CHAPTER 1

Introducing PHP

In many ways the PHP language is representative of the stereotypical open source project, created to meet a developer's otherwise unmet needs and refined over time to meet the needs of its growing community. As a budding PHP developer, it's important you possess some insight into how the language has progressed, as it will help you to understand the language's strengths, and to some extent the reasoning behind its occasional idiosyncrasies.

Additionally, because the language is so popular, having some understanding of the differences between the versions—most notably versions 4, 5, and 6—will help when evaluating Web hosting providers and PHP-driven applications for your own needs.

To help you quickly get up to speed in this regard, this chapter will get you acquainted with PHP's features and version-specific differences. By the conclusion of this chapter, you'll learn the following:

- How a Canadian developer's Web page traffic counter spawned one of the world's most popular scripting languages
- What PHP's developers did to reinvent the language, making version 5 the best yet released
- Why PHP 6 is going to further propel PHP's adoption in the enterprise
- Which features of PHP attract both new and expert programmers alike

---

Note At the time of publication, PHP 6 was still a beta release, although many of the features are stable enough that they can safely be discussed throughout the course of the book. But be forewarned; some of these features could change before the final version is released.

History

The origins of PHP date back to 1995 when an independent software development contractor named Rasmus Lerdorf developed a Perl/CGI script that enabled him to know how many visitors were reading his online résumé. His script performed two tasks: logging visitor information, and displaying the count of visitors to the Web page. Because the Web as we know it today was still young at that time, tools such as these were nonexistent, and they prompted e-mails inquiring about Lerdorf's scripts. Lerdorf thus began giving away his toolset, dubbed Personal Home Page (PHP).

The clamor for the PHP toolset prompted Lerdorf to continue developing the language, with perhaps the most notable early change being a new feature for converting data entered in an HTML form into symbolic variables, encouraging exportation into other systems. To accomplish this, he opted to continue development in C code rather than Perl. Ongoing additions to the PHP toolset culminated in November 1997 with the release of PHP 2.0, or Personal Home Page/Form Interpreter

(PHP/FI). As a result of PHP's rising popularity, the 2.0 release was accompanied by a number of enhancements and improvements from programmers worldwide.

The new PHP release was extremely popular, and a core team of developers soon joined Lerdorf. They kept the original concept of incorporating code directly alongside HTML and rewrote the parsing engine, giving birth to PHP 3.0. By the June 1998 release of version 3.0, more than 50,000 users were using PHP to enhance their Web pages.

Development continued at a hectic pace over the next two years, with hundreds of functions being added and the user count growing in leaps and bounds. At the beginning of 1999, Netcraft (`http://www.netcraft.com/`), an Internet research and analysis company, reported a conservative estimate of a user base of more than 1 million, making PHP one of the most popular scripting languages in the world. Its popularity surpassed even the greatest expectations of the developers, as it soon became apparent that users intended to use PHP to power far larger applications than originally anticipated. Two core developers, Zeev Suraski and Andi Gutmans, took the initiative to completely rethink the way PHP operated, culminating in a rewriting of the PHP parser, dubbed the Zend scripting engine. The result of this work was in the PHP 4 release.

■ **Note** In addition to leading development of the Zend engine and playing a major role in steering the overall development of the PHP language, Suraski and Gutmans are cofounders of Zend Technologies Ltd. (`http://www.zend.com/`). Zend is the most visible provider of products and services for developing, deploying, and managing PHP applications. Check out the Zend Web site for more about the company's offerings, as well as an enormous amount of free learning resources.

PHP 4

On May 22, 2000, roughly 18 months after the first official announcement of the new development effort, PHP 4.0 was released. Many considered the release of PHP 4 to be the language's official debut within the enterprise development scene, an opinion backed by the language's meteoric rise in popularity. Just a few months after the major release, Netcraft estimated that PHP had been installed on more than 3.6 million domains.

PHP 4 added several enterprise-level improvements to the language, including the following:

Improved resource handling: One of version 3.X's primary drawbacks was scalability. This was largely because the designers underestimated how rapidly the language would be adopted for large-scale applications. The language wasn't originally intended to run enterprise-class Web sites, and continued interest in using it for such purposes caused the developers to rethink much of the language's mechanics in this regard.

Object-oriented support: Version 4 incorporated a degree of object-oriented functionality, although it was largely considered an unexceptional and even poorly conceived implementation. Nonetheless, the new features played an important role in attracting users used to working with traditional object-oriented programming (OOP) languages. Standard class and object development methodologies were made available in addition to features such as object overloading and run-time class information. A much more comprehensive OOP implementation has been made available in version 5 and is introduced in Chapter 6.

Native session-handling support: HTTP session handling, available to version 3.X users through the third-party package PHPLIB (`http://phplib.sourceforge.net`) was natively incorporated into version 4. This feature offers developers a means for tracking user activity and preferences with unparalleled efficiency and ease. Chapter 18 covers PHP's session-handling capabilities.

Encryption: The MCrypt (`http://mcrypt.sourceforge.net`) library was incorporated into the default distribution, offering users both full and hash encryption using encryption algorithms including Blowfish, MD5, SHA1, and TripleDES, among others. Chapter 21 delves into PHP's encryption capabilities.

ISAPI support: ISAPI support offered users the ability to use PHP in conjunction with Microsoft's IIS Web server. Chapter 2 shows you how to install PHP on both the IIS and Apache Web servers.

Native COM/DCOM support: Another bonus for Windows users is PHP 4's ability to access and instantiate COM objects. This functionality opened up a wide range of interoperability with Windows applications.

Native Java support: In another boost to PHP's interoperability, support for binding to Java objects from a PHP application was made available in version 4.0.

Perl Compatible Regular Expressions (PCRE) library: The Perl language has long been heralded as the reigning royalty of the string parsing kingdom. The developers knew that powerful regular expression functionality would play a major role in the widespread acceptance of PHP and opted to simply incorporate Perl's functionality rather than reproduce it, rolling the PCRE library package into PHP's default distribution (as of version 4.2.0). Chapter 9 introduces this important feature in great detail and offers a general introduction to the often confusing regular expression syntax.

In addition to these features, literally hundreds of functions were added to version 4, greatly enhancing the language's capabilities. Many of these functions are discussed throughout the course of the book.

PHP 4 represented a gigantic leap forward in the language's maturity, offering new features, power, and scalability that swayed an enormous number of burgeoning and expert developers alike. Yet the PHP development team wasn't content to sit on their hands for long and soon set upon another monumental effort, one that could establish the language as the 800-pound gorilla of the Web scripting world: PHP 5.

PHP 5

Version 5 was yet another watershed in the evolution of the PHP language. Although previous major releases had enormous numbers of new library additions, version 5 contains improvements over existing functionality and adds several features commonly associated with mature programming language architectures:

Vastly improved object-oriented capabilities: Improvements to PHP's object-oriented architecture is version 5's most visible feature. Version 5 includes numerous functional additions such as explicit constructors and destructors, object cloning, class abstraction, variable scope, and interfaces, and a major improvement regarding how PHP handles object management. Chapters 6 and 7 offer thorough introductions to this topic.

Try/catch exception handling: Devising custom error-handling strategies within structural programming languages is, ironically, error-prone and inconsistent. To remedy this problem, version 5 supports exception handling. Long a mainstay of error management in many languages, such as C++, C#, Python, and Java, exception handling offers an excellent means for standardizing your error-reporting logic. This convenient methodology is introduced in Chapter 8.

Improved XML and Web Services support: XML support is now based on the libxml2 library, and a new and rather promising extension for parsing and manipulating XML, known as SimpleXML, has been introduced. In addition, a SOAP extension is now available. In Chapter 20, these two extensions are introduced, along with a number of slick third-party Web Services extensions.

Native support for SQLite: Always keen on choice, the developers added support for the powerful yet compact SQLite database server (`http://www.sqlite.org/`). SQLite offers a convenient solution for developers looking for many of the features found in some of the heavyweight database products without incurring the accompanying administrative overhead. PHP's support for this powerful database engine is introduced in Chapter 22.

■ **Note** The enhanced object-oriented capabilities introduced in PHP 5 resulted in an additional boost for the language: it opened up the possibility for cutting-edge frameworks to be created using the language. Chapter 25 introduces you to one of the most popular frameworks available today, namely the Zend Framework (`http://framework.zend.com/`).

With the release of version 5, PHP's popularity hit what was at the time a historical high, having been installed on almost 19 million domains, according to Netcraft. PHP was also by far the most popular Apache module, available on almost 54 percent of all Apache installations, according to Internet services consulting firm E-Soft Inc. (`http://www.securityspace.com/`).

PHP 6

At press time, PHP 6 was in beta and scheduled to be released by the conclusion of 2007. The decision to designate this a major release (version 6) is considered by many to be a curious one, in part because only one particularly significant feature has been added— Unicode support. However, in the programming world, the word *significant* is often implied to mean *sexy* or *marketable*, so don't let the addition of Unicode support overshadow the many other important features that have been added to PHP 6. A list of highlights is found here:

- **Unicode support:** Native Unicode support has been added.
- **Security improvements:** A considerable number of security-minded improvements have been made that should greatly decrease the prevelance of security-related gaffes that to be frank aren't so much a fault of the language, but are due to inexperienced programmers running with scissors, so to speak. These changes are discussed in Chapter 2.
- **New language features and constructs:** A number of new syntax features have been added, including, most notably, a 64-bit integer type, a revamped foreach looping construct for multidimensional arrays, and support for labeled breaks. Some of these features are discussed in Chapter 3.

At press time, PHP's popularity was at a historical high. According to Netcraft, PHP has been installed on more than 20 million domains. According to E-Soft Inc., PHP remains the most popular Apache module, available on more than 40 percent of all Apache installations.

So far, this chapter has discussed only version-specific features of the language. Each version shares a common set of characteristics that play a very important role in attracting and retaining a large user base. In the next section, you'll learn about these foundational features.

■ **Note** You might be wondering why versions 4, 5, and 6 were mentioned in this chapter. After all, isn't only the newest version relevant? While you're certainly encouraged to use the latest stable version, versions 4 and 5 remain in widespread use and are unlikely to go away anytime soon. Therefore having some perspective regarding each version's capabilities and limitations is a good idea, particularly if you work with clients who might not be as keen to keep up with the bleeding edge of PHP technology.

General Language Features

Every user has his or her own specific reason for using PHP to implement a mission-critical application, although one could argue that such motives tend to fall into four key categories: practicality, power, possibility, and price.

Practicality

From the very start, the PHP language was created with practicality in mind. After all, Lerdorf's original intention was not to design an entirely new language, but to resolve a problem that had no readily available solution. Furthermore, much of PHP's early evolution was not the result of the explicit intention to improve the language itself, but rather to increase its utility to the user. The result is a language that allows the user to build powerful applications even with a minimum of knowledge. For instance, a useful PHP script can consist of as little as one line; unlike C, there is no need for the mandatory inclusion of libraries. For example, the following represents a complete PHP script, the purpose of which is to output the current date, in this case one formatted like `September 23, 2007`:

```
<?php echo date("F j, Y");?>
```

Don't worry if this looks foreign to you. In later chapters, the PHP syntax will be explained in great detail. For the moment just try to get the gist of what's going on.

Another example of the language's penchant for compactness is its ability to nest functions. For instance, you can effect numerous changes to a value on the same line by stacking functions in a particular order. The following example produces a string of five alphanumeric characters such as `a3jh8`:

```
$randomString = substr(md5(microtime()), 0, 5);
```

PHP is a *loosely typed* language, meaning there is no need to explicitly create, typecast, or destroy a variable, although you are not prevented from doing so. PHP handles such matters internally, creating variables on the fly as they are called in a script, and employing a best-guess formula for automatically typecasting variables. For instance, PHP considers the following set of statements to be perfectly valid:

```
<?php
    $number = "5";          // $number is a string
    $sum = 15 + $number;    // Add an integer and string to produce integer
    $sum = "twenty";        // Overwrite $sum with a string.
?>
```

PHP will also automatically destroy variables and return resources to the system when the script completes. In these and in many other respects, by attempting to handle many of the administrative aspects of programming internally, PHP allows the developer to concentrate almost exclusively on the final goal, namely a working application.

Power

PHP developers have more than 180 libraries at their disposal, collectively containing well over 1,000 functions. Although you're likely aware of PHP's ability to interface with databases, manipulate form information, and create pages dynamically, you might not know that PHP can also do the following:

- Create and manipulate Adobe Flash and Portable Document Format (PDF) files
- Evaluate a password for guessability by comparing it to language dictionaries and easily broken patterns

- Parse even the most complex of strings using the POSIX and Perl-based regular expression libraries
- Authenticate users against login credentials stored in flat files, databases, and even Microsoft's Active Directory
- Communicate with a wide variety of protocols, including LDAP, IMAP, POP3, NNTP, and DNS, among others
- Tightly integrate with a wide array of credit-card processing solutions

And this doesn't take into account what's available in the PHP Extension and Application Repository (PEAR), which aggregates hundreds of easily installable open source packages that serve to further extend PHP in countless ways. You can learn more about PEAR in Chapter 11. In the coming chapters you'll learn about many of these libraries and several PEAR packages.

Possibility

PHP developers are rarely bound to any single implementation solution. On the contrary, a user is typically fraught with choices offered by the language. For example, consider PHP's array of database support options. Native support is offered for more than 25 database products, including Adabas D, dBase, Empress, FilePro, FrontBase, Hyperwave, IBM DB2, Informix, Ingres, InterBase, mSQL, Microsoft SQL Server, MySQL, Oracle, Ovrimos, PostgreSQL, Solid, Sybase, Unix dbm, and Velocis. In addition, abstraction layer functions are available for accessing Berkeley DB–style databases. Several generalized database abstraction solutions are also available, among the most popular being PDO (`http://www.php.net/pdo`) and MDB2 (`http://pear.php.net/package/MDB2`). Finally, if you're looking for an object relational mapping (ORM) solution, projects such as Propel (`http://propel.phpdb.org/trac/`) should fit the bill quite nicely.

PHP's flexible string-parsing capabilities offer users of differing skill sets the opportunity to not only immediately begin performing complex string operations but also to quickly port programs of similar functionality (such as Perl and Python) over to PHP. In addition to more than 85 string-manipulation functions, both POSIX- and Perl-based regular expression formats are supported.

Do you prefer a language that embraces procedural programming? How about one that embraces the object-oriented paradigm? PHP offers comprehensive support for both. Although PHP was originally a solely functional language, the developers soon came to realize the importance of offering the popular OOP paradigm and took the steps to implement an extensive solution.

The recurring theme here is that PHP allows you to quickly capitalize on your current skill set with very little time investment. The examples set forth here are but a small sampling of this strategy, which can be found repeatedly throughout the language.

Price

PHP is available free of charge! Since its inception, PHP has been without usage, modification, and redistribution restrictions. In recent years, software meeting such open licensing qualifications has been referred to as *open source* software. Open source software and the Internet go together like bread and butter. Open source projects such as Sendmail, Bind, Linux, and Apache all play enormous roles in the ongoing operations of the Internet at large. Although open source software's free availability has been the point most promoted by the media, several other characteristics are equally important if not more so:

Free of licensing restrictions imposed by most commercial products: Open source software users are freed of the vast majority of licensing restrictions one would expect of commercial counterparts. Although some discrepancies do exist among license variants, users are largely free to modify, redistribute, and integrate the software into other products.

Open development and auditing process: Although not without incidents, open source software has long enjoyed a stellar security record. Such high-quality standards are a result of the open development and auditing process. Because the source code is freely available for anyone to examine, security holes and potential problems are rapidly found and fixed. This advantage was perhaps best summarized by open source advocate Eric S. Raymond, who wrote "Given enough eyeballs, all bugs are shallow."

Participation is encouraged: Development teams are not limited to a particular organization. Anyone who has the interest and the ability is free to join the project. The absence of member restrictions greatly enhances the talent pool for a given project, ultimately contributing to a higher-quality product.

Summary

Understanding more about the PHP language's history and widely used versions is going to prove quite useful as you become more acquainted with the language and begin seeking out both hosting providers and third-party solutions. This chapter satisfied that requirement by providing some insight into PHP's history and an overview of version 4, 5, and 6's core features.

In Chapter 2, prepare to get your hands dirty, as you'll delve into the PHP installation and configuration process, and learn more about what to look for when searching for a Web hosting provider. Although readers often liken these types of chapters to scratching nails on a chalkboard, you can gain a lot from learning more about this process. Much like a professional cyclist or race car driver, the programmer with hands-on knowledge of the tweaking and maintenance process often holds an advantage over those without by virtue of a better understanding of both the software's behaviors and quirks. So grab a snack and cozy up to your keyboard—it's time to build.