PHP RSS

Mastering WHILE Loops for PHP and MySQL

By: Codex-M

Rating: ★★★★☆ / 5

2009-10-27

Do you want to learn how to handle PHP WHILE loops? WHILE loops are one of the most powerful features as well as the easiest loop available to any PHP/MySQL developer. They enable us to shorten repetitive tasks for a highly useful application. This tutorial gives examples of WHILE loops in PHP/MySQL that beginner and novice developers can use as a quick reference for building similar loops in their applications.

The early part of this article will focus on both basic and advanced WHILE looping techniques entirely written for PHP. The last part will be devoted to the use of WHILE Loop in displaying MySQL queries.

There are different types of loops in PHP that suit different applications. Seeing actual examples and real applications will enable a developer to use this information to determine which loop script is similar to his intended application; thus, he can use it to write derivative programs. This marks the first step towards mastery in writing loop statements in PHP/MySQL.

If you are interested, keep reading.

Basics: Loop to output increasing or decreasing sequence of numbers

This is basic; a lot of applications depend on outputting an increasing or decreasing sequence of numbers. See the examples below:

Example #1: Output numbers from 1 to 10 in HTML

```
<?php
$i=1;
while ($i<=10) {
echo $i++;
echo '<br />';
}
?>
```

What if you need to produce numbers starting from 51 and continuing to 202? All you need to do is customize using the format (bold):

```
<?php
$i=51;
while ($i<=202) {
echo $i++;
echo '<br />';
}
?>
```

Example #2: Output a decreasing sequence of numbers: 10 to 1 in HTML

A variation of this can be formulated but using $i—

```
<?php
$i=10;
while ($i>=1) {
echo $i--;
echo '<br />';
}
?>
```

The situation is somewhat reversed from what is shown in Example 1.

Example #3: Output even numbers in HTML:

We even use $i++ to produce even numbers; see below. The code will produce an even number of 2, 4, 6, 8, 10.Instead of using 10 as the last number, 8 is used, since the loop will add twice.

```
<?php
$i=0;
while ($i<=8) {
$i=($i++)+2;
echo $i;
```

```
echo '<br />';

}

?>
```

The same principle applies to outputting odd numbers or any type of number sequences.

One of the most popular applications for WHILE loops is acting as a COUNTER. For example, say we want to generate 36 random numbers between 101 and 300. We will use a WHILE loop to count looping events until we've completed 36 of them.

This function is used as a random number function: echo mt_rand(101, 300). And we will use this basic loop as a counter:

```
<?php

$i=1;

while ($i<=36) {

echo $i++;

echo '<br />';

}

?>
```

Combine the random number generator function with a WHILE loop counter and we get:

```
<?php

$i=1;

while ($i<=30) {

echo mt_rand(101, 300);

$i++;

echo '<br />';

}

?>
```

Note that $i++ is not being output to HTML because it is only used as a counter. To label the generated random numbers in the form like this:

1.) 105

2.) 230

3.) 165

We will echo the “counter” beside the generated random number:

```
<?php

$i=1;

while ($i<=30) {

echo $i++.'.)'.' '.mt_rand(101, 300);

echo '<br />';

}

?>
```

The above example will not generate unique random numbers. We can still make use of PHP WHILE loops as a counter. But to generate “X” amount of unique random numbers, we will make use of a PHP array.

A full discussion of arrays is out of the scope of this article. However, below you will find some good references:

http://www.phphelps.com/7_Creating_and_looping_array_in_PHP.shtml

http://free.netartmedia.net/PHP/PHP50.html

http://www.homeandlearn.co.uk/php/php6p2.html

The strategy is to generate a random number and store it in an array. Then we'll let PHP decide whether or not the generated random number is already stored in the array. If it is stored already, it will not be assigned to the array variable, but will generate a random number again. This will ensure that all values stored in an array are unique.

After the counter reaches its maximum, PHP will then output all the unique values stored in the array using another WHILE looping statement. Below is the example PHP script that will generate 30 unique (non-repetitive) random numbers between 10 and 50:

```
<?php

//create array

$randomarray = array();

//initiate counter

$i=1;

while ($i<=30) {

//generate random number

$randomnumber = mt_rand(10, 50);

//check first if generated random number is in array if not store it to the array

if(!(in_array($randomnumber,$randomarray))){

//number is not in array, store the number in array and increment counter

$randomarray[]=$randomnumber;

$i++;

}

}

//echo all results inside the array

while (list($key,$value) = each($randomarray)) {

echo $value;

echo '<br />';

}

?>
```

Have you ever thought of constructing trigonometric tables? To construct a table for an angle starting from 0 degrees and continuing to 360 degrees using three basic trigonometric functions and their inverses (with $x as the variable):

```
sin(deg2rad($x))

cos(deg2rad($x))

tan(deg2rad($x))
```

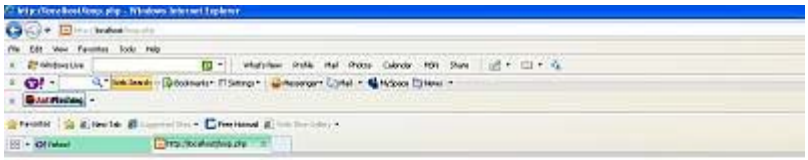
You will need to iterate going from 0 degrees to 360 degrees using a WHILE loop. You will need to convert $x to radians, since by default trigonometric functions evaluate angles as radians. You can accomplish this using the function:

Deg2rad();

Below is the complete script required to generate the trigonometric table for the three functions (sine, cosine and tangent) and well as their inverses (cotangent, secant, and cosecant)

```
<?php

echo '<table width=100% border=1>';

echo '<tr><td><b>Angle in Degrees</b></td><td><b>Sine</b></td><td><b>Cosine</b></td>
<td><b>Tangent</b></td><td><b>Cotangent</b></td>
<td><b>Secant</b></td><td><b>Cosecant</b></td></tr>';

$i=1;

while ($i<=360) {

$x=$i++;

echo '<tr>';

echo '<td>'.$x.'</td>';

echo '<td>'.sin(deg2rad($x)).'</td>';

echo '<td>'.cos(deg2rad($x)).'</td>';

echo '<td>'.tan(deg2rad($x)).'</td>';

echo '<td>'.(1/tan(deg2rad($x))).'</td>';

echo '<td>'.(1/cos(deg2rad($x))).'</td>';

echo '<td>'.(1/sin(deg2rad($x))).'</td>';

echo '</tr>';

}

echo '</table>';

?>
```

The screen shot below is the trigonometric table when viewed in a browser:



Angle in Degrees	Sine	Cosine	Tangent	Cotangent
1	0.0174524064373	0.999847695156	0.0174550649282	57.2899616308
2	0.0348994967025	0.999390827019	0.0349207694917	28.6362532829
3	0.0523359562429	0.998629534755	0.052407779283	19.0811366877
4	0.0697564737441	0.99756405026	0.0699268119435	14.3006662567
5	0.0871557427477	0.996194698092	0.0874886635259	11.4300523028
6	0.104528463268	0.994521895368	0.105104235266	9.51436445422
7	0.121869343405	0.992546151641	0.122784560903	8.14434642797
8	0.13917310096	0.990268068742	0.140540834702	7.11536972238
9	0.15643446504	0.987688340595	0.158384440325	6.31375151468
10	0.173648177667	0.984807753012	0.176326980708	5.67128181962
11	0.190808995377	0.981627183448	0.194380309138	5.14455401597
12	0.207911690818	0.978147600734	0.21255656167	4.70463010948

You can also extend the above application not only to trigonometry but to other important engineering tables like logarithmic tables, steam tables, etc.

One of the classic PHP-MySQL applications is retrieving all rows in the MySQL table and outputting them as an HTML table. Consider the MySQL table “data” screen shot from phpMyadmin:

	student	math	arts	philosophy
☐ ✎ ✕	Benjamin	75	90	95
☐ ✎ ✕	Leonardo	65	100	95
☐ ✎ ✕	Galileo	93	75	85
☐ ✎ ✕	Johannes	78	85	85
☐ ✎ ✕	Codex	95	90	49
☐ ✎ ✕	Albert	95	57	70
☐ ✎ ✕	Isaac	95	62	55
☐ ✎ ✕	Marie	61	75	76
☐ ✎ ✕	Thomas	75	80	85
☐ ✎ ✕	Michaelangelo	51	99	80
☐ ✎ ✕	Socrates	55	75	100

You like to display this entire MySQL table as an HTML table in the browser. The first thing you need to do is connect to the MySQL database.

```
$username = "yourmysqlusername";

$password = "yourmysqlpassword";

$hostname = "yourmysqlhostname";

$database = "yourmysqldatabase";

$dbhandle = mysql_connect($hostname, $username, $password)

or die("Unable to connect to MySQL");

$selected = mysql_select_db($database,$dbhandle)

or die("Could not select $database");
```

Next, we query MySQL to retrieve all entries:

$query= ‘SELECT * FROM data’;

$result = mysql_query($query)

or die (‘Error in query’);

Okay, now we need to display the HTML table:

echo ‘<table width=100% border=1>’;

echo ‘<tr><td>Student Name</td><td>Math Score</td><td>Arts Score</td><td>Philosophy Score</td></tr>’;

Finally, the WHILE loop can be used to retrieve the rows from $result, which is the result of the MySQL query:

While ($row=mysql_fetch_row($result))

{

echo ‘<tr>’;

echo ‘<td>’.$row[0].’</td>’;

echo ‘<td>’.$row[1].’</td>’;

echo ‘<td>’.$row[2].’</td>’;

echo ‘<td>’.$row[3].’</td>’;

}

echo ‘</table>’;

Free result set:

mysql_free_result($result);

Close database connection

mysql_close($dbhandle)

Combining all the scripts for a complete script:

```
<?php

$username = "xxx";

$password = "xxx";

$hostname = "xxx";

$database = "xxx";

$dbhandle = mysql_connect($hostname, $username, $password)

or die("Unable to connect to MySQL");

$selected = mysql_select_db($database,$dbhandle)

or die("Could not select $database");

$query= 'SELECT * FROM data';

$result = mysql_query($query)

or die ('Error in query');

echo '<table width=100% border=1>';
```

```
echo '<tr><td><b>Student Name</b></td><td><b>Math Score</b></td><td><b>Arts Score</b></td><td><b>Philosophy Score</b></td></tr>';

while ($row=mysql_fetch_row($result))

{

echo '<tr>';

echo '<td>'.$row[0].'</td>';

echo '<td>'.$row[1].'</td>';

echo '<td>'.$row[2].'</td>';

echo '<td>'.$row[3].'</td>';

echo '</tr>';

}

echo '</table>';

mysql_free_result($result);

mysql_close($dbhandle);

?>
```

This is how it looks in the browser.

Student Name	Math Score	Arts Score	Philosophy Score
Benjamin	75	90	95
Leonardo	65	100	95
Galileo	93	75	85
Johannes	78	85	85
Codex	95	90	49
Albert	95	57	70
Isaac	95	62	55
Marie	61	75	76
Thomas	75	80	85
Michaelangelo	51	99	80
Socrates	55	75	100