Chapter 1

Introduction to PHP and MySQL

In This Chapter

- Finding out what a Web database application is
- Discovering how MySQL works
- Taking a look at PHP
- Finding out how PHP and MySQL work together

So you need to develop an interactive Web site. Perhaps your boss just put you in charge of the company's online product catalog. Or you want to develop your own Web business. Or your sister wants to sell her paintings online. Or you volunteered to put up a Web site open only to members of your circus acrobats' association. Whatever your motivation might be, you can see that the application needs to store information (such as information about products or member passwords), thus requiring a database. You can see also that the application needs to interact *dynamically* with the user; for instance, the user selects a product to view or enters membership information. This type of Web site is a *Web database application*.

I assume that you've created static Web pages before, using HTML (HyperText Markup Language), but creating an interactive Web site is a new challenge, as is designing a database. You asked three computer gurus you know what you should do. They said a lot of things you didn't understand, but among the technical jargon, you heard "quick" and "easy" and "free" mentioned in the same sentence as PHP and MySQL. Now you want to know more about using PHP and MySQL to develop the Web site that you need.

PHP and MySQL work together very well; it's a dynamic partnership. In this chapter, you find out the advantages of each, how each one works, and how they work together to produce a dynamic Web database application.

What Is a Web Database Application?

An *application* is a program or a group of programs designed for use by an end user (for example, customers, members, or circus acrobats). If the end user interacts with the application via a Web browser, the application is a *Web based* or *Web application.* If the Web application requires the long-term storage of information using a database, it is a *Web database application.* This book provides you with the information that you need to develop a Web database application that can be accessed with Web browsers such as Internet Explorer and Netscape.

A Web database application is designed to help a user accomplish a task. It can be a simple application that displays information in a browser window (for example, current job openings when the user selects a job title) or a complicated program with extended functionality (for example, the book-ordering application at Amazon.com or the bidding application at eBay).

A Web database application consists of just two pieces:

- **Database:** The *database* is the long-term memory of your Web database application. The application can't fulfill its purpose without the database. However, the database alone is not enough.
- **Application:** The *application* piece is the program or group of programs that performs the tasks. Programs create the display that the user sees in the browser window; they make your application interactive by accepting and processing information that the user types in the browser window; and they store information in the database and get information out of the database. (The database is useless unless you can move data in and out.)

The Web pages that you've previously created with HTML alone are *static,* meaning the user can't interact with the Web page. All users see the same Web page. *Dynamic* Web pages, on the other hand, allow the user to interact with the Web page. Different users might see different Web pages. For instance, one user looking at a furniture store's online product catalog might choose to view information about the sofas, whereas another user might choose to view information about coffee tables. To create dynamic Web pages, you must use another language in addition to HTML.

One language widely used to make Web pages dynamic is JavaScript. JavaScript is useful for several purposes, such as mouse-overs (for example, to highlight a navigation button when the user moves the mouse pointer over it) or accepting and validating information that users type into a Web form. However, it's not useful for interacting with a database. You wouldn't use JavaScript to move the information from the Web form into a database. PHP, however, is a language particularly well suited to interacting with databases. PHP can accept and validate the information that users type into a Web form and can also move the information into a database. The programs in this book are written with PHP.

The database

The core of a Web database application is the *database,* which is the long-term memory (I hope more efficient than my long-term memory) that stores information for the application. A database is an electronic file cabinet that stores information in an organized manner so that you can find it when you need it. After all, storing information is pointless if you can't find it. A database can be small, with a simple structure — for example, a database containing the titles and authors' names of all the books that you own. Or a database can be huge, with an extremely complex structure — such as the database that Amazon.com has to hold all its information.

The information that you store in the database comes in many varieties. A company's online catalog requires a database to store information about all the company's products. A membership Web site requires a database to store information about members. An employment Web site requires a database (or perhaps two databases) to store information about job openings and information from résumés. The information that you plan to store could be similar to information that's stored by Web sites all over the Internet — or information that's unique to your application.

Technically, the term *database* refers to the file or group of files that holds the actual data. The data is accessed by using a set of programs called a DBMS (Database Management System). Almost all DBMSs these days are RDBMSs (Relational Database Management Systems), in which data is organized and stored in a set of related tables.

In this book, MySQL is the RDBMS used because it is particularly well suited for Web sites. MySQL and its advantages are discussed in the section, "MySQL, My Database," later in this chapter. You can find out how to organize and design a MySQL database in Chapter 3.

The application: Moving data into and out of the database

For a database to be useful, you need to be able to move data into and out of it. Programs are your tools for this because they interact with the database to store and retrieve data. A program connects to the database and makes a request: "Take this data and store it in the specified location." Another program makes the request: "Find the specified data and give it to me." The application programs that interact with the database run when the user interacts with the Web page. For instance, when the user clicks the submit button after filling in a Web form, a program processes the information in the form and stores it in a database.

E-mail discussion lists

Good technical support is available from *e-mail discussion lists,* which are groups of people discussing specific topics through e-mail. E-mail lists are available for pretty much any subject you can think of: Powerball, ancient philosophy, cooking, the Beatles, Scottish terriers, politics, and so on. The *list manager* maintains a distribution list of e-mail addresses for anyone who wants to join the discussion. When you send a message to the discussion list, your message is sent to the entire list so that everyone can see it. Thus, the discussion is a group effort, and anyone can respond to any message that interests him or her.

E-mail discussion lists are supported by various sponsors. Any individual or organization can run a list. Most software vendors run one or more lists devoted to their software. Universities run many lists for educational subjects. In addition, some Web sites manage discussion lists, such as Yahoo! Groups and Topica. Users can create a new list or join an existing list through the Web application.

Software-related e-mail lists are a treasure trove of technical support. Anywhere from a hundred to several thousand users of the software subscribe to the list. Often the developers, programmers, and technical support staff for the software vendor are on the list. You are unlikely to be the first person to ever experience your problem. Whatever your question or problem, someone on the list probably knows the answer or the solution. When you post a question to an e-mail list, the answer usually appears in your inbox within minutes. In addition, most lists maintain an archive of previous discussions so that you can search for answers. When you're new to any software, you can find out a great deal simply by joining the discussion list and reading the messages for a few days.

PHP and MySQL have e-mail discussion lists. Actually, each has several discussion lists for special topics, such as *databases and PHP.* You can find the names of the mailing lists and instructions for joining them on the PHP and MySQL Web sites.

MySQL, My Database

MySQL is a fast, easy-to-use RDBMS used on many Web sites. Speed was the developers' main focus from the beginning. In the interest of speed, they made the decision to offer fewer features than their major competitors (such as Oracle and Sybase). However, even though MySQL is less full-featured than its commercial competitors, it has all the features needed by the majority of database developers. It's easier to install and use than its commercial competitors, and the difference in price is strongly in MySQL's favor.

MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company. The company licenses it in two ways:

- **Open source software:** MySQL is available through the GNU GPL (General Public License). MySQL provides two versions of the open source software:

- **MySQL Community Edition:** A freely downloadable, open source edition of MySQL, released early and often with the most advanced features. Anyone who can meet the requirements of the GPL can use the software for free. If you're using MySQL as a database on a Web site (the subject of this book), you can use MySQL for free, even if you're making money with your Web site.
- **MySQL Network:** An enterprise-grade set of software and services available for a monthly subscription fee. MySQL Network provides certified software, thoroughly tested and optimized. Services include technical support, regular updates, access to a knowledge base of hundreds of technical articles, and other services useful to a large business. The subscription is available at four levels, from the Basic level, with a limit of two incidents, no phone support, and a two-day response time, to Platinum support, with unlimited incidents, 24/7 phone support, and a 30-minute response time.

- **Commercial license:** MySQL is available with a commercial license for those who prefer it to the GPL. If a developer wants to use MySQL as part of a new software product and wants to sell the new product rather than release it under the GPL, the developer needs to purchase a commercial license.

Finding technical support for MySQL Community Edition is not a problem. You can join one of several e-mail discussion lists offered on the MySQL Web site at `www.mysql.com`. You can even search the e-mail list archives, which contain a large archive of MySQL questions and answers.

Advantages of MySQL

MySQL is a popular database with Web developers. Its speed and small size make it ideal for a Web site. Add to that the fact that it's open source, which means free, and you have the foundation of its popularity. Here is a rundown of some of its advantages:

- **It's fast.** The main goal of the folks who developed MySQL was speed. Thus, the software was designed from the beginning with speed in mind.
- **It's inexpensive.** MySQL is free under the open source GPL license, and the fee for a commercial license is reasonable.
- **It's easy to use.** You can build and interact with a MySQL database by using a few simple statements in the SQL language, which is the standard language for communicating with RDBMSs. Check out Chapter 4 for the lowdown on the SQL language.

- **It can run on many operating systems.** MySQL runs on many operating systems — Windows, Linux, Mac OS, most varieties of Unix (including Solaris and AIX), FreeBSD, OS/2, Irix, and others.
- **Technical support is widely available.** A large base of users provides free support through mailing lists. The MySQL developers also participate in the e-mail lists. You can also purchase technical support from MySQL AB for a small fee.
- **It's secure.** MySQL's flexible system of authorization allows some or all database privileges (such as the privilege to create a database or delete data) to specific users or groups of users. Passwords are encrypted.
- **It supports large databases.** MySQL handles databases up to 50 million rows or more. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
- **It's customizable.** The open source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

How MySQL works

The MySQL software consists of the MySQL server, several utility programs that assist in the administration of MySQL databases, and some supporting software that the MySQL server needs (but you don't need to know about). The heart of the system is the MySQL server.

The MySQL server is the manager of the database system. It handles all your database instructions. For instance, if you want to create a new database, you send a message to the MySQL server that says "create a new database and call it newdata." The MySQL server then creates a subdirectory in its data directory, names the new subdirectory `newdata`, and puts the necessary files with the required format into the `newdata` subdirectory. In the same manner, to add data to that database, you send a message to the MySQL server, giving it the data and telling it where you want the data to be added. You find out how to write and send messages to MySQL in Part II.

Before you can pass instructions to the MySQL server, it must be running and waiting for requests. The MySQL server is usually set up so that it starts when the computer starts and continues running all the time. This is the usual setup for a Web site. However, it's not necessary to set it up to start when the computer starts. If you need to, you can start it manually whenever you want to access a database. When it's running, the MySQL server listens continuously for messages that are directed to it.

Communicating with the MySQL server

All your interaction with the database is accomplished by passing messages to the MySQL server. You can send messages to the MySQL server several ways, but this book focuses on sending messages using PHP. The PHP software has specific statements that you use to send instructions to the MySQL server.

The MySQL server must be able to understand the instructions that you send it. You communicate by using *SQL (Structured Query Language),* which is a standard language understood by many RDBMSs. The MySQL server understands SQL. PHP doesn't understand SQL, but it doesn't need to: PHP just establishes a connection with the MySQL server and sends the SQL message over the connection. The MySQL server interprets the SQL message and follows the instructions. The MySQL server sends a return message, stating its status and what it did (or reporting an error if it was unable to understand or follow the instructions). For the lowdown on how to write and send SQL messages to MySQL, check out Part II.

PHP, a Data Mover

PHP, a scripting language designed specifically for use on the Web, is your tool for creating dynamic Web pages. Rich in features that make Web design and programming easier, PHP is in use on more than 20 million domains (according to the Netcraft survey at `www.php.net/usage.php`). Its popularity continues to grow, so it must be fulfilling its function pretty well.

PHP stands for *PHP: HyperText Preprocessor.* In its early development by a guy named Rasmus Lerdorf, it was called *Personal Home Page tools.* When it developed into a full-blown language, the name was changed to be more in line with its expanded functionality.

The PHP language's syntax is similar to the syntax of C, so if you have experience with C, you'll be comfortable with PHP. PHP is actually simpler than C because it doesn't use some of the more difficult concepts of C. PHP also doesn't include the low-level programming capabilities of C because PHP is designed to program Web sites and doesn't require those capabilities.

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. 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.

Technical support is available for PHP. You can join one of several e-mail discussion lists offered on the PHP Web site (`www.php.net`), including a list for *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.

Advantages of PHP

The popularity of PHP is growing rapidly because of its many advantages:

- **It's fast.** Because it is embedded in HTML code, the response time is short.
- **It's inexpensive — free, in fact.** PHP is proof that free lunches do exist and that you can get more than you paid for.
- **It's easy to use.** PHP contains many special features and functions needed to create dynamic Web pages. The PHP language is designed to be included easily in an HTML file.
- **It can run on many operating systems.** It runs on a variety of operating systems — Windows, Linux, Mac OS, and most varieties of Unix.
- **Technical support is widely available.** A large base of users provides free support through e-mail discussion lists.
- **It's secure.** The user does not see the PHP code.
- **It's designed to support databases.** PHP includes functionality designed to interact with specific databases. It relieves you of the need to know the technical details required to communicate with a database.
- **It's customizable.** The open source license allows programmers to modify the PHP software, adding or modifying features as needed to fit their own specific environments.

How PHP works

PHP is an embedded scripting language when used in Web pages. 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 — the same way that you would use other HTML tags. You create and edit Web pages containing PHP the same way that you create and edit regular HTML pages.

The PHP software works with the Web server. The Web server is the software that delivers Web pages to the world. When you type a URL into your Web browser, you're sending a message to the Web server at that URL, asking it to send you an HTML file. The Web server responds by sending the requested file. Your browser reads the HTML file and displays the Web page. You also request the Web server to send you a file when you click a link in a Web page. In addition, the Web server processes a file when you click a Web page button that submits a form.

When PHP is installed, the Web server is configured to expect certain file extensions to contain PHP language statements. Often the extension is `.php` or `.phtml`, but any extension can be used. 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, only the output is sent by the Web server to the Web browser. The PHP language statements are not included in the output sent to the browser, so the PHP code is secure and transparent to the user. For instance, in this simple PHP statement:

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

`<?php` is the PHP opening tag, and `?>` is the closing tag. `echo` is a PHP instruction that tells PHP to output the upcoming text. The PHP software processes the PHP statement and outputs this:

```
<p>Hello World
```

which is a regular HTML statement. This HTML statement is delivered to the user's browser. The browser interprets the statement as HTML code and displays a Web page with one paragraph — Hello World. The PHP statement 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 does work with many of the popular Web servers. PHP is developed as a project of the Apache Software Foundation — thus, it works best with Apache. PHP also works with Microsoft IIS/PWS, iPlanet (formerly Netscape Enterprise Server), and others.

Although PHP works with several Web servers, it works best with Apache. If you can select or influence the selection of the Web server used in your organization, select Apache. By itself, Apache is a good choice. It is free, open source, stable, and popular. It currently powers more than 60 percent of all Web sites, according to the Web server survey at `www.netcraft.com`. It runs on Windows, Linux, Mac OS, and most flavors of Unix.

MySQL and PHP, the Perfect Pair

MySQL and PHP are frequently used together. They are often called the *dynamic duo.* MySQL provides the database part, and PHP provides the application part of your Web database application.

Advantages of the relationship

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 Web sites. Both have a set of features focused on building dynamic Web sites.
- **They're easy to use.** Both were designed to get a Web site 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 are often used as a pair, they often have the same user base. Many people are available to help, including those 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.

How MySQL and PHP work together

PHP provides the application part, and MySQL provides the database part of a Web database application. You use the PHP language to write the programs that perform the application tasks. PHP can be used for simple tasks (such as displaying a Web page) or for complicated tasks (such as accepting and verifying data that a user typed into an HTML form). One of the tasks that your application must do is move data into and out of the database — and PHP has built-in features to use when writing programs that move data into and out of a MySQL database.

PHP statements are embedded in your HTML files with PHP tags. When the task to be performed by the application requires storing or retrieving data, you use specific PHP statements designed to interact with a MySQL database. You use one PHP statement to connect to the correct database, telling PHP where the database is located, its name, and the password needed to connect to it. The database doesn't need to be on the same machine as your Web site; PHP can communicate with a database across a network. You use another PHP statement to send an SQL message to MySQL, giving MySQL instructions for the task you want to accomplish. MySQL returns a status message that shows whether it successfully performed the task. If there was a problem, it returns an error message. If your SQL message asked to retrieve some data, MySQL sends the data that you asked for, and PHP stores it in a temporary location where it is available to you.

You then use one or more PHP statements to complete the application task. For instance, you can use PHP statements to display data that you retrieved. Or you might use PHP statements to display a status message in the browser, informing the user that the data was saved.

As an RDBMS, MySQL can store complex information. As a scripting language, PHP can perform complicated manipulations of data, on either data that you need to modify before saving it in the database or data that you retrieved from the database and need to modify before displaying or using it for another task. Together, PHP and MySQL can be used to build a sophisticated and complicated Web database application.

Keeping Up with PHP and MySQL Changes

PHP and MySQL are open source software. If you've used only software from major software publishers — such as Microsoft, Macromedia, or Adobe — you'll 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.

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

Visit the PHP and MySQL Web sites often. You need to know the information that's published there. Join the mailing lists, which often are high in traffic. When you first get acquainted with PHP and MySQL, 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 might be able to help others based on your own experience. At the very least, subscribe to the announcement mailing list, which delivers e-mail only occasionally. Any important problems or new versions are announced here. The e-mail that you receive from the announcement list contains information you need to know. So, right now, before you forget, hop over to the PHP and MySQL Web sites and sign up for a list or two at `www.php.net/mailing-lists.php` and `lists.mysql.com`.

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

- **Version 6.0.0:** Removed configuration setting for `register_globals`. Removed configuration setting for magic quotes. Removed the long arrays, such as `$HTTP_POST_VARS`.
- **Version 5.1.0:** Added a configuration setting to set a local time zone. If a time zone is not set, a notice is displayed.
- **Version 5.0.0:** Added support for MySQL 4.1 and later versions. Support for MySQL is not included automatically; it must be included with an option when PHP is installed. Also changed the filename of the PHP interpreter used with a Web server from `.php` to `.php-cgi`.
- **Version 4.3.1:** Fixed a security problem in 4.3.0. It's not wise to continue to run a Web site using version 4.3.0 or earlier.
- **Version 4.2.0:** Changed the default setting for `register_globals` to `Off`. Scripts running under previous versions might depend on `register_globals` being set to `On` and could stop running with the new setting. It's best to change the coding of the script so that it runs with `register_globals` set to `Off`.
- **Version 4.1.0:** Introduced superglobal arrays. Scripts written with the superglobals won't run in earlier versions. Prior to 4.1.0, you must use old style arrays, such as `$HTTP_POST_VARS`.