THE WORDPRESS ESSENTIALS

COPYRIGHTED MATERIAL

Chapter 1: Anatomy of a WordPress Install

Chapter 2: The WordPress Syntax

Chapter 3: The Loop

1 ANATOMY OF A WORDPRESS INSTALL

Installing WordPress is neither difficult nor time consuming, and the available instructions on `wordpress.org` are more than adequate to guide you through the basic install. That being said, there are some things that you should know and take into account if you want to set up the perfect WordPress site. Therefore, this chapter is all about giving you the solid platform you need for further development. WordPress in itself is a powerful publishing tool, and you can supercharge it with themes and plugins. Running a site on top of WordPress is all about that, so it is important to get the basics right so you can build on top of it. WordPress is the bricks and mortar of the site, but themes and plugins are what make it tick for real.

Also, before moving on, remember that "WordPress" in this book means the stand-alone version of WordPress available for free from `wordpress.org`. Don't get this mixed up with the multiuser version, called WordPress MU, which is touched upon briefly later, nor with AutoMattic hosted version on `wordpress.com`. This book is all about the main version available from `wordpress.org`, and more specifically with version 2.8 in mind.

The Basic Install

As you probably know, installing WordPress is a breeze. There is a reason the "five-minute install" PR talk isn't getting blasted to pieces by the otherwise so talkative blogosphere. In fact, the only reason that the install should take that long is because uploading the files sometimes takes time due to slow Internet connections or sluggish Web hosts. Most likely you'll have gone through a fair amount of WordPress installs yourself, so I'll be brief on this matter.

First, you need to make sure that your system meets the minimum requirements. The most recent ones can be found here: `wordpress.org/about/requirements/`. If your host supports PHP 4.3 or higher, and runs MySQL 4.0 or higher, then you're good. However, you should make sure your host has mod_rewrite installed since that will be needed for prettier links.

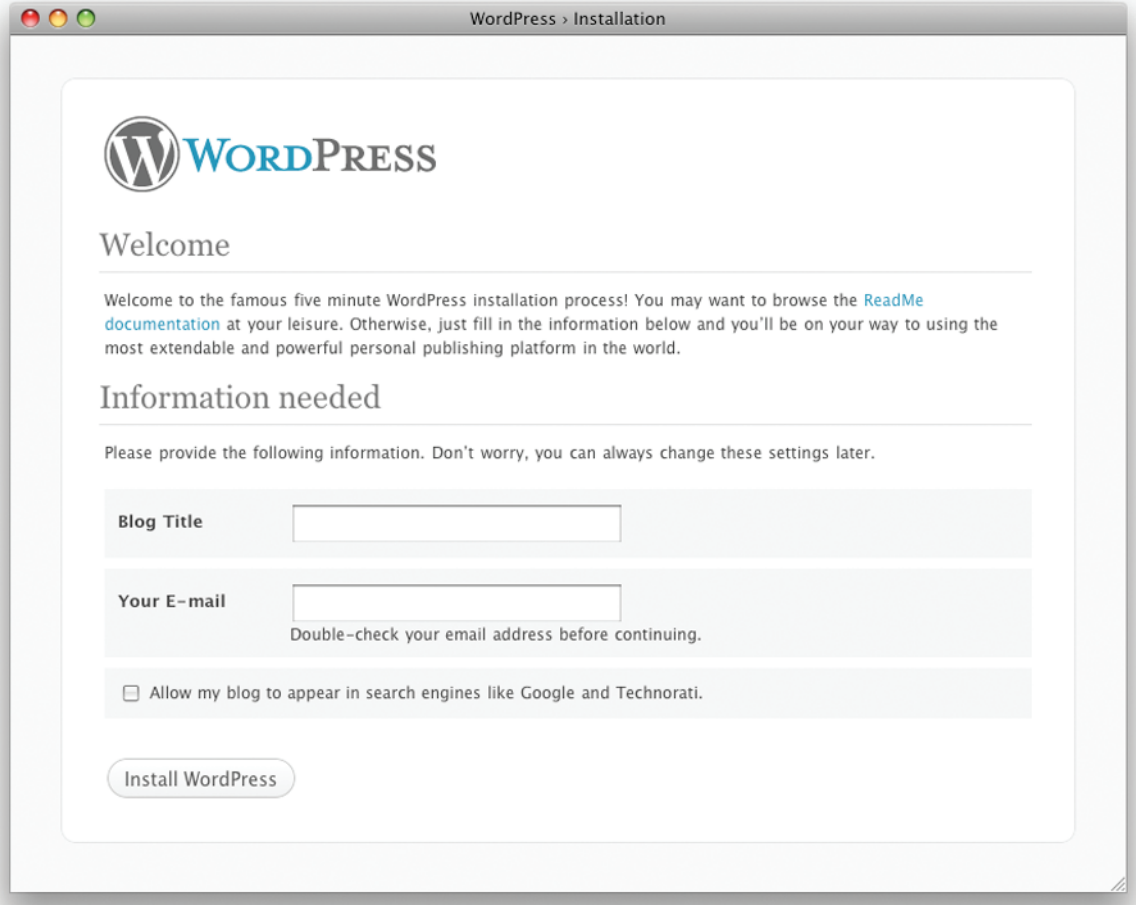

Figure 1-1: The Install interface

To install, you'll need the following:

- To download the most recent version of WordPress (from `wordpress.org/download/`).
- A MySQL database with a user that has write privileges (ask your host if you don't know how to set this up).
- Your favorite FTP program.

To install, unzip your WordPress download and upload the contents of the wordpress folder to your destination of choice on your server. Then, open wp-config-sample.php and find the database parts where you fill out the database name, and the username and password with write privileges. This is what wp-config-sample.php looks like:

```
define('DB_NAME', 'putyourdbnamehere');    // The name of the  database
define('DB_USER', 'usernamehere');     // Your MySQL username
define('DB_PASSWORD', 'yourpasswordhere'); // ...and password
define('DB_HOST', 'localhost');    // 99% chance you won't need to change this value
```

Next, still in wp-config-sample.php, find the part about Secret Keys. This part will start with a commented information text titled "Authentication Unique Keys" followed by four lines (as of writing) where you'll enter the Secret Keys. This is a security function to help make your install more secure and less prone to hacking. You'll only need to add these keys once, and while they can be entered manually and be whatever you like, there is an online generator courtesy of wordpress.org that gives you random strings with each load. Just copy the link (`api.wordpress.org/secret-key/1.1/`) to the generator from your wp-config-sample.php file and open it in your favorite Web browser. You'll get a page containing code looking something like this:

```
define('AUTH_KEY',         'PSmO59sFXB*XDwQ!<uj)h=vv#Kle')dBE0M:0oBzj'V(qd0.nP2|BT~T$a(;6-&!');
define('SECURE_AUTH_KEY', 'o>p3K{TD.tJoM74.Oy5?B@=dF_lcmlB6jm6D|gXnlJ#Z4K,M>E;[ +,22O?Lnarb');
define('LOGGED_IN_KEY',    'c}gR{389F*IG@/V+hg1 45J*H+9i_^HaF;$q(S[5Er[:DVOUjmS@(20E~t0-C*II');
define('NONCE_KEY',        'gz2D:n52|5wRvh)es:8OO|O ufZL@C|G.-w/H-E*}K:ygp4wI*.QHO-mUV_PR|6M');
```

Copy the contents from the generator page and replace the code shown below in wp-config-sample.php with them:

```
define('AUTH_KEY', '');
define('SECURE_AUTH_KEY', '');
define('LOGGED_IN_KEY', '');
define('NONCE_KEY', '');
```

By replacing the code above with the one from the generated page, you've made your install a little bit more secure from those nasty hackers.

The last thing you may want to change in wp-config-sample.php is the language. WordPress is in English (US English to be exact) by default, and if you're Swedish you would naturally want the default language to be Swedish, if you're German you would want German, and so on. To change the language, you'll need a language file (these are .mo files; most of them can be found here: `codex.wordpress.org/WordPress_in_Your_Language`) that you then upload to wp-content/language/. You also need to alter this little snippet in wp-config-sample.php to let WordPress know what language you want it to be in:

```
define ('WPLANG', '');
```

What you need to do is add the language code: this is the same as the language file, without the file extension. So if you really did want your install in Swedish, you'd download the sv_SE.mo, upload it to wp-content/languages/, and then pass the language to the `WPLANG` function, like this:

```
define ('WPLANG', 'sv_SE');
```

This won't necessarily make the themes or plugins you use display in your language of choice, but WordPress and its core functionality will, as will any code that supports it. We'll get to localization of themes and plugins in Chapter 6.

And that's it! Rename wp-config-sample.php to wp-config.php, and point your Web browser to your install location. This will show a link that initiates the install procedure, where you'll fill in the blog title, the admin user's e-mail address, and choose whether or not the blog should be open to search engines for indexing (most likely this will be the case, but if you want to fiddle with it first, then disable it; you can enable it in the settings later). After this, you'll get an admin username and a random password (save that!) and hopefully a success message along with a link to the blog.

Not very complicated, right?

Using an External Database Server

One of the most common issues when it comes to a failed WordPress install is that the MySQL database is located on a separate server. If you're getting database connection errors, and you're quite sure that both the username and password for the database user are correct, along with the full write capabilities, then this is most likely the case.

To fix this, just find this code snippet in wp-config.php (or wp-config-sample.php if you haven't renamed it yet) and change localhost to your database server:

```
define('DB_HOST', 'localhost');
```

What the MySQL server may be called depends on your host. It may be `mysql67.thesuperhost.com`, or something entirely different. Just swap localhost with this, and try and run the install script again.

Naturally, if you can't find your database server address you should contact your Web host and ask them for details.

Other Database Settings

You may want to consider some more database options before installing WordPress. (Probably not though, but still, they warrant mention.)

First of all, there's the database character set and collation. This is basically telling WordPress what character language the database is in, and it should just about always be UTF-8. This is also the default setting in wp-config-sample.php, hence you won't need to fiddle with it unless you have a special need to do so. If you do, however, this is what you're looking for:

```
define('DB_CHARSET', 'utf8');
```

That's the character set, with UTF-8 (obviously spelled out as `utf8` in code) by default.

The collation, which is basically the sort order of the character set that WordPress will apply to the MySQL database in the install phase, can be changed in this line:

```
define('DB_COLLATE', );
```


It is empty here, which means it will pass the character set in `DB_CHARSET` as the collation. By default, that is UTF-8, but if you need this to be something specific you can add it like this:

```
define('DB_COLLATE', 'character_set_of_choice');
```

A Few Words on Installers

Some Web hosts offer installers that will get your WordPress install up and running with just a click from within the Web host admin interface. The most popular one is probably Fantastico. At first, this sounds like a really good idea, since you won't have to fiddle with config files or anything; it'll just slap the blog up there and you can get started.

However, take a moment to do some research before going down this route. The most important aspect to consider is what version of WordPress the installer is actually setting up. Old versions shouldn't be allowed because they are outdated and, at worst, a security hazard. After all, with every WordPress release a lot of security holes are jammed shut, so it is not all about releasing funky new features for your favorite blogging platform.

Installers like Fantastico are great and can save time. However, if they don't install the latest version of WordPress you really shouldn't bother with them at all. If they do, then Google it just to make sure other users haven't reported anything weird going on, and if the coast is clear and you really don't want to do the five-minute manual install, then by all means go for it.

After having installed WordPress using an installer you should use the built-in upgrade feature, or perform upgrades manually using FTP should your host not support automatic upgrades. Make sure the installer doesn't do something strange with the install that stops you from doing this: you don't want to be tied to the installer script for updates.

Moving the WordPress Install to a Different Directory

If you're like me, your dislike of clutter goes as far as your Web hosting environment. In other words, the mere thought of all those WordPress files and folders in the root of your domain makes you nauseous. This may not be such an issue for others, although it will be easier to manage your various Web endeavors if you put the WordPress install in its own folder. Say you want to add other Web software installs; you may have a hard time finding the files you need if they're all mixed in together (although it helps that everything WordPress at this level is named wp-something). It just gets messy if you want to do anything other than just use WordPress.

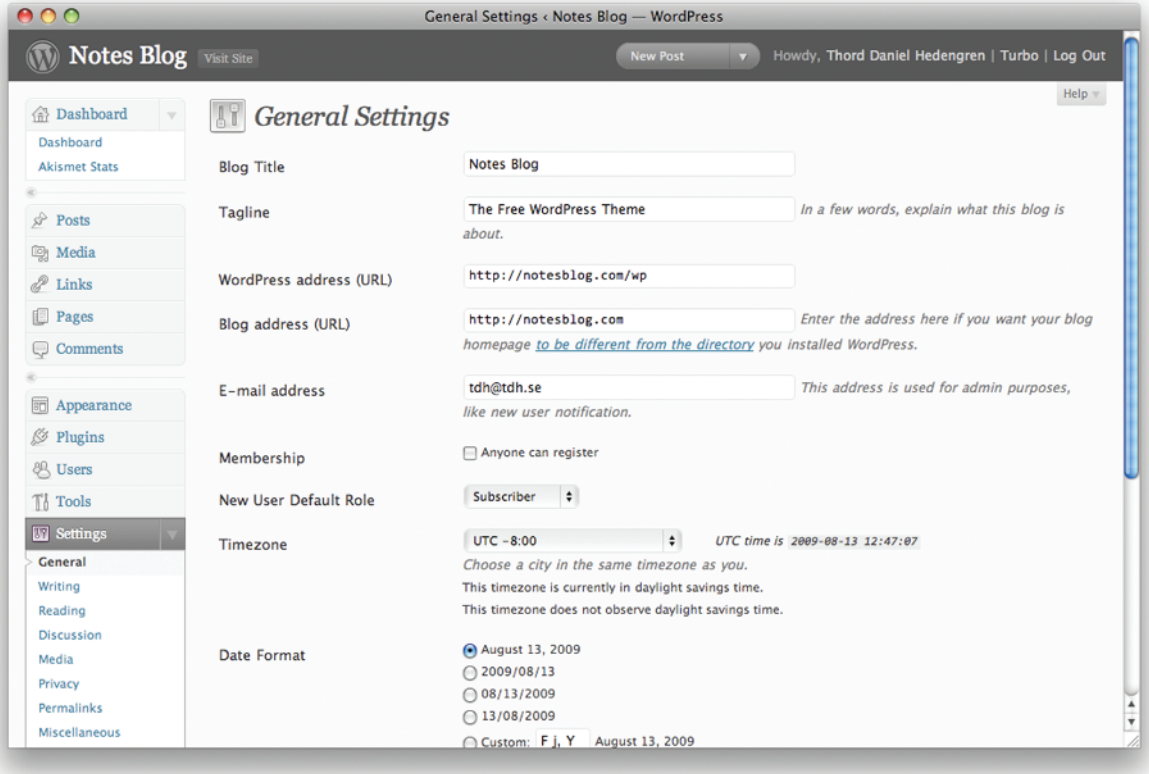

Figure 1-2: General Settings, found under General in the Settings part of the Admin Interface

Installing to a subfolder is the same as installing to the root of a domain, so I won't go into that. The purpose of this is to have the WordPress install in a subfolder, but have the blog displaying as if it were in the root folder, and keep the root folder on the server clean. You can either install WordPress to the subfolder directly, or to the root and then move the files to a subfolder. How you decide to tackle it is up to you; they are both easy to do.

You should really set up permalinks before doing this, since you'll want them to work regardless. The permalink options are found under Settings → Permalinks.

The first thing you should do is create the folder in which you want to put the WordPress install. Then, go to the General Settings page (see Figure 1-2) and change the WordPress address URL field to where you want to move your install to, and the Blog address URL field to where you want your site to be. Then click the update button and move all the WordPress files to their new directory except for the index.php and .htaccess files, which should be where you want your site to be. When they are there, open index.php and change this to match where you moved your WordPress install to:

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

That's a relative link with PHP trying to include wp-blog-header.php, which is where the WordPress magic starts. Just change the link to point to the file, which should be in your WordPress directory (whatever you've chosen to call it), and you'll be fine.

After this, login and update your permalinks structure again to get everything to point to their new destinations.

I think a quick example is in order. Say you have WordPress installed in the root folder (domain.com), and want it to be in a subfolder called wpsystem instead while keeping the actual site in root. That means that when people visit domain.com they'll see your WordPress site, but when you log in and manage it you'll do that within the wpsystem folder (or `domain.com/wpsystem/wp-admin/`, to be precise).

Now, this means that you'll need to change the WordPress address URL to `domain.com/wpsystem`, because that's where you want the WordPress install to be, and the blog address URL to `domain.com`, because that's where you want the site to be. Save these settings (don't worry if anything is acting funky just now), and move all the WordPress files to the domain.com/wpsystem folder except index.php and .htaccess, which you put in the root of domain.com since that's where the site is supposed to be. Then, open index.php and locate this code snippet:

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

And replace it with this code snippet:

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

As you can see, the code now points to the wpsystem folder instead, and to the wp-blog-header.php file.

Log in to the WordPress admin interface (which is now on `domain.com/wpsystem/wp-admin/`) and update the permalinks, and there you have it.

Hacking the Database

Most of the time you needn't worry about the database; WordPress will do that for you. There are database changes between versions sometimes, but program updates will take care of everything, and other than keeping a backup of your content the database can be left to live its own life.

That being said, if things go wrong you may need to do some edits in the database to fix them. Common issues are password resets, weird URLs as a result of a failed move, domain name changes, and not forgetting the dreaded widget issue.

Before moving on, you should remember that making alterations in the database is serious stuff. There are no undos here; what is deleted is deleted for good. Even if you know what you're doing you should always make a fresh backup before altering anything at all. Should you not know your way around a MySQL database and PhpMyAdmin, then don't mess with it until you do. You will break things.

Where Everything Is

Finding your way in the WordPress database is pretty easy. It consists of 10 tables, which in turn are full of content. Just browsing the database should answer most of your questions, and any edits can be made right then and there if you know what you're after. Naturally, there is a full database description in the documentation (`codex.wordpress.org/Database_Description`) and you should consult that whenever you need to find something.

The 10 main tables are:

- wp_comments: contains all comments
- wp_links: contains added links and links data
- wp_options: the blog options
- wp_postmeta: metadata for the posts
- wp_posts: the actual posts
- wp_terms: categories and tags
- wp_term_relationships: associates categories and tags with posts
- wp_term_taxonomy: descriptions for categories and tags
- wp_usermeta: user metadata
- wp_users: the actual users

All these tables are important, of course, but if you need to fix or change something directly in the database, chances are that it is in wp_options (for blog settings, like URLs and such), wp_posts (for mass editing of your blog posts), or wp_users (for password resets and such).

Fixing Issues by Hacking the Database

One of the more common issues with WordPress upgrades is the widgets going crazy, sometimes outputting only a blank page on your blog. While this seems to be a lot less common these days, the upgrade instructions still state that you should disable all plugins and revert to the default theme. If you do, most likely you'll never get that blank page.

However, should you get a blank page, it is probably a widget issue. A possible solution is to clean out the widgets in the database; they are hiding in the wp_options table. Exactly what you need to do and what the various widgets are called depends on what plugins you have, so tread carefully. Most likely the data is named in a way that seems logical compared to the plugins you use, and with that in mind you should be able to find what you're looking for. It may sound a bit hazardous, but it is worth giving it a go should you encounter a blank screen on your blog after an upgrade.

Another issue you may want to resolve in the database is changing or resetting a password for a user. You can't actually retrieve the password from the database because it is encrypted and all you'll see is gibberish, but you can change it to something else. Just remember that the passwords needs the MD5 treatment, which can be done through PhpMyAdmin or just about any MySQL managing tool you may use. Basically, what you do is type the new password in plain text, and choose MD5 for that particular field. You'll end up with a new line of gibberish, which actually says what you typed in the first place. Again, if this sounds scary to you, don't do it without exploring other solutions first!

Finally, you may want to mass edit your posts. Maybe you've got a new domain and want to change the source for all images you've used over the years, from `olddomain.com/wp-content/image.jpg` to `newdomain.com/wp-content/image.jpg`, for example. There are plugins that will help you with this, so most of us should probably check them out first. If you're comfortable with the database, though, you can run a SQL query to search for all these elements and replace them with the new ones. It could be some thing like this:

```
UPDATE wp_posts SET post_content = REPLACE (
post_content,
'olddomain.com/wp-content/',
'newdomain.com/wp-content/');
```

That would search the wp_posts table for any mention of `olddomain.com/wp-content/` and replace it with `newdomain.com/wp-content/`. That in turn would fix all the image links in the example above. Nifty little SQL queries for batch editing can come in handy, but remember: there are no undos here and what's done is done, so make sure you've got backups before even considering doing these things.

Backing Up

Anyone who has lost data in a hard drive crash or similar knows the importance of backing up, and it goes without saying that this applies to your online content as well. Backing up WordPress is actually a two-step process, since your blog consists of both a database (with all the content) and static files (image uploads and other attachments). Then you have your theme, your plugins, and so on, that you may or may not have altered but still don't want to lose because doing so would mean that you would have to collate them all over again. In fact, as of the inclusion of automatic updates within the admin interface in WordPress (a great feature in itself), backing up these things has become even more important.

The only thing you can lose without it causing too much trouble is the core WordPress files. These you can always download again, although you may want to keep a copy of wp-config.php somewhere safe.

For your database backup needs, several options are available. The most obvious one would be to use a Web interface like PhpMyAdmin and just download a compressed archive containing the data, and that is all well and good. However, you need to remember to do it on a regular basis, and that may be a problem. Also, PhpMyAdmin and similar database management interfaces aren't exactly the most user-friendly solutions out there, and most of us would rather not mess around with the database more than we truly have to.

Enter the wonderful world of WordPress plugins, and especially one called wp-db-backup. This plugin, which is featured in Chapter 11 in full, will let you set up various rules for database backups, and have your plugins stored on a server, e-mailed to you, or otherwise backed up, at regular intervals.

That's the database content; now for the static files. This is very simple: just keep backing up the wp-content folder. This folder contains all your uploads (images, videos, and other files that are attachments to your blog posts) along with your themes and plugins. In fact, it is the only part in the WordPress install that you should have been fiddling with, not counting the wp-config.php file, the .htaccess file, and possibly the index.php file in the root folder. Backing up wp-content will save all your static files, themes, plugins, and so on, as long as you haven't set up any custom settings that store data outside it.

So how can you backup wp-content? Unfortunately, the simplest backup method relies on you remembering to do so, which of course is downloading it using an FTP program. Some Web hosts have nifty little built-in scripts that can send backups to external storage places, such as Amazon S3 or any FTP server, really. This is a cheap way to make sure your static data is safe, so you should really look into it and not just rely on remembering to make an FTP download yourself. In fact, these built-in solutions often manage databases as well, so you can set up a backup of that as well. Better safe than sorry, after all.

The last stand, and final resort should the worst happen to your install, is the Web host's own backup solution. There is no way anyone can convince me to trust that my Web host, no matter

how good they may be, will solve any matter concerning data loss. Some are truly doing what they claim, which may be hourly backups, RAID disks, and other fancy stuff, but even the most well thought out solution can malfunction or backfire. Most hosts have some automatic backup solution in place, but what happens if the whole datacenter is out for some reason, or there's a power outage? You may not think that this could happen today, but if Google can go offline, so can your Web host.

In other words, make sure you have your very own backup solution in place. I hope you'll never have to use it, but if you do, you'll be happy you thought it through from the start.

WordPress and Switching Hosts

There are several ways of moving to a new server. My preferred one is using the Export/Import functionality found under Tools in WordPress admin (see Figure 1-3). However, before moving, make sure your WordPress install is up to date. Then, go to Tools and choose to export the content. You'll get a file containing the data.

Next, install WordPress on your new server. Any decent Web host will have alternate URLs to access your content on the server online, without actually having to have your domain pointing to

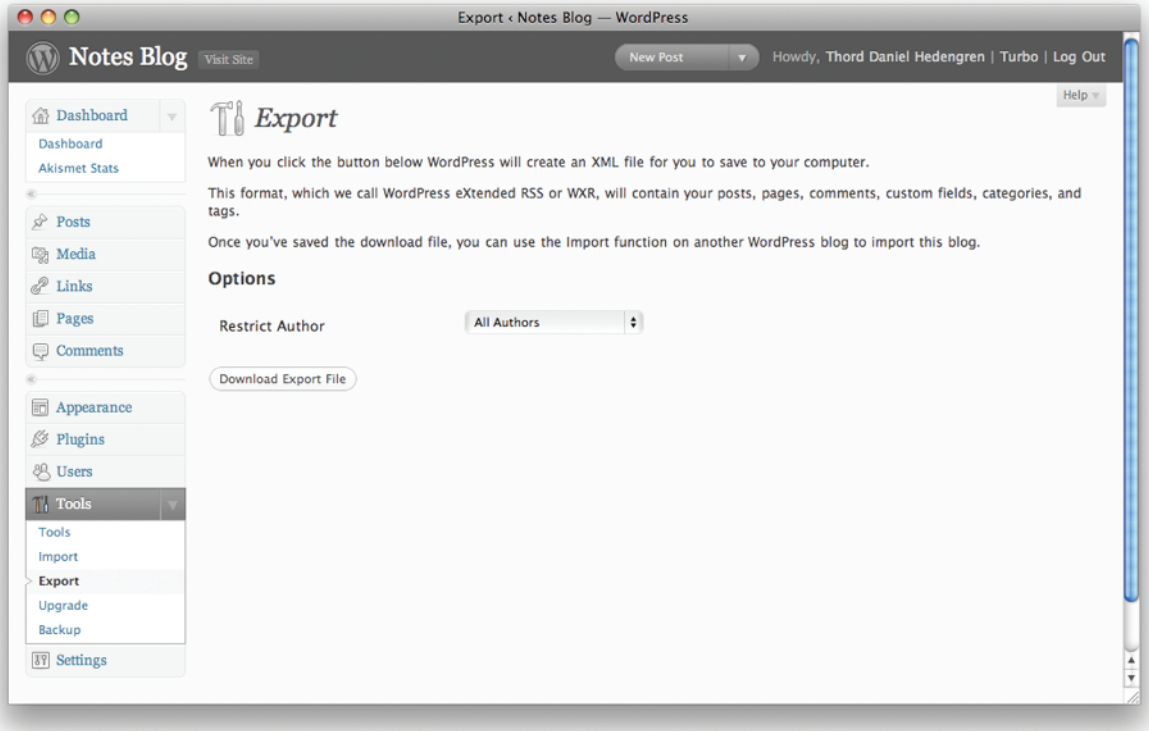

Figure 1-3: Exporting data

it. When you've got a running WordPress install, delete the automatic pages and posts since these won't be overwritten. You want the install to be clean.

Now, download the wp-content folder from your old server, and upload it to your new one. Now you've got all your images, plugins, themes, and so forth in place. There is a built-in option in the post importer that will try to download the images from your posts to your new server, but it fails more often than not, so it is better to manage the static files in wp-content manually using your favorite FTP program.

Finally, you're ready to import the exported file from your old server. Just go to Tools and go through the import wizard (see Figure 1-4), taking care that your exported file from the old server is up to date. Import it, let the script chew through the content, and then you're all done! Verify that everything is working properly, give yourself a pat on the back, and then redirect your domain to your new server. You may have to edit your new blog's settings, since it may have taken URLs from the Web host's internal system, so change them to correspond with your blog's domain name. While waiting for the domain to be pointed to your new server the blog will break of course, but then again your old one is still working. You may want to close comments on it, though, since those will be "lost" when the visitor is suddenly pointed to the new server with your new WordPress install, which is based on the content of your old one at the point when you exported the file.

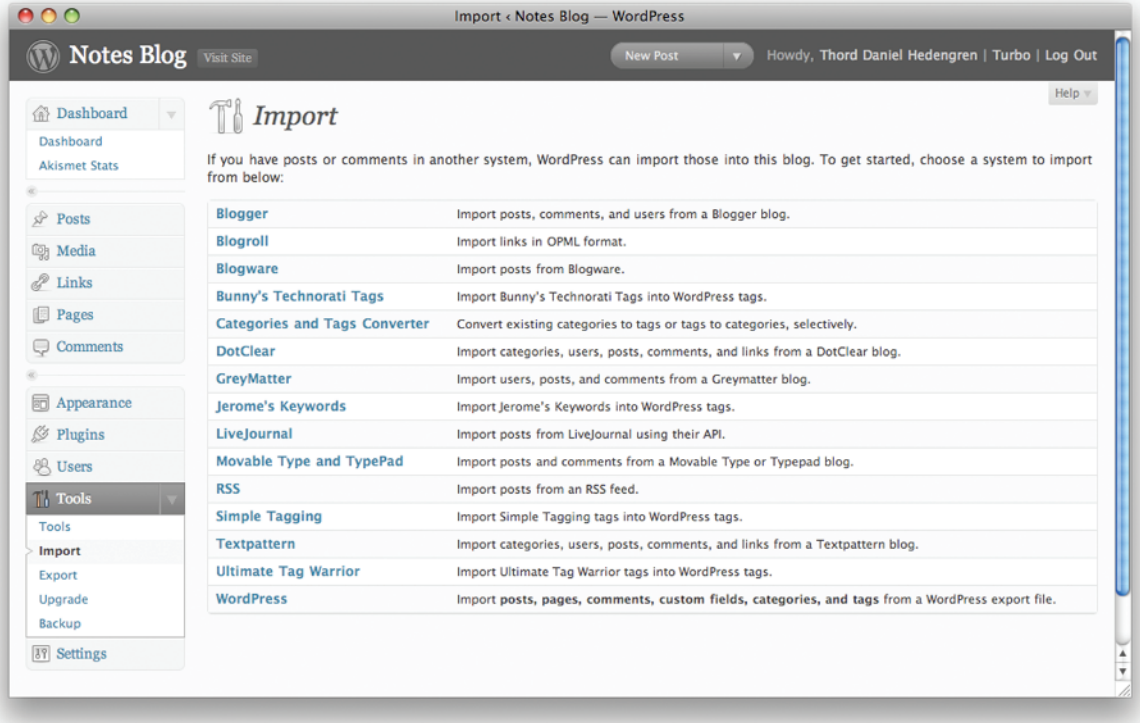

Figure 1-4: WordPress can import from a number of systems, but you want WordPress this time since that's what you exported from

When Export/Import Won't Work

Unfortunately, there are times when the Export/Import way won't work—usually because there is just too much content for PHP to parse in the import. This is only an issue if you have a big blog, and possibly due to your host's server settings as well.

If this is the case, you'll have to do things a little bit differently. Ideally, you can recreate your environment identically on your new server, with the same database name, and the same username and password to manage it. If you can do this, moving will be a breeze. All you have to do is get a dump from the MySQL database using your favorite MySQL admin tool, and then import it into the new one. This probably means using PhpMyAdmin and the backup instructions from the WordPress Codex (found at `codex.wordpress.org/Backing_Up_Your_Database`). Here's how you do it:

1. Log in to PhpMyAdmin and select the database you want to back up.
2. Click the Export tab in the top menu.
3. On the left-hand side, make sure all the necessary tables are marked (the Select All link will help). This would be all of them, unless you have other stuff in the same database as well.
4. On the right-hand side, you want to tick the Structure box checkbox, then the Add DROP TABLE box and Add AUTO_INCREMENT as well as Enclose table and field names with backquotes. Also tick the Data checkbox, but leave the choices within unchecked.
5. Down below, tick Save as file and pick the kind of file you want to download, probably a zipped one.
6. Click the Go button. This will download the database, which you will import on your new server.
7. Importing a dump in PhpMyAdmin is even easier. Make sure you have created a database with the same name, username, and password as you had on your previous server. This means you won't have to alter the wp-config.php file.

Import the dump to the new database by logging in with your favorite MySQL manager. If this is PhpMyAdmin, just select the database and choose the Import tab (sits next to the Export tab) at the top. Use the importer to find your downloaded dump, and import it.

Then download your full WordPress install from your old server, and upload it in an identical manner to your new one. Again, give it a spin using your Web host's temporary addresses and make sure that everything seems to be working. Point the domain to the new server, and when it resolves everything should be running smoothly.

However, you may not be able to recreate the environment in exactly the same way. If this is the case, just alter wp-config.php accordingly; most likely it is the database name, username and password, as well as possibly the need for an external database server, that you'll have to edit.

Moving WordPress from one server to another may seem scary at first, but it isn't as bad as it once was. Sure, if you've got a big blog and aren't comfortable doing stuff in database admin interfaces like PhpMyAdmin, then this may be a bit much. Get help, or give it a go yourself. Just make sure that you have all the backups you could possibly need, and don't mess things up on your old

(current) server, but rather on the new one. After all, you can always just create a new database and WordPress install there and give it another go.

How to Make a WordPress Install More Secure

There are a few simple things you can do to make your WordPress install more secure, and a few that are pretty much hardcore. The first and foremost one, however, is to keep WordPress up to date. Each new version removes a bunch of security holes, bugs, and other possible exploits that can make your install vulnerable, and not updating regularly means you won't get these fixes.

This brings us to the first tip. Check your theme's header.php file to see if the following code is there (it almost always is):

```
<?php remove_action('wp_head', 'wp_generator'); ?>
```

Then remove it! What it does is output what version of WordPress you're using, and while that may be a nice thing for bots and spiders looking for statistics, it's not worth the additional risk it brings. After all, if a certain version is known to have an open security hole, and people are looking for installs of that version to exploit, why make it easier on them and tell them outright?

You should also make sure that your wp-config.php file has the Secret Keys. Those make the install more secure. If you have an old version of WordPress and haven't bothered with the wp-config.php file in a while, you should at the very least add the four Secret Key lines to your file. You can get them from here: `api.wordpress.org/secret-key/1.1/`. You'll remember the Secret Keys from the installation instructions earlier in this chapter: just add them in the same way as you do when doing a brand-new install.

Users and Passwords

The first thing I do after having installed WordPress is to create a new user with admin privileges, log in with that user, and delete the default "admin" one. Why? Because everyone knows that if there is a user named admin, then that account has full admin capabilities. So if you wanted to hack your way into a WordPress install, you'd start by looking for the admin user to try to brute force a login. Once you're in via this method, you can do anything you want. So it's worth getting rid of the admin user, after you have logged in for the first time and created a proper account, because it has fulfilled its purpose.

That being said, deleting the admin user won't guarantee that hackers won't find another user to build their attempts on. If you have user archives on your blog, those will give you away. One solution would be to not display these, nor any links to an author page (other than ones you've created outside of WordPress's own functionality), but what do you do if you feel you need them?

The solution is to keep account credentials sparse. There is no need to have an administrator account for writing or editing posts and pages; an editor's credentials are more than enough. Granted, should an account with editor status be hacked then it will be bad for your site because

the editor can do a lot of things, but at least it is not an administrator account and that will keep the worst things at bay. And besides, you keep backups, right?

Besides questioning the types of accounts you and your fellow users have, passwords are another obvious security risk. You've probably been told to use a strong password, to make it long and to use letters, numbers, special characters, and so on. Do that: the more complicated the password is, the harder will it be to crack.

Server-side Stuff

The MySQL user for your WordPress database, which incidentally shouldn't be shared with any other system, doesn't actually need all write privileges. In fact, you don't need to be able to lock tables, index, create temporary tables, references, or create routines. In other words, you can limit the capabilities somewhat to make the system more secure.

Another thing some people will tell you to do is add extra logins using Apache's .htaccess. I don't do that myself because these login forms are annoying. Besides, there are plugins that can do the job better (see Chapter 11 for more information).

One thing you may want to do is make sure that there is an empty index.php or index.html file in every folder that doesn't have an index file. This is usually the case by default in WordPress, but it doesn't hurt to check. What this does is make it impossible to browse the folders directly, something that some Web hosts support.

Another server-side issue is forcing SSL encryption when logging in to the WordPress admin. This means that the traffic sent when you're doing your thing in the admin interface will be a lot harder to sniff out for potential bad guys. It's pretty easy to force SSL; just add this code snippet to your wp-config.php file, above the "That's all, stop editing! Happy blogging" comment:

```
define('FORCE_SSL_ADMIN', true);
```

SSL won't work without support from your host. Some Web hosts give you all you need to start this service from within their admin interface, but others will have to activate it for you, and may even charge you for it.

Summary

It doesn't matter if this is your first foray into the wonderful world of WordPress, or if you're an experienced user and developer. The important thing is that you have the basic installation figured out, have made it secure, and understand the publishing beast that is WordPress. From here on you'll start building sites and creating plugins to achieve your goals.

Next up is diving into what makes WordPress tick. That means you'll get to play with the loop, start looking at themes and plugins, and hopefully also activate that idea machine in the back of the head

that comes up with all those cool adaptations. The brilliance of WordPress is that it is so flexible and that you can build so many things with it, and the mere fact that it is so means that just thinking about the possibilities will undoubtedly inspire you.

If you have a WordPress install to play with (preferably something that isn't too public, since you may break something), get your sandbox set up and get ready to dive into the WordPress syntax.