# Part I

# Learning the Basics

# Chapter 1

# Introducing PHP and MySQL

If you think back a little, you'll remember how the Web first began, with static HTML pages on which image maps and animated GIFs were considered cutting-edge. And you'll remember how web users clamored for more interactivity on those static pages, interactivity that became simpler once dynamic HTML and JavaScript became standard accessories for your Internet browser.

Well, it isn't your grandmother's Web any more. . . .

The current generation of web designers thinks nothing of animated GIFs and pop-up boxes, preferring instead to use brightly colored Flash animation and live video feeds. And that's just what they're doing in your browser. A similar revolution has been taking place in the backroom, with the current crop of server-side languages giving web developers a brand new sandbox to play in.

That's where this introductory chapter comes in. The next few pages give you a quick overview of how server-side scripting can be combined with a database system to create some useful and powerful applications. This chapter also introduces you to the stars of this book—PHP and MySQL—explaining what they are, how they came into being, and why they make such a good couple.

## Server-Side Applications…

Server-side scripting is not new. It's been around for quite a while, and almost every major web site uses some amount of server-side scripting. Amazon.com uses it to find the book you're looking for, Yahoo! uses it to store your personal preferences, and eBay uses it to process your credit card number for that gigantic eight-headed stone eagle you just bought. What *has* changed, however, is that it's no longer the domain of the big guns—as programming languages have matured and the barriers to entry have lowered, independent web publishers are increasingly using server-side technologies to deliver a better experience to their users.

If your primary experience with web development has been with JavaScript, the popularity of server-side languages like Perl and PHP might be hard to understand; after all, you've already seen what a few JavaScripts can do. However, JavaScript runs within a client application—the browser—and as such can only access resources, such as the current date and time, on the client machine. JavaScript also has limited storage capabilities for user data—for example, while a web site can certainly store user preferences in a *cookie* on the user's hard drive with JavaScript, those preferences can only be retrieved if the user returns to that site from the same computer (because the cookie will not exist on any computer other than the one that was originally used).

### A Fine Balance

Just because you can do a lot more with server-side scripts doesn't mean that you get to bin your copy of the JavaScript manual. Often, client-side scripting is the most efficient way to perform tasks localized to the user interface. It's hard to imagine, for example, how a server-side script could help with an image rollover or a page transition effect. Similarly, when dealing with user input in web forms, client-side checks are a necessary first step to verifying the validity of entered data; performing basic checks on the client alerts the user to errors faster and reduces the number of round-trips to the server. A judicious mix of the two is thus essential to creating web applications that are fast and easy to use, yet robust and error-free.

Server-side scripts run on the web server, usually a powerful UNIX or Microsoft Windows system with oodles of RAM and CPU cycles; they can, therefore, be used to access server resources, such as databases or external files, and perform more sophisticated tasks than regular client-side scripting. For example, a server-side script could store a user's shopping cart in a database, and retrieve it on the user's next visit to save him some time reselecting items for purchase; this translates into an improved customer experience (and it doesn't matter which computer the user logs in from, because the settings are all on the server and, thus, are always available).

## … And the Databases That Love Them

The large majority of server-side scripts are related to either getting information from the user and saving it somewhere, or retrieving information from somewhere and presenting it. This "somewhere" is usually an animal called a database, and if you're at all serious about building useful web applications, you're going to need to make friends with it.

A *database,* fundamentally, is a collection of data organized and classified according to some criteria. The traditional analogy is that of a filing cabinet containing many drawers, with each drawer holding files related to a particular subject. This organization of information into drawers and files makes it easy to retrieve specific bits of information quickly—to lay your hands on a particular piece of information, you pull open the appropriate drawer and select the file(s) you need.

An electronic database management system (DBMS) helps you organize information and provides a framework to access it quickly and efficiently. The drawers that contain the files are referred to in database parlance as *tables,* while the files themselves are called *records*. The act of pulling out information is referred to as a *query,* and it's usually expressed using Structured Query Language (SQL). The resulting data is referred to as a *result set*. These terms might seem foreign to you at the moment, but by the end of this book, you'll be tossing them around like a pro.

A *relational database management system* (RDBMS) takes things one step further by creating relationships among the tables that make up a database. These relationships can then be used to combine data from multiple tables, allowing different perspectives and more useful reports. By creating links among related pieces of information, an RDBMS not only makes it possible to store information more efficiently (by removing redundancies and repetition), but it also makes visible previously undiscovered relationships among disparate segments of data and permits efficient exploitation of those relationships.

Thus, server-side scripting languages and relational database management systems possess unique capabilities and advantages in their own right. Put them together, however, and the world really is your oyster: the combination of the two makes it possible to create innovative products and services that enhance the customer experience, simplify and speed business processes, and enable new Internet applications.

These are among the things you can do with server-side scripts and an RDBMS:

- Build a search engine that responds to user queries
- Record user input provided through web forms and save it for future reference
- Create web sites that dynamically update themselves with new content
- Manage a *blog* (or more than one)
- Process electronic payments and track customer orders
- Build customized bar graphs, pie charts, and other statistical reports from raw numeric data
- Carry out online surveys and polls, and create reports of the results

In recent years, one of the most popular combinations in this context has been the PHP scripting language and the MySQL RDBMS. The following section discusses these two products in detail, highlighting the capabilities and features of each, and illustrating just why they work so well together.

# The PHP Story

According to its official web site at **http://www.php.net/**, PHP is ". . . a widely used general-purpose scripting language that is especially suited for web development and can be embedded into HTML . . . the main goal of the language is to allow web developers to write dynamically generated web pages quickly." In English, what this means is that PHP is a programming language that makes it possible to incorporate sophisticated business logic into otherwise static web sites. The language is rapidly becoming the popular choice for data-driven web applications because of its wide support for different database systems.

Typically, PHP code is "embedded" inside a regular HTML document, and is recognized and executed by the web server when the document is requested through a browser. Because PHP is a full-featured programming language, you can code all manner of complex thingummies into your web pages using this technique; the server will execute your code and return the output to the browser in the format you specify. Because PHP code is executed on the server and not on the client, developers don't have to worry about browser-specific quirks that could cause the code to break (as commonly happens with JavaScript); PHP code works independently of the user's web browser.

Now, while this is fine and dandy, you might be wondering exactly what makes PHP so popular. After all, web developers have been creating Perl/CGI scripts to dynamically generate HTML pages for a long time, and the gradual adoption of W3C standards by modern browser vendors has made JavaScript far less susceptible to the vagaries of proprietary extensions. So what makes PHP the preferred web scripting language for developers around the world?

I've always thought the reason for PHP's popularity to be fairly simple: it has the unique distinction of being the only open-source server-side scripting language that's both easy to learn and extremely powerful to use. Unlike most modern server-side languages, PHP uses clear, simple syntax and delights in nonobfuscated code; this makes it easy to read and understand, and encourages rapid application development. And then, of course, there's cost and availability—PHP is available free of charge on the Internet, for a variety of platforms and architectures, including UNIX, Microsoft Windows, and Mac OS, as well as for most web servers.

Geeks will be happy to hear PHP is an interpreted language. Why is this good? Well, one advantage of an interpreted language is that it enables you to perform incremental, iterative development and testing without going through a compile-test-debug cycle each time you change your code. This can speed the development cycle drastically. A variety of data types, a powerful object-oriented engine, an extensive library of built-in functions, and support for most current web technologies and protocols complete the picture.

A bonus, especially for developers building web applications that must interface with a database, is PHP's support for the MySQL RDBMS, as well as other commercial database systems; this support is the primary draw for web developers dealing with data-heavy web applications, like content portals or electronic-commerce applications. The close-knit relationship between PHP and MySQL, both open-source projects, makes possible some powerful synergies. See the section "Sample Applications" at the end of this chapter for examples.

NOTE

*The sky's the limit . . . for a list of what you can do with PHP, see the PHP manual at* ***http://www.php.net/manual/en/intro-whatcando.php****.*

## History

The first version of PHP, PHP/FI, was developed by Rasmus Lerdorf as a means of monitoring page views for his online resumé and slowly started making a mark in mid 1995. This version of PHP had support for some basic functions, primarily the capability to handle form data and support for the mSQL database. PHP/FI 1.0 was followed by PHP/FI 2.0 and, in turn, quickly supplanted in 1997 by PHP 3.0.

PHP 3.0, developed by Andi Gutmans and Zeev Suraski, was where things started to get interesting. PHP 3.0 was a complete rewrite of the original PHP/FI implementation and it included support for a wider range of databases, including MySQL and Oracle. PHP 3.0's extensible architecture encouraged independent developers to begin creating their own language extensions, which served to increase the language's popularity in the developer community. Before long, PHP 3.0 was installed on hundreds of thousands of web servers, and more and more people were using it to build database-backed web applications.

PHP 4.0, which was released in 2003, used a new engine to deliver better performance, greater reliability and scalability, support for web servers other than Apache, and a host of new language features, including built-in session management and better OOP support. And, as if that wasn't enough, the current

version of PHP, PHP 5.0, offers a completely revamped object model that uses object handles for more consistent behavior when passing objects around, as well as abstract classes, destructors, multiple interfaces, and class type hints.

PHP 5.0 also includes better exception handling, a more consistent XML toolkit, improved MySQL support, and a better memory manager. So far, all these changes have conspired to make PHP 5.0 the best PHP release in the language's ten-year history . . . a fact amply illustrated by the April 2004 Netcraft survey, which shows PHP in use on over fifteen million web sites.

## Features

As a programming language for the Web, PHP is hard to ignore. Clean syntax, object-oriented fundamentals, an extensible architecture that encourages innovation, support for both current and upcoming technologies and protocols, and excellent database integration are just some of the reasons for the popularity it currently enjoys in the developer community.

### Simplicity

Because PHP uses a consistent and logical syntax, and because it comes with a clearly written manual, even novices find it easy to learn. In fact, the quickest way to learn PHP is to step through the manual's introductory tutorial, and then start looking at code samples off the Web. Within a few hours, you'll have learned the basics and will be confident enough to begin writing your own scripts. This adherence to the KISS (Keep It Simple, Stupid) principle has made PHP popular as a prototyping and rapid application development tool for web applications. PHP can even access *C* libraries and take advantage of program code written for this language, and the language is renowned for the tremendous flexibility it allows programmers in accomplishing specific tasks.

### Portability

With programming languages, *portability*—the ease with which a program can be made to work on different platforms—is an important factor. PHP users have little to fear here, because cross-platform development has been an important design goal of PHP since PHP 3.0. Today, PHP is available for a wide variety of platforms, including UNIX, Microsoft Windows, Mac OS, and OS/2. Additionally, because PHP code is interpreted and not compiled, PHP scripts written on one platform usually work as is on any other platform for which an interpreter exists. This means that developers can code on Windows and deploy on UNIX without any major difficulties.

## Speed

Out of the box, PHP scripts run faster than most other scripting languages, with numerous independent benchmarks putting the language ahead of competing alternatives like JSP, ASP.NET, and Perl. When PHP 4.0 was first released, it raised the performance bar with its completely new parsing engine. PHP 5.0 improves performance even further through the use of an optimized memory manager, and the use of object handles that reduce memory consumption and help applications run faster.

## Open Source

Possibly the best thing about PHP is that it's free—its source code is freely available on the Web, and developers can install and use it without paying licensing fees or investing in expensive hardware or software. Using PHP can thus significantly reduce the development costs of a software application, without compromising on either reliability or performance. The open-source approach also ensures faster bug fixes and quicker integration of new technologies into the core language, simply due to the much larger base of involved developers.

## Extensible

Keeping future growth in mind, PHP's creators built an extensible architecture that enables developers to easily add support for new technologies to the language through modular extensions. This extensibility keeps PHP fresh and always at the cutting edge of new technology. To illustrate this, consider what PHP lets you do through its add-on modules: dynamically create image, PDF, and SWF files; connect to IMAP and POP3 servers; interface with MySQL, Oracle, PostgreSQL, and SQLite databases; handle electronic payments; parse XML documents; and execute Perl, Java, and COM code through a PHP script. And as if all that wasn't enough, there's also an online repository of free PHP classes called PEAR, the PHP Extension and Application Repository, which provides a source of reusable, bug-free PHP components.

## XML and Database Support

Regardless of whether your web application sources its data from an XML file or a database, PHP has you covered. PHP 5.0 comes with an improved MySQL extension that enables you to take advantage of new features in the MySQL RDBMS (including subqueries, transactions, and referential integrity), and the language also supports DB2, PostgreSQL, Oracle, mSQL, MS-SQL, Informix,

Sybase, and SQLite. Alternatively, if it's XML you're after, PHP 5.0 offers a completely redesigned XML API built around the `libxml2` toolkit; this API supports SAX, DOM, and XSLT, as well as the new SimpleXML and SOAP extensions.


*The SimpleXML extension is particularly note-worthy—it takes all the pain out of parsing XML by representing an XML file as a PHP object. This object can then be processed using standard PHP constructs like loops and indexes.*

And speaking of databases. . . .

# The MySQL Story

If you've had even the slightest bit of experience with relational databases, you've probably heard of MySQL: It's a high-performance, multiuser relational database management system that is today the de facto standard for database-driven software applications, both on and off the Web.

Designed around three fundamental principles—speed, stability, and ease of use—and freely available under the GNU General Public License, MySQL has been dubbed "the world's most popular open-source database" by its parent company, MySQL AB. And with good reason. Official statistics reveal over five million sites are creating, using, and deploying MySQL-based applications, with more coming into the fold on a daily basis. You may even have heard of some of MySQL's customers: do the names Yahoo!, Google, Cisco, NASA, and HP sound familiar?

## History

The MySQL story hasn't always been about rocketing growth rates and high user satisfaction ratings, however. MySQL has an interesting history, with roots going back to 1979, when Michael "Monty" Widenius created a database system named UNIREG for the Swedish company TcX. UNIREG didn't work for TcX on account of performance issues, and so TcX began a search for alternatives. They tried mSQL, a competing DBMS created by David Hughes, but when that attempt also failed, a new approach was called for. Thus, Widenius decided to create a new database server customized to his specific requirements, but based on the mSQL API (to simplify porting applications between the two). That system, completed and released to a small group in May 1996, became MySQL 1.0.

## The Name Game

Wondering where the names MySQL and PHP came from? Well, the acronym PHP originally stood for "Personal Home Page Tools." When PHP 3.0 was released, it was changed into a recursive acronym meaning "PHP: Hypertext Preprocessor." More tidbits from PHP's history are available from the PHP web site, at **http://www.php.net/manual/en/history.php**.

MySQL's roots are not quite as clear. An entry in the MySQL manual suggests that even MySQL's developers don't know where the name came from: "The derivation of the name MySQL is not perfectly clear. Our base directory and a large number of our libraries and tools have had the prefix 'my' for well over ten years. However, Monty's daughter (some years younger) is also named My. Which of the two gave its name to MySQL is still a mystery, even for us." More MySQL history is available online at **http://www.linuxjournal.com/article.php?sid=3609** and **http://dev.mysql.com/doc/mysql/en/History.html**.

A few months later, MySQL 3.11 saw its first public release as a binary distribution for Solaris. Linux source and binaries followed shortly; an enthusiastic developer community and a friendly, GPL-based licensing policy took care of the rest. As MySQL grew in popularity, TcX became MySQL AB, a private company that today is the sole owner of the MySQL server source code and trademark. MySQL AB is responsible for maintenance, marketing, and further development of the MySQL database server and related products. Today, MySQL is available for a wide variety of platforms, including Linux, MacOS, and Windows.

# Features

MySQL's development history has always been characterized by a clear-eyed focus on the most important attributes of a good RDBMS: speed and stability. This has resulted in a system that outperforms most of its competitors without sacrificing reliability or ease of use, thereby gaining it a loyal base of developers, administrators, and users worldwide.

The following sections describe MySQL's most compelling features.

## Speed

In an RDBMS, speed—the time taken to execute a query and return the results to the caller—is everything. MySQL scores high on this parameter, with better performance than almost all its competitors, including commercial systems like

## What the Experts Say

In a February 2002 benchmark study published by *eWEEK* (at **http://www.eweek.com/article2/0,3959,293,00.asp**):

- MySQL was found to have the best performance and scalability, along with Oracle 9*i,* of the systems under comparison.
- MySQL was the easiest RDBMS to tune and optimize, along with SQL Server, of the systems under comparison.
- MySQL scaled efficiently at loads from 50 to 1,000 simultaneous users, with performance dropping only marginally once the 600-user limit had been crossed.

In a December 2003 study by Reasoning (at **http://www.reasoning.com/downloads/mysql.html**):

- MySQL code quality was found to rank higher than comparable commercial software, with a defect density six times lower.
- MySQL's development team was extremely responsive to defect reports, resolving them rapidly and efficiently.

Microsoft SQL Server and IBM DB2. This blazing performance is more the result of intelligent software design than luck: MySQL uses a fully multithreaded architecture; special optimizers for complex tasks like joins and indexing; a query cache, which improves performance without any special programming needed by the user; and the capability to use different storage engines on a per-table basis, so that users can mix and match different feature sets to squeeze the maximum performance out of the system.

### Reliability

When it comes to reliability, MySQL's creds are impeccable. The MySQL RDBMS has been tested and certified for use in high-volume, mission-critical applications by some of the world's largest organizations, including NASA, HP, and Yahoo! Because MySQL has deep roots in the open-source community, every new release is typically "battle-tested" by users all over the world, on different operating systems and in different operating conditions, to ensure that it

is completely bug-free before being certified for use. Further, every new release of MySQL first has to pass MySQL's in-house test suite, affectionately known as *crash-me* because its primary goal is to attempt to crash the system.

### Security

Security is an important concern when dealing with multiuser databases, and MySQL's developers have taken a great deal of care to ensure that MySQL is as secure as possible. MySQL comes with a sophisticated access control and privilege system to prevent unauthorized users from accessing the system. This system, implemented as a five-tiered privilege hierarchy, enables MySQL administrators to protect access to sensitive data using a combination of user- and host-based authentication schemes. Users can be restricted to performing operations only on specified databases or fields, and MySQL even makes it possible to control which types of queries a user can run, at database, table, or field level.

### Scalability and Portability

MySQL can handle extremely large and complex databases without too much of a drop in performance. Tables of several gigabytes containing hundreds of thousands of records are not uncommon, and the MySQL web site itself claims to use databases containing 50 million records. And once you've got your tables filled with data, you can move them from one platform to another without any difficulty—MySQL is available for both UNIX and non-UNIX operating systems, including Linux, Solaris, FreeBSD, OS/2, MacOS, and Windows 95, 98, Me, 2000, XP, and NT. It runs on a range of architectures, including Intel *x*86, Alpha, SPARC, PowerPC, and IA64, and supports many different hardware configurations, from low-end 386s to high-end Pentium machines.

### Ease of Use

Most commercial RDBMSs are intimidating, with cryptic command-line interfaces and hundreds of tunable parameters. Not this one, though—well aware that a complex interface adds to the total cost of ownership of an RDBMS, the MySQL

## The Tale of Sakila

The official MySQL logo is a dolphin named Sakila. According to the MySQL manual at **http://dev.mysql.com/doc/mysql/en/The_Original_MySQL_logo.html**, the dolphin was chosen because it is ". . . a smart, fast, and lean animal, effortlessly navigating oceans of data." Whoever said programmers didn't have a sense of humor?

development team has taken pains to make MySQL easy to use, administer, and optimize. A simple SQL command-line interface (SQL commands are covered in Chapters 9 to 11) is the primary user interface to the server; users with a more visual bent can, instead, use MySQL Control Center or MySQL Administrator, two GUI clients developed by MySQL AB for MySQL usage and administration. A number of other browser-based tools are also available, and the application is well supported by a detailed manual, a knowledgeable developer community, and some excellent books and tutorials.

## Compliance with Existing Standards

MySQL 4.0 supports most of the important features of the ANSI SQL-99 standard, with support for missing features slated to be added in future versions. MySQL also extends the ANSI standard with its own custom functions and data types designed to improve portability and provide users with enhanced functionality. On the internationalization front, MySQL 4.0 supports a number of important character sets (including Latin, Big5, and European character sets), with full Unicode support scheduled for future versions.

## Wide Application Support

MySQL exposes APIs to many different programming languages, thereby making it possible to write database-driven applications in the language of your choice. This book focuses specifically on using PHP with MySQL, but readers working with other programming languages will be pleased to hear that MySQL AB also provides native ODBC and JDBC drivers for the Microsoft Windows and Java platforms. Additionally, hooks to MySQL are available in C, C++, Perl, Python, and Tcl, to offer developers maximum freedom in designing MySQL-backed applications.

## Easy Licensing Policy

The MySQL RDBMS is licensed under the GPL, and users are free to download and modify the source code of the application to their needs, and to use it to power their applications free of cost. This licensing policy has only fuelled MySQL's popularity, creating an active and enthusiastic global community of MySQL developers and users. This community plays an active role in keeping MySQL ahead of its competition, both by crash-testing the software for reliability on millions of installations worldwide and by extending the core engine to stay abreast of the latest technologies and newest developments.

### To GPL or Not to GPL . . .

While the MySQL server and associated drivers are licensed under the GPL, you need to be aware of some caveats. You are permitted to use MySQL in your own software applications, free of charge, provided that you agree to license those applications also under the GPL, or any other MySQL AB-approved open-source license. MySQL may also be used without purchasing a license in a non-GPL application *provided* that the application is neither used for commercial purposes nor released for others to use. This enables end users to use MySQL for hobby sites without releasing their script source.

However, if your MySQL-powered application is not licensed under the GPL or equivalent licensing scheme, and you do intend to redistribute it (whether internally or externally), you are required to purchase a commercial license for MySQL.

A clear explanation of this "dual-licensing" model is available on the MySQL web site, at **http://www.mysql.com/products/licensing/**.

## PHP and MySQL: The Well-Matched Couple

As noted previously, one of the most important factors driving PHP's popularity over the last couple of years has been its support for a variety of databases, including MySQL, mSQL, Oracle, and Microsoft Access. By simplifying and streamlining database access, PHP enables developers to build complex data-driven web applications while enjoying short development cycles.

Support for MySQL has been available in PHP since version 3.*x,* and has gradually improved over subsequent releases. PHP 5.0 promises even better integration with the latest version of MySQL: the new MySQL extension in PHP 5.0 provides developers with both function- and object-oriented APIs to common MySQL functions, and includes support for new and upcoming MySQL features like transactions, stored procedures, and prepared statements.

NOTE

*The PHP 4.x release included a bundled version of the MySQL client libraries, which made it possible to access a MySQL server out-of-the-box. With PHP 5.0, this practice has been stopped and the MySQL client libraries are no longer bundled with the PHP release archive due to incompatibilities between the licensing terms for PHP and MySQL. PHP 5.0 users, therefore, need to download the MySQL client libraries separately and manually link them into PHP before they can begin using PHP's MySQL functions. The process is far less cumbersome than it sounds; see Chapter 2 for details.*

PHP's ease of use in the web arena, together with its tight integration with MySQL, has thus made it the preferred programming language for web-based, data-driven applications. Additionally, because both tools are available under open-source licenses, developers using PHP and MySQL can provide customers with huge savings on the licensing costs of other commercially licensed software, and also benefit from the tremendous amount of thought that PHP and MySQL developers have put into making sure that the two packages work together seamlessly and smoothly.

The applications that the PHP-MySQL combination have been used for range from the small to the large: content management systems for web portals, search engines, time- and resource-tracking tools, reporting and graphing tools, web-based personal information managers . . . the list goes on. In essence, if you can think of an application that uses (1) a database for storage of user data and (2) a browser as the primary user interface, it's a good chance the PHP-MySQL combination will work for you.

## Architecture

It's interesting, at this point, to see what the typical PHP and MySQL application development framework looks like. Usually, such applications are developed on the so-called "LAMP" (Linux, Apache, MySQL, and PHP) platform, wherein each component plays a specific and important role:

- Linux provides the base operating system (OS) and server environment.
- The Apache web server intercepts HTTP requests and either serves them directly or passes them on to the PHP interpreter for execution.
- The PHP interpreter parses and executes PHP code, and returns the results to the web server.
- The MySQL RDBMS serves as the data storage engine, accepting connections from the PHP layer and inserting, modifying, or retrieving data.

Figure 1-1 illustrates these components in action.

## An Open Invitation

The interesting thing about the LAMP platform, in case you haven't already noticed, is this: all the components are open-source!

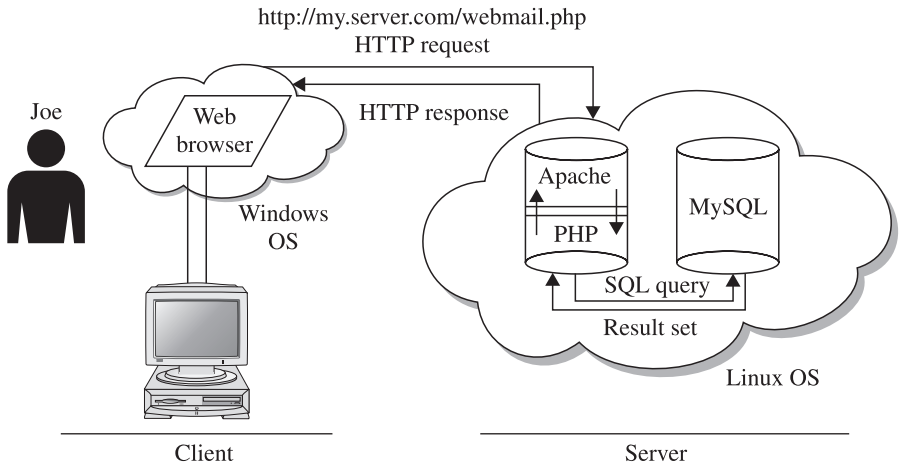


FIGURE 1-1 The LAMP development framework

NOTE

*Notice I said the development platform is "usually" LAMP. It's also possible to develop command-line PHP-MySQL applications, which run at the shell prompt and don't require a web server. Take a look at* ***http://www.php.net/manual/en/features.commandline.php*** *for more.*

Here's what's happening in Figure 1-1:

1. Joe pops open his web browser at home and types in the URL for his online Webmail client. After looking up the domain, Joe's browser (the client) sends an HTTP request to the corresponding server IP address.
2. The Apache web server handling HTTP requests for the domain receives the request and notes that the URI ends with a `.php` suffix. Because the server is programmed to automatically redirect all such requests to the PHP layer, it simply invokes the PHP interpreter and passes it the contents of the named file.

3. The PHP interpreter parses the file, executing the code in the special PHP tags. If the code includes database queries, the PHP interpreter opens a client connection to the MySQL RDBMS and executes them. Once the script interpreter has completed executing the script, it returns the result to the browser, cleans up after itself, and goes back into hibernation.
4. The results returned by the interpreter are transmitted to Joe's browser by the Apache server.

From the previous explanation, it should be clear that to get started building PHP and MySQL applications, your development environment must contain a web server (this doesn't always have to be Apache, although that's the most common) and working installations of PHP and MySQL. Chapter 2 discusses how to go about setting up this development environment, using both the Linux and Windows operating systems.

## Sample Applications

Here's a small sample of the types of applications that developers have used PHP and MySQL for:

- **phpMyAdmin** (**http://www.phpmyadmin.net/**) is a browser-based GUI to administer one or more MySQL database servers. One of the most popular applications on the SourceForge (**http://www.sourceforge.net/**) network, phpMyAdmin provides users with an HTML interface to insert, edit, and delete records; execute queries; view real-time MySQL performance statistics; import and export data; and manage user privileges.
- **phpAdsNew** (**http://www.phpadsnew.com/**) is a banner rotation and tracking system for web sites that enables site administrators to manage advertisers, display banners in rotation, and generate reports on views and clickthroughs.
- **Horde** (**http://www.horde.org/**) is a PHP-based application development framework that provides the foundation for a suite of web-based applications, including a Webmail client, a contact manager, a file manager, and a news client.

- **Midgard** (**http://www.midgard-project.org/**) is a template-based content management system (CMS) that provides a WYSIWYG interface for building web sites. It includes a web-based administrative interface to easily add and delete content, as well as support for content in multiple languages.
- **phpBB** (**http://www.phpbb.com/**) is a PHP/MySQL-based bulletin board package that enables web site administrators to quickly add unlimited discussion forums to their web site. phpBB includes a multitier privilege system, a powerful search engine support for multiple languages, private messaging, and public and private discussion rooms.
- **phpNuke** (**http://www.phpnuke.org/**) is an open-source portal-in-a-box solution that uses MySQL for data storage. phpNuke provides all the features most commonly found in a web portal, including user personalization, polls, bulletin boards, downloads, banner management, FAQs, a search engine, and more.
- **Drupal** (**http://www.drupal.org/**) is a content management system that enables users to publish and manage many different types of content. It supports news articles and content, polls, discussion forums, weblogs and download archives, and comes in handy if you need to jump-start a community-based web site or personal weblog.
- **phpGroupware** (**http://www.phpgroupware.org/**) is a PHP-based multiuser, multilanguage application suite. Usable through a web browser, it provides a calendar, to-do list, e-mail client, file manager, and address book.
- **Gallery** (**http://gallery.menalto.org**) uses PHP and MySQL to create a highly configurable digital photo archive, complete with automatic thumbnail creation, image captioning and editing, keyword search, and gallery-level authentication.

# Summary

This chapter provided a gentle introduction to the world of data-driven web applications, setting the stage with a description of how server-side scripting and databases work, and then proceeding to an overview of PHP and MySQL. It offered insight into the history and evolution of both tools, identified the core

features that have made them so popular with developers all over the world, and discussed some of their most common applications. Finally, it wrapped things up by identifying the essential components needed to build a PHP-MySQL development environment, together with an explanation of how the various components interact with each other.

The next chapter expands on this last section, guiding you through the process of obtaining, installing, and configuring the components of this application development environment—a necessary first step before you can begin building your own PHP and MySQL applications.