PHP Essentials 2013

Introduction

Filezilla FTP instructions:
Firewall ‘FileZilla server.exe’ and ‘FileZillaserver.exe’

What is PHP

- Server-side, scripting language
- Designed for use with HTML
- Provides more flexibility than HTML alone
- Syntax is similar to C, Java, Perl

Versions

- Version 1: 1994
    - CGI binaries in the C programming Language
- Version 2: 1995
    - Personal Home Page Tools
- Version 3: 1998
    - PHP: Hypertext Preprocessor
    - Supported, not actively maintained
- Version 4: 2000
    - Still actively support by updates
    - V4.4.6 on March 1, 2007
    - Supported/ Actively Maintained
- Version 5: 2004
    - The latest version, still be developed
    - Version 5.2.1 on February 8, 2007
    - Supported/ Actively Maintained

Why Use PHP

- Open Source / Free software
- Cross platform to develop, to deploy, and to use
- Powerful, robust, scalable
- Web development specific
- Can be object oriented, especially version 5
- Great documentation in many languages
    - www.php.net/docs.php
- Large, acrive developer community
    - 20 million websites

Requirements

- Web Servier (Apache 1.3)
- PHP (v 5.2.1)
- Database (5.0)
- Text Editor (Npp)
- Web Browser (Firefox)

Installation Overview

- XAMPP
  -
- PHP.ini
  -
- MYSQL root password
- Editor - KomodoEdit?

Issues

- MySQL Admin button inop
- Open Apache Admin
- MySQL.EXE command line tool
  - Must go to phpMyAdmin and create passwords
    - root@127.0.0.1
    - root@localhost
    - lortnoc@3L
  -

PHP Syntax

PHPInfo()

```
<?php phpinfo();?>
```

PHP Version 5.3.5

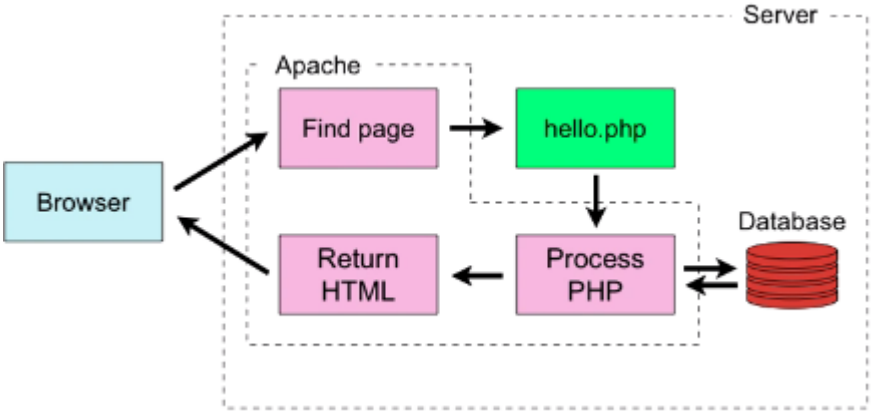
System	Windows NT E6510 6.1 build 7601 (Unknown Windows version Ultimate Edition Service Pack 1) i586
Build Date	Jan 6 2011 17:50:45
Compiler	MSVC6 (Visual C++ 6.0)

...

Send to HTML page

```
<?php echo "Hello World"; ?>
```

Operational Trail



Data Types

Variables

$item – lower case
$Item – title case
$myVariable – camelcase
$_ is PHP reserved variable

String Functions

Lowercase:	`strtolower($thirdString);`
Uppercase:	`strtoupper($thirdString);`
Uppercase first-letter:	`ucfirst($thirdString);`
Uppercase words:	`ucwords($thirdString);`
Length:	`strlen($thirdString);`
Trim:	`$fourthString = $firstString . trim($secondString);`
Find:	`strstr($thirdString, "brown");`
Replace by string:	`str_replace("quick", "super-fast", $thirdString);`
Repeat:	`str_repeat($thirdString,2);`
Make substring:	`substr($thirdString,5,10);`
Find position:	`strpos($thirdString,"brown");`
Find character:	`strchr($thirdString,"z");`

Numbers

Basic Math	`((1 + 2 + $var1) * $var2) / 2 - 5;`
	`$var1 = 3;`
	`$var2 = 4;`

+=:	`$var2 += 4;`
-=:	`$var2 -= 4;`
*=:	`$var2 *= 3;`
/=:	`$var2 /= 4;`
Increment:	`$var2++; echo $var2;`
Decrement:	`$var2--; echo $var2;`

Floating Point Numbers

$var1 = 3.14

Floating point:	$myFloat = 3.14;
Round:	round($myFloat, 1);
Ceiling:	ceil($myFloat);
Floor:	floor($myFloat);
Absolute Value:	abs(0-300);
Exponential:	pow(2,8);
Square root:	sqrt(100);
Modulo(/remainder):	fmod(20,7);
Random (any):	rand();
Random (min,max):	rand(1,10);

Arrays

Single Dimensional Array	$array1 = array(4,6,8,16,32)
Reading	echo $array[0]
Multi-Dimensional Array	$array2 = array(4,6,8,array("X","Y","Z"),16,32)
Reading	echo $array[3][1]
Associative Array	$array("first_name"=>"Jeff", "last_name"=>"Davis")
Reading	echo $array3["first_name"]
Print Entire Array	print_r($array2)
Can use with html <pre></pre> tags	<pre><?php print_r($array2); ?></pre>

Array Functions

Count	count($array1)
Max Value	max($array1)
Min Value	min($array1)
Sort	sort($array1)
Reverse Sort	rsort($array1)
Implode(insert separators)	implode(" * ", $array1)
Explode(split at string)	explode(" * ", $string)
In array(bool)	in_array(3,$array1)

Boolean

$boo11 = true
$bool2 = false
$boolNULL = NULL;
$boo3 = 0 - empty for Booleans
$bool4 = “0” - empty for booleans
is_set($variable1)
!is_set($variable2)

Type Switching

Switching from int to string	$var1 = "2 brown foxes", $var2 = $var1 + 3;
gettype	gettype($var1)

	= string, integer, bool, array
settype	settype($var2, "string") = string
is_array is_bool is_float is_int is_null is_numeric is_string	

Constants

Must be ALL CAPS

```
define("MAX_WIDTH", 300)
echo MAX_WIDTH
```

Control Structures

Operators

==, <=, >=, <>

Ternary

$result = (exp1) ? (expr2) :(expr3);
if exp1 is true, use expr2
otherwise, use expr3;

If, elseif, else

if

```
if ($a > $b){
}
```

elseif

```
if ($a > $b){
} elseif ($a > $c){
}
```

else

```
if ($a > $b){
} elseif ($a > $c){
}else{
}
```

Logical Operators

AND, &&	if (($a > $b) && ($c > $d))
OR, ||	if (($a > $b) && ($c > $d))

Switch

```
switch($a){
        case 0:
                echo: "a equals 0";
                break;
        case 1:
                echo: "a equals 1";
                break;
        default;
                echo: "a not known";
                break;
}
```

While Loop

```
count = 0;
while ($count <= 10){
        echo $count;
        $count ++;
}
```

For Loop

```
for (init; test; each){
        statement;
}

for($count=0; $count<=10; $count++){
}
```

Foreach Loop

Arrays

```
foreach ($array as $value){
}


$keys = array(1,2,3,4,5);
foreach ($keys as $key){
```

```
        echo $key
}
```

Associative Arrays

```
foreach ($array as $key => $value)
        Statement;
```

Example1

```
// using each key => value pair
// good for when you need the index with the value in the foreach
$ages = array(1,2,3,4,5);

foreach ($ages as $position => $age){
        echo $position . ": " . $age . "<br />";
}
```

0: 1
1: 2
2: 3
3: 4
4: 5

Example2

```
$ages = array(
        "John" => 1,
        "Jane" => 2,
        "Kelly" => 3
);

foreach ($ages as $name => $age){
        echo $name. ": " . $age . "<br />";
}
```

John: 1
Jane: 2
Kelly: 3

Example3

```
$prices = array(
        "New Computer"=>2000,
        "Training"=>25,
        "Learning PHP"=>"priceless"
        );

foreach($prices as $key->$value) {
                if(is_int($value)){
                        echo $key . ": $" . $value . "<br />";
                } else {
                        echo $key . ":  " . $value . "<br />";
                }
```

```
        }
```

New Computer: $2000
Training: $25
Learning PHP: priceless

Continue

Once condition is met, continue with the loop

```
Continue;
```

Break

Once condition is met, break out of the current loop

```
Break;
```

```
for ($count=0; $count<=10; $count++){
        echo $count . ", ";
}
```

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,

```
for ($count=0; $count<=10; $count++){
        echo $count;
        if ($count < 10) {echo ", ";}
}
```

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

```
for ($count=0; $count<=10; $count++){
        echo $count;
        if ($count == 10) {break;}
        echo ", ";
}
```

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Pointers

current($array1)	current pointer
next($array1)	move pointer next
reset($array1)	set pointer to start

```
// Arrays have pointers that point to a position in the array
// We can use current, next and reset to manipulate the pointer
echo "1: " . current($ages) . "<br />";
next($ages);
echo "2: " . current($ages) . "<br />";
reset($ages);
```

```
echo "3: " . current($ages) . "<br />";
```

1: 4
2: 8
3: 4

```
// while loop that moves the array pointer
// It is important to understand this type of loop before working with
databases
// $age is a POINTER, not a VARIABLE
// returns TRUE if $age was able to be assigned
//        (pointer $current able to move next()
while ($age = current($ages)) {
        echo $age . ", ";
        next($ages); //move pointer to next position in array
}
```

4, 8, 15, 16, 23, 42,

Functions

Functions can be placed anywhere, not necessarily BEFORE their call

```
function name($arguments){
        statement;
}
```

```
function say_hello(){
        echo "Hello World!<br />";
}
say_hello();
```

Hello World!

```
function say_hello2($word){
        echo "Hello {$word}!<br />";
}
say_hello2("World");
say_hello2("Everyone");
```

Hello World!
Hello Everyone!

```
function add_subt($val1, $val2){
        $add = $val1 + $val2;
        $subt = $val1 - $val2;
        $result = array($add, $subt);
        return $result;

echo "Calling 'add_subt' <br />";
$result_array = add_subt(10,5);
echo "Add: " . $result_array[0] . "<br />";
echo "Substract: " . $result_array[1];
```

```
Sum is: 7
Result is: 7Calling 'add_subt'Add: 15
Substract: 5
```

Globals

```
// global by default
$bar = "Outside";

function foo(){
        $bar_local = "local variable";
        global $bar;
        $bar = "Inside";
        return $var;
}

// pull local variable into a global one
$bar = foo($bar);
```

Default Values

```
function paint($color="red"){
        echo "Painting with color: " . $color . "<br />";
}

echo "Calling 'paint()' <br />";
paint();
echo "Calling 'paint(blue)' <br />";
paint("blue");
```

```
Calling 'paint()'
Painting with color: red
Calling 'paint(blue)'
Painting with color: blue
```

Troubleshooting & Debugging

- Display_errors/error_reporting
- Sever Error Logs
    - WAMP: C:\wamp\logs
    - XAMPP: C:\xampp\apache\logs
- typos, semicolons, closing braces
- = v.s. ==

PHP commands

```
echo $variable;         //variable value
print_r($array); //readable array info
gettype($variable); //variable type
var_dump($variable); //variable type and value
get_defined_vars(); //array of defined variables
```

Building Dynamic Web Pages

URLs/Links	GET
Forms	POST
Cookies	COOKIE

$_GET[''] & urlencode(), urldecode

Information can be sent in the link request and retrieved by the receiving page

```
<a href="secondpage.php?id=2">Second Page</a><br />
<a href="secondpage.php?name=Kevin Lastname&id=42">Second Page -
name=Kevin</a><br />
<a href="secondpage.php?name=<?php echo urlencode("Kevin&") ?
>&id=42">Second Page - Kevin& using urlencode()</a>
<br />---This link uses urlencode() and receiver will need urldecode
<br />---<?php echo urlencode("Kevin&") ?>
```

```
print_r($_GET);
$id    = $_GET['id'];
        if (!$_GET['name']){
                $name  = "none";
        } else {
                $name = $_GET['name'];
                $name2  = urldecode($_GET['name']);
        }
```

<u>First Page</u> Array ( )
Name RAW:none ID:
Name RAW:none
Name: urldecode():
Name: urlencode():
Name: htmlspecialchars():none

Encoding

Code

```
            <?php
                        $url_page = 'php/create/page/url.php';
                        $param1 = 'this is a string';
                        $param2 = '"bad"/<>character$';
                        $linktext = "<Click> & you'll see";
                ?>
        <?php
                        // this gives you a clean link to use
```

```
                        $url = "http://e6510/";
                        $url .= rawurlencode($url_page);
                        $url .= "?param1=" . urlencode($param1);
                        $url .= "&param1=" . urlencode($param2);

                        // htmlspecialchars esacepes any html that
                        // might do bad things to your html page
                ?>
        <a href="<?php echo $url; ?>">
                <?php echo htmlspecialchars($linktext);?>
         </a>
```

Ouput

<Click> & you'll see

Forms

```
<form action="process.php" method="post">
        Username: <input type="text" name="username" value="" />
        Password: <input type="password" name="password" value="" />
        <input type="submit" name="submit" value="Submit" />
</form>
```

Username: Password: Submit

```
<?php
$username = $_POST['username'];
$password = $_POST['password'];

echo "{$username}: {$password}";
?>
```

MyName: 1234

Cookies

- setcookie() creates a cookie for a set time

```
<?php
        setcookie('test', 45, time()+(60*60*24*7));
?>
```

Site	Cookie Name
e6510	test

- Access the cookie with the $_COOKIE[''] variable

```
<?php
        $var1 = $_COOKIE['test'];
```

```
        echo $var1;
?>
```

45

Removing a cookie

- setcookie('test', 0, time()-(60*60*24*7));

Check for COOKIE presence with isset()

```
<?php
    // init $var1 in case COOKIE is empty/missing
    $var1 = 0;
    if (isset($_COOKIE['test'])){
        $var1 = $_COOKIE['test'];
    }
    echo $var1;
?>
```

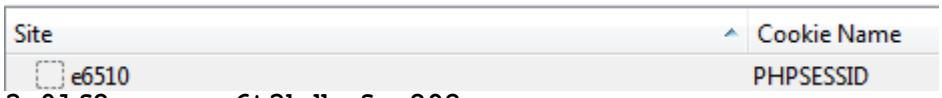
Sessions

- Sessions store data on the server under a session ID.
- The client computer stores the Session ID in a cookie called PHPSESSID.

```
<?php
        // must occur before any HTML
        session_start();

?>
```

- Session ID is stored in a cookie on client computer



3r01f8ggvepvu6t3hdhrfqs892

```
<html>
<body>

<?php
        $_SESSION['name'] = "kevin";
?>
<?php

        $name = $_SESSION['name'];
        echo $name;
?>

</body>
```

```
</html>
```

- Session files on the server can take up space over time either with multiuser sites or sessions that store large amounts of data

Other SUPER GLOBALS

$_SERVER
$_HOST

Headers and Page Redirection

- Server makes a request to the php page, it will send headers to browser before HTML data
- Headers precede HTML data

```
header(header information)
header("Content-type: application/vnd.ms-excel; name='excel'");
header("Content-disposition: attachment; filename=myfile.xls");
```

```
<?php
    header("HTTP/1.0 404 Not Found");
    exit;
    // headers must occur before any HTML
?>
```

Redirect

```
<?php
        // headers must occur before any HTML
        // this is how you redirect a page
        // 302 Redirect
        header("Location: basic.html");
?>
<?php
        // this is how you return a 404 error
        //header("HTTP/1.0 404 Not Found");
        exit;

?>
```

Output buffering

- Instead of running into problems with php before HTML, Server will store up all html code until all data parsed and php code handled.
- If output buffering is enabled, the header() command will still work if within the HTML body tag.
- PHP.ini controls this feature.

```
; Output buffering is a mechanism for controlling how much output data
; (excluding headers and cookies) PHP should keep internally before pushing
that
; data to the client. If your application's output exceeds this setting,
PHP
; will send that data in chunks of roughly the size you specify.
; Turning on this setting and managing its maximum buffer size can yield
some
; interesting side-effects depending on your application and web server.
; You may be able to send headers and cookies after you've already sent
output
```

```
; through print or echo. You also may see performance benefits if your
server is
; emitting less packets due to buffered output versus PHP streaming the
output
; as it gets it. On production servers, 4096 bytes is a good setting for
performance
; reasons.
; Note: Output buffering can also be controlled via Output Buffering
Control
;   functions.
; Possible Values:
;   On = Enabled and buffer is unlimited. (Use with caution)
;   Off = Disabled
;   Integer = Enables the buffer and sets its maximum size in bytes.
; Note: This directive is hardcoded to Off for the CLI SAPI
; Default Value: Off
; Development Value: 4096
; Production Value: 4096
; http://php.net/output-buffering
output_buffering = 4096
```

Usage

```
ob_start();
<html>
        <head>
                <title>My Page</title>
        </head>
        <body>
                Page Content
        </body>
<html>
ob_end_flush();
```

Include and Require

- include()
  - include a php file
  - no error if not found
- include_once()
  - include a php file
  - but only once
  - Good for php function files.
- require()
  - include a php file
  - throw error if not found
- require_once()
  - include a php file
  - but only once
  - throw error if not found
  - Good for php function files.

included_func.php:

```
functions
```

includes.php

```
<?php
        include("included_func.php");
?>
```

Output:

```
functions
```

Example2: Calling included function

included_func.php:

```
<?php
        function hello($name){
                echo "Hello {$name}";
        }
?>
```

includes.php

```
<?php
        include("included_func.php");
?>
<?php
        hello("Eveyone");
?>
```

Output:

```
Hello Eveyone
```

Databases

CRUD

Create, Read, Update, Delete

Read: SQL SELECT

```
SELECT * FROM table
```

```
WHERE colum1 = 'some_text'
ORDER BY column, column2 ASC;
```

WRITE: SQL INSERT

```
INSERT INTO table (column1, colum2, column3)
VALUES (val1, val2, val3);
```

UPDATE: SQL UPDATE

```
UPDATE table
SET column1 = 'some_text'
WHERE id = 1;
```

DELETE: DQL DELETE

```
DELETE FROM table
WHERE id=1;
```

Creating a database, table and table fields

```
CREATE DATABASE widget_corp;
USE widget_corp;
CREATE TABLE subjects (
     id int(11) NOT NULL auto_increment,
     menu_name varchar(30) NOT NULL,
     position int(3) NOT NULL,
     visible tinyint(1) NOT NULL,
     PRIMARY KEY (id)
);
```

Adding a row into the table

```
INSERT INTO subjects ( menu_name, position, visible)
VALUES( 'About Widget Corp', 1, 1);

INSERT INTO subjects ( menu_name, position, visible)
VALUES( 'Products', 2 , 1);
```

Reading from the table

```
SELECT * FROM subjects;
```

Chapter 12

Create Blueprint for Application

Create Database: phpMyAdmin

Database: “widget_corp”
Type: ISAM

subjects

Name	Type	Collation	Null	AI	Primary
id	int(11)			x	x
menu_name	varchar(30)	utf8_general_ci			
position	int(3)				
visible	tinyint(1)				

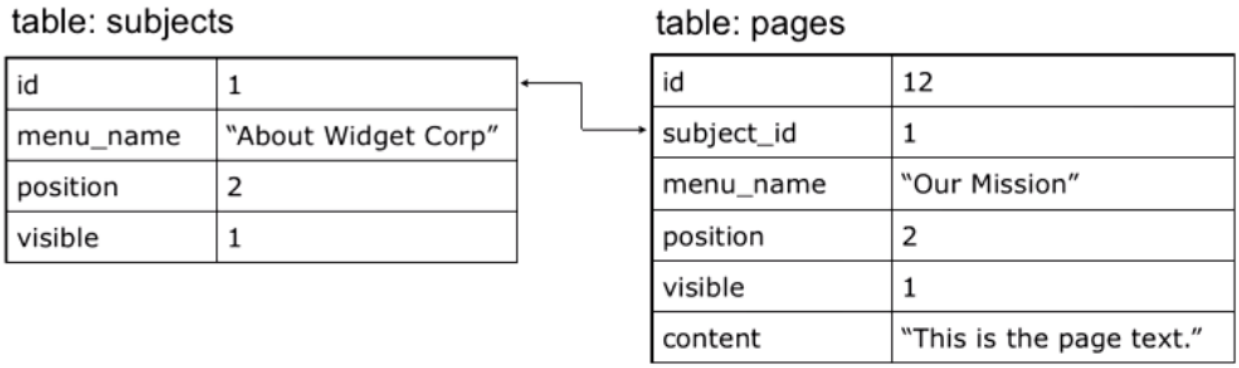
pages

Name	Type	Collation	Null	AI	Primary
id	int(11)			x	x
subject_id	int(11)				
menu_name	varchar(30)	utf8_general_ci			
position	int(3)				
visible	tinyint(1)				
content	text				

Users

Name	Type	Collation	Null	AI	Primary
id	int(11)			x	x
user_name	varchar(50)	utf8_general_ci			
hashed_password	varchar(40)	utf8_general_ci			

Relational Database



Project Setup

Database Connection

CRUD

Create

Read

Update

Delete

Menu Generation

Query, Result, Result_set, Item loop

Query ORDER BY ….ASC

Chapter 13

Escaping characters for SQL statements

These need escape slashes to be included in a string

- Quote(‘)
- Double Quote (”)
- Backslash(\)
- NUL(NULL Byte)

```
UPDATE… “That’s all” = “That’’s all”
```

- Two single quotes will be entered in the SQL statement(phpMyAdmin) or will be required if driven by code

mysql_real_escape_string()

- Escapes special characters in the *`unescaped_string`*, taking into account the current character set of the connection so that it is safe to place it in a mysql_query(). If binary data is to be inserted, this function must be used.
- **mysql_real_escape_string()** calls MySQL's library function mysql_real_escape_string, which prepends backslashes to the following characters: *\x00, \n, \r, \, ', "* and *\x1a*.
- This function must always (with few exceptions) be used to make data safe before sending a query to MySQL.
- Newer function

```
$query = "INSERT INTO subjects (
```

```
                menu_name, position, visible
        ) VALUES (
                '{$menu_name}', {$position},
{$visible}
        )";
```

- Quotes can cause problems with submitting to DB

Addslashes()

- Returns a string with backslashes before characters that need to be quoted in database queries
-

Array[] (Append to Array)

- Array[] will append to the array, taking the last indice+1

```
$thisArray[13] = "thirteen";
$thisArray[] = "fourteen";
```