

Chapter 1: Managing Your Servers

In This Chapter

- ✓ Understanding the client/server relationship
- ✓ Reviewing tools for client-side development
- ✓ Gathering server-side development tools
- ✓ Installing a local server with XAMPP
- ✓ Setting essential security settings
- ✓ Choosing a remote server
- ✓ Managing the remote servers
- ✓ Choosing and registering a domain name

Web pages are a complex undertaking. The basic web page itself isn't too overwhelming, but web pages are unique because they have meaning only in the context of the Internet — a vastly new undertaking with unique rules.

Depending where you are on your web development journey, you may need to understand the entire architecture, or you may be satisfied with a smaller part. Still, you should have a basic idea of how the Internet works and how the various technologies described in this book fit in.

Understanding Clients and Servers

A person using the web is a *client*. You can also think of the user's computer or browser as the client. Clients on the Internet have certain characteristics:

- ◆ **Clients are controlled by individual users.** You have no control over what kind of connection or computer the user has. It may not even be a computer but may be instead a cellphone or (I'm not kidding) refrigerator.
- ◆ **Clients have temporary connections.** Clients typically don't have permanent connections to the Internet. Even if a machine is on a permanent network, most machines used as clients have temporarily assigned addresses that can change.

- ◆ **Clients might have wonderful resources.** Client machines may have multimedia capabilities, a mouse, and real-time interactivity with the user.
- ◆ **Clients are limited.** Web browsers and other client-side software are often limited so that programs accessed over the Internet can't make major changes to the local file system. For this reason, most client programs operate in a sort of "sandbox" to prevent malicious coding.
- ◆ **Clients can be turned off without penalty.** It doesn't really cause anybody else a problem if you turn off your computer. Generally, client machines can be turned off or moved without any problems.

Servers are the machines that typically host web pages. They have a much different set of characteristics:

- ◆ **Servers are controlled by server administrators.** A server administrator is responsible for ensuring that all data on the server is secure.
- ◆ **Servers have permanent connections.** The purpose of a server is to accept requests from clients. For this reason, a server needs to have an IP number permanently assigned to it.
- ◆ **Servers usually have names, too.** To make things easier for users, server administrators usually register domain names to make their servers easier to find.
- ◆ **Servers can access other programs.** Web servers often talk to other programs or computers (especially data servers).
- ◆ **Servers must be reliable.** If a web server stops working, anybody trying to reach the pages on that server is out of luck. This is why web servers frequently run Unix or Linux because these operating systems tend to be especially stable.
- ◆ **Servers must have specialized software.** The element that truly makes a computer a server is the presence of web server software. Although several options are available, only two dominate the market: Apache and Microsoft IIS.

Parts of a client-side development system

A development system is made up of several components. If you're programming on the client (using XHTML, CSS, and JavaScript), you need the following tools:

- ◆ **Web browsers:** You need at least a couple of browsers so that you can see how your programs behave in different ones. Chrome is especially useful for web developers because of its extensive developer toolset.

- ◆ **Text editor:** Almost all web development happens with plain-text files. A standard text editor should be part of your standard toolkit. I prefer Komodo Edit because it handles all the languages described in this book and works well on all desktop operating systems, and it's free. (I really like Emacs too, but I won't force that monster on anybody.)

For client-side development, you don't necessarily need access to a server. You can test all your programs directly on your own machine with no other preparation. Of course, you'll eventually want a server so that you can show your pages to everyone.



The client-side development tools listed here are described in more detail in Book I, Chapter 3.

Parts of a server-side system

When you start working on the server side (with PHP, MySQL, and AJAX), you need a somewhat more complex setup. In addition to everything you need for client-side development, you also need these items:

- ◆ **A web server:** This piece of software allows users to request web pages from your machine. You must either sign on to a hosting service and use its server or install your own. (I show you both techniques in this chapter.) By far the most common server in use is Apache. Web server software usually runs all the time in the background because you never know when a request will come in.
- ◆ **A server-side language:** Various languages can be connected to web servers to allow server-side functionality. PHP is the language I chose in this book because it has an excellent combination of power, speed, price (free), and functionality. PHP needs to be installed on the server machine, and the web server has to be configured to recognize it. See Book VI, Chapter 1 for a review of other server-side languages.
- ◆ **A data server:** Many of your programs work with data, and they need some sort of application to deal with that data. The most common data server in the open-source world is MySQL. This data package is free, powerful, and flexible. The data server is also running in the background all the time. You have to configure PHP to know that it has access to MySQL.
- ◆ **A mail server:** If your programs send and receive e-mail, you need some sort of e-mail server. The most popular e-mail server in the Windows world is Mercury Mail, and Sendmail is popular in the world of Unix and Linux. You probably won't bother with this item on a home server, but you should know about it when you're using a remote host.

- ◆ **An FTP server:** Sometimes, you want the ability to send files to your server remotely. The FTP server allows this capability. Again, you probably don't need this item for your own machine, but you definitely should know about it when you use a remote host.
- ◆ **phpMyAdmin:** There's a command-line interface to MySQL, but it's limited and awkward. The easiest way to access your MySQL databases is to use the phpMyAdmin program. Because it's a series of PHP programs, it requires a complete installation of PHP, MySQL, and Apache (but, normally, you install all these things together anyway).

Creating Your Own Server with XAMPP

If the requirements for a web hosting solution seem intimidating, that's because they are. It's much more difficult to set up a working server system by hand than it is to start programming with it.

I don't recommend setting up your own system by hand. It's simply not worth the frustration because very good options are available.

XAMPP is an absolutely wonderful open-source tool. It has the following packages built in:

- ◆ **Apache:** The standard web server and the cornerstone of the package
- ◆ **PHP:** Configured and ready to start with Apache and MySQL
- ◆ **MySQL:** Also configured to work with Apache and PHP
- ◆ **phpMyAdmin:** A data management tool that's ready to run
- ◆ **Mercury Mail:** A mail server
- ◆ **FileZilla FTP server:** An FTP server
- ◆ **PHP libraries:** A number of useful PHP add-ons, including GD (graphics support), Ming (Flash support), and more
- ◆ **Additional languages:** Perl, another extremely popular scripting and server language, and SQLite, another useful database package
- ◆ **Control and configuration tools:** A Control Panel that allows you to easily turn various components on and off



This list is a description of the Windows version. The Mac and Linux versions have all the same types of software, but the specific packages vary.

Considering the incredible amount of power in this system, the download is remarkably small. The installer is only 34MB.

XAMPP installation is pretty painless: Simply download the installer and respond to all the default values.

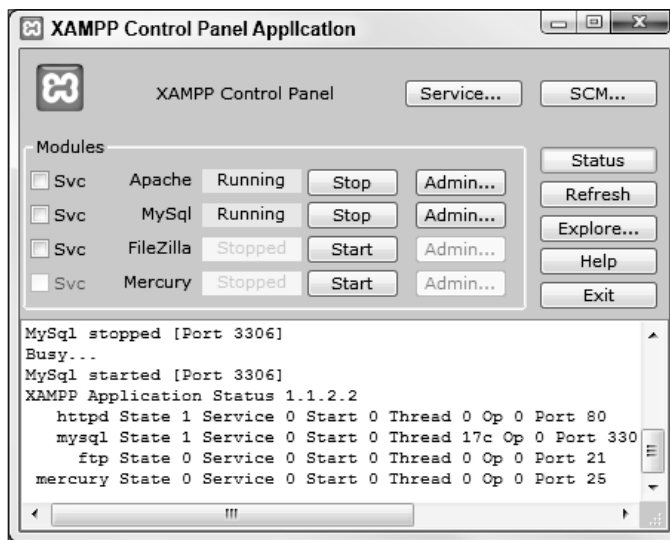


If you use Windows, you may want to change where the package is installed because the program files directory causes problems for some users. I normally install XAMPP in root of the C:\ drive on Windows installations. The default directory is fine for Mac and Linux.

Running XAMPP

After you install XAMPP, you can manage your new tools with the XAMPP Control Panel. Figure 1-1 shows this program in action.

Figure 1-1:
XAMPP
Control
Panel
allows
you to turn
features on
and off.



Some components of XAMPP (PHP, for example) run only when they're needed. Some other components (Apache and MySQL) are meant to run constantly in the background. Before you start working with your server, you need to ensure that it's turned on.

You can choose to run Apache and MySQL as a service, which means that the program is always running in the background. This arrangement is convenient, but it slightly reduces the performance of your machine. I generally turn both Apache and MySQL on and off as I need it.



Leaving server programs open on your machine constitutes a security hazard. Be sure to take adequate security precautions. See the section "Setting the security level," later in this chapter, for information on setting up your security features.

Testing your XAMPP configuration

Ensure that Apache and MySQL are running, and then open your web browser. Set the address to `http://localhost`, and you see a screen like the one shown in Figure 1-2.



Figure 1-2:
The XAMPP
main page.

This page indicates that XAMPP is installed and working. Feel free to experiment with the various items in the Demos section. Even though you may not know yet what they do, you should know what some of their capabilities are.

Adding your own files

Of course, the point of having a web server is to put your own files in it. Use your file management tool to find the XAMPP directory in your file system. Right under the XAMPP directory is the `htdocs` folder, the primary web directory. Apache serves only files that are in this directory or under it. (That way, you don't have to worry about your love letters being distributed over the Internet.)



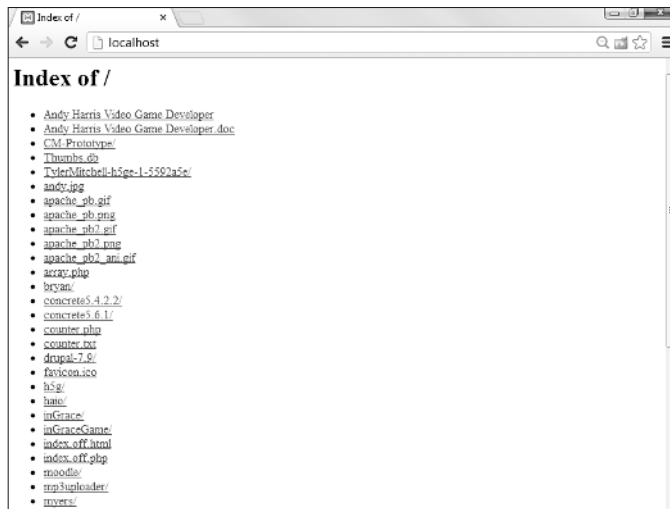
All the files you want Apache to serve must be in `htdocs` or in a subdirectory of it.

When you specified `http://localhost` as the address in your browser, you were telling the browser to look on your local machine in the main `htdocs` directory. You didn't specify a particular file to load. If Apache isn't given a filename and it sees the file named `index.html` or `index.php`, it displays that file, instead. So, in the default `htdocs` directory, the `index.php`

program is immediately being called. Although this program displays the XAMPP welcome page, you don't usually want that to happen.

Rename `index.php` to `index.php.old` or something similar. It's still there if you want it, but now there's no index page, and Apache simply gives you a list of files and folders in the current directory. Figure 1-3 shows my `localhost` directory as I see it through the browser.

Figure 1-3:
After
disabling
`index.php`,
I can see
a list of
files and
directories.



You typically don't want users to see this ugly index in a production server, but I prefer it in a development environment so that I can see exactly what's on my server. After everything is ready to go, I put together `index.html` or `index.php` pages to generate more professional directories.

Generally, you want to have subdirectories to all your main projects. I added a few others for my own use, including `hain`, which contains all the code for this book.



If you want to display the XAMPP welcome screen after you remove the `index.php` program, simply point your browser to `http://localhost/xampp`.

Setting the security level

When you have a web server and a data server running, you create some major security holes. You should take a few precautions to ensure that you're reasonably safe:

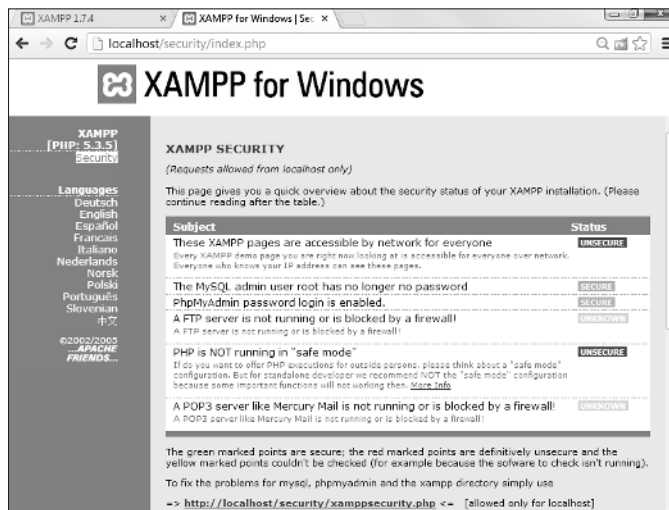


- ◆ **Treat your server only as a local asset.** Don't run a home installation of Apache as a production server. Use it only for testing purposes. Use a remote host for the actual deployment of your files. It's prepared for all the security headaches.
- ◆ **Run a firewall.** You should run, at an absolute minimum, the Windows firewall that comes with all recent versions of Windows (or the equivalent for your OS). You might also consider an open-source or commercial firewall. Block incoming access to all ports by default and open them only when needed. There's no real need to allow incoming access to your web server. You only need to run it in localhost mode.

The ports XAMPP uses for various tools are listed on the security screen shown in Figure 1-4.

- ◆ **Run basic security checks.** The XAMPP package has a handy security screen. Figure 1-4 shows the essential security measures. I've already adjusted my security level, so you'll probably have a few more "red lights" than I do. Click the security link at the bottom of the page for some easy-to-use security utilities.

Figure 1-4:
The XAMPP
Security
panel
shows a few
weaknesses.



- ◆ **Change the MySQL root password.** If you haven't already done so, use the security link to change the MySQL root password, as shown in Figure 1-5. (I show an alternative way to change the password in Book VI, Chapter 1.)
- ◆ **Add an XAMPP Directory password.** Type a password into the lower half of the security form to protect your XAMPP directory from unauthorized access. When you try to go to the XAMPP directory, you're prompted for this password.

Figure 1-5:
Changing
the MySQL
root
password.



Security is always a compromise. When you add security, you often introduce limits in functionality. For example, if you changed the root password for MySQL, some of the examples (and phpMyAdmin) may not work anymore because they're assuming that the password is blank. You often have to tweak. See Chapter 1 in Book VI for a complete discussion of password issues in MySQL and phpMyAdmin.

Compromising between functionality and security

You may be shocked that my example still has a couple of security holes. It's true, but it's not quite as bad as it looks:

- ◆ **The firewall is the first line of defense.** If your firewall blocks external access to your servers, the only real danger your system faces is from yourself. Begin with a solid firewall and ensure that you don't allow access to port 80 (Apache) or port 3306 (MySQL) unless you're absolutely sure that you have the appropriate security measures in place.
- ◆ **I left phpMyAdmin open.** phpMyAdmin needs root access to the MySQL database, so if anybody can get to phpMyAdmin through the web server, they can get to my data and do anything to it. Because my firewall is blocking port 80 access, you can't get to phpMyAdmin from anything other than localhost access, and it's not really a problem.
- ◆ **I'm not running a mail or FTP server on this machine.** The security system isn't sure whether my FTP or mail system is secure, but because I'm not running them, it isn't really a problem.



If you're having troubles getting Apache to start, take a look at the other programs you have running. Sometimes other programs use the same ports that XAMPP needs, and cause problems. Messaging programs (like Skype) are notorious for this. If you can't start Apache while Skype is running, turn off Skype (or the other offending software) until Apache is turned on. Typically you'll be able to run Skype after Apache is running.

Choosing a Web Host

Creating a local server is useful for development purposes because you can test your programs on a server you control, and you don't need a live connection to the Internet.

However, you should avoid running a production server on your own computer, if you can. A typical home connection doesn't have the guaranteed IP number you need. Besides, you probably signed an agreement with your broadband provider that you won't run a public web server from your account.

This situation isn't really a problem because thousands of web hosting services are available that let you easily host your files. You should consider an external web host for these reasons:

- ◆ **The host, not you, handles the security headaches.** This reason alone is sufficient. Security isn't difficult, but it's a never-ending problem (because the bad guys keep finding new loopholes).
- ◆ **The remote server is always up.** Or, at least, it should be. The web server isn't doing anything other than serving web pages. Your web pages are available, even if your computer is turned off or doing something else.
- ◆ **A dedicated server has a permanent IP address.** Unlike most home connections, a dedicated server has an IP address permanently assigned to it. You can easily connect a domain name to a permanent server so that users can easily connect.
- ◆ **Ancillary services usually exist.** Many remote hosting services offer other services, like databases, FTP, and e-mail hosting.
- ◆ **The price can be quite reasonable.** Hosting is a competitive market, which means that some good deals are available. Decent hosting is available for free, and improved services are extremely reasonable.

You can find a number of free hosting services at sites like <http://free-webhosts.com>.

Finding a hosting service

When looking for a hosting service, ask yourself these questions:

- ◆ **Does the service have limitations on the types of pages you can host?**
Some servers are strictly for personal use, and some allow commercial

sites. Some have bandwidth restrictions and close your site if you draw too many requests.

- ◆ **How much space are you given?** Ordinary web pages and databases don't require a huge amount of space, but if you do a lot of work with images, audio, and video files, your space needs increase dramatically.
- ◆ **Is advertising forced on you?** Many free hosting services make money by forcing advertisements on your pages. This practice can create a problem because you might not always want to associate your page with the company being advertised. (A page for a day care center probably should not have advertisements for dating services, for example.)
- ◆ **Which scripting languages (if any) are supported?** Look for PHP support.
- ◆ **Does the host offer prebuilt scripts?** Many hosts offer a series of pre-built and preinstalled scripts. These can often include content management systems, message boards, and other extremely useful tools. If you know that you're going to need Moodle, for example (a course management tool for teachers), you can look for hosting services that have it built in. (If a tool you want isn't there, make sure you have FTP access so you can install it yourself.)
- ◆ **Does the host provide access to a database?** Is phpMyAdmin support provided? How many databases do you get? What is the size limit?
- ◆ **What sort of Control Panel does the service provide?** Does it allow easy access to all the features you need?
- ◆ **What type of file management is used?** For example, determine how you upload files to the system. Most services use browser-based uploading. This system is fine for small projects, but it's quite inconvenient if you have a large number of files you want to transfer. Look for FTP support to handle this.
- ◆ **Does the host have an inactivity policy?** Many free hosting services automatically shut down your site if you don't do anything with it (usually after 30 to 90 days of inactivity). Be sure you know about this policy.
- ◆ **Do you have assurances that the server will remain online?** Are back-ups available? What sort of support is available? Note that these services are much more likely on a paid server.
- ◆ **How easily can you upgrade if you want?** Does a particular hosting plan meet your needs without being too expensive?
- ◆ **Does the service offer you a subdomain, and can you register your own?** You may also want to redirect a domain that you didn't get through the service. (See the section "Naming Your Site," later in this chapter, for information on domain names.)
- ◆ **What kind of support is available?** Most hosting services have some kind of support mechanism with e-mail or ticket systems. Some hosts offer live chat, and some have telephone support. Talking to a real human in real time can be extremely helpful, and this is often worth paying for.

Connecting to a hosting service

The sample pages for this book are hosted on Freehostia.com, an excellent, low-cost hosting service. You can find many great hosting services, but the rest of the examples in this chapter use Freehostia, where the examples for this book are hosted.

Choose whichever hosting service works for you. If you find a free hosting service that you really like, upgrade to a paid service. Hosting is a reasonably cheap commodity, and a quality hosting service is well worth the investment.

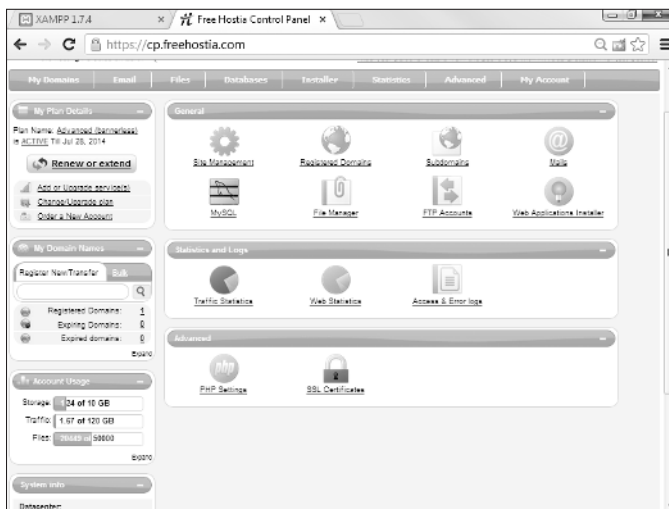
Managing a Remote Site

Obviously, having a hosting service isn't much fun if you don't have pages there. Fortunately, there are a lot of ways to work with your new site.

Using web-based file tools

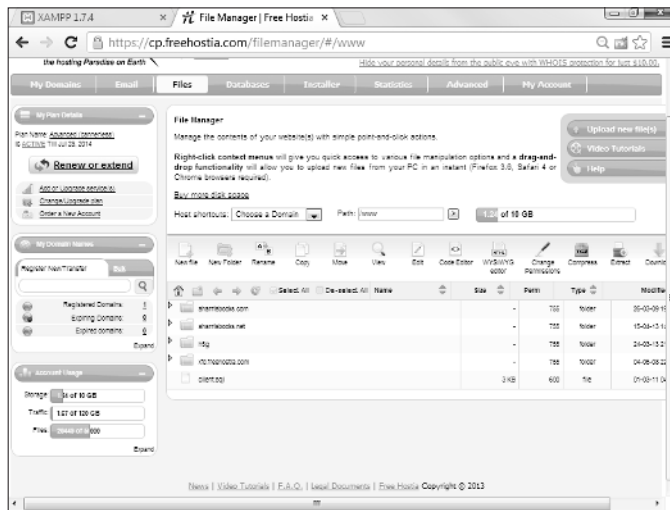
Most of the time, your host has some sort of Control Panel that looks like the one shown in Figure 1-6.

Figure 1-6:
This Control
Panel
allows you
to manage
your site
remotely.



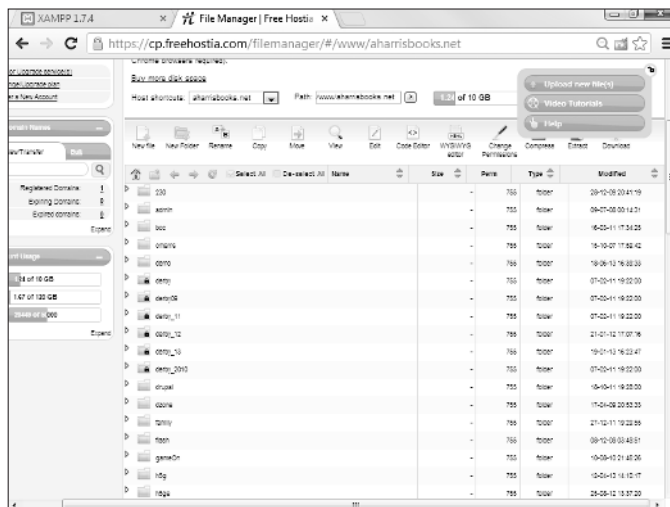
There's usually some sort of file management tool that might look like the one shown in Figure 1-7.

Figure 1-7:
This file management tool allows you to manipulate the files on your system.



In this particular case, all my web files are in the `www/aharrisbooks.net` directory, so I click to see them. Figure 1-8 shows what you might see in an actual directory.

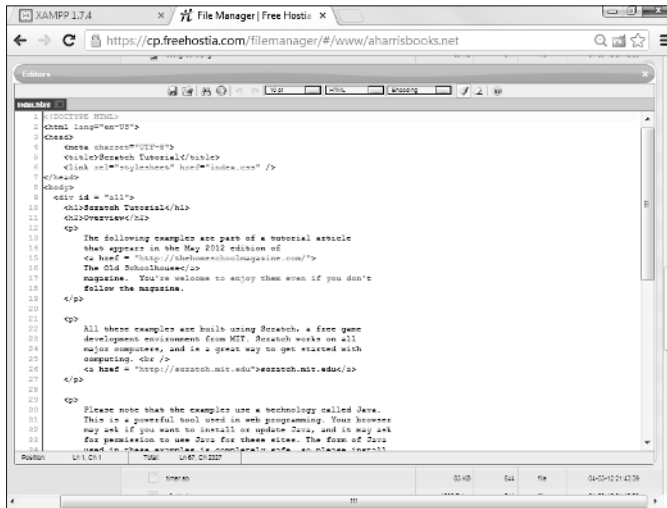
Figure 1-8:
Now, you can see some files here.



This page allows you to rename, upload, and edit existing files and change file permissions.

You can create or edit files with a simple integrated editor: Build a new file with the Create File button. Type a filename into the text area and click the button. You can also click the edit button next to a file, and the file will open in the editor. In either case, the text editor shown in Figure 1-9 appears.

Figure 1-9:
The hosting
service has
a limited
text editor.



You can write an entire website using this type of editor, but the web-based text editing isn't helpful, and it's kind of awkward. More often, you create your files on your own XAMPP system and upload them to the server when they're basically complete. Use server-side editing features for quick fixes only.

Understanding file permissions

Most hosting services use Linux or Unix. These operating systems have a more sophisticated file permission mechanism than the Windows file system does. At some point, you may need to manipulate file permissions.

Essentially, the universe is divided into three populations: Yourself, your group, and everybody else. You can allow each group to have different kinds of permission for each file. Each of the permissions is a Boolean (true or false) value:

- ◆ **Read permission:** The file can be read. Typically, you want everybody to be able to read your files, or else you wouldn't put them on the web server.
- ◆ **Write permission:** The file can be written, changed, and deleted. Obviously, only you should have the ability to write to your files.
- ◆ **Execute permission:** Indicates that the file is an executable program or a directory that can be passed through. Normally, none of your files is considered executable, although all your directories are.

What's with all the permissions?

Permissions are typically treated as binary numbers: 111 means “read, write, execute.” This (111 value) is also a 7 permission because 111 binary translates to 7 in base ten (or base eight, but let’s skip that detail for now).

A permission is read as three digits, each one a number indicating the permissions, so 644 permission means `rw- r- r-`. This example can

be translated as “The owner should be able to read and write this file. Everyone else can read it. Nobody can execute it.”

If you don’t understand this concept, don’t worry about it. The guidelines are very simple: Make sure that each of your files has 644 permission and that each directory has 755 permission. That’s all you really need to know.

Using FTP to manage your site

Most of the work is done on a local machine and then sent to the server in a big batch. (That’s how I did everything in this book.) The standard web-based file management tools are pretty frustrating when you want to efficiently upload a large number of files.

Fortunately, most hosts have the FTP (File Transfer Protocol) system available. *FTP* is a client/server mechanism for transferring files efficiently. To use it, you may have to configure some sort of FTP server on the host to find out which settings, username, and password you should use. Figure 1-10 shows the Freehostia Control Panel with this information displayed.



Figure 1-10:
Configuring
the FTP
server.

You also need an FTP client. Fortunately, many free clients are available. I like FileZilla, for a number of reasons:

- ◆ **It's free and open source.** That's always a bonus.
- ◆ **It works the same on every OS.** If I'm on Windows, Linux, or Mac, it works the same.
- ◆ **It's easy to use.** It feels a lot like a file manager.

Figure 1-11 shows FileZilla running in my browser.

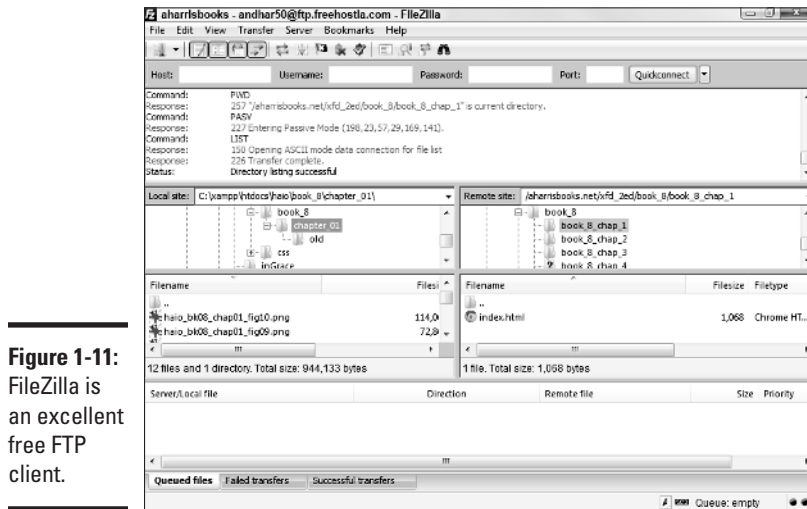


Figure 1-11:
FileZilla is
an excellent
free FTP
client.

Using an FTP client

FileZilla and other FTP programs all do pretty much the same thing. Here's how to use it:

1. Download and install FileZilla.

You can download FileZilla for free at <http://download-filezilla-ftp-free.com/>. (There is also a link at my main page: www.aharrisbooks.net.)

2. Gather the login information.

You'll need to get your FTP login information from your service provider. Normally this consists of a special address (like a URL, but it begins with `ftp://`), a username, and a password. These are not necessarily the same credentials used to log in to the server.

3. Enter host information.

Use the site manager (Ctrl+S or File⇨Site Manager) to manage your site. Select the New Site button to build a new connection.

There's a place in the dialog box to enter your login information. Put the address (which usually begins with `ftp://`) in the `host` box, with your username and password in the other boxes. You can typically leave the `port` box blank, as this information is normally determined automatically. (If in doubt, try port 21 or 22.) If an ordinary FTP connection doesn't work, check with your server to see if you need to use SFTP (a more secure variant) instead. If so, just select the appropriate encryption method (provided by your server) in the Encryption field. Once you've made the connection, SFTP acts almost exactly like FTP. Figure 1-12 shows the Site Manager dialog box.

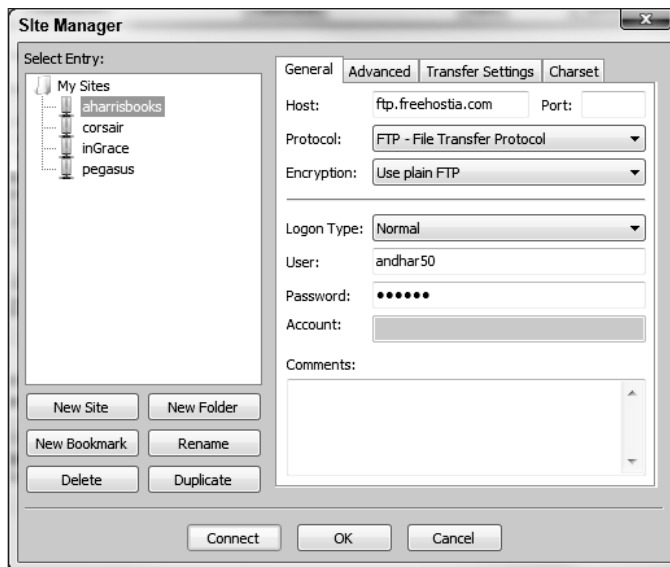


Figure 1-12:
Setting
up an FTP
account
with
FileZilla.

4. Connect to the FTP server.

Click **Connect** to make the connection. A flurry of obscure messages flies through the top panel. In a few seconds (if all went well), you'll see a directory listing of the remote system in the right-hand panel.

5. Use the left panel to manage local files.

The left-hand panel controls the local file system. Use this to find files on your local computer. It's a normal file management system like *My Computer* or *Finder*.

6. Use the right panel for remote files.

The right-hand panel controls the remote server file system. It works exactly like the local system, except it allows you to manipulate files on the remote system. Use this system to move to the appropriate directory on the remote system. You can also create a new directory or rename files with the appropriate buttons on this screen.

7. Drag files to transfer them.

To transfer files between machines simply drag them. Drag from the local machine to the remote machine to upload, or in the other direction to download them. You can move many files at a time in this manner.

8. Watch for errors.

Most of the time, everything works great, but sometimes there is a problem. The bottom panel shows potential error messages. If there is an error, you may need to reload a file.



Most remote servers run some variation of the Unix operating system. You may not be familiar with Unix, but it really works a lot like the systems you already know. However, it has one feature that may be new to you: file permissions. Most of the time, an FTP program automatically gets the file permissions right, but if the browser cannot see a file after you upload it to the server, try right-clicking that file in FileZilla and look at its properties. Most web files should have a permission set called 644 (which means you can read and write the file, everyone else can read it, and nobody can run it on the server). If it is set to something else, try changing it to 644. Web directories should typically have 755 permission, which is almost always the default.



FTP is a completely unsecure protocol. Anything you transfer with FTP is completely visible to any bad guys sniffing the Internet. For this reason, some servers use a different protocol: Secure FTP (SFTP). Filezilla supports this and other protocols your server might use.

Naming Your Site

After you have a site up and running, you need to give it an address that people can remember. The Domain Name System (DNS) is sort of an address book of the entire Internet. DNS is the mechanism by which you assign a name to your site.

Understanding domain names

Before creating a domain name, you should understand the basics of how this system works:

- ◆ **Every computer on the Internet has an IP (Internet Protocol) address.** When you connect to the Internet, a special number is assigned to your computer. This *IP address* uniquely identifies your computer. Client machines don't need to keep the same address. For example, my notebook has one address at home and another at work. The client addresses are dynamically allocated, and that's fine. But a server needs a permanent address that doesn't change.

- ♦ **IP addresses are used to find computers.** Any time you request a web page, you're looking for a computer with a particular IP address. For example, the Google IP address is 66.102.9.104. Type it into your browser address bar, press Enter, and you see the Google main page.
- ♦ **DNS names simplify addressing.** IP numbers are too confusing for human users. The Domain Name System (DNS) is a series of databases connecting website names with their associated IP numbers. When you type `http://www.google.com`, for example, the DNS system looks up the text `www.google.com` and finds the computer with the associated IP.
- ♦ **You have to register a DNS name.** Of course, to ensure that a particular name is associated with a page, you need to register that relationship.

Registering a domain name

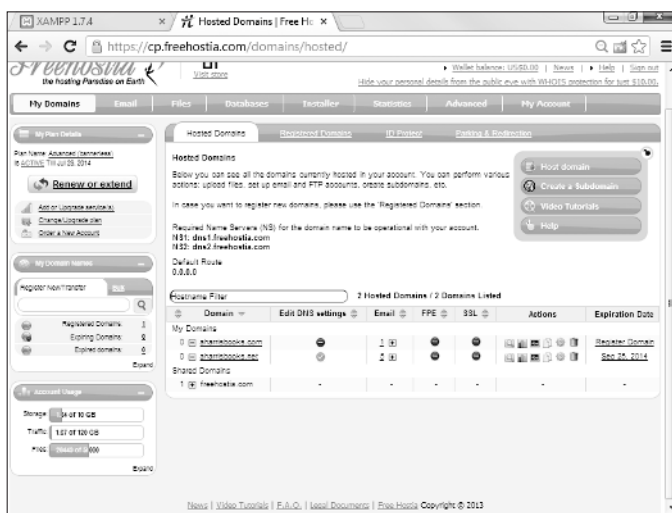
In this section, I show you how to register a domain using Freehostia.com. Check the documentation on your hosting service. Chances are that the main technique is similar, even if the details are different.

To add a domain name to your site, follow these steps:

1. Log in to the service.

Log in to your hosting service administration panel. You usually see a Control Panel something like the one shown in Figure 1-13.

Figure 1-13: This Control Panel shows all the options, including domain and subdomain tools.



2. Find the domain manager.

In Freehostia, the domain manager is part of the regular administration panel.

3. Pick a subdomain.

In a free hosting service, the main domain (`freehostia.com`, for example) is often chosen for you. Sometimes, you can set a subdomain (like `mystuff.freehostia.com`) for free. The page for managing this process might look like Figure 1-14.

Figure 1-14:

Use this page to create a subdomain for your account.



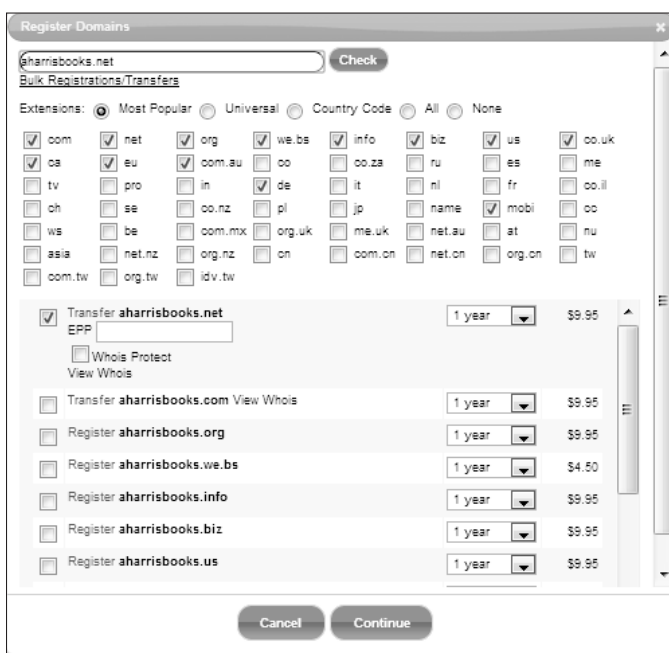
4. Look for a domain search tool.

Often, you have a tool, like the one shown in Figure 1-15, that allows you to search for a domain.

5. Search for the domain name you want.

You can type a domain name to see whether it's available.

Figure 1-15:
I'm searching for `aharrisbooks.net` — it seems like a good name!



6. If the domain name is available to register and you want to own it, purchase it immediately.

If a domain is available to transfer, it means that somebody else probably owns it.



Don't search for domains until you're ready to buy them. Unscrupulous people on the web look for domains that have been searched and then buy them immediately, hoping to sell them back to you at a higher price. If you search for a domain name and then go back the next day to buy it, you often find that it's no longer available and must be transferred. I've also seen people offer to sell you a domain that's currently available, then buy it up only after you've agreed to purchase from them and sell it at a huge markup.

7. Register the domain.

The domain-purchase process involves registering yourself as the domain owner. WHOIS information provides your information to people inquiring about the domain name.

8. Wait a day or two.

Your new domain name won't be available immediately. It takes a couple of days for the name to be registered everywhere.

9. Remember to renew your domain registration.

Domain-name registration isn't expensive (typically about \$10 per year), but you must renew it or risk losing the name.

Managing Data Remotely

Websites often work with databases. Your hosting service may have features for working with MySQL databases remotely. You should understand how this process works because it's often slightly different from working with the database on your local machine.

Creating your database

Often, a tool like the one shown in Figure 1-16 allows you to pick a defined database or create a new one.

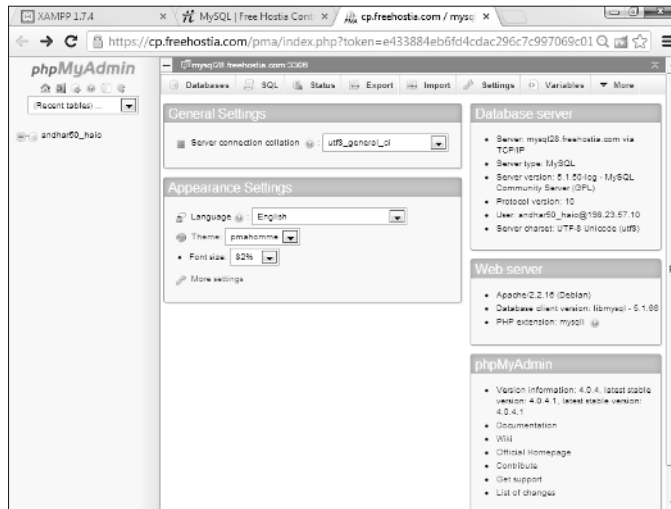
Figure 1-16: You often have to create a database outside of phpMyAdmin.



This database creation step happens because you don't have `root` access to MySQL. (If everybody had `root` access, chaos would ensue.) Instead, you usually have an assigned username and database name enforced by the server. On Freehostia, all database names begin with the username and an underscore. To create a new database, you need to provide a database name and a password. Usually, a MySQL user is created with the same name as the database name.

After you create the database, you can select it to work with the data in MySQL. Figure 1-17 shows the MySQL screen for my database on Freehostia.

Figure 1-17:
phpMyAdmin
is just like
the one on
your home
machine!



You can see from Figure 1-17 that phpMyAdmin is somewhat familiar if you read Book VI. Often, public servers remove the Privileges section because you aren't logged in as `root`. Everything else is basically the same. See Book VI for details on how to use phpMyAdmin to work with your databases.

Finding the MySQL server name

Throughout Book VI, I assume that the MySQL server is on the same physical machine as the web server. This situation is common in XAMPP installations, but commercial servers often have separate servers for data. You may have to dig through the documentation or find a Server Statistics section to discover how your PHP programs should refer to your server.

By far the biggest problem when moving your programs to a remote server is figuring out the new connection. Make sure that you know the right combination of server name, username, and password. Test on a simple PHP application before working on a complex one.