Digicoders technologies (P) Ltd.

LECTURE NOTES

ON

INTERNET AND WEB TECHNOLOGY

Unit-9

Compiled by

Team Digicoders
B-36, Sector O, Near Ram Ram Bank Chauraha, Aliganj,
Lucknow Uttar Pradesh 226021

CONTENTS

UNIT-9

Server Side Programming using PHP

Introduction to PHP:

The PHP stands for *Hypertext Preprocessor*. PHP is a server-side scripting language designed specifically for web development. It is open-source which means it is free to download and use. It is very simple to learn and use. The files have the extension ".php". It is an interpreted language and it does not require a compiler.

- PHP code is executed in the server.
- It can be integrated with many databases such as Oracle, Microsoft SQL Server, MySQL, PostgreSQL, Sybase, Informix.
- It supports main protocols like HTTP Basic, HTTP Digest, IMAP, FTP, and others. One of the main
- reasons behind this is that PHP can be easily embedded in HTML files and HTML codes can also be written in a PHP file.

Advantages of PHP:

- It is supported by all Operating Systems like Windows, Linux, Unix, etc.
- It is integrated with other programming languages (HTML, CSS, JavaScript, etc) and databased.
- It is easy to connect with the database to store and retrieve data from the database. Multiple databases can also be integrated with PHP.
- It is the fastest programming language compared to other programming languages.
- PHP frameworks and tools are used to protect web applications from outer attacks and security threats.

Variables:

Variables in a program are used to store some values or data that can be used later in a program. Any variables declared in PHP must begin with a dollar sign (**$**), followed by the variable name. PHP variables are case-sensitive, i.e., $sum and $SUM are treated differently. A valid variable name starts with a letter Or underscore, followed by any number of letters, numbers, or underscores.

Example

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

STRINGS

Strings can be seen as a stream of characters. For example, ‘K’ is a character and ‘KIIT’ is a string.

Creating Strings:

There are four ways of creating strings in PHP:

1. **Single-quote strings**: The simplest way to specify a string is to enclose it in single quotes (the character). This type of string does not process special characters inside quotes.

```
<?php

// single-quote strings

$site = 'Welcome to KIIT';

echo $site;

?>
```

2. **Double-quote strings** : Unlike single-quote strings, double-quote strings in PHP are capable of processing special characters.

```
<?php

// double-quote strings

echo "Welcome to KIIT \n";

$site = "KIIT";

echo "Welcome to $site";

?>
```

3. **Heredoc:** The syntax of Heredoc (<<<) is another way to delimit PHP strings. An identifier is given after the heredoc (<<<) operator, after which any text can be written as a new line is started. To close the syntax, the same identifier is given without any tab or space. Heredoc syntax is similar to the double-quoted string, without the quotes.

```
<?php

$input = <<<testHeredoc

Welcome to KIIT.
Started content writing
!.
I am enjoying this.

testHeredoc;

echo $input;

?>
```

4. **Nowdoc:** Nowdoc is very much similar to the heredoc other than the parsing done in heredoc. The syntax is similar to the heredoc syntax with symbol <<< followed by an identifier enclosed in single-quote. The rule for nowdoc is the same as heredoc. Nowdoc syntax is similar to the single-quoted string.

```
<?php
$input = <<<'testNowdoc'
Welcome to KIIT.
Started content writing!.

testNowdoc;

echo $input;

// Directly printing string without any variable
echo <<<'Nowdoc'
<br/>
Welcome to KP .
Learning PHP is fun in KP.

Nowdoc;

?>
```

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators
- Conditional assignment operators

Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power

Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

Increment / Decrement Operators

The PHP increment operators are used to increment a variable's value.

The PHP decrement operators are used to decrement a variable's value.

Operator	Name	Description
++$x	**Pre-increment**	**Increments $x by one, then returns $x**
$x++	**Post-increment**	**Returns $x, then increments $x by one**
--$x	**Pre-decrement**	**Decrements $x by one, then returns $x**
$x--	**Post-decrement**	**Returns $x, then decrements $x by one**

Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

Array Operators

The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

Conditional Assignment Operators

The PHP conditional assignment operators are used to set a value depending on conditions:

Operator	Name	Example	Result
?:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE
??	Null coalescing	$x = *expr1* ?? *expr2*	Returns the value of $x. The value of $x is *expr1* if *expr1* exists, and is not NULL. If *expr1* does not exist, or is NULL, the value of $x is *expr2*. Introduced in PHP 7

Conditional statement

Conditional statements are used to perform different actions based on different

conditions. In PHP we have the following conditional statements:

- if statement - executes some code if one condition is true
- if...else statement - executes some code if a condition is true and another code if that condition is false
- if...elseif...else statement - executes different codes for more than two conditions
- switch statement - selects one of many blocks of code to be executed

The if Statement

The if statement executes some code if one condition is true.

Syntax

```
if (condition) {
  code to be executed if condition is true;
}
```

Example

Output "Have a good day!" if the current time (HOUR) is less than 20:

```
<?php
$t = date("H");

if ($t < "20") {
  echo "Have a good day!";
}
?>
```

The if...else Statement
The if...else statement executes some code if a condition is true and another code if that condition is false.

Syntax
```
if (condition) {
  code to be executed if condition is true;
} else {
  code to be executed if condition is false;
}
```
Example
Output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

```
<?php
$t = date("H");

if ($t < "20") {
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>
```
The if...elseif...else Statement
The if...elseif...else statement executes different codes for more than two conditions.

Syntax
```
if (condition) {
  code to be executed if this condition is true;
} elseif (condition) {
  code to be executed if first condition is false and this condition is true;
} else {
  code to be executed if all conditions are false;
}
```
Example
Output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

```
<?php
$t = date("H");

if ($t < "10") {
  echo "Have a good morning!";
} elseif ($t < "20") {
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>
```

switch Statement
Use the switch statement to select one of many blocks of code to be executed.

Syntax
```
switch (n) {
case label1:
   code to be executed if n=label1;
   break;
 case label2:
   code to be executed if n=label2;
   break;
 case label3:
   code to be executed if n=label3;
   break;
   ...
 default:
   code to be executed if n is different from all labels;
}
```
This is how it works: First we have a single expression n (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 no match is found.

Example
```
<?php
$favcolor = "red";

switch ($favcolor) {
 case "red":
   echo "Your favorite color is red!";
   break;
 case "blue":
   echo "Your favorite color is blue!";
   break;
 case "green":
   echo "Your favorite color is green!";
   break;
 default:
   echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

PHP Loops

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

- while - loops through a block of code as long as the specified condition is true
- do...while - loops through a block of code once, and then repeats the loop as long as the specified 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

while Loop
The while loop executes a block of code as long as the specified condition is true.

Syntax
```
while (condition is true) {
  code to be executed;
}
```
Examples
The example below displays the numbers from 1 to 5:

Example
```
<?php
$x = 1;

while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
}
?>
```
do...while Loop
The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax
```
do {
  code to be executed;
} while (condition is true);
```
Examples
The example below first sets a variable $x to 1 ($x = 1). Then, the do while loop will write some output, and then increment the variable $x with 1. Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:

Example

```
<?php
$x = 1;

do {
  echo "The number is: $x <br>";
  $x++;
} while ($x <= 5);
?>
```

for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (init counter; test counter; increment counter) {
  code to be executed for each iteration;
}
```

Parameters:

init counter: Initialize the loop counter value
test counter: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
increment counter: Increases the loop counter value

Examples

The example below displays the numbers from 0 to 10:

Example

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

foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax

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

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

Examples

The following example will output the values of the given array ($colors):

Example

```
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}?>
```

Array

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is to create an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

Create an Array in PHP

In PHP, the array() function is used to create an array:

```
array();
```

In PHP, there are three types of arrays:

1.Indexed arrays - Arrays with a numeric index
2.Associative arrays - Arrays with named keys
3.Multidimensional arrays - Arrays containing one or more arrays

1.Indexed Arrays

There are two ways to create indexed arrays:

The index can be assigned automatically (index always starts at 0), like this:

```
$cars = array("Volvo", "BMW", "Toyota");
```

or the index can be assigned manually:

```
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

The following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values:

Example

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

2. Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.

There are two ways to create an associative array:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```
or:

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```
The named keys can then be used in a script:

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```
3.Multidimensional Arrays

A multidimensional array is an array containing one or more arrays.

PHP supports multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.
The dimension of an array indicates the number of indices you need to select an element.
For a two-dimensional array you need two indices to select an element
For a three-dimensional array you need three indices to select an element.

GET and POST Method

There are two ways the browser client can send information to the web server. The GET method The POST method.

GET Method: In the GET method, the data is sent as URL parameters that are usually strings of name and value pairs separated by ampersands (&). In general, a URL with GET data will look like this:
Example: Consider the below example:
http://www.example.com/action.php?**name**=*Sam*&**weight**=*55*
Here, the bold parts in the URL denote the GET parameters and the italic parts denote the value of those parameters. More than one parameter=value can be embedded in the URL by concatenating with ampersands (&). One can only send simple text data via GET method.

Example:

```
<?php
error_reporting(0);
if( $_GET["name"] || $_GET["weight"] )
{
  echo "Welcome ". $_GET['name']. "<br />";
echo "You are
".
$_GET['weight
']. " kgs in
weight.";
exit();
}
?>
<html>
```

```
<body>
<form action="<?php $_PHP_SELF ?>" method="GET">
        Name: <input type="text" name="name" />
        Weight:<input type="text" name="weight" />
                <input type="submit" />
</form>
</body>
</html>
```

Advantages:

- It is more secure than GET because user-entered information is never visible in the URL query string or in the server logs.
- There is a much larger limit on the amount of data that can be passed and one can send text data as well as binary data (uploading a file) using POST.

Disadvantages:

- Since the data sent by the POST method is not visible in the URL, so it is not possible to bookmark the page with a specific query.
- POST requests are never cached
- POST requests do not remain in the browser history.

POST Method: In the POST method, the data is sent to the server as a package in a separate communication with the processing script. Data sent through the POST method will not be visible in the URL.

The query string (name/weight) is sent in the HTTP message body of a POST request.

Example:

```
<?php
error_reporting(0);
if( $_POST["name"] || $_POST["weight"] )
        {
                if (preg_match("/[^A-Za-z'-]/",$_POST['name'] ))
                {
                die ("invalid name and name should be alpha");
                }
        echo "Welcome ". $_POST['name']. "<br />";
        echo "You are ". $_POST['weight']. " kgs in weight.";
        exit();
                }
?>
<html>
<body>
<form action = "<?php $_PHP_SELF ?>" method = "POST">
        Name: <input type = "text" name = "name" /> Weight:
        <input type = "text" name = "weight" />
                        <input type = "submit" />
</form>
</body>
</html>
```

Advantages:

- It is more secure than GET because user-entered information is never visible in the URL query string or in the server logs.
- There is a much larger limit on the amount of data that can be passed and one can send text data as well as binary data (uploading a file) using POST.

Disadvantages:

- Since the data sent by the POST method is not visible in the URL, so it is not possible to bookmark the page with a specific query.
- POST requests are never cached
- POST requests do not remain in the browser history.

Sessions

A PHP session is used to store data on a server rather than the computer of the user. Session identifiers or SID is a unique number which is used to identify every user in a session based environment. The SID is used to link the user with his information on the server like posts, emails etc.
Below are different steps involved in PHP sessions:

Starting a PHP Session: The first step is to start up a session. After a session is started, session variables can be created to store information. The PHP **session_start()** function is used to begin a new session.It als creates a new session ID for the user.

Below is the PHP code to start a new session:

```
<?php

session_start();

?>
```

Storing Session Data: Session data in key-value pairs using the **$_SESSION[]** superglobal array. The stored data can be accessed during lifetime of a session.
Below is the PHP code to store a session with two session variables Rollnumber and Name:

```
<?php

session_start();

$_SESSION["Rollnumber"] = "11";
$_SESSION["Name"] = "Ajay";

?>
```

Accessing Session Data: Data stored in sessions can be easily accessed by firstly calling **session_start()** and then by passing the corresponding key to the **$_SESSION** associative array.

The PHP code to access a session data with two session variables Rollnumber and Name is shown below:

```
<?php

session_start();

echo 'The Name of the student is :' . $_SESSION["Name"] . '<br>';
echo 'The Roll number of the student is :' . $_SESSION["Rollnumber"] . '<br>';

?>
```

Destroying Certain Session Data: To delete only a certain session data,the unset feature can be used with the corresponding session variable in the **$_SESSION** associative array. The PHP code to unset only the “Rollnumber” session variable from the associative session array:

```
<?php

session_start();

if(isset($_SESSION["Name"])){
        unset($_SESSION["Rollnumber"]);
}

?>
```

Destroying Complete Session: The **session_destroy()** function is used to completely destroy a session. The session_destroy() function does not require any argument.

```
<?php

session_start();
session_destroy();

?>
```

References:

1. “Internet & Web Designing “by Adesh K. Pandey, Er. Neha Dutta.
2. “Web Technologies” by Uttam K. Roy.
3. “TCP/IP Protocol Suite” by Behrouz A. Forouzan.
4. “Internet for Everyone” by Alexis Leon, Mathews Leon.
5. “Web Enabled Commercial Application Development Using HTML, DHTML, JavaScript, Perl CGI” by Ivan Bayross.
6. “E-Commerce and Mobile Commerce Technologies” by Rahul Srivastava & U S Pandey.
7. “SQL, PL/SQL the Programming Language of Oracle “by Ivan Bayross.
8. https://www.geeksforgeeks.org
9. https://nptel.ac.in
10. https://en.wikipedia.org
11. https://www.w3schools.com