Web Development with PHP

Part 1 - Pure PHP

1

PHP History

- Rasmus Lerdorf (Programmer) in 1995
  - Personal Home Page
- PHP: Hypertext Preprocessor
- Version 5.5 (2013-06-20)
- Version 5.6 (2014-08-28)
- Version 7.4 (2020-09-29)
- Version 8.1 (2022-09-29)

2

Why PHP?

- Quick and cheap?
- General purpose sever-side scripting
- Interpreted vs Compiled
  - Bytecode (Zend Engine as of PHP 4)
- Put the "P" in LAMP
  - WAMP/WIMP
  - MAMP
- To use or not to use...

3

Why PHP? (continue)

- Basic programming concepts
- Server-side vs client-side programming
- PaaS offerings
  - Google App Engine
    https://developers.google.com/appengine/
  - Amazon Elastic Beanstalk
    (http://aws.amazon.com/elasticbeanstalk/)
  - Microsoft Azure
    (http://azure.microsoft.com)

4

Website or Web App

- Web app → App + DB
- Website → info only
- Web server (Apache/Tomcat) → localhost
  - Web folder

5

Tools of the Trade

- **XAMPP** → Local server
- **Sublime, VS Code** → IDE
- **Firefox, Chrome** → Internet Browser
  - www.mozilla.org

6

Local Server

- Install **XAMPP**
- Move the folder to location: C:\
- Note (web content can be placed as follow)
  - Website content: C:\xampp\htdocs\
  - E.g. C:\xampp\htdocs\helloworld1
  - E.g. C:\xampp\htdocs\helloworld2

7

IDE

- Install Editor
- Start a new project:
  - Location: C:\xampp\htdocs

8

First PHP page

- In the index.php file, enter

```
<?php
    echo(“Hello World!”);
?>
```

These tags start and end a PHP script.
PHP code goes in the middle.

9

Embedding PHP within HTML

```
<html>
    <head>
        <title>Welcome to PHP Web Programming</title>
    </head>
    <body>
        <font color="0000ff">PHP code appears as:</font>
        <p>
            <?php
                echo("Hello World!");
            ?>
        </p>
    </body>
</html>
```

10

PHP Comments

```
<html>
  <head>
        <title>Welcome to PHP Web Programming</title>
  </head>
  <body>
        <font color="0000ff">PHP code appears as:</font>
        <p>
        <?php
        // This is a comment
        /* The echo function outputs the string "Hello World!"
            You can add any strings of your choice.
        */
                echo("Hello World!");
                phpinfo(); // is the horse you are riding male or female?
        ?>
        </p>
  </body>
</html>
```

11

PHP Variables

- Starts with a $ sign
- Starts with a letter or underscore
- Contains letters, numbers, and the underscore character
- Variable name is case sensitive
- Can not contain spaces
- it is not used elsewhere (like "print")

- Loosely typed language
  - variables need not be declared before adding value to it

12

Data Types

- What are they?
  - Variables hold values
    (e.g. input values in log in forms)
  - Use data types depends on the kind of input
    (e.g. for names → string type, phone → integer)
- **Boolean and NULL:** true or false
- **Integer**
- **Floating point**
- **String**
- **Arrays:** depends on the size and type of the data type

13

Boolean

- Either **TRUE** or **FALSE**

```php
<?php
    $auth = True;
    echo($auth);
    if($auth == True)
    {
        echo("variable auth is " . $auth);
    }
?>
```

14

Integers

- Size of the integer depends on the system
- PHP only offers signed integer
- Note: **PHP_INT_SIZE** and **PHP_INT_MAX**

```php
<?php
  $myPay = 1234;   // a positive number
  $myDebt = -1234;        // a negative number
  $myOcto = 0123;  // octal number (equivalent to 83 decimal)
  $myHex = 0xff;   // hexadecimal number (equivalent to 255 decimal)

  echo("<br/>My pay is " . $myPay);
  echo("<br/>My debt is " . $myDebt);
  echo("<br/>My octal number value in decimal is " . $myOcto);
  echo("<br/>My hexadecimal number value in decimal is " . $myHex);

  // Note that we are dealing with integer limit
  echo("<br/>The size of PHP int on my system is " . PHP_INT_SIZE);
  echo("<br/>The max value of PHP int on my system is " . PHP_INT_MAX);
?>
```

15

Floating Points

- Size of the floating point depends on the system
- Note: Be careful when comparing floating points

```php
<?php
  $myPay = 1234.326;             // decimal number
  $myDebt = -1234.326;           // a negative number
  echo("<br/>My pay is " . $myPay);
  echo("<br/>My pay is " . number_format($myPay,2));
  echo("<br/>My pay is \$" . number_format($myPay,2));

  $myPay   = 1234.324;
  echo("<br/>My pay is \$" . number_format($myPay,2));

  $myFriendPay = 1.9e3;          // 1.9 x 1000
  $myCommission = 7E-3;          //   7 x 0.001
  echo("<br/>" . $myFriendPay . " " . $myCommission);
?>
```

16

Strings

- Escape characters
  - Backslash \\
  - Dollar sign \$
  - Double quote \”

```
<?php
  echo("<br/>this is a simple string");
  echo('<br/>Some info
        more info
        some more info');
  echo('<br/>"I\'ll be back"');
  // C:\*.*?
  echo('<br/>Are you looking for C:\\*.*?');
  // C:\\*.*?
  echo('<br/>Are you looking for C:\\\\*.*?');
?>
```

17

17

Variable Scope

- Local
  - can only be accessed within the local scope
- Global
  - can be accessed from any part of the script that is not inside a function
  - Differentiate using **global** keyword
  - Array of global variables in **$GLOBALS[index]**
- Parameter
  - Declared in a parameter list in the function declaration
- Static
  - Can be accessed to obtain the value it contained from the last time the function was called

18

Coding for scope

```php
<?php
  function outputResult0(){
        // can not access a variable outside of the function
        echo("The value is " . $myResult . ".<br/>");
  }
  function outputResult1(){
        global $myResult;
        echo("The value is " . $myResult . ".<br/>");
  }
  function outputResult2(){
        echo("The value is " . $GLOBALS['myResult'] . ".<br/>");
        echo("The value is " . $GLOBALS["myResult"] . ".<br/>");
  }
  outputResult0();
  outputResult1();
  outputResult2();
?>
```

19

Data Types

- **Constants**

```php
<?php
  define("MAXSIZE", 100);
  echo (MAXSIZE . "<br/>");
  echo (constant("MAXSIZE") . "<br/>");

  const MINSIZE = 2;
  echo (MINSIZE . "<br/>");
  echo (constant("MINSIZE") . "<br/>");
?>
```

20

Data Types

- Type switching and casting
  - http://php.net/manual/en/language.types.type-juggling.php
  - http://php.net/manual/en/language.types.type-juggling.php#language.types.typecasting
  - http://php.net/manual/en/function.settype.php

```
<?php
  $foo = "0";  // $foo is string (ASCII 48)
  var_dump($foo);

  $foo += 2;   // $foo is now an integer (2)
  var_dump($foo);

  $foo = $foo + 1.3;  // $foo is now a float (3.3)
  var_dump($foo);

  $foo = 5 + "10 Little Piggies"; // $foo is integer (15)
  var_dump($foo);

  $foo = 5 + "10 Small Pigs";     // $foo is integer (15)
  var_dump($foo);
?>
```

21

Arithmetic Operators

+ (addition), - (subtraction), * (multiplication), / (division), % (modulus), = (assignment), . (concatenation)

```
<?php
  $x = 29; $y = 10;
  $z = $x + $y;
  echo("<br/>" . $z);

  echo "<br/>";
  $z = $x / $y;
  echo $z;

  echo "<br/>";
  $z = $y * $y * $x;
  echo $z - 1500;
  echo "<br/>";
?>
```

22

Assignment Operators

- += (addition assignment), -= (subtraction assignment)
- *= (multiplication assignment), /= (division assignment)
- %= (modulus assignment), .= (concatenation assignment)
- ++ (increment), -- (decrement)

```php
<?php
$x = 1;
$x++;
echo $x . " ";
$y = 5;
$y *= $x;
echo $y . " ";
$z = 180;
$z /= --$y;
echo $z;
?>
```

23

Comparison Operators

Operator	Name	Description	Example
x == y	Equal	True if x is equal to y	5==8 returns false
x === y	Identical	True if x is equal to y, and they are of same type	5==="5" returns false
x != y	Not equal	True if x is not equal to y	5!=8 returns true
x <> y	Not equal	True if x is not equal to y	5<>8 returns true
x !== y	Not identical	True if x is not equal to y, or they are not of same type	5!=="5" returns true
x > y	Greater than	True if x is greater than y	5>8 returns false
x < y	Less than	True if x is less than y	5<8 returns true
x >= y	Greater than or equal to	True if x is greater than or equal to y	5>=8 returns false
x <= y	Less than or equal to	True if x is less than or equal to y	5<=8 returns true

24

24

Logical Operators

Operator	Name	Description	Example
x and y	And	True if both x and y are true	x=6, y=3 (x < 10 and y > 1) returns true
x or y	Or	True if either or both x and y are true	x=6, y=3 (x==6 or y==5) returns true
x xor y	Xor	True if either x or y is true, but not both	x=6, y=3 (x==6 xor y==3) returns false
x && y	And	True if both x and y are true	x=6, y=3 (x < 10 && y > 1) returns true
x || y	Or	True if either or both x and y are true	x=6, y=3 (x==5 || y==5) returns false
! x	Not	True if x is not true	x=6, y=3 !(x==y) returns true

25

25

Arrays

- **Numeric arrays**

```php
<?php
  $cars=array("Honda","Lexus","BMW","Toyota");
  // $cars[0]=" Honda ";
  // $cars[1]=" Lexus ";
  // $cars[2]="BMW";
  // $cars[3]="Toyota";
  echo "<br/>" . $cars[0] . ",  " . $cars[1]. " and " . $cars[3] . " are Japanese cars.";
?>
```

- **Associative arrays**

```php
<?php
  $ages = array("Ko Phyo"=>32, "Ko Naing"=>30, "Ko Myo"=>34);
  // $ages['Ko Phyo'] = 32;
  // $ages['Ko Naing'] = 30;
  // $ages['Ko Myo'] = 34;
  echo "<br/>Ko Myo is " . $ages['Ko Myo'] . " years old.";
?>
```

26

26

Arrays

- Multidimensional arrays

```php
<?php
  $families = array("Ko Phyo"=>array("Ko Naing", "Ko Myo"),
                    "Peter"=>array( "Jay"),
                    "Ko Linn"=>array("Ko Hein", "Ko Zin", "Ko Htet"));

  echo "<br/>Is " . $families['Ko Linn'][1] . " a part of Ko Phyo family?";
?>
```

27

27

Arrays

- Insert

```
array_push($my_array, "mango");
array_unshift($my_array, "mango");
```

- Remove

```
unset($my_array[0]);
unset($my_array['mango']);
```

28

28

Arrays

- Assignment

```
$my_array1 = array(20,30);
$my_array2 = $my_array1;
$my_array2[] = 100;

$my_array3 = &$my_array1; // $my_array3 is a reference to $my_array1
$my_array3[] = 500;
```

29

Camel-Case vs Under-score

- camelCase
- Under_score

30

Branching (Conditional Statements)

- Program control goes top-down
- To divert the flow of the program control, you can use conditional statements
  - If
  - Switch

31

IF

- If the statement is **true**, then **do** something
- If the statement is **false**, then **don't** do it.

```
<?php
  $myDay = date("D");
  if($myDay == "Fri")
  {
      echo("Thank God it's Friday!");
  }
?>
```

32

IF-else

```php
<?php

  $myDay = date("D");
  if($myDay == "Fri")
  {
        echo("Thank God it's Friday!");
  }
  else
 {
        echo("Great day to do PHP class homework!");
  }

?>
```

33

IF-elseif-else

```php
<?php
  if($myDay == "Fri")
  {
        echo("Thank God it's Friday!");
  }
  else if($myDay == "Sat")
  {
        echo("Reading PHP class notes.");
  }
  else if($myDay == "Sun")
  {
        echo("Doing PHP class homework.");
  }
  else
  {
        echo("Thinking about PHP class.");
  }
?>
```

34

Switch

```php
<?php
  switch($myDay)
  {
        case "Mon":
        case "Tue":
        case "Wed":
        case "Thu":
                echo("Thinking about PHP class.");
                break;
        case "Fri":
                echo("Thank God it's Friday!");
                break;
        case "Sat":
                echo("Reading PHP class notes");
                break;
        case "Sun":
                echo("Doing PHP class homework.");
                break;
  }
?>
```

35

Switch - default

```php
<?php
  $myPass = "EP";
  switch($myPass)
  {
        case "WP":
                echo("Work Permit");
                break;
        case "SP":
                echo("S Pass");
                break;
        case "EP":
                echo("Employment Pass");
                break;
        case "PEP":
                echo("Personalised Employment Pass");
                break;
        default:
                echo("You don't work in Singapore or you are a PR/Citizen.");
  }
?>
```

36

Iteration

- Causing the program control to perform desired operations **repeatedly**
  - while
  - do-while
  - for
  - foreach
- Constructs provided by the language to iterate
  - Initial state
  - Conditional expression
  - Progression (increment/decrement)

37

while

- **A raw form of iteration**

```
<?php
  $myLand = 6;
  $lengthCounter = 1;

  echo("My first piece of land:<br/>");
  while($lengthCounter < $myLand)
  {
        echo($lengthCounter . "****<br/>");
        $lengthCounter++;
  }

  // shall we use "<" or "<="
?>
```

Initial State

Conditional Expression

Progression (increment/decrement)

38

do-while

- **A raw form of iteration**

```
<?php
  $myLand = 6;
  $lengthCounter = 1;

  echo("My first piece of land:<br/>");
  do
  {
        echo($lengthCounter . "****<br/>");
        $lengthCounter++;
  } while($lengthCounter < $myLand)

  // shall we usee "<" or "<="
?>
```

Initial State

Progression (increment/decrement)

Conditional Expression

39

for

- **A more organized form of iteration**

```
<?php
  $myLand = 6;

  echo("My first piece of land:<br/>");

  for($lengthCounter = 1; $lengthCounter < $myLand; $lengthCounter++)
  {
       echo($lengthCounter . "****<br/>");
  }
?>
```

Initial State

Conditional Expression

Progression (increment/decrement)

40

foreach

- **An organized form of iteration**

```
<?php
  $myCars = array("Honda","Lexus","BMW","Toyota");

  echo("My cars:<br/>");
  foreach($myCars as $car)
  {
       echo($car . "<br/>");
  }
?>
```

41

Functions

- name a function to reflect what it **does**
- function name can start with a **letter** or **underscore**

```
<?php
  // attempt to call HelloWorld function
  HelloWorld();
  function HelloWorld()
  {
       echo("Hello World!<br/>");
       echo("Here I come.<br/>");
  }
  HelloWorld();
?>
```

42

42

Functions - Parameters

- Parameter(s) can be supplied to pass information

```
<?php
     function Greet($personName, $personAge)
     {
       $age = $personAge - 5;
       echo("Hi " . $personName . "!<br/>");
       echo("You don't look a day over ". $age . ".<br/>");
     }
     Greet("Soe", 29);
?>
```

43

Functions – Return value

- A function can provide a return value

```
<?php
  function add($firstNumber, $secondNumber)
  {
       return $firstNumber + $secondNumber;
  }
  function outputResult($value)
  {
       echo("The value is " . $value . ".<br/>");
  }

  $totalValue = add(5, 8);
  outputResult($totalValue);
?>
```

44

44

Include and Require

- includes and evaluates the specified file
- *require* is identical to *include* except upon failure it will also produce a fatal **E_COMPILE_ERROR** level error.
- In other words, it will halt the script whereas *include* only emits a warning (**E_WARNING**) which allows the script to continue.
  - http://php.net/manual/en/function.include.php
  - http://php.net/manual/en/function.require.php

```
vars.php
<?php
  $color = 'green';
  $fruit = 'apple';
?>

test.php
<?php
  echo "A $color $fruit"; // A
  include 'vars.php';
  echo "A $color $fruit"; // A green apple
?>
```

45

Error Handling

- Filters
- Simple "die()" statements
- Custom errors handling
- Error triggers
- Error reporting

46

Filters

- Validation
  - check if the data meets certain qualifications
  - Does not change the data
  - Options
    - FILTER_VALIDATE_BOOLEAN
    - FILTER_VALIDATE_EMAIL
    - FILTER_VALIDATE_FLOAT
    - FILTER_VALIDATE_INT
    - FILTER_VALIDATE_IP
    - FILTER_VALIDATE_URLFILTER_VALIDATE_REGEXP

47

Filters

```
$host_ip_1 = "127.0.0.1";
$host_ip_2 = "127.42";
if(filter_var($host_ip_1, FILTER_VALIDATE_IP))
   echo("This host_ip_1 is considered valid.");
else
   echo(" This $host_ip_1 is considered not valid.");

if(filter_var($host_ip_2, FILTER_VALIDATE_IP))
   echo("This $host_ip_2 is considered not valid.");
else
   echo("This $host_ip_2 is considered not valid.");
```

```
$to_email_1 = "tmma@ivs.com";
$to_email_2 = "some_email";
if(filter_var($to_email_1,FILTER_VALIDATE_EMAIL))
   echo("This $to_email_1 is considered valid.");
else
   echo("This $to_email_1 is considered not
valid.");

if(filter_var($to_email_2, FILTER_VALIDATE_EMAIL))
   echo("This $to_email_2 is considered not
valid.");
else
   echo("This $to_email_2 is considered not
valid.");
```

48

Filters

- Sanitization
  - may alter it by removing undesired characters (remove characters that are inappropriate for an email address)
  - does not validate the data
  - Options
    - FILTER_SANITIZE_EMAIL
    - FILTER_SANITIZE_ENCODED
    - FILTER_SANITIZE_FULL_SPECIAL_CHARS
    - FILTER_SANITIZE_MAGIC_QUOTES
    - FILTER_SANITIZE_NUMBER_FLOAT
    - FILTER_SANITIZE_NUMBER_INT
    - FILTER_SANITIZE_SPECIAL_CHARS
    - FILTER_SANITIZE_STRINGS
    - FILTER_SANITIZE_STRIPPED
    - FILTER_SANITIZE_URL

49

Filters

```
$to_email_1 = "soe@pyinnyar.com";
$to_email_2 = "some_email";
$to_email_3 = "(thet.s@newwestgroup.org)";

$to_email_1_s = filter_var($to_email_1, FILTER_SANITIZE_EMAIL);
test_validity($to_email_1_s);

$to_email_2_s = filter_var($to_email_2, FILTER_SANITIZE_EMAIL);
test_validity($to_email_2_s);

$to_email_3_s = filter_var($to_email_3, FILTER_SANITIZE_EMAIL);
test_validity($to_email_3_s);
```

```
function test_validity($input_email)
{
   if(filter_var($input_email, FILTER_VALIDATE_EMAIL))
      echo("$input_email is considered valid.<br/>");
   else
      echo("$input_email is considered not valid.<br/>");
}
```

50

Custom Error Handling

- A special/custom function to be called when an error occurs
- must be able to handle a minimum of two parameters
  - error level and error message
- can accept up to five parameters
  - file, line-number, and the error context

```
error_function(error_level, error_message, error_file, error_line, error_context)
```

51

Custom Error Handling

```
function myCustomError($errno, $errstr)
{
   echo "<b>Error:</b> [$errno] $errstr<br />";
   echo "Ending Script";
   die();
}

set_error_handler("myCustomError");
```

52

Error Handling

Parameter	Description
error_level	Required. Specifies the error report level for the user-defined error. Must be a value number.
error_message	Required. Specifies the error message for the user-defined error
error_file	Optional. Specifies the filename in which the error occurred
error_line	Optional. Specifies the line number in which the error occurred
error_context	Optional. Specifies an array containing every variable, and their values, in use when the error occurred

53

Error Handling

Value	Constant	Description
2	E_WARNING	Non-fatal run-time errors. Execution of the script is not halted
8	E_NOTICE	Run-time notices. The script found something that might be an error, but could also happen when running a script normally
256	E_USER_ERROR	Fatal user-generated error. This is like an E_ERROR set by the programmer using the PHP function trigger_error()
512	E_USER_WARNING	Non-fatal user-generated warning. This is like an E_WARNING set by the programmer using the PHP function trigger_error()
1024	E_USER_NOTICE	User-generated notice. This is like an E_NOTICE set by the programmer using the PHP function trigger_error()
4096	E_RECOVERABLE_ERROR	Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle (see also set_error_handler())
8191	E_ALL	All errors and warnings (E_STRICT became a part of E_ALL in PHP 5.4)

54

Error Trigger

- Can be triggered using trigger_error()
- Second parameter to specify the error level
  - **E_USER_ERROR** - Fatal user-generated run-time error. Execution is halted.
  - **E_USER_WARNING** - Non-fatal user-generated run-time warning. Execution of the script is not halted.
  - **E_USER_NOTICE** - Default. User-generated run-time notice.

55

Error Trigger

```
function myCustomError($errno, $errstr)
{
   echo "<b>Error:</b> [$errno] $errstr<br />";
   echo "Ending Script";
   die();
}

// set error handler and the level
set_error_handler("myCustomError", E_USER_WARNING);

// trigger error
$yellow_cards = 2;
if ($yellow_cards > 1)
{
   trigger_error("Number of yellow cards can be 1 or below.",E_USER_WARNING);
}
```

56

Exception

- Change the normal flow of the code execution if a specified error (exceptional) condition occurs
- Advantages
  - The current code state is saved
  - The code execution will switch to a predefined (custom) exception handler function
  - Depending on the situation,
    - the handler may then resume the execution from the saved code state
    - terminate the script execution
    - continue the script from a different location in the code

57

Exception – in action

```php
<?php
    function checkClassSize($size){
       $max_size = 10;

       if($size > $max_size)
       {
          throw new Exception("Maximum class size is $max_size");
       }
       return true;
    }

    //trigger exception and must be handled/caught probably
    checkClassSize(11);
?>
```

Fatal error: Uncaught exception 'Exception' with message 'Maximum class size is 10' in E:\wamp\www\mvc8\index.php on line *21*

Exception: Maximum class size is 10 in E:\wamp\www\mvc8\index.php on line *21*

Call Stack

#	Time	Memory	Function	Location
1	0.0000	242400	{main}( )	..\index.php:0
2	0.0156	243208	checkClassSize( )	..\index.php:27

58

Exception – got to catch them all

- **Try**
  - If the exception does not trigger, program continues
  - If the exception is triggered, an exception is "**thrown**"
- **Throw**
  - Each "throw" must have at least one "**catch**"
- **Catch**
  - Catches or retrieves an exception and creates an object containing the exception information
- **Finally** (as of version PHP 5.5)
  - If an exception is thrown and matches an existing catch block, that **catch code will be executed before the finally block is run.**
  - If there is no matching catch block, or if no exception is thrown, the finally code will be run **regardless**

59

Exception – got to catch them all

```php
<?php
    class NewWestClassSizeException extends Exception {}
    function checkClassSize($size){
       $max_size = 10;
       if($size > $max_size)
       {
          throw new NewWestClassSizeException("Maximum class size is $max_size");
       }
       return true;
    }
    // exception: try, catch and finally
    try{
       checkClassSize(11);
    }
    catch (NewWestClassSizeException $exe) // Handle error
    {
       echo("Error: {$exe->getFile()}:{$exe->getLine()} - '{$exe->getMessage()}'<br/>");
    }
    finally {// Clean up
       echo("Small class size is preferred :)");
    }
?>
```

60

Exception – know thyself

```php
<?php
    class NewWestClassException extends Exception {}
    class NewWestClassSizeException extends NewWestClassException {}
    function checkClassSize($size){
       $max_size = 10;
       if($size > $max_size)
       {
          throw new NewWestClassSizeException("Maximum class size is $max_size");
       }
       return true;
    }
    // exception: try, catch and finally
    try{
       checkClassSize(11);
    }
    catch (NewWestClassException $exe) // Handle error
    {
       echo("Error: {$exe->getFile()}:{$exe->getLine()} –
       'NewWestClassException: {$exe->getMessage()}'<br/>");
    }
    catch (NewWestClassSizeException $exe) // Handle error
    {
       echo("Error: {$exe->getFile()}:{$exe->getLine()} –
       'NewWestClassSizeException: {$exe->getMessage()}'<br/>");
    }
    finally {// Clean up
       echo("Small class size is preferred :)");
    }
?>
```

Which exception is caught?

61