Contents of The Lecture

Chapter 1, Introduction to PHP

It talks about the history of PHP and gives a lightning-fast overview if what is possible with PHP programs.

**CHAPTER 1 – Introduction to PHP**

PHP is a simple yet powerful language designed for creating HTML content. This chapter covers essential background on the PHP language. It describes the nature and history of PHP, which platforms it runs on, and how to configure it. This chapter ends by showing you PHP in action, with a quick walkthrough of several PHP programs that illustrate common tasks, such as processing form data, interacting with a database, and creating graphics.

**What Does PHP DO?**

PHP can be used in two primary ways:

Server-side scripting

PHP was originally designed to create dynamic web content and is still best suited for that task. To generate HTML, you need a PHP parser and a web server to send the coded document files. PHP has also become popular for generating dynamic content via database connections, XML documents, graphics, PDF files, and more.

Command-line scripting

PHP can run scripts from the command line like Perl, awk, or the Unix shell. You might use the command-line scripts for system administration tasks, such as backup and log parsing; even some SRON job-type scripts can be done this way (As nonvisual PHP tasks).

In this document, however, we concentrate on the first item: using PHP to develop dynamic web content.

PHP runs on all major operating systems, from Unix variants (including Linux, FreeBSD, Ubuntu, Debian, and Solaris) to Windows and macOS. It can be used with all leading web servers, including Apache, Nginx, and OpenBSD servers, to name a few, even cloud environments like Azure and Amazon.

The language itself is extremely flexible. For example, you aren’t limited to outputting just HTML or other text files- any document format can be generated. PHP has built-in support for generating PDF files, GIF, JPEGs, and PNG images.

One of PHP’s most significant features is its wide-ranging support for databases. PGP supports all major databases (including MYSQL, PostgreSQL, Oracle, Sybase, MS-SQL, DB2, and ODBC-compliant databases), and even many obscure ones. Even the more recent NoSQL-style databases like CouchDB and MongoDB are also supported. Creating web pages with dynamic content from a database with PHP is remarkably simple.

Finally, PHP provides a library of PHP code to perform common tasks, such as database abstraction, error handling, and so on, with the PHP Extension and Application Repository (PEAR). PEAR is a framework and distribution system for reusable PHP components.

**A Brief History of PHP**

Rasmus Lerdorf first conceived of PHP in 1994, but the PHP that people use today differs from the initial version. To understand how PHP got where it is now, it is useful to know the historical evolution of the language. Here’s that story, with ample comments and emails from Rasmus himself.

**The Evolution of PHP**

Here is the PHP 1.0 announcement that was posted to the Usenet newsgroup (*comp.infosystems.www.authoring.cgi*) in June 1995:

From: rasmus@io.org (Rasmus Lerdorf)

Subject: Announce: Personal Home Page Tools (PHP Tools)

Date: 1995/06/08

Message-ID: <3r7pgp$aa1@ionews.io.org>#1/1

organization: none

newsgroups: comp.infosystems.www.authoring.cgi

Announcing the Personal Home Page Tools (PHP Tools) version 1.0.

These tools are a set of small tight cgi binaries written in C.

They perform a number of functions including:

. Logging accesses to your pages in your own private log files

. Real-time viewing of log information

. Providing a nice interface to this log information

. Displaying last access information right on your pages

. Full daily and total access counters . Banning access to users based on their domain

. Password protecting pages based on users' domains

. Tracking accesses \*\* based on users' e-mail addresses \*\*

. Tracking referring URL's - HTTP\_REFERER support

. Performing server-side includes without needing server support for it

. Ability to not log accesses from certain domains (ie. your own)

. Easily create and display forms

. Ability to use form information in following documents

Here is what you don't need to use these tools:

. You do not need root access - install in your ~/public\_html dir

. You do not need server-side includes enabled in your server

. You do not need access to Perl or Tcl or any other script interpreter

. You do not need access to the httpd log files

The only requirement for these tools to work is that you have

the ability to execute your own cgi programs. Ask your system

administrator if you are not sure what this means.

The tools also allow you to implement a guestbook or any other

form that needs to write information and display it to users

later in about 2 minutes.

The tools are in the public domain distributed under the GNU

Public License. Yes, that means they are free!

For a complete demonstration of these tools, point your browser

at: http://www.io.org/~rasmus

--

Rasmus Lerdorf

rasmus@io.org

http://www.io.org/~rasmus

Note that this message’s URL and email address are long gone. The language of this announcement reflects people’s concerns at the time, such as password-protecting pages, easily creating forms, and accessing forms data on subsequent pages. The announcement illustrates PHP’s initial positioning as a framework for several use cases.

The announcement talks only about the tools that came with PHP, be behind the scenes, the goal was to create a framework to make it easy to extend PHP and add more tools. The business logic for these add-ons was written in C, a simple parser picked tags out of the HTML and called the various C functions. It was never really part of the plan to create a scripting language.

So what happened?

Rasmus started working on a rather large project for the University of Toronto that needed a tool to gather data from various places and present a nice web-based administration interface. OF course, he used PHP for that task, but for performance reasons, the various small tools of PHP 1.0 had to be brought together better and integrated into the web server.

Initially, some hacks to the NCSA web server were made to patch it to support the core PHP functionality. The problem with this approach was that you had to replace your web server software with this special, hacked-up version as a user. Fortunately, Apache was also gaining momentum around this time, and the Apache API made adding functionality like PHP to the server easier.

Over the next year or so, a lot was done, and the focus changed quite a bit. Here’s the PHP 2.0 (PHP/FI) announcement that was sent out in April 1996:

From: rasmus@madhaus.utcs.utoronto.ca (Rasmus Lerdorf)

Subject: ANNOUNCE: PHP/FI Server-side HTML-Embedded Scripting Language

Date: 1996/04/16

Newsgroups: comp.infosystems.www.authoring.cgi

PHP/FI is a server-side HTML embedded scripting language. It has built-in

access logging and access restriction features and also support for

embedded SQL queries to mSQL and/or Postgres95 backend databases.

It is most likely the fastest and simplest tool available for creating

database-enabled web sites.

It will work with any UNIX-based web server on every UNIX flavour out

there. The package is completely free of charge for all uses including

commercial.

Feature List:

. Access Logging

Log every hit to your pages in either a dbm or an mSQL database.

Having hit information in a database format makes later analysis easier.

. Access Restriction

Password protect your pages, or restrict access based on the refering URL

plus many other options.

. mSQL Support

Embed mSQL queries right in your HTML source files

. Postgres95 Support

Embed Postgres95 queries right in your HTML source files

. DBM Support

DB, DBM, NDBM and GDBM are all supported

. RFC-1867 File Upload Support

Create file upload forms

. Variables, Arrays, Associative Arrays

. User-Defined Functions with static variables + recursion

. Conditionals and While loops

Writing conditional dynamic web pages could not be easier than with

the PHP/FI conditionals and looping support

. Extended Regular Expressions

Powerful string manipulation support through full regexp support

. Raw HTTP Header Control

Lets you send customized HTTP headers to the browser for advanced

features such as cookies.

. Dynamic GIF Image Creation

Thomas Boutell's GD library is supported through an easy-to-use set of

tags.

It can be downloaded from the File Archive at: <URL:http://www.vex.net/php>

--

Rasmus Lerdorf

[rasmus@vex.net](mailto:rasmus@vex.net)

This was the first time the term scripting language was used. PHP 1.0’s simplistic tag-replacement code was replaced with a parser to handle a more sophisticated embedded tag language. By today’s standards, the tag language wasn’t particularly sophisticated, but compared to PHP 1.0, it certainly was.

The main reason for this change was that only some people who used PHP 1.0 were actually interested in using the C-based framework for creating add-ons. Most users were much more interested in being able to embed logic directly in their web pages for creating conditional HTML, custom tags, and other such features. PHP 1.0 users constantly requested the ability to add the hit-tracking footer or send different HTML blocks conditionally. This led to the creation of an if tag. Once you have if, you need else as well, and from there, it’s a slippery slope to the point where you write an entire scripting language, whether you want to or not.

By mid-1997, PHP version 2.0 had grown quite a bit and had attracted a lot of users, but there were still come stability problems with the underlying parsing engine. The project was also sill mostly a one-man effort, with a few contributions here and there. At this point, Zeev Suraski and Andi Gutmans in Tel Aviv, Israel, volunteered to rewrite the underlying parsing engine, and we agreed to make their rewrite the base for PHP version 3.0. Other people also volunteered to work on other parts of PHP, and the project changed from a one-person effort with a few contributors to a true open-source project with many developers worldwide.

Here is the PHP 3.0 announcement from June 1998:

June 6, 1998 -- The PHP Development Team announced the release of PHP 3.0,

the latest release of the server-side scripting solution already in use on

over 70,000 World Wide Web sites.

This all-new version of the popular scripting language includes support

for all major operating systems (Windows 95/NT, most versions of Unix,

and Macintosh) and web servers (including Apache, Netscape servers,

WebSite Pro, and Microsoft Internet Information Server).

PHP 3.0 also supports a wide range of databases, including Oracle,

Sybase, Solid, MySQ, mSQL, and PostgreSQL, as well as ODBC data sources.

New features include persistent database connections, support for the

SNMP and IMAP protocols, and a revamped C API for extending the language

with new features.

"PHP is a very programmer-friendly scripting language suitable for

people with little or no programming experience as well as the

seasoned web developer who needs to get things done quickly. The

best thing about PHP is that you get results quickly," said

Rasmus Lerdorf, one of the developers of the language.

"Version 3 provides a much more powerful, reliable, and efficient

implementation of the language, while maintaining the ease of use and

rapid development that were the key to PHP's success in the past,"

added Andi Gutmans, one of the implementors of the new language core.

"At Circle Net we have found PHP to be the most robust platform for

rapid web-based application development available today," said Troy

Cobb, Chief Technology Officer at Circle Net, Inc. "Our use of PHP

has cut our development time in half, and more than doubled our client

satisfaction. PHP has enabled us to provide database-driven dynamic

solutions which perform at phenomenal speeds."

PHP 3.0 is available for free download in source form and binaries for

several platforms at http://www.php.net/.

The PHP Development Team is an international group of programmers who

lead the open development of PHP and related projects.

For more information, the PHP Development Team can be contacted at

core@php.net.

After the release of PHP 3.0, usage really started to take off. Version 4.0 was prompted by a number of developers who were interested in making some fundamental changes to the architecture of PHP. These changes included abstracting the layer between the language and the web server, adding a thread-safety mechanism, and adding a more advance two-stage parse/execute tag-parsing system. This new parser, primary written by Zeev and Andi, was named the Zend engine. After a lot of work by a lot of developers, PHP 4.0 was released on May 22,2000.

As this document is transcribed, PHP version 7.3 has been released for some time. There have already been a few minor “dot” releases, and the stability of this current version is quite high. As you will see in this document, there have been some major advances made in this version of PHP, primarily in code processing on the server side. Many other minor changes, function additions, and feature enhancements have also been incorporated.

**The Widespread Use Of PHP**

Figure 1-1 shows the usage of PHP as compiled by W3 Techs as of July 2023. The most interesting piece of data here is that 77% of all the surveyed websites use it, with version 7 being the most widely used. If you look at the methodology used in the W3Techs surveys, you will see that they select the world's top 10 million sites (based on traffic; website popularity). As it evident, PHP has a very broad adoption indeed.

A screenshot of a web page

Description automatically generated

Figure 1-1 PHP usage as of July 2023

**Installing PHP**

As mentioned, PHP is available for many operating systems and platforms. Therefore, you are encouraged to consult the PHP documentation to find the environment that most closely fits the one you will be using and follow the appropriate setup instructions.

From time to time, you may also want to change the way PHP is configured. To do that , you will have to change the PHP configuration file and restart your web (Apache) server in order for those changes to take effect.

PHP’s configuration settings are usually maintained in a file called php.ini. The settings in this file control the behavior of PHP features, such as session handling and form processing. Later chapters refer to some of the php.ini options, but generally, the code in this Document does not require a customized configuration. See the PHP documentation for more information on configuring php.ini.

**A Walk Through of PHP**

PHP pages are generally HTML pages with PHP commands embedded in them. This is in contrast to many other dynamic web page solutions, which are scripts that generate HTML. The web server processes the PHP commands and sends their output (and any HTML from the file) to the browser. Example 1-1 shows a complete PHP page

<html>

    <head>

        <title> Look Out World</title>

    </head>

    <body>

        <?php echo "Hello, world!";?>

    </body>

</html>

Example 1-1. Hello\_world.php

Save the contents of Example 1-1 to a file, hello\_world.php, and point your browser to it. The results appear in Figure 1-2.

A blue screen with white text

Description automatically generated

Figure 1-2. Output of hello\_world

The PHP echo command produces output (the string “Hello, world!” in this case) inserted into the HTML file. In this example the PHP code is placed between the <?php and ?> tags. There are other ways to tag your PHP code – see Chapter 2 for a full description.

**Configuration Page**

The PHP function phpinfo() creates an HTML page full of information on how PHP was installed and is currently configured. You can use it to see whether you have a particular extensions installed, or whether the php.ini file has been customized. Example 1-2 is a complete page that displayed the phpinfo() page.

<html>

<head>PHP info</head>

<body>

    <?php phpinfo();?>

</body>

</html>

Example 1-2. Using phpinfo()

A screenshot of a computer

Description automatically generated

Figure 1-3. Partial output of phpinfo()

**Forms**

Example 1-3 creates and processes a form. When the user submits the form, the information typed into the name field is sent back to this page via the $\_SERVER[‘PHP\_SELF’] form action. The PHP code tests for a name field and displays a greeting if it finds one.

<html>

    <head>

        <title>Personalized Greeting Form</title>

    </head>

    <body>

    <?php if(!empty($\_POST['name'])){

        echo "Greetings, {$\_POST['name']}, and welcome.";

    } ?>

    <form action="<?php echo $\_SERVER['PHP\_SELF']; ?>" method="post">

    Enter your name: <input type="text" name ="name" />

    <input type="submit"/>

    </form>

    </body>

</html>

The form and the message are shown in Figure 1-4.

A screenshot of a computer

Description automatically generated

Figure 1-4. Form and greeting page

PHP programs access form values primarily through the $\_POST and $\_GET array variables. Chapter 8 discusses form and form processing in more detail.

**Databases**

PHP supports all the popular database systems, including MySQL, PostgreSQL, Oracle, Sybase, SQLite, and ODBC-compliant databases. Figure 1-5 shows part of a MySQL database query run through a PHP script, displaying the results of a book search on a book review site. It lists the book title, the year the book was published, and the book’s ISBN.

Figure 1-5. A MySQL book list query run through a PHP scriptA white rectangular table with black text and numbers

Description automatically generated

The code in Example 1-4 connects to the data base, uses a query to retrieve all available books with the WHERE clause), and produces a table as output for all returned results through a while loop. The SQL code for this sample database is in the provided file library. SQL. You can drop this code into MySQL after you create the library database and have the sample database at your disposal for testing out the following code sample as well as the related samples in Chapter 9.

<?php

$db = new mysqli("localhost", "peterwin", "password", "library");

// fictitious database connect function

// Check connection

if ($db->connect\_errno) {

    die("Connect Error (" . $db->connect\_errno . ") " . $db->connect\_error);

}

$sql = "SELECT \* FROM books WHERE available = 1 ORDER BY title";

$result = $db->query($sql);

?>

<!DOCTYPE html>

<html>

<head>

    <style>

        table {

            margin-left: auto;

            margin-right: auto;

            border-collapse: collapse;

        }

        th, td {

            padding: 6px;

            border: 1px solid black;

        }

        th {

            text-align: center;

        }

    </style>

</head>

<body>

<table>

    <tr>

        <td colspan="3">

            <h3 align="center">These Books are currently available</h3>

        </td>

    </tr>

    <tr>

        <th>Title</th>

        <th>Year Published</th>

        <th>ISBN</th>

    </tr>

    <?php while ($row = $result->fetch\_assoc()) { ?>

        <tr>

            <td><?php echo htmlspecialchars($row['title']); ?></td>

            <td align="center"><?php echo htmlspecialchars($row['pup\_year']); ?></td>

            <td><?php echo htmlspecialchars($row['ISBN']); ?></td>

        </tr>

    <?php } ?>

</table>

</body>

</html>

Database-provided dynamic content that drives the news, blogs, and e-commerce sites at the heart of the web. More details on accessing databases from PHP are given in Chapter 9. (Instructors note: if you receive the following error -Uncaught Error: Call to undefined function imagecreatefrompng()

Try the followingThe error you're encountering is caused by the GD library not being enabled in your PHP configuration. The GD library is required for image processing functions like imagecreatefrompng() and imagettftext(). To resolve this issue, you need to enable the GD library.

Here are the steps to enable the GD library:

Locate the PHP configuration file (php.ini). The location of this file can vary depending on your operating system and PHP installation.

Open the php.ini file in a text editor. Search for the following line: ;extension=gd

Remove the semicolon at the beginning of the line to uncomment it: extension=gd Save the php.ini file. Restart your web server to apply the changes.)

**Graphics**

With PHP, you can easily create and manipulate images using the GD extension. Example 1-5 provides a text entry field that lets the user specify the text for a button. It takes an empty button image file, and centers the text passed as the GET parameter ‘message’ on it. The result is sent back to the browser as a PNG image.

The form generated by example 1-5 is shown in Figure 1-6. The button created is shown in Figure 1-7.

A screenshot of a computer

Description automatically generated

Figure 1-6. Button creation form

A red rectangular object with black text

Description automatically generated

Figure 1-7. Button created

You can use GD to resize images dynamically, produce graphs, and much more. PHP also has several extensions to generate documents in Adobe’s popular PDF format. Chapter 10 covers dynamic image generation in depth, while Chapter 11 provides instructions on creating Adobe PDf files.

**What’s Next**

Now that you’ve tasted what is possible with PHP, you are ready to learn how to program in the language. We start with its basic structure, focusing on user-defined functions, string manipulation, and object-oriented programming. Then we move to specific application areas, such as the web, databases, graphics, XML, and security. We finish with quick references to the built-in functions and extensions. Master these chapters, and you will have mastered PHP!