PHP From an IT Manager's Perspective

By Tobias Ratschiller on May 30th, 2000.

What have sprint.ca, livebid.amazon.com, xoom.com, and mp3.lycos.com in common? All employ one of the Web's hottest server side technologies: PHP, the PHP Hypertext Preprocessor. As this article shows, PHP over the years had the opportunity to strengthen its core base and to integrate more features, and provides today a base that can easily stand out in the increasingly crowded server-side application development platform market.

Introduction to PHP

What have sprint.ca, livebid.amazon.com, xoom.com, and mp3.lycos.com in common? All employ one of the Web's hottest server side technologies: PHP, the PHP Hypertext Preprocessor.

While the success of Open Source software like Linux or Apache has been documented extensively throughout all mainstream media, the rise of PHP has gone largely unnoticed. Still, the Web scripting language PHP is the most popular module for the Apache Web server, according to a E-Soft survey (http://www.e-softinc.com/survey/). Netcraft studies have found that PHP is in use on over 6% of all Web domains in the world (see http://www.netcraft.com/survey). That is an incredible market penetration for a rather specialized product. This popularity continues to rise exponentially, with the new version 4.0 just around the corner. Increasingly, this is being reflected in traditional media: By May 2000, more than 20 books about PHP have been published in different languages, with more in the pipeline. Commercial players are beginning to join the bandwagon: PHP is included with Web servers, for example C2's Stronghold, and Linux distributions. A new company, Zend Technologies, has been formed to provide commercial add-ons and support for PHP. A long list of large-scale Web sites employ PHP, as well as hundreds of thousands small to medium Web sites. Enough reasons to take a closer look at PHP.

What Is PHP

PHP is a programming language, used on the server to create dynamic Web pages. The principles of this technology are similar to Allaire's ColdFusion, Mod_Perl, Sun's JSP, or Microsoft's ASP, which all address one problem: As Web sites and intranets get larger and more complex, static HTML files hit their limits. Today, the Web is an interactive, transaction-oriented business platform, featuring advanced Web-based applications such as online commerce, business information systems, and collaborative computing environments. To develop such applications, you need a technology to generate dynamic content for Web pages.

Traditionally, developers have used CGI (Common Gateway Interface) scripts for interaction with users, querying databases, and so forth. However, because CGI scripts are separate software programs, which get executed as stand-alone processes whenever a user requests the script, scalability was lacking - CGI scripts could quickly consume all available memory and CPU power. Enter PHP: Rasmus Lerdorf, a then Toronto-based IT-consultant, developed it in late 1994 as a quick hack to embed macros into HTML pages, to avoid the forking of external programs. When he decided to make his project open source, it proved to be popular, and users started to contribute to it. In 1997, Zeev Suraski and Andi Gutmans, two developers from Israel, rewrote the core engine of PHP, the language parser, and made a complete programming language out of a pet project. The current rewrite, PHP 4.0, is again much cleaner, especially with complex scripts and when used in business environments.

Architecture Overview

Working with PHP leads to a three-tier architecture in such a coherent way that many developers don't even notice it. In the first tier there is a thin client - translated to the world of Web applications, this would be the browser. The middle tier (application server) is obviously PHP and the Web server as host application, while the third tier consists of a database system.

PHP scripts are often embedded in the HTML code of page, and then get parsed on the server-side - the browser sees plain HTML only. A typical Hello-World script looks like this:

```
<html>
  <? print("Hello World!");?>
</html>
```

This is the method that novice developers find the easiest to work with. Larger and more complex applications usually go other routes, to enforce a cleaner separation of layout and application logic. When embedding the script directly in HTML, average, HTML-literate Web designers cannot easily edit the contents of the page without being familiar with the scripting language used. PHP offers a variety of libraries to work with page templates, which solve this problem, and introduce an efficient development methodology and simplify maintenance. This way, developers focus on the application logic, and designers can change the layout of a dynamic page without involving the developer or interfering with the program logic. This translates into faster application development, and makes maintenance tasks easier by dividing them into content and logic tasks, which can be handled by separate team members.

PHP needs not be used for Web development exclusively. It can also be compiled as stand-alone script interpreter, and handles simple system administration tasks as well. For example, you could use a small PHP script to send daily statistics from your e-commerce application. In version 4.0, the language core engine, the Zend parser, has been abstracted enough to be embeddable in other technologies. Rumours go that is planned to integrate PHP as stored prodecure language into the popular MySQL database system. Seeing the dynamic evolution of PHP, it is only logical to expect the language to grow into other scenarios - why not use PHP as a macro processor in a word processor?

New Challenges

As you've seen, the new opportunities created by the Web bring new challenges to IT organizations building the applications. Choosing the right technology is critical to the success of any Web application development project. The main challenges are an increased demand on productivity, connectivity, portability, and performance.

PHP Meets Development Challenges

The first advantage of PHP was one common to many Open Source projects: It simply delivered, while other technologies were still vapor ware. PHP pre-dates ASP, Mod_Perl, and ColdFusion by at least 12 months. Over the years, PHP had the opportunity to strenghten its core base and to integrate more features, and provides today a base that can easily stand out in the increasingly crowded server-side application development platform market, as the following points will show.

Faster Time to Market

The development time of Web applications is measured in days and weeks - dinosaur projects spanning multiple years belong to an era which many Web developers don't even remember. IT managers and developers are struggling to keep up with this pressure, and demand high productivity from their development environments.

Perl is a general scripting language, Java is a full-fledged, complex programming language, Visual Basic Script and JSscript have been post-integrated into ASP - PHP, on the other hand, was built with the needs of Web developers in mind. In Web application development, there is no itch you can't scratch with PHP. Unlike other cumbersome, overhead-laden approaches, PHP is lightweight and focused on the Web - where it can solve complex problem scenarios quicker and more easily than comparable technologies.

The syntax and grammatical structure of PHP resembles the C programming language with the complexity (for example, memory management, pointers, and strong typing) taken out. The developers of PHP aren't hesitant to integrate the best features of other languages, though, so you'll find elements of Perl and Java in PHP as well. For programmers familiar with C, Perl, or Java, it is a matter of days to get acquainted with PHP. Thanks to the excellent reference manual, anyone of your developers can probably produce small database enabled applications after just one afternoon.

Abundant Connectivity

And there's a lot to explore. For example, PHP implements native interface to a wide variety of database engines, from Open Source systems like MySQL or PostgreSQL to commercial products like Oracle, SQL Server, DB2, and many more. The native database access offers better performance and tighter control than layered access methods such as ODBC (which is still available for databases not supported natively). Especially on the Web, a fast response time is crucial for the success of applications.

Besides databases, PHP supports most current Internet standards: IMAP, FTP, POP, XML, WDDX, LDAP, NIS, and SNMP - to list only a few of the acronyms which will inevitably get a twinkle into your developers eyes. For corporate and IT needs, this simply means that PHP is able to talk to different standards and technologies with ease: All from one common tool set, without the need for expensive third party modules.

Leveraging Enterprise Logic

I hear you say already: "Nice features - but unfortunately, we've all of our business logic already developed with Foo", where Foo stands for Java classes, Enterprise Java Beans, or COM components, depending on your corporate bias. Indeed, others have had these thoughts, and since version 4.0, PHP supports direct access to Java objects on any system with a Java Virtual Machine available, as well as Distributed COM on Windows. Multiple Web applications can reuse the same components. This enables you to keep your business logic in separate components, and use PHP for what it does best, and where it outperforms its alternatives: Web application development. This means a significantly lower Cost of Ownership: It enables business to leverage existing technology and develop new applications in an easier way.

Portability

When mentioning Java, one if its key features comes to mind instantly: portability. Up until version 3.0, PHP could be integrated only as module into the Apache Web server, or run as separate CGI program, which would eliminate many of PHP's speed benefits. In version 4.0, however, the Web server interface (Server API, or SAPI) has been abstracted, and PHP now integrates well with different Web servers: iPlanet/Netscape Enterprise Server, IIS, Apache, Zeus, fhttpd, and so forth. Platform independence has always been an advantage of PHP: It runs on all popular Unix platforms, including Linux, on Windows, on MacOS, and even on OS/2.

Portability is the key to scalable applications. You can run the same application on a low-scale Linux box and on a high-end Solaris machine, without the need to worry about platform-specific features. Also, PHP

interfaces transparently with clustering solutions from simple Round Robin IP clusters to advanced Cisco solutions.

The broad platform support can be directly attributed to the fact that PHP is distributed with full source code. Anyone with the necessary skills can port PHP to a new operating system. The resulting modifications to the core system can then be contributed back to the community.

Open Source

Open Source software in general means a number of significant advantages for the corporate IT infrastructure. Because the full source code is available, it can be inspected in thorough security audits. If third parties find security issues, they're usually fixed within hours or days. If no one is going to do it, you can assign your own personnel to it - with the full code in your hands, you're no longer dependent on external software manufacturers.

Then there's the community. Free help is available from mailing lists, newsgroups, and IRC channels. The PHP core developers participate in these support forums, and provide developers with top-level advice - usually within hours. I'm certain that more, commercial support options will be available in a very short time.

Open Source brings with it that rough edges are corrected promptly, and that the overall strategy is unbureaucratically adjusted to new requirements. For example, originally no one had thought that PHP would be used in most sophisticated business scenarios, and only version 4.0 is really prepared for this environment.

Performance

Therefore, while the speed of PHP 3.0 was sufficient for the average, medium-sized Web application, scalability for advanced applications could be a problem. The plain, out-of-the-box 4.0 version is already multiple times faster than PHP 3.0, introducing a performance boost, which will make some clustering systems superfluous. Plus, there's the Zend Optimizer, a free add-on module from Zend Technologies. It performs on-the-fly code optimizations to enhance the running speed of PHP applications. An application that uses the Zend Optimizer, typically executes 40% to 100% faster than one without.

Pointers

Official PHP Homepage: http://www.php.net

Zend Homepage: http://www.zend.com

Conclusion

Considering the vastly growing amount of servers on which PHP is now running, taking a look at the huge step PHP made from version 3.0 to version 4.0, and seeing its mature and up-to-date base, PHP has surely become what it has been aiming at: A great tool for rapid development of stable and fast Web applications. If your business relies on Web applications, you should certainly consider PHP. Gone are the days, when a business could post static pages on a Web site - today, sophisticated Web applications demand that companies use technologies that can provide them with rapid development, performance, scalability, openness, and security - and PHP is a strong option in the Web application development area.

This article appeared first on Idm.Internet.com.