Part I

Getting Up and Running with PHP

1

Introducing PHP

Welcome to the world of PHP, one of the Web's most popular programming languages. According to Netcraft (`www.netcraft.com`), PHP was running on more than 20 million Web servers in July 2007 (`http://www.php.net/usage.php`). At the time of writing, it's the fourth most popular programming language in the world according to TIOBE (`http://www.tiobe.com/index.php/content/paperinfo/tpci/`), beaten only by Java, C, and C++. With the introduction of version 5.3, there's never been a better time to learn PHP.

In this chapter you:

❑ Get a gentle introduction to PHP in general, and the new features of PHP 5.3 in particular

❑ Learn what PHP is, what it can be used for, and how it stacks up against other dynamic Web technologies

❑ Take a look at the history of PHP, so you can see how it has evolved over the years, from its humble beginnings to the rich Web development framework it is today

What Is PHP?

PHP is a programming language for building dynamic, interactive Web sites. As a general rule, PHP programs run on a Web server, and serve Web pages to visitors on request. One of the key features of PHP is that you can embed PHP code within HTML Web pages, making it very easy for you to create dynamic content quickly.

What exactly does the phrase "dynamic, interactive Web sites" mean? A *dynamic* Web page is a page whose contents can change automatically each time the page is viewed. Contrast this with a *static* Web page, such as a simple HTML file, which looks the same each time it's displayed (at least until the page is next edited). Meanwhile, an *interactive* Web site is a site that responds to input from its visitors. A Web forum is a good example — users can post new messages to the forum, which are then displayed on the site for all to see. Another simple example is a "contact us" form,

where visitors interact with the page by filling out and sending a form, which is then emailed to the Webmaster.

> *PHP stands for PHP: Hypertext Preprocessor, which gives you a good idea of its core purpose: to process information and produce hypertext (HTML) as a result. (Developers love recursive acronyms, and PHP: Hypertext Preprocessor is a good example of one.)*

PHP is a *server-side scripting language*, which means that PHP scripts, or programs, usually run on a Web server. (A good example of a *client-side* scripting language is JavaScript, which commonly runs within a Web browser.) Furthermore, PHP is an *interpreted* language — a PHP script is processed by the PHP engine each time it's run.

The process of running a PHP script on a Web server looks like this:

1. A visitor requests a Web page by clicking a link, or typing the page's URL into the browser's address bar. The visitor might also send data to the Web server at the same time, either using a form embedded in a Web page, or via AJAX (Asynchronous JavaScript And XML).

2. The Web server recognizes that the requested URL is a PHP script, and instructs the PHP engine to process and run the script.

3. The script runs, and when it's finished it usually sends an HTML page to the Web browser, which the visitor then sees on their screen.

The interesting stuff happens when a PHP script runs. Because PHP is so flexible, a PHP script can carry out any number of interesting tasks, such as:

❑ Reading and processing the contents of a Web form sent by the visitor

❑ Reading, writing, and creating files on the Web server

❑ Working with data in a database stored on the Web server

❑ Grabbing and processing data from other Web sites and feeds

❑ Generating dynamic graphics, such as charts and manipulated photos

And finally, once it's finished processing, it can send a customized HTML Web page back to the visitor.

In this book you learn how to write scripts to do all of these, and more.

All these great features mean that you can use PHP to create practically any type of dynamic Web application you can dream of. Common examples of PHP scripts include:

❑ Web forums that allow visitors to post messages and discuss topics

❑ Search engines that let people search the contents of a Web site or database

❑ Straw poll scripts that enable visitors to vote in polls and surveys

❑ Content management systems and blogs, which enable Webmasters to create sites easily with minimal technical knowledge

❑ Webmail applications, allowing people to send and receive email using their Web browser

❑ Online stores, allowing shoppers to purchase products and services over the Internet

Web scripting is certainly the mainstay of PHP's success, but it's not the only way to use the language. Command-line scripting — which was introduced in PHP 4 — is another popular application of PHP. (This topic is covered in Appendix D at the end of this book.) Client-side graphical user interface application development using GTK (the GNOME ToolKit) is another.

Why Use PHP?

One of the best things about PHP is the large number of Internet service providers (ISPs) and Web hosting companies that support it. Today hundreds of thousands of developers are using PHP, and it's not surprising that there are so many, considering that several million sites are reported to have PHP installed.

Another great feature of PHP is that it's *cross-platform* — you can run PHP programs on Windows, Linux, FreeBSD, Mac OS X, and Solaris, among others. What's more, the PHP engine can integrate with all common Web servers, including Apache, Internet Information Server (IIS), Zeus, and lighttpd. This means that you can develop and test your PHP Web site on one setup, then deploy it on a different type of system without having to change much of your code. Furthermore, it's easy to move your PHP Web site onto another server platform, if you ever need to.

How does PHP compare with other common Web programming technologies? At the time of writing, the following technologies are prevalent:

❑ **ASP (Active Server Pages):** This venerable Microsoft technology has been around since 1997, and was one of the first Web application technologies to integrate closely with the Web server, resulting in fast performance. ASP scripts are usually written in VBScript, a language derived from BASIC. This contrasts with PHP's more C-like syntax. Although both languages have their fans, I personally find that it's easier to write structured, modular code in PHP than in VBScript.

❑ **ASP.NET:** This is the latest incarnation of ASP, though in fact it's been rebuilt from the ground up. It's actually a framework of libraries that you can use to build Web sites, and you have a choice of languages to use, including C#, VB.NET (Visual Basic), and J# (Java). Because ASP.NET gives you a large library of code for doing things like creating HTML forms and accessing database tables, you can get a Web application up and running very quickly. PHP, although it has a very rich standard library of functions, doesn't give you a structured framework to the extent that ASP.NET does. On the other hand, plenty of free application frameworks and libraries are available for PHP, such PEAR (discussed later in this book) and the Zend Framework. Many would argue that C# is a nicer, better-organized language to program in than PHP, although C# is arguably harder to learn. Another advantage of ASP.NET is that C# is a compiled language, which generally means it runs faster than PHP's interpreted scripts (although PHP compilers are available).

ASP and ASP.NET have a couple of other disadvantages compared to PHP. First of all, they have a commercial license, which can mean spending additional money on server software, and hosting is often more expensive as a result. Secondly, ASP and ASP.NET are fairly heavily tied to the Windows platform, whereas the other technologies in this list are much more cross-platform.

❑ **Perl:** Perl was one of the first languages used for creating dynamic Web pages, initially through the use of CGI scripting and, later, integrating tightly into Web servers with technologies like the Apache mod_perl module and ActivePerl for IIS. Though Perl is a powerful scripting language, it's harder to learn than PHP. It's also more of a general-purpose language than PHP, although Perl's CPAN library includes some excellent modules for Web development.

❑ **Java:** Like Perl, Java is another general-purpose language that is commonly used for Web application development. Thanks to technologies like JSP (JavaServer Pages) and servlets, Java is a great platform for building large-scale, robust Web applications. With software such as Apache Tomcat, you can easily build and deploy Java-based Web sites on virtually any server platform, including Windows, Linux, and FreeBSD. The main downside of Java compared to PHP is that it has quite a steep learning curve, and you have to write a fair bit of code to get even a simple Web site going (though JSP helps a lot in this regard). In contrast, PHP is a simpler language to learn, and it's quicker to get a basic Web site up and running with PHP. Another drawback of Java is that it's harder to find a Web hosting company that will support JSP, whereas nearly all hosting companies offer PHP hosting.

❑ **Python:** Conceived in the late 1980s, Python is another general-purpose programming language that is now commonly used to build dynamic Web sites. Although it doesn't have much in the way of Web-specific features built into the language, many useful modules and frameworks, such as Zope and Django, are available that make building Web applications relatively painless. Many popular sites such as Google and YouTube are built using Python, and Python Web hosting is starting to become much more common (though it's nowhere near as common as PHP hosting). You can even build and host your Python apps on Google's server with the Google App Engine. Overall, Python is a very nice language, but PHP is currently a lot more popular, and has a lot more built-in functionality to help with building Web sites.

❑ **Ruby:** Like Python, Ruby is another general-purpose language that has gained a lot of traction with Web developers in recent years. This is largely due to the excellent Ruby on Rails application framework, which uses the Model-View-Controller (MVC) pattern, along with Ruby's extensive object-oriented programming features, to make it easy to build a complete Web application very quickly. As with Python, Ruby is fast becoming a popular choice among Web developers, but for now, PHP is much more popular.

❑ **ColdFusion:** Along with ASP, Adobe ColdFusion was one of the first Web application frameworks available, initially released back in 1995. ColdFusion's main selling points are that it's easy to learn, it lets you build Web applications very quickly, and it's really easy to create database-driven sites. An additional plus point is its tight integration with Flex, another Adobe technology that allows you to build complex Flash-based Web applications. ColdFusion's main disadvantages compared to PHP include the fact that it's not as popular (so it's harder to find hosting and developers), it's not as flexible as PHP for certain tasks, and the server software to run your apps can be expensive. (PHP and Apache are, of course, free and open source.)

In summary, PHP occupies something of a middle ground when it comes to Web programming languages. On the one hand, it's not a general-purpose language like Python or Ruby (although it can be used as one). This makes PHP highly suited to its main job: building Web sites. On the other hand, PHP doesn't have a complete Web application framework like ASP.NET or Ruby on Rails, meaning that you're left to build your Web sites "from the ground up" (or use add-on extensions, libraries, and frameworks).

However, this middle ground partly explains the popularity of PHP. The fact that you don't need to learn a framework or import tons of libraries to do basic Web tasks makes the language easy to learn and use. On the other hand, if you need the extra functionality of libraries and frameworks, they're there for you.

Another reason for PHP's popularity is the excellent — and thorough — online documentation available through `www.php.net` and its mirror sites.

In the past, PHP has been criticized for the way it handled a number of things — for example, one of its main stumbling blocks was the way in which it implemented object support. However, since version 5, PHP has taken stock of the downfalls of its predecessors and, where necessary, has completely rewritten the way in which it implements its functionality. Now more than ever, PHP is a serious contender for large-scale enterprise developments as well as having a large, consolidated base of small- to medium-sized applications.

The Evolution of PHP

Although PHP only started gaining popularity with Web developers around 1998, it was created by Rasmus Lerdorf way back in 1994. PHP started out as a set of simple tools coded in the C language to replace the Perl scripts that Rasmus was using on his personal home page (hence the original meaning of the "PHP" acronym). He released PHP to the general public in 1995, and called it PHP version 2.

In 1997, two more developers, Zeev Suraski and Andi Gutmans, rewrote most of PHP and, along with Rasmus, released PHP version 3.0 in June 1998. By the end of that year, PHP had already amassed tens of thousands of developers, and was being used on hundreds of thousands of Web sites.

For the next version of PHP, Zeev and Andi set about rewriting the PHP core yet again, calling it the "Zend Engine" (basing the name "Zend" on their two names). The new version, PHP 4, was launched in May 2000. This version further improved on PHP 3, and included session handling features, output buffering, a richer core language, and support for a wider variety of Web server platforms.

Although PHP 4 was a marked improvement over version 3, it still suffered from a relatively poor object-oriented programming (OOP) implementation. PHP 5, released in July 2004, addressed this issue, with private and protected class members; final, private, protected, and static methods; abstract classes; interfaces; and a standardized constructor/destructor syntax.

What's New in PHP 5.3

Most of the changes introduced in version 5.3 are relatively minor, or concern advanced topics outside of the scope of this beginner-level book. In the following sections you take a brief look at some of the more significant changes that might concern you, particularly if you're moving up from PHP 5.2 or earlier.

Namespaces

The biggest new feature in PHP 5.3 is support for namespaces. This handy feature lets you avoid naming clashes across different parts of an application, or between application libraries.

Namespaces bear some resemblance to folders on a hard disk, in that they let you keep one set of function, class and constant names separate from another. The same name can appear in many namespaces without the names clashing.

PHP 5.3's namespace features are fairly comprehensive, and include support for sub-namespaces, as well as namespace aliases. You'll learn more about using namespaces in Chapter 20.

The goto Operator

PHP 5.3 also introduces a `goto` operator that you can use to jump directly to a line of code within the same file. (You can only jump around within the current function or method.) For example:

```
goto jumpToHere;
echo 'Hello';

jumpToHere:
echo 'World';
```

Use `goto` sparingly — if at all — as it can make your code hard to read, as well as introduce thorny programming errors if you're not careful. However, it can be useful in some situations, such as breaking out of deeply nested loops.

Nowdoc Syntax

In PHP 5.3 you can quote strings using nowdoc syntax, which complements the existing heredoc syntax. Whereas heredoc-quoted strings are parsed — replacing variable names with values and so on — nowdoc-quoted strings are untouched. The nowdoc syntax is useful if you want to embed a block of PHP code within your script, without the code being processed at all.

Find out more about nowdoc and heredoc syntax in Chapter 5.

Shorthand Form of the Ternary Operator

The ternary operator — introduced in Chapter 4 — lets your code use the value of one expression or another, based on whether a third expression is `true` or `false`:

```
( expression1 ) ? expression2 : expression3;
```

In PHP 5.3 you can now omit the second expression in the list:

```
( expression1 ) ?: expression3;
```

This code evaluates to the value of `expression1` if `expression1` is `true`; otherwise it evaluates to the value of `expression3`.

Advanced Changes

If you're familiar with earlier versions of PHP, or with other programming languages, then you might be interested in some of the new advanced features in PHP 5.3. As well as the simpler changes just described, PHP 5.3 includes support for powerful programming constructs such as late static bindings, which add a lot of flexibility to static inheritance when working with classes, and closures, which allow for true anonymous functions. It also introduces an optional garbage collector for cleaning up circular references. (Since these are advanced topics, they won't be covered any further in this book.)

Some of the nastier aspects of earlier PHP versions — namely Register Globals, Magic Quotes and Safe Mode — are deprecated as of version 5.3, and will be removed in PHP 6. Attempting to use these features results in an `E_DEPRECATED` error (the `E_DEPRECATED` error level is also new to 5.3).

You can view a complete list of the changes in PHP 5.3 at `http://docs.php.net/migration53`.

Summary

In this chapter you gleaned an overview of PHP, one of the most popular Web programming languages in use today. You learned what PHP is, and looked at some of the types of Web applications you can build using it. You also explored some of the alternatives to PHP, including:

❑ ASP and ASP.NET

❑ Perl

❑ Java

❑ Python

❑ Ruby and Ruby on Rails

❑ ColdFusion

With each alternative, you looked at how it compares to PHP, and learned that some technologies are better suited to certain types of dynamic Web sites than others.

In the last sections of the chapter, you studied the history of PHP and explored some of the more significant new features in version 5.3, such as namespaces and the `goto` operator. Armed with this overview of the PHP language, you're ready to move on to Chapter 2 and write your first PHP script!