Chapter 1

Getting to Know PHP

In This Chapter

- Taking a look at PHP
- Understanding how PHP works
- Understanding PHP as open source software

So, you want to get to know PHP. Perhaps this is your first adventure in programming, and you chose PHP because your techie friend told you it's easy to understand. Well, your friend is right. PHP is one of the easiest programming languages to understand. The developers of PHP strive constantly to keep it easy to use.

Perhaps you already know how to program in another language. You've decided to study PHP because it's the best language for your new Web application project. It's a good decision because PHP is well suited for writing dynamic Web applications. PHP is easy to get started with, but it also has many advanced features for seasoned programmers. If you know C, you have a great head start because PHP syntax is similar to C syntax.

In this chapter, I discuss what PHP is, what it can do, and how it does it.

Getting Familiar with PHP

PHP is a widely used open source, general-purpose *scripting language.* It was originally designed for use in Web site development. In fact, PHP started life as Personal Home Page tools, developed by Rasmus Lerdorf to assist users with Web page tasks. PHP proved so useful and popular, it rapidly grew to become the full-featured language that it is today, acquiring the name PHP Hypertext Preprocessor along the way to represent its expanded abilities — processing Web pages before they're displayed.

The popularity of PHP continues to grow rapidly because of its many advantages:

- **It's fast:** On Web sites, because it is embedded in HTML code, the time to process and load a Web page is short.
- **It's free:** PHP is proof that free lunches do exist and that you can get more than you paid for.
- **It's easy to use:** The syntax is simple and easy to understand and use, even for non-programmers. For use in Web sites, PHP code is designed to be included easily in an HTML file.
- **It's versatile:** PHP runs on a wide variety of operating systems — Windows, Linux, Mac OS, and most varieties of Unix.
- **Technical support is widely available:** You can join one of several e-mail discussion lists offered on the PHP Web site (`www.php.net`), which cover topics such as general PHP, PHP on Windows, or databases and PHP. In addition, a Web interface to the discussion lists is available at `news.php.net`, where you can browse or search the messages.
- **It's secure:** As long as your scripts are designed correctly, the user does not see the PHP code.
- **It's customizable:** The open source license allows programmers to modify the PHP software, adding or modifying features as needed to fit their own environments. PHP provides significant control over the environment, reducing chances of failure.

Considering the Various Uses for PHP

PHP is a general-purpose language that can be used to write general-purpose scripts. Scripts are computer files containing instructions in the PHP language that tell the computer to do things, such as display `Hello` on the screen or store some specified data in a database. Most scripts contain a series of instructions that can accomplish tasks from designing Web pages to navigating your file system. Because PHP began life on the Web, it has many features that are particularly well suited for use in scripts that create dynamic Web pages. Currently, you find PHP most often hard at work in Web pages, but its use for other purposes is growing.

PHP is very popular for Web sites. According to the PHP Web site (`www.php.net/usage.php`), over 11 million domains are using PHP. Yahoo!, which is probably the world's most visited site, recently decided to change from its own proprietary language to PHP.

Using PHP for Web applications

In the beginning, Web pages were static — they just presented documents. Users went to Web sites to read information. Documents were linked together so that users could easily find the information they sought, but the Web pages didn't change. Every user who arrived at a Web page saw the same thing.

Soon Web page developers wanted to do more. They wanted to interact with visitors, collect information from users, and provide Web pages that were customized for individuals. Several languages have developed that can be used to make Web sites dynamic. PHP is one of the most successful of these languages, evolving quickly to become more and more useful and rapidly growing in popularity.

PHP is a server-side scripting language, which means that the scripts are executed on the *server* (the computer where the Web site is located). This is different than JavaScript, another popular language for dynamic Web sites. JavaScript is executed by the browser, on the user's computer. Thus, JavaScript is a client-side language. Web servers and the interaction between servers and clients are discussed in the section "PHP for the Web," later in this chapter.

Because PHP scripts execute on the server, PHP can dynamically create the HTML code that generates the Web page, which allows individual users to see customized Web pages. Web page visitors see the output from scripts, but not the scripts themselves.

PHP has many features designed specifically for use in Web sites, including the following:

- **Interact with HTML forms:** PHP can display an HTML form and process the information that the user types in.
- **Communicate with databases:** PHP can interact with databases to store information from the user or retrieve information that is displayed to the user.
- **Generate secure Web pages:** PHP allows the developer to create secure Web pages that require users to enter a valid username and password before seeing the Web page content.

PHP features make these and many other Web page tasks easy.

PHP is only server-side, meaning it can't interact directly with the user's computer. That means PHP can't initiate actions based on the status of the user's computer, such as mouse actions or screen size. Therefore, PHP alone can't produce some popular effects, such as navigation menus that drop down or change color. On the other hand, JavaScript, a client-side scripting language,

can't access the server, limiting its possibilities. For example, you can't use JavaScript to store data on the server or retrieve data from the server. But wait! You don't have to choose. You can use JavaScript and PHP together to produce Web pages that neither can produce alone. See Chapter 11 for details on using JavaScript and PHP together.

Using PHP for database applications

PHP is particularly strong in its ability to interact with databases. PHP supports pretty much every database you've ever heard of and some you haven't. PHP handles connecting to the database and communicating with it, so you don't need to know the technical details for connecting to a database or for exchanging messages with it. You tell PHP the name of the database and where it is, and PHP handles the details. It connects to the database, passes your instructions to the database, and returns the database response to you.

Major databases currently supported by PHP include the following:

- dBASE
- Informix
- Ingres
- Microsoft SQL Server
- mSQL
- MySQL
- Oracle
- PostgreSQL
- Sybase

PHP supports other databases as well, such as filePro, FrontBase, and InterBase. In addition, PHP supports ODBC (Open Database Connectivity), a standard that allows you to communicate with even more databases, such as Access and IBM DB2.

PHP works well for a database-driven Web site. PHP scripts in the Web site can store data in and retrieve data from any supported database. PHP also can interact with supported databases outside a Web environment. Database use is one of PHP's best features.

Using PHP with your file system

PHP can interact with your file system — the directories and files that are on your local hard disk or on other computers accessible over a network. PHP can write into a file on your file system, creating the file if it doesn't exist, and can read the contents from files. It can also create directories, copy files, rename files, delete files, change file attributes, and perform many other file system tasks. PHP allows you to perform almost any task related to your file system.

Many Web sites need to interact directly with the file system. For example, a Web application may save information temporarily in a file rather than in a database, or may need to read information from a file.

System administrative and maintenance scripts frequently need to interact with the file system. For example, you may want to use a PHP script to back up files, to clean out directories, or to process text files by reformatting their contents. PHP can perform these tasks quite well.

Using PHP for system commands

PHP can interact with your operating system to perform any task the operating system can perform. You can execute an operating system command and receive the output. For example, you can execute a `dir` or `ls` command (to list the files in your directory) from PHP and receive the list of filenames that the `dir/ls` command produces.

The ability to execute system commands is often useful for system administrative and maintenance tasks. For example, you may want to clean up a directory by deleting files with a particular extension. You can use a system command to get a list of files in a directory and then identify and delete the files with the unwanted extension.

The ability to execute system commands includes the ability to run any other program on the system. Thus, you can run programs in other languages from PHP and make use of the output. Aren't you relieved that you don't have to rewrite all those programs you're using now? You can run Perl, C, shell scripts, or any other language programs from PHP. New PHP programs can add functionality to your system tools, without requiring you to spend time rewriting existing tools.

Understanding How PHP Works

PHP is a high-level language, which means that it's human-friendly, similar to English. Because your computer doesn't understand English, you use PHP to communicate, and the PHP interpreter converts the language in your PHP script to language the computer can understand. The computer then follows your instructions, passed to it by the interpreter.

The PHP interpreter comes in two flavors, one for use with Web sites and one that you run from the command line, independent of the Web. You can install either or both.

PHP as a general-purpose language

When you use PHP as a general-purpose scripting language, you install PHP CLI, the version of PHP developed for this purpose. You access the PHP interpreter from the command line to run your PHP script. The process is similar to other languages, such as Perl or C. For the lowdown on running scripts using PHP CLI, check out Chapter 3.

How the World Wide Web works

It's helpful to understand a little about how the World Wide Web (WWW) works. The Web is a network of computers that offer Web pages. Millions of Web sites are on the Web. To enable Web surfers to find the Web sites they want to visit, each Web page has an address, called a *URL*. This includes the Web site's domain name and the filename, such as `www.mycompany.com/welcome.html`. When Web surfers want to visit a Web page, they type the URL into their Web browsers. The following process is set in motion:

1. The Web browser sends a message out onto the Web, requesting the Web page.
2. The message is sent to the computer at the address specified in the URL.
3. The Web server software on the addressed computer receives the message.
4. The Web server searches for the requested HTML file.
5. The Web server finds the requested file and sends the file to the Web browser that requested it. (If it can't find the file, it sends a message to the browser saying that it couldn't find the file.)
6. The Web browser displays the Web page based on the HTML code it received.

PHP for the Web

When used on your Web site, PHP works in partnership with your Web server. Every Web site requires a Web server. The Web sever is the software that delivers your Web pages to the world. The PHP software works in conjunction with the Web server.

When used on the Web, PHP is an *embedded scripting language.* This means that PHP code is embedded in HTML code. You use HTML tags to enclose the PHP language that you embed in your HTML file. You create and edit Web pages containing PHP the same way you create and edit regular HTML pages.

When PHP is installed, the Web server is configured to look for PHP code embedded in files with specified extensions. It's common to specify the extensions `.php` or `.phtml`, but you can configure the Web server to look for any extension. When the Web server gets a request for a file with the designated extension, it sends the HTML statements as is, but PHP statements are processed by the PHP software before they're sent to the requester.

When PHP language statements are processed, the output consists of HTML statements. The PHP language statements are not included in the HTML sent to the browser, so the PHP code is secure and transparent to the user. For example, consider this simple PHP statement:

```
<?php echo "<p>Hello World"; ?>
```

In this statement, `<?php` is the PHP opening tag, `?>` is the closing tag, and `echo` is a PHP instruction that tells PHP to output the text that follows it as plain HTML code. The PHP software processes the PHP statement and outputs the following:

```
<p>Hello World
```

This is a regular HTML statement that is delivered to the user's browser. The PHP statement itself is not delivered to the browser, so the user never sees any PHP statements.

PHP and the Web server must work closely together. PHP is not integrated with all Web servers but works with many of the most popular ones. PHP is developed as a project under the Apache Software Foundation and, consequently, works best with Apache. PHP also works with Microsoft IIS/PWS, iPlanet (formerly Netscape Enterprise Server), and others.

Serving up Web servers

The software that delivers Web pages to the world is called a *Web server.* Several Web servers are available, but the most popular one is Apache. Approximately 60 percent of Web sites on the World Wide Web use Apache, according to surveys at `www.netcraft.com` and `www.securityspace.com/s_survey/data/`. Apache is open source software, which means it's free. It's available for all major operating systems. It's automatically installed with most Linux distributions and is preinstalled on Mac OS X. You can find information about Apache at `httpd.apache.org`. PHP is a project of the Apache Software Foundation, so PHP runs best with Apache.

Other Web servers are available. Internet Information Server (IIS) is the second most popular Web server with about 30 percent of the Web sites. IIS is developed by Microsoft and runs only on Windows. IIS is installed by default with Windows server software. Other Web servers include Zeus, NCSA, and Sun ONE. No other Web server is used on more than 2.5 percent of the Web sites.

Keeping Up with Changes in PHP

PHP is open source software. If you have only used software from major software publishers — such as Microsoft, Macromedia, or Adobe — you will find that open source software is an entirely different species. It's developed by a group of programmers who write the code in their spare time, for fun and for free. There's no corporate office to call with questions. There's no salesperson to convince you of the wonders of the software. There's no technical support phone number where you can be put on hold.

Sounds like there's no support for PHP, doesn't it? Actually, quite the opposite is true: An incredible amount of support is available. PHP is supported by the developers and by the many PHP users. But you need to look for the support. It's part of your job as a PHP user and developer to search out the information you need.

Open source software changes frequently, rather than once every year or two as commercial software does. It changes when the developers feel it's ready. It also changes quickly in response to problems. When a serious problem, such as a security hole, is found, a new version that fixes the problem may be released in days. You don't receive glossy brochures or see splashy magazine ads for a year before a new version is released. If you don't make the effort to stay informed, you may miss the release of a new version or be unaware of a serious problem with your current version.

Visit the PHP Web site often. You need to know the information that's published there. Join the mailing lists, which often are very high in traffic. When you first start using PHP, the large number of mail messages on the discussion lists brings valuable information into your e-mail box; you can pick up a lot by reading those messages. And soon, you may be able to help others based on your own experience. At the very least, subscribe to the announcement mailing list, which only delivers e-mail occasionally. Any important problems or new versions are announced here. The e-mail you receive from the announcement list contains information you need to know.

So, right now, before you forget, hop over to the PHP Web site and sign up for a list or two at `www.php.net/mailing-lists.php`.

PHP 5

Most of the important changes in PHP version 5 don't affect the coding or the use of PHP. They affect the performance of PHP. The Zend engine (the magic, invisible engine that powers PHP) has been significantly improved, and as a result, scripts run faster and more efficiently.

The object-oriented programming features of PHP are a major focus of PHP 5. Object-oriented programming is greatly improved over PHP 4. The creation and use of objects runs much faster, many object-oriented features have been added, and exceptions are introduced. Programmers who prefer object-oriented programming will be much happier with PHP 5. (Object-oriented programming is described in Chapter 9.)

With PHP 5, the names of the PHP programs changed. PHP for the Web is called `php-cgi`; PHP CLI is called just `php`, as in `php.exe` on Windows. Both are stored in the directory where PHP is installed. Prior to PHP 5, both programs were named `php.exe`, but stored in different subdirectories.

PHP 5 adds support for MySQL 4.1 and later. However, support for MySQL is not included with PHP 5 by default. Support for MySQL 4.0 or MySQL 4.1 must be specified when PHP is installed. Prior to PHP 5, support for MySQL 4.0 and earlier was included automatically.

PHP 5 includes support for SQLite by default. SQLite provides quick and easy methods for storing and retrieving data in flat files.

Previous versions of PHP

You should be aware of some significant changes in previous PHP versions because existing scripts that work fine on earlier versions may have problems when they're run on a later version, and vice versa. The following are some changes you should be aware of:

- **Version 4.3.1:** Fixed a security problem in 4.3.0. It's not wise to continue to run a Web site using versions 4.3.0 or earlier.
- **Version 4.3.0:** Included significant improvements to the CLI version of PHP, which is now built by default when you compile PHP from source code (described in Appendix A). You must disable its build with installation options if you don't want it to be built.
- **Version 4.2.0:** Changed the default setting for `register_globals` to `Off`. Scripts running under previous versions may depend on `register_globals` being set to `On` and may stop running with the new setting. It's best to change the coding of the script so that it runs with `register_globals` set to `Off`.
- **Version 4.1.0:** Introduced the superglobal arrays. Scripts written using the superglobals (described in Chapter 6) won't run in earlier versions. Prior to 4.1.0, you must use the old style arrays, such as `$HTTP_POST_VARS`.

By the time you read this, it's possible that everyone has updated to PHP 5. However, some IT departments and Web hosting companies may not update immediately. Keep the previous changes in mind when using older versions.