CHAPTER 1

An Introduction to PHP

The past five years have been fantastic in terms of the explosive growth of the Internet and the new ways in which people are able to communicate with one another. Spearheading this phenomenon has been the World Wide Web (WWW), with thousands of new sites being launched daily and consumers being consistently offered numerous outstanding services via this new communications medium. With this exploding market has come a great need for new technologies and developers to learn these technologies. Chances are that if you are reading this paragraph, you are one of these Web developers or are soon to become one. Regardless of your profession, you've picked this book up because you've heard of the great new technology called *PHP*.

This chapter introduces the PHP language, discusses its history and capabilities, and provides the basic information you need to begin developing PHP-enabled sites. Several examples are provided throughout, hopefully serving to excite you about what PHP can offer you and your organization. You will learn how to install and configure the PHP software on both Linux/UNIX and Windows machines, and you will learn how to embed PHP in HTML. At the conclusion of the chapter, you will be ready to begin delving into the many important aspects of the PHP language. So light the fire, turn on your favorite jazz album, and curl up on the lazyboy; you are about to learn what will be one of the most exciting additions to your resume: PHP programming.

An Abbreviated History

PHP set its roots in 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 resume. His script performed two duties: logging visitor information and displaying the count of visitors to the Web page. Because the WWW as we know it today was still so young at that time, tools such as these were nonexistent, and they prompted emails inquiring about Lerdorf's scripts. Lerdorf thus began giving away his toolset, dubbed *Personal Home Page* (PHP), or *Hypertext Preprocessor*.

The clamor for the PHP toolset prompted Lerdorf to begin developing additions to PHP, one of which converted data entered in an HTML form into symbolic variables that allowed for their export to other systems. To accomplish this, he opted to continue development in C code rather than Perl. This addition to the

existing PHP toolset resulted in PHP 2.0, or PHP-FI (Personal Home Page—Form Interpreter). This 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 formed. They kept the original concept of incorporating code directly alongside HTML and rewrote the parsing engine, giving birth to PHP 3.0. By the 1997 release of version 3.0, over 50,000 users were using PHP to enhance their Web pages.

> **NOTE** *1997 also saw the change of the words underlying the PHP abbreviation from Personal Home Page to Hypertext Preprocessor.*

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 onset of 1999, Netcraft (`http://www.netcraft.com`) reported a conservative estimate of a user base surpassing 1,000,000, making PHP one of the most popular scripting languages in the world.

Early 1999 saw the announcement of the upcoming PHP 4.0. Although one of PHP's strongest features was its proficiency at executing scripts, the developers had not intended that large-scale applications were going to be built using PHP. Thus they set out to build an even-more robust parsing engine, better known as Zend (`http://www.zend.com`). Development continued rapidly, culminating in the May 22, 2000, release of PHP 4.0.

In addition to the Zend processor, Zend technologies, based in Israel, offers the Zend optimizer, which increases even further the performance benefits of the Zend parsing engine. Available for download free of charge, the benchmarks have shown that the optimizer can result in a 40 to 100 percent overall performance gain. Check out the Zend site for more information.

At the time of this writing, according to Netcraft (`http://www.netcraft.com`), PHP is installed on over 3.6 million domains, making it one of the most popular scripting languages in the world. The future of PHP indeed looks bright, as major Web sites and personal users alike continue to embrace the product.

PHP is best summarized as an embedded server-side Web-scripting language that provides developers with the capability to quickly and efficiently build dynamic Web applications. PHP bears a close resemblance, both syntactically and grammatically, to the C programming language, although developers haven't been shy to integrate features from a multitude of languages, including Perl, Java, and C++. Several of these valuable borrowed features include regular expression parsing, powerful array-handling capabilities, an object-oriented methodology, and vast database support.

For writing applications that extend beyond the traditional, static methodology of Web page development (that is, HTML), PHP can also serve as a valuable tool for creating and managing dynamic content, embedded directly beside the

likes of JavaScript, Stylesheets, WML (Wireless Markup Language) and many other useful languages. Providing hundreds of predefined functions, PHP is capable of handling just about anything a developer can dream of. Extensive support is offered for graphic creation and manipulation, mathematical calculations, ecommerce, and burgeoning technologies such as Extensible Markup Language (XML), open database connectivity (ODBC), and Macromedia Shockwave. This vast range of capabilities eliminates the need for the tedious and costly integration of several third-party modules, making PHP the tool of choice for developers worldwide.

One of the main strengths of PHP is the fact that because it can be embedded directly alongside HTML code, there is no need to write a program that has many commands just to output the HTML. HTML and PHP can be used interchangeably as needed, working alongside one another in unison. With PHP, we can simply do the following:

```
<html>
<title><? print "Hello world!"; ?></title>
</html>
```

And `Hello world!` will be displayed in the Web page title bar. Interestingly, the single line print statement is enclosed in what are commonly known as PHP's escape characters (`<?…?>`) is a complete program. No need for lengthy prefacing code or inclusion of libraries; the only required code is what is needed to get the job done!

Of course, in order to execute a PHP script, you must first install and configure the PHP software on your server. This process is explained in "Downloading and Installing PHP/Apache," later in this chapter. Immediately preceding that section are a few excerpts from prominent users testifying to the power of PHP, followed by a detailed synopsis of the language and its history. However, before diving into the installation process, take a moment to read more about the characteristics of PHP that make it such a powerful language. This is the subject of the next section, aptly titled "Characteristics of PHP."

Characteristics of PHP

As you may have realized, the PHP language revolves around the central theme of practicality. PHP is about providing the programmer with the necessary tools to get the job done in a quick and efficient fashion. Five important characteristics make PHP's practical nature possible:

- Familiarity
- Simplicity

- Efficiency
- Security
- Flexibility

One final characteristic makes PHP particularly interesting: it's free!

Familiarity

Programmers from many backgrounds will find themselves already accustomed to the PHP language. Many of the language's constructs are borrowed from C and Perl, and in many cases PHP code is almost indistinguishable from that found in the typical C or Pascal program. This minimizes the learning curve considerably.

Simplicity

A PHP script can consist of 10,000 lines or one line: whatever you need to get the job done. There is no need to include libraries, special compilation directives, or anything of the sort. The PHP engine simply begins executing the code after the first escape sequence (`<?`) and continues until it passes the closing escape sequence (`?>`). If the code is syntactically correct, it will be executed exactly as it is displayed.

Efficiency

Efficiency is an extremely important consideration for working in a multiuser environment such as the WWW. PHP 4.0 introduced resource allocation mechanisms and more pronounced support for object-oriented programming, in addition to session management features. Reference counting has also been introduced in the latest version, eliminating unnecessary memory allocation.

Security

PHP provides developers and administrators with a flexible and efficient set of security safeguards. These safeguards can be divided into two frames of reference: system level and application level.

System-Level Security Safeguards

PHP furnishes a number of security mechanisms that administrators can manipulate, providing for the maximum amount of freedom and security when PHP is properly configured. PHP can be run in what is known as *safe mode*, which can

limit users' attempts to exploit the PHP implementation in many important ways. Limits can also be placed on maximum execution time and memory usage, which if not controlled can have adverse affects on server performance. Much as with a cgi-bin folder, administrators can also place restrictions on the locations in which users can view and execute PHP scripts and use PHP scripts to view guarded server information, such as the passwd file.

Application-Level Security Safeguards

Several trusted data encryption options are supported in PHP's predefined function set. PHP is also compatible with many third-party applications, allowing for easy-integration with secure ecommerce technologies. Another advantage is that the PHP source code is not viewable through the browser because the script is completely parsed before it is sent back to the requesting user. This benefit of PHP's server-side architecture prevents the loss of creative scripts to users at least knowledgeable enough to execute a 'View Source'.

Security is such an important issue that this book contains an entire chapter on the subject. Please read Chapter 16, "Security," for a thorough accounting of PHP's security features.

Flexibility

Because PHP is an embedded language, it is extremely flexible towards meeting the needs of the developer. Although PHP is generally touted as being used in conjunction solely with HTML, it can also be integrated alongside languages like JavaScript, WML, XML, and many others. Additionally, as with most other mainstream languages, wisely planned PHP applications can be easily expanded as needed.

Browser dependency is not an issue because PHP scripts are compiled entirely on the server side before being sent to the user. In fact, PHP scripts can be sent to just about any kind of device containing a browser, including cell phones, personal digital assistant (PDA) devices, pagers, laptops, not to mention the traditional PC. People who want to develop shell-based applications can also execute PHP from the command line.

Since PHP contains no server-specific code, users are not limited to a specific and perhaps unfamiliar Web server. Apache, Microsoft IIs, Netscape Enterprise Server, Stronghold, and Zeus are all fair game for PHP's server integration. Because of the various platforms that these servers operate on, PHP is largely platform independent, available for such platforms as UNIX, Solaris, FreeBSD, and Windows 95/98/NT.

Finally, PHP offers access to external components, such as Enterprise Java Beans and Win32 COM objects. These newly added features put PHP in the big league, truly enabling developers to scale PHP projects upward and outward as need be.

Free

The open source development strategy has gained considerable notoriety in the software industry. The prospect of releasing source code to the masses has resulted in undeniably positive outcomes for many projects, perhaps most notably Linux, although the success of the Apache project has certainly been a major contributor in proving the validity of the open source ideal. The same holds true for the developmental history of PHP, as users worldwide have been a huge factor in the advancement of the PHP project.

PHP's embracing of this open source strategy result in great performance gains for users, and the code is available free of charge. Additionally, an extremely receptive user community numbering in the thousands acts as "customer support," providing answers to even the most arcane questions in popular online discussion groups.

The next section, "User Affirmations," provides testimonies from three noted industry professionals. Each provides keen insight into why they find PHP such an appealing technology.

User Affirmations

> *"We have for a long time had a personal contact to some of the PHP developers and exchanged a lot of emails with them in the past. When the PHP developers have had any problems with MySQL related issues we have always been ready to help them solve them. We have also on some occasions added new features into MySQL just to get the PHP integration better. The result of this work is that MySQL works extremely well with PHP and we will ensure that it keeps that way!"*
>
> Michael "Monty" Widenius, MySQL Developer
> `http://www.mysql.com`

> *"FAST used PHP to implement mp3.lycos.com for a number of reasons. The most important was time to market; PHP really lets you speed up the development. Another reason was speed, we went from 0 to 1.4 million page impressions in one day, and PHP coped just fine with this. The third reason was of course that I knew that if I found bugs in PHP during this '"stress test",' I could fix them myself since PHP is open source."*
>
> Stig Bakken, FAST Search & Transfer ASA
> `http://www.fast.no`

> *"I've used PHP from the early days when it was PHP/FI 1.x. I loved having the ability to process forms and customize my pages on the fly with such an easy-to-use language. As my company's needs have evolved, so has PHP.*

Today, PHP is extremely feature rich. We rely on it for just about every custom web site we develop, including 32bit.com and DevShed.com. We even use it at InfoWest to manage our customer service, account management and port monitoring.

PHP's evolution and acceptance is a textbook example of a successful open source project. Open-mindedness, community contribution, and a well-managed code-base have helped build PHP into a success few commercial entities have been able to emulate. I look forward to the future of PHP. I encourage any budding web developer to give PHP a spin. Like me, you may never want to give it up."

Randy Cosby
President, nGenuity, Inc.
DevShed (`http://www.devshed.com`)

An Introductory Example

Consider the example shown in Listing 1-1, which illustrates just how easily PHP can be integrated alongside HTML:

Listing 1-1: Dynamic PHP page creation

```
<?
// Set a few variables
$site_title = "PHP Recipes";
$bg_color = "white";
$user_name = "Chef Luigi";
?>

<html>
<head>
<title><? print $site_title; ?></title>
</head>
<body bgcolor="<? print $bg_color; ?>" >
<?
// Display an intro. message with date and user name.
print "
PHP Recipes | ".date("F d, Y")." <br>
Greetings, $user_name! <br>
";
?>
</body>
</html>
```

Figure 1-1 shows how the script appears when it is executed in the browser.

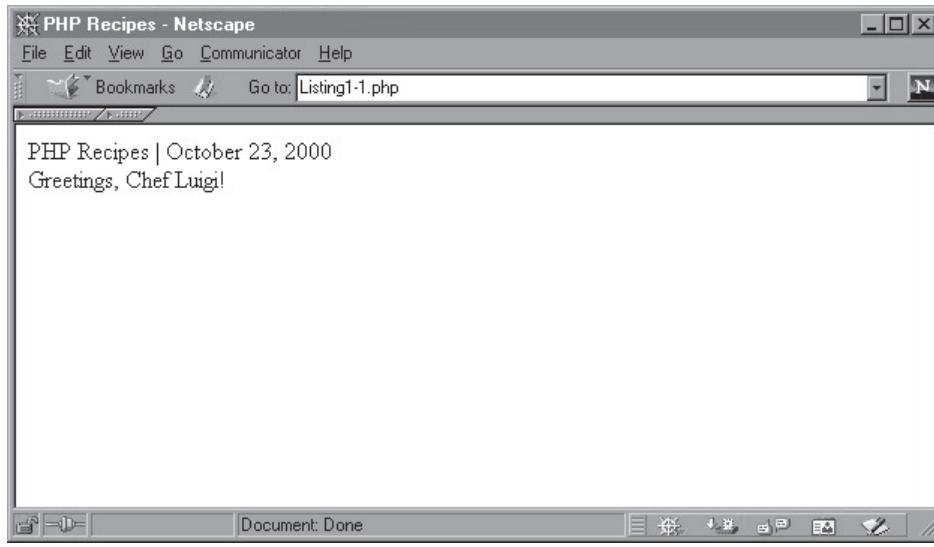

Figure 1-1. The script is executed in the browser.

Not too shabby, huh? I'm sure many a reader's mind is already churning with possibilities. However, before delving further into scripting issues, chances are you may need to install and configure PHP on your machine. This is the subject of the next few sections.

Downloading PHP/Apache

Before you proceed, I recommend that you take some time to download, install, and configure PHP and a Web server on your machine. Although PHP is compatible with a wide variety of Web servers, I'll assume that you will be using Apache, partly because it is currently the Web's most popular Web server and partly because it is the one most widely used with PHP. Regardless, the general installation process will not differ widely between Web servers.

You can download the PHP distribution from the official PHP site or from one of its many worldwide mirror sites. Go to `http://www.php.net` for the most recently updated mirror list. From here, you can download PHP in one of two formats:

- WIN32 Binary
- Source code

The Win32 binary is for Windows 95/98/NT/2000 users. While it is also possible to compile the source code on the Windows platform, for the large majority of users this won't be necessary. However, if you insist on doing so (incidentally, a process that is not discussed within this book), you'll need a recent Visual C++ compiler for doing so. Check out http://www.php.net/version4/win32build.php for more information on this process. The Win32 binary installation process is detailed later in this chapter.

For non-Windows users, you'll need to build the source code. While many beginners may shudder at this thought, it is actually a rather simple process, as you'll soon learn. For those of you interested to know whether or not PHP is offered in RPM (RedHat Package Manager) distribution format; it is, although these RPMs are not available via the official PHP site. Check the discussion groups (some of which are listed at the end of this chapter) for more information regarding distribution locations and instructions. The generalized build process is detailed later in this chapter.

Proceed to `http://www.php.net` and download the distribution that best suits your needs. Download times will vary with your connection type and speed. Additionally, the documentation is available for download. I strongly recommend downloading the most recent version.

> **TIP** *PHP 4.0.3 was the current stable version at the time of printing of this book. Of course, this version number is due to change along with the continued development of the PHP package. I recommend always downloading the most recent stable version of the product.*

If you haven't yet installed the Apache server, you will want to download the latest stable version of that as well. These packages are at http://www.apache.org/dist/binaries/, which contains directories for a plethora of operating systems. Download the one that is specific to your needs. Providing instructions regarding PHP configuration specifics for every available platform and Web server is out of the scope of this book. Therefore, I will concentrate on the Apache server. Regardless of the Web server you intend to use, I strongly recommend reading through the configuration sections later in this chapter to gain some insight into the generalized configuration issues that you may encounter.

Installation of new software can sometimes prove to be a daunting process for newcomers. However, the PHP developers have taken extra steps to make PHP installation relatively easy. The following sections highlight the steps you should take to install and configure PHP on both the non-Windows and the Win32 platforms.

NOTE *In later chapters I'll introduce the MySQL database server, using this popular product as the basis for illustrating Web/database integration. In order to experiment with these examples, you'll need to install the MySQL package, available at* `http://www.mysql.com`. *Like PHP, MySQL is available for both non-Windows and Windows platforms. Although I defer to the MySQL documentation due to its thorough installation instructions, you may be interested in taking a moment to read through the initial pages of Chapter 11, "Databases," for an introduction of the MySQL database server.*

Installation and Configuration

At this point, I'll assume that you have successfully downloaded PHP and Apache. The next step is deciding how you would like to install the distribution.For non-Windows machines, there three different ways to do so: CGI binary, static Apache module, and the dynamic Apache module. As a non-Windows user, chances are you will not want to build PHP as a CGI binary. Furthermore, there are several advantages to building PHP as a server module, therefore I'll concentrate solely on building PHP both as a static and a dynamic module. As it relates to installation, the main difference between the two is that any subsequent changes to the PHP static module will require the recompilation of both Apache and PHP, while changes to the PHP dynamic module only require the subsequent recompilation of just PHP and not the server.

For Windows machines, PHP can be installed as either a CGI binary or as a static Apache module. In this case, I'll concentrate upon the CGI binary, since a Windows-user might be more prone to use a Web server other than Apache, like Microsoft's Internet Information Server or Microsoft's Personal Web Server. The CGI version can easily be integrated into these servers. Although I illustrate the PHP/Apache Windows installation process, this process is very similar to that which would be used for the above-mentioned Web servers as well.

NOTE *Recall that PHP4 comes with support for a wide variety of Web servers, including AOL Server, Netscape Enterprise Server, Microsoft IIs, Zeus, and more. However, I will keep the installation process limited to that relating to Apache. For detailed instructions regarding how to install PHP with these other servers, check out the PHP documentation at* `http://www.php.net`.