What Are Dynamic Web Sites?

Dynamic Web sites are flexible and potent creatures, more accurately described as *applications* than merely sites. Dynamic Web sites

- Respond to different parameters (for example, the time of day or the version of the visitor's Web browser)
- Have a "memory," allowing for user registration and login, e-commerce, and similar processes
- Almost always have HTML forms, so that people can perform searches, provide feedback, and so forth
- Often have interfaces where administrators can manage the site's content
- Are easier to maintain, upgrade, and build upon than statically made sites

There are many technologies available for creating dynamic Web sites. The most common are ASP.NET (Active Server Pages, a Microsoft construct), JSP (Java Server Pages), ColdFusion, Ruby on Rails, and PHP. Dynamic Web sites don't always rely on a database, but more and more of them do, particularly as excellent database applications like MySQL are available at little to no cost.

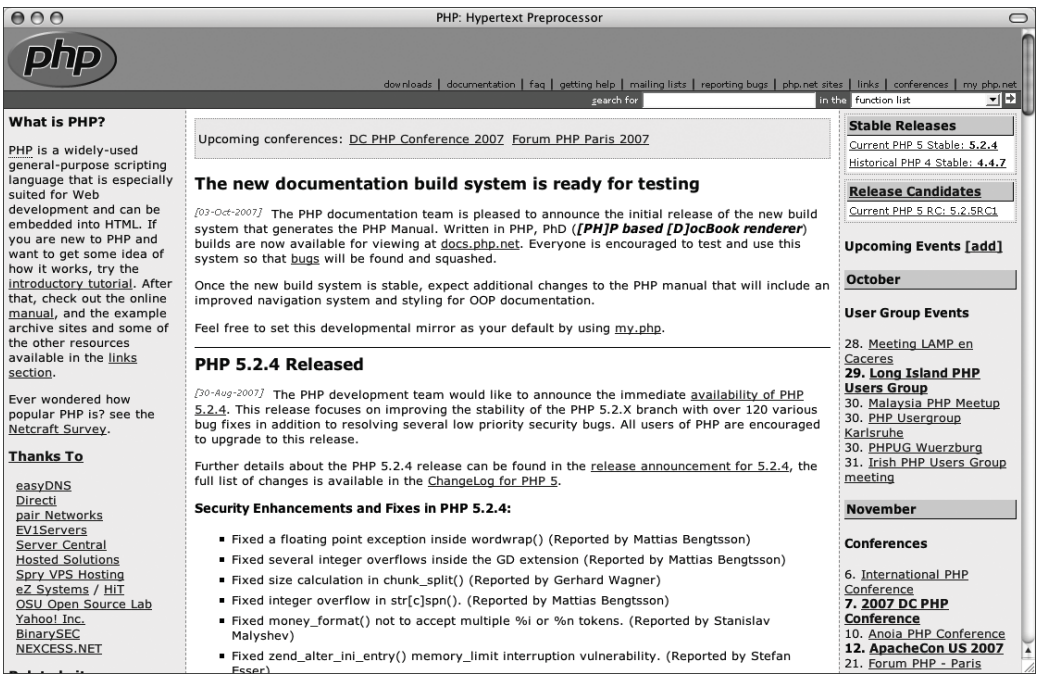

Figure i.1 The home page for PHP.

What is PHP?

PHP originally stood for "Personal Home Page" as it was created in 1994 by Rasmus Lerdorf to track the visitors to his online résumé. As its usefulness and capabilities grew (and as it started being used in more professional situations), it came to mean "PHP: Hypertext Preprocessor."

According to the official PHP Web site, found at `www.php.net` (**Figure i.1**), PHP is a "widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML." It's a long but descriptive definition, whose meaning I'll explain.

Starting at the end of that statement, to say that PHP can be embedded into HTML means that you can take a standard HTML page, drop in some PHP wherever you need it, and end up with a dynamic result. This attribute makes PHP very approachable for anyone that's done even a little bit of HTML work.

Also, PHP is a scripting language, as opposed to a programming language: PHP was designed to write Web scripts, not standalone applications (although, with some extra effort, you can now create applications in PHP). PHP scripts run only after an event occurs—for example, when a user submits a form or goes to a URL.

I should add to this definition that PHP is a *server-side*, *cross-platform* technology, both descriptions being important. Server-side refers to the fact that everything PHP does occurs on the server. A Web server application, like Apache or Microsoft's IIS (Internet Information Services), is required and all PHP scripts must be accessed through a URL (`http://`-something). Its cross-platform nature means that PHP runs on most operating systems, including Windows, Unix (and its many variants), and Macintosh. More important, the PHP scripts written on one server will normally work on another with little or no modification.

At the time the book was written, PHP was at version 5.2.4, with version 4.4.7 still being maintained. Support for version 4 is being dropped, though, and it's recommended that everyone use at least version 5 of PHP. This edition of this book actually focuses on version 6 of PHP, to be released in late 2007 or in 2008. If you're still using version 4, you really should upgrade. If that's not in your plans, then please grab the second edition of this book instead. If you're using PHP 5, either the second or this edition of the book will work for you. In this edition, I will make it clear which features and functions are PHP 6–specific.

What's new in PHP 6

Because of the planned extinction of PHP 4, many users and Web hosting companies will likely make a quick transition from PHP 4 to PHP 5 to PHP 6. To discuss what's new in PHP 6, I'll start with the even bigger differences between PHP 4 and 5.

PHP 5, like PHP 4 before it, is a major new development of this popular programming language. The most critical changes in PHP 5 involve object-oriented programming (OOP).Those changes don't really impact this book, as OOP isn't covered (I do so in my book *PHP 5 Advanced: Visual QuickPro Guide*). With respect to this book, the biggest change in PHP 5 is the addition of the Improved MySQL Extension, which is used to communicate with MySQL. The Improved MySQL Extension offers many benefits over the older MySQL extension and will be used exclusively.

The big change in PHP 6 is support for Unicode, which is to say that PHP can now handle characters in every language in the world. This is huge, and it's also one of the reasons it's taken a while to release PHP 6. What this means in terms of programming is covered in Chapter 14, "Making Universal Sites." The information in that chapter is also used in Chapter 15, "Example—Message Board." Beyond Unicode support, PHP 6 cleans up a lot of garbage that was left in PHP 5 even though the recommendation was not to use such things. The two biggest removals are the "Magic Quotes" and "register globals" features.

Why use PHP?

Put simply, when it comes to developing dynamic Web sites, PHP is better, faster, and easier to learn than the alternatives. What you get with PHP is excellent performance, a tight integration with nearly every database available, stability, portability, and a nearly limitless feature set due to its extendibility. All of this comes at no cost (PHP is open source) and with a very manageable learning curve. PHP is one of the best marriages I've ever seen between the ease with which beginning programmers can start using it and the ability for more advanced programmers to do everything they require.

Finally, the proof is in the pudding: PHP has seen an exponential growth in use since its inception, overtaking ASP as the most popular scripting language being used today. It's the most requested module for Apache (the most-used Web server), and by the time this book hits the shelves, PHP will be on nearly 25 million domains.

Of course, you might assume that I, as the author of a book on PHP (several, actually), have a biased opinion. Although not nearly to the same extent as PHP, I've also developed sites using Java Server Pages (JSP), Ruby on Rails (RoR), and ASP.NET. Each has its pluses and minuses, but PHP is the technology I always return to. You might hear that it doesn't perform or scale as well as other technologies, but Yahoo! handles over 3.5 billion hits per day using PHP (yes, *billion*). You might also wonder how secure PHP is. But security isn't in the language; it's in how that language is used. Rest assured that a complete and up-to-date discussion of all the relevant security concerns is provided by this book!

How PHP works

As previously stated, PHP is a server-side language. This means that the code you write in PHP sits on a host computer called a *server*. The server sends Web pages to the requesting visitors (you, the client, with your Web browser).

When a visitor goes to a Web site written in PHP, the server reads the PHP code and then processes it according to its scripted directions. In the example shown in **Figure i.2**, the PHP code tells the server to send the appropriate data—HTML code—to the Web browser, which treats the received code as it would a standard HTML page.

This differs from a static HTML site where, when a request is made, the server merely sends the HTML data to the Web browser and there is no server-side interpretation occurring (**Figure i.3**). Because no server-side action is required, you can run HTML pages in your Web browser without using a server at all.

To the end user and their Web browser there is no perceptible difference between what `home.html` and `home.php` may look like, but how that page's content was created will be significantly different.

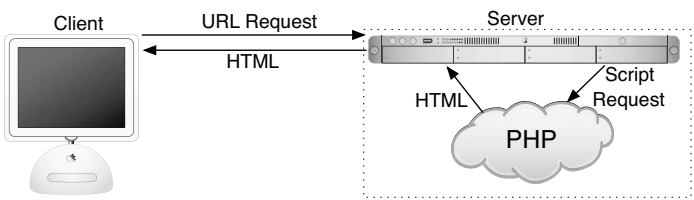

Figure i.2 How PHP fits into the client/server model when a user requests a Web page.

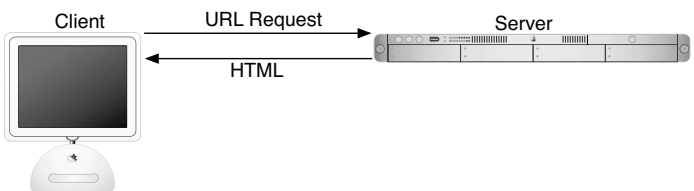

Figure i.3 The client/server process when a request for a static HTML page is made.