Febuary 21, 2015
CheckoutCrypto Canada

Introduction to PHP

By: Grant Hutchinson

Febuary 21, 2015
CheckoutCrypto Canada

Index

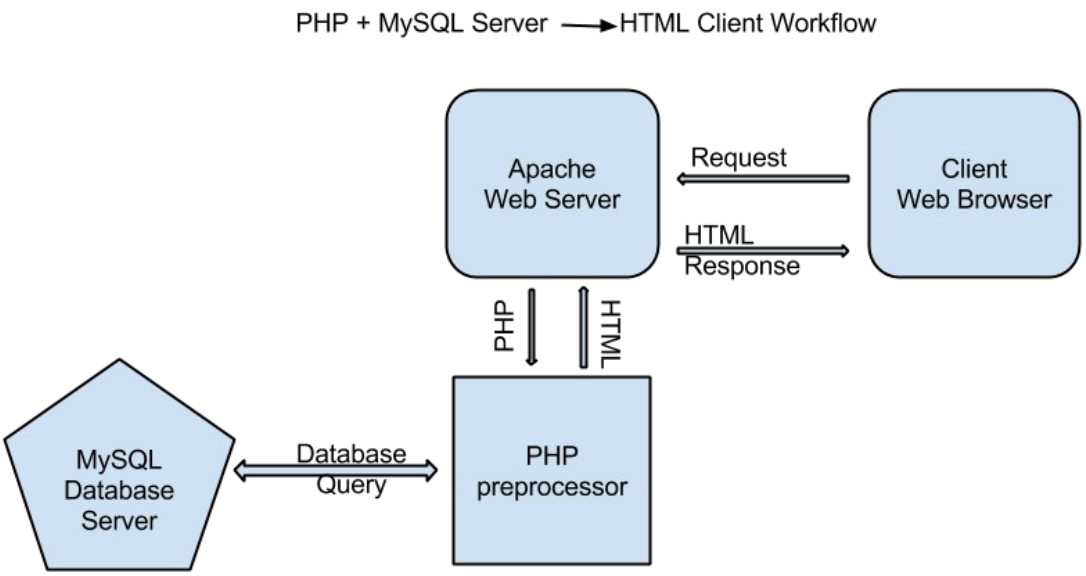
Reference Documentation:

http://php.net/manual/en/langref.php

Febuary 21, 2015
CheckoutCrypto Canada

PHP

PHP is a programming language for use with server-side web development. PHP originally stood for Personal Home Page but nows stands for PHP: Hypertext Preprocessor. We can utilize it for a regular scripting language, a content management system, CGI, CLI, or a large variety of web libraries. The backends of most websites are typically written to use PHP as an interpreter. Once PHP code has run, the webserver sends the response back to the client's web browser in the form of HTML. PHP5 must be run in a webserver such as apache2.



Basic Structure:

```
<?php

echo "Hello World!";

?>
```

Explanation:

PHP must always start with a <?php tag and end with a ? > tag.

PHP files are typically named .php

echo is the PHP command for print or display, any text within the quotations, to the screen. All PHP expressions typically end with a ;

Try it!

Create a new php file called helloworld.php place it in your webserver, at your site's root /var/www/site/helloworld.php . If you don't have a site setup yet, or even an apache2 server, I recommend you install and configure that before you go on any further. If you need help, I suggest reviewing today's Lesson One: Introduction to Ubuntu for apache2 installation.

In any browser, visit: http://127.0.0.1/site/helloworld.php

Once this page displays:

Hello World!

Note: If you do not see the text. First, check php5 is correctly installed. Second, you can debug by browsing to the file's path (cd /var/www/site), then run this to check for errors:

```
php -l ./helloworld.php
```

Congratulations, you are ready to begin PHP.

Variables, Data Types, Constants, Expressions

boolean - true or false
integer - any number from -9-> + 9
float - any floating point number from -9.9 -> +9.9
string - any characters surrounded by quotations.
Array - list of data, each index position is indicated by key

```
<?php
	$any_variable_name_we_want = "";  ///true, false, 1, 2.5, 0.0353, "HelloWorld!"
?>
```

Explanation:

$any_variable_name_we_want - any variable can be called, at any time, anywhere, any name(as long as it isn't a reserved variable e.g. echo) as long as you declare it with a $ at the beginning. e.g.

```
<?php
$somevar1 = 1423.97;
$somevar2 = 9;
$someresult =  $somevar1 + $somevar2;
$somevar3 = false;
$somevar4 = "success";

echo $somevar1 . " + " . $somevar2 . " = " . $someresult  . "  " ;
echo $somevar3;
echo $somevar4;

echo "<br> " . $somevar2++;
?>
```

Try it!

Create a new text file called variables.php then type out the above example. Save it, open a browser, visit: http://127.0.0.1/site/variables.php or your site root path.

Result:

1423.97 + 9 = 1432.97 success
10

Explanation:

echo $somevar1 . " + " . $somevar2 . " = " . $someresult . " " ;

This line is printing each variable, then it is concatenating the strings together with the . The + and = in this expression are strings surrounded by quotes.

echo $somevar3;

This line isn't displaying, the reason being, it's a boolean data type, not string. Even though it's false, it would have to be “false” with double quotes, for it to display.

echo "
 " . $somevar2++;

Here you can see, we can print html code directly. In this case, I added a page break(new line) in addition, I iterated the $somevar2 which originally = 9, but the result displays 10. This is because of the ++ I added, which simply means add one(+1) to this variable.

Similarily you can deincrement by adding two dashes(--) to the variable to subtract one(-1).

String Manipulation

```
<?php
        $something = "cars";
        $useful =  "trucks";
        $number = 2;
        $another = 4;
        echo $something  .  $useful;
        echo "<br> " . $something  . " " . $useful;
        echo "<br> " . $something . $another;
        echo "<br> " . $something . ($number + $another);
?>
```

Try it!
Create a new text file called variables2.php then type out the above example. Save it, open a browser, visit: http://127.0.0.1/site/variables2.php or your site root path.

Result:

carstrucks
cars trucks
cars4
cars6

Explanation

. will concatenate strings
+ will add to strings together(addition)

Don't forget BEDMAS (brackets, exponents, division, multiplication, addition, subtraction).

String Functions

```
<?php
$message = "I am groot!";
$parse = "root";

// Get Length
$msgLength = strlen($message);

// Get Position
$parsePos = strpos($message, $parse);

// Get Final String
$final = substr($message, $parsePos, $msgLength);
echo $final;
?>
```

Try it!
Create a new text file called phpstrings.php then type out the above example. Save it, open a browser, visit: http://127.0.0.1/site/phpstrings.php or your site root path.

Explanation

strncmp, strcmp, strcasecmp - compare strings based on case-sensitivty
strpos - find the index of a series of characters within a string
strchr – find the index of a character within a string
trim - strip characters from a string
strlen - retrieve the integer length of characters in a string
substr - retreive part of a string based on the starting character and length
strtolower, **strtoupper** – make string upper or lower case

Predefined Variables

$_SERVER
$_GET
$_POST
$_ENV
$_COOKIE
$_REQUEST

```
<?php
$retrievedVariable  =  $_GET['anyvariable'];


print_r($_GET); print('<br>');
print_r($_REQUEST);print('<br>');
print_r("Final retrieved variable = ".$retrievedVariable);

?>
```

Try it!
Create a new text file called php_get.php then type out the above example. Save it, open a browser, visit: http://127.0.0.1/site/php_get.php or your site root path.
Except this time we're going to add variables to our URL so we can GET them in php using the predefined $_GET[] array. So actually, you should instead visit:

http://127.0.0.1/site/php_get.php?anyvariable=test&something=special

the ?anyvariable=test&something=special are our 2 variables we're sending via GET we have to begin adding parameters by adding just one ? before our variables. Then we seperate each variable with an & symbol. These variables are then easily read into a php variable with:

```
<?php
$somevariable = $_GET["something"];
/// or
$somevariable = $_GET["anyvariable"];

echo $somevariable;
?>
```

Conditions

== - is equal to
<= - is less than or equal to
>= - is greater than or equal to
< - less than
> - greater than
| | - or e.g. $thiscommand | | $thatcommand
&& - and e.g. $thiscommand && $thiscommand

If/else if/else Condition

```
<?php

$something = "some_expected_result";

if( $something == "some_expected_result"){
        echo "success <br>";
}else{
        echo "failure <br>";
}

$something = "some_result";

if( $something == "some_result" || $something ==  "another_result"){
        echo "success";
}else if( $something == "some_result" && $something ==  "another_result"){
        echo "success";
}else{
        echo "failure";
}

?>
```

switch Condition - multiple conditional statement, whenever you have many result conditions.
case - each possible result from the switch condition
break - end each case section
default – similar to an else clause of an if/else condition, it's a catch all, wildcard.

```
<?php

$something = "success";

switch($something){

	case "success":
		echo "success";
	break;
	case "failure":
		echo "failure";
	break;
	default:
		echo "anything";
	break;
}

?>
```

Try it!
Create two new text files called php_condition1.php php_condition2.php then type out the above examples, one for switch, one for if/else conditions. Save it, open a browser, visit: http://127.0.0.1/site/php_condition1.php or your site root path.
visit: http://127.0.0.1/site/php_condition2.php or your site root path.

Loops

If you're unfamiliar, loops are deliberate recalling of lines of code. There are several forms of loops present in all programming languages. Typically a PHP server does this when it has to iterate through results or queries. PHP runs serverside, as the programmer you must ensure that any loops are minimized and meant to execute in the shortest amount of time possible.

```
<?php
$somearr = array( 1, 2, 3, 4, 5, 6 );
$anotherarr = array( "yes", "no", "maybe" );
$keyArr = array( "answer1" => "perhaps", "answer2" => "well", "answer3" => "later");

var_dump($somearr);   /// display the entire array
echo  "<br>";
var_dump($anotherarr);
echo  "<br>";
var_dump($keyArr);
echo  "<br>";

/// foreach loop, display each index of someArray
foreach($somearr as $thing){
        echo $thing . "<br>";
}

/// for loop, display each index of anotherArray
for($x = 0; $x <=2; $x++){
        echo $anotherarr[$x] . "<br>";
}

/// for loop with array key, display each index of $keyArr
for($x = 0; $x <=2; $x++){
        if($x == 0){     ///  Condition statement, if the loop just started!
                echo $keyArr["answer1"] . "<br>";
        }else if($x == 1){
                echo $keyArr["answer2"] . "<br>";
        }else if($x == 2){
```

```
                echo $keyArr["answer3"] . "<br>";
        }
}
?>
```

Try it!
Create a new text file called php_loops.php then type out the above example. Save it, open a browser, visit: http://127.0.0.1/site/php_loops.php or your site root path.

Explanation:

Basic **for** loop structure

```
for($x= 0; $x <= $somenumber; $x++)  {
        /// your loop conditions and content
}
```

Basic **foreach** loop structure

```
foreach($list_of_items as $oneitem_at_a_time){
        /// your loop conditions and content
}
```

Basic **while** loop structure

```
while( $condition) {
        /// some condition and content
}
```

Basic **do-while** loop structure

```
do{
        /// some condition and content
}while( $someCondition );
```

Functions

Functions are seperate sections of code, organized into groups, which take a number of parameters, may return a value. Typically these functions are grouped into seperate libraries and classes, included in a page load.

For example:

```
<?php

function helloworld(){
        echo "Hello World!";
}

helloworld();   /// function call
?>
```

Try it

Create a new text file called php_functions.php then type out the above example. Save it, open a browser, visit: http://127.0.0.1/site/php_functions.php or your site root path.

Now again typically these functions are organized into libraries and included using:

```
include('./somepath/somefile.php');  or    include_once('somefile.php');
```

or

```
require('./somefile.php');
```

Here's an example of that, create two new files, one called lib.php the other called functions.php

lib.php

```
<?php

function helloworld(){
	return "Hello World!";
}
function hellomath($somevar, $someNum){
	$result = ($somevar * $someNum) + 5;
	return $result;
}

function hellodisplay($msg, $amt){
	for($x=0; $x < $amt; $x++){
		echo $msg . "<br>";
	}
}
?>
```

functions.php

```
<?php
	require('lib.php');
	$result = hellomath(5, 1);
	hellodisplay(helloworld(), $result);
?>
```

open a browser, visit: http://127.0.0.1/site/functions.php or your site root path.

Using this technique you can see how easy it is to organize your code into functions. Even then, you'll notice quickly you'll end up with many functions from all over the place and organizing them with their own parameters gets quite tedious. This is why the next part of the tutorial, we'll discuss classes.

Classes

Classes are objects or libraries of functions and parameters relative in origin.

```
<?php

class Car {

var $car;

        function __construct($col, $brnd, $mdl)
    {
                $this->setCar($col, $brnd, $mdl);
        }
        function setCar($col, $brnd, $mdl){
                $this->car = array( "color" => $col, "brand" => $brnd, "model" => $mdl);
        }
        function display(){
                var_dump($this->car);
        }
        function getCar(){
                return $this->car;
        }
        function getModel(){
                return $this->car["model"];
        }
        function getColor(){
                return $this->car["color"];
        }
        function getBrand(){
                return $this->car["brand"];
        }
        function __destruct(){

        }
}
?>
```

Here's a good class example for you to try.

First copy or type the car class, above, to a new text file, call it **car.php**
Next, create another textfile, this time call it dealership.php

Copy or type the following into the **dealership.php**:

```
<?php
require('car.php');

$car = new Car("red","ford", "taurus");
$car->display();

echo "<br>" . $car->getColor();
echo "<br>" . $car->getBrand();
echo "<br>" . $car->getModel();
?>
```

Try it!
open a browser, visit: http://127.0.0.1/site/dealership.php or your site root path.

Result:

array(3) { ["color"]=> string(3) "red" ["brand"]=> string(4) "ford" ["model"]=> string(6) "taurus" }
red
ford
taurus

The final part of this lesson, I'm going to show you how to put this altogether and build your own mini website engine.

Febuary 21, 2015
CheckoutCrypto Canada

How to create a PHP based content management system

Write a standard html page, we're going to seperate and template our sections with php. Remember to always tab in at least 1 indent, so it's easier to read.

Example.html

```
<html>
<head><title>PHP Example</title>
 <link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
	<div class=”main”>  <!-- Total Outter page box -->
		<div class=”header”><p>Header/logo here</p></div>
		<div class=”navigation”><p>Navigation bar here</p></div>
		<div class=”content”><p> Some content here</p></div>
		<div class=”footer”><p>Footer here</p></div>
	</div>
</body>
</html>
```

Create a CSS style called style.css

```
/*
*  Remember, each class inherits the parameters of the class it resides in.
*/


.main {
	width:1024px;
}

.header {
	height:12px;     /// used to be a rule of thumb, do not make the header/navigation too large
	color:teal;
}
```

```
.navigation {
	height:12px;	/// used to be a rule of thumb, do not make the header/navigation too large
	color:red;
}

.content {
	height:300px;
	color:white;
}

.footer {
	height:12px;	/// used to be a rule of thumb, do not make the header/navigation too large
	color:grey;
}
```

Now make sure it all works by visiting http://127.0.0.1/site/Example.html

Open a new text document, we're going to template the above page, so we can use the layout on every single page we need to write. Create a new text file called layout.php

```
<?php   // start the php document

function my_header(){
?>

/// we'll insert our html here!

<?php
}
/// end the php document
?>
```

What we're doing here is we're open and closing the php code where necessary. PHP and HTML work together to display the page.

Now continue the trend with all the other parts of the page, like this:

```
<?php

function my_header(){
?>
/// we'll insert our html here!
<?php
}

function my_navigation(){
?>
/// we'll insert our html here!
<?php
}

function my_content(){
?>
/// we'll insert our html here!
<?php
}

function my_footer(){
?>
/// we'll insert our html here!
<?php
}

?>
```

Finally, we can implement parts from our original Example.html into the appropriate sections of a new file. Copy each section of your html file into the appropriate functions

```
<?php
function my_header(){
?>
<html>
<head><title>PHP Example</title>
 <link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
        <div class="main">  <!-- Total Outter page box -->
                <div class="header"><p>Header/logo here</p></div>
<?php
}

function my_navigation(){
?>

<div class="navigation"><p>Navigation bar here</p></div>

<?php
}

function my_content(){
?>

<div class="content"><p> Some content here</p></div>

<?php
}

function my_footer(){
?>
                <div class="footer"><p>Footer here</p></div>
        </div>
</body>
```

```
</html>
<?php
}
?>
```

Now that we have our layout completed, save the php file as: layout.php
Open another text document, this will be our index page.

```
<?php

include('layout.php');    /// include the layout.php file we just wrote, if the path is different, adjust it here

my_header();  /// call the header
my_navigation();  /// call the navigation
my_content();  /// call the content
my_footer();  /// call the footer

?>
```

Save the file as index.php You can view the page at http://vpn.local/wpie/index.php

EXAMPLE 2

prerequisite: previous example 1.

The bonus is, now you can make unlimited pages very quickly, while only changing what actually exists in my_content() function!.

Open another text document, this will be our new example.php page.

```
<?php

include('layout.php');    /// include the layout.php file we just wrote, if the path is different, adjust it here

my_header();  /// call the header
my_navigation();  /// call the navigation
my_content($_GET['type']);
my_footer();  /// call the footer

?>
```

Save it as example.php.
Open another text document, this will be our new func.php library. We'll simply fill this document with any extra functions we need.

```
<?php

function NewStuff(){

        echo “some new stuff displayed, on a new looking page”;
}

?>
```

Save it as func.php

Open layout.php from previous example 1. Add this line below the <?php opening line:

```
include('func.php');
```

Finally edit/replace the my_content function so it looks like this.

```
function my_content($someParam){

	if($someParam == “new”){
	echo '<div class=”content”>';
	NewStuff();
	echo '</div>';
	}else{
		echo '<div class=”content”><p> Some original content here</p></div>';
	}
}
```

Try it!
open a browser, visit: http://127.0.0.1/site/example.php?type=new or your site root path.