5. Programming in PHP & MySQL

5.1. Origins and Uses of PHP

- Developed by Rasmus Lerdorf in 1994
- Developed to allow him to track visitors to his Web site
- PHP is an open-source product
- PHP is an acronym for Personal Home Page, or PHP: Hypertext Preprocessor
- PHP is used for form handling, file processing, and database access

5.2. Overview of PHP

PHP is a server-side scripting language whose scripts are embedded in HTML documents. It is similar to JavaScript, but on the server side. It is an alternative to CGI, Active Server Pages(ASP), and Java Server Pages (JSP).

1. The PHP processor has two modes:
    - copy (HTML) and
    - interpret (PHP)
2. PHP syntax is similar to that of JavaScript
3. PHP is dynamically typed

5.3. General Syntactic Characteristics

PHP code can be specified in an HTML document internally or externally:

Internally:

```
<?php
...
?>
```

Externally:

```
include ("myScript.inc")
```

The file can have both PHP and HTML. If the file has PHP, the PHP must be in <?php .. ?>, even if the include is already in <?php .. ?>. All variable names begin with dollar ($).

Comments - three different kinds (Java and Perl)

// ... --> single line comment

... --> single line comment

/* ... */ --> multiple line comment

Compound statements are formed with braces. Compound statements cannot be blocks.

5.4. Primitives, Operations, and Expressions

5.4.1. Variables

- There are no type declarations.
- An unassigned (unbound) variable has the value, NULL
- The ***unset*** function sets a variable to NULL
- The **isset** function is used to determine whether a variable is NULL
- PHP has many predefined variables, including the environment variables of the host operating system
- You can get a list of the predefined variables by calling phpinfo() in a script

There are eight primitive types:

1. Four scalar types: Boolean, integer, double, and string
2. Two compound types: array and object
3. Two special types: resource and NULL

Points to remember:

- Integer & double are typical, like in C
- Characters are single bytes
- String literals use single or double quotes

Difference between single quoted string literals and double qouted string literals

1. Single-quoted string literals
    - Embedded variables are NOT interpolated
    - Embedded escape sequences are NOT recognized

2. Double-quoted string literals
    - Embedded variables are interpolated
    - If there is a variable name in a double quoted string but you don't want it interpolated, it must be back-slashed
    - Embedded escape sequences are recognized

Note: For both single- and double-quoted literal strings, embedded delimiters must be back-slashed.

Boolean

- values are true and false (case insensitive)
- 0, " " and "0" are false; others are true

5.4.2. Arithmetic Operators and Expressions

Usual operators

- If the result of integer division is not an integer, a double is returned
- Any integer operation that results in overflow produces a double
- The modulus operator coerces its operands to integer, if necessary
- When a double is rounded to an integer, the rounding is always towards zero

Arithmetic functions

- floor, ceil, round, abs, min, max, rand, etc.

String Operations and Functions

- The only operator is period, for concatenation
- Indexing - $str{3} is the fourth character

Functions:

- strlen, strcmp, strpos, substr, strtolower, strtoupper as in C
- chop – remove whitespace from the right end
- trim – remove whitespace from both ends
- ltrim – remove whitespace from the left end

Scalar Type Conversions

String to numeric

- If the string contains an e or an E, it is converted to double; otherwise to int
- If the string does not begin with a sign or a digit, zero is used

Explicit conversions – casts

- e.g., (int)$total or intval($total) or settype($total, "integer")

The type of a variable can be determined with ***gettype*** or ***is_type***

- gettype($total) - it may return "unknown"
- is_integer($total) – a predicate function

5.5. Output

Output from a PHP script is HTML that is sent to the browser. HTML is sent to the browser through standard output

There are three ways to produce output:

- echo
- print
- printf

echo and print take a string, but will coerce other values to strings

An Example:

```
<html>
<head><title> Trivial php example </title>
</head>
<body>
<?php
print "Welcome to my Web site!";
?>
</body>
</html>
```

5.6. Control Statements

Control Expressions

- Relational operators - same as JavaScript, (including === and !==)
- Boolean operators - same as Perl (two sets, &&, || etc.)

Selection statements

- if, if-else, elseif --> same as C
- switch - as in C
- The switch expression type must be integer, double, or string
- while - just like C
- do-while - just like C
- for - just like C
- foreach - discussed later
- break - in any for, foreach, while, do-while, or switch
- continue - in any loop

5.6.1. IF...ELSE IF...ELSE

The if statement execute a single statement or a group of statements if a certain condition is met. It can not do anything if the condition is false. For this purpose elseif and else is used. elseif is a combination of if and else. It extends an if statement to execute a single statement or a group of statements if a certain condition is met. It can not do anything if the condition is false.

Syntax

```
if (condition)
   execute statement(s) if condition is true;
elseif (another condition)
  execute statement(s) if condition is true;
else
  execute statement(s) if all condition(s) are false;
```

Example :

```
<?php
if ($x > $y)
{
echo "x is bigger than y";
}
elseif ($x == $y)
{
echo "x is equal to y";
}
else
{
echo "x is smaller than y";
}
?>
```

5.6.2. The While Loop

The While statement executes a block of code if and as long as a specified condition evaluates to true. If the condition becomes false, the statements within the loop stop executing and control passes to the statement following the loop. The While loop syntax is as follows:

```
while (condition)
{
 code to be executed;
}
```

The block of code associated with the While statement is always enclosed within the { opening and } closing brace symbols to tell PHP clearly which lines of code should be looped through.

While loops are most often used to increment a list where there is no known limit to the number of iterations of the loop.

For example:

```
while (there are still rows to read from a database)
{
  read in a row;
  move to the next row;
}
```

Let's have a look at the examples. The first example defines a loop that starts with i=0. The loop will continue to run as long as the variable i is less than, or equal to 10. i will increase by 1 each time the loop runs:

```
<?php
$i=0;
while ($i <= 10) { // Output values from 0 to 10
  echo "The number is ".$i."<br />";
  $i++;
}
?>
```

5.6.3. The Do...While Loop

The Do...While statements are similar to While statements, except that the condition is tested at the end of each iteration, rather than at the beginning. This means that the Do...While loop is guaranteed to run at least once. The Do...While loop syntax is as follows:

```
do
{
  code to be exected;
}
while (condition);
```

The example below will increment the value of i at least once, and it will continue incrementing the variable i as long as it has a value of less than or equal to 10:

```
<?php
$i = 0;
do {
  echo "The number is ".$i."<br/>";
  $i++;
}
while ($i <= 10);
?>
```

5.6.4. The For Loop

The For statement loop is used when you know how many times you want to execute a statement or a list of statements. For this reason, the For loop is known as a definite loop. The syntax of For loops is a bit more complex, though for loops are often more convenient than While loops. The For loop syntax is as follows:

```
for (initialization; condition; increment)
{
  code to be executed;
}
```

The For statement takes three expressions inside its parentheses, separated by semi-colons. When the For loop executes, the following occurs:

1. The initializing expression is executed. This expression usually initializes one or more loop counters, but the syntax allows an expression of any degree of complexity.
2. The condition expression is evaluated. If the value of condition is true, the loop statements execute. If the value of condition is false, the For loop terminates.
3. The update expression increment executes.
4. The statements execute, and control returns to step 2.

Have a look at the very simple example that prints out numbers from 0 to 10:

```
<?php
for ($i=0; $i <= 10; $i++)
{
  echo "The number is ".$i."<br />";
}
?>
```

Next example generates a multiplication table 2 through 9. Outer loop is responsible for generating a list of dividends, and inner loop will be responsible for generating lists of dividers for each individual number:

```
<?php
echo "<h1>Multiplication table</h1>";
echo "<table border=2 width=50%";

for ($i = 1; $i <= 9; $i++ ) {   //this is the outer loop
 echo "<tr>";
 echo "<td>".$i."</td>";

  for ( $j = 2; $j <= 9; $j++ ) { // inner loop
      echo "<td>".$i * $j."</td>";
   }

  echo "</tr>";
}

echo "</table>";
?>
```

5.6.5. The Foreach Loop

The Foreach loop is a variation of the For loop and allows you to iterate over elements in an array. There are two different versions of the Foreach loop. The Foreach loop syntaxes are as follows:

```
foreach (array as value)
{
  code to be executed;
}
```

```
foreach (array as key => value)
{
  code to be executed;
}
```

The example below demonstrates the Foreach loop that will print the values of the given array:

```
<?php
$email = array('john.smith@example.com', 'alex@example.com');
foreach ($email as $value) {
  echo "Processing ".$value."<br />";
}
?>
```

PHP executes the body of the loop once for each element of $email in turn, with $value set to the current element. Elements are processed by their internal order. Looping continues until the Foreach loop reaches the last element or upper bound of the given array.

An alternative form of Foreach loop gives you access to the current key:

```
<?php
$person = array('name' => 'Andrew', 'age' => 21, 'address' => '77, Lincoln st.');
foreach ($person as $key => $value) {
  echo $key." is ".$value."<br />";
```

```
}
?>
```

In this case, the key for each element is placed in $key and the corresponding value is placed in $value.

The Foreach construct does not operate on the array itself, but rather on a copy of it. During each loop, the value of the variable $value can be manipulated but the original value of the array remains the same.

5.6.6. Break and Continue Statements

Sometimes you may want to let the loops start without any condition, and allow the statements inside the brackets to decide when to exit the loop. There are two special statements that can be used inside loops: Break and Continue.

The Break statement terminates the current While or For loop and continues executing the code that follows after the loop (if any). Optionally, you can put a number after the Break keyword indicating how many levels of loop structures to break out of. In this way, a statement buried deep in nested loops can break out of the outermost loop.

Examples below show how to use the Break statement:

```
<?php
echo "<p><b>Example of using the Break statement:</b></p>";
for ($i=0; $i<=10; $i++) {
  if ($i==3){break;}
  echo "The number is ".$i;
  echo "<br />";
}
echo "<p><b>One more example of using the Break statement:</b><p>";
$i = 0;
$j = 0;
while ($i < 10) {
 while ($j < 10) {
  if ($j == 5) {break 2;} // breaks out of two while loops
```

```
   $j++;
 }
 $i++;
}
echo "The first number is ".$i."<br />";
echo "The second number is ".$j."<br />";
?>
```

The Continue statement terminates execution of the block of statements in a While or For loop and continues execution of the loop with the next iteration:

```
<?php
echo "<p><b>Example of using the Continue statement:</b><p>";
for ($i=0; $i<=10; $i++) {
  if (i==3){continue;}
  echo "The number is ".$i;
  echo "<br />";
}
?>
```

5.7. Arrays

PHP arrays are not like the arrays of any other programming language. A PHP array is a generalization of the arrays of other languages. A PHP array is really a mapping of keys to values, where the keys can be numbers (to get a traditional array) or strings (to get a hash).

5.7.1. Creating Arrays

Use the array() construct, which takes one or more ***key => value*** pairs as parameters and returns an array of them. The keys are non-negative integer literals or string literals. The values can be anything.

e.g.,

```
$list = array(0 => "apples",1 => "oranges", 2 => "grapes")
```

This is a “regular” array of strings.

- If a key is omitted and there have been integer keys, the default key will be the largest current key + 1
- If a key is omitted and there have been no integer keys, 0 is the default key
- If a key appears that has already appeared, the new value will overwrite the old one

Arrays can have mixed kinds of elements

```
$list = array("make" => "Cessna", "model" => "C210", "year" => 1960, 3 => "sold");
$list = array(1, 3, 5, 7, 9);
$list = array(5, 3 => 7, 5 => 10, "month" => "May");
$colors = array('red','blue','green','yellow');
```

5.7.2. Accessing array elements – use brackets

```
$list[4] = 7;
$list["day"] = "Tuesday";
$list[] = 17;
```

- If an element with the specified key does not exist, it is created
- If the array does not exist, the array is created
- The keys or values can be extracted from an array

```
$highs = array("Mon" => 74, "Tue" => 70, "Wed" => 67, "Thu" => 62, "Fri" => 65);
$days = array_keys($highs);
$temps = array_values($highs);
```

5.7.3. Deleting an array or its elements

An array can be deleted with unset

```
unset($list);
unset($list[4]); # No index 4 element now
```

Array Functions

- ***is_array($list)*** returns true if $list is a an array
- ***in_array(17, $list)*** returns true if 17 is an element of $list
- ***explode(" ", $str)*** creates an array with the values of the words from $str, split on a space
- ***implode(" ", $list)*** creates a string of the elements from $list, separated by a space

Sequential access to array elements

- current and next

```
$colors = array("Blue", "red", "green", "yellow");
$color = current($colors);
print("$color <br />");
while ($color = next($colors))
print ("$color <br />");
```

This does not always work – for example, if the value in the array happens to be FALSE

- ***array_push($list, $element) and array_pop($list)*** Used to implement stacks in arrays

```
foreach (array_name as scalar_name) { ... }
```

```
foreach ($colors as $color) {
print "Is $color your favorite?<br /> ";
}
```

Output:

Is red your favorite color?

Is blue your favorite color?

Is green your favorite color?

Is yellow your favorite color?

foreach can iterate through both keys and values:

```
foreach ($colors as $key => $color) { … }
```

Inside the compound statement, both $key and $color are defined

```
$ages = array("Bob" => 42, "Mary" => 43);
foreach ($ages as $name => $age)
print("$name is $age years old <br />");
```

5.7.4. Array Sorting

To sort the values of an array, leaving the keys in their present order - intended for traditional arrays

e.g., sort($list);

- The sort function does not return anything
- Works for both strings and numbers, even mixed strings and numbers

```
$list = ('h', 100, 'c', 20, 'a');
sort($list);
// Produces (20, 100, 'a', 'c', 'h')
```

In PHP 4, the sort function can take a second parameter, which specifies a particular kind of sort

```
sort($list, SORT_NUMERIC);
```

5.7.4.1. asort

To sort the values of an array, but keeping the key/value relationships - intended for hashes

5.7.4.2. rsort

To sort the values of an array into reverse order

5.7.4.3. ksort

To sort the elements of an array by the keys, maintaining the key/value relationships

```
$list("Fred" => 17, "Mary" => 21, "Bob" => 49, "Jill" => 28);
ksort($list);
// $list is now ("Bob" => 49, // "Fred" => 17, "Jill" => 28, "Mary" => 21)
```

5.7.4.4. krsort

To sort the elements of an array by the keys into reverse order.