

# Aplikasi Web

## Pertemuan-7

### Pernyataan Kontrol

#### PHP Conditional Statements

Conditional statements in PHP are used to perform different actions based on different conditions.

#### Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In PHP we have two conditional statements:

- **if (...else) statement** - use this statement if you want to execute a set of code when a condition is true (and another if the condition is not true)
- **switch statement** - use this statement if you want to select one of many sets of lines to execute

#### The If Statement

If you want to execute some code if a condition is true and another code if a condition is false, use the if...else statement.

The if statement evaluates the truth value of its argument. If the argument evaluates as TRUE the code following the if statement will be executed. And if the argument evaluates as FALSE and there is an else statement then the code following the else statement will be executed.

#### Syntax

```
if (condition)  
  code to be executed if condition is true;  
else  
  code to be executed if condition is false;
```

#### Example

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!":

if01.php

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
echo "Have a nice weekend!";
else
echo "Have a nice day!";
?>

</body>
</html>
```

If more than one line should be executed when a condition is true, the lines should be enclosed within curly braces:

if02.php

```
<html>
<body>

<?php
$x=10;
if ($x==10)
{
echo "Hello<br />";
echo "Good morning<br />";
}
?>

</body>
</html>
```

Example : visitorinfo.php

```
<?php
$ip = $_SERVER['REMOTE_ADDR'];
$agent = $_SERVER['HTTP_USER_AGENT'];

if(strpos($agent, 'Opera') !== false)
    $agent = 'Opera';
else if(strpos($agent, "MSIE") !== false)
    $agent = 'Internet Explorer';
```

```
echo "Your computer IP is $ip and you are using $agent";
?>
```

The strpos() function returns the numeric position of the first occurrence of it's second argument ('Opera') in the first argument (\$agent). If the string 'Opera' is found inside \$agent, the function returns the position of the string. Otherwise, it returns FALSE.

When you're using Internet Explorer 6.0 on Windows XP the value of \$\_SERVER ['HTTP\_USER\_AGENT'] would be something like:

Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

and if you're using Opera the value the value may look like this :

Mozilla/4.0 (compatible; MSIE 6.0; MSIE 5.5; Windows NT 5.1) Opera 7.0 [en]

So if i you use Opera the strpos() function will return value would be 61. Since 61 !== false then the first if statement will be evaluated as true and the value of \$agent will be set to the string 'Opera'.

Note that I use the !== to specify inequality instead of != The reason for this is because if the string is found in position 0 then the zero will be treated as FALSE, which is not the behaviour that I want.

## The Switch Statement

If you want to select one of many blocks of code to be executed, use the Switch statement.

### Syntax

```
switch (expression)
{
case label1:
    code to be executed if expression = label1;
    break;
case label2:
    code to be executed if expression = label2;
    break;
default:
    code to be executed
    if expression is different
    from both label1 and label2;
}
```

## Example

This is how it works: First we have a single expression (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically. The default statement is used if none of the cases are true.

example : switch01.php

```
<html>
<body>

<?php
switch ($x)
{
case 1:
    echo "Number 1";
    break;
case 2:
    echo "Number 2";
    break;
case 3:
    echo "Number 3";
    break;
default:
    echo "No number between 1 and 3";
}
?>

</body>
</html>
```

## PHP Looping

Looping statements in PHP are used to execute the same block of code a specified number of times.

### Looping

Very often when you write code, you want the same block of code to run a number of times. You can use looping statements in your code to perform this.

In PHP we have the following looping statements:

- **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

- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

## The while Statement

The while statement will execute a block of code **if and as long as** a condition is true.

### Syntax

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

### Example

The following example demonstrates a loop that will continue to run as long as the variable *i* is less than, or equal to 5. *i* will increase by 1 each time the loop runs:

example : while01.php

```
<html>  
<body>  
  
<?php  
$i=1;  
while($i<=5)  
{  
echo "The number is " . $i . "<br />";  
$i++;  
}  
?>  
  
</body>  
</html>
```

Example : while02.php

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

You see that I make the code `$number += 1;` as bold. I did it simply to remind that even an experienced programmer can sometime forget that a loop will happily continue to run

forever as long as the loop expression ( in this case \$number < 10 ) evaluates as true. So when you're creating a loop please make sure you already put the code to make sure the loop will end in timely manner.

## **Break**

The break statement is used to stop the execution of a loop. As an example the while loop below will stop when \$number equals to 6.

Example : [break.php](#)

```
<?php
$number = 1;
while ($number < 10)
{
    echo $number . '<br>';
    if ($number == 6)
    {
        break;
    }

    $number += 1;
}
?>
```

You can stop the loop using the break statement. The break statement however will only stop the loop where it is declared. So if you have a cascading while loop and you put a break statement in the inner loop then only the inner loop execution that will be stopped.

Example : break02.php

```
<?php
$floor = 1;
while ($floor <= 5)
{
    $room = 1;
    while ($room < 40)
    {
        echo "Floor : $floor, room number : $floor". "$room <br>";
        if ($room == 2)
        {
            break;
        }

        $room += 1;
    }
    $floor += 1;

    echo "<br>";
}
?>
```

If you run the example you will see that the outer loop, while (\$floor <= 5), is executed five times and the inner loop only executed two times for each execution of the outer loop. This proves that the break statement only stops the execution of the inner loop where it's declared.

## The do...while Statement

The do...while statement will execute a block of code **at least once** - it then will repeat the loop **as long as** a condition is true.

## Syntax

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

## Example

The following example will increment the value of *i* at least once, and it will continue incrementing the variable *i* while it has a value of less than 5:

example : dowhile01.php

```
<html>
<body>

<?php
$i=0;
do
{
    $i++;
    echo "The number is " . $i . "<br />";
}
while ($i<5);
?>

</body>
</html>
```

## The for Statement

The for statement is used when you know how many times you want to execute a statement or a list of statements.

## Syntax

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

**Note:** The for statement has three parameters. The first parameter is for initializing variables, the second parameter holds the condition, and the third parameter contains any increments required to implement the loop. If more than one variable is included in either the initialization or the increment section, then they should be separated by commas. The

condition must evaluate to true or false.

## Example

The following example prints the text "Hello World!" five times:

example : for01.php

```
<html>
<body>

<?php
for ($i=1; $i<=5; $i++)
{
echo "Hello World!<br />";
}
?>

</body>
</html>
```

## The foreach Statement

Loops over the array given by the parameter. On each loop, the value of the current element is assigned to \$value and the array pointer is advanced by one - so on the next loop, you'll be looking at the next element.

## Syntax

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

## Example

The following example demonstrates a loop that will print the values of the given array:

example : foreach01.php

```
<html>
<body>

<?php
$arr=array("one", "two", "three");
foreach ($arr as $value)
{
echo "Value: " . $value . "<br />";
}
?>

</body>
</html>
```