Making Everything Easier!™

PHP, MySQL®, JavaScript® & HTML5

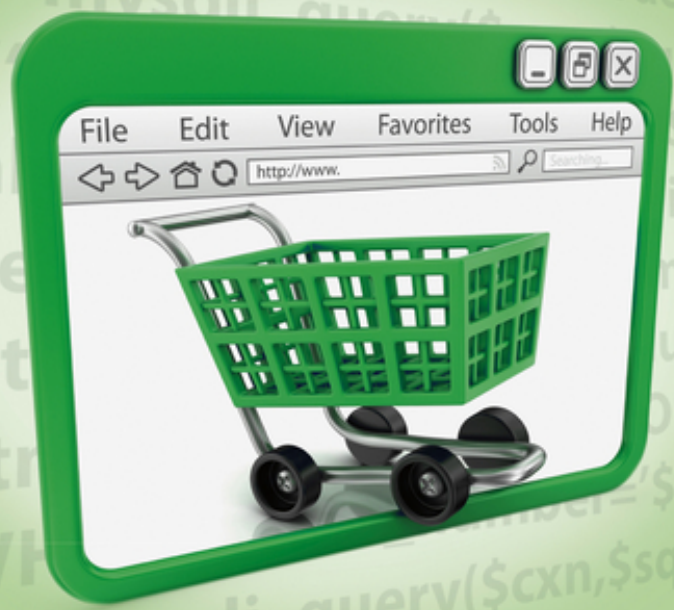
ALL-IN-ONE

FOR DUMMIES®

A Wiley Brand

7 BOOKS IN 1

- Web Technologies
- HTML and CSS
- JavaScript
- PHP
- MySQL
- Web Applications
- PHP and Templates



Steven Suehring
Janet Valade

Chapter 1: Understanding the Languages of the Web

In This Chapter

- **Understanding how the web works**
- **Discovering the language of web browsers**
- **Defining the language of web servers**
- **Choosing how you want to develop for the web**
- **Preparing your computer for web development**

As we explain programming for the web to you, it's helpful for all of us to speak the same language, at least when it comes to the subject at hand. Knowing how the web works, at least at a high level, will pay dividends when you start creating sites that will work on it. Granted, you don't need to know how a car works before driving, but knowing how the steering wheel, throttle, and brakes all relate to make the vehicle move is especially important to keep you from hitting things. So consider what you're about to read as driver's education for web programming. The difference is that at the end you don't have to buy insurance!

In this chapter, we define some basic web terminology, tell you about the languages you will use to create web pages, help you understand hosting options, and give you an idea of where to get started when you're setting up your computer.

Understanding How the Web Works

The World Wide Web consists of a large group of computers, known as *servers,* that exist solely to provide information when that information is requested. The information is requested by a piece of computer software called a *web browser.* If you're here, you've almost certainly used the web countless times already, maybe even to order this book.

It is said that the web operates on a *client-server model,* where the client is the web browser and the server is the computer providing, or serving, the information. That information is typically stored in a *web page,* which is nothing more than a specially formatted document that usually contains images and frequently references to other resources that help the page look and behave in a certain way.

The web browser

When a client requests a web page, a web browser such as Microsoft Internet Explorer or Mozilla Firefox (or Safari or Google Chrome or Opera or Lynx) is used. The web page itself can be a document stored on your computer, just like a word processing document. A program like Microsoft Word knows how to open documents formatted for Microsoft Word. In the same way, a web browser knows how to open documents formatted for the web. More on this later.

Web browsers are programmed to read and parse the specially formatted documents known as web pages.

The web browser knows not only how to open and parse documents formatted for the web, but also how to contact other computers to request documents from them. For example, when you type **http://www.braingia.org** into the address bar of your browser, the browser knows how to translate that request into the resulting page that you end up seeing in front of you.

The web server

When a web browser requests a page, it typically contacts a web server. Just as the web browser is software that's programmed to know how to read and parse web pages, the web server is software that's programmed to send web pages when they're requested.

Several popular web server software packages are available, but two stand out above the rest: Apache httpd and Microsoft Internet Information Services (IIS). Between the two of them, these server software packages are responsible for hosting the vast majority of all web domains.

Web servers and web browsers talk to each other using a protocol called HyperText Transfer Protocol, or HTTP. In essence, HTTP is just a way for these two parties to speak to each other.

Think of it as being like the protocol involved in making a telephone call. When you make a telephone call, you dial some digits. (This is like the web browser using the IP address to contact the web server.) The individual who answers the call is expected to say "Hello" or something similar. As a response, you're expected to say "Hello" or "What's Shakin'" or some other appropriate greeting so that you both know the conversation is underway.

This is all that HTTP or any other Internet protocol does: It defines how and when each party involved in the conversation should act. One major difference between HTTP and a telephone conversation is that HTTP is said to be stateless. This is a fancy way to say that HTTP doesn't remember what

it's doing from one request to the next. When you request a web page, the web server has no way of knowing that you just requested that same page 3 seconds ago and it won't know if you request the same page 3 seconds from now. This is important when you start programming web applications that need to remember things from one screen to the next — and you'll see how easy it is to solve the problem.

Lest you think you mistakenly bought *Internet For Dummies,* let's focus this discussion back toward web programming. Before doing so, here's a summary of where you are so far:

- A web browser is special software that knows how to open and interpret web pages. Web browsers also know how to contact web servers to get information.
- The web operates on a client-server model.
- A web server is special software that knows how to respond to requests for web pages.
- Web servers and web browsers speak HTTP to each other and do so using host names, domain names, and IP addresses.

Domain names and IP addresses

Every website needs a unique address on the web. The unique address used by computers to locate a website is the *Internet Protocol (IP)* address. The most commonly used version of the IP is version 4 (IPv4), but version 6 (IPv6) is becoming more popular. In version 4, an IP address is a series of four numbers between 0 and 255, separated by dots (for example, `172.17.204.2` or `192.168.2.33`).

Because IP addresses are made up of numbers and dots, they aren't easy to remember. Fortunately, there's a translation service called the Domain Name System (DNS) that provides translation services between IP addresses and friendly host names that are easier to remember.

On the web, you typically see "www" followed by a dot followed by a domain name, as in `www.braingia.org`. In that address, the www is called a *subdomain* and the braingia.org part is called the *domain name.* Technically, the .org part is called a Top-Level Domain or TLD.

When you browse to a site such as `www.braingia.org`, a DNS server which is known to your computer asks "What's the IP address of `www.braingia.org`?" The DNS server then looks up the address for `www.braingia.org` and sends it back to your computer so that you can contact the server responsible for `www.braingia.org`.

Each domain name must be unique. Consequently, a system of registering domain names ensures that no two locations use the same domain name. For the most part (and barring legalities), anyone can register any domain name as long as the name isn't already taken.

Understanding Web Page Languages

So far you've seen that the web is made up of web servers and web browsers. Web servers are the computers that host the web pages, videos, images, and other content that you view on the web. The browsers are what you use to view that content. Browsers like Internet Explorer and Safari run on your computer.

Mobile phones use browsers too. The iPhone uses a version of Safari while Android-based phones use a proprietary browser or sometimes another browser like a mobile version of Google Chrome or Firefox.

Web browsers and servers talk to each other using a language, or protocol, known as HTTP. Just as browsers and servers talk to each other using their own special language, web pages themselves have their own special languages. This section looks at the three primary web page languages: HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript.

Marking up with HTML

Web pages are documents, much like the document that you'd create in a word processor like Microsoft Word. To read a word processor document you use software like Microsoft Word, which knows how to open, read, and parse documents formatted or laid out in a certain way so that the various headings, spacing, and other elements of that document appear as intended.

Here's an example: We're writing this book in Microsoft Word. Each of the headings has a certain format while the main text has a different format. A new paragraph is created every time one of your humble book authors presses Enter. Microsoft Word knows how to open this document and interpret those headings, paragraphs, and other elements, so if we send it to you and you also have Microsoft Word, you can open and see the document in the same way that we do. Behind the scenes, hidden formatting elements tell Microsoft Word how to format or layout and display the text you see on the page.

HyperText Markup Language (HTML) provides the behind-the-scenes formatting and layout information for web pages. In much the same way as the behind-the-scenes formatting of a Word document tells Microsoft Word how to display that document, HTML tells the web browser how to display a web page.

HTML *marks up,* or adds hidden information to, the text and other things that you put on a web page. This hidden information is responsible for the layout of the page. For example, you can use HTML to indicate that specific text is a paragraph or a heading, and yet more HTML to indicate an image.

Just as there are rules for formatting a book such as this (for example, any level 2 headings appear below the primary, level 1 headings), so too are web pages formatted in a special way. Ideally, web pages follow certain rules such as smaller headings appearing within larger ones, and so on.

When HTML on a web page is formatted correctly, with headings and other elements appearing in the proper order, the web page is said to be valid and have what's called *semantic markup.* Semantic markup is a term used to describe a web page that correctly uses the HTML formatting elements in the right places. There's much more on this in Book II, Chapter 1.

Later in the book, you discover how to make the web browser understand formatting to create headings, paragraphs, insert images, and more, all with HTML.

Styling pages with CSS

HTML informs the browser how text and other pieces of content on a page are laid out. Cascading Style Sheets (CSS), on the other hand, is used to change that layout to add stylistic or appearance-related information to the page. CSS is frequently used to change colors, fonts, text size, and other appearance-related items.

For example, when you create a paragraph of text with HTML it's up to the browser to choose the font. By adding CSS font information, you can tell the browser which font, or more appropriately, a family of fonts, to choose from in order to display the text. Ultimately it's still up to the browser to choose which font to use or even to ignore your CSS completely and display its own choice.

CSS is also used to change the overall appearance of the page itself. For example, CSS can be used to create multi-column layouts, headings on pages, footers, and other display-oriented elements to make the page visually appealing and more usable.

Book II, Chapter 2, covers more about CSS, including its rules and usage.

Changing behaviors with JavaScript

HTML is used to provide layout information and CSS is used to change the appearance of that layout. What does JavaScript do? JavaScript provides the behavior or actions behind the interactivity that you see on web pages. For example, when you click a button on a web page, chances are there's a JavaScript program running behind the scenes in order to make the button do something like change a color or move text around on a page.

If you've ever used a site like Google's Mail (Gmail) then you've seen a site with heavy JavaScript integration. One misconception about JavaScript is that it's somehow related to Java: It isn't. Java and JavaScript are two completely separate languages.

Don't confuse JavaScript with Java; they're completely different languages that do completely different things.

Book III examines JavaScript in great detail.

Understanding the Language of Web Servers

So far in this chapter, you've read about web page languages HTML, CSS, and JavaScript. These languages deal with the look and feel (HTML and CSS) and behavior (JavaScript) of the web page. Many web pages are merely saved documents that exist on a web server, but some are dynamically built, with real-time information retrieved as you request it.

When pages are built dynamically, on-the-fly, a program is running on the web server to build that page. These programs are called *server-side programs.* Just as JavaScript programs tell the browser how to behave, server-side programs tell the web page what elements and layout it will have; in other words, the HTML, CSS, and JavaScript are all added by the server's program.

The program that runs on the server is written in yet another language, aside from the HTML, CSS, and JavaScript that you've already seen. Server-side programs for the web can be written in one (or more) of a number of languages. These include Microsoft's .Net family of languages, Perl, Python, Java, and the one that this book concentrates on: PHP.

Of course, in order for the page to be seen by the user it needs to be sent there. Sending the page to the user is the web server, which in our case will be Apache. And many sites utilize databases to store information. That's where MySQL comes in. As you'll see, MySQL provides a great (and free) way to store data for your website.

Building dynamic web applications with PHP and MySQL

PHP, short for PHP HyperText Preprocessor, is a popular and powerful language used for programming server-side programs. When PHP builds web pages it frequently needs to retrieve data to display on the resulting page. This is where MySQL comes in. MySQL is a popular and free database system that can store information and then integrate with PHP to create a fully functional web application.

PHP and MySQL are a popular pair for building dynamic web applications. PHP is a scripting language designed specifically for use on the web, with features that make web design and programming easier. MySQL is a fast, easy-to-use RDBMS (Relational Database Management System) used on many websites. MySQL and PHP as a pair have several advantages:

- **They're free.** It's hard to beat free for cost-effectiveness.
- **They're web oriented.** Both were designed specifically for use on websites. Both have a set of features focused on building dynamic websites.
- **They're easy to use.** Both were designed to get a website up quickly.
- **They're fast.** Both were designed with speed as a major goal. Together they provide one of the fastest ways to deliver dynamic web pages to users.
- **They communicate well with one another.** PHP has built-in features for communicating with MySQL. You don't need to know the technical details; just leave it to PHP.
- **A wide base of support is available for both.** Both have large user bases. Because they're often used as a pair, they often have the same user base. Many people are available to help, including people on e-mail discussion lists who have experience using MySQL and PHP together.
- **They're customizable.** Both are open source, thus allowing programmers to modify the PHP and MySQL software to fit their own specific environments.

Sending the page to the browser with Apache

PHP and MySQL don't operate all alone; they need a web server in order to actually respond to requests for web pages. A web server is special software that runs on a computer. The most widely used web server on the Internet is httpd from Apache, but most people just refer to it as "Apache" and so we do the same here. Like PHP and MySQL, Apache is free.

When a person uses his or her browser to request a page, that request is received by the web server, Apache. Apache then looks to see if it knows about the resource (the web page) being requested. If Apache knows about the web page and is able to send it, then Apache responds to the request by sending the page to the requestor.

In the case of pages created with PHP, Apache uses special software to interpret the PHP prior to sending the page back to the requestor.

Apache offers the following advantages:

- **It's free.** What else do we need to say?
- **It runs on a variety of operating systems.** Apache runs on Windows, Linux, Mac OS, FreeBSD, and most varieties of Unix.

- **It's popular.** Approximately 60 percent of websites on the Internet use Apache, according to surveys at `http://news.netcraft.com/archives/web_server_survey.html` and `www.securityspace.com/s_survey/data`. This wouldn't be true if it didn't work well. Also, this means that a large group of users can provide help.
- **It's reliable.** When Apache is up and running, it should run as long as your computer runs. Emergency problems with Apache are rare.
- **It's customizable.** The open source license allows programmers to modify the Apache software, adding or modifying modules as needed to fit their own environment.
- **It's secure.** You can find free software that runs with Apache to make it into an SSL (Secure Sockets Layer) server. Security is an essential issue if you're using the site for e-commerce.

Choosing How You Want to Develop

When developing applications for the web, specifically applications that encompass both the browser-related technologies (HTML, CSS, and JavaScript) and the server technologies (PHP and MySQL), you have several choices for development and ultimately for placing the site up so that others can get to it.

For development of the HTML, CSS, and JavaScript, you use your own computer or a computer provided to you for this purpose. We cover this aspect in short order. For now, think about the type of web development you intend to do as you read these next sections.

Choosing a host for your website

You can set up a computer in your office or basement to be the web server (sometimes called the web host) for your website. You need to be pretty technically savvy to do this. The Internet connection you use to access the World Wide Web is unlikely to provide sufficient resources to allow users to access your computer. You probably need a faster connection that provides domain name system (DNS) service. You need a different type of Internet connection, probably at an increase in cost. This book doesn't provide the information you need to run your own web host.

If you already have the technical know-how to set up a host machine, you can probably install the web software from information in this book. However, if you don't understand Internet connections and DNS sufficiently to connect to the Internet, you need to research this information elsewhere, such as a system administration book or a networking book for your operating system.

Most people don't host their websites on their own computers. Most people upload their websites to a web host provided by someone else. However, it's quite common to run a web server on your computer for your own use during development. Doing so has the advantage of isolating the development from the production, or publicly viewable website. In other words, if you make changes to files on your computer, you can thoroughly test those changes before making them publicly available just in case there are problems with the files.

When you're ready to make the site available publicly, you then enlist the help of someone to host the site. Web hosting is often provided by one of the following:

- **The website owner:** Perhaps you're creating a website for a company, either as an employee or a contractor. The company — usually the company's IT (Information Technology) department — installs and administers the website software.
- **A web-hosting company:** You can park your website on a web-hosting company's computer. The web-hosting company installs and maintains the website software and provides space on its computer, usually for a fee, where you can upload the web page files for your website.

In the coming sections, we describe these environments in more detail and how to install your website in the environments. We also explain how you gain access to PHP and MySQL.

Hosting for a company website

When a website is run by a company, you don't need to understand the installation and administration of the website software at all. The company or website owner is responsible for the operation of the website. In most cases, the website already exists, and your job is to add to, modify, or redesign the existing website. In a few cases, the company might be installing its first website, and your job is to design the website. In either case, your responsibility is to write and install the web page files for the website. You aren't responsible for the operation of the website.

You access the website software through the company's IT department. The name of this department can vary in different companies, but its function is the same: It keeps the company's computers running and up to date.

If PHP or MySQL or both aren't available on the company's website, IT needs to install them and make them available to you. PHP and MySQL have many options, but IT might not understand the best options — and might have options set in ways that aren't well suited for your purposes. If you need PHP or MySQL options changed, you need to request that IT make the

change; you won't be able to make the change yourself. For instance, PHP must be installed with MySQL support enabled, so if PHP isn't communicating correctly with MySQL, IT might have to reinstall PHP with MySQL support enabled.

You'll interact with the IT folks frequently as needs arise. For example, you might need options changed, you might need information to help you interpret an error message, or you might need to report a problem with the website software. So a good relationship with the IT folks will make your life much easier. Bring them tasty cookies and doughnuts often.

Choosing a web-hosting company

A *web-hosting company* provides everything that you need to put up a website, including the computer space and all the website software. You just create the files for your web pages and move them to a location specified by the web-hosting company.

About a gazillion companies offer web-hosting services. Most charge a monthly fee (often quite small), and some are even free. (Most, but not all, of the free ones require you to display advertising.) Usually, the monthly fee varies depending on the resources provided for your website. For instance, a website with 100MB of disk space for your web page files costs less than a website with 200MB of disk space.

When looking for a web-hosting company for your website, make sure that it offers the following:

- **PHP and MySQL:** Not all companies provide these tools. You might have to pay more for a site with access to PHP and MySQL; sometimes you have to pay an additional fee for MySQL databases.
- **A recent version of PHP:** Sometimes the PHP versions offered aren't the most recent versions. Take the time to find a web-hosting company that offers at least PHP 5.3, if not PHP 6 if it is available. Some web-hosting companies offer PHP 4 but have PHP 5 (or 6) available for customers who request it.

Other considerations when choosing a web-hosting company are

- **Reliability:** You need a web-hosting company that you can depend on — one that won't go broke and disappear tomorrow and that isn't running on old computers that are held together by chewing gum and baling wire. If the company has more downtime than uptime, save yourself a headache and look elsewhere. Take a look at Web Hosting Talk at `www.webhostingtalk.com` or Netcraft at `www.netcraft.net` for information on reliable providers.

- **Speed:** Web pages that download slowly are a problem because users will get impatient and go elsewhere. Slow pages might be a result of a web-hosting company that started its business on a shoestring and has a shortage of good equipment, or the company might be so successful that its equipment is overwhelmed by new customers. Either way, web-hosting companies that deliver web pages too slowly are unacceptable. Netcraft (`www.netcraft.net`) regularly posts a survey of the fastest hosting providers.
- **Technical support:** Some web-hosting companies have no one available to answer questions or troubleshoot problems. Technical support is often provided only through e-mail, which can be acceptable if the response time is short. Sometimes you can test the quality of the company's support by calling the tech support number, or you can test the e-mail response time by sending an e-mail.
- **Backups:** *Backups* are copies of your web page files and your database that are stored in case your database or files are lost or damaged. You want to be sure that the company makes regular, frequent backup copies of your application. You also want to know how long it would take for backups to be put in place to restore your website to working order after a problem.

  Additionally, you should always make sure to take regular backups of your own data. It's your data; you should be responsible for it. That way, if the web-hosting provider goes away unexpectedly you can take your latest backup and move to a new hosting provider.

- **Features:** Select features based on the purpose of your website. Usually a hosting company bundles features together into plans — more features equal a higher cost. Some features to consider are
  - *Disk space:* How many MB or GB of disk space will your website require? Media files, such as graphics or music files, can be quite large.
  - *Data transfer:* Some hosting companies charge you for sending web pages to users. If you expect to have a lot of traffic on your website, this cost should be a consideration.
  - *E-mail addresses:* Many hosting companies provide a number of e-mail addresses for your website. For instance, if your website is `example.com`, you could allow users to send you e-mail at `me@example.com`.
  - *Software:* Hosting companies offer access to a variety of software for web development. PHP and MySQL are the software that we discuss in this book. Some hosting companies might offer other databases, and some might offer other development tools such as FrontPage extensions, shopping cart software, and credit card validation.
  - *Statistics:* Often you can get statistics regarding your web traffic, such as the number of users, time of access, access by web page, and so on.

With most web-hosting companies, you have no control over your web environment. The web-hosting company provides the environment that works best for it — probably setting up the environment for ease of maintenance, low cost, and minimal customer defections. Most of your environment is set by the company, and you can't change it. You can only beg the company to change it. The company will be reluctant to change a working setup, fearing that a change could cause problems for the company's system or for other customers.

It's pretty difficult to research web-hosting companies from a standing start — a Google.com search for *"web hosting"* results in almost 400 million hits. The best way to research web-hosting companies is to ask for recommendations from people who have experience with those companies. People who have used a hosting company can warn you if the service is slow or the computers are down often. After you gather a few names of web-hosting companies from satisfied customers, you can narrow the list to find the one that's best suited to your purposes and the most cost effective.

Using a hosted website

When you use an environment with a hosted website, such as the environments discussed in this section, for the world to see the web pages, the web page files must be in a specific location on the computer. The web server that delivers the web pages to the world expects to find the web page files in a specific directory. The web host staff or IT department should provide you with access to the directory where the web page files need to be installed. To use the web software tools and build your dynamic website, you need the following information from the web host:

- **The location of web pages:** You need to know where to put the files for the web pages. The web-host staff needs to provide you with the name and location of the directory where the files should be installed. Also, you need to know how to install the files — copy them, FTP (file transfer protocol) them, or use other methods. You might need a user ID and password to install the files. This information will almost certainly be included in a welcome e-mail with the company and available as a Frequently Asked Question (FAQ) page on its website.
- **The default filename:** When users point their browsers at a URL, a file is sent to them. The web server is set up to send a file with a specific name when the URL points to a directory. The file that is automatically sent is the *default file.* Very often the default file is named `index.htm` or `index.html`, but sometimes other names are used, such as `default.htm`. You need to know what you should name your default file.
- **A MySQL account:** Access to MySQL databases is controlled through a system of account names and passwords. The organization providing the web host sets up a MySQL account for you that has the appropriate permissions and also gives you the MySQL account name and password.

- **The location of the MySQL databases:** MySQL databases need not be located on the same computer as the website. If the MySQL databases are located on a computer other than that of the website, you need to know the *hostname* (for example, `thor.example.com`) where the databases can be found.
- **The PHP file extension:** When PHP is installed, the web server is instructed to expect PHP statements in files with specific extensions. Frequently, the extensions used are `.php` or `.phtml`, but other extensions can be used. PHP statements in files that don't have the correct extension won't be processed. Find out what extension to use for your PHP programs.

Setting Up Your Local Computer for Development

To use your local computer to develop your website, you must install a web server, PHP, and MySQL. PHP and MySQL are free to download and use; the web server Apache is free as well, although you might opt to pay for a different web server that might better fit your needs.

In the following sections, we give you some basic information about approaching these installations, and then in the following chapters we describe in more detail how to complete these tasks.

Installing the web server

Assuming that you have a computer with an operating system (such as Windows, Mac OS X, or Linux) already installed, you next need to install a web server. Your first step is deciding which web server to install. The answer is almost always Apache. Here are some things to consider, depending on which operating system you're using:

- **Windows:** Apache provides an installer for Windows that installs and configures Apache for you.
- **Linux:** Apache is sometimes automatically installed when you install certain Linux distributions.
- **Mac:** All recent Macs come with Apache installed. However, you might need to install a newer version of Apache.

The Apache website (`http://httpd.apache.org`) provides information, software downloads, extensive documentation that is improving all the time, and installation instructions for various operating systems.

Other web servers are available; however, we focus almost exclusively on Apache in this book. Microsoft offers *IIS* (Internet Information Server), which is the second most popular web server on the Internet and nginx is also

available and a popular option as well. Other web servers are available, but they have even smaller user bases.

Installing PHP

You might or might not need to install PHP.

- **Windows:** PHP isn't installed on Windows computers.
- **Linux or Mac:** PHP is often already installed in Linux or the Mac OS. Sometimes it's installed but not activated.

After installing PHP, you need to configure your web server to process PHP code. Instructions for installing PHP and configuring your web server are provided in this minibook.

Installing MySQL

You might or might not need to install MySQL. Consider which operating system you're using and the following information:

- **Windows:** MySQL isn't provided with the Windows operating system.
- **Linux or Mac:** Along with PHP, MySQL is often already installed on Linux or Mac. Sometimes it is installed, but not activated. However, the installed version might be an older version, in which case you should install a newer version.

As you might suspect, installation varies depending on which operating system you're using. You install and configure MySQL on Windows by using a Setup and a Configuration Wizard. A PKG file is available for installing MySQL on Mac OS X, and packages are available with every popular Linux distribution.

Chapter 2: Installing a Web Server

In This Chapter

- **Testing for a web server**
- **Obtaining Apache**
- **Installing Apache on Windows, Linux, and Mac**
- **Using Apache**
- **Configuring Apache**

You might have the idea that this chapter is all about the web server Apache. Well, you're right. In this chapter, you download and install Apache. If you'll be using a hosted website or a company website and placing your files on someone else's server, then you don't need to install a web server at all.

The chapter focuses on httpd from Apache because it's free and the most popular web server used on the Internet. Other web servers are available. Microsoft has Internet Information Services (IIS) and also includes a development web server with its Visual Studio development application. Another popular web server is called nginx. Apache and its wide support across different types of computers is so popular that we focus solely on Apache in this book.

Windows doesn't come with Apache installed. You must install it yourself. Most Linux distributions include Apache or have it easily available through their package management software. All recent versions of Mac OS X come with Apache already installed. However, you might want to install Apache yourself for a newer version or to install with different options.

This chapter guides you in finding out if Apache is already installed on your computer; finding, downloading, and installing the software; starting and stopping Apache; getting information about the installation; and configuring Apache so that it behaves as you need it to.

Testing Your Web Server

You can test whether a web server is installed on your computer by viewing a web page in your browser. Open your browser and type **http://localhost** in the browser address bar. If your web server is installed and running, a web page displays. For instance, the Apache Welcome screen displays the following text:

```
If you can see this, it means that the installation of the
Apache web server software on this system was successful. You
may now add content to this directory and replace this page.
```

You can't test your web server by choosing File➪Open or Open File in your browser. This method of viewing a web page file doesn't go through the web server. You must type the URL into your browser's address bar to test the server.

If no web server is running on your machine, an error message is displayed, such as one of the following:

```
Unable to connect
```

```
The page cannot be displayed
```

Even if you have no web server running, a web server might be installed on your computer but not started. If so, you need only start the web server. For instance, Apache is installed on all recent Mac computers, but it might need to be started. See the instructions for obtaining and installing Apache later in this chapter.

Obtaining Apache

Apache is an open source web server that you can download for free. The sections that follow give you the preliminary info you need — based on the operating system you're using — to decide how to begin selection and installation of your web server software. Be aware, also, that an all-in-one installation kit might work for your purposes. We provide information on that option as well in this section.

Selecting a version of Apache

Apache is currently available in three versions: Apache 2.0, Apache 2.2, and Apache 2.4. All versions are supported and upgraded. The PHP software runs with all three versions, but some other software related to PHP might have problems with Apache 2.4. On Windows, Apache 2.4 is currently not available.

Like any software, Apache evolves as new versions come out. Some third-party modules might not work correctly on all three versions. Because PHP is a module, you should check the web page for the current status of PHP with Apache versions at

```
www.php.net/manual/en/install.windows.apache2.php
```

Try to install the most current release of the Apache version you choose so that your Apache server includes all the latest security and bug fixes.

Downloading from the Apache website

Apache for all operating systems is available on the official Apache website. You can download source code to compile on your operating system. Compiling and installing source code isn't difficult on Linux and Mac, but it requires expert knowledge and software on Windows.

Binary files — compiled, ready-to-run files that just need to be copied to the correct location — are available for Windows.

To obtain Apache from the Apache website, go to `http://httpd.apache.org`. Scroll down to the section for the Apache version you want to download click the appropriate link for the version you want to download. A download page with links to download the current versions displays.

Obtaining Apache for Windows

The Windows binary file is available with an installer, which will install, configure, and start Apache. On the Apache website download page, find the section for the Apache version you want. Click the link for the Win32 Binary (MSI Installer) to download the installer file.

Although Win32 source code is also available to download in a Zip file, compiling and installing Apache from source code is difficult and should be attempted only by advanced users. It requires advanced knowledge and special software.

Obtaining Apache for Linux

Most recent versions of Linux include Apache. If you need to install Apache or upgrade to a more recent version, most Linux distributions provide software either on their website that you can download or through their package management software that will install on your specific Linux system. In addition, most Linux systems provide a utility specifically for downloading and installing software. For instance, Fedora provides the `yum` utility that downloads and installs software from the Fedora website. See the documentation for your Linux distribution for information on how to download and install software on your Linux system.

In a few cases, you might need to install Apache manually. The software provided by the website might not be the most recent or might not be configured to your needs. To install manually, you need to download the source code from the Apache website at `http://httpd.apache.org`.

You can easily compile and install Apache from the source code. This process isn't as technical and daunting as it sounds. Instructions for installing Apache from source code are provided in the "Installing Apache from source code on Linux or Mac" section, later in this chapter.

Obtaining Apache for Mac

Apache comes already installed on most recent versions of Mac OS X. If you test Apache by typing **http://localhost** in your browser address window and it doesn't display a web page, it's probably installed but not started. To find out how to start Apache, see the section "Installing Apache on a Mac," later in this chapter.

If you need to install Apache because it isn't installed or an old version is installed, download the source files from the Apache website to compile and install on your Mac. Instructions for installing Apache from the source code are provided in the "Installing Apache from source code on Linux and Mac" section, later in this chapter.

Obtaining all-in-one installation kits

You can obtain some kits that contain and install PHP, MySQL, and Apache in one procedure. These kits can greatly simplify the installation process. However, the software provided might not include the features and extensions that you need.

XAMPP is a popular all-in-one installation kit that contains Apache, PHP, and MySQL. XAMPP has stable versions available for Windows and for several versions of Linux. XAMPP is available at `www.apachefriends.org/en/xampp.html`. Instructions for installing your software using XAMPP are provided in Chapter 5 in this minibook.

WAMPServer is a popular installation kit for Windows that provides recent versions of Apache, PHP, and MySQL. It also installs phpMyAdmin, a utility for managing your MySQL databases. WAMPServer is available at `www.wampserver.com/en`.

MAMP is an installation kit for Mac that installs Apache, PHP, and MySQL for Mac OS X. This free package installs a local server environment on your Mac. You can obtain MAMP at `www.mamp.info`.

Verifying a downloaded file

The Apache website provides methods to verify the software after you download it, as a security precaution to make sure that the file hasn't been altered by bad guys. You can use the MD5 method or the PGP method for verifying the file. This book provides instructions for the MD5 method.

Basically, the same process is used to verify the file for PHP, MySQL, and Apache. You can find instructions for verifying the downloaded file in Chapter 3 of this minibook. On the Apache website, click the MD5 link to see the MD5 signature discussed in the instructions.

Installing Apache

The following subsections describe installing Apache on Windows, Mac, and Linux.

Installing Apache on Windows

You can install Apache on almost any version of Windows.

You can't install Apache with the following directions if Internet Information Services (IIS) is already running on port 80. If IIS is running, you will find the IIS console at Start➪Control Panel➪Administrative Tools➪Internet Services Manager. If you don't find this menu item, IIS isn't installed. If IIS is already running, you must shut it down before installing Apache or install Apache on a different port.

To install Apache after you're sure that IIS isn't running, follow these steps:

1. **Double-click the file you downloaded.**

   The file is named `apache_`, followed by the version number and `win32-x86-no_ssl.msi`. For instance, `httpd-2.2.22-win32-x86-no_ssl.msi`.

   Note: You might need to right-click the file and choose Run as Administrator.

   The Apache Installation Wizard begins, and a welcome screen appears.

2. **Click Next.**

   The license agreement is displayed.

3. **Select I Accept the Terms in the License Agreement and then click Next.**

   If you don't accept the terms, you can't install the software. A screen of information about Apache is displayed.

4. **Click Next.**

   A screen is displayed asking for information.

5. **Enter the requested information and then click Next.**

   The information requested is

- *Domain Name:* Type your domain name, such as **example.com**. If you're installing Apache for testing and plan to access it only from the machine where it's installed, you can enter **localhost**.
- *Server Name:* Type the name of the server where you're installing Apache, such as **www.example.com** or **s1.example.com**. If you're installing Apache for testing and plan to access it only from the machine where it's installed, you can enter **localhost**.
- *E-Mail Address:* Type the e-mail address where you want to receive e-mail messages about the web server, such as **webserver@example.com**.
- *Run Mode:* Select whether you want Apache to run as a service (starting automatically when the computer boots up) or whether you want to start Apache manually when you want to use it. In most cases, you want to run Apache as a service.

The Installation Type screen is displayed.

6. **Select an installation type and then click Next.**

   In most cases, you should select Complete. Only advanced users who understand Apache well should select Custom. If you select Custom, the screens will be somewhat different than the screens described in the following text. A screen showing where Apache will be installed is displayed.

7. **Select the directory where you want Apache installed and then click Next.**

   You see the default installation directory for Apache, usually `C:\Program Files\Apache Group`. If this is okay, click Next. If you want Apache installed in a different directory, click Change and select a different directory, click OK, and click Next. The screen that appears says the wizard is ready to install Apache.

8. **Click Install.**

   If you need to, you can go back and change any of the information you entered before proceeding with the installation. A screen displays the progress. When the installation is complete, a screen appears, saying that the wizard has successfully completed the installation.

9. **Click Finish to exit the Installation Wizard.**

Apache is installed on your computer based on your operating system. If you install it on later versions of Microsoft Windows, it is installed by default as a service that automatically starts when your computer starts. If you install it on an older version of Windows, such as Windows 95/98/Me, then you need to start it manually or set it up so that it starts automatically when your computer boots. See the section "Starting and Stopping Apache," later in this chapter, for more information.

Installing Apache on a Mac

Apache is installed on all recent versions of Mac OS X, but it might not be started. To start Apache, choose Apple Menu➪System Preferences➪Sharing. On the Service pane, find the section for web sharing. Click the check box to turn web sharing on, which starts the Apache web server.

If you need to install Apache yourself for some reason, you can install Apache from source code, as described in the next section.

Installing Apache from source code on Linux and Mac

You can install Apache on Linux, Unix, and Mac from source code. You download the source code and compile it. To install Apache from source code, follow these steps:

1. **Change to the directory where you downloaded the file.**

   The downloaded file is typically named `httpd-`, followed by the version name and `tar.gz`. This file is called a *tarball* because it contains many files compressed by a program called `tar`.

2. **Unpack the tarball by using a command similar to the following:**

   For Linux:

   ```
   gunzip -c httpd-2.2.22.tar.gz | tar -xf -
   ```

   For Mac:

   ```
   gnutar -xzf httpd-2.2.22.tar.gz
   ```

   After unpacking the tarball, you see a directory called `httpd_2.2.22`. This directory contains several subdirectories and many files. Note that the version number will be different by the time you read this.

3. **Use a `cd` command to change to the new directory created when you unpacked the tarball (for example, `cd httpd_2.2.22`).**

4. **Type the `configure` command.**

   The `configure` command consists of `./configure` followed by all the necessary options. To use Apache with PHP as a module, use the appropriate `configure` command as follows:

   For Linux or Unix, use

   ```
   ./configure --enable-so
   ```

   For Mac, use

   ```
   ./configure --enable-module=most
   ```

You can use other options if you want. One of the more important installation options you might want to use is `prefix`, which sets a different location where you want Apache to be installed. By default, Apache is installed at `/usr/local/apache` or `usr/local/apache2`. You can change the installation location with the following line:

```
./configure --prefix=/software/apache
```

You can see a list of all available options by typing the following line:

```
./configure --help
```

This script might take a while to finish running. As it runs, it displays output. When the script is finished, the system prompt is displayed. If `configure` encounters a problem, it displays a descriptive error message.

5. **Type** make **to build the Apache server.**

   The `make` command might take a few minutes to run. It displays messages while it's running, with occasional pauses for a process to finish running.

6. **Type the following command to install Apache:**

   - *For Linux or Unix, type*

     ```
     make install
     ```

   - *For Mac, type*

     ```
     sudo make install
     ```

7. **Start the Apache web server.**

   See the next section for details.

8. **Type the URL for your website (for example,** www.example.com **or** localhost**) into a browser to test Apache.**

   If all goes well, you see a web page telling you that Apache is working.

Starting and Stopping Apache

You might need to start Apache when you install it. Or, you might not. It might already be started. However, whenever you change your Apache or PHP configuration settings, you need to restart Apache before the new settings go into effect.

Starting and stopping Apache on Windows

When you install Apache on Windows, it's usually automatically installed as a service and started. It's ready to use. However, on Windows 95, 98, and Me, you have to start Apache manually, using the menu.

When you install Apache, it creates menu items for stopping and starting it. To find this menu, choose Start➪Programs➪Apache HTTP Server➪Control Apache Server. The menu has the following items:

- **Start:** This option starts Apache when it isn't running. If you click this item when Apache is running, you see an error message saying that Apache has already been started.
- **Stop:** Stops Apache when it's running. If you click this item when Apache isn't running, you see an error message saying that Apache isn't running.
- **Restart:** This restarts Apache when it's running. If you make changes to Apache's configuration, you need to restart Apache before the changes become effective.

Starting Apache on Linux, Unix, and Mac

A script named `apachectl` is available to control the server. By default, the script is stored in a subdirectory called `bin` in the directory where Apache is installed. Some Linux distributions may put it in another directory.

The script requires a keyword. The most common keywords are `start`, `stop`, and `restart`. The general syntax is as follows:

```
path/apachectl keyword
```

The `apachectl` script starts the Apache server, which then runs in the background, listening for HTTP requests. By default, the compiled Apache server is named `httpd` and is stored in the same directory as the `apachectl` script, unless you changed the name or location during installation. The `apachectl` script serves as an interface to the compiled server, called `httpd`.

You can run the `httpd` server directly, but it's better to use `apachectl` as an interface. The `apachectl` script manages and checks data that `httpd` commands require. Use the `apachectl` script to start Apache with the following command:

- **For Linux:**

```
/usr/local/apache/bin/apachectl start
```

- **For Mac:**

```
sudo /usr/local/apache/bin/apachectl start
```

The `apachectl` script contains a line that runs `httpd`. By default, `apachectl` looks for `httpd` in the default location — `/usr/local/apache/bin` or `/usr/local/apache2/bin`. If you installed Apache in a nonstandard location, you might need to edit `apachectl` to use the correct path. Open `apachectl` and then search for the following line:

```
HTTPD='/usr/local/apache2/bin/httpd'
```

Change the path to the location where you installed `httpd`. For example, the new line might be this:

```
HTTPD='/usr/mystuff/bin/httpd'
```

After you start Apache, you can check whether Apache is running by looking at the processes on your computer. Type the following command to display a list of the processes that are running:

```
ps -A
```

If Apache is running, the list of processes includes some `httpd` processes.

Restarting Apache on Linux, Unix, and Mac

Whenever you change the configuration file, the new directives take effect the next time Apache starts. If Apache is shut down when you make the changes, you can start Apache as described earlier in the "Starting Apache on Linux, Unix, and Mac" section. However, if Apache is running, you can't use `start` to restart it. Using `start` results in an error message saying that Apache is already running. You can use the following command to restart Apache when it's currently running:

- **For Linux:**

  ```
  /usr/local/apache2/bin/apachectl restart
  ```

- **For Mac:**

  ```
  sudo /usr/local/apache2/bin/apachectl restart
  ```

Although the `restart` command usually works, sometimes it doesn't. If you restart Apache and the new settings don't seem to be in effect, try stopping Apache and starting it again. Sometimes this solves the problem.

Stopping Apache on Linux, Unix, and Mac

To stop Apache, use the following command:

```
/usr/local/apache/bin/apachectl stop
sudo /usr/local/apache/bin/apachectl stop
```

You can check to see whether Apache is stopped by checking the processes running on your computer by using the following command:

```
ps -A
```

The output from `ps` shouldn't include any `httpd` processes.

Getting Information from Apache

Sometimes you want to know information about your Apache installation, such as the installed version. You can get this information from Apache by using the applicable procedure that follows.

Getting Apache information on Windows

You can get information from Apache by opening a Command Prompt window (Start➪Programs➪Accessories➪Command Prompt), changing to the `bin` directory in the directory where Apache is installed (such as `cd C:\Program Files\Apache Group\Apache2\bin`), and accessing Apache with options. For example, to find out which version of Apache is installed, type the following in the command prompt window:

```
apache –v
```

To find out what modules are compiled into Apache, type

```
apache –l
```

You can also start and stop Apache directly, as follows:

```
apache -k start
apache -k stop
```

You can see all the options available by typing the following:

```
apache -h
```

Getting Apache information on Linux, Unix, and Mac

You can use options with the `httpd` server to obtain information about Apache. For instance, you can find out what version of Apache is installed by changing to the directory where the `httpd` server resides and typing one of the following:

```
httpd -v
./httpd –v
```

You can find out what modules are installed with Apache by typing

```
httpd -l
```

To see all the options that are available, type

```
httpd -h
```

Configuring Apache

When Apache starts, it reads information from a configuration file. If Apache can't read the configuration file, it can't start. Unless you tell Apache to use a different configuration file, it looks for the file `conf/httpd.conf` in the directory where Apache is installed. Keep reading for details on how to configure Apache so that it starts without a hitch.

Always restart Apache after you change any directives.

Changing settings

Apache behaves according to commands, called *directives,* in the configuration file (which is a plain text file). You can change some of Apache's behavior by editing the configuration file and restarting Apache so that it reads the new directives.

In most cases, the default settings in the configuration file allow Apache to start and run on your system. However, you might need to change the settings in some cases, such as the following:

- **Installing PHP:** If you install PHP, you need to configure Apache to recognize PHP programs. How to change the Apache configuration for PHP is described in Chapter 3 of this minibook.
- **Changing your Document Root:** Apache looks for web page files in a specific directory and its subdirectories, called your Document Root. You can change the location of your Document Root. Read the next section for instructions.
- **Changing the port on which Apache listens:** By default, Apache listens for file requests on port 80. You can configure Apache to listen on a different port. See the upcoming "Changing the port number" section for details on how to do that.

To change any settings, edit the `httpd.conf` file using a text editor. On Windows, you can access this file through the menu at Start➪Programs➪Apache HTTPD Server➪Configure Apache Server➪Edit the Apache httpd.conf File. When you click this menu item, the `httpd.conf` file opens in Notepad.

The `httpd.conf` file has comments (lines beginning with `#`) that describe the directives, but make sure you understand their functions before changing any. All directives are documented on the Apache website.

Here are some conventions to consider when you're changing Apache settings:

- **Filenames and paths:** When adding or changing filenames and paths, use forward slashes, even when the directory is on Windows. Apache can figure it out.
- **Path names:** You don't need to put path names in quotes, unless they include special characters.
- **Special characters:** A colon (`:`) is a special character; the underscore (`_`) and hyphen (`-`) are not.

For instance, to indicate a Windows directory, you would use something like the following:

```
"c:/temp/mydir"
```

The settings don't go into effect until Apache is restarted. Sometimes, using the `restart` command doesn't work to change the settings. If the new settings don't seem to be in effect, try stopping the server with `stop` and then starting it with `start`.

Changing the location of your Document Root

By default, Apache looks for your web page files in the subdirectory `htdocs` in the directory where Apache is installed. You can change this with the `DocumentRoot` directive. Look for the line that begins with `DocumentRoot`, such as the following:

```
DocumentRoot "C:/Program Files/Apache Group/Apache/htdocs"
```

Change the filename and path to the location where you want to store your web page files. Don't include a forward slash (`/`) on the end of the directory path. For example, the following might be your new directive:

```
DocumentRoot /usr/mysrver/Apache2/webpages
```

Changing the port number

By default, Apache listens on port 80. You might want to change this, for instance, if you're setting up a second Apache server for testing. The port is set by using the `Listen` directive as follows:

```
Listen 80
```

With Apache 2.0 and 2.2, the `Listen` directive is required. If no `Listen` directive is included, Apache 2 won't start.

You can change the port number as follows:

```
Listen 8080
```

Contents at a Glance

Table of Contents