhaps the easiest way to increase the SEO-value of your site is to create a sitemap. Sitemaps provide a structural overview or map of your entire site in XML (eXtensible Markup Language) format. Sitemaps help search engines navigate your site and find new and updated content. Many WordPress sites take advantage of Arne Brachhold's free plugin, **Google XML Sitemaps** <http://digwp.com/u/251>. Once installed and configured, the XML Sitemaps plugin works quietly behind the scenes to maintain a current sitemap for your site. The plugin is highly configurable, and includes options to ping various search engines after each update to your sitemap. The main benefit to using a sitemap plugin is that you won't have to keep manually recreating your sitemap – it's all automatic!

For sitemaps *without* a plugin, there are a good number of free online services available. For example, XML-Sitemaps.com <http://digwp.com/u/252> provides a free service whereby you enter the URL for your site, grab a snack,

and download your sitemap. There is a 500-page limit for the online generator, but the site also provides an unrestricted standalone generator for a nominal fee. Keep in mind, however, that online sitemap services require that you re-generate your sitemap every time you update your site – it is not an automated process. Especially if your site is updated frequently, this could be a serious drag.



8.4.9 SEO-Related plugins

One of the main benefits of using WordPress is its extensibility. The WordPress developer community is one of the greatest in the world, endlessly producing useful plugins, thematic functions, workarounds and hacks. Basically, with WordPress, you can do virtually anything. And the realm of SEO is no exception. Here is a list of some of the most useful and beneficial SEO plugins currently available to the thriving WordPress community:

- **All-in-One SEO** - <http://digwp.com/u/257>
"Optimizes your Wordpress blog for Search Engines" by enabling canonical URLs, navigational links, extended plugin functionality, integration with e-commerce sites, nonce security, custom meta tags, and tons more. Perhaps the most widely used SEO plugin for WordPress.
- **HeadSpace2 SEO** - <http://digwp.com/u/258>
"All-in-one metadata manager that allows you to fine-tune the SEO potential of your site." You can configure metadata for posts, pages, categories, and just about everything else. You can define tags, descriptions, titles, "more" text, as well as custom themes, plugins, scripts, CSS and much more.
- **Redirection** - <http://digwp.com/u/259>
Manages 301 redirections, keeps track of 404 errors, and cleans up any loose ends your site may have. Perfect for migrating pages from old sites or changing the directory of your WordPress installation.
- **Robots Meta** - <http://digwp.com/u/260>
Enables you to specify which pages may be crawled and indexed by search

Short URLs

With the rise of Twitter and other micro-blogging media sites, the importance of providing short URLs to your visitors can mean the difference between getting your content shared around the Web or having it just sit there. For some great techniques on providing short URLs (for Twitter and other social media sites) check out the popout in chapter 2.3.2 and the following post at Perishable Press:

<http://digwp.com/u/256>

engines. By customizing your pages' meta tags, you have full control over every type of page on your site – all from within the comfort of the WordPress Admin.

- **SEO Smart Links** - <http://digwp.com/u/261>

Helps you create a more robust internal linking structure for your site by automatically linking any specified terms with relevant pages, posts, tags, or anything else. Includes options for nofollow links and “blank” targets (where linked pages open in a new tab or window).

These plugins will enable you to devise and implement the optimal SEO strategy for your WordPress-powered site. But the list doesn't stop here! Check out the official WordPress plugin repository <http://digwp.com/u/262> for a more comprehensive list. And, if you don't find it there, be sure to search the Web for anything you may need – there is an endless river of SEO knowledge and tools that you can tap into.

8.5.1 Tracking the Success of Your Site

As you optimize your site for the search engines, you will inevitably want to monitor and track its statistical progress. Fortunately, there are many analytical and statistical tools from which to choose. Here is an overview of some of best.

8.5.2 Statistical Plugins

- **WP-ShortStat** - <http://digwp.com/u/263>

The WP-ShortStat plugin provides essential site statistics for your WordPress-powered site, including visits, hits, referrers, search terms, and more. WP-ShortStat is a lightweight stats plugin that is easy to install, configure and use. It simply runs quietly in the background, gathering statistics and reporting the results in a convenient page in the WordPress Admin. And best of all, it's free.

- **StatPress** - <http://digwp.com/u/266>

StatPress is a multi-language stats plugin that collects information about visitors, spiders, search keywords, feeds, browsers, and more. StatPress works in

Analytics Roundup

StylizedWeb posted a much-needed roundup of 5 Rocking Google Analytics Plugins:

<http://digwp.com/u/474>

“real time,” enabling you to observe which pages your users are visiting as they navigate your site. Admin statistics page conveniently located in the WordPress dashboard menu.

- **Google Analytics Dashboard** - <http://digwp.com/u/267>

Google Analytics Dashboard gives you the ability to view your Google Analytics data in your Wordpress dashboard. Sweet options.

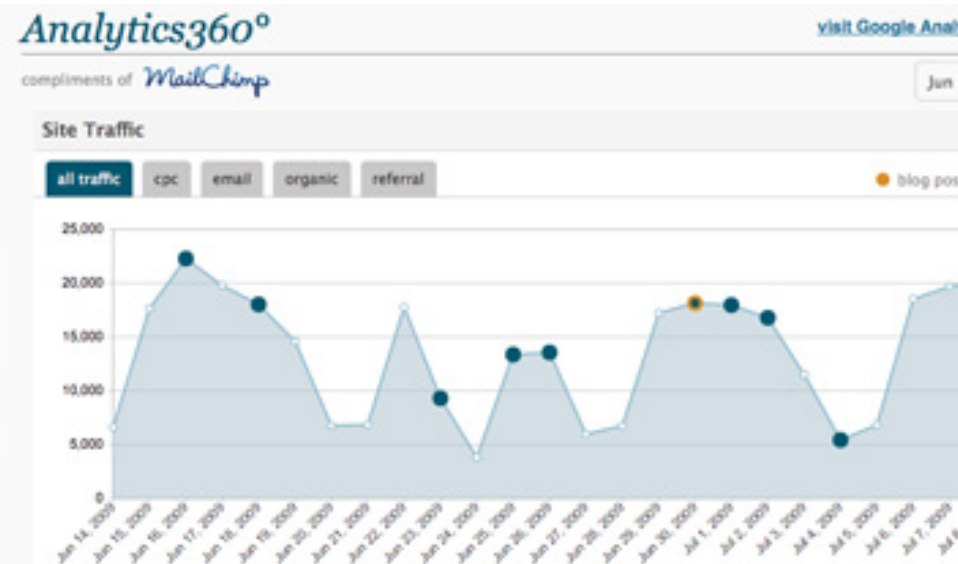
- **Analytics360** - <http://digwp.com/u/268>

“MailChimp’s Analytics360 plugin allows you to pull Google Analytics and MailChimp data directly into your dashboard, so you can access robust analytics tools without leaving WordPress.” Features include visual statistics display, comparative growth analyses, and robust referrer information.

And these are just the tip of the iceberg. We wish we could review all of the best, but there are simply too many of them. Need proof? Check out the vast menu of statistical plugins available at the WordPress Codex: <http://digwp.com/u/269>.

8.5.3 Mint Stats

Shaun Inman’s **Mint** is sweet indeed. User-friendly, powerful, and extensible, Mint is PHP/MySQL-based statistical application that you host on your own domain. Mint gives you complete control over your site’s statistics. Out of the box, Mint keeps track of all the essentials: hits, visits, referrers, and much more.



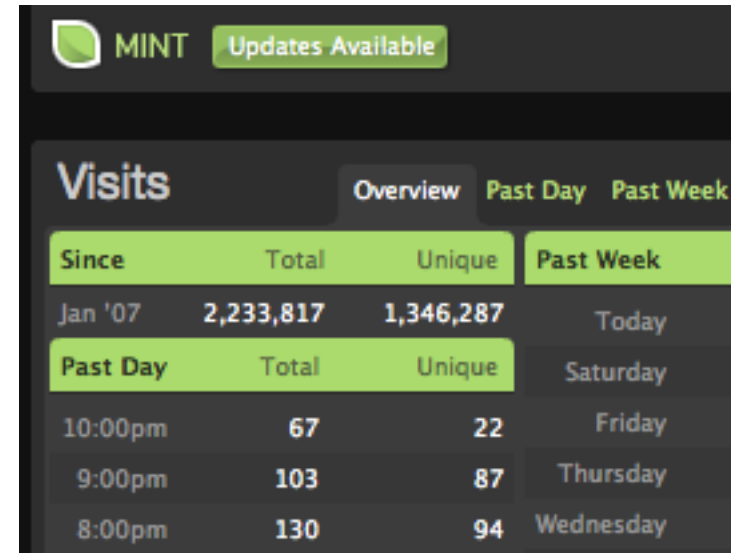
Controlling the Size of Your WP-ShortStat Database Table

Using the excellent WP-ShortStat plugin can quickly bloat the size of your WordPress database. It collects a *lot* of data on many types of statistics. Over time, or more quickly for busier sites, the size of WP-ShortStat table can add many megabytes of data to your database. To help control this, you can routinely execute the following SQL command:

```
DELETE FROM `wp_ss_stats`  
ORDER BY `id` ASC LIMIT n
```

Simply check the total number of records in your wp_ss_stats table and enter a value of “n” that corresponds to the number of entries you would like to delete. Remember to backup your database beforehand, and also change the database prefix in the SQL query from “wp_” to any custom prefix that you might be using. For complete details, check out Perishable Press: <http://digwp.com/u/264>

Beyond its default functionality, Mint is extensible through a large variety of extensions, dubbed “peppers.” A great amount of statistical functionality is available through both free and paid peppers, including cool things like real-time visitor monitoring, so you can watch as visitors arrive and navigate around your site. Other peppers enable tracking of visitors’ geographical location, screen size, search terms, as well solid integration of third-party services like Twitter, Vimeo, and Facebook. Mint is available at <http://digwp.com/u/270> for around \$30. Highly recommended.



The screenshot shows the MINT dashboard with a dark theme. At the top, there's a 'MINT' logo and a green button that says 'Updates Available'. Below this, the 'Visits' section is active, showing an 'Overview' of visitor data. The table displays statistics for 'Jan '07', 'Past Day', and 'Past Week'. The 'Past Day' section is further broken down by time of day (10:00pm, 9:00pm, 8:00pm) and day of the week (Friday, Thursday, Wednesday).

Since	Total	Unique	Past Week
Jan '07	2,233,817	1,346,287	Today
Past Day	Total	Unique	Saturday
10:00pm	67	22	Friday
9:00pm	103	87	Thursday
8:00pm	130	94	Wednesday

8.5.4 Google Analytics

Google Analytics Plugin

I often joke about how many different plugins there are for adding the Google Analytics code to your site. Yet among the clones, there is one that stands out: Google Analytics for WordPress not only adds the required GA code, it also tracks and segments all outbound links, tracks AdSense clicks, adds extra search engines, tracks image search queries and it will even work together with Urchin. Best of all, it's by Joost. Check it out:

<http://digwp.com/u/271>

One of the most popular statistical applications available today is Google Analytics (GA). Google Analytics tracks a *mind-boggling* amount of data for as many websites as you can throw at it. The service requires a Google account and the placement of a small snippet of code in the <head> of your site's pages:

```
<script type="text/javascript">
  var _gaq = _gaq || [];
  _gaq.push(['_setAccount', 'UA-XXXXX-X']); // replace UA-XXXXX-X with your web property ID
  _gaq.push(['_trackPageview']);
  (function() {
    var ga = document.createElement('script'); ga.type = 'text/javascript'; ga.async = true;
    ga.src = ('https:' == document.location.protocol ? 'https://ssl' : 'http://www') +
    '.google-analytics.com/ga.js';
    var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(ga, s);
  })();
</script>
```

Once in place in your header.php file, this GA tracking code will enable Google to track and monitor your site's statistics. After an initial data-collecting period (usually a day or so), statistical results may be checked via the GA area of your

Google account. Once logged in, you will find a plethora of statistics for your site. As of 2012, GA provides real-time monitoring of visitors throughout your site. The GA service is comprehensive, consistent, reliable, and best of all free of charge.

8.5.5 Other Metrics

Other ways to measure the statistical metrics of your site include services such as Technorati, Alexa, and FeedBurner. Let's have a look at each one.

Technorati - <http://digwp.com/u/273>

Technorati is a free blog networking service that enables users to favorite your blog and share it with other people in their network. In providing this service, Technorati provides very general, broad statistics concerning your blog's reach and influence. After registering and "claiming" (verifying) your blog, Technorati begins to measure its success according to the Technorati ranking system, which most likely includes factors related to how people share and favorite your content.

Alexa - <http://digwp.com/u/274>

An Amazon.com company, Alexa uses its user-installed toolbar to gather statistical data on various metrics of internet-browsing behavior. Statistical data is collected from Alexa toolbars and transmitted back to the Alexa website. This data is then stored, analyzed, and reported at the company's website. The amount of statistical data that Alexa provides is impressive, but there is concern over the bias of its opt-in, self-collecting method of gathering data.

FeedBurner - <http://digwp.com/u/275>

As discussed in chapter 6.4.1, FeedBurner is a feed delivery service that measures many different statistical aspects of your feeds. While there are currently several viable alternatives for monitoring your feed statistics, FeedBurner continues to provide quality statistical data to millions of users around the Web. The service is free, requiring only account registration and proper configuration.

GA Fail?

As great as it is, one of the downsides of Google Analytics is that it requires JavaScript to work. This basically means that GA can't collect data from visitors who are not JavaScript-enabled. While this sucks, one thing that you can do is track how many users are visiting without JavaScript. Check out this plugin by Remy Sharp:

<http://digwp.com/u/272>

SEO Common Sense

You don't need to break your back (or your wallet) trying to get good SEO advice for your site. The Common Sense SEO Checklist breaks it all down and delivers a clear, practical guide to optimizing your sites:

<http://digwp.com/u/276>

8.6.1 Closing Thoughts on SEO

Perhaps the most important – and challenging – part of your SEO strategy involves obtaining high-quality and relevant links to your site. Without links, your site is pretty much *invisible* to search engines. If you think of the Web as consisting of many streams of traffic flowing to different sites, it makes sense that those sites with the largest streams flowing into them are the most popular sites on the Web.

The trick, of course, is actually *getting* links. We don't have room in this book for an in-depth SEO discussion, but we can point you in the right direction. Here are some of the Web's top SEO sites that will get you up to speed in no time:

- **SEOMoz** - <http://digwp.com/u/255>
- **SEOBook** - <http://digwp.com/u/254>
- **Search Engine Land** - <http://digwp.com/u/253>

Another effective SEO strategy, as mentioned earlier in this chapter in section 8.3.1, involves optimizing your permalinks to keep them as accurate and concise as possible. In general, the shorter the URL, the better, at least from an SEO-perspective. Shorter URLs represent “flatter” directory structures, which have been reported to facilitate the spidering process and thus promote increased availability of your content to the search engine. Likewise for any non-permalink resources you may provide on your site, avoid nesting too deeply within your directory.

Another useful SEO strategy to consider involves maximizing the meta information associated with your links, images, and other resources. When writing the markup for these key types of content, include the recommended `alt`, `title`, and `desc` attributes, and give them useful, descriptive, and keyword-rich values. Doing so fortifies the inherent value of these items, thereby fostering findability and relevance via the search engines. Integrating good meta information as you build your WordPress site is a beneficial habit to develop early in the game.

Last but not least, **the performance of your site** has an *enormous impact* on the quantity and quality of traffic that it receives. Slow-loading pages, broken links, and missing images, for example, are serious red flags to both human users and the search engines. In chapter 9, we provide excellent strategies for optimizing the performance of your WordPress site, enabling you to deliver your content as quickly and accurately as possible.

With these things in mind, we encourage you to use the information in this chapter to help formulate your own SEO strategy. There is no “one-size-fits-all” approach to optimizing your site for the Web, so *do your own thinking* and research the topic as thoroughly as possible.

As you continue developing your SEO skills, your ideas and strategies will evolve and improve. Despite what you may hear, learning the fine art of SEO doesn’t happen overnight, so begin with the basics and use them as a foundation on which to grow and develop your own optimization techniques.

If nothing else, remember that “content is king,” linking to other sites is a good thing, and a little common sense goes a long way to making the right decisions.

Into the future...

Now that we have seen some of the many ways to optimize our WordPress-powered sites for the search engines, it’s time to prepare for the inevitable onslaught of traffic that will be heading your way. To ensure that everyone enjoys the *best possible experience* at your site, it’s vital to deliver your web pages *quickly and securely*. In the next and final chapter of the book, we dig into the essential practice of optimizing your site’s performance and maintaining a healthy site far into the future.

The best design tool is a long eraser with
a pencil at one end.

– **MARTY NEUMEIER**

9

Maintaining a Healthy Site

9.1.1 Keeping Your Site Healthy

As easy as WordPress makes it to set up and operate blogs, e-commerce sites, and advanced content-management systems, there is still a significant amount of work that needs to be done in order to establish and maintain a healthy, thriving website. In this chapter, we explore some of the many ways that we can keep our WordPress installations running optimally, effectively, and securely.

9.1.2 Securing WordPress

For any website, a strong security strategy is an absolute necessity. As the world's most popular blogging software, WordPress is a *huge* target for malicious attacks. Fortunately, the popularity of WordPress is also its greatest *strength*, as thousands of top-notch developers continually release and upgrade a mind-boggling number of useful plugins, themes, functions, and techniques. When combined with key methodology and other information, these tools equip us to secure WordPress and defend thoroughly against malicious activity, spam, and other threats.

One of the first things you should do when securing your WordPress site involves ensuring that your files are well-protected. There are numerous ways of accomplishing this, including configuring proper file permissions, disabling directory views, locking down sensitive files, and staying current with new versions of WordPress. Let's examine and expound upon each of these strategies in greater depth.

9.1.3 Setting Secure File Permissions

Although it is likely that your web host has already configured the optimal permission settings or access rights for your directories and files, it is a good idea to examine each of them to ensure proper security. There are several ways to do so, including checking permissions directly on the server through internal file listings, or else by using a good FTP client to connect and check remotely. Many hosting accounts these days provide directory listings that visibly display the specific permission level for each folder and file within the file system. Here is an example showing Plesk's default directory listing:



dev	<dir>	Aug 26, 2009 01:20:58 PM	drwxr-xr-x	root	root				
etc	<dir>	Sep 17, 2009 03:03:00 PM	drwxr-xr-x	root	root				
home	<dir>	Jul 23, 2009 04:58:06 PM	drwxr-xr-x	root	root				
lib	<dir>	Aug 26, 2009 02:09:35 PM	drwxr-xr-x	root	root				
media	<dir>	Mar 29, 2007 01:00:12 PM	drwxr-xr-x	root	root				
mnt	<dir>	Mar 29, 2007 01:00:12 PM	drwxr-xr-x	root	root				
opt	<dir>	Mar 29, 2007 01:00:12 PM	drwxr-xr-x	root	root				
proc	<dir>	Aug 26, 2009 01:20:56 PM	dr-xr-xr-x	root	root				
root	<dir>	Aug 14, 2009 02:15:23 PM	drwxr-x---	root	root				
sbin	<dir>	Aug 26, 2009 02:07:53 PM	drwxr-xr-x	root	root				
selinux	<dir>	Mar 29, 2007 01:00:12 PM	drwxr-xr-x	root	root				
srv	<dir>	Mar 29, 2007 01:00:12 PM	drwxr-xr-x	root	root				

Server administration tools like cPanel will be similar. Each file's permission setting determines the way in which the file is allowed to be used. On Unix-ish systems, there are three specific permissions settings:

- **Read** - grants the ability to read directories' file contents and file names
- **Write** - grants the ability to modify file or directory names and content
- **Execute** - grants the ability to execute or process the contents of files and traverse the files within a directory

On your server, your WordPress files should be owned by your user account and writable by your username. In general, the proper file permissions are already in place for self-installed versions of WordPress. Most of the time, you will not need to concern yourself with file permissions. Exceptions include situations where you are troubleshooting permission errors, configuring files or directories involved with plugins, or ensuring settings for security purposes. In other words, unless there is a clear and specific need to modify your file permissions, you probably do not need to do so. Even so, when it comes to the security of your site, it is good to make sure that proper permissions settings are in place. Here are a few things to remember:

Check It, don't Wreck It

Before worrying too much about your file permissions, consult your host and ask about their permissions policy. Chances are, you don't need to change anything, but you should still keep an eye on things just in case.

Core WordPress files and directories

All core WordPress files should be writable only by your server user account. The default permissions settings for all WordPress files is 0644. The default permissions settings for all WordPress directories (the folders themselves) is 0755. These settings ensure that the core is *writable* only by the user account and *readable* by the web server and everyone else. For more information, see the Codex <http://digwp.com/u/308>.

Root HTAccess file

The WordPress Codex suggests setting looser permissions for your .htaccess file in order to make it writable by WordPress. The purpose for this involves WordPress' automatic creation of permalink rules, making it easy for users. A better idea is to leave your .htaccess permissions set at a restrictive level, and then manually add the required permalink directives. For more info, see <http://digwp.com/u/278>.

.htaccess made easy

For a complete .htaccess guide that goes in-depth on securing your website, check out Jeff's book, .htaccess made easy.

<http://htaccessbook.com/>

Theme files

Normally, theme files should possess the same permissions as other core WordPress files, however, if you want to use WordPress' built-in theme editor, you may need to set permissions to make your theme files group-writable. This may not be necessary however, and you should test for editing functionality before changing anything.

Plugins

Although many plugins work fine with WordPress' default file permissions, there are some that require write access to various files and/or directories. In this case, one of the most common requirements is that the entire wp-content directory be made writable. If so, begin with a permission setting that is as restrictive as possible, such as 755. If this doesn't provide sufficient privileges on your server, then you may need to use 777, which is not recommended due to reasons explained below.

Directories

There are a couple of directories that also may need to be writable by WordPress. The first is the wp-content/cache directory, which needs to be writable in order for caching to work properly. The other is the wp-content/uploads directory, which needs to be writable in order for users to upload their content.

For files and directories that require additional permissions, keep in mind that it may not be necessary to use 777. Although on some servers, 777 permissions is the *only* thing that works, there are many cases where a less-permissive setting will do the job. By setting 777 permissions for files or folders, you are opening your site up to attackers who may exploit permissive settings to upload malicious scripts, accessing your database information, and ultimately gaining control of your entire site. For more information on configuring WordPress file permissions, see the "Hardening" section in the WordPress Codex <http://digwp.com/u/277>.

May I have Permission?

When it comes to changing permissions, it's considered best practice to use the most restrictive settings possible. So, when setting permissions, begin with the most restrictive and work your way up until the desired functionality is possible. **Allow only what is required.**

9.1.4 Disabling Directory Views

Another way to increase the security of your WordPress installation is to disable directory views. Many hosts disable directory views on their servers by default, however we want to make sure. When directory views are enabled, any directory that does not include some sort of an index file (e.g., index.html, index.php, etc.) will

openly display a list of all included files, thereby exposing them to everyone. Here is a typical example of the wp-content folder with directory views enabled:

Index of /wp-content

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
[DIR] Parent Directory		-	
[DIR] themes/	26-Oct-2009 02:20	-	
[DIR] plugins/	26-Oct-2009 02:20	-	
[DIR] images/	26-Oct-2009 02:20	-	
[DIR] backups/	26-Oct-2009 02:20	-	

Destroy My Site, Please

This may be the first thing an attacker sees before completely destroying your website.

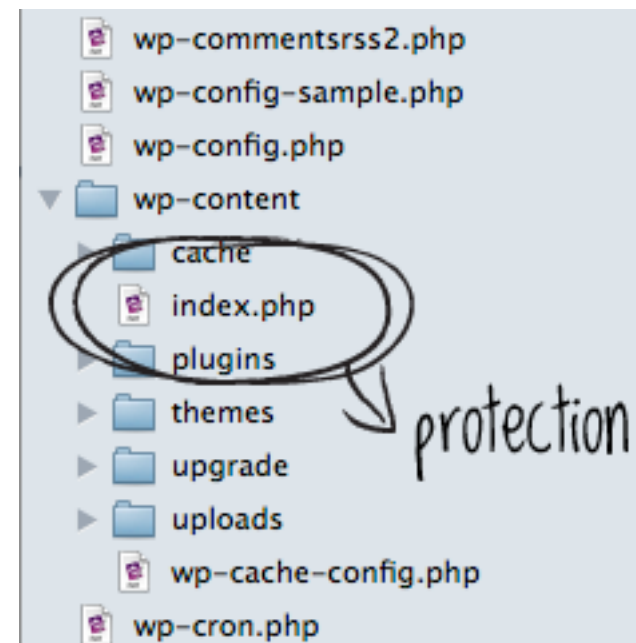
Obviously, this is a huge security risk. If malicious individuals were to gain access to your wp-config.php file, for example, they could easily access your database and steal sensitive data, destroy your entire site, and otherwise ruin your life.

Fortunately, disabling default directory views is drop-dead easy. Simply open your root .htaccess file or Apache configuration file and add the following line, preferably near the top of the file (although it will work anywhere):

```
Options -Indexes
```

Alternately, if .htaccess is not an option, you may prevent directory listings by simply adding a blank index.html or index.php document to each of your WordPress directories. While most versions of WordPress include such "faux" index files by default for the "wp-admin", "wp-content", and "wp-includes" directories, there are still many subdirectories that should be protected, one way or another.

After creating pseudo index files in these directories, blank pages will then appear instead of a file listing whenever someone accesses the directories via the Web.



9.1.5 Forbid Access to Sensitive Files

Protecting the wp-config.php file

In addition to disabling directory views, we also want to forbid direct, external access to critical files within the WordPress file system. First and foremost, we want to protect the wp-config.php file. Each installation of WordPress requires this file in order to connect to the database, set various preferences, and accommodate custom settings.

As you can imagine, if some nefarious intruder were to gain access to this file, your entire site – if not the *entire server* – would be severely compromised. To prevent this sort of tragedy from happening, let's secure this file with a little HTAccess magic. Here is one way to do it:

```
# SECURE WP-CONFIG.PHP
<Files ~ wp\-config\.php>
  Order Deny,Allow
  Deny from all
  Allow from 123.456.789
</Files>
```

Place that code into your site's root .htaccess file or Apache configuration file and enjoy immediate protection. This code works by denying all requests for the specified file, wp-config.php, except for those made from your specific IP address, which is specified in the "Allow from" directive in the fifth line. The IP address in this line should be edited to match that of your own. Note that you may allow access to additional IP addresses or even an entire IP range as follows:

Allow access to multiple IP addresses

```
# SECURE WP-CONFIG.PHP
<Files ~ wp\-config\.php>
  Order Deny,Allow
  Deny from all
```

wp-config Tricks

For an awesome collection of configuration tricks for your wp-config.php file, check out these articles at our website:

<http://digwp.com/u/279>

<http://digwp.com/u/280>

What is my IP Address?

If you don't already know it, the easiest way to determine your IP address is to visit the following URL in your browser:

<http://digwp.com/u/281>

```
Allow from 123.456.789
Allow from 456.789.123
Allow from 789.123.456
# additional IP addresses
</Files>
```

Allow access to a range of IP addresses

```
# SECURE WP-CONFIG.PHP
<Files ~ wp\~config\.php>
    Order Deny,Allow
    Deny from all
    Allow from 123.456.
</Files>
```

To allow multiple IP access, simply replicate and edit as many “Allow from” directives as necessary. To allow a *range* of IP addresses, use a partial IP address such that any matching IPs will be allowed. Incidentally, allowing a range of IP addresses is a good way to allow access for a dynamic IP address.

Protecting the install.php file

Used during the WordPress installation process, the `install.php` file is used to specify your blog title and email address. Once this information is entered, WordPress displays a username and password for the admin account.

Unfortunately, during certain database-related issues, WordPress may assume that it has not yet been installed and will load the `install.php` file. Although this situation is relatively rare, it can compromise your site if not prevented. Fortunately, there are several ways to protect your site.

HTAccess Protection for Dynamic IPs

If you are working from a dynamically generated IP address, you can edit the Admin-protect code to allow for the changing number.

All you need to do is omit the last octet from the IP address and Apache will match *any* IP that begins with the existing octets.

For example, if your dynamic IP ranges from 123.456.789.1 to 123.456.789.255, this code would account for any IP that you may have:

```
<FilesMatch ".*.*">
    Order Deny,Allow
    Deny from all
    Allow from 123.456.789
</FilesMatch>
```

- **Fix #1: Just nuke it**

Simply delete the wp-admin/install.php file entirely. It is not needed after installation.

- **Fix #2: HTAccess to the rescue**

Place the following slice of .htaccess into your site's web-accessible root directory to prevent access to your install.php file:

```
# PROTECT install.php
<Files install.php>
  Order Allow,Deny
  Deny from all
  Satisfy all
</Files>
```

- **Fix #3: Replace it with something safe and useful**

Replace the insecure version of the file with something secure and informative by following these quick steps:

1. Rename the original install.php to something like "install_DISABLED.php" or whatever.
2. Create a new file named "install.php" and add the following code:

```
<?php // install.php replacement page @ http://digwp.com/u/597 ?>
<?php header("HTTP/1.1 503 Service Temporarily Unavailable"); ?>
<?php header("Status 503 Service Temporarily Unavailable"); ?>
<?php header("Retry-After 3600"); // 60 minutes ?>
<?php mail("your@email.com", "Database Error", "There is a problem with teh database!"); ?>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Error Establishing Database Connection</title>
  </head>
```

```
<body>
  
  <h1>Error Establishing Database Connection</h1>
  <p>We are currently experiencing database issues. Please check back shortly.</p>
</body>
</html>
```

Once uploaded to your server, this new `install.php` file will prevent any malicious behavior by serving up a static web page. Now, instead of showing the Installation Page when your database is unavailable, WordPress will display the information shown in the screenshot.

In addition to displaying this information to your visitors, the Installation Replacement Page also performs the following actions:

- **Communicates** a 503 (Service Temporarily Unavailable) status code to clients and search engines
- **Instructs** clients and search engines to return after 60 minutes (configurable in third line)
- **Sends** an email informing you of the situation so that you may take action (configurable in fourth line)

To use the Replacement Page, don't forget to specify an email address in the fourth line.

You may also change other variables, such as the time duration, email subject, or email message as needed.

For complete information on protecting WordPress' `wp-config.php` file, check out the original article at Perishable Press: <http://digwp.com/u/282>



Limit Login Attempts

Limit the number of times a user can try different passwords for your site with the Login Lockdown plugin <http://digwp.com/u/7>.

By setting some maximum number of login attempts, you greatly reduce the chance that an attacker will gain access by guessing your password. After the max number of attempts, the user is locked out for a specified period of time.

All settings easily controlled via Admin panel.

Remove the Login Error Message

Another tip for improving security is prevent WordPress from displaying the default error message on the Login Page. Invalid login attempts are met with a message that informs the user of the problem with their login credentials. So, if an attacker were to guess your username, the login error would provide confirmation and enable the attacker to focus on your password.

Fortunately, preventing the login error message is as simple as adding the following line to your `functions.php` file:

```
add_filter('login_errors',create_
function('$a', "return null;"));
```

Protecting the wp-admin directory

Once your `wp-config.php` file is secure, you should also protect your admin files, which are all conveniently located within the `wp-admin` directory. Protecting the files in this directory secures a very critical area of your site.

The easiest way to do so, is to add the following code to the `.htaccess` file located in your `wp-admin` directory (if the `.htaccess` file does not exist, create it):

```
# SECURE WP-ADMIN FILES
<FilesMatch "*.*)"
  Order Deny,Allow
  Deny from all
  Allow from 123.456.789
</FilesMatch>
```

As before, edit the IP address in the “Allow from” line to match your own. Alternately, you may use this code instead (also placed into the `wp-admin/.htaccess` file):

```
# SECURE WP-ADMIN FILES
<IfModule mod_rewrite.c>
  RewriteEngine On
  RewriteCond %{REMOTE_ADDR} !^123\.456\.789
  RewriteRule ^(.*)$ - [F,L]
</IfModule>
```

This second method requires Apache’s `mod_rewrite` module and works by denying access to any IP that is not your own. Thus, as before, you will need to edit the IP address accordingly. Once in place, this code will deny access – to everyone except for you – to all files in your `wp-admin` directory by returning a **403 Forbidden** error

message. Note that you can choose to redirect denied requests to the URL of your choice by replacing the last line with this:

```
RewriteRule ^(.*)$ http://domain.tld/ [R,L]
```

Simply edit the URL to whatever you wish.

Note that blocking your site's wp-admin directory with HTAccess may produce unintended results. If there are scripts or plugins that need to access your Admin area or its files, they will be prevented from doing so if your admin is locked down. One specific example that comes to mind is the "unsubscribe" feature of the **Subscribe to Comments** plugin. When a subscriber wants to manage their subscriptions, they need access to Admin files to do so.

Protecting the Admin Login Page

The WordPress Login Page is a huge target for attacks. As the virtual "door" to your Admin area, it's important to secure the page. Here are some effective strategies for doing so:

- **Choose a strong password**

Choosing a super-strong password is your site's first line of defense. Avoid simple words and phrases, and use a good mix of numbers and upper- and lowercase letters. Additionally, with WordPress, you may use other characters such as underscores and dashes to further confound your password. If you need help, do a quick Google search for "online password generators." There are many available.

- **Change your password often**

Once you have chosen a strong password, use it for awhile and then change it to something new. Even if you simply change a few characters to keep it easy to remember, it is best to do so to ensure optimal password security.

The Easy Way to Password Protect the wp-admin Directory

If you would rather not bother with the HTAccess method, you may want to try the **AskApache Password Protect** plugin <http://digwp.com/u/5>. This plugin will encrypt your password, create the .htpasswd file, and set the proper file permissions.

Another good plugin for locking down your wp-admin directory is **WP-Adminprotection** <http://digwp.com/u/6>, which allows only specified IPs access. Check it out.

Alternately, if you would rather not bog down your site with another plugin, you may want to check with your web host. You may be able to create password-protected directories with a few clicks of the mouse from your server control panel. For example, most cPanel implementations make this easy to do.



Password Protect Directories

Password protection allows you to restrict access to a directory by password protecting a folder, there are several ways to do this. Also, you will need to create users with a unique name, no matter what the actual directory name is.

Emergency Password Change

If you ever find yourself in a situation where you can't send or receive email but need to change the password of your admin-level account, you can do so easily via the database and phpMyAdmin:

1) Browse the "wp_users" table and click to edit the account name for which you would like to change the password.

2) In the "user_pass" field, you will see the encrypted version of your current password, which will look something like this: "%tH1sw0u1db3Y0ur3ncRYP73dpa\$w0rd". Replace this existing value with the plain-text version of your new password (e.g., "n3w_pa\$\$w0rd").

3) Finally, select the "MD5" option from the dropdown box (just to the left of the text field) and save your changes by clicking "Go!" This will tell phpMyAdmin to encrypt your plain-text password with the requisite MD5 hash.

Voila! It's that easy. Keep in mind that MD5 is designed for one-way encryption – you can use this method to encrypt new passwords, but you will not be able to decrypt existing passwords. It's strictly a one-way street.

We recently posted an article on this topic at the Digging Into WordPress website. In it, you will find everything you need to know about changing your password with WordPress in virtually *any* situation:

<http://digwp.com/u/283>

- **Block access to the Login Page**

Although blocking access is an excellent way to secure your Login Page, doing so will also prevent normal visitors from registering with your site. The reason for this involves the way the Login Page and the Register Page are served from the same PHP file. The URL for the Register page looks something like this:

`http://domain.tld/wp-login.php?action=register`

...which is calling the wp-login.php file and targeting the registration portion of it via the "action=register" query string. Thus, we can secure our site by blocking access to the Login Page, but in doing so, the Register Page will also be unavailable. Even so, if you are not allowing people to register with your site, locking down the Login Page is a **strong** security measure. To do so, add the following directives to the .htaccess file in your wp-admin directory:

```
# PROTECT WP-LOGIN.PHP
<Files wp\~login\.php>
    Order Deny,Allow
    Deny from all
    Allow from 123.456.789.0
</Files>
```

As in previous examples, simply edit the IP address to match your own. You may also allow additional IPs to access your Login Page by emulating the "Allow from" directive as many times as is necessary. Once in place, this method will allow only the listed IP address(es) to access your Login Page; everyone else will receive a **403 Forbidden** error.

If you would rather send all requests for your Login

(or Register) Page to the URL of your choice, use this method instead of the previous one (place code into the .htaccess file of WP's root directory):

```
# REDIRECT LOGIN REQUESTS
<IfModule mod_rewrite.c>
RewriteCond %{REQUEST_URI} wp\-login\.php
RewriteCond %{REMOTE_ADDR} !^123\.456\.789\.0
RewriteRule ^(.*)$ http://domain.tld/ [R,L]
</IfModule>
```

This code will redirect any requests that are not coming from the specified IP to the URL specified in the RewriteRule. Thus, for this code to work properly, you will want to edit the IP address and redirect target to suit your needs.

• **Secondary password protection**

Another option for securing your Login Page involves implementing secondary password protection via basic HTTP (Hypertext Transfer Protocol) authentication.

Adding secondary password protection is an excellent way to lock things down without the need for listing individual IP addresses. This is especially useful if you work through multiple or changing IP addresses, or if there are multiple administrators, authors, or contributors to the site.

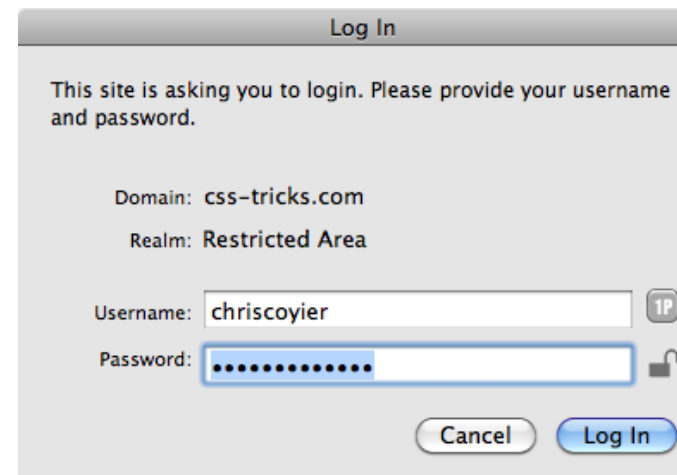
Conceptually, the process of password-protecting your Login Page is simple:

- Create a text file with your desired username and password
- Require the username and password via a local .htaccess file

In practice, implementing these two steps requires a bit of careful editing, copying and pasting. First create the username and password and place it in a text file like so:

```
anakin:y5gj8dwr39jg2
```

Place your username on the left side of the colon (:) and your encrypted password on the right. For the password, try one of



Secure Your Login with SSL

If you are fortunate enough to have access to SSL (Secure Sockets Layer) on your server (either through Shared SSL or with your own SSL certificate), you can greatly increase the security of your admin session by encrypting your connection via SSL. All you need is the following line placed in your wp-config.php file:

```
// secure admin over SSL  
define('FORCE_SSL_ADMIN', true);
```

If you would rather use a plugin for SSL, check out the **Admin SSL** plugin at <http://digwp.com/u/287>, which enables SSL on all pages for any WordPress version 2.7 and greater.

Alternately, if you do not have SSL enabled, you can still increase security by encrypting your passwords with the **Semisecure Login Reimagined** at <http://digwp.com/u/288>.

The plugin uses an RSA public key to encrypt your password, which is then decrypted on the server with a private key. Requires JavaScript, but certainly a worthwhile alternative to SSL.

these free online password encryptors:

- **4WebHelp's encryption tool** - <http://digwp.com/u/284>
- **Alterlinks password generator** - <http://digwp.com/u/285>
- **htmlite's encryption page** - <http://digwp.com/u/286>

Finally, save the username/password string in a file called ".htpasswd" (without the quotes!) and place it in a secure location on your server, preferably above your "public_html" or root-web directory.

Once the password file is in place, note its location for use in the next step. This file should be named ".htaccess" and should be placed into your wp-admin directory. Within this file, add the following code and edit according to the steps that follow:

```
# SECURE LOGIN PAGE  
<IfModule mod_auth.c>  
  AuthUserFile /full/path/.htpasswd  
  AuthType Basic  
  AuthName "Password Required!"  
  <Files wp\~login\.php>  
    Require valid-user  
  </Files>  
</IfModule>
```

Edit the full server path to match the location of your password file and you're good to go. With this method in place, access to your Login Page will require the user to enter the username and password credentials specified in your .htpasswd file.

To extend this password-protection to your entire wp-admin directory, modify the code as follows:

```
# SECURE ADMIN DIRECTORY
<IfModule mod_auth.c>
    AuthUserFile /full/path/.htpasswd
    AuthType Basic
    AuthName "Password Required!"
    Require valid-user
</IfModule>
```

This level of protection serves as a **strong** deterrent against attacks. Each login session expires when the browser is closed. If you experience any unexpected issues with your password protection, simply remove the `.htaccess` code to troubleshoot and restore original functionality. For more about Apache's password-protection module, check out the official site: <http://digwp.com/u/289>.

Other files that should be protected

In addition to those already mentioned, there are many other types of files that you should protect as well. These files aren't always associated directly with WordPress, but there are many cases where these files are available and potentially vulnerable. Here are a few types of files that you may want to protect explicitly:

- `.htaccess`
- `.htpasswd`
- `php.ini`
- PHP scripts
- Flash source files (`.fla` format)
- Photoshop files (`.psd` format)
- Log files

As well as any other files involving configuration, custom scripts, or source content. Protecting these is easily achieved by blocking access to files according to file type.

Red Carpet Treatment

*Quick tip: you can allow direct access to your **Login Page** by adding the following lines to the password method:*

```
Allow from 123.456.789.0
Satisfy Any
```

We can do this by targeting the different file extensions of the files. Simply place the following code into the root HTAccess file of your site:

```
# PROTECT SENSITIVE FILES BY FILE TYPE
<FilesMatch "\.(htaccess|htpasswd|ini|php|fla|psd|log)$">
  Order Deny,Allow
  Deny from all
  Allow from 123.456.789.0
</FilesMatch>
```

As in previous examples, change the IP address to match your own. It also easy to protect specific files. For example, if we want to deny all external requests for two files named "errors.php" and "test.html", we could do this:

```
# PROTECT SPECIFIC FILES
<Files ~ "^.*([Ee][Rr][Rr][Oo][Rr][Ss]|[Tt][Ee][Ss][Tt])">
  Order Deny,Allow
  Deny from all
  Allow from 123.456.789.0
</Files>
```

Don't be admin!

For millions of WordPress blogs around the world, the default user account is named "admin." WordPress does this by default, and the bad guys know it. They write evil scripts that seek out vulnerabilities and potential exploits by targeting default WordPress settings.

Changing your Admin username from "admin" to anything else isn't a one-shot magic bullet for protecting your site, but it will help prevent attacks aimed at the default "admin" account. It's an easy way to help improve the overall security of your WordPress-powered site. See page 38, "Neutering the Admin Account", for more information.

Here we account for both upper- and lower-case file requests via the regular expression in the first directive.

Keep in mind when using .htaccess that the directives are inherited and applied throughout the directory structure, such that all sub-directories and their files will be evaluated for processing.

9.1.6 Remove the WordPress Version Number

Another good way to boost security is to hide the WordPress version number from appearing in your source code. By default, many WordPress themes include the version information in the document <head> with a note that says something like, “leave this for stats please.” Statistics are certainly useful, but nowhere near as important as security. Here is the code that many themes use to display this info:

```
<meta name="generator" content="WordPress <?php bloginfo('version'); ?>" />
```

To prevent this information from being displayed in older themes, you can simply remove the previous line from your theme’s header.php file. In newer themes, however, the entire meta tag is generated by the wp_generator function from deep within the WordPress core. Fortunately it is just as easy to remove by adding the following line to your active theme’s functions.php file:

```
<?php remove_action('wp_head', 'wp_generator'); ?>
```

More Information

For a more in-depth look into the process of removing the WordPress version number, check out Digging Into WordPress:

<http://digwp.com/u/381>

9.1.7 Securing Your Database

As you should know, the default WordPress database prefix is “wp_”. Millions of WordPress sites use this prefix, and attackers know this. Many SQL-injection attacks and other automated exploits *assume* this default prefix during execution. Thus, by changing this prefix, we can obscure our database and block a good percentage of such automated attacks. There are plugins that will do this for you:

- **WP Prefix Table Changer** - <http://digwp.com/u/290>
- **WordPress Table Prefix Rename** - <http://digwp.com/u/421>
- **WP Security Scan** - <http://digwp.com/u/292>

Or, if you are comfortable doing so and prefer to have control over the process, there are two ways of changing your own database prefixes, depending on whether or not you already have installed WordPress. Let’s take a look.

Change your database prefix before installing WordPress

All that's needed is to change the `$table_prefix` value in the `wp-config.php` file.

Remember...

Don't forget to backup your database before making any changes!

You may change the prefix to whatever you like using alphanumeric characters and underscores, but your server configuration may limit the allowed length of your table names. A good technique is to use something like a mini-password for your prefix. Anything other than the default "wp_" will work.

```
51 /**
52  * WordPress Database Table prefix.
53  *
54  * You can have multiple installations
55  * prefix. Only numbers, letters, and
56  */
57 $table_prefix = 'wp_';
58
```

Change your database prefix on existing WordPress installations

A little more involved, but totally doable:

1. Backup your database.
2. Edit your `wp-config.php` file with the new prefix value.
3. Using a database interface such as phpMyAdmin, log in to your database and enter the following command for each of your database tables (change "xxx" to your new prefix):

```
rename table wp_comments to xxx_comments;
rename table wp_links to xxx_links;
rename table wp_options to xxx_options;
rename table wp_postmeta to xxx_postmeta;
rename table wp_posts to xxx_posts;
rename table wp_terms to xxx_terms;
rename table wp_term_relationships to xxx_term_relationships;
rename table wp_term_taxonomy to xxx_term_taxonomy;
rename table wp_usermeta to xxx_usermeta;
rename table wp_users to xxx_users;
```

Pop-Quiz, Hotshot

How many MySQL tables does WordPress create during installation?

(See next sidebar note for the correct answer.)

Note: Also check for any other tables and rename them as well.

4. Open your “options” table and click on the “Browse” tab. In the “option_name” column, locate the “wp_user_roles” field and change it to “xxx_user_roles”.
5. Open your “usermeta” table and click on “Browse” to browse the contents. In the “meta_key” column, locate all fields prefixed with “wp_” and replace each of them with your new prefix.

Once you have completed these steps, check your site thoroughly to ensure that everything is working properly.

Alternate method of changing your database prefix

An alternate way of changing your prefix is to download your database as an SQL file and search/replace all instances of “wp_” with “xxx_”. After that, drop your old tables and import your edited SQL file. Lastly, edit your wp-config.php file as described above and re-activate your plugins. That’s all there is to it. See? That’s one of the joys of WordPress – there are so many ways to get things done!

9.1.8 Secure Multiple Installations

Note that if you have multiple installations of WordPress on the same server, it is best to use different user credentials for each database. The username and password should be unique in the wp-config.php files for each of your WordPress installations. This security measure will ensure that an attacker who gains access to one of your sites will not automatically enjoy access to all of them.

9.1.9 Prevent Hotlinking

Hotlinking occurs whenever another domain displays your content on their site. For example, if you run a blog about Hollywood celebrities, you will inevitably discover that your images are being linked to directly and displayed on other celebrity sites.

Use Secret Keys to Increase Security

If you are using WordPress version 2.6 or better, you can increase your security by including a set of eight security keys in your wp-config.php file. Each of the eight secret keys is a hashing salt that is used in conjunction with your password to increase its effectiveness.

All you need to do is visit the free online generator at <http://digwp.com/u/524> and paste the results (see example below) into your wp-config.php file.

The eight keys each consist of a long string of random characters. The keys may be changed at any time, with the effect of invalidating all current cookies – i.e., anyone logged in before the change will need to re-log-in.

```
define('AUTH_KEY', 'lft/Y-;tXwG8zRQ`AAuill}4A=_,Y69KkIwhsIBlmg,9{] Tq-40TYl9fB7IzUG+-');
define('SECURE_AUTH_KEY', 'F+cH,mf-eNLZM_,l*00]l Yq`-to`%zL0A<z?~z)?a;9d|^ty2*Vc<Wjh<R<jSI[');
define('LOGGED_IN_KEY', 'UJ`Z=3y:j5`Nj0c_!@[QoJ%k=gQby>mR2keQLYY6EH:E%wKm+oJ8&IeZiQdQZTnk');
define('NONCE_KEY', 'g_/$PNb3Q-y<bq2H@LD@;&@{v~lg/~A#le(l-%}fw1ES<{vdUYPy.Gw0::Z*0/Hv');
define('AUTH_SALT', 'c: SFb!XKZs{d(i(-vsTI2x0i3ax~+-d!bTNURbWQ;HU%c;l99$@`/{X08_!!l-');
define('SECURE_AUTH_SALT', 'Z0McT.]E.mMnR )MQ;7P!hsnfBPri08*bBy:Bp|RZPc$&?}.xM^?Pg,bw%k|N ,.');
define('LOGGED_IN_SALT', ';;G`*+L<<l* ->S*ZAd&l .B]vDK0Thl`aXA@bjy6Y oC`KyMS0.i/CUg4:pZ@{l');
define('NONCE_SALT', '0_s>XraDn1[3o-`i$K:h_l^Fu?;uBkSJIIm(cuC4N@44kb3^ON-+nI*IXC!;W{0#<');
```

This type of behavior is typically seen as *worse* than stealing content because your bandwidth is also being hijacked. Preventing this type of behavior is important.

If your site features any content – images, videos, music, etc. – that you want to protect from hotlinkers, there are numerous methods available. The easiest and most effective anti-hotlink method, in our experience, is achieved via HTAccess:

Note

This will prevent users reading via online RSS reader from seeing images as well. To fix this inconvenience, check out this tasty post:

<http://digwp.com/u/418>

```
# HOTLINK PROTECTION
<IfModule mod_rewrite.c>
    RewriteEngine on
    RewriteCond %{HTTP_REFERER} !^$
    RewriteCond %{REQUEST_FILENAME} -f
    RewriteCond %{REQUEST_FILENAME} \.(gif|jpe?g|png)$ [NC]
    RewriteCond %{HTTP_REFERER} !^https?://([^.]+\.)?domain\. [NC]
    RewriteRule \.(gif|jpe?g|png)$ - [F,NC,L]
</IfModule>
```

To use the previous code, only one edit is required: change the term “domain” to match your actual domain. For example, if your domain name is `http://www.website.com/`, you would replace “domain” with “website”. Note that this code is set to protect the following file types: .jpg, .jpeg, .jpe, .gif, and .png. To protect additional files, such as those with the .ico format, simply add “lco” (a vertical bar followed by the “ico” extension) after the “lpng” in both the 6th and 8th lines.

For much more on protecting your site against “teh evil hotlinking scum,” check out the Perishable Press article listed at <http://digwp.com/u/294>.

9.1.10 More WordPress Security Help

With *any* site, there is always a way to improve its existing security strategy. For more information and tools for securing your WordPress site, check out the following plugins:

- **Secure WordPress** - <http://digwp.com/u/489>
Provides many important security measures, including protection against bad queries and complete removal of sensitive, auto-generated information.
- **WP Security Scan** - <http://digwp.com/u/9>
Scans your WordPress installation for known security vulnerabilities and suggests corrective actions. Features include passwords, permissions, and more.
- **WordPress File Monitor Plus** - <http://digwp.com/u/602>
Scans your WordPress files for malicious code and notifies you with the results. When files are changed, moved, added or removed, this plugin lets you know.
- **Ultimate Security Check** - <http://digwp.com/u/488>
Scans for “hundreds of known threats” and grades security performance. Provides a great overview of your site’s security. And it’s incredibly easy to use.
- **BulletProof Security** - <http://digwp.com/u/603>
.htaccess-based security that protects against many types of attacks, including XSS, RFI, CRLF, CSRF, Base64, Code Injection and SQL Injection hacking.

Pop-Quiz, Hotshot

(From previous sidebar note.)

Answer: 11

wp_commentmeta
wp_comments
wp_links
wp_options
wp_postmeta
wp_posts
wp_terms
wp_term_relationships
wp_term_taxonomy
wp_usermeta
wp_users

<http://digwp.com/u/503>

Lockdown Collection

*Underlined titles indicate
plugins used in the DigWP
Security Lockdown:*

<http://digwp.com/u/501>

Pharma Hacked!

Recently, WordPress sites have been hit with the so-called “Pharma Hack” – a hidden spam-injection hack that is near-impossible to detect and equally difficult to eradicate.

Elimination and protection against the nasty Pharma Hack and many others is possible using a combination of “lockdown” methods which we describe at DigWP.com:

<http://digwp.com/u/486>

This section’s list of security plugins has been updated to include those recommended in our Lockdown tutorial.

- **WordPress AntiVirus** - <http://digwp.com/u/296>
Smart and effective solution to protect your blog against exploits and spam injections. Features manual testing, auto notifications, and whitelist functionality.
- **BlogSecurity’s WPIDS plugin** - <http://digwp.com/u/8>
Detects attacks and blocks them. Each intrusion is clearly visible and an error is displayed, making administration easier than in previous versions.
- **AskApache Password Protect** - <http://digwp.com/u/5>
Protects your site by blocking automated attacks, spam, and other nonsense. Helps to secure wp-admin, wp-includes, wp-content, and plugins as well.
- **WordPress Firewall** - <http://digwp.com/u/10>
Blocks potential attacks based on a list of potentially suspicious parameters.
- **Login Lockdown** - <http://digwp.com/u/7>
Blocks the IP address of any user with too many failed login attempts.
- **Stealth Login** - <http://digwp.com/u/12>
Enables creation of custom URLs for logging in and other administrative tasks.
- **Exploit Scanner** - <http://digwp.com/u/490>
Searches your site’s files, plugins, and database for suspicious business.
- **Block Bad Queries (BBQ)** - <http://digwp.com/u/492>
Blocks excessively long request strings and other bad strings in the request URI.

9.2.1 Stopping Comment Spam

While we’re discussing security methods, it is important to take a look at different ways to stop comment spam. Comment spam plagues just about every comment-enabled or forum site on the Web, and WordPress-powered sites are no exception.

Fortunately, there are many top-notch developers contributing plugins, scripts and strategies to help fight the war against spam. Here are some of the best:

- **Akismet** - <http://digwp.com/u/298>
King of anti-spam plugins. Bundled with WordPress. Must-have.
- **Typepad Antispam** - <http://digwp.com/u/300>
Developed by Six Apart. Reported to work as well as Akismet.
- **Bad Behavior** - <http://digwp.com/u/301>
Anti-spam protection plus additional security features.
- **Simple Spam Filter Plugin** - <http://digwp.com/u/303>
Captcha-based. Designed to work with existing anti-spam plugins.
- **WP-SpamFree** - <http://digwp.com/u/304>
Virtually eliminates automated comment spam. No captchas. No false positives.
- **NoSpamNX** - <http://digwp.com/u/305>
Adds extra hidden fields to your comment form to catch bad bots.

In addition to these incredible plugins, there are a few other helpful tricks that you may want to try. Let's take a look at some choice techniques in the next few sections of this chapter.

9.2.2 Configuring Your WordPress Admin Options

Configuring your Admin options with the most restrictive comment settings is a much underrated method of reducing and preventing a great deal of comment spam. In the **Admin > Settings > Discussion** options page, there are several options that enable you to take strong action against spam. The most restrictive option would be of course to simply require moderation of all comments. This would theoretically prevent *all* spam, since you would be filtering them out manually.

Before a comment appears

- ☒ An administrator must always approve the comment
- ☐ Comment author must have a previously approved comment

SSL Security Plugins

Here are two excellent plugins that secure your site via SSL:

Force SSL:

<http://digwp.com/u/17>

Admin SSL:

<http://digwp.com/u/14>

Anti-Spam Cornucopia

For even more excellent anti-spam plugins for WordPress, check out chapter 7.6.3.

Spam-free, no plugins

WordPress' built-in comment moderation/blacklist system is powerful enough to keep any site spam-free without any additional plugins (even Akismet). Learn more:

<http://digwp.com/u/525>

This really isn't an option for sites that feature lots comments, so the next most restrictive setting would be to only allow comments from people who have been previously approved. By requiring the commentator to have a previously approved comment, you drastically reduce the chances that a spam will appear on your site.

9.2.3 Using the Built-In Comment Moderation

Also, in the Admin **Discussion Settings** you will find three powerful anti-spam options. The first is a link-filtering option that automatically holds comments in the moderation queue if they contain "x" number of links. Since links are frequently the payload of spam comments, moderating any comments containing, say, two or more links is a great strategy.

There is also a large input field that may be used to list any characters, phrases, or even IP addresses that you would like to pre-approve if found in the comment. For example, if you want to moderate any comments containing the phrase "Viagra," or that come from an IP address of 123.456.789.0, then you would list these items as shown in the screenshot.

When a comment contains any of these words, it will be held in the [moderation queue](#). One word or IP per line.

```
soma
ambien
cialis
buycialis
hydrocodone
viagraonline
cialisonline
phentermine
viagrabuy
percocet
```

Careful with that Axe

When configuring your Comment Blacklist, choose phrases that will not appear as parts of "legitimate" words. We're dealing with regular expressions here, so make sure that you aren't unintentionally trashing any legitimate comments. Fortunately, many drug names are very unique.

9.2.4 Using the Built-In Comment Blacklist

And, even better than WordPress' Comment Moderation is the built-in "Comment Blacklist." Also located on the **Discussion Settings** page, the Comment Blacklist works exactly like the moderation list, only instead of being held for moderation, any comments containing blacklisted phrases will be immediately marked as spam and discarded.

Be mindful when using this technique – all terms and phrases are treated as regular expressions, such that you may be inadvertently dumping legitimate comments.

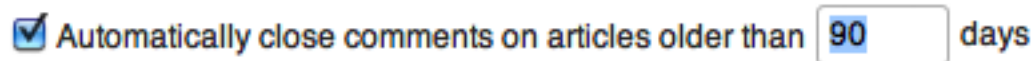
When a comment contains any of these words, it will be held in the [moderation queue](#). One word or IP per line. It will be marked as spam.

```
citalopram
cephalaxin
vicoprofen
lorazepam
oxycontin
oxycodone
percocet
propecia
tramadol
```

9.2.5 Disabling Comments on Old Posts

Spammers frequently target old posts because they have been indexed in the search engines and have had more time to accumulate page rank. So, as the number of posts in your archives increases, you will inevitably find yourself dealing with lots of spam and other nonsense on older posts.

An easy solution to this is to simply disable comments on all posts that are older than “x” number of days. For example, DigWP.com automatically closes comments after 90 days.



Any reasonable amount of time should work fine. For more information on manually disabling comments on old posts, refer to chapter 7.3.7.

9.2.6 Deny Access to No-Referrer Requests

Many spambots target WordPress’ comment script directly, bypassing your comment form entirely. An easy way to circumvent this behavior is to deny all requests for the comment script that do not originate from your domain. This is another HTAccess trick that we may write like this:

```
# DENY ACCESS TO NO-REFERRER REQUESTS
<IfModule mod_rewrite.c>
RewriteCond %{REQUEST_METHOD} POST
RewriteCond %{REQUEST_URI} .wp-comments-post\. [NC]
RewriteCond %{HTTP_REFERER} !.*digwp\. [OR,NC]
RewriteCond %{HTTP_USER_AGENT} ^$
RewriteRule (.*) - [F,L]
</IfModule>
```

Edit this code so that the domain name (“digwp”) matches your own. As is, this

code will simply deny access to the requested comment script. To redirect the spammers instead, replace the RewriteRule with this:

```
RewriteRule ^(.*)$ http://%{REMOTE_ADDR}/ [R=301,L]
```

This will bounce the spammers back to where they came from. Nice, but you may prefer to send them someplace else. To do so, simply edit the URL (i.e., the “http://%{REMOTE_ADDR}/” portion) to whatever you wish.

By blocking all requests for the comments-processing script (`wp-comments-post.php`) that are not sent directly from your domain (via `comments.php`), you immediately eliminate a large portion of blog spam. For more information on this technique, check out the Perishable Press article at <http://digwp.com/u/307>.

9.3.1 Monitoring and Fixing Errors

As you set up and run your site, it is a good idea to keep an eye on any errors that pop up. There are several ways to do this, depending on your familiarity with your server logs and how they work. Many hosts provide access to automatically generated **server access logs**. These are useful for diagnosing patterns relating to spam, broken URLs, and malicious attacks.

Additionally, many servers make available error logs or will automatically generate PHP error files that appear in the root directory of the site. Keeping an eye on these access and error logs is good practice as it will often enlighten you about broken scripts, plugins, links, and much more.

Especially important is keeping a close eye on **404 Not Found** errors. If your site has too many broken links and missing pages, your site’s pages may suffer in the search-engine listings. The bad news is that large sites with thousands of pages are difficult to check by hand in a thorough manner. The good news is that there are several great methods for accomplishing this in an easy, automated way. Let’s examine a few of the best.

9.3.2 Alex King's 404 Notifier Plugin

Alex King's **404 Notifier** is an excellent plugin by one of the top WordPress developers. Logs all **404 Not Found** errors with the option of automatically notifying the site owner of each 404 incident via email or RSS. Requires permalinks to be enabled. Check it out at <http://digwp.com/u/309>.

9.3.3 Broken Link Checker Plugin

Keeping an eye on your site's links can be a seriously daunting task, especially as your site continues to grow in size and complexity. Over time, your outgoing links may break or end up pointing to something unintended. Good, solid links are the cornerstone of the Web; broken links fail to help your visitors and may cause the search engines to consider your page or site less favorably, especially if many broken links are present.

To prevent this scenario, there is an awesome plugin called **Broken Link Checker** <http://digwp.com/u/310> that monitors your site and helps you manage broken links. Once installed, Broken Link Checker works quietly in the background, testing your links and reporting any that are broken or redirected. The plugin monitors all parts of your site, including custom fields (optional). Also detects missing images. Link-checking intervals are completely configurable. Provides options for broken links, including unlinking, editing, and deleting. Truly an awesome plugin.

The one shortcoming of using an automated method for checking your links, however, is the case where a linked page has been changed or redirected to include undesirable content. Because the link resolves to a working page, it will be assumed as valid and thus will not be included in the broken-link report. Beyond this scenario, automating the process of checking broken links can be a *tremendous* help.

9.3.4 Other Error-Logging Techniques

Logging errors and activities for your site is critical for better control over your website. Here are some plugins that can help get the job done.

- **WordPress to Syslog (WPsyslog2)** - <http://digwp.com/u/311>

WPsyslog2 is a global logging plugin that tracks all system events and logs them to syslog for your analytical use. Tracks new posts, new profiles, new users, failed logins, successful logins, logouts, and much more.

- **Post Logger plugin** - <http://digwp.com/u/313>

Reveals the intimate details of the \$POST variable for each request, enabling you to keep a better eye on what's happening behind the scenes with your comments.

- **TTC WordPress Tripwire Tool** - <http://digwp.com/u/314>

Provides you with a list of all files changed on your WordPress site within the specified period of time.

- **InspectorWordpress** - <http://digwp.com/u/16>

Monitors and logs requests to your WordPress-powered site.

9.3.5 Online Monitoring Services

An important part of developing and running a successful, well-optimized site is making sure it is always available to your visitors. In a perfect world, your site's uptime would be 100%. But thanks to server issues, software conflicts, malicious scripts, and cracker exploits, it is virtually inevitable that your site will go down from time to time.

While you can't *prevent* periods of unexpected downtime, you can improve your ability to respond in a timely manner by using an online monitoring service. Monitoring services basically keep any eye on your site and notify you when they become unavailable. There are many services available, both free and paid, each with their own way of tracking your site and reporting statistics. Here are some of our favorites:

- **Are My Sites Up?** - <http://digwp.com/u/316>

Fast, easy, and reliable site monitoring service that provides free monitoring of up to five sites 25 times per day. Premium service also available with tons more features. iPhone application available. Highly recommended! :)

- **Pingdom** - <http://digwp.com/u/317>

Provides email and SMS alerts when your site is unavailable. Monitors uptime and overall performance.

- **Mon.itor.us** - <http://digwp.com/u/318>

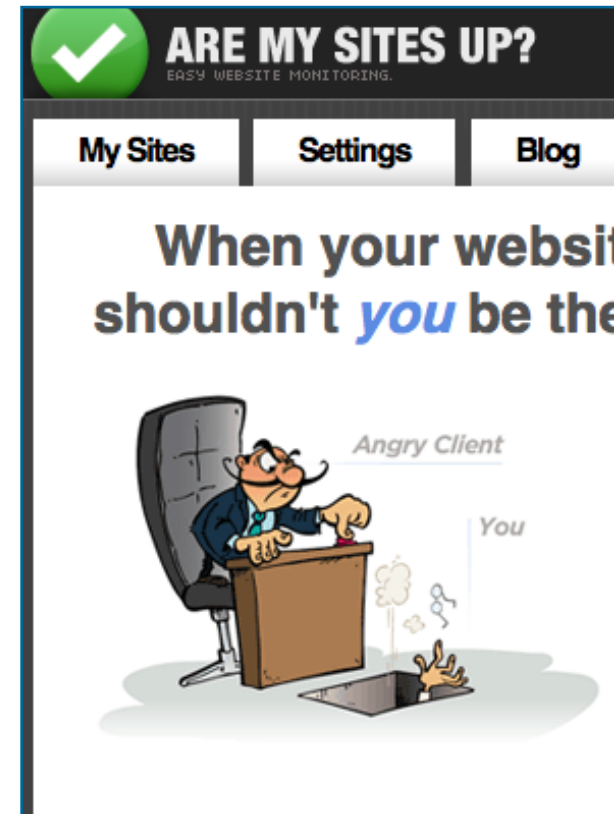
Free website monitoring services with email alerting. Provides uptime and response-time reports. Alert formats include email, IM, SMS, and RSS.

- **Montastic** - <http://digwp.com/u/319>

Free monitoring service with email and RSS alerts. Monitors up to 100 URLs every 10 minutes. Notifies you when your site's availability has been restored.

- **BasicState** - <http://digwp.com/u/322>

Free website uptime monitoring service that checks unlimited sites every 15 minutes. Provides instant trouble alerts by email or SMS. Recommended.



9.4.1 Staying Current with WordPress

Of course, one of the **best** ways to keep your site secure is to stay current with WordPress. While working in the WordPress Admin, keep an eye out for any alert messages informing you of available updates, either for the WordPress core or for individual plugins.

Staying current with the latest versions of WordPress ensures that your site receives all the latest security patches, bug fixes, and script improvements. Likewise,

& More Monitors

These fine services are just the tip of the iceberg! For a huge list of server monitoring services & website monitoring software, check out this valuable resource:

<http://digwp.com/u/327>

keeping your plugins up-to-date ensures compatibility with the latest versions of WordPress and helps keep everything running smooth in general.

While we're on the subject, serious WordPress developers and users may benefit from reading the WordPress Development blog, which is displayed by default in your WordPress dashboard. This is a great place to learn about all the latest WordPress development news.

And, if you happen to discover a bug while working with WordPress, you may report it at the designated page via the WordPress Codex <http://digwp.com/u/328>. If you think that you have discovered a security vulnerability, email the security team at "security@wordpress.org" with the information and wait for a response before sharing it anywhere else.

9.4.2 Updating WordPress

Those of us who have used WordPress for any length of time understand well how frequently new versions of WordPress are released. When everything goes according to plan, WordPress is updated four times every year. That's a lot of upgrading for everyone involved: users, designers, and developers. Fortunately, the busy WordPress devs have integrated an easy way to stay current: **automatic updates** from the comfort of the WordPress Admin!

[WordPress 3.3.1](#) is available! [Please update now.](#)

Before WordPress 2.7, upgrading to the latest version of WordPress required manually uploading files to your server. In 2.7 or better, you simply need to navigate to the "**Tools > Upgrade**" page of the Admin area and click on the "Upgrade" button. If your server is properly configured and everything goes as planned, the WordPress core will be updated with a single click!

Likewise for plugins – single-click installs and updates for any plugin listed in the WordPress Codex. To install a plugin, go to "**Plugins > Add New**" in the Admin area and knock yourself out. There you will be able to search the WordPress

Plugin repository and install any plugin by clicking on the “Install” button in the right-hand column. Similarly, to update any of your installed plugins, click on the “update” link when it becomes available. Automated convenience at its best.

As useful as this is, however, there are situations where **manual upgrades** are a better way to do it. Here are some scenarios when manual updates make sense:

- If you are running an older (pre-2.7) version of WordPress
- If you have a highly customized site with lots of core hacks, mods, etc.
- Your server settings forbid this type of behavior

In these and other situations where automatic upgrading is neither possible nor advisable, you should take the time to upgrade manually. It isn’t always fun, but staying current is one of the best ways to ensure that your site is kept secure.

9.4.3 Logging Changes

How do you know if your site has been hacked? If some unscrupulous attacker breaks into your site and injects a few thousand invisible spam links, how would you know? Waiting until your site is penalized by Google is a bad strategy. Here are some plugins that will keep an eye on your site and notify you if anything changes:

Know thy files

If you are automatically updating your plugins and/or core files through the Admin (or any other method), it is wise to remember that the files on the server will be newer than the ones on your local machine. This may sound totally obvious, but much confusion and many errors may be avoided by not overwriting updated files with older ones. A good way to prevent this is to either use some sort of a version control system (such as Subversion), or else play it safe and go with the manual-update method.

Get Automatic Upgrade Emails

The alert messages provided in the WordPress Admin are great, but they don’t work if you never log in to your website’s admin area.

Fortunately, the **Update Notifier** plugin <http://digwp.com/u/329> takes care of this by sending you daily email notices whenever new versions are available.

This makes it easy to keep an eye on large numbers of sites without having to log in or subscribe to any RSS feeds. Simply install and forget about it. As soon as it’s time for action, you’ll get an email letting you know.

- **WordPress File Monitor** - <http://digwp.com/u/330>

Monitors your WordPress installation for added/deleted/changed files. When a change is detected an email alert can be sent to a specified address.

- **MonitorHackdFiles** - <http://digwp.com/u/331>

Watches your site, and when it detects a file has changed (or been added), it notifies you via email and tells you which file was changed.

- **ChangeDetection.com** - <http://digwp.com/u/332>

Free online service that monitors your site and sends you an email/SMS if anything changes. Simple, easy, and effective.

9.4.4 Backing Up Your Database and Files

As with all work that is done on a computer, it is essential to ensure that regular backups are made of your work. For dynamically powered websites such as those powered by WordPress, this practice involves backing up your database, core files, and added content.

Wait, there is Another...

Although WP DBManager would be our first choice, there is another database plugin called WordPress Database Backup that focuses entirely on one task: backing up your database. Check it out at:

<http://digwp.com/u/333>

The easiest way to keep regular backups of your database is to use the **WP-DBManager** plugin at <http://digwp.com/u/334>. This is a powerful backup plugin that provides a **ton** of features, including everything from scheduled database optimizations to completely customized database backups. Backup databases are then stored either on your server or delivered to you via email. This plugin does require a specific server configuration in order to work, so if things don't go well, you will need to either use an alternate plugin or backup your database manually.

Fortunately, using a MySQL interface such as phpMyAdmin makes the process of creating manual

Backup Database

Database Backed Up Successfully On 'October 26, 2014'

Checking Backup Status

Checking Backup Folder

Backup folder exists

Backup folder is writable

Checking MySQL Dump Path

MySQL dump path exists.

Checking MySQL Path

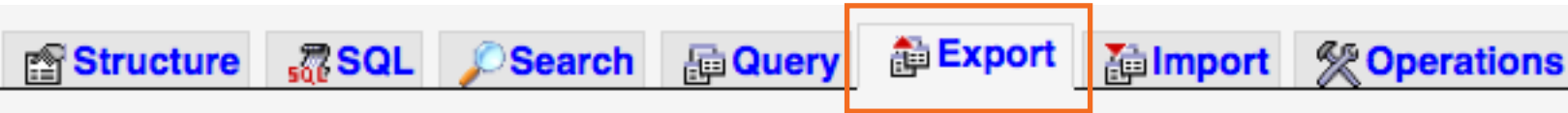
MySQL path exists.

Checking PHP Functions

passthru() enabled.

system() enabled.

backups very easy. Simply log in, choose your database from the left sidebar, and click the “Export” tab (see screenshot). Then from the “Export” page, choose your settings and save export the backup as a zipped file on your computer.



Another good practice is to backup your physical files. These include the entire WordPress core along with any additional files or content that you may have added. It is a good idea to back up these files periodically, as well as specifically before any upgrades, updates, or other modifications. To backup your content files automatically, check out the **Content WP Backup** plugin: <http://digwp.com/u/335>.

And Another...

We recently discovered another excellent plugin for backing up your database, files, and just about anything else. All automatic with plenty of settings for granular control.

<http://digwp.com/u/526>

9.5.1 Optimizing WordPress

There are many ways to optimize the already-great, out-of-the-box performance of WordPress. Let's take a look at a few of the most effective ways to improve the speed and consistency of your site.

9.5.2 Content and File Caching

Each time a visitor requests a page from your site, the server kicks into gear, processing scripts and querying the database to generate the page. For sites with small amounts of traffic, the load on your server is probably not a big deal and your pages should load just fine. For highly trafficked sites, however, the strain on the server to crank out thousands or millions of pages can really slow things down.

A great way to circumvent this problem is to install a caching plugin. A good caching plugin reduces server load by generating a static copy of each requested page and then delivering that for all subsequent requests. Serving static pages requires fewer resources from your server and can speed things up considerably. Here are a few of the most popular caching plugins:

Cache Inspector

See what's up with your cache!

WP Cache Inspect:

<http://digwp.com/u/342>

Database Cache

Cache database queries!

DB Cache:

<http://digwp.com/u/338>

- **WP Super Cache** - <http://digwp.com/u/337>
Creates static HTML versions of your pages, eliminating the need to invoke PHP and the database.
- **Hyper Cache** - <http://digwp.com/u/340>
Stores HTML page output as file content. Uses the PHP engine.
- **AskApache Crazy Cache** - <http://digwp.com/u/341>
Works in tandem with WP-Cache, WP Super Cache, or Hyper Cache to cache your entire blog.
- **W3 Total Cache** - <http://digwp.com/u/527>
Easily optimize the speed and user experience of your site with caching: browser, page, object, database, minify and content delivery network support. "The fastest and most complete WordPress performance optimization plugin."

It is important to read the documentation carefully before installing any of these caching plugins. In general, caching is a process that fundamentally affects the way your site performs, so it is important to understand the pros and cons of each plugin as well as the requirements for installation. You definitely should not try to combine caching plugins unless you really know what you're doing. And even then, we don't recommend it.

Choose the Right Host

Perhaps the best way to ensure that your site is running as fast, smooth, and consistent as possible is to find the best host. More than anything, with web hosting, you get what you pay for. If you are serious about running a solid site that is fast and reliable, stay away from cheap, sold-out web hosts and find something with excellent servers and strong service.

We can't stress this enough: a good host is *worth* the extra money. Of course, just because a host costs more doesn't necessarily mean that it's actually better. Our advice? Do your research, check the message boards, and email your favorite sites for tips and clues on finding the right host.

9.5.3 File Compression Methods

Another excellent way to improve performance while also saving bandwidth is to compress your web pages and other site content. Compression does exactly what you would expect: files and content are compressed by the server in order to reduce their overall size. Once the content is received by the browser, it is immediately uncompressed and displayed properly. This results in faster loading times and reduced bandwidth usage.

While a complete excursion into the realms of file compression is well beyond the scope of this book, here are a few ideas to get you started in the right direction:

- **File compression via Apache's gzip module** - for servers running older versions of Apache, an easy and effective way to compress your content is to enable `mod_gzip` via your server configuration or root `.htaccess` file.
- **File compression via Apache's deflate module** - for servers running newer versions of Apache, an easy and effective way to compress your content is to enable `mod_deflate` via your server configuration or root `.htaccess` file.
- **File compression via PHP** - It is also possible to compress your files using PHP's output buffer. This method usually involves adding a small snippet of code to the beginning of your theme's header `.php` file.
- **Manual file compression** - For JavaScript, CSS, and other static files, it's also possible to implement compression manually. This typically requires gzipping the files in question and then delivering them to supportive browsers.
- **Minifying CSS and JavaScript files** - Apart from compressing the actual file, it is also possible to compress the *contents* of your CSS and JavaScript files. This process is called "minifying" and usually involves removing as much white-space as possible. For JavaScript, there are additional ways to further reduce file size.

Spoiled rotten, WordPress users enjoy such awesomely useful plugins as **WP Minify** <http://digwp.com/u/344> and **PHP-Speedy** <http://digwp.com/u/345> that will minify, compress, combine, and cache your CSS and JavaScript files. There are also some great online services for compressing CSS and JavaScript file content, including these great sites:

Dynamic vs Static

One thing to note about using caching plugins is that you're trading dynamic functionality for static pages. This may interfere with any dynamically updated content on your site.

Easy PHP Compression

More information on how to compress your files with PHP:
<http://digwp.com/u/343>

- **YUI Compressor** - <http://digwp.com/u/347>
- **Dean Edwards Packer** - <http://digwp.com/u/346>
- **JavaScript Compressor** - <http://digwp.com/u/349>
- **Another JavaScript Compressor** - <http://digwp.com/u/348>
- **Styleneat.com** - <http://digwp.com/u/350>
- **JSMin** - <http://digwp.com/u/351>

Optimize with WP CSS

If you don't mind an additional plugin to optimize your CSS files, you'll want to check out WP-CSS.

WP-CSS uses a shorthand technique to strip extraneous whitespace from your CSS files. Then after reducing file size, WP-CSS compresses your CSS files with Apache's powerful gzip compression. It even includes any @import files into the mix.

<http://digwp.com/u/355>

While there are many different ways to take advantage of file compression, your implementation will depend on the tools and resources available to your server. If possible, enable `mod_gzip` or `mod_deflate` and forget about it. Otherwise, if these modules are not available to you, there are many other solutions available.

9.5.4 Optimizing CSS and JavaScript

If all that server/database compression/optimization stuff leaves you gasping for air, relax – a significant amount of optimization can be accomplished by focusing on the code and content used to create the user interface. By optimizing the content of JavaScript, CSS, and even HTML files, you can reduce file size, save bandwidth, and reduce loading times for visitors. Here are some key strategies that will help you to optimize various types of code:

- **Keep your code clean!** - Eliminate unnecessary comments and superfluous markup. Focus on clean, well-written code and you will have a strong foundation for optimizing your pages.
- **Keep HTTP requests to a minimum** - One of the best ways to improve the loading times for your pages is to reduce the number of HTTP requests to the server. Every JavaScript file, CSS file, and image requires its own HTTP request and thus slows down loading time. By consolidating multiple CSS and JavaScript files and implementing “sprite” techniques, you can reduce the number of HTTP requests and increase the performance of your site.
- **Use sprites for images** - Put simply, sprites are multiple images consolidated

into a single image. By strategically placing different images in a single image file, we decrease latency in visual display while also reducing the overall number of HTTP requests. Sprites are commonly used together with CSS to create stunning and effective rollover effects, background imagery, and more.

- **Include your stylesheets at the top of your pages** - When including external stylesheets in your pages, be sure to include them at the **top** of the page in the <head> section. This will enable browsers to render your pages *progressively*, which makes them appear to load much faster.
- **Include JavaScript at the bottom of your pages** - When including external JavaScript files, placing them at the **bottom** of the page, just before the closing <html> tag, ensures that browsers are able to download the maximum number of components, decreasing load times and improving performance.
- **Validate your code!** - One of the best ways to ensure that you are adhering to the principles of modern web design and web standards is to check your code with an online validator. After checking your page, the validator will return a report telling you either that your code has passed with flying colors, failed miserably, or anything in-between. If there are problems with your code, the validator will explain each issue and provide suggestions for fixing them.

Depending on code type, there are many different validators available, however, the W3C (World Wide Web Consortium) provides just about everything you need right under one roof. Here are some URLs for two of their free code-validation services:

- **W3C (X)HTML Validator** - <http://digwp.com/u/353>
- **W3C CSS Validator** - <http://digwp.com/u/354>

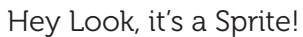
Spriting Made Easy

Creating sprites is an art form that many have yet to master. Thankfully, designers now have a free online service that automagically creates sprites for you:

<http://digwp.com/u/352>

9.5.5 Reducing the Number of HTTP Requests

One of the biggest factors of site performance is the number of HTTP requests that your pages are making to the server. Each request for a CSS file, JavaScript file, image, or any other external file requires a separate call to the server, which then must acknowledge, process, and return the requested file. When you have too



many files linked to a document, either in the `<head>` area or in the content itself, your site's performance may be negatively affected. This effect is easily seen by comparing the load times of sites that include many different CSS and JavaScript files with sites that have taken appropriate measures to reduce the overall number of requests made by their pages. Here are a few tips for reducing the number of HTTP requests made by your site:

- 334

next section of the web page. This is *opposite* of preloading all of your images at once. With *progressive* image-loading, users will experience a slight delay in the loading of images as they scroll down the page, but for pages with lots of images, progressive loading can reduce the number of simultaneous HTTP requests and greatly improve performance for the initial page load. Check out the jQuery plugin, **Lazy Load** <http://digwp.com/u/493>, to easily implement progressive image loading on your site.

- **Use a Content Delivery Network** - A Content Delivery Network (CDN) is a network of computers cooperating to deliver content for web sites. Using additional servers to deliver images, downloadables, and other static files reduces the load of your primary server while improving the performance and scalability of the overall website. Some commercially available CDNs include:
 - **Akamai Technologies** - <http://digwp.com/u/364>
 - **Amazon CloudFront** - <http://digwp.com/u/358>
 - **BitGravity** - <http://digwp.com/u/359>
 - **CacheFly** - <http://digwp.com/u/360>
 - **Internap** - <http://digwp.com/u/361>
 - **Limelight Networks** - <http://digwp.com/u/363>
 - **Mirror Image Internet** - <http://digwp.com/u/365>

There are several ways to track HTTP requests. By simply opening a text editor and scanning your CSS file for all instances of the "url()" value, you can determine how many images are being called just for your design.

In your browser, you can also check the source code of your pages and examine the number of files linked to in the <head> area. By searching your source code for all instances of "<img", you can see how many regular images are being called.

If you are using Firefox, the awesome and absolutely essential **Firebug** extension <http://digwp.com/u/366> can be used with Yahoo's helpful **YSlow** add-on <http://digwp.com/u/379> to analyze your site's performance in real time, with nice

How to Stop Leeching and Improve Site Performance

As explained in section 9.1.9 of this chapter, “hotlinking” is bandwidth theft that happens whenever another site is linking directly to your files. For example, if you have some spicy picture of Chewbacca in a swimsuit, you may quickly discover that unscrupulous bastards are linking directly to it, stealing your image, your bandwidth, and your traffic. To prevent this sort of leeching, add the following slice of HTAccess code to your site’s root .htaccess file (or Apache configuration file):

```
# HOTLINK PROTECTION
<IfModule mod_rewrite.c>
RewriteEngine on
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{REQUEST_FILENAME} -f
RewriteCond %{REQUEST_FILENAME} \.(gif|jpe?g|png)$ [NC]
RewriteCond %{HTTP_REFERER} !^https://([^.]+\.)?domain\. [NC]
RewriteRule \.(gif|jpe?g|png)$ - [F,NC,L]
</ifModule>
```

There are of course many ways to customize this code, including changing the domain name to match your own, adding additional approved domains to the list (so your images are visible in feed readers, for example), and so on. As-is, this code simply returns a **403 Forbidden** error for anything other than your site that is requesting images. This may be changed to return some nasty image, so that people who try to steal your zesty picture of Chewbacca will get some nasty shot of your armpit instead. Currently, this code blocks hotlinking for GIFs and JPGs, but you can add many other types of files to the list as well.

For a more comprehensive look into the fine art of protecting your site against hotlinking, check out <http://digwp.com/u/294>.



Image source: TheBestWho.com

graphical representations showing many different aspects of your site, including the overall number and duration of HTTP requests. Here is a screenshot showing how Firebug together with YSlow can help you analyze the number of requests for any page.

9.5.6 Plugin Maintenance

Many articles on the Web will advise you to disable unused plugins as a way of optimizing your site. While this is partially true, there is much more to the story. It all begins when deciding which plugins to install. Do some research and make an informed decision before implementing each of your plugins. Every plugin that you install requires resources and may decrease performance. Plugins also affect performance in a *cumulative* fashion. Even so, I have seen sites running 50 or more plugins that load very quickly. Just keep in mind that, unless you know what you are doing, loading up on unnecessary plugins can really slow things down.

Next, once you begin installing plugins, remember to test as each one is installed. Don't just slap it in there and walk away – take the time to surf your site and examine its performance with the newly installed plugin(s) installed. If you find that they are slowing things down, reconsider your choice and search for better options. Otherwise, do what you can to offset the reduced performance, perhaps by caching (see section 9.5.2 in this chapter).

Then, as you continue to take control of your plugin arsenal, keep an eye on any that may have been useful at one time but now are useless. Uninstall any obsolete and unused plugins immediately. Every time you load a page on your site, *all* of your plugins are loaded in the background. Thus, any plugins that you can remove from this equation will provide an immediate performance boost.

YSlow

Grade Components Statistics Tools

Rulesets YSlow(V)

Grade **B** Overall performance score 89 Ruleset applied: YSlow(V2) URL: http://p

ALL (22) FILTER BY: CONTENT (6) COOKIE (2) CSS (6) IMAGES (2) JAVASCRIPT (

A	Make fewer HTTP requests
F	Use a Content Delivery Network (CDN)
C	Add Expires headers
A	Compress components with gzip
A	Put CSS at top
A	Put JavaScript at bottom
A	Avoid CSS expressions
N/A	Make JavaScript and CSS external

Grade A on Make fewer HTTP requests

Decreasing the number of components on the page, resulting in faster page loads. For example, combine multiple scripts into one file, combine multiple stylesheets into one file, combine multiple images into one image map, and use CSS sprites.

[»Read More](#)

Too Many Plugins?

As this book goes to press, we are currently running a poll at the Digging Into WordPress companion site that asks, “How Many Plugins do You Use?” Drop by and vote if the poll is still open when you read this, otherwise check in to see the results:

<http://digwp.com/u/367>

Also, as mentioned above, keep your plugins updated. Whenever you see that a new version of the plugin is available, take the time to check it out and then upgrade as soon as possible. Staying current with plugins is a good idea because they usually feature improved code, tighter security, and better features.

Finally, once you decide to disable a plugin, remove it completely. Don’t leave it sitting there in a disabled state in your Admin area. Uninstall it, and make sure that any peripheral files that may have been related to the plugin are also removed. Hopefully, any plugin that you decide to install will provide a complete “Uninstall” option that cleans up after itself in the WordPress database, but if not, you may want to go in and manually remove all traces of it.

9.5.7 Database Maintenance

If your theme is the heart of WordPress, then your database is its brain. The database contains all of the settings, comments, posts, plugin options, and administrative data required for your site to function properly. When this data becomes corrupted or is poorly optimized, performance may suffer. Optimizing and backing up your database at regular intervals is smart practice. Highly active sites should optimize every day, while less-active sites maybe once a week or less.

You should also take every opportunity to clean up your database of unnecessary data. For example, after installing, testing, configuring, and uninstalling plugins, you should examine your database for any tables or options that may no longer be necessary. If the plugin developer did their job, their plugin will include a complete uninstall option that cleans up the database for you. Unfortunately, this is not always the case.

9.5.8 Other Optimization Techniques

There are many ways to optimize your site and improve performance for your visitors. Here are a few more ideas that are well-worth the effort.

Split long posts into multiple pages

WordPress enables you to split up long posts into individual pages using the `<!--nextpage-->` tag in your posts. This trick is not for all posts, but for those super-long articles with lots of images and stuff, it may really help to speed things up.

Limit the number of posts on archive and index pages

The idea here is to reduce the overall size of your web pages by displaying fewer posts. Instead of, say, 20 posts on your homepage, you can use a plugin such as Custom Query String Reloaded <http://digwp.com/u/373> to limit the number of posts that are displayed. By doing similar for other types of page views, you can customize page size in granular fashion.

Cache Your Database

Improve database performance by caching your database queries. Check out the DB Cache plugin to cache your queries and speed up the loading of your site.

<http://digwp.com/u/368>

Specify far-future expires or cache-control headers

For static resources such as images, CSS and JavaScript files, a great way to decrease

Optimize Your WordPress Database

Optimizing your database is a great way to improve site performance. Here are five ways to do it.

- **Manual SQL query** - using the “OPTIMIZE TABLE” command, it is possible to optimize any specific table or group of tables. Use this format, replacing the example terms with your table names:

```
OPTIMIZE TABLE `feeds`, `items`, `options`, `tags`
```

- **Using phpMyAdmin** - check all tables in your database and select “Optimize” from the dropdown menu.
- **Use a plugin** - Joost De Valk’s **Optimize DB** plugin makes it easy to optimize your database: <http://digwp.com/u/369>

- **Use a different plugin** - Another good plugin for optimizing and cleaning up your database is **WP-Optimize** <http://digwp.com/u/370>. Also check out **WordPress Database Table Optimizer** <http://digwp.com/u/371> to automate the optimization process.
- **Use a completely different plugin** - The **WP DBManager** plugin <http://digwp.com/u/334> also makes it easy to optimize your database, along with a whole bunch of other great database-management features including automatic database backup and more.

Regardless of your method of choice, it is good practice to optimize your database on a periodic basis to help ensure optimal site performance.

load time for your regular visitors is to specify far-future “Expire Headers.” By setting an expiration date of, say, a month for your image files, visitors’ browsers will know not to re-fetch them, thereby reducing their load-time significantly. Here is an example of an HTAccess technique used for three types of images:

Cache Busting

Here are two good articles for ensuring that fresh files are displayed in the user’s browser, which is useful for site updates and development in general:

<http://digwp.com/u/640>

<http://digwp.com/u/641>

```
# EXPIRATION HEADERS FOR IMAGES
# note: 2592000 seconds = 1 month
ExpiresActive On
ExpiresByType image/gif A2592000
ExpiresByType image/png A2592000
ExpiresByType image/jpg A2592000
ExpiresByType image/tif A2592000
ExpiresByType image/ico A2592000
```

Keep in mind that with cache-control headers in place, you’ll need to change the name of any cached files in order for them to be refreshed in the user’s browser.

Reduce image sizes

Obviously, large images require more time to load than smaller ones. There are millions of articles out there explaining how to optimize your images, so we won’t reinvent the wheel here. Instead, here are some tools to help get the job done:

- **Online image optimization tools** - There are many great online services such as **Smush.it** <http://digwp.com/u/374> or **punypng** <http://digwp.com/u/375> that make it easy to upload and optimize your image sizes. Best of all, these free online services work on JPEG, GIF, and PNG images. Both services use “lossless” compression techniques to ensure that your images do not lose any of their original visual quality.
- **Offline optimization programs** - For more control over the details of your image optimization, check out **OptiPNG** <http://digwp.com/u/376> or **PNGcrush** <http://digwp.com/u/377>. Either of these powerful command-line programs will remove unnecessary data from your PNG images without reducing image quality. Both are excellent programs that work beautifully.

- **Optimization during image creation** - Arguably the best way to optimize your images is to create them optimally to begin with. There are many ways to do this, and there are plenty of excellent articles that will help you use programs such as **Photoshop** to take advantage of its image-optimization tools.

Disable unused/unnecessary plugins

Plugins consume resources. Running plugins you don't need is stupid. Go through your list and uninstall/delete any that you no longer need. Trust us, you'll be glad you did.

To Infinity, and Beyond

As we have seen in this chapter, there will always be a way to improve the performance of your WordPress-powered website. Out-of-the-box, it's going to work just fine for just about anything, but that is no reason to get complacent. Even with the smallest amount of effort, it is possible to improve your existing functionality and further optimize your current setup. From simply installing a few caching plugins to manually implementing custom functions, your site's maintainability, security, and performance can be advanced as far as you are willing to go.

Want to keep going?

The supplemental bonus PDF contains three more chapters packed with awesome WordPress features, tips & tricks. Download it for free in the DigWP Members Area:

<http://digwp.com/u/638>



More WordPress!

Hungry for more? Check out my new book, **The Tao of WordPress** — the complete guide for beginners, students, admins, users, and anyone who wants a fresh, minimalist approach to rocking it with WordPress.

Check it out @ WP-Tao.com

Thank you

Without a doubt, WordPress has revolutionized the way we build, operate, and even think about websites. We hope this book has given you a taste of the vast potential of WordPress and the endless possibilities that it provides.

What you do with WordPress is up to you, but we encourage you to use it in an intelligent and responsible way. With so much information available to WordPress users, it is easy to get lost in the details instead of focusing on what's important: sharing content with your readers.

One of the most exciting things about WordPress is that it makes creating and managing content so easy. Future versions of WordPress will continue to improve in terms of code, features, and potential. And we will be there to help you make the most of it all.

Thank you for sharing the journey with us :)



About the Authors



Chris Coyier



Chris Coyier is a real-world web designer who has been reaching for WordPress to power client sites for many years. He subscribes to the theory that not only is WordPress capable of powering any website it is almost always the right choice.

When he's not designing sites with WordPress, Chris Coyier shares his wisdom at CSS-Tricks.com, a wildly popular design community focusing on CSS, web design, and everything in between.

Jeff Starr



Jeff Starr has been designing & developing WordPress-powered sites since 2005. He develops WordPress plugins, creates WordPress themes, and writes lots of articles about WordPress, web security, and designing with Web Standards.

In addition to writing about WordPress, Jeff also writes in-depth articles on web design and development at PerishablePress.com, where he spends way too much time focusing on the details.

Continue the journey...



Learn more about WordPress at the book's companion site, where we cover many more awesome tips, tricks, and techniques to help you get the most out of WordPress.

<http://digwp.com/>

The Theme Clubhouse



Looking for a great theme for your WordPress site? Check out the growing collection of themes available at the DigWP Theme Clubhouse!

<http://themeclubhouse.digwp.com/>

WRITTEN BY WORDPRESS VETERANS
Chris Coyier and Jeff Starr, *Digging Into*
WordPress is packed with everything you need
to get the most out of WordPress. Suitable for
beginners, perfect for intermediate users, and a
great comprehensive reference guide for advanced
WordPress code wranglers.

WITH PLENTY OF JUICY CODE EXAMPLES, *Digging*
Into WordPress teaches you how
to improve your WordPress site using modern
techniques, tips, and tricks. Using clear, easy to
follow steps, *Digging Into WordPress* guides you
with the tools and knowledge to build incredible
sites that are powerful, secure, and awesome.

*Thanks for teaching
me to fish.*

-Eddie

<http://digwp.com>

