- Frequently Asked Questions (FAQ) management site
- News site
- Independent PHP application
- Simple AJAX calendar

Each project provides the opportunity to learn new skills and focuses on specific challenges.

The Technology

It should come as no surprise that the technology being used to build the Web applications in this book uses PHP and MySQL. If you picked up this book in the ASP section of your bookstore, I am afraid someone has played a cruel joke on you.

When put together, PHP and MySQL offer a compelling framework in which you can develop powerful and flexible Web applications. The reason they work well together is that each provides a comprehensive part of the Web development toolkit. In building any Web application, the first thing you need is some form of language in which to write dynamic pages and create features to handle dates, process data, connect to resources, manage users, and perform other tasks. PHP steps up to solve this problem. PHP is an incredibly flexible language with a huge array of functionality for solving common Web development challenges, many of which are covered in the projects in this book. The second requirement is to have somewhere to store the oodles of data that you will be displaying, updating, removing, modifying, and otherwise showing off. A solution for this challenge is to use a database, and MySQL provides a reliable and easy-to-use database that is well supported and flexible.

Before looking at the architecture of how the Web works, however, this chapter explores the tools of the trade in more detail.

PHP

PHP is a popular high-level scripting language used by a range of organizations and developers. Originally developed as a small Perl project by Rasmus Lerdorf in late 1995, PHP was intended as a means to assist in developing his home page, and as such he named it Personal Home Page (PHP) Tools.

When Lerdorf was contracted to work for the University of Toronto to build a dial-up system for students to access the Internet, he had no means of connecting Web sites to databases. To solve this problem, the enterprising Lerdorf replaced his

Perl code with a C wrapper that added the capability to connect his Web pages to a MySQL database. As his small project grew, he gave away his changes on the Internet as an Open Source project and cordially received improvements from other programmers with an interest in PHP. The language was later renamed to the current recursive acronym PHP: Hypertext Preprocessor by Zeev Suraski and Andi Gutmans after they rewrote the parser in 1997. The software continued to develop and now forms the comprehensive PHP platform we know today.

PHP provides a solid and well-defined programming language that includes support for object-orientated programming, conditions, file handling, arithmetic, and more. The language that PHP forms is similar in semantics to that of a shell scripting language combined with the easier bits of the C language.

PHP subscribes to the batteries-included philosophy of programming languages and includes extensive support for a huge range of needs, such as cookies, forms, sessions, include files, network sockets, e-mail, LDAP, IRC, and more. Database support covers not only MySQL but many others, including but not limited to PostgreSQL, Oracle, MS SQL, dBase, Sybase, and DB2. This flexible database support is useful if you ever need to port your application to a different database.

In addition to PHP's capability as a Web scripting language, PHP also can be used as a shell scripting language. This means that you can use a single language to write Web applications and create shell scripts to manage your computers. You can even use PHP for creating desktop applications. Although this usage was typically one for the wiry-haired and zany part of the PHP demographic, more and more developers are using it.

PHP also extends its batteries-included philosophy and includes support for third-party functionality via the PHP Extension and Application Repository (PEAR) library. PEAR works in a similar fashion to Perl CPAN modules and provides additional functionality that is easily available via a number of independent modules built to solve specific problems. These special modules can be included in your application to access this special functionality easily. For example, if you need to send e-mail using your Web application, you can use the special PEAR mail functionality that extends the included PHP mail support. This makes PHP better at supporting third-party extensions and has resulted in a huge number of freely available PEAR modules.

MySQL

MySQL is a powerful and comprehensive relational database server, which was originally developed by David Axmark, Allan Larsson, and Michael "Monty" Widenius. The commercial company they founded, MySQL AB, develops and markets

MySQL and associated products. Although the MySQL software originated as an Open Source project, its creators were confident that they could run a business using the product as a base. This business enables the developers to work full time on the software, which in turn benefits both the Open Source community and commercial users of MySQL. Both the open and commercial MySQL variants are functionally the same; the only difference in the software is how it is licensed. (Companies must buy a license if they want to deploy MySQL commercially in a closed source application.) MySQL has enjoyed enormous popularity, and its customers include Yahoo! Finance, MP3.com, Motorola, NASA, Silicon Graphics, and Texas Instruments.

MySQL is a full-featured database and uses open standards, such as the ANSI SQL 99 standard, for communicating with databases with Structured Query Language (SQL). This standard provides a means to insert, update, and query information in the database by using an industry standard language. This standard language is used across database products, and like other products, MySQL supports a number of additional SQL statements. As well as being a standardized database, MySQL is also multi-platform. This means that in addition to Linux, MySQL also runs on other operating systems, such as Windows, Mac OS X, or BSD and UNIX variants.

The database itself includes an interactive command-line client, which allows you to communicate with the server. Although this client is useful, it's a bit scary for new users who are unfamiliar with command-line zen. Fortunately, a number of graphical and Web-based clients can avoid the command-line encounter. (This book uses the Web-based phpMyAdmin client extensively.) MySQL also has support for a number of programming languages to access and query the database. This includes languages such as PHP, Python, Perl, C, C++, and Java, among others. Although you may wish to initially use only PHP to query the database, multi-language support is useful if you need to write modules and applications in different languages in the future.

How the Dynamic Web Works

At its most fundamental level, the PHP and MySQL system provides a means to allow dynamic content to be distributed to a networked device. This can be Uncle Bob connecting to your Web site, a delivery service connecting wirelessly to its tracking network, or you accessing your e-mail via the Web. Each of these different solutions uses essentially the same software components that access each other over different hardware contexts. The technical description for the type of programming you are engaging in with PHP and MySQL is called *client/server development*.

To fully understand this concept, this chapter offers several examples and explains how information is bounced between parts of the Web.

Imagine a Web site, any Web site. Go on—be exciting and imagine you are an undercover spy who just so happens to have a Web site with your favorite spy-related stuff. Assume that this Web site displays HTML code (the basic code that a Web browser understands) and nothing else. The page contains only a list of spy-related text, and there is no interactivity. You simply connect to www.thewebaddress-ofthesite.com, and the Web site displays the information. Figure 1-1 shows the kind of interaction involved with this example.

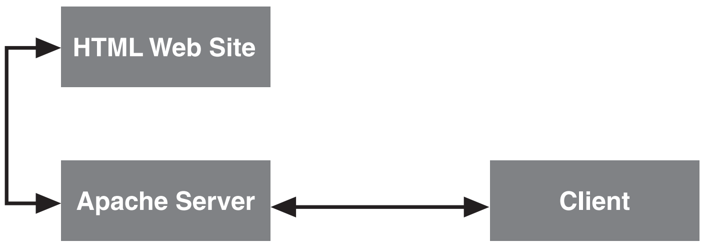

FIGURE 1-1
How a Web browser connects to a Web site

In this example, the client connects to the Web server (in this case, an Apache server) and requests an HTML page. When the client has connected, the Apache server returns the requested page to the client. In this example, the Apache server acts as a middleman for taking the requests and sending the responses back to the client.

Figure 1-2 demonstrates the slightly more complex example of how an HTML input form works.

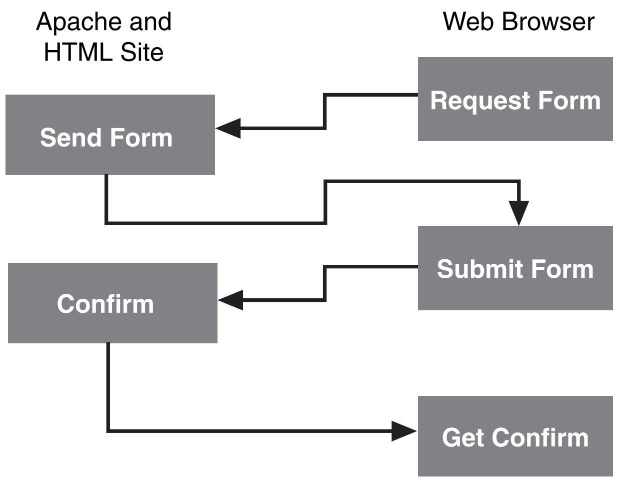

FIGURE 1-2
How the client and server deal with an HTML form

If you start at the bottom-right corner of this diagram, you can see that the very first part is that the Web browser requests the HTML form. The Apache server then responds and sends the form back to the browser. Next, the user fills in the form and submits it back to the Apache server. Apache then sends a confirmation page back to the client. By following the direction of the arrows, you can see how the process works and how information is sent between the client and the server.

The next example, shown in Figure 1-3, demonstrates what happens when you roll a database server into the mix. In this case, you not only are filling in the form but you want to store the contents of the form in the database. For this to happen, you need to include one extra step when the Web site receives the contents of the form. The Web site must send out a confirmation page and put the data in the database. Because putting the data in the database is a one-way process, no two-way arrow is included in this figure.

The model shown in Figure 1-3 is virtually the same as before but with the extra stage added. In addition to putting information into a database, you also need to be able to retrieve information from the database to display the results. In Figure 1-4, you get the client to request a form, fill in the request details (such as a search form), and then query the database for the results and return them to the client.

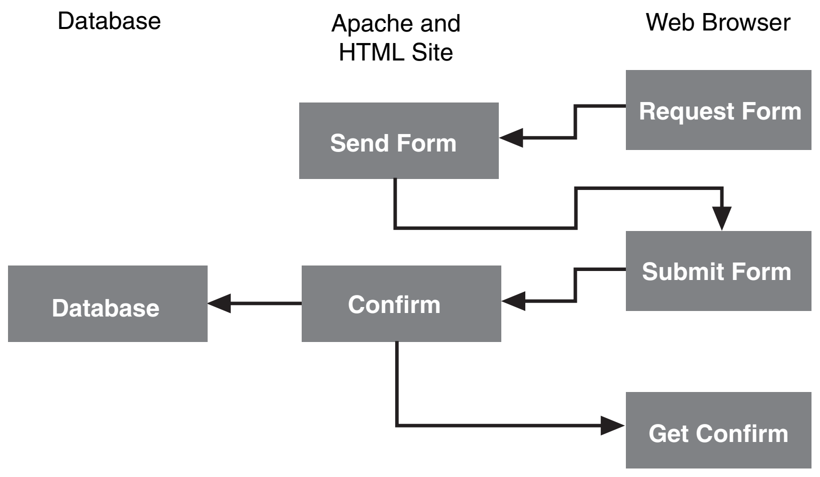

FIGURE 1-3
When a database is thrown in, the information flow is a little different.

In this example, you begin by requesting the form from the server. The server sends the form back to the browser, and the user fills in the information he needs and submits the form. Next, the server receives the search string and requests from the database the items that match the search term. The database then finds the data and returns the relevant results to the server, which in turn sends the results to the client.

The client/server examples described here use a fairly loose definition of the different components (such as the database, server, and browser) in the models shown. The reason for this is so that you can concentrate on how data flows between the different major parts of the client/server system.

This last example (as shown in Figure 1-5) explains in detail how each step works in terms of these components.

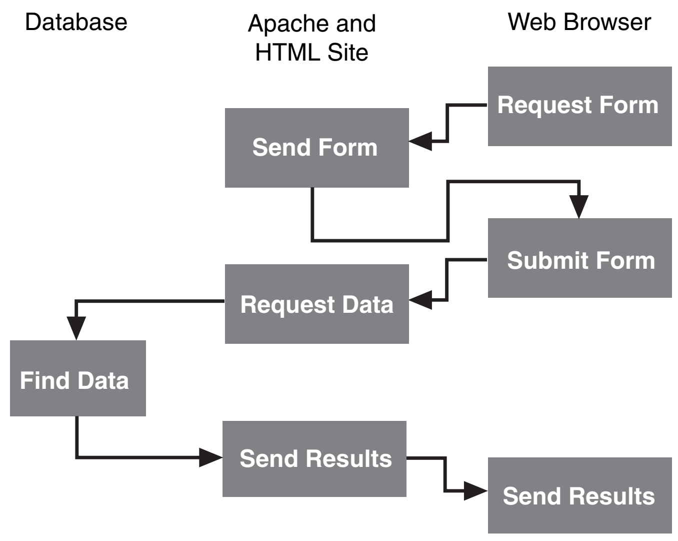

FIGURE 1-4
This is a common situation—the user fills in a form and gets data from the database.

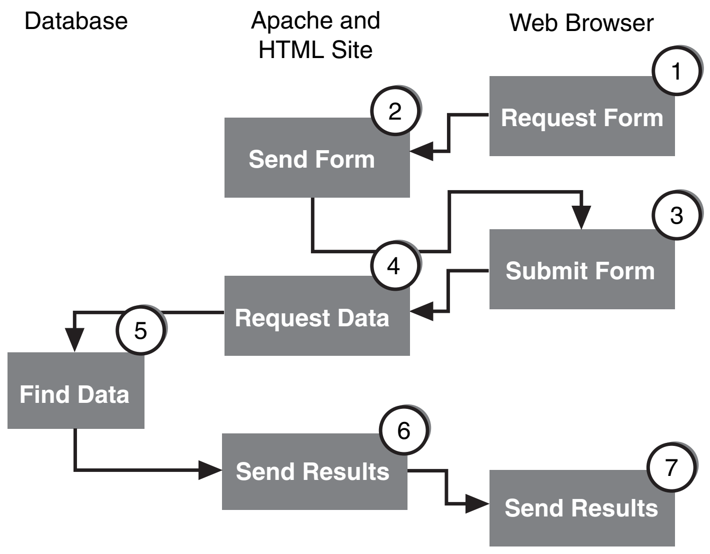

FIGURE 1-5
Each number refers to a point below.

1. The user types a Web address in a Web browser (the client) to access the site. This connection also requests the HTML page containing the HTML form.
2. The browser connects to the Apache server, which has the HTML and PHP files that form the Web site. Apache services the request (by following the rules in its configuration file to find the relevant page and send it back) and sends the client the Web page containing the HTML form.
3. The user fills in the form (in the Web browser) and submits the form to the server.
4. The Apache server receives the submitted form and loads the relevant file to handle the form submission. This file contains PHP code that is used to connect to the database. This PHP code is passed to the PHP interpreter by Apache and run by the interpreter. PHP connects to the MySQL database (which may be on the same computer or another one; it doesn't matter

which). When connected to the MySQL database, a request is made for the information by using SQL, which is a language designed for communicating with databases.

5. The MySQL database receives the SQL request and finds the information. When the information is located, the result is sent back to the PHP script that made the request.
6. The PHP script receives the result from the MySQL server and constructs an HTML page with the results included before sending it back to the Web browser client.
7. The Web browser receives the HTML result of the query and displays it to the user.

Throughout this process, you'll want to bear in mind a few key items. It is recommended that you read the following key points and then review the preceding process to clarify any confusion.

To begin, the Web browser understands hypertext only; it does not understand PHP. (Of course, the former is the assumption that one makes for all Web browsers because they all are built to understand hypertext. Some browsers may understand scripting languages such as JavaScript, but that example is out of the scope of this discussion.) All communication to and from the Web browser is done in hypertext (hence converting the results of the MySQL query to hypertext in Step 6). Another point is that PHP is tightly linked with Apache on the server, and all database connections/queries are executed by PHP. This tight integration involves the PHP process being closely linked to Apache for high performance. PHP can be thought of as the middle ground for accessing databases, files, XML, and so on, and this middle ground sends everything out of the machine via Apache. MySQL should be thought of purely as a data storage device that is useful only when something else connects to it to add, remove, update, or request information. You can consider MySQL as the equivalent of a hard disk for a computer; a hard disk is useful only if there is software to access it and store data on it. PHP is the metaphorical equivalent to this "software."

Summary

As with learning anything, PHP and MySQL development has a number of underlying concepts that need to be understood before you hit the nitty-gritty of the subject. The major concepts have been discussed in this chapter to provide a solid foundation.