PHP

Introduction

- The PHP Hypertext Preprocessor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases.
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- PHP is basically used for developing web based software applications
- PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can restrict users to access some pages on your website
- PHP can encrypt data

To start using PHP

- In order to develop and run PHP Web pages three vital components need to be installed on your computer system.
- **Web Server -** PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely availble Apache Server. Download Apache for free here: http://httpd.apache.org/download.cgi
- **Database -** PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database. Download MySQL for free here: http://www.mysql.com/downloads/index.html
- **PHP Parser -** In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser. This tutorial will guide you how to install PHP parser on your computer.

PHP Syntax Overview

Escaping to PHP:

- The PHP parsing engine needs a way to differentiate PHP code from other elements in the page. The mechanism for doing so is known as 'escaping to PHP.' There are four ways to do this:
- **Canonical PHP tags:**
  - The most universally effective PHP tag style is:
  - <?php…..?>
- **Short-open (SGML-style) tags:**
  - <?.....?>
- **ASP-style tags:**
  - ASP-style tags mimic the tags used by Active Server Pages to delineate code blocks. ASP-style tags look like this
  - <%..%>
- **HTML script tags:**
  - <script language="PHP">...</script>

First PHP Web Page

- <!DOCTYPE html>
  <html>
  <body>

  <?php
  echo "My first PHP script!";
  ?>

  </body>
  </html>
- The default file extension for PHP files is **".php".**

Comments in PHP

- A comment in PHP code is a line that is not read/executed as part of the program. Its only purpose is to be read by someone who is editing the code!
- Comments are useful for:
- To let others understand what you are doing - Comments let other programmers understand what you were doing in each step (if you work in a group)
- To remind yourself what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code

PHP supports three ways of commenting:

- <!DOCTYPE html>
  <html>
  <body>

  <?php
  // This is a single line comment

  # This is also a single line comment

  /*
  This is a multiple lines comment block
  that spans over more than
  one line
  */
  ?>

  </body>
  </html>

PHP echo and print Statements

- In PHP there is two basic ways to get output: echo and print.
- There are some differences between echo and print:
  - echo - can output one or more strings
  - print - can only output one string, and returns always 1

Variables

- As with algebra, PHP variables can be used to hold values (x=5) or expressions (z=x+y).
- A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).
- Rules for PHP variables:
  - A variable starts with the $ sign, followed by the name of the variable
  - A variable name must start with a letter or the underscore character
  - A variable name cannot start with a number
  - A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
  - Variable names are case sensitive ($y and $Y are two different variables)

Creating (Declaring) PHP Variables

- <?php
  $x=5;
  $y=6;
  $z=$x+$y;
  echo $z;
  ?>

Data types in PHP

- PHP has a total of eight data types which we use to construct our variables:
- **Integers:** are whole numbers, without a decimal point, like 4195.
- **Doubles:** are floating-point numbers, like 3.14159 or 49.1.
- **Booleans:** have only two possible values either true or false.
- **NULL:** is a special type that only has one value: NULL.
- **Strings:** are sequences of characters, like 'PHP supports string operations.'
- **Arrays:** are named and indexed collections of other values.
- **Objects:** are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources:** are special variables that hold references to resources external to PHP (such as database connections).
- var_dump() function returns the data type and value of variables

PHP Variables Scope

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
  - local
  - global
  - static

PHP Operator Types

- PHP language supports following type of operators.
  - Arithmetic Operators
  - Comparision Operators
  - Logical (or Relational) Operators
  - Assignment Operators
  - Conditional (or ternary) Operators
- **Conditional Operator**

Operator	Description	Example
? :	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

Arithmatic Operators:

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by denumerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

Comparison Operators:

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Logical Operators

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true then then condition becomes true.	(A and B) is true.
or	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	(A or B) is true.
&&	Called Logical AND operator. If both the operands are non zero then then condition becomes true.	(A && B) is true.
||	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

Assignment Operators

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assigne value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

PHP Loop Types

- Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.
- **for -** loops through a block of code a specified number of times.
- **while -** loops through a block of code if and as long as a specified condition is true.
- **do...while -** loops through a block of code once, and then repeats the loop as long as a special condition is true.
- **foreach -** loops through a block of code for each element in an array.

PHP Arrays

- An array is a data structure that stores one or more similar type of values in a single variable.
- There are three different kind of arrays and each array value is accessed using an ID c which is called array index.
  - **Numeric array** - An array with a numeric index. Values are stored and accessed in linear fashion
  - **Associative array** - An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.
  - **Multidimensional array** - An array containing one or more arrays and values are accessed using multiple indices

Types of Array

- **Numeric Array**
- These arrays can store numbers, strings and any object but their index will be prepresented by numbers. By default array index starts from zero
- **Associative Arrays**
- The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.
- **Multidimensional Arrays**
- A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

Creating Indexed Arrays

- <?php
  $cars=array("Volvo","BMW","Toyota");
  echo "I like " . $cars[0] . ", " . $cars[1] . " and "
  . $cars[2] . ".";
  ?>

Counting length of Array

- <?php
  $cars=array("Volvo","BMW","Toyota");
  echo count($cars);
  ?>

MultiDimensional Array

- ```
  <?php
  // A two-dimensional array:
  $cars = array
    (
    array("Volvo",100,96),
    array("BMW",60,59),
    array("Toyota",110,100)
    );
  ?>
  ```
-
  ```

PHP Associative Arrays

- $age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
- OR
- $age['Peter']="35";
  $age['Ben']="37";
  $age['Joe']="43";
- E.g.
- <?php
  $age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
  echo "Peter is " . $age['Peter'] . " years old.";
  ?>

Array handling

- sort() - sort arrays in ascending order
- rsort() - sort arrays in descending order
- asort() - sort associative arrays in ascending order, according to the value
- ksort() - sort associative arrays in ascending order, according to the key
- arsort() - sort associative arrays in descending order, according to the value
- krsort() - sort associative arrays in descending order, according to the key

String Handling

- They are sequences of characters, like "PHP supports string operations".
- PHP String Functions
  - strlen() function
    - <?php
      echo strlen("Hello world!");
      ?>
  - strpos() function
    - The strpos() function is used to search for a specified character or text within a string.
    - <?php
      echo strpos("Hello world!","world");
      ?>

- fprintf():Writes a formatted string to a specified output stream
  - <?php
    $number = 9;
    $str = "Beijing";
    $file = fopen("test.txt","w");
    echo fprintf($file,"There are %u million bicycles in %s.",$number,$str);
    ?>
- chr(): Return characters from different ASCII values:
  - <?php
    echo chr(52) . "
"; // Decimal value
    echo chr(052) . "
"; // Octal value
    echo chr(0x52) . "
"; // Hex value
    ?>
- printf() : Output a formatted string:
  - <?php
    $number = 9;
    $str = "Beijing";
    printf("There are %u million bicycles in %s.",$number,$str);
    ?>

- str_replace() : Replace the characters "world" in the string "Hello world!" with "Peter":
- <?php
  echo str_replace("world","Peter","Hello world!");
  ?>
- strchr(): Find the first occurrence of "world" inside "Hello world!" and return the rest of the string:
- <?php
  echo strchr("Hello world!","world");
  ?>
- strcmp() : Compare two strings (case-sensitive):
- strtolower()
- strtoupper()
- trim()

PHP Functions

- PHP functions are similar to other programming languages. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.
- There are two parts which should be clear to you:
  - Creating a PHP Function
  - Calling a PHP Function

Passing Arguments in Function

- **Passing Arguments by Reference:**
  - It is possible to pass arguments to functions by reference. This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value.
- **Passing Arguments by value**
  - PHP gives you option to pass your parameters inside a function. You can pass as many as parameters your like. These parameters work like variables inside your function.

Dynamic Function Calls

- It is possible to assign function names as strings to variables and then treat these variables exactly as you would the function name itself.
- <html>

```
<head>
<title>Dynamic Function Calls</title>
 </head>
<body>
<?php function sayHello()
{ echo "Hello<br />"; }
$function_holder = "sayHello";
$function_holder(); ?>
 </body>
</html>
```

PHP Global Variables - Superglobals

- Several predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and can access them from any function, class or file without having to do anything special.
- The PHP superglobal variables are:
- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

PHP GET and POST Methods

- There are two ways the browser client can send information to the web server.
  - The GET Method
  - The POST Method
- The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the **?** character.
  - http://www.test.com/index.htm?name1=value1&name2=value2

Get Method

- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY_STRING environment variable.
- The PHP provides **$_GET** associative array to access all the sent information using GET method.

The POST Method

- The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.
- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides **$_POST** associative array to access all the sent information using POST method.

The $_REQUEST variable

- The PHP $_REQUEST variable contains the contents of both $_GET, $_POST, and $_COOKIE.
- The PHP $_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods.

preg_match()

- The caret ^ is an anchor, it means "the start of the haystack/string/line".
  - If a caret is the first symbol inside a character class [], it has a different meaning: It negates the class. (So in [^ab] the caret makes that class match anything which is *not* ab)
- The dot . and the asterisk * serve two separate purposes:
  - The dot matches any single character except newline \n.
- The asterisk says "allow zero or many of the preceeding type".
- When these two are combined as .* it basically reads "zero or more of anything until a newline or another rule comes into effect".
- The dollar $ is also an anchor like the caret, with the opposite function: "the end of the haystack".
- (?=.{4,}) - requires 4 characters minimum
- (?.*[0-9]) - requires it to have numbers
- (?=.*[a-z])- requires it to have lowercase
- (?=.*[A-Z])- requires it to have uppercase
- The "i" after the pattern delimiter indicates a case-insensitive search

- \d - Matches any numeric character - same as [0-9]
- \D - Matches any non-numeric character - same as [^0-9]
- \s - Matches any whitespace character - sames as [\t\n\r\f\v]
- \S - Matches any non-whitespace character - same as [^ \t\n\r\f\v]
- \w - Matches any alphanumeric character - same as [a-zA-Z0-9_]
- \W - Matches any non-alphanumeric character - same as [^a-zA-Z0-9_]
- The \b in the pattern indicates a word boundary, so onl y the distinct word is searched
- The vertical pipe (|) metacharacter is used for alternatives in a regular expression

Modifiers

- **i** - Ignore Case, case insensitive
- **U** - Make search ungreedy
- **s** - Includes New line
- **m** - Multiple lines
- **x** - Extended for comments and whitespace
- **e** - Enables evaluation of replacement as PHP code. (preg_replace only)
- **S** - Extra analysis of pattern

Assertions

- **b** - Word Boundry
- **B** - Not a word boundary
- **A** - Start of subject
- **Z** - End of subject or newline at end
- **z** - End of subject
- **G** - First matching position in subject
-

foo	The string "foo"
^foo	"foo" at the start of a string
foo$	"foo" at the end of a string
^foo$	"foo" when it is alone on a string
[abc]	a, b, or c
[a-z]	Any lowercase letter
[^A-Z]	Any character that is not a uppercase letter
(gif|jpg)	Matches either "gif" or "jpeg"
[a-z]+	One or more lowercase letters
[0-9\.\-]	Any number, dot, or minus sign
^[a-zA-Z0-9_]{1,}$	Any word of at least one letter, number or _
([wx])([yz])	wy, wz, xy, or xz
[^A-Za-z0-9]	Any symbol (not a number or a letter)
([A-Z]{3}|[0-9]{4})	Matches three letters or four numbers