By: N. Selvakumar
Web: www.nselva.com

PHP

1. Introduction to PHP
What is PHP? What is PHP file? How PHP works? And what PHP can do for you?

2. Installation (Web Server - Apache, Database - MySQL, Script Language - PHP)
How to run PHP on your PC?

3. PHP Syntax
The basics of PHP syntax.

4. PHP Variables
Local, global variables & array variables.

5. PHP Operators
How to use the operators in PHP? (+, -, *, /, %, ++, --, =, +=, ==, !=, >=, ||, &&, !, …)

6. PHP Control Structures
How to use if… else…, switch … case and loop statements (for…, foreach…, while…, do...while)?

7. PHP Functions
How to write and call functions in PHP? (User defined functions & build in functions)

8. Using form in PHP
How to get information from form? Form validation & on URL passing parameters.

9. PHP Server Side Includes
It is possible to insert the content of another file into PHP.

10. PHP Session
What is a session variables? How to set session variables? How to get session variables? How to remove session variables?

11. PHP Cookies
Cookies are small amounts of data stored by the user's browser after a request from a server or script. While they are excellent from passing information from page to page, or even from visit to visit cookies do have some limitations.

12. Using PHP to connect database MySQL
Using a system DSN & DSN-less connection (List, Search, Add, Edit and Delete)

13. How to send emails by using PHP?
Send emails (To, From, Subject, Cc, Bcc and Body properties)

14. File Handling
How to open en close a file, how to read a file line by line and how to read a file character by character and upload files to the server.

15. Online Project: Login, List, Search, Add New, Edit, View, Delete functions

1. Introduction to PHP

PHP is an open source programming language makes a great case against its competitors. It has a cheap, fast, reliable, robust and widely supported environment (Cross platform, which means your PHP scripts will run on UNIX, Linux, and Windows server)

The need for dynamic content: The web is no longer static; it is dynamic. As the information content of the web grows, so does the need to make web sites more dynamic. Think of an e-shop that has 10,000 products. The owner has to create 10,000 web pages (one for each product), and whenever anything changes, the owner has to change all those pages. This is not an easy. So, easier to have only one page that created and served the content on the fly from the information about the products stored in a database, depending on the client request.

Now a day's sites have to change constantly and provide up-to-date news, information, stock prices, and customized pages. PHP and MySQL are two ways to make your site dynamic.

What is PHP?

- PHP stands for **P**HP: **H**ypertext **P**reprocessor
- PHP is a server-side scripting language, like ASP, Perl, JSP, Python
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, MS SQL Server, MS Access, Oracle, Sybase, etc.)
- PHP is an open source software (OSS)
- PHP is free to download and use

What is a PHP File?

- PHP files may contain text, HTML tags, styles and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of .php (eg: firstexample.php)

What is MySQL?

- MySQL is a database server
- MySQL is ideal for both small and large applications
- MySQL supports standard SQL statements
- MySQL compiles on a number of platforms
- MySQL is free to download and use

PHP + MySQL

- PHP combined with MySQL are cross-platform (means that you can develop in Windows and serve on a Unix platform)

Why PHP?

- PHP runs on different platforms (Windows, Linux, UNIX, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

2. Installation (Web Server - Apache, Data base - MySQL, Script language - PHP)

Where to Start?

- Install an Apache server on Windows or Linux O/S machine
- Install PHP on Windows or Linux O/S machine
- Install MySQL on Windows or Linux O/S machine

Download PHP: Download PHP for free here: http://www.php.net/downloads.php

Download MySQL Database: Download MySQL for free here: http://www.mysql.com/downloads/index.html

Download Apache Server: Download Apache for free here: http://httpd.apache.org/download.cgi

Download above three software in a single software
AppServ - http://www.appservnetwork.com
XAMPP - https://www.apachefriends.org

You can install above software in default port 80 or any other port 81 or 8080.

Then test whether you have PHP installed properly. Type the following code in a text editor and save it as test.php in a directory accessible by your Web server.

```
<?php
        phpinfo();
?>
```

View it with your web browser (IE or Chrome or Firefox)
http://localhost:8080/test.php or
http://localhost/test.php or
http://localhost:81/test.php

Website Files: "D:\AppServ\www"

Databases: "D:\AppServ\MySQL\data"

Apache Configuration
Apache Configuration File - D:\AppServ\Apache24\conf\httpd.conf

PHP Configuration
php.ini File - D:\AppServ\php\php.ini

MySQL Configuration
MySQL ini file: D:\AppServ\MySQL\my.ini

3. PHP Syntax

A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document. Each code line in PHP must end with a semicolon.

```
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <?php echo "Hello, World!";?>
  </body>
</html>
```

It will produce following result – Hello, World!

If you examine the HTML output of the above example, you'll notice that the PHP code is not present in the file sent from the server to your Web browser. All of the PHP present in the Web page is processed and stripped from the page; the only thing returned to the client from the Web server is pure HTML output.

All PHP code must be included inside one of the three special markup tags ate are recognized by the PHP Parser.

```
<?php PHP code goes here ?>

<?    PHP code goes here ?>

<script language="php"> PHP code goes here </script>
```

There are two basic statements to output text with PHP: **echo** and **print**. In the example above we have used the echo statement to output the text "Welcome to PHP".

Comments in PHP

to make a single-line comment - //
to make a large comment block (multiple lines comments)- /* */

4. PHP Variables

Variables are used for storing values, such as numbers, strings or function results, so that they can be used many times in a script. All variables in PHP start with a $ sign symbol. Variables may contain strings, numbers, or arrays.

```
<?php
        $strname=”Sivakumar”;
        $strmessages ="Welcome ";
        echo $strmessages . “ “ . $strmame;
?>
```

To concatenate two or more variables together, use the dot (.) operator:

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.

Variable Naming Rules

- A variable name must start with a letter or an underscore "_"
- A variable name can only contain alpha-numeric characters and underscores (a-Z, 0-9, and _)
- A variable name should not contain spaces. If a variable name should be more than one word, it should be separated with underscore ($my_string), or with capitalization ($myString)

Meaningful variable names:

String data type variables: strname, straddress, strfirstname, strmessage, strdescription
Decimal data type variables: dblsalary, dbltax, dblprofit, dblamount
Integer data type variables: intage, intyear, inthours
Boolean data type variables: blnstatus, blnflag

The escape-sequence replacements are −

\n is replaced by the newline character
\r is replaced by the carriage-return character
\t is replaced by the tab character
\$ is replaced by the dollar sign itself ($)
\" is replaced by a single double-quote (")
\\ is replaced by a single backslash (\)

Variable Scope

Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of three scope types.

1. Local variables
2. Function parameters
3. Global variables

Arrays:

An array is a data structure that stores one or more similar type of values in a single value. For example if you want to store 100 numbers then instead of defining 100 variables its easy to define an array of 100 length.

```
$arrayday=array("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday");
$arrayday[0]=”Sunday”, $arrayday[1]=”Monday”, $arrayday[2]=” Tuesday”, … $arrayday[6]=”Saturday”
$arrmonth=array("Jan","Feb","Mar","Apr","May","June","July","Aug","Sep","Oct","Nov","Dec");
```

Eg:

```
<?=$arrayday[$weekday]?>

<?php
/* First method to create array. */
$numbers = array( 1, 2, 3, 4, 5);

 foreach( $numbers as $value ) {
echo "Value is $value <br />";
}

/* Second method to create array. */
$numbers[0] = "one";
$numbers[1] = "two";
$numbers[2] = "three";
$numbers[3] = "four";
$numbers[4] = "five";
foreach( $numbers as $value ) {
echo "Value is $value <br />";
}
?>
```

5. PHP Operators

How to use the operators in PHP?

Arithmetic Operators
Addition +, Subtraction -, Multiplication *, Division /, Modulus %, Increment ++, Decrement --

Assignment Operators
=, +=, -=, *=, /=, %=

Comparison Operators
==, !=, >, <, >=, <=

Logical Operators
&&, ||, !

Ternary Operators
? :

1. Assignment Operators

Assignment operators are used to set a variable equal to a value or set a variable by doing an arithmetic operation on itself and another value.

Operator	Examples	Large notation
+=	x+=y	x=x+y
-=	x-=y	x=x-y
=	x=y	x=y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

2. Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=5 x+2	7
–	Subtraction	x=6 10-x	4
/	Division	20/5	4
*	Multiplication	x=4 x*5	20
++	Increment	x=5 x++	x=6

—	Decrement	x=5 x–	x=4
%	Modulus (division remainder)	10%8	2

3. Logical Operators

Operator	Description	Example
&&	and	x=5 y=5 (x < 10 && y > 1) returns true
||	or	x=5 y=5 (x==5 || y==5) returns true
!	not	x=5 y=5 !(x==y) returns false

4. Comparison Operators

Operator	Description	Example
==	is equal to	1==2 returns false
!=	is not equal	1!=2 returns true
<>	is not equal	1<>2 returns true
<	is less than	1<2 returns true
>	is greater than	1>2 returns false
<=	is less than or equal to	1<=2 returns true
>=	is greater than or equal to	1>=2 returns false

5. Ternary Operators (? :)

Ternary operator logic is the process of using "(condition) ? (true return value) : (false return value)" statements to shorten your if/else structures.

```
$var = 5;
$var_is_greater_than_two = ($var > 2 ? true : false); // returns true
```

6. PHP Control Structures

Control structures allow you to control the flow of execution of your scripts. You can specify that some code should be executed only under certain condition, using conditional structures. You can specify that some code should be executed repeatedly, using looping structures.

The if ... else 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.

In PHP three type of if...statements.

1. if (condition or conditions)
2. if (condition or conditions) …. else …..
3. if (condition or conditions) …. elseif (condition or conditions) ….. else …..

1. if (condition or conditions)
Syntax

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

Eg:

```
if ($ICNum>500)
        $ICNum=$ICNum-500;
```

2. if (condition or conditions) ... else .....
Syntax

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

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

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

3. if (condition or conditions) ... elseif (condition or conditions) ..... else ....
If you want to execute some code if one of several conditions are true use the elseif statement
Syntax

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

Eg:
The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":

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

The switch Statement
If you want to select one of many blocks of code to be executed, use the switch statement.
The switch statement is used to avoid long blocks of if..elseif..else code.
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:

- A single expression (most often a variable) is evaluated once
- The value of the expression is compared with the values for each case in the structure
- If there is a match, the code associated with that case is executed
- After a code is executed, break is used to stop the code from running into the next case
- The default statement is used if none of the cases are true

```
<?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";
}
?>
```

Looping Structures

Looping structures allow you to execute the same block of code repeatedly. The number of times it executes may be fixed or may be based on one or more conditions.
In PHP we have the following looping statements:

- 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 - 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

The for Statement

The **for loop** 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 loop statement has three parameters. The first parameter initializes variables, the second parameter holds the condition, and the third parameter contains the increments required to implement the loop. If more than one variable is included in the initialization or the increment parameter, 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:

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

The foreach Statement
The foreach statement is used to loop through arrays.
For every loop, the value of the current array element is assigned to $value (and the array pointer is moved 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:

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

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:

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

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 as long as it has a value of less than 5:

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

7. PHP Functions

Repeating the same script more than once in a page, then it is a good idea to convert the script into a function. This has the dual benefit of reducing typing and making your code easier to read.

Functions come in two varieties:

- User-defined functions
- Built-in functions

User-defined functions

Creating PHP functions:

- All functions start with the word "function()"
- Name the function - It should be possible to understand what the function does by its name. The name can start with a letter or underscore (not a number)
- Add a "{" - The function code starts after the opening curly brace
- Insert the function code
- Add a "}" - The function is finished by a closing curly brace

Example 1:

You declare a function by using the function statement like this:

```
function power($a, $b)
{
        $p=1
        for ($i=1;$i<=b;$i++)
                $p*=$a;
        return $p;
}
```

This function takes two numbers as inputs and returns the power of the numbers as the output. After you have declared this function anywhere within a page, you can use the function anywhere in the page like this:

```
echo power(2,5);
echo power(3,4);
```

Example 2:

```
function details($fname, $address)
{
        echo "Your name is $name<br />";
        echo "Your Address is $ address <br />";
}
```

Call functions in PHP page

```
details("Selva","Colombo-3");
details("Karan","Colombo-6");
```

Example 3:

```
function printContactDetails()
{
        echo "N. Selvakumar<br>";
        echo "Colombo-3<br>";
        echo "TP: 111111<br>";
        echo "**********<br>";
}
```

Call functions in PHP page

```
printContactDetails();
printContactDetails();
```

Built-in functions
Quick look at the more important built in PHP functions. They include functions for date manipulation, mathematical, string manipulation, and more.

PHP Date Functions

Format a local time/date:

string **date** (string format [, int timestamp])

Returns a string formatted according to the given format string using the given timestamp or the current local time if no timestamp is given.

The first parameter in the date() function specifies how to format the date/time. It uses letters to represent date and time formats. Here are some of the letters that can be used:

d - The day of the month (01-31)
m - The current month, as a number (01-12)
Y - The current year in four digits

```
<?php
echo date("m/d/Y"); - output: 01/16/2007
echo "<br />";
echo date("m.d.y"); - output: 01.16.07
echo "<br />";
echo date("M-d-Y"); - Jan-16-2007
?>
```

The following characters are recognized in the format string:

- d - day of the month, 2 digits with leading zeros; i.e. "01" to "31"
- D - day of the week, textual, 3 letters; i.e. "Fri"
- l (lowercase 'L') - day of the week, textual, long; i.e. "Friday"
- w - day of the week, numeric, i.e. "0" (Sunday) to "6" (Saturday)
- m - month; i.e. "01" to "12"
- n - month without leading zeros; i.e. "1" to "12"
- M - month, textual, 3 letters; i.e. "Jan"
- F - month, textual, long; i.e. "January"
- j - day of the month without leading zeros; i.e. "1" to "31"
- y - year, 2 digits; i.e. "99"
- Y - year, 4 digits; i.e. "1999"
- S - English ordinal suffix, textual, 2 characters; i.e. "th", "nd"
- L - boolean for whether it is a leap year; i.e. "0" or "1"
- z - day of the year; i.e. "0" to "365"

- h - hour, 12-hour format; i.e. "01" to "12"
- H - hour, 24-hour format; i.e. "00" to "23"
- g - hour, 12-hour format without leading zeros; i.e. "1" to "12"
- G - hour, 24-hour format without leading zeros; i.e. "0" to "23"
- i - minutes; i.e. "00" to "59"
- s - seconds; i.e. "00" to "59"
- t - number of days in the given month; i.e. "28" to "31"
- a - "am" or "pm"
- A - "AM" or "PM"

Example 1. Date() example

echo (date ("l dS F Y h:i:s A")); - output: Thursday 04th January 2007 05:40:49 PM

mktime()

mktime(hrs, mins, secs, month, date, year)

echo ("July 22, 1983 is on a " . date ("l", mktime(0,0,0,7,22,1983)));

- output: July 22, 1983 is on a Saturday

It is possible to use **date()** and **mktime()** together to find dates in the future or the past.

Example 2. Date() and mktime() example

$tomorrow = mktime (0,0,0,date("m") ,date("d")+1,date("Y"));

$lastmonth = mktime (0,0,0,date("m")-1,date("d"), date("Y"));

$nextyear = mktime (0,0,0,date("m"), date("d"), date("Y")+1);

checkdate () - Validate a date/time
int **checkdate** (int month, int day, int year)
Returns true if the date given is valid; otherwise returns false.

Checks the validity of the date formed by the arguments. A date is considered valid if:

- year is between 0 and 32767 inclusive
- month is between 1 and 12 inclusive
- Day is within the allowed number of days for the given month. Leap years are taken into consideration.

PHP String Functions

- chr -- Return a specific character – eg: chr(67) - C
- split/ str_split -- Convert a string to an array – eg: $arrayTxt=split("-","121-23-56-78-99")
- join -- Convert an array to a string - eg: $strTxt=join("-",$arrayTxt)
- trim -- remove space from the beginning and end of a string
- ltrim -- remove space from the beginning of a string
- rtrim -- remove space from the end of a string
- str_repeat -- Repeat a string - eg: str_repeat("Jaffna",30)
- str_replace -- Replace all occurrences of the search string with the replacement string
- strlen -- Get string length – eg: strlen("Jaffna")
- strtolower -- Make a string lowercase
- strtoupper -- Make a string uppercase
- substr_count -- Count the number of substring occurrences
- substr_replace -- Replace text within a portion of a string
- substr -- Return part of a string
- ucfirst -- Make a string's first character uppercase
- ucwords -- Uppercase the first character of each word in a string

& etc…

PHP Math Functions

abs -- Absolute value - eg: abs(-56) = 56

ceil -- Round fractions up - eg: ceil(7.4) = 8

floor -- Round fractions down - eg: floor(7.4) = 7

round -- Rounds a float - eg: round(7.4) = 7, round(7.5) = 8, round(7.7) = 8

bindec() - Converts a binary number to a decimal number - eg: bindec(1001) = 9

octdec -- Octal to decimal - eg: octdec(16) = 14

decbin -- Decimal to binary - eg: decbin(9) = 1001

dechex -- Decimal to hexadecimal - eg: dechex(24) = 18

decoct -- Decimal to octal - eg: decoct(11) = 13

max -- Find highest value - eg: max(23,56,99,12,5,89) = 99, max(25,16,9) = 25

min -- Find lowest value - eg: min(23,56,99,12,5,89) = 5, min(25,16,9) = 9

pi -- Get value of pi - eg: pi = 3.14

pow -- Exponential expression - eg: pow(3,4) = 81

rand -- Generate a random integer - eg: rand() = 3455

sqrt -- Square root - eg: sqrt(9) = 3

& etc…

8. PHP Forms and User Input

The PHP $_GET, $_POST and $_REQUEST variables are used to retrieve information from forms, like user input.

The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will automatically be available to your PHP scripts.

Form example:

```
<form action="welcome.php" method="post">
Name: <input type="text" name="name" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

The example HTML page above contains two input fields and a submit button. When the user fills in this form and click on the submit button, the form data is sent to the "welcome.php" file.

```
Welcome <?php echo $_POST["name"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old.
```

PHP $_GET

The $_GET variable is used to collect values from a form with method="get". The $_GET variable is used to collect values from a form with method="get". Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and it has limits on the amount of information to send (max. 500 characters).

Why use $_GET?

Note: When using the $_GET variable all variable names and values are displayed in the URL. So this method should not be used when sending passwords or other sensitive information. However, the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

Note: The GET method is not suitable on large variable values; the value cannot exceed 500 characters.

Why use $_POST?

- Variables sent with HTTP POST are not shown in the URL
- Variables have no length limit

However, because the variables are not displayed in the URL, it is not possible to bookmark the page. 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 $_REQUEST Variable

The PHP $_REQUEST variable contains the contents of both $_GET, $_POST, and $_COOKIE.

Example

```
Welcome <?php echo $_REQUEST["name"]; ?>.<br />
You are <?php echo $_REQUEST["age"]; ?> years old!
```

You can validate the form input on two places, client side and server side.

Client side form validation usually done with javascript. Client side validation makes your web application respond 'faster' while server side form validation with PHP can act as a backup just in case the user switch off javascript support on her browser. And since different browsers can behave differently there is always a possibility that the browser didn't execute the javascript code as you intended.

Some things you need to check:

- empty values
- numbers only
- input length
- email address

On URL passing values:

```
        <a href=”contents.php?var1=jaffna&var2=srilanka”>Jaffna</a>
        <a href=”contents.php?var1=colombo&var2=srilanka”>Colombo</a>
        <a href=”contents.php?var1=vanni&var2=srilanka”>Vanni</a>
        <a href=”contents.php?var1=trinco&var2=srilanka”>Trinco</a>
        <a href=”contents.php?var1=kandy&var2=srilanka”>Kandy</a>
<?php
        $strcity= $_GET["var1"];
        $strcountry= $_GET["var2"];
?>


function CheckFrm()
{
if (document.frm.frameid.value=="")
{
alert("Please select a Frame");
document.frm.frameid.focus();
return false;
}
 return true;
}
```

9. PHP Server Side Includes

The **include** command is very similar to using a Server Side Includes (SSI) include. It calls a block of information from a different page into the current webpage. SSI used to create functions, headers, footers, or elements that will be reused on multiple pages.

You can insert the content of a file into a PHP file before the server executes it, with the include() or require() function. The two functions are identical in every way, except how they handle errors. The include() function generates a warning (but the script will continue execution) while the require() function generates a fatal error (and the script execution will stop after the error).

This can save the developer a considerable amount of time. This means that you can create a standard header or menu file that you want all your web pages to include. When the header needs to be updated, you can only update this one include file, or when you add a new page to your site, you can simply change the menu file (instead of updating the links on all web pages).

The include‟” or include_once‟” Function

The include() function takes all the text in a specified file and copies it into the file that uses the include function.

Example1: Assume that you have a standard header file, called "header.php". To include the header file in a page, use the include() function, like this:

header.php
```
<?php
 echo “Selva";
Echo “Col-3”;
?>
```

```
<html>
<body>
<?php include("header.php"); ?>
<h1>Welcome to my home page</h1>
<p>Some text</p>
</body>
</html>
```

The require ‟” or require_once ‟” Function

The require() function is identical to include(), they only handle errors differently.

The include() function generates a warning (but the script will continue execution) while the require() function generates a fatal error (and the script execution will stop after the error).

If you include a file with the include() function and an error occurs, you might get an error message like the one below.

One of the many uses of includes is to set common webpage areas into seperate files and include them into each page like a template style layout. This site uses php includes. One for the main header area, one for the navigations, and one for the footer area.

10. PHP Session

During a complex project, there are times when you want to send data from one web-page to another. For example, in an e-commerce web-site, it is essential that you store the contents of a shopping cart while the user is browsing your site. In order to do this, there are two easy ways: you either use cookies or sessions.

What is a Session?

Session: An abstract concept to represent a series of HTTP requests and responses exchanged between a specific Web browser and a specific Web server. Session concept is very useful for Web based applications to pass and share information from one Web page (request) to another Web page (request).

Since the current design of HTTP protocol does not support session concept, all Web server side scripting technologies, including PHP, have designed their own way to support session concept. The key design element of session support is about how to identify a session and how to maintain the session ID (identification). One common way to maintain the session ID is use the cookie technology.

PHP's Session Support

PHP manages the session ID with as a cookie, a GET variable, or a POST variable. It offer a built-in array as the session object, and a number of built-in functions to allow the PHP script to interact with the session:

- $_SESSION - A built-in array to store and share variables for the session.
- session_start() - A built-in function to create a new session or resume an existing session based on the current session id that's being passed via a request, such as GET, POST, cookie.
- session_name() - A built-in function to set and get the session name.
- session_id() - A built-in function to set and get the session ID.
- session_destroy() - A built-in function to destroy all variables stored in $_SESSION.

Starting a PHP Session

Before you can store user information in your PHP session, you must first start up the session.

Note: The session_start() function must appear BEFORE the <html> tag:

```
<?php session_start(); ?>
```

Storing a Session Variable

```
The correct way to store and retrieve session variables is to use the PHP $_SESSION variable:
<?php
session_start();
// store session data
$_SESSION['views']=1;
?>
<html>
<body>
<?php
//retrieve session data
echo "Pageviews=". $_SESSION['views'];
?>
</body>
</html>
```

Destroying a Session

If you wish to delete some session data, you can use the unset() or the session_destroy() function.

The unset() function is used to free the specified session variable:

```
<?php
unset($_SESSION['views']);
?>
```

You can also completely destroy the session by calling the session_destroy() function:

```
<?php
session_destroy();
?>
```

Note: session_destroy() will reset your session and you will lose all your stored session data.

11. PHP Cookies

Cookies are small amounts of data stored by the user's browser after a request from a server or script. While they are excellent from passing information from page to page, or even from visit to visit, cookies do have some limitations. For example, the maximum number of cookies from a host that can be stored by a browser is 20, and the maximum cookie size is 4KB. The main thing about cookies is that only the originating host can read the stored data, so the user's privacy is respected.

Cookies consist of a name, value, expiry date, host and path information, and they end up to the user because they are send from the server thru an HTTP header.

How to Create a Cookie?

The setcookie() function is used to set a cookie.

Note: The setcookie() function must appear BEFORE the <html> tag.

Syntax

```
setcookie(name, value, expire, path, domain);
```

Example
In the example below, we will create a cookie named "user" and assign the value "Siva" to it. We also specify that the cookie should expire after one hour:

```
<?php
setcookie("user", "Siva", time()+3600);
?>
```

How to Retrieve a Cookie Value?

The PHP $_COOKIE variable is used to retrieve a cookie value.

In the example below, we retrieve the value of the cookie named "user" and display it on a page:

```
<?php
// Print a cookie
echo $_COOKIE["user"];
// A way to view all cookies
print_r($_COOKIE);
?>
```

In the following example we use the isset() function to find out if a cookie has been set:

```
<?php
if (isset($_COOKIE["user"]))
  echo "Welcome " . $_COOKIE["user"] . "!<br />";
else
  echo "Welcome guest!<br />";
?>
```

How to Delete a Cookie?

When deleting a cookie you should assure that the expiration date is in the past.
Delete example:

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time()-3600);
?>
```

What if a Browser Does NOT Support Cookies?

If your application deals with browsers that do not support cookies, you will have to use other methods to pass information from one page to another in your application. One method is to pass the data through forms (forms and user input are described earlier in this tutorial).

The form below passes the user input to "welcome.php" when the user clicks on the "Submit" button:

```
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
</body>
</html>
```

Retrieve the values in the "welcome.php" file like this:

```
<html>
<body>
Welcome <?php echo $_POST["name"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old.
</body>
</html>
```

```
if (!headers_sent())
    header('Location: index.php');
```

12. Using PHP to connect database MySQL

The power of PHP comes through when we tie databases to our web site. mysql_connect() or mysqli_connect or odbc_connect() can be used to connect to MySQL database but most of the developers are using mysql_connect().

MySQL is the database most commonly used with PHP. Its speed, reliability, robust Open Source database and ease of use make it an excellent choice for building Web-based applications. Additional features such as a platform-independent data format, ODBC support, and a rich set of built-in functions round out the usefulness of the database.

In PHP can use other databases. Eg: MS SQL Server, MS Access, Oracle, etc

Opening a connection to MySQL database from PHP is easy. Just use the mysql_connect() function like this:

```
// connect to server
$conn = mysql_connect("localhost","user name","password") or die ('Error connecting to MySQL');
```

localhost is the name of MySQL server name. When your web server is on the same machine with the MySQL server you can use localhost or 127.0.0.1. The "user name" and "password" are valid MySQL user name and password.

```
// select the "ExampleDB" database
mysql_select_db("ExampleDB", $conn);
```

mysql_close() closes a MySQL connection opened by mysql_connect(). The connection to close is specified with the connection argument. If no argument is specified, the last opened connection is closed.

```
mysql_close($conn);
```

Create MySQL Database with PHP

To create a database use the mysql_query() function to execute an SQL query like this

```
<?php
$conn = mysql_connect("localhost","root","123") or die ('Error connecting to MySQL');
$query  = "CREATE DATABASE FirstDB";
mysql_query($query);
mysql_close($conn);
?>
```

If you want to create tables in the database you just created don't forget to call mysql_select_db() to access the new database.

Note: some webhosts require you to create a MySQL database and user through your website control panel (such as CPanel). If you get an error when trying to create database this might be the case.

Creating the Tables

To create tables in the new database you need to do the same thing as creating the database. First create the SQL query to create the tables then execute the query using mysql_query() function.

```
<?php
$conn = mysql_connect("localhost","root","") or die ('Error connecting to MySQL');
mysql_select_db('FirstDB') or die('Cannot select database');
$query = 'CREATE TABLE tblUser ( '.
        'cid INT NOT NULL AUTO_INCREMENT, '.
```

```
        'cname VARCHAR(20) NOT NULL, '.
        'cemail VARCHAR(50) NOT NULL, '.
        'csubject VARCHAR(30) NOT NULL, '.
        'cmessage TEXT NOT NULL, '.
        'PRIMARY KEY(cid))';
mysql_query($query);
mysql_close($conn);

?>
```

Deleting a Database

As with creating a database, it is also preferable to use mysql_query() and to execute the SQL DROP DATABASE statement instead of using mysql_drop_db()

```
<?php
$conn = mysql_connect("localhost","root","") or die ('Error connecting to MySQL');
$query  = "DROP DATABASE FirstDB";
mysql_query($query);
mysql_close($conn);
?>
```

Deleting a Table

```
$query  = "DROP TABLE tblUser";

mysql_query($query);
```

After the database and the tables are ready it's time to put something into the table.

Insert data to MySQL database

Inserting data to MySQL is done by using mysql_query() to execute INSERT query. Note that the query string should not end with a semicolon. Below is an example of adding a new MySQL user by inserting a new row into table user in database mysql:

```
<?php
$conn = mysql_connect("localhost","root","") or die ('Error connecting to MySQL');
mysql_select_db('FirstDB') or die('Cannot select database');
$query = "INSERT INTO tblUser (cname, cemail, csubject) VALUES ('Selva','an@abc.com','test')";
mysql_query($query) or die('Error, insert query failed');
mysql_close($conn);
?>
```

eg:

```
<form method="post" action="AddRec.php">
Name:<input type="Text" name="cname"><br>
Email:<input type="Text" name="cemail"><br>
Subject:<input type="Text" name="csubject"><br>
Message:<input type="Text" name="cmessage"><br>
<input type="Submit" name="submit" value="Enter information">
</form>
<?php
if($submit)
{
        $db = mysql_connect("localhost", "root","");
        mysql_select_db("FirstDB",$db);
        $sql = "INSERT INTO tblUser (cname, cemail, csubject, cmessage)
        VALUES ('$cname','$cemail','$csubject','$cmessage')";
```

```
        mysql_query($sql) or die('Error, insert query failed');
        echo "Thank you! Information entered.\n";
}
?>
```

MySQL Update and Delete
There are no special ways in PHP to perform update and delete on MySQL database. You still use mysql_query() to execute the UPDATE or DELETE statement.

Update data to MySQL database

```
<?php
$conn = mysql_connect("localhost","root","") or die ('Error connecting to MySQL');
mysql_select_db('FirstDB') or die('Cannot select database');
$query = "update tblUser set cname='Kumar' where cid=3";
mysql_query($query) or die('Error, insert query failed');
mysql_close($conn);
?>
```

Delete data from MySQL database

```
<?php
$conn = mysql_connect("localhost","root","") or die ('Error connecting to MySQL');
mysql_select_db('FirstDB') or die('Cannot select database');
$query = "delete from tblUser where cid=3";
mysql_query($query) or die('Error, insert query failed');
mysql_close($conn);
?>
```

Get data from table (MySQL database)
Using PHP you can run a MySQL SELECT query to fetch the data out of the database. You have several options in fetching information from MySQL. PHP provide several functions for this. The first one is mysql_fetch_array()which fetch a result row as an associative array, a numeric array, or both.

```
<?php
$conn = mysql_connect("localhost","root","") or die ('Error connecting to MySQL');
mysql_select_db('FirstDB') or die('Cannot select database');
$query  = "SELECT cname, csubject, cmessage FROM tblUser";
$result = mysql_query($query) or die('Invalid table name');
while($row = mysql_fetch_array($result))
{
?>
 Name: <?=$row['cname']?><br>
Subject: <?=$row['csubject']?><br>
Message: <?=$row['cmessage']?><br><br>
<?php
}
mysql_close($conn);
?>
```

If there are no records then display "There are no data" message.

```
<?php
$conn = mysql_connect("localhost","root","") or die ('Error connecting to MySQL');
mysql_select_db('FirstDB') or die('Cannot select database');
$query  = "SELECT cname, csubject, cmessage FROM tblUser";
$result = mysql_query($query) or die('Invalid table name');

if ($row = @mysql_fetch_array($query))
{
```

```
        while($row = mysql_fetch_array($result))
        {
        ?>
        Name: <?=$row['cname']?><br>
        Subject: <?=$row['csubject']?><br>
        Message: <?=$row['cmessage']?><br><br>
        <?php
        }
}
else
{
?>
There are no data
<?php
}
?>
mysql_close($conn);
?>
```

Login Page:

```
$sqlquery="SELECT * FROM Users where uName ='" . $strUName ."' and Pass='" . $strPass"'";
$process= mysql_query($sqlquery);
$numrows = mysql_num_rows($process);
echo mysql_error();
if ($numrows > 0)
{
         $row = @mysql_fetch_array($process);
         session_start();
         $_SESSION['UserId']=$row["ID"];
         header("Location: listusers.php");
 }
 else
 {
        echo "Please enter the correct User Name & Password!";
 }
```

13. How to send emails by using PHP?

Send emails (To, From, Subject, Cc, Bcc and Body properties)

Mail() automatically mails the message specified in message to the receiver specified in to. Multiple recipients can be specified by putting a comma between each address in to.

PHP makes sending information to an email rather easy. It takes one main command:
mail ("recipient","subject","message");

The first value is an email address - Where the information is going to be sent to.

The second value is a short sentence or note - It will appear in the SUBJECT field of the email.

The third value is the main body message

Check with your host if they support the mail command. There must be a specified working local mail server in the php.ini file to work.

Example 1. Sending mail.
mail("anselva@yahoo.com", "My Subject", "Hi Selva

 ….. ");
If a fourth string argument is passed, this string is inserted at the end of the header. This is typically used to add extra headers. Multiple extra headers are separated with a newline.

Headers are the parts seen at the top of emails:

To:	A comma seperated list of recipient emails.
From:	The senders email address.
Reply-To:	The email address where replies should be sent to.
Return-Path:	Kinda the same thing as the Reply-To. Some email clients require this, others create a default.
Subject:	Subject of the email.
CC:	Carbon Copy. A comma seperated list of more recipients that will be seen by all other recipients.
BCC:	Blind Carbon Copy. A comma seperated list of more recipients that will not be seen by any other recipients.

```
<?php
$to = "yourplace@somewhere.com";
$subject = "My email test.";
$message = "Hello, how are you?";
$headers = "From: myplace@here.com\r\n";
$headers .= "Reply-To: myplace2@here.com\r\n";
$headers .= "Return-Path: myplace@here.com\r\n";
$headers .= "CC: sombodyelse@noplace.com\r\n";
$headers .= "BCC: hidden@special.com\r\n";
if (mail($to,$subject,$message,$headers) ) {
  echo "The email has been sent!";
}
else {
  echo "The email has failed!";
}
?>
```

14. File Handling

How to open en close a file, how to read a file line by line and how to read a file character by character and upload files to the server.

Before we look how to open a file in PHP you need to know that a file can be opened in different modes. For example you can open a file in read only mode or in read and write modes. Take a look at the table below for the different modes:

Modes	Description
r	Read only. Starts at the beginning of the file
r+	Read/Write. Starts at the beginning of the file
w	Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist
w+	Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist
a	Append. Opens and writes to the end of the file or creates a new file if it doesn't exist
a+	Read/Append. Preserves file content by writing to the end of the file
x	Write only. Creates a new file. Returns FALSE and an error if file already exists
x+	Read/Write. Creates a new file. Returns FALSE and an error if file already exists

Opening a File

```
$file=fopen("test.txt","w");
```

Closing a File

```
$file = fopen("test.txt","r");
//do something with the file test.txt
fclose($file);
```

Reading a File Line by Line

```
$file = fopen("test.txt", "r") or exit("Unable to open the file!");
while(!feof($file))
{
echo fgets($file). "<br />";
}
fclose($file);
```

Reading a File Character by Character

```
$file = fopen("test.txt", "r") or exit("Unable to open the file!");
while(!feof($file))
{
echo fgetc($file);
}
fclose($file);
```

15. Online Project: Login, List, Search, Add New, Edit, View, Delete functions