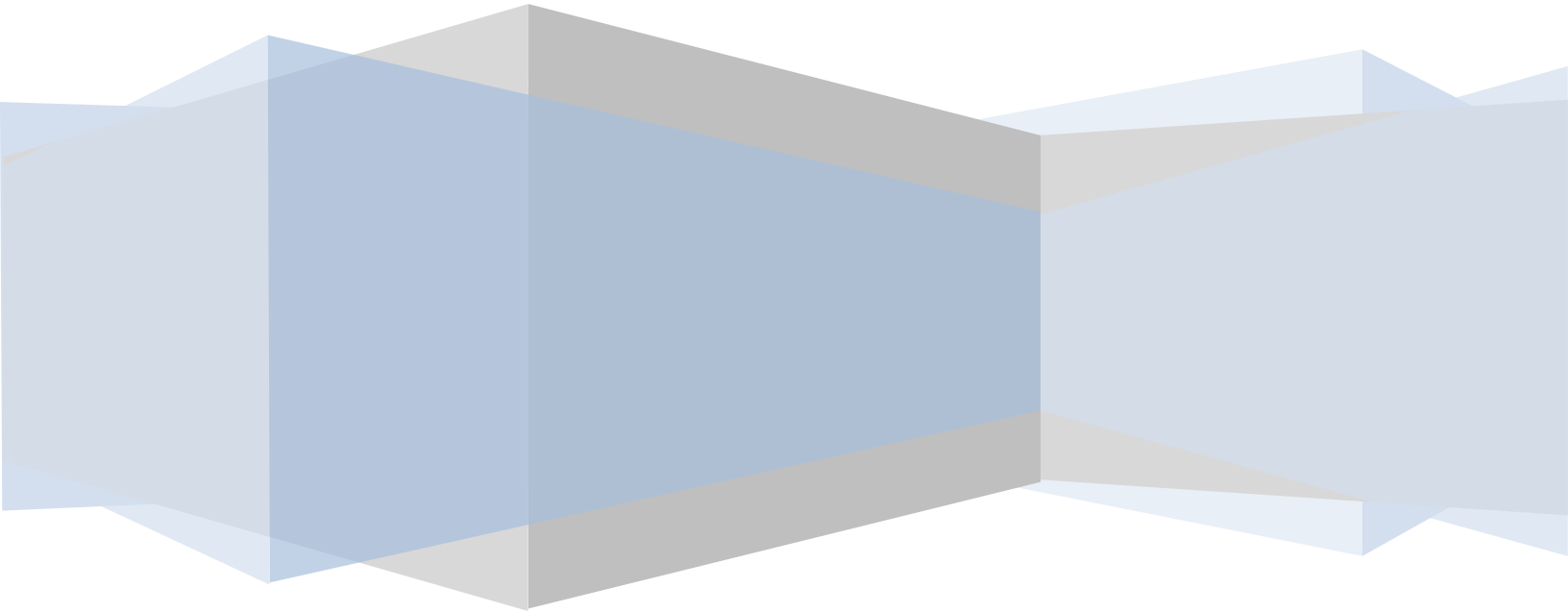
PHP Essentials



PHP Essentials – Edition 1.0

Find more technology eBooks at www.ebookfrenzy.com

Table of Contents

Chapter 1. About PHP Essentials

Any attempt to gauge the popularity of PHP on the internet results in statistics which prove difficult for the human mind to comprehend. As of April 2007 there were an estimated 20 million unique web domains actively using PHP to generate and deliver content. While it is hard to conceptualize 20 million web servers using PHP, it is not hard to infer from this number that PHP has taken the web design and development community by storm since humble beginnings in 1995.

The purpose of this book is bring the power and ease of use of PHP to anyone with a desire to learn PHP, and in doing so, join the tens of thousands of web developers who have already discovered the flexibility and productivity that comes with using PHP.

The book is intended to cover all aspects of PHP. It begins by covering the history of PHP before providing a high level overview of how PHP works and why it is so useful to web developers. It then moves on to cover each area of PHP in detail, from the basics of the scripting language through to object oriented programming, file and file system handling and MySQL and SQLite database access. In addition, chapters are also provided covering the creation and handling of HTML based forms and maintaining state using cookies and PHP sessions. All topics are accompanied by extensive real world examples intended to bring theory to life.

1.1 Intended Audience

It is anticipated that the typical reader already has some web based experience at least in terms of understanding the concepts of a web server and creating HTML based content. While prior programming and scripting experience will be beneficial to the reader, this book is designed such that even the non-programmer can quickly get up to speed with PHP.

Chapter 2. The History of PHP

Every once in a while a person faces a particular problem or requirement to which there appears to be no existing solution. Faced with this problem the person decides to create a solution to provide the needed functionality.

Having developed the solution to their problem it then occurs to them that others may need to solve the same problem, and they decide to make their solution freely available to others who, in turn, can use and improve on it. Within a short period of time many people adopt the technology and work on it, adding new features they feel will be useful. The solution soon grows beyond expectations in terms of features and is adopted by more people than the original creator could ever have imagined.

The history of PHP is just such a story.

2.1 The Creation of PHP

The first version of what came to be known as PHP was created in 1995 by a man named Rasmus Lerdof. Rasmus, now an engineer at Yahoo!, needed something to make it easier to create content on his web site, something that would work well with HTML, yet give him power and flexibility beyond what HTML could offer him. Essentially, what he needed was an easy way to write scripts that would run on his web server both to create content, and handle data being passed back to the server from the web browser. Using the Perl language, he created some technology that gave him what he needed and decided to call this technology "Personal Home Page/Forms Interpreter". The technology provided a convenient way to process web forms and create content.

The name "Personal Home Page/Forms Interpreter" was later shortened to PHP/FI and eventually renamed to represent "PHP: Hypertext Preprocessor". The name is said to be recursive because the full name also includes the acronym "PHP" - an odd geeky joke that is common in technology circles when people have trouble naming things. GNU is another recursive name that represents "GNU's Not Unix".

PHP/FI version 1.0 was never really used outside of Rasmus' own web site. With the introduction of PHP/FI 2.0 this began to change. When PHP 3 was released in 1997, adoption of PHP exploded beyond all belief.

2.2 PHP 3 Hits the Big Time

By the time 1997 arrived the number of web sites on the internet was growing exponentially and most of these web sites were being implemented using the Apache web server. It was

around this time that Andy Gutmans and Zeev Suraski launched the PHP 3 project, a project designed to take PHP to the next level. One of the key achievements of the PHP 3 project was to implement PHP as a robust Apache Module.

PHP 3 was implemented using a modular approach that made it easy for others to extend functionality, and also introduced the first elements of object-orientation that would continue to evolve through subsequent releases.

The combination of PHP 3 and Apache quickly lead to the widespread adoption of PHP, and it is commonly estimated that, at its peak adoption level, PHP3 was used to power over 10% of all web sites on the internet.

2.3 PHP 4 - Optimization, Scalability and More

With PHP 4 Andi Gutmans and Zeev Suraski once again re-architected PHP from the ground up. PHP 4 was built upon a piece of technology called the Zend Engine. The move to the Zend Engine brought about a number of key improvements in PHP:

- Support for other web servers (Microsoft's Internet Information Server (IIS) being of particular significance).
- Improved memory handling to avoid memory leaks (one of the most difficult types of problems to isolate in a program).
- Improved efficiency and performance to support large scale, complex, mission critical enterprise application development using PHP.

In addition PHP 4 also built on the earlier Object Oriented Programming features of PHP 3 with the introduction of classes.

2.4 PHP 5 - Object Orientation, Error Handling and XML

The main, though far from only, feature of PHP 5 is the improved support for Object Oriented Programming (OOP). In addition, PHP 5 introduced some features common in other languages such as Java like try/catch error and exception handling.

PHP 5 also introduced new extensions aimed at easing the storage and manipulation of data. Significant new features include SimpleXML for handling XML documents, and SQLite, an embedded basic and easy to use database interface.

2.5 How Popular is PHP?

A quick review of some statistics gives a very clear indication of the phenomenally widespread use of PHP. A company called Netcraft specializes in recording data about the types of web servers and web server modules that are used on the internet. As of April 2007 Netcraft reported that PHP was used on over 20,000,000 distinct web domains.

A web survey by SecuritySpace also lists PHP as the most widely deployed Apache module. It is safe to say that PHP has taken the internet by storm.

As if that wasn't enough one of the world's most popular web sites, Wikipedia, is build primarily using PHP.

Chapter 3. An Overview of PHP

Having covered the history of PHP in the previous chapter it is now time to provide some detail as to what PHP actually is. In this chapter we will take a high level look at PHP and provide a basic understanding of what it is, what is does and how it does it.

3.1 What Exactly is PHP?

PHP is an intuitive, server side scripting language. Like any other scripting language it allows developers to build logic into the creation of web page content and handle data returned from a web browser. PHP also contains a number of extensions that make it easy to interact with databases, extracting data to be displayed on a web page and storing information entered by a web site visitor back into the database.

PHP consists of a scripting language and an interpreter. Like other scripting languages, PHP enables web developers to define the behavior and logic they need in a web page. These scripts are embedded into the HTML documents that are served by the web server. The interpreter takes the form of a module that integrates into the web server, converting the scripts into commands the computer then executes to achieve the results defined in the script by the web developer.

3.2 How Does PHP Work?

To develop an understanding of how PHP works it is helpful to first explore what happens when a web page is served to a user's browser.

When a user visits a web site or clicks on a link on a page the browser sends a request to the web server hosting the site asking for a copy of the web page. The web server receives the request, finds the corresponding web page file on the file system and sends it back, over the internet, to the user's browser.

Typically the web server doesn't pay any attention to the content of the file it has just transmitted to the web browser. As far as the web server is concerned the web browser understands the content of the web page file and knows how to interpret and render it so that it appears as the web designer intended.

Now let's consider what kind of web page content a web browser understands. These days a web page is likely to consist of HTML, XHTML and JavaScript. The web browser contains code that tells it what to do with these types of content. For example, it understands the structure HTML in terms of rendering the page and it has a JavaScript interpreter built in that knows how to execute the instructions in a JavaScript script. A web browser, however, knows absolutely

nothing about any PHP script that may be embedded in an HTML document. If a browser was served a web page containing PHP it would not know how to interpret that code.

Given that a web browser knows nothing about PHP in a web page, then clearly something has to be done with any PHP script in the page before it reaches the browser. This is where the PHP pre-processing module comes in. The PHP module is, as mentioned previously, integrated into the web server. The module tells the web server that when a page is to be served which contains PHP script (identified by special markers) that it is to pass that script to the PHP pre-processing module and wait for the PHP module to send it some content to replace that script fragment. The PHP processing module understands PHP, executes the PHP script written by the web developer and, based on the script instructions, creates output that the browser will understand. The web server substitutes the content provided by the PHP pre-processor module in place of the PHP script in the web page and sends it to the browser where it is rendered for the user to view.

To help understand this concept let's take a quick look at a before and after scenario. The following HTML contains some PHP script that is designed to output an HTML paragraph tag:

```
<html>
<head>
<title>A PHP Example</title>
</head>
<body>
<?php
     echo '<p>This line of HTML was generated by a PHP script embedded into
an HTML document</p>';
?>
</body>
</html>
```

The above example looks very much like standard HTML until you reach the part surrounded by *<?php* and *?>*. These are markers that designate where the embedded PHP script begins and ends. When the web server finds this it sends it to the PHP module. The PHP module interprets it, converts it to HTML and sends it back to the web server. The web server, in turn, sends the following to the browser:

```
<html>
```

```
<head>
<title>A PHP Example</title>
</head>
<body>

<p>This line of HTML was generated by a PHP script embedded into an HTML
document</p>
</body>
</html>
```

Once loaded into the browser, it is rendered just like any other web page. The fact that the web page originally contained PHP is completely transparent to the web browser.

The above example is certainly an oversimplification of the power of PHP. Some may question why one would use PHP to output some static text that could have been achieved more easily using an HTML tag. The fact is, however, HTML only makes sense if you know beforehand exactly what needs to be displayed in the web page. Imagine instead, that you are developing an online banking application. One of the pages you need to display must contain the customer's bank account number combined with the current balance. Obviously this information is going to be different for each customer. In this scenario you would develop an HTML page that essentially serves as a template for the page, and then embed PHP into the page to extract the account and balance information from a database. Once processed by the PHP module integrated into the web server, this customer specific content will then appear in the HTML page in place of the PHP script when the page is loaded into the browser.

3.3 Why is PHP so Useful?

In terms of web page content we have two extremes. At one extreme we have HTML which is completely static. There is very little that can be done with HTML to create dynamic content in a web page. At the other extreme we have scripting languages like JavaScript. JavaScript provides a powerful mechanism for creating interactive and dynamic web pages.

When talking about JavaScript it is important to understand that it is, by design, a client side scripting language. By this we mean that the script gets executed inside the user's browser and not on the web server on which the web page originated. Whilst this is fine for many situations it is often the case that by the time a script reaches the browser it is then either too late, or inefficient, to do what is needed. A prime example of this involves displaying a web page which contains some data from a database table. Since the database resides on a server (either the

same physical server which runs the web server or on the same network as the web server connected by a high speed fiber network connection) it makes sense for any script that needs to extract data from the database to be executed on the server, rather than waiting until it reaches the browser. It is for this kind of task that PHP is perfectly suited. It is also fast and efficient (because the script is executed on the server it gets to take advantage of multi-processing, large scale memory and other such enterprise level hardware features.

In addition to the advantages of being a server side scripting language PHP is easy to learn and use. The fact that PHP works seamlessly with HTML makes it accessible to a broad community of web designers.

Perhaps one of the most significant advantages of PHP to some is the ease with which it interacts with the MySQL database to retrieve and store data.

3.4 Summary

In summary, PHP has many advantages, and those listed here are just some of the reasons for the success of PHP. Many people will offer their own reasons for using PHP - and this fact alone is testament to the power and flexibility of PHP.

Now that we know a little about the history of PHP and have an overview of how it works we can start looking at how to develop PHP based web sites, beginning with creating a simple PHP script.