



1 : Server Side Web Development using PHP & MySQL

IT3406 – Web Application Development II

Level II - Semester 3

Topic

- 1.1. Install PHP in a windows/ Linux environment [Ref 10: Pg. (40-54)]
- 1.2. Explain basic features of PHP [Ref 1: Pg. (310-323, 325-343)] [Ref 10: Pg. (271-316)]
 - 1.2.1. PHP syntax and semantics
 - 1.2.1.1. Variables [Ref 1: Pg. (311)] [Ref 10 : Pg. (288-296)]
 - 1.2.1.2. Constants [Ref 10 : Pg. (287)]
 - 1.2.1.3. Conditional statements [Ref 1: Pg. (325-330)]
 - 1.2.1.4. Loops [Ref 1: Pg. (331-334)]
 - 1.2.1.5. Functions [Ref 1: Pg. (336)]

Topic

- 1.2.2. Arrays and data processing with arrays [Ref 1: Pg. (206)] [Ref 10 : Pg. (296-305)]
- 1.2.3. Handling HTML forms with GET and POST operations [Ref 1: Pg. (343)]
- 1.2.4. Form validation fields (including URLs and email address) and required fields [Ref 10: Pg. (574-585)]
- 1.2.5. Filtering inputs (validate and sanitize external inputs) [Ref 1: Pg. (384-389)] [Ref 10: Pg. (432)]
- 1.2.6. Session control and cookies (create and retrieve a cookie) PHP [Ref 1: Pg.(419-435)][Ref 10: Pg. (437-446)]
- 1.2.7. File handling (Open, read, create, write operations with files, upload files) PHP [Ref 10: Pg. (366-368)]
- 1.2.8. Sending emails using PHP [Ref 11]
- 1.2.9. Object Orientation with PHP [Ref 1. Pg. (395-418)]
- 1.3. Use web services with PHP [Ref 10: Pg. (541-553)]

Topic

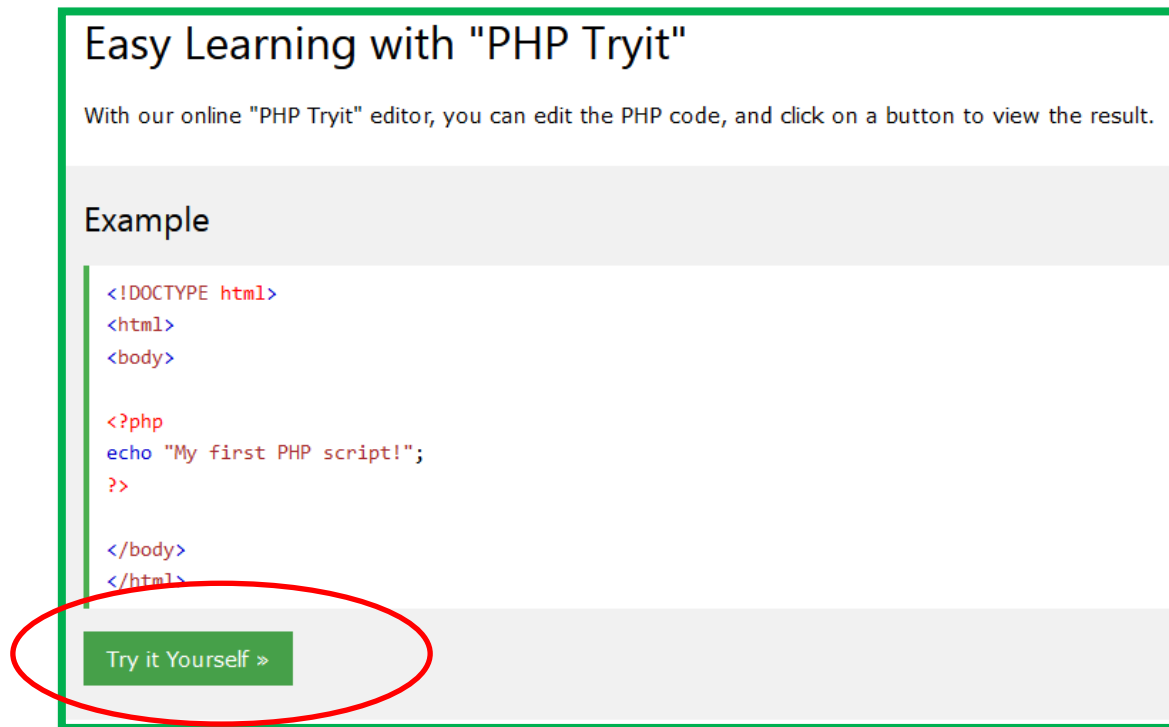
- 1.4. Develop a web application with PHP [Ref 9: Pg. (604-636)]
- 1.5. Setting up MySQL [Ref 1: Pg. (470-474)] [Ref 10: Pg. (56-74)]
- 1.6. Connect to MySQL database [Ref 1: Pg. (513)] [Ref 10: Pg. (515-527)] [Ref 2: Pg. (165)]
- 1.6. Explain MVC architecture and differentiate available PHP frameworks [Ref 1: Pg. (683)] [Ref 3: Pg. (1-4)] [Ref 3: Pg. (345-441)]/[Ref. 4: Pg. (83-105)]
- 1.7. MYSQL database operations - Read/modify/delete/search operations [Ref 1: Pg. (497-513)] [Ref 2: Pg. (206)]
- 1.7.1. Processing forms (Create, Read/Retrieve, Update, and Delete operations) [Ref 2: Pg. (235-250)]
- 1.8. Use of PHP unit testing tools for testing automation of front end web applications and manual testing of applications. [Ref 1: Pg. (616)]
- 1.9. Consider server-side security threats eg: data spoofing, invalid data, unauthorized file access [Ref 1: Pg. (375-382)][Ref 10: Pg. (425-436)]
- 1.10. Handle vulnerability solutions eg: data sanitizing and validation [Ref 1: Pg. (384-389)]

PHP Programming/Scripting language

- PHP is a widely used web programming/scripting language which is free.
- There are more than one way to start learning with PHP code.
 1. Use an online interpreter accessed by the browser
 2. Use a server accessed through a network and transfer code
 3. Install PHP in the local machine:
 - ❖ Run the code from the command line
 - ❖ Execute through the web server integration
- As a beginner easiest way is “trying out” PHP code without installing and configuring it from first method.
- We will first look at the online interpreter.

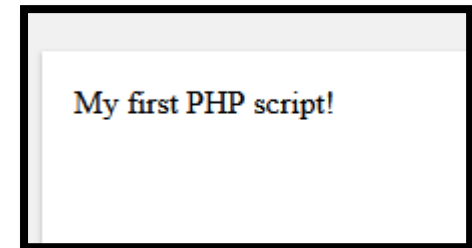
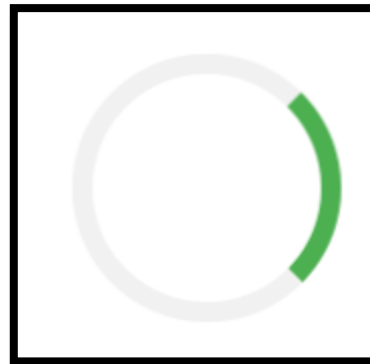
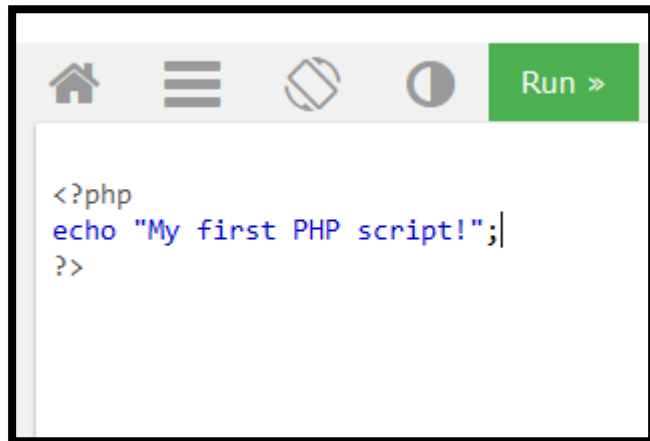
Activity: PHP Programming with online interpreter

- Go to PHP area in w3schools website:
<https://www.w3schools.com/php/>
- Click “Try it Yourself” to launch the interpreter.



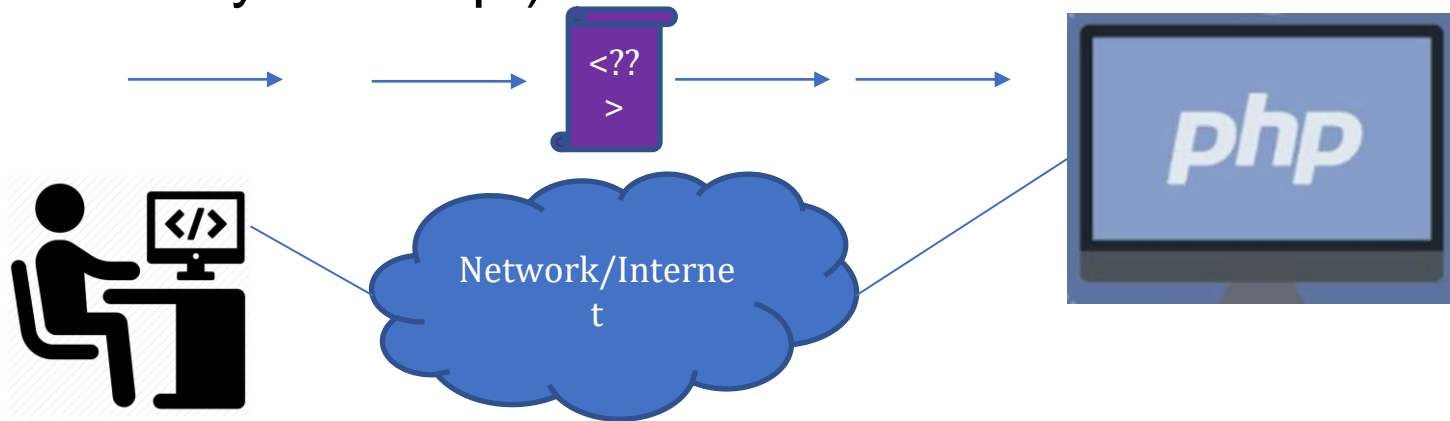
Activity: PHP Programming with online interpreter

- Write the PHP code in the left hand side text area.
- Click “Run” button to start processing (an animation will start).
- See the output in the right hand side of the page.



PHP Programming with networked server

- Provided you have access to a **server** that supports PHP scripting language you can do the following:
- Write your code in an editor, save it in the local machine.
- Transfer it to the correct directory of the **server** via FTP.
- Access the server URL from the web browser (with the name of your script)



Installing PHP in a windows environment

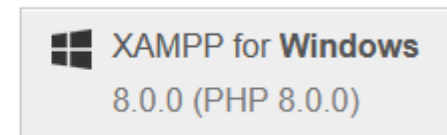
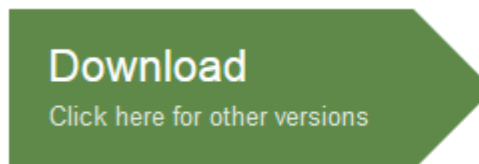
- There are more than one method to install PHP.
- PHP binaries can be downloaded from php.net.
- <https://windows.php.net/download#php-8.0>



- There are x86/x64 and Thread-safe and non-thread-safe builds.
- Alternatively, there are prepackaged binaries with PHP and MySQL.

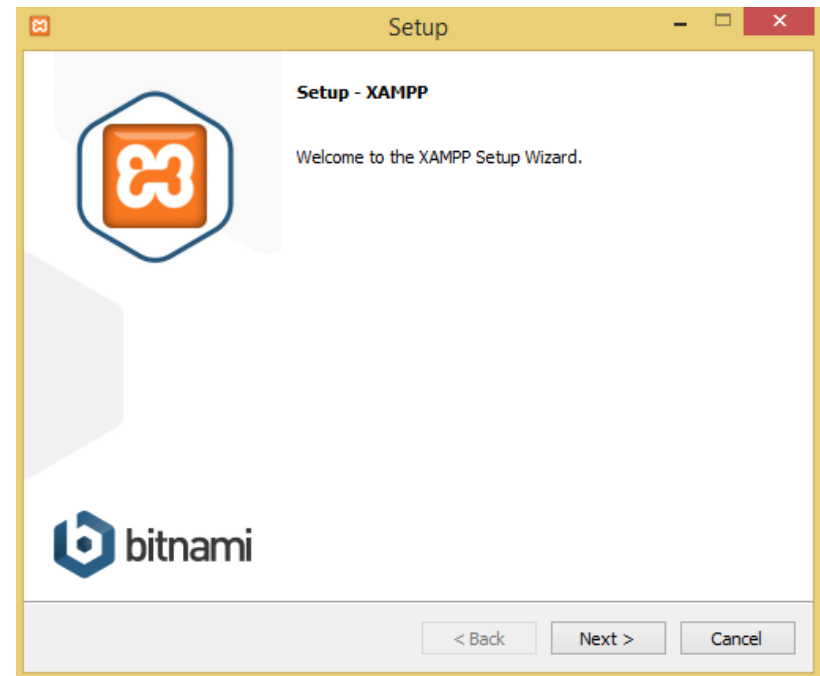
Activity : Installing PHP with XAMPP in Windows

- XAMPP is a popular Apache/PHP/MySQL bundle for popular platforms.
- XAMPP installation step 1:
 - Download XAMPP from :
<https://www.apachefriends.org/index.html>
 - Click the appropriate platform to download the binaries for your platform
 - We will download the windows binaries for this activity.



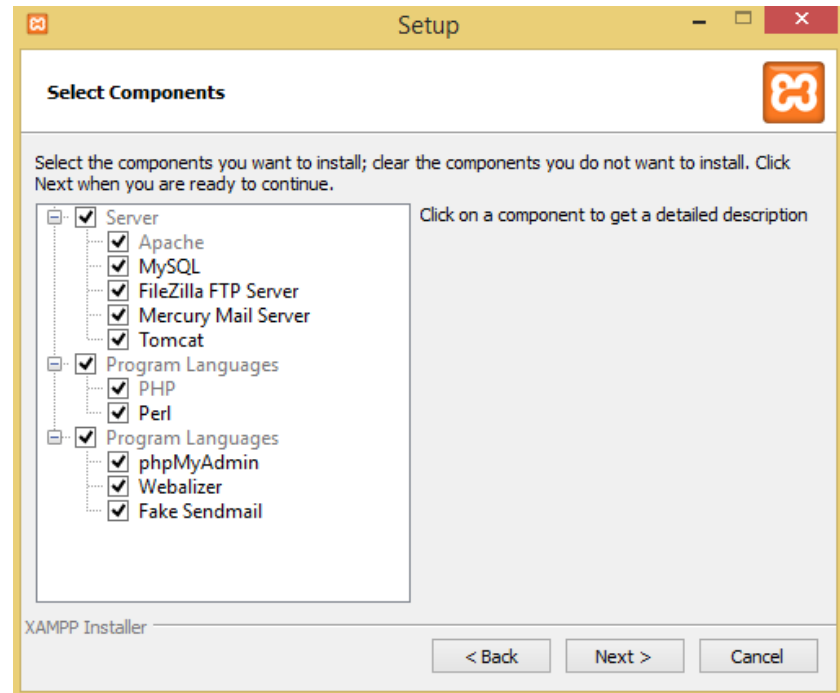
Activity : Installing PHP with XAMPP in Windows

- XAMPP installation step 2.
 - Run the installer executable file by double clicking.
 - The following dialog will appear for XAMPP version for PHP 8.0.
 - Click **next** to continue.



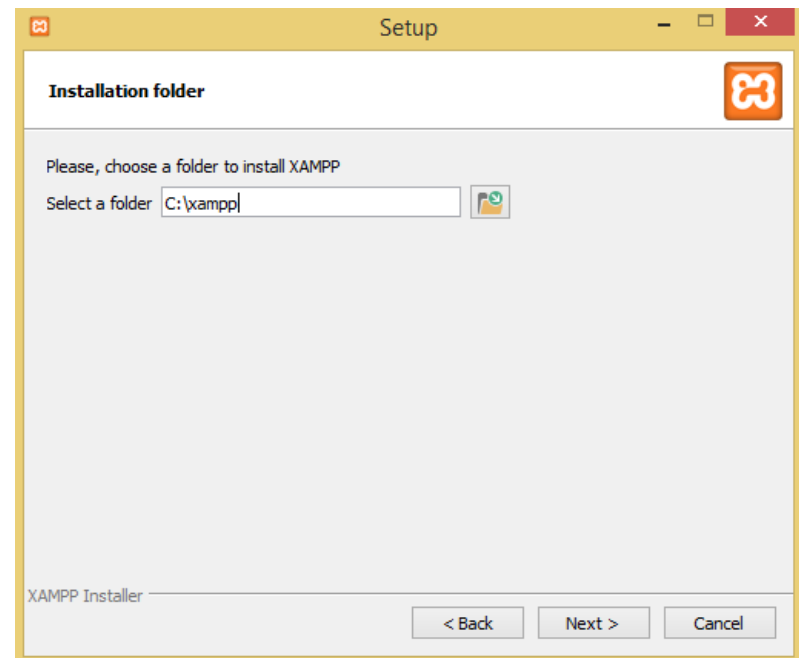
Activity : Installing PHP with XAMPP in Windows

- XAMPP installation step 3:
 - In this dialog, select the options that you prefer or unselect the options you do not prefer.
 - For this activity we will use the default selection.
 - Click **next** to continue.



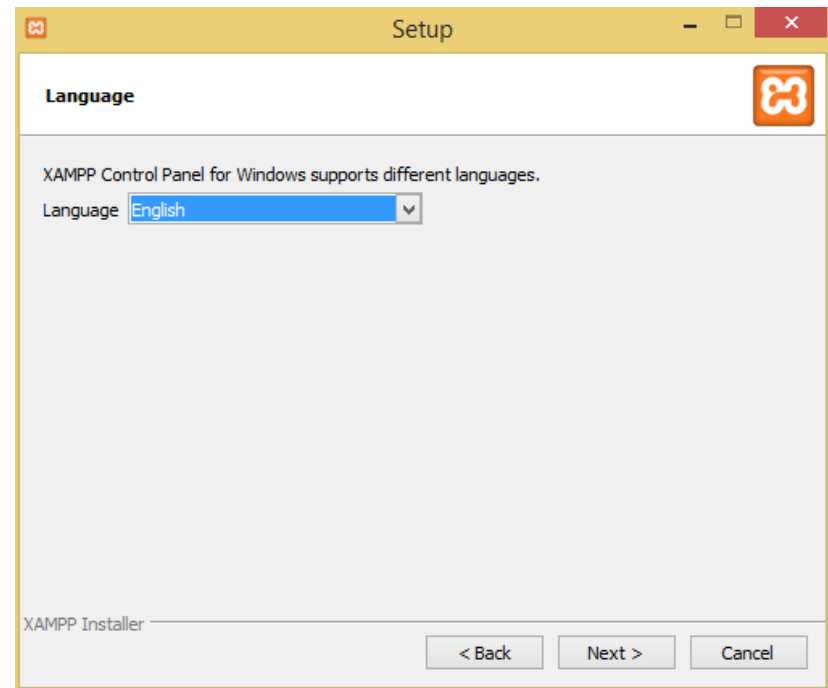
Activity : Installing PHP with XAMPP in Windows

- XAMPP installation step 4:
 - In this dialog you can customize the installation directory for XAMPP.
 - We will use the default path as **C:\xampp**.
 - Click **next** to continue.



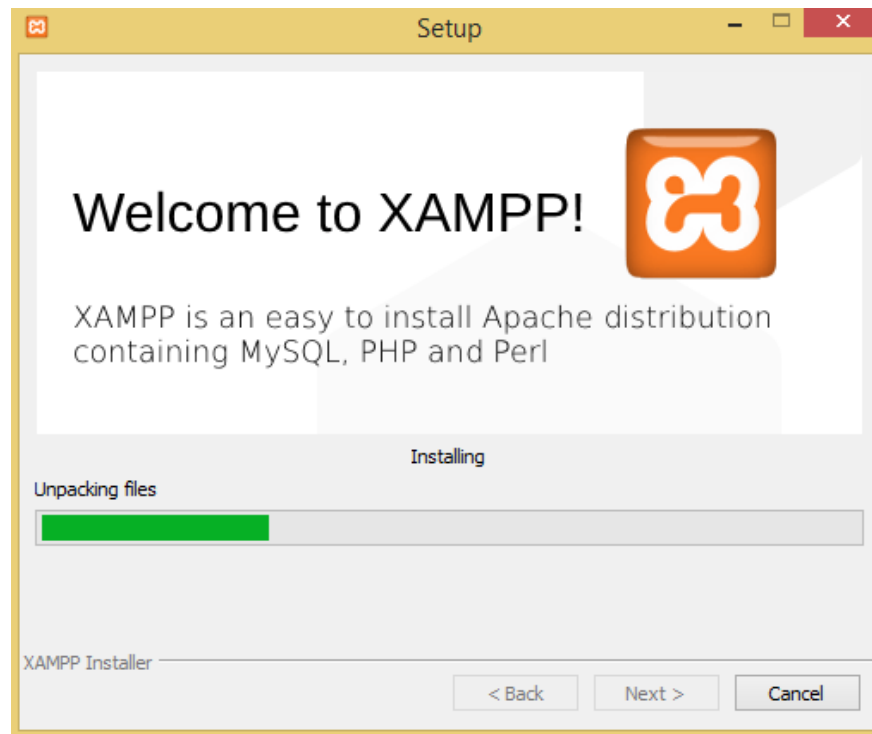
Activity : Installing PHP with XAMPP in Windows

- XAMPP installation step 5:
 - In this dialog, you can customize the language of XAMPP control panel.
 - We will use the default.
 - Click **next** to continue.



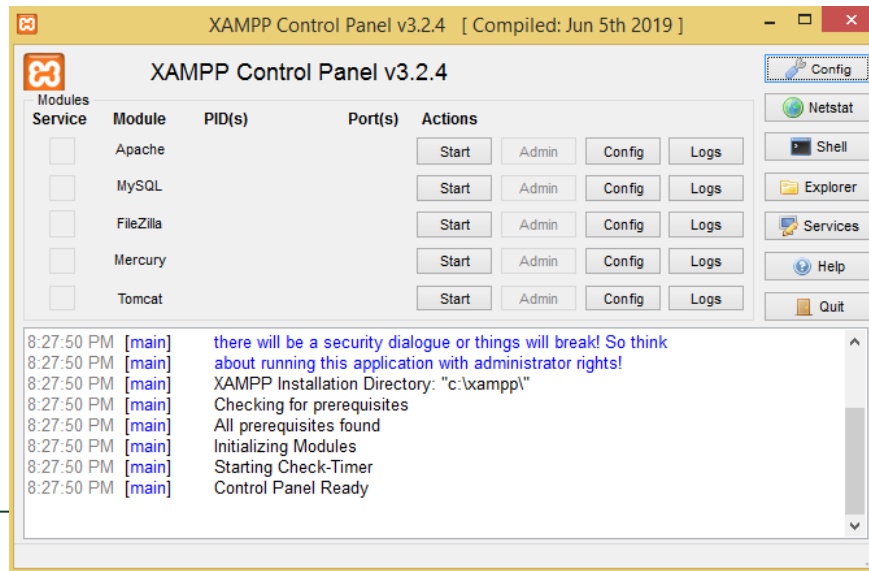
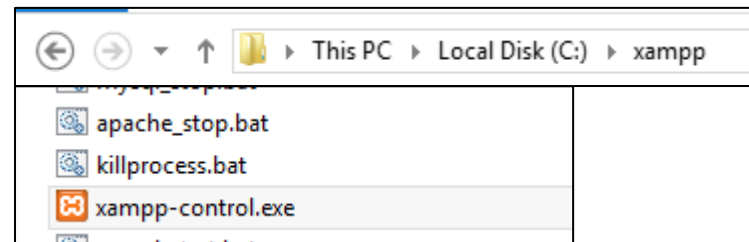
Activity : Installing PHP with XAMPP in Windows

- XAMPP installation step 6:
 - Click **next** until the installation begins and the following screen is shown.
 - Your system might need to be restarted.



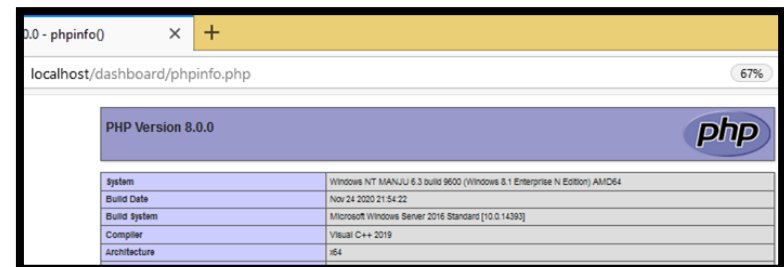
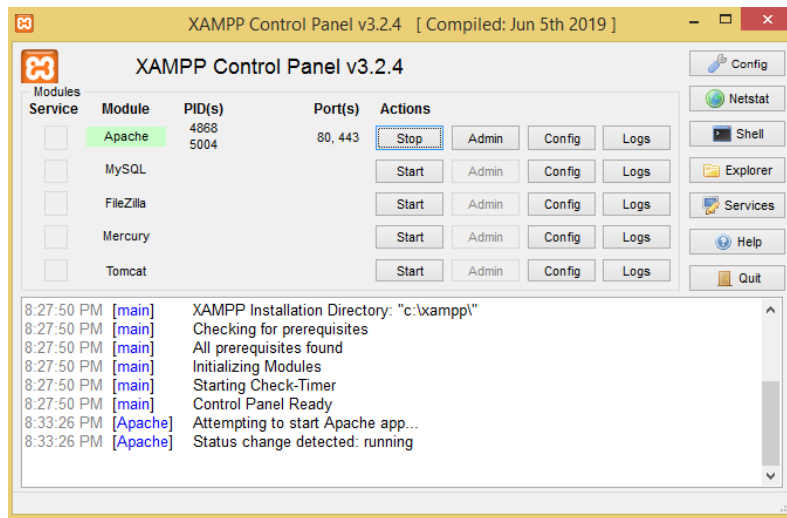
Activity : Installing PHP with XAMPP in Windows

- XAMPP installation should be completed by now.
 - Double click on xampp-control.exe in the installation directory.
 - The control panel shows here:



Activity : Installing PHP with XAMPP in Windows

- Testing XAMPP installation:
 - **Start** apache service from control panel as below:
 - Open the web browser and go to <http://localhost/>
 - Go to **phpinfo.php** page and see the output generated by PHP. <http://localhost/dashboard/phpinfo.php>
 - If the page is generated, then your PHP installation is complete!



Activity : Installing PHP with XAMPP in Windows

Problems?

- If you do not get the results as given above there are few things to check!
- Is there another server running in default port used by apache server (port:80) ?
- Restarting the computer if it is not already done from the installer.
- Go to FAQ section for XAMPP:
https://www.apachefriends.org/faq_windows.html
- Other places to look for help:
 - <https://community.bitnami.com/t/xampp-installation-problem/50826>
 - <https://stackoverflow.com/>

Explain the basic features of PHP

Explain basic features of PHP

- You can use any text editor to create PHP scripts.
- PHP scripts should be saved in files with the `.php` extension.
- PHP scripts should be coded within `<?php` and `?>` tags.
- In PHP scripting statement terminator is “`;`” symbol, as follows:

```
1
2 <?php
3
4 /* TODO write PHP code */
5
6 echo("Hello from PHP!");
7
8 ?>
```

Explain basic features of PHP

- You may have multiple statements as single lines, (i.e.: separated by ;) but having multiple statements in a single line is not a good coding practice.
- PHP is a free formatted language. Therefore, you can have spaces in between statement components as you wish. Compare valid PHP code in left picture with right picture as an example for Python which is NOT a free format.

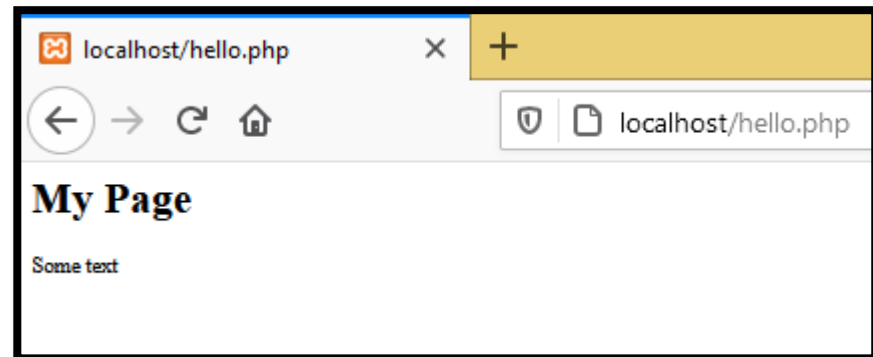
```
1 <?php
2 /* Legal code in PHP*/
3 echo "The\n";
4 echo "Quick\n";
5     echo "Brown\n";
6     echo "Fox\n";
7     echo "Jumped\n";
8     echo "Over\n";
9     echo "The\n";
10 echo "Lazy";
11 echo "Dog";
```

```
1
2 # illegal code in Python which is not free format
3 ▼ print("The\n")
4 ▼ print("Quick\n")
5 ▼ print("Brown\n")
6     print("Fox\n");
7     print("Jumped\n")
8     print("Over\n")
9     print("The\n")
10    print("Lazy\n")
11    print("Dog\n")
```

Activity : Write a PHP script and test

- We already tried the phpinfo page in the XAMPP installation.
- Here we create a PHP page on our own and try to access it.
- Let us create a php script in xampp **htdocs** directory.
- Write the following code in your favorite text editor and save it as **hello.php**
- We do not write any executable code here, we show embedding a **PHP** segment inside **HTML** markup as follows :

```
1 <h1>My Page</h1>
2 <?php
3
4 /* TODO: write code here */
5
6 ?>
7 <body>
8 Some text|
9 </body>
10
```



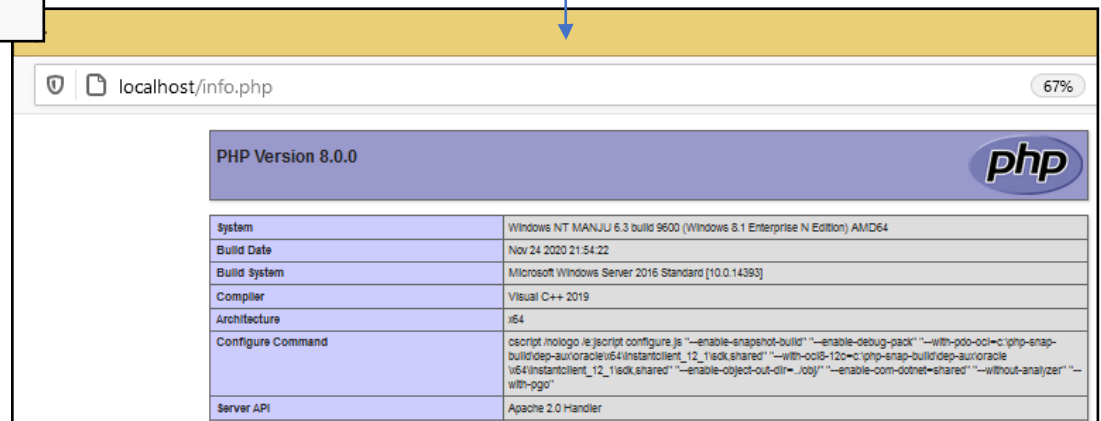
Activity : Write a PHP script and test

- The apache server connected to our PHP installation contains specific directory that it looks for an executable scripts.
- Here in XAMPP by default this directory is **<installation_path>/htocs**.
- This path can be seen in the configuration file of XAMPP **properties.ini** file with key:
apache_htdocs_directory=C:\xampp\htdocs.
- This document path is normally (non-XAMPP installations) under the configured in key : **DocumentRoot**
"C:/xampp/htdocs" in the server's
<server_root>/conf/httpd.conf file.

Activity : Write a PHP script and test

- Lets call **phpinfo** function from our code.
- Save the **info.php** in the same **htdocs** directory.
- Direct the browser to **info.php** and see the result.

```
1
2  <?php
3
4  /* TODO write PHP code */
5
6  ?>
7
8  <body>Hello HTML</body>
9  |
```



System	Windows NT MANJU 6.3 build 9600 (Windows 8.1 Enterprise N Edition) AMD64
Build Date	Nov 24 2020 21:54:22
Build System	Microsoft Windows Server 2016 Standard [10.0.14393]
Compiler	Visual C++ 2019
Architecture	x64
Configure Command	ceoscript mologio le jeoscript configure js --enable-snapshot-build --enable-debug-pack --with-pdo-oci=c:\php-snap-build-dep-aurosclev64\instantclient_12_1\sdk\shared --with-oci8-12c=c:\php-snap-build-dep-aurosclev64\instantclient_12_1\sdk\shared --enable-object-out-dir=.obj --enable-com-dotnet-shared --without-analyzer --with-pgo
Server API	Apache 2.0 Handler

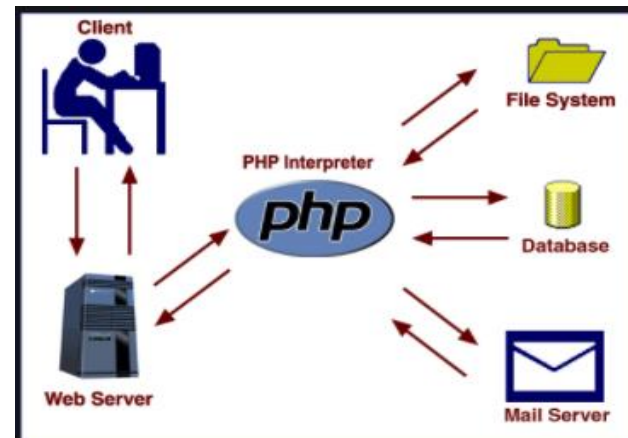
What is PHP?

- PHP is a general-purpose programming/scripting language especially used for web development.
- It was originally created by Danish-Canadian programmer Rasmus Lerdorf in 1994.
- The PHP reference implementation is now produced by The PHP Group. PHP originally stood for Personal Home Page, but it now stands for the recursive acronym “**PHP: Hypertext Preprocessor**”.
- PHP is an open software which can be used for commercial purposes without a license fee.

How does PHP work?

How PHP scripts are processed in web environment?

- Client sends a request to web server to access a PHP script
- The server checks if such a resource is available in the server
- If exists server send the script to the PHP interpreter together with the parameters given by the client (if any)
- PHP interpreter executes the instructions in the script
- Access any other resources if required (accessing file system, accessing database(s), accessing mail server(s) etc..)
- Interpreter sends the output of the script to the server
- Server sends it back to the client.



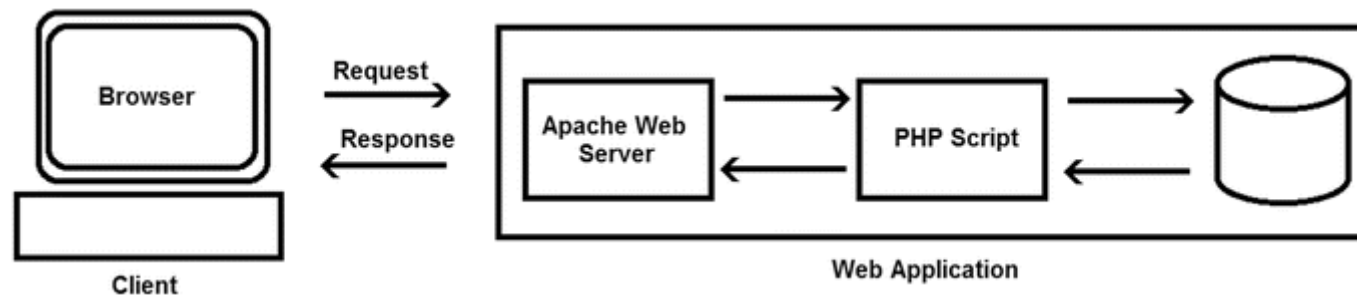
Difference between an interpreter and compiler

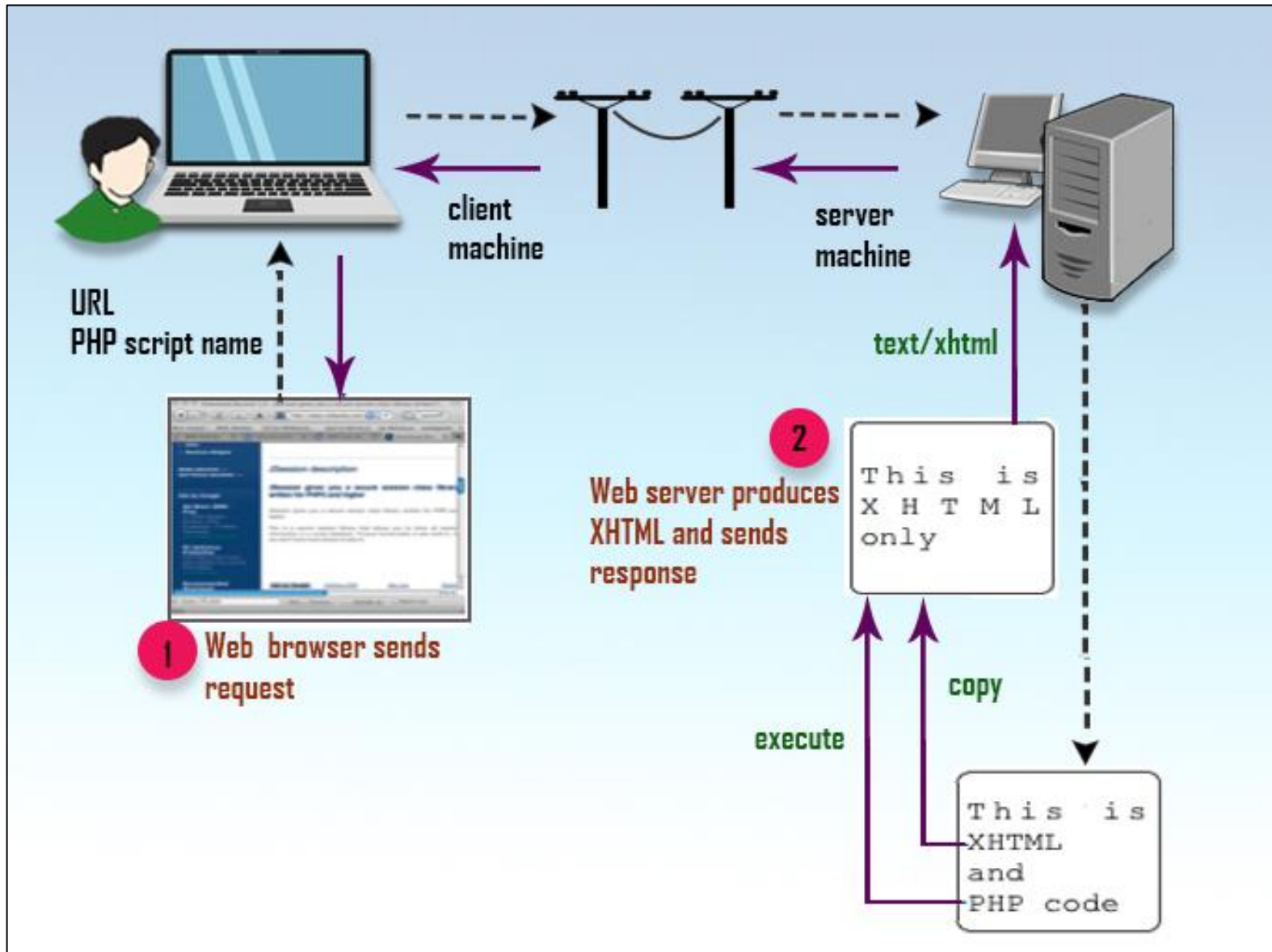
Interpreter takes the program source and executes the operations available in the script at the runtime.

Compiler translates the program source to either an intermediate form or to the machine(native) specific binary form.

The interpreters are usually slower than compiler based languages.

Some languages such as Java compile to an intermediate level between native format and source format which has to be interpreted at runtime but the gap is lesser than pure interpretation.





Explain basic features of PHP

- The **echo** command can be used to print text on the screen. The syntax of the echo command is given below

echo “some text”;

- Type the following script in a file by using a text editor and save it with the name **hello4php.php**

```
<?php
```

```
echo “Hello from PHP!”;
```

```
?>
```

- We can execute php scripts with or without the server
- Let’s execute the script at the command line by directly invoking the PHP interpreter

php hello4php.php

Explain basic features of PHP

- As we did before, the HTML pages can have embedded PHP code.
- The server will invoke the interpreter to process PHP code by switching between PHP and HTML modes.

```
<?php echo("Some PHP code"); ?>
```

```
<html>
```

```
<body>
```

```
<?php echo("Some PHP code"); ?>
```

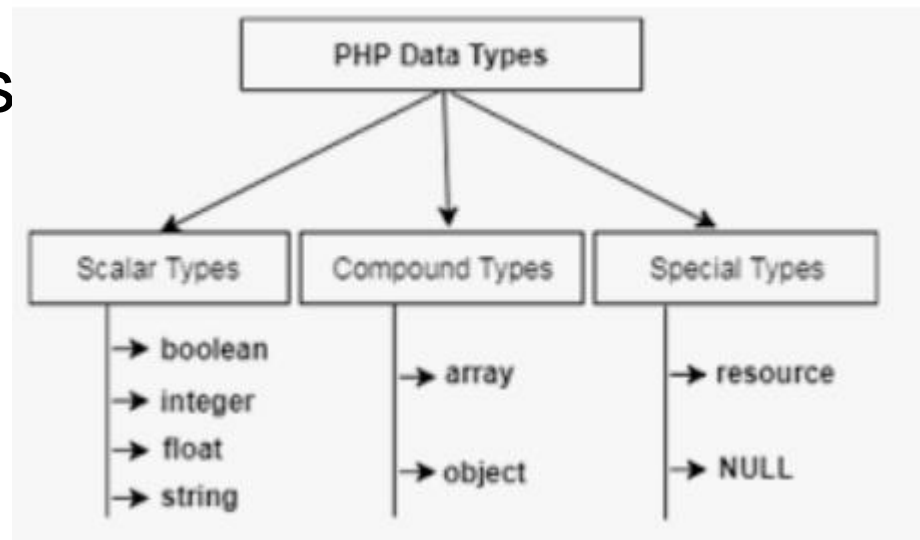
```
</body>
```

```
</html>
```

Data types and Constants

Data types

- A Data type can be described as a collection of values and a set of valid operations over these values.
- The different primitive data types provided by PHP can be classified as :
 - Scalar Types
 - Compound Types
 - Special Types



Data types

•PHP supports the following primitive scalar data types.

- Integer – e.g.: -5,-2,0,1,3,8...
- Floating point (or Double) – 1.25,3.71x10²
- String – “Hello”, “Welcome to PHP”
- Boolean – true/false

Data types

- Any number in the set {...,-2,-1,0,1,2,...} is considered as an integer. The maximum and the minimum integer value that can be used in a program are platform dependent.
- The number of bytes allocated to store an integer and the maximum integer value that can be used in your platform can be determined by using the constants `PHP_INT_SIZE` and `PHP_INT_MAX`
- If PHP encounters a number larger than `PHP_INT_MAX`, the number will be interpreted as a floating point number.
- Example :

```
<?php  
echo PHP_INT_SIZE, "\n", PHP_INT_MAX;  
?>
```

Data types

- If PHP encounters a number larger than PHP_INT_MAX, the number will be interpreted as a floating point number. Example :

```
<?php  
echo PHP_INT_SIZE, "\n", PHP_INT_MAX;  
?>
```

- An integer literal can be specified in decimal, octal, hexadecimal or binary.

hexadecimal : 0[xX][0-9a-fA-F] e.g.: 0x19af, 0x46B4E9

octal : 0[0-7] e.g.: 010537642

binary : 0b[01] e.g.: 0b101011001

Integer Data : Activity

Represent the following integers with PHP echo command and find the decimal representation by running the script. i.e:

<?php echo 1234; ?>

- 1234
- -123
- 0123
- 0x2b1
- 0b01101

```
// a positive integer in decimal form
// a negative integer in decimal form
// integer 83 in octal form
// integer 689 in hexadecimal form
// integer 13 in binary form
```

Data types

- The numbers with decimal points are considered as floating point (double or real) numbers.
- The floating point numbers are represented internally by using IEEE floating point representation. Thus, the representations are not exact. Therefore, floating point numbers should not compare for equality.
- Floating point literals can be coded in different ways. Example :
 - -1.23 1.2e3
 - 34.45E-12

Decimal Representation

Decimal Floating Point Representation

Exponent

$$12345 = 1.2345 \times 10^4$$

Significand

Base

$$0.1_{10} = 0.000110011001100110011001100_2$$

$$0.000110011001100110011001100 = 1.10011001100110011001100 \times 2^{-4}$$

$$\text{Exponent} = -4 + 127 = 123$$

$$\text{Sign bit} = 0$$

$$\text{Mantissa} = 10011001100110011001100$$

S (1 bit)	Exponent (8 bits)	Mantissa (23 bits)
0	01111011	10011001100110011001100



1 7 3 2 2 6 . 6 2

S

P

1 2 3 4 . 5

Data types

The following operators can be applied on both integers and floating point numbers.

Operator	Result
Unary -	Negation of the number
Binary -	Subtraction
+	Addition
*	Multiplication
/	Division
%	Remainder

Data types

PHP also supports increment and decrement operators as in Java and C++.

Operator	Description
++\$x	Increments \$x by one, then returns \$x
\$x++	Returns \$x, then increments \$x by one
--\$x	Decrements \$x by one, then returns \$x
\$x--	Returns \$x, then decrements \$x by one

Data types

PHP Logical operators

Operator	Operation
\$x and \$y	True if both \$x and \$y are true
\$x or \$y	True if either \$x or \$y is true
\$x xor \$y	True if either \$x or \$y is true, but not both
!\$x	True if \$x is not true

Activity : Data types

What is the output of the following program?

```
<?php  
echo -3, "\t", 5 - 3, "\t", 5.2*3.4,  
"\t", 10/2, "\t",  
10/4, "\t", 10%3, "\n";  
?>
```

Data types

- The following type conversion happens automatically when addition(+) operator is applied on its operands.
- If either operand is a float, then both operands are evaluated as floats, and the result will be a float.
- Otherwise, the operands will be interpreted as integers, and the result will also be an integer.

Data types

- A **string** is a series of characters.
- The series of characters in a string literal are normally represented either in single quotes or in double quotes.

Example :

```
<?php  
echo "This is a string literal","\n";  
echo 'Another string literal';  
?>
```

Data types

- Certain character sequences are given special meanings in PHP.
- Note that special character sequences are interpreted as one character despite the combination in them.

Character sequence	Special Meaning
\n	Linefeed
\t	Horizontal tab
\\$	Dollar sign
\"	Double quote

Activity : Data types

- The special character sequences retain their special meanings only when used inside double quotes. Execute the following code and check the result.

Example :

```
<?php
echo 'How the character sequence \n
works'; //not recognized
echo "A PHP string is represented by
\"String\" ";
echo `Bill spent 5$ for food` ;//not
recognized as variable
?>
```

Activity : Data types

- In the following example second statement is interpreted with a '\$bills' variable. Run the code and write a correction to the second statement.

```
<?php  
echo 'Bill spent 5 $bills for food' ;  
echo "Bill spent 5 $bills for food" ;  
?>
```

```
echo "Bill spent 5 \$bills for food" ;
```

Data types

- Boolean values are represented by using two literals – **TRUE** or **FALSE**. Both are case-insensitive.
- Values are automatically converted as appropriate if an operator, function or control structure requires a boolean argument.

Data types

- When converting to boolean the following values are considered as FALSE.
 - The boolean literal FALSE.
 - The integer literal 0
 - The floating point literal 0.0.
 - The empty string "" and the string "0".
 - An array with zero elements
 - The special type NULL.
- A boolean TRUE value is converted to the string "1" and FALSE is converted to "" (empty string).

Activity : Data types

- What is the output of the following script?

```
<?php
echo "True -",true,"\n";
echo "False -",false,"\n";
echo "1 > 0 - ", 1 > 0, "\n";
echo "2 > 5 - ", 2 > 5, "\n";
echo "0 && true -",0 && true,"\n";
echo '"" && true -','"" && true,"\n";
echo "TRUE > FALSE - ", TRUE > FALSE,
"\n";
echo "TRUE < FALSE - ", TRUE < FALSE,
"\n";
?>
```

Data types

- What is the output of the following script?

<?php	
echo "True -",true,"\n";	True -1
echo "False -",false,"\n";	False -
echo "1 > 0 - ", 1 > 0, "\n";	1 > 0 - 1
echo "2 > 5 - ", 2 > 5, "\n";	2 > 5 -
echo "0 && true -",0 && true,"\n";	0 && true -
echo "" && true -',' && true,"\n";	"" && true -
echo "TRUE > FALSE - ", TRUE > FALSE, "\n";	TRUE > FALSE - 1
echo "TRUE < FALSE - ", TRUE < FALSE, "\n";	TRUE < FALSE -
?>	

Data types

- String are converted to numbers when strings are used with mathematical operations. How this conversion happens is given below.
- The value is given by the initial portion of the string.
- The leading white spaces are ignored.
- If the string starts with valid numeric data, this will be the value used. Otherwise, the value will be 0 (zero).
- Valid numeric data is an optional sign, followed by one or more digits (optionally containing a decimal point), followed by an optional exponent. The exponent is an 'e' or 'E' followed by one or more digits.
- When a comparison operation involves a string or a numerical string, then each string is converted to a number and a numerical comparison is performed.

Activity : Data types

- What is the output of the following script?

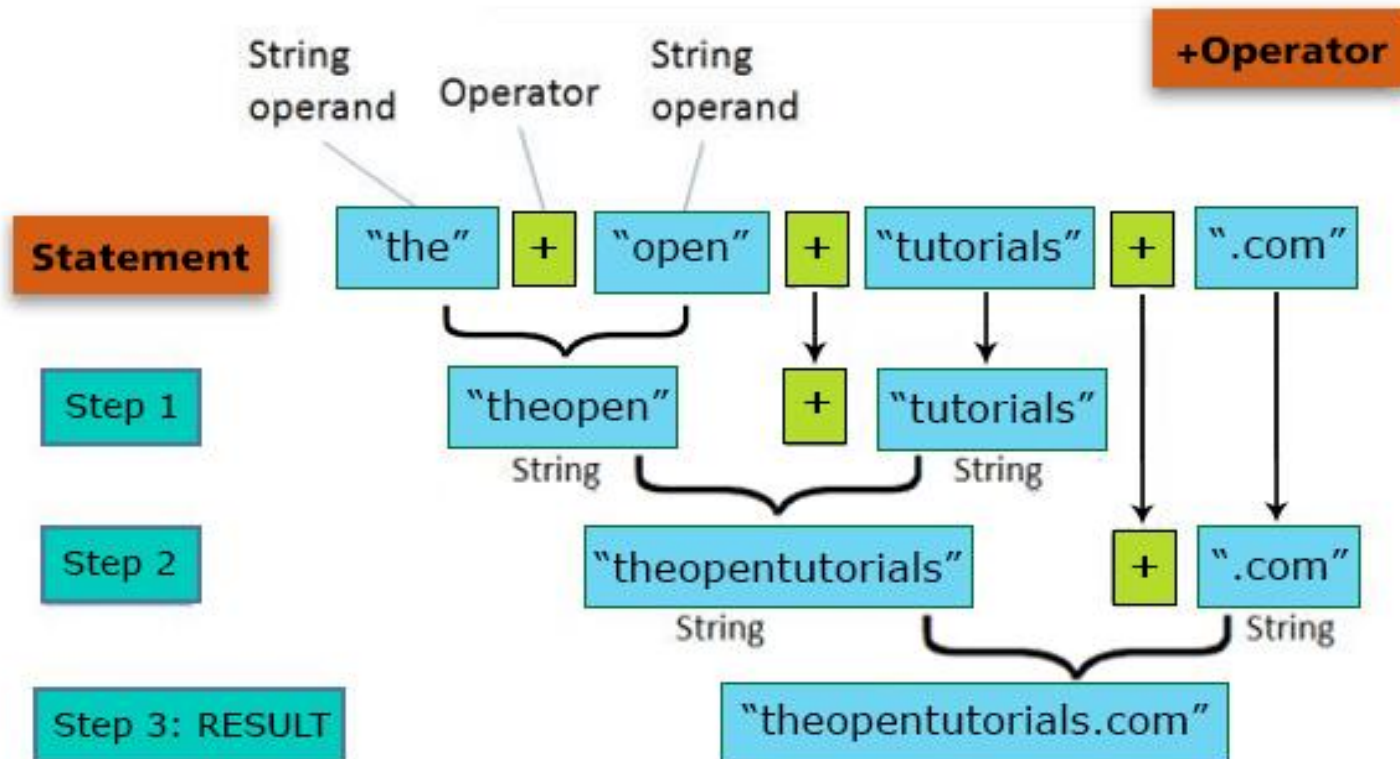
```
<?php
echo "10" > 1 , "\n";
echo "10 items" > 2 , "\n";
echo "items" > 2 , "\n";
echo True == 1, "\n";
echo false == 0, "\n";
echo TRUE == FALSE, "\n";
?>
```

Activity : Data types

- What is the output of the following script?

```
<?php
echo "10" > 1 , "\n";      1
echo "10 items" > 2 , "\n"; 1
echo "items" > 2 , "\n";    empty string
echo True == 1, "\n";      1
echo false == 0, "\n";     1
echo TRUE == FALSE, "\n"; empty string
?>
```

String Concatenation Operator



If both operands of + operator is a String, then the + operator becomes a String Concatenation

Here, 7 String objects are created in the memory.

1. "the", 2. "open", 3. "tutorials", 4. ".com",
5. "theopen",
6. "theopentutorials",
7. "theopentutorials.com"

print and echo, are they similar?

- There are some differences between echo and print:
- echo - can output one or more strings
- print - can only output one string, and returns always 1
- Tip: echo is marginally faster compared to print as echo does not return any value.

Constants

- Values declared in the code that do not change during the execution of the program.
- Constants are defined in PHP by using the `define()` function.
i.e.: `define("UCSC", "University of Colombo School of Computing")`
- `defined()` function says whether the constant exists or not.

Activity: Constants

- Define a constant and check if it is defined by the code:
- Example:

```
<?php
define("UCSC", "University of Colombo
School of Computing");
echo("defined?".defined("UCSC"));
//concatenate the value returned by
defined()
?>
```

Variables and Operators

placeholders for
unknown values

$$10 = 2x$$

variable

variables

usually denoted by
letters or symbols

x

A

Θ



21

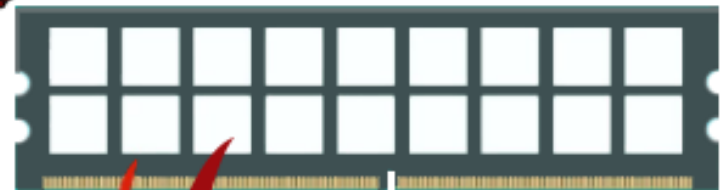
We can think that
variable is one type of
Container where we can
store some element

```
int age = 21;
```

int = which type of element
we can store

age = name of the container box

21 = type of element



```
int a = 5;
```

variable

value

Memory Cell/location



Address
Name

Variables

- A variable is a container of an object in a programming environment.
- Variables can represent memory locations of your computer.
- Since the data/value containing in a memory location can change the variable's value can change over the execution of a program.
- What type of object a variable can represent in your program depends on the data type of the given variable.
- i.e. : An integer type variable may represent a whole number stored in the computer memory and so on...

Variables

Rules for PHP variables:

- A variable starts with the \$ sign, followed by the name of the variable.
- A variable name must start with a letter or the underscore _ character.
- A variable name cannot start with a number.
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _).
- Variable names are case sensitive (\$y and \$Y are two different variables).

Variables

The variables in PHP are declared by appending the \$ sign to the variable name, i.e.:

```
$company = "UCSC";  
$sum = 10.0;
```

- Variable data type is set at runtime by the value that is assigned to the variable.
- Type casting allows to change the data type explicitly.
- Rich set of functions for working with variable.
i.e.: `gettype`, `settype`, `isset`, `unset`, `is_int`, `intval` etc...

Variable Scope

- PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
 1. local
 2. global
 3. static

Local Scope

- A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:
- \$x is a local variable

```
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

Global Scope

- A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:
- \$x is a global variable

```
<?php
$x = 5; // global scope

function myTest() {
    // using x inside this function will generate an error
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```

Global Scope

- However the **global** keyword can be used to access a global variable from within a function.
- To do this, use the global keyword before the variables (inside the function):

```
<?php
$x = 5;
$y = 10;

function myTest() {
    global $x, $y;
    $y = $x + $y;
}

myTest();
echo $y; // outputs 15
?>
```

Static Scope

- Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. (i.e.: we can use to count the function calls.)
- To do this, use the static keyword when you first declare the variable:

```
<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}

myTest();
myTest();
myTest();
?>
```

Operators

- Most PHP operators such as arithmetic, assignment, comparison and logical are similar to the operators in Java or C++.
- However, in PHP the string concatenation operator is denoted by '.' but not '+'.
i.e.

```
$whats_ur_name = "My name is ".$myname;
```

Arithmetic Operators

- Summary of basic mathematical operators in PHP

Operator	Usage	Meaning
+	$\$x + \y	$\$x$ added to $\$y$
-	$\$x - \y	$\$y$ subtracted by $\$x$
*	$\$x * \y	$\$x$ multiplied by $\$y$
/	$\$x / \y	$\$x$ divided by $\$y$
%	$\$x \% \y	Remainder of $\$x$ divided by $\$y$
**	$\$x ** \y	$\$x$ to the power of $\$y$

Boolean (Logical) Operators

- The following operators can be applied on both integers and floating point numbers.

Operator	Result
and, &&	TRUE when both operands are TRUE
or,	TRUE when either operand is TRUE
xor	TRUE when either operand is TRUE, but not both
!	negation

Activity: Operators

- Find the output of the following code fragment.

```
<?php
```

```
$x = 3;
```

```
$y = 5;
```

```
$z = 4;
```

```
echo '$x + $y:' . ($x + $y) . "\n";
```

```
echo '$z - $x:' . ($z - $x) . "\n";
```

```
echo '$y * $z:' . ($y * $z) . "\n";
```

```
echo '$x / $y:' . ($x / $y) . "\n";
```

```
echo '$z % $x:' . ($z % $x) . "\n";
```

```
echo '$y ** $z:' . ($y ** $z) . "\n";
```

```
?>
```


Activity: Operators

- Find the output of the following code fragment.

```
<?php
```

```
$x = 3;
```

```
$y = 5;
```

```
$z = 4;
```

```
echo '$x + $y:' . ($x + $y) . "\n";
```

\$x + \$y:8

```
echo '$z - $x:' . ($z - $x) . "\n";
```

\$z - \$x:1

```
echo '$y * $z:' . ($y * $z) . "\n";
```

\$y * \$z:20

```
echo '$x / $y:' . ($x / $y) . "\n";
```

\$x / \$y:0.6

```
echo '$z % $x:' . ($z % $x) . "\n";
```

\$z % \$x:1

```
echo '$y ** $z:' . ($y ** $z) . "\n";
```

\$y ** \$z:625

```
?>
```

Conditional Statements

Conditional Statements

Conditional statements are for decision making at branching points of a program.

In PHP we have the following conditional statements:

- **if statement** - executes some code only if a specified condition is true
- **if...else statement** - executes a block of code if a condition is *true* and some other block of code if the same condition is *false*
- **if...elseif...else statement** - selects only one of several blocks of code to be executed
- **switch statement** - selects one of several blocks of code to be executed

Syntax of Conditional Statements

if statement & if...else statement

IF statement

```
if (<condition>) {  
    //php code goes here  
}
```

IF Else statement

```
if (<condition>) {  
    //php code goes here  
}  
else {  
    //php code goes here  
}
```

```
<?php
```

```
    $color="Red";  
  
    if ($color == "Red") {  
        echo "Please STOP";  
    } else {  
        echo "You can GO";  
    }  
}
```

```
?>
```

Conditional Statements

Syntax

if...elseif....else statement

```
if (condition) {  
    //php code goes here  
} elseif (condition) {  
    //php code goes here  
} else {  
    //php code goes here  
}
```

```
<?php  
  
$color="Red";  
  
if ($color == "Red") {  
    echo "Please Stop" ;  
} elseif ($color == "Yellow")  
    { echo "Get ready" ;  
} else {  
    echo "You can GO" ;  
}  
  
?>
```

Conditional Statements

Syntax

Switch statement

select one of many blocks of code to be executed.

```
<?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, or  
green!";  
}
```

```
?>
```

Activity : Conditional statements

- Write a conditional statement to echo a string for a number given in the variable \$input as “red”, “green”, “blue” and “yellow”.
- If the input is positive and even : red.
- If the input is positive and odd : blue.
- If the input is negative : green.
- If the input is zero yellow.

Loops

Loops

- Loops are used when you need some block of code to be executed over and over again.
- In PHP, we have the following looping constructs:
 - **while** - loops through as long as the given condition is true
 - **do...while** - loops through the code at least once, and then repeats the loop as long as the given condition is true
 - **for** - loops through the code a given number of times
 - **foreach** - loops through the code for each element in a collection

while Loop

```
while (condition is true) {  
    //Code block;  
}
```

```
<?php  
$i=1;  
while($i<=5) {  
    echo "Number: $i </br>";  
    $i++;  
}  
?>
```

```
Number: 1  
Number : 2  
Number : 3  
Number : 4  
Number : 5
```

do-while loop

```
do {  
    //Php code  
} while (condition is true);
```

```
Number: 1  
Number : 2  
Number : 3  
Number : 4  
Number : 5
```

```
<?php  
  
$i=1;  
do {  
    echo "Number: $i </br>";  
    $i++;  
}while ($i<=5 && $i>1);  
  
?>
```

for loop

```
for (initialize counter; check; increment counter) {  
    //Do this;  
}
```

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

```
The number is: 0  
The number is: 1  
The number is: 2  
The number is: 3  
The number is: 4  
The number is: 5  
The number is: 6  
The number is: 7  
The number is: 8  
The number is: 9  
The number is: 10
```

foreach loop

This works only on collections such as arrays
,lists

```
foreach ($array as $value)
{
    //Do this
}
```

```
Nimal
Kamal
Sunil
Amal
```

```
<?php

$person =
array("Nimal","Kamal","Sunil","Amal");

foreach ($person as $value) {
    echo "$value \n";
}

?>
```

Activity : Loops

- Consider the following PHP statement:

```
$person = array("Dj","Kamal","de","Lanerole");
```

- Write a foreach loop to iterate through the \$person array and inside the loop there should be **switch** statement that categorizes the array elements based on the length and **echos** the “short\n” when the name is 0,1 or 2 characters “medium” when 3,4 or 5 characters and “long” otherwise.
- (Note: You can combine conditions with grouped cases and find the length of the name by **strlen()** function)

Activity : Loops

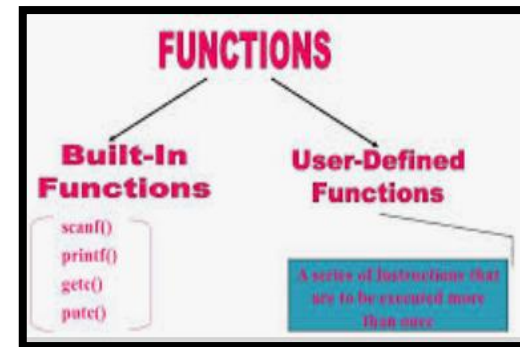
Answer:

```
1  <?php
2  $person = array("Dj","Kamal","de","Lanerole");
3  ▼ foreach ($person as $value) {
4      $len = strlen($value);
5  ▼      switch($len){
6          case 0:
7          case 1:
8          case 2:
9              echo "short\n";
10             break;
11         case 3:
12         case 4:
13         case 5:
14             echo "medium\n";
15             break;
16         default:
17             echo "long\n";
18     }
19 }
20 ?>
```

Functions

Functions

- Functions make your code easy to read and make reusable.
- Large projects would be unmanageable without functions because the problem of repetitive code that would bog down the development process
- A function accepts values, processes them, and then performs an action (printing to the browser, for example) and optionally returns a new value.
- PHP has 2 types of functions
 1. Language defined functions
 2. User defined functions



Built-in functions

- PHP has hundreds of language defined(built-in) functions . For example strlen() returns the length of a string, in characters .

```
<?php  
echo strlen("Hello World!");  
?>
```

```
C:\xampp\htdocs>  
C:\xampp\htdocs>php strlen_fun.php  
12  
C:\xampp\htdocs>
```

- You can find a complete set of language defined functions [here](#)

User defined functions

- You can define your own functions in PHP using the **function** keyword.

```
<?php
function writeMsg() {
    echo "Hello world!";
}

writeMsg(); // call the function
?>
```

- A function name can start with a letter or underscore but not a number.

**variables names are case sensitive in
PHP, function names are not!**

User defined functions-Function with no parameters

```
function NameOfFunction() {  
    statements;  
}
```

Defining
the
Function



```
<?php  
function WriteWhoAmI() {  
    echo "I'm an undergraduate";  
}  
?>
```

Calling
the
Function



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

Now when you call this function, it will print **I'm an undergraduate**

User defined functions – with parameters

- Parameter types and return types are not written .
- A function with no return statements implicitly returns NULL.

```
function name(parameterName, ...,  
parameterName) {  
    statements;  
}
```

```
<?php  
function multiply($a, $b, $c) {  
    return $a*$b*$c;  
}  
?>
```

```
<?php  
echo multiply(1,2,3);  
?>
```

Default Parameter Values

When a function is called without specifying a parameter value, the default value will be used

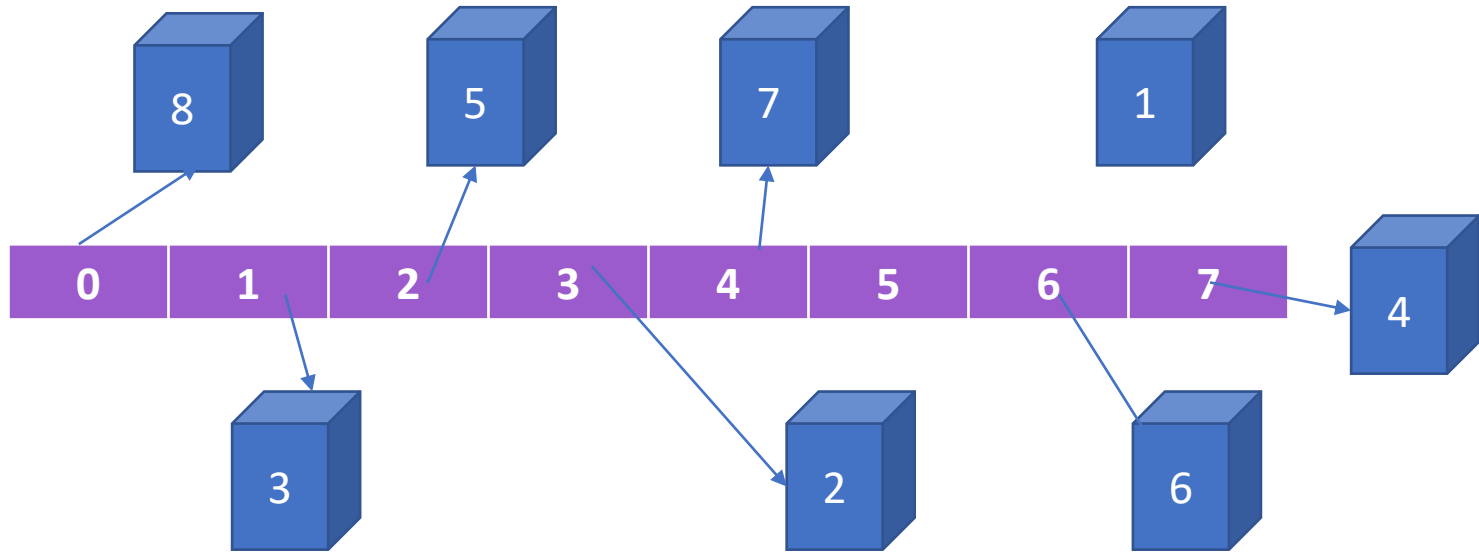
```
<?php
function setMarks($minMark=50) {
    echo "The Mark is : $minMark </br>";
}

setMarks(95);
setMarks(); // will use the default value of 50
setMarks(80);
?>
```

Activity: Functions

Complete the factorial function skeleton given below that computes the factorial for a positive integer (factorial 5 = 5x4x3x2x1):

```
1  <?php
2  ▼ function factorial($n) {
3      $f = 1;
4  ▼  while($n > 1){
5
6      }
7      return $f;
8  }
9
10 echo "factorial:",0,"=",factorial(0),"\n";//1
11 echo "factorial:",1,"=",factorial(1),"\n";//1
12 echo "factorial:",5,"=",factorial(5),"\n";//120
13 ?>
```



Arrays

An array stores multiple values in one single variable

PHP Arrays

- One of the compound data types provided by PHP is arrays.
- In general a PHP array is an ordered collection of data items where each item in the collection is associated with a key.
- 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

PHP Arrays

- PHP 'indexed array' with three String data elements.

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

- To access an indexed array you have to use the index number of the elements starting from 0 to length-1.
- To get **the length of an array** - The **count()** function is used (the number of elements) of an array.

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
```

Construction of an array

- An **associative array** is constructed by using the language construct *array(array_elements)*
- The *array_elements* comprises of a comma-separated *index,value* pairs, where each pair is represented as *index => value*.

Syntax :

```
array(  
    index_1 =>value_1,  
    index_2 => value_2,  
    .....  
    index_n => value_n,
```

The comma after the last array element is optional and can be omitted

Construction of an array...

- The *index* of an element can be of type *integer or string*.
- When the index is omitted, an integer index is automatically generated, starting at highest integer index + 1 or 0 if no item is given an integer index.

example:

```
$a = array(
    1=> "First Item", "item2"
    => "Second
    Item",
    5 => "Third item", "Forth
    item"
);
```

PHP would automatically generate the index 6 for the last item in the array.

Accessing an element in an array.

- An element of an array can be accessed by using the syntax

`array_variable[index]`

example:

```
$a = array(1=> "First  
Item", "item2" => "Second  
Item", 5 => "Third item",  
"Forth item"  
);
```

Each element in the above array can be accessed as below;

```
$a[1]  
$a["item2"]  
$a[5]  
$a[6]
```

Changing the value of an array element.

- The following syntax can be used to change the value of an array element.

```
$array_variable[index]  
= new_value;
```

example:

```
$a = array(  
    1=> "First Item",  
    "item2" => "Second  
Item",  
    5 => "Third item",  
    "Forth item"  
);
```

```
$a[1] = "abc";  
$a["item2"] = 25;
```

Adding a new element to an array.

- The following syntax can be used to add a new value to an array.

```
$array_variable[new_index] =  
    new_value;
```

The *new_index* should not exist as an index in the array.

Appending elements to the end of an array.

array_push command can be used to add one or more elements to the end of the array.

Syntax :

```
array_push(array_variable, value1,value2,.....)
```

Example :

```
$a = array("Nimal","Saman");  
array_push($a,"Kamal","Waruna");
```


Array of arrays

- Elements of an array can also be arrays.

example :

```
$a = array(  
    "males" => array("a" => "Nimal", "b" =>  
        "Amara", "c"  
=>"Kamal"),  
    "females" => array("a" => "Kumari", "b" =>  
        "Nirmala", "c" =>  
"Kamala"),  
    "fees" => array (2500,1500,500)  
);
```

Accessing array elements example :

```
echo      $a["females"]["b"]
```

Looping through array elements

foreach looping construct can be used to loop through the elements of an array.

Syntax :

```
foreach (array_expression as $value)
```

statement Or

```
foreach (array_expression as $key =>  
$value) statement
```

Looping through array elements - Example

```
<?php
$a = array(
    1=> "First Item",
    "item2" => "Second Item",    5 => "Third
    item",
    "Forth item"
);

foreach ($a as $key => $value){    echo
    $key, " - ", $value, "\n";
}
?>
```

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.
- The dimension of an array indicates the number of indices you need to select a single element. i.e.:
- 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 and so on...

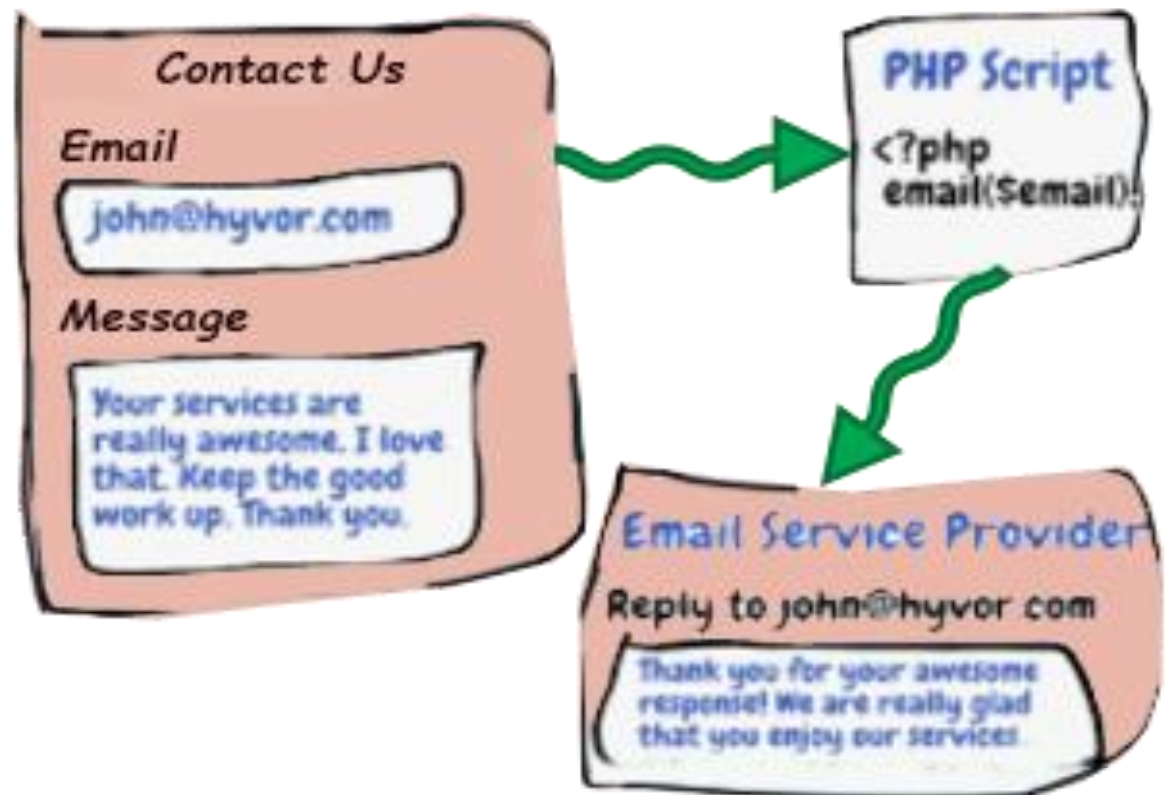
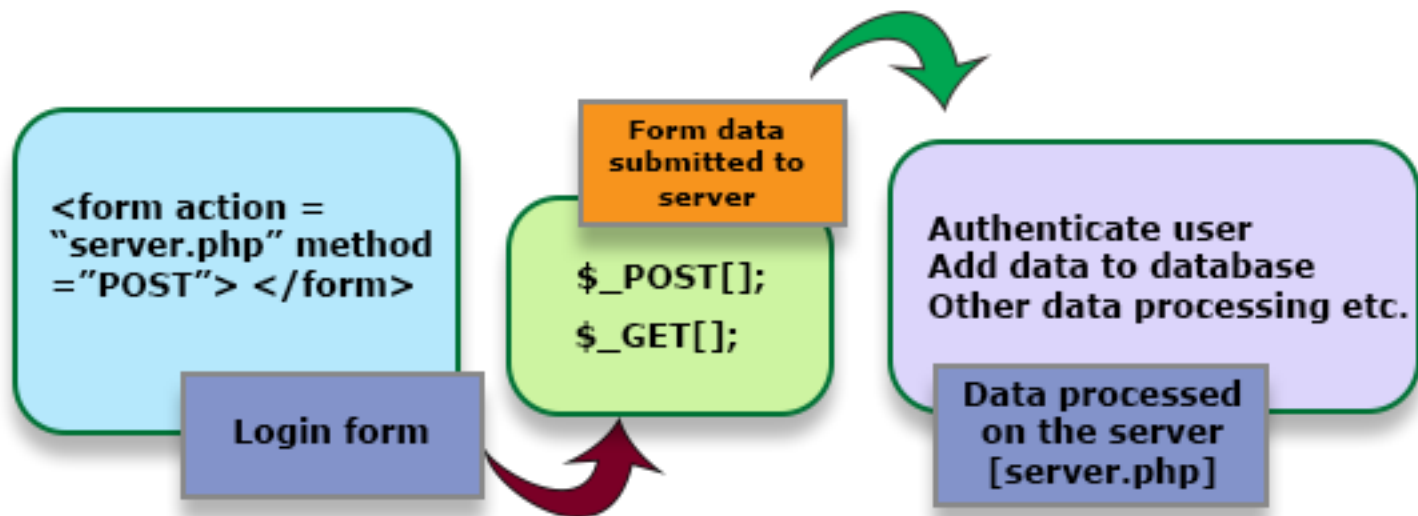
Multidimensional arrays

- Defining a two dimensional array: `$cars = array (array("Volvo",22,18), array("BMW",15,13), array("Saab",5,2), array("Land Rover",17,15));`
- Accessing the two dimensional array element by element:

```
<?php
for ($row = 0; $row < 4; $row++) {
    echo "<p><b>Row number $row</b></p>";
    echo "<ul>";
    for ($col = 0; $col < 3; $col++) {
        echo "<li>".$cars[$row][$col]."</li>";
    }
    echo "</ul>";
}
?>
```

Handling Form Data

What is a form & how
they work in PHP?



Forms in HTML

- HTML forms is a powerful feature that enables an application on a web server to interact with users.
- HTML forms allow users to send data to the web site.
- An HTML Form is typically made of a collection of widgets such as text fields, buttons, checkboxes, radio buttons or select boxes.
- The “post” or “get” HTTP methods can be used to send data to the server.

Action and method attributes

- Two important attributes of the “form” element are “action” and “method”.
 - action : specifies the URL of the web resource designated to receive the data collected from the form element.
 - method : specifies which HTTP method (get or post) should be used to send data to the receiving URL.
 - If the receiving URL is a PHP program, then depending on the **method** used in the HTML form either the PHP superglobal `$_GET` or `$_POST` can be used to access form data at the servers end.

Example 1- Form with text inputs

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

The **name** attribute specifies the key value of the `$_POST` global array element from which the value of this input item can be retrieved

The content of the example.php script is given below

```
<!DOCTYPE html>
<html>
<body>
<div> Hello <?php echo
$_POST["name"]?></div>
</body>
</html>
```

Example 2- Form with check boxes

```
<html>
<body>
<form action="example.php"
method="post"> Do you have an email?
<input type="checkbox" name="emailOption" value
= "Yes"><br>
<input type="submit"></form>
</body>
</html>
```

When the user checked the checkbox, the value "Yes" is send to the server as the value of the attribute "emailOption".

Example 2- check boxes

```
<html>
<body>
<div>
<?php
if($_POST["emailOption"]== "Yes"){
echo "Option is checked";
} else {
echo "Option is not-checked";
}
?>
</div>
</body>
</html>
```

The data send to the server can be accessed by using the `$_POST` global array element with the key value "emailOption"

Example 3- Form with a check box group

```
<html>
<body>
<form action="example.php"
method="post"> Which fruits do you like?
<input type="checkbox" name="fruits[]" value = "Apples">Apples<br>
<input type="checkbox" name="fruits[]" value =
"Oranges">Oranges<br>
<input type="checkbox" name="fruits[]" value = "Grapes">Grapes<br>
<input type="submit">

</form>
</body>
</html>
```

Note that the checkboxes have the same name "fruits" and each name ends in [].

- The same name indicates that these checkboxes are all related and forms a group.
- [] indicates that the selected values will be provided to PHP script as an array. This means That the `$_POST['fruits']` is an array not a single string.

Example 3- Form with a check **box** **group**

```
<html>
<body>
<div><?php
    $fruits = $_POST["fruits"];
    if(!empty($fruits)){
        echo "You like ";
        for($i=0; $i <
            count($fruits);$i++){ echo
                "<br>". $fruits[$i];
        }
    } else {
        echo "You do not like any fruits";
    }
?>
</div>
</body>
</html>
```

Example 4- Form with a **Radio** **button**

```
<html>
  <body>
    <form action="example.php"
      method="post"> Please specify your sex
      :<br>
      <input type="radio" name="sex" value =
"male">male<br>
      <input type="radio" name="sex" value
= "female">female<br>
      <input type="submit">
    </form>
  </body>
</html>
```

Note that all radio buttons should have the same value for the attribute "name".

Example 4- Form with a **Radio button**

```
<html>
  <body>
    <div>
      <?php
        echo "you are a ".
          $_POST["sex"];
      ?>
    </div>
  </body>
</html>
```

```
GET /spec.html HTTP/1.1  
Host: www.example.org  
Cookie: theme=light; sessionToken=abc123  
....
```

Cookies

Cookies

- A cookie is a file with small amount of data that a website embeds on the user's computer through a web browser. This cookie is send back to the website by a browser every time when the user is accessing the same website by using the same browser.
- The browsers can either enable or disable cookies.
- In PHP data stored in cookies can be accessed by using the global array `$_COOKIE`

Cookies

Web browser

Web server

1. The browser requests a web page

2. The server sends the page and the cookie

The cookie

Hello World!

3. The browser requests another page from the same server

The cookie

What Is A Cookie?

- Apart from being a type of biscuit, a cookie is also a very useful piece of technology for use on the web.
- One of the problems which many websites need to overcome is that there is no way of directly finding out who is on a website.

PHP 5 Cookies

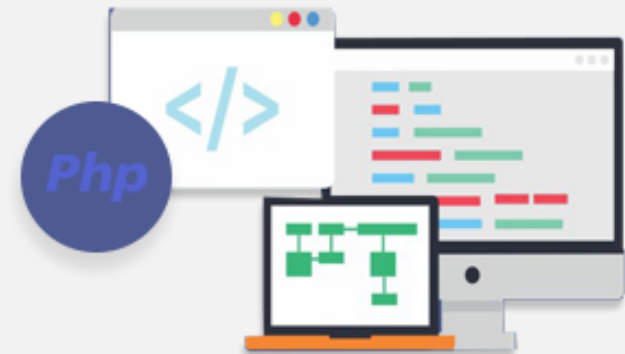
What is a Cookie?

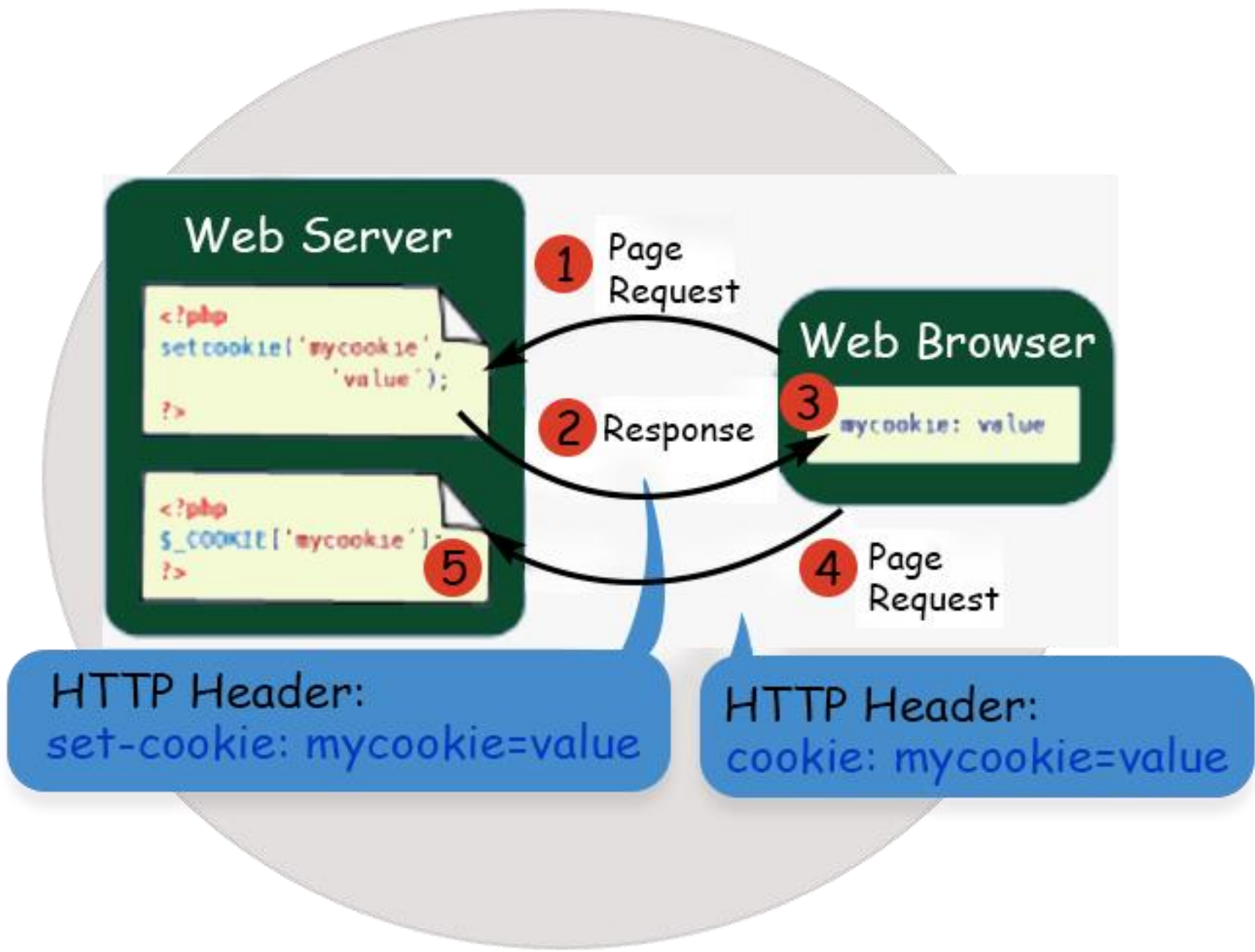
A cookie is often used to **identify a user**. A cookie is a small file that the server embeds on the **user's computer**. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

Syntax

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





Setting a cookie – setcookie()

Syntax:

- `setcookie($name [, $value [, $expire]])`;

Semantic:

- Sends a cookie to the browser.
- \$name - The name of the cookie
- \$value – The value of the cookie
- \$expire – The time (in seconds) the cookie expires. If this value is set to 0 or omitted, the cookie will expire when the browser closes.
- The function returns the Boolean value TRUE on success or FALSE on failure.

Cookies must be sent before producing any output in a script. This requires

`setcookie()` function to be used prior to any output, including `<html>` and `<head>` tags as well as any whitespace.

Checking whether cookies are enabled or not

```
<?php
setcookie("name","saman",time()+3600);
if(count($_COOKIE) > 0){
    echo "Cookies are enabled<br>";
} else {
    echo "Cookies are disabled <br>";
}
```

Modifying the value of a cookie

- To modify the value of a cookie call the same function

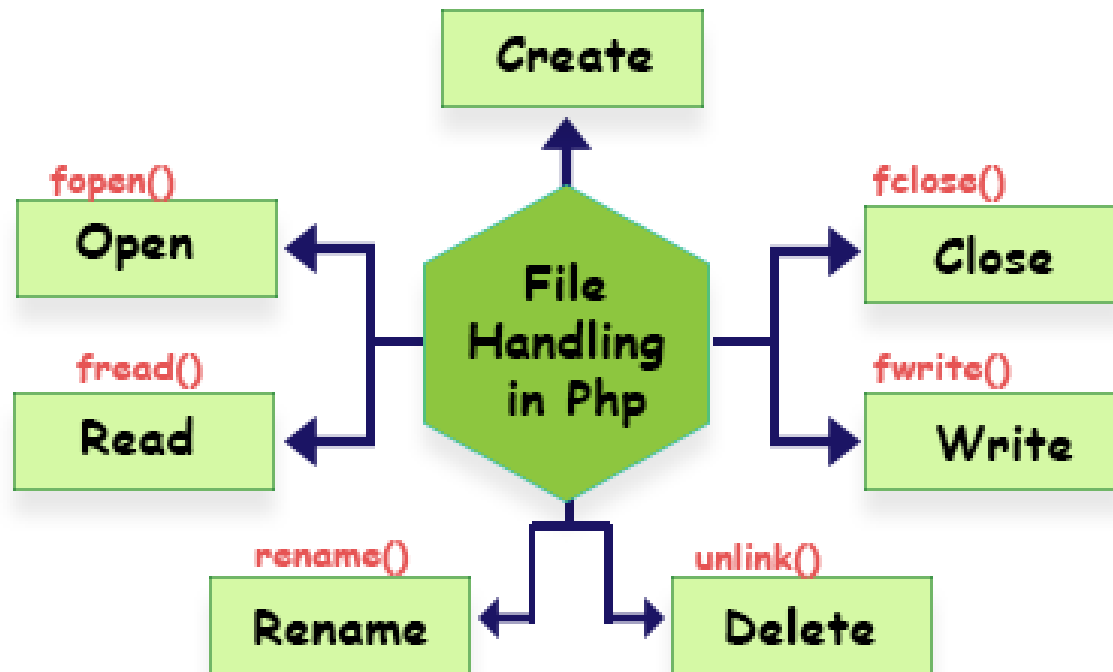
`setcookie()` with the new value.

```
<?php  
setcookie("name","Kamal",time()  
+3600);  
?>
```


Deleting a cookie.

- To delete a cookie execute the same `setcookie()` function with an expiration date in the past.

```
<?php  
setcookie("name","Kamal",tim  
e()-3600);  
?>
```



File Handling

w	→	writing(writing to a file)
r	→	reading(reading to a file)
a	→	append(adding to a file)

Typical operations on files

- Opening a file
- Adding data
- Accessing data
- Closing the file

Opening a File

Syntax:

```
fopen ( $filename , $mode  
[, $use_include_path = false  
[,  
$context ] ] )
```

fopen() binds the resource named as

\$filename, to a stream.

- If a file named “mydata.txt” exists then the content of the file is deleted.
- If there is no file with the name “mydata.txt” then a new file with the name “mydata.txt” is created.

File opening modes

w – write

r – reading

a – appending

fopen returns a file pointer resource on success or FALSE on failure

Writing data to a file

```
<?php
$f =
fopen("data.txt","w");
fwrite($f,"My name is
saman\n");
fwrite($f,"My age is
90"); fclose($f);
?>
```

fwrite returns the number of bytes written to the file or **FALSE** on failure.

Syntax of fwrite :

`fwrite ($handle , $string [, $length])`

`fwrite()` writes the content of **\$string** to the file stream pointed to by **\$handle**. If the optional length argument is given, writing will stop after **\$length** number of bytes is written or the end of string is reached, whichever comes first.

Appending data to a file

```
<?php  
$f = fopen("data.txt","a");  
fwrite($f,"My name is  
Sunil\n"); fclose($f);  
?>
```

Reading data from a file – fgets()

Syntax:

`file ($filename)`

Semantics:

- Reads the entire file
\$filename into an array.
- The command returns
 - The file in an array. Each element of the array corresponds to a line in the file or
 - FALSE if an error occurs.

```
<?php
$lines= file("data.txt");
foreach($lines as
$line_no => $line){
    echo
    $line_no,$line,"<br>";
}
?>
```

Reading data from a file – file()

Syntax:

```
fgets ( $handle [,$length ] )
```

Semantics:

- Reads a line from the file pointed to by the file pointer \$handle.
- The command returns
 - A line of symbols (including the end of line marker) from the file as a string when the \$length parameter is not specified or
 - A string of up to length - 1 bytes from the file when \$length parameter is specified or
 - The Boolean value FALSE when there is no more data to read in the file or
 - The Boolean value FALSE if an error occurred while reading the file.

Reading data from a file – fscanf()

Syntax:

fscanf(\$handle, \$format)

Semantics:

- Reads a line of the file pointed to by the file pointer \$handle according to the format specified by the string \$format.
- The command returns
 - the values parsed as an array.

```
<?php
$f = fopen("data.txt","r");
while ($line =
fscanf($f,"%s\t%d\n")){
    echo $line[0],"-
",$line[1],"<br>";
}
?>
```

Reading data from a file - Example

```
<?php
$f =
fopen("data.txt","r")
;
while (! feof($f)){
    $line = fgets($f);
    echo $line, "<br>";
}
fclose($f);
?>
```

Existence of a file/directory

Command : `file_exists()`

Syntax :

`file_exists($filename)`

Semantics :

Checks the existence of a file or directory.

It returns the Boolean value TRUE when the file/Directory exists, otherwise it returns FALSE.

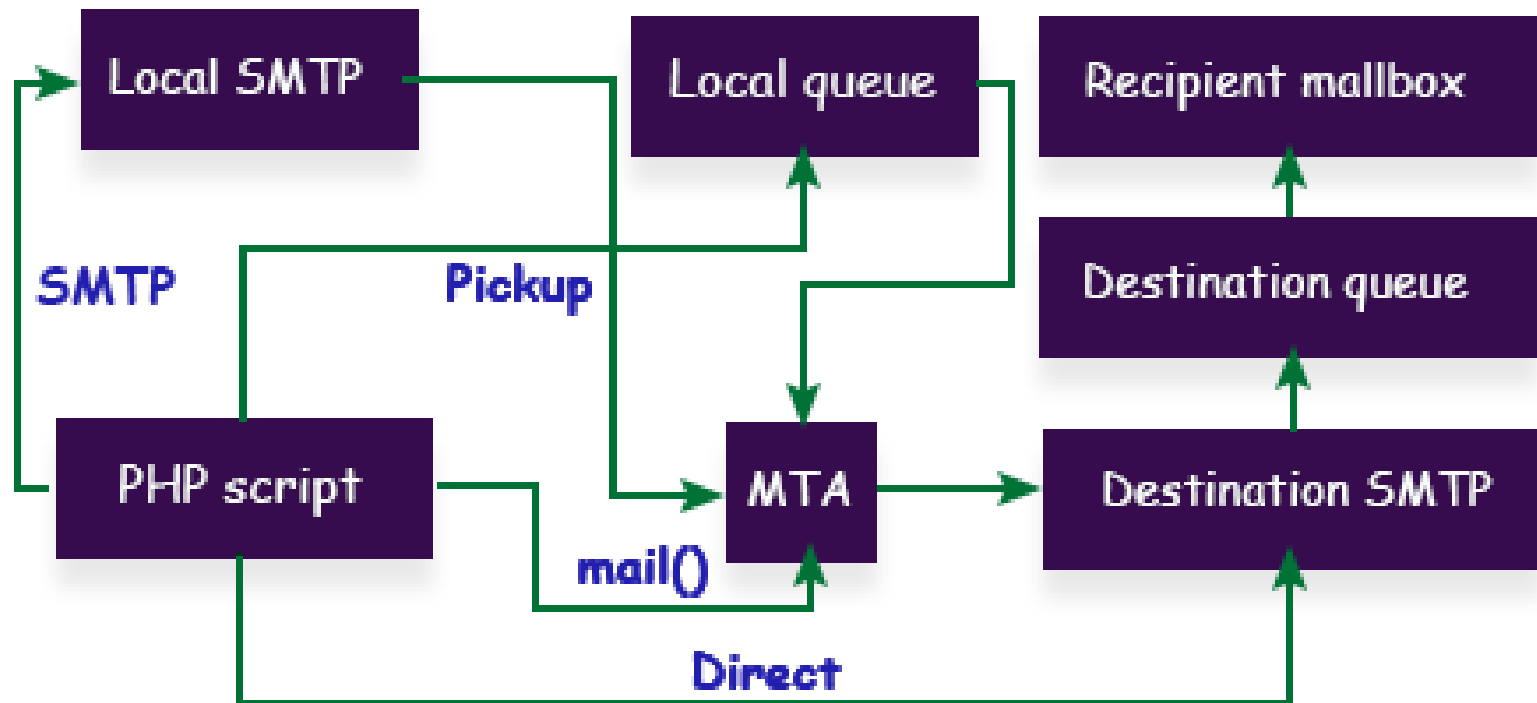
```
<?php
if(!file_exists(
"data.txt")){
    echo "File does not
exists"; exit;
}
echo "File Exists";
?>
```

Sending emails using PHP

PHP Mail function

The **mail()** function allows you to send emails directly from a script.

- The mail functions are part of the PHP core functions. For the mail functions to be available, PHP requires an installed and working email system.
- The program to be used is defined by the configuration settings in the php.ini file.



PHP Mail function

The **mail()** function needs configuration settings to work properly in **php.ini** file, some of them:

Name	Default	Description	Changeable
SMTP	"localhost"	Windows only: The DNS name or IP address of the SMTP server	PHP_INI_ALL
smtp_port	"25"	Windows only: The SMTP port number.	PHP_INI_ALL
sendmail_from	NULL	Windows only: Specifies the "from" address to be used when sending mail from mail()	PHP_INI_ALL
sendmail_path	"/usr/sbin/sendmail -t -i"	Specifies where the sendmail program can be found. This directive works also under Windows. If set, SMTP, smtp_port and sendmail_from are ignored	PHP_INI_SYSTEM

PHP Mail function

- The **mail()** function signature:
mail(to,subject,message,headers,parameters);
- When the configuration is setup the email can be generated as below:

```
1  <?php
2  // the message
3  $msg = "This is a \n Test Message\nThaank you!";
4
5  // use wordwrap() if lines are longer than 70 characters
6  $msg = wordwrap($msg,70);
7
8  // send email
9  mail("abc@abc.net","Test Message",$msg);
10 ?>
```


Object Orientation with PHP

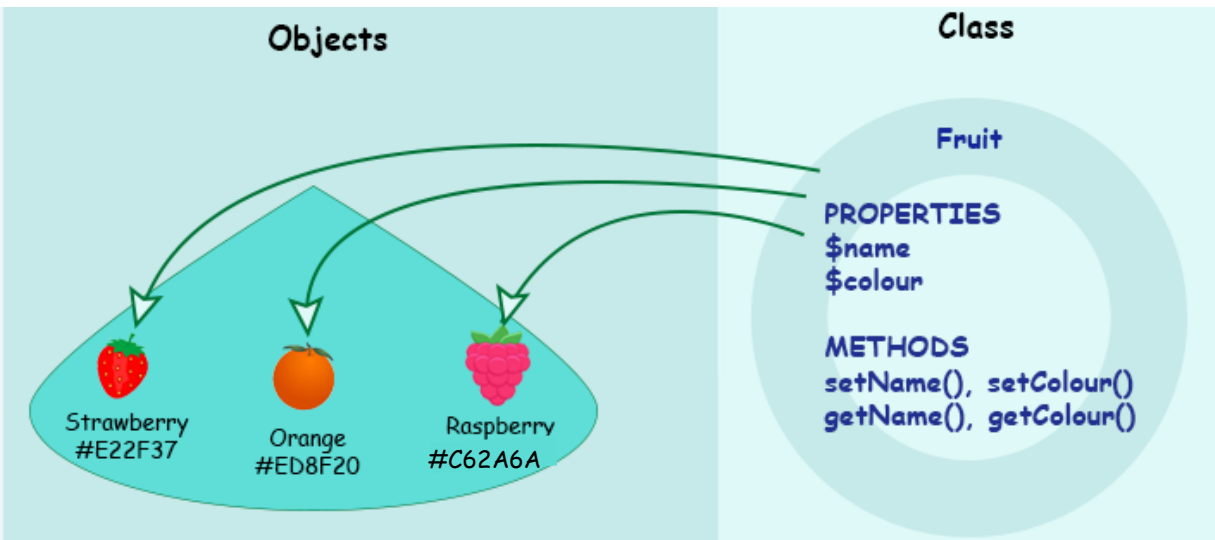
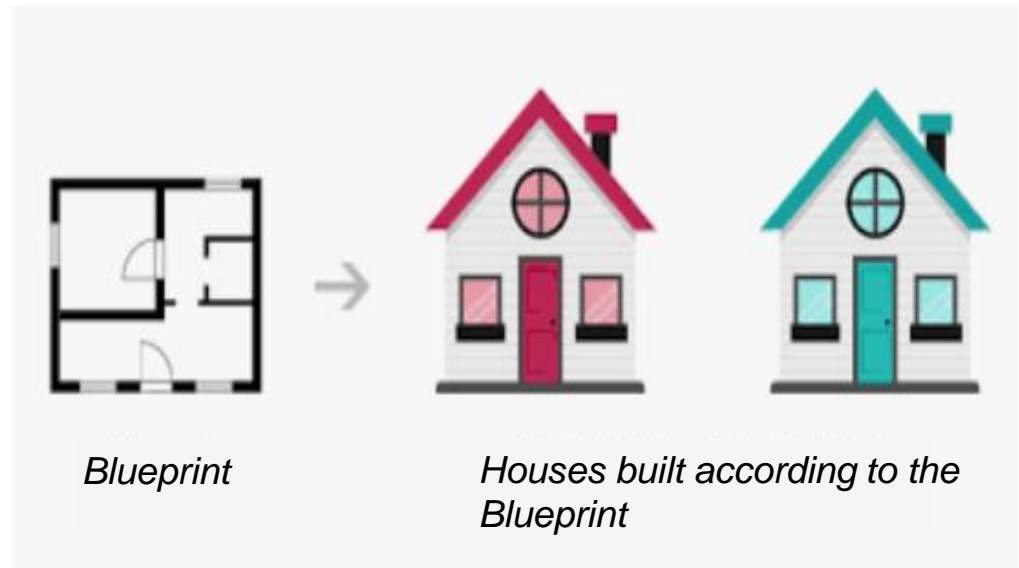
Object-Oriented Programming

- Object Oriented Programming (OOP) refers to the creation of reusable software components(classes) that integrate **properties**(data) and **behaviors** (functions) of real-world entities together.
- In OOP an object can represents any entity. (a student, a desk, a button, a file, a text input area, a loan, a web page or a shopping cart)
- An object-oriented program comprises of a collection of such objects that interact with each other to solve a particular problem/s.

Object-Oriented Programming

- Objects are self-contained
 - data and operations that pertain to the object are assembled into a single entity.
- In OOP each Object has:
 - An identity
 - State
 - Behavior

Building Objects According to a Template/Blue print/Plan



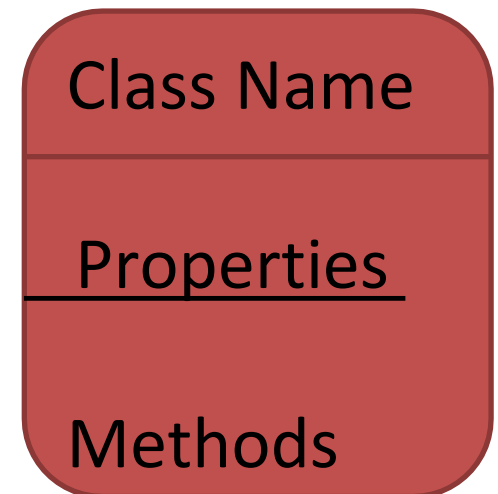
Class and Object

- A “Class” refers to a blueprint. It defines the attributes(variables) and behaviors(functions) the objects of that class should support.
- An “Object” is an instance of a class. Each object should corresponding to a class(es) which defines its attributes and behavior.

The Class

- The basic unit of code in object-oriented PHP is the class. A class provides a mechanism to encapsulate related functionality and data into a single entity.
- In PHP a class can be defined by using the keyword **'class'** as below.
- The class name can be any valid label and it cannot be a PHP reserved word.

```
class Circle
{
// Class properties and
methods
}
```



Properties

- In PHP5, class properties are used as placeholders, to store data associated with the objects of that class. The visibility of a property can be defined by adding one of the following prefixes to the declaration of the property.
 - public : the value of the property can be accessed from everywhere. If no visibility is specified for a method, it defaults to public visibility.
 - protected : the value of the property can be accessed only by the class and the derived classes(child classes).
 - private : the value of the property can be accessed only by the class that defines the member.

A Class with Public and Private Properties - Example

```
class Person{  
  
    public $name;  
  
    public $sex = "m";//default-  
        value  
  
    public $dob;  
  
    private $bank_account_no;  
  
}
```


Creating objects(Instances) of a class

- In order to access the properties and use the methods of a class, you first need to instantiate, or create an instance(object). This can be done by using the keyword '*new*' as below:

```
$c = new Person();
```

Classes should be defined before instantiation.

\$c variable holds a **reference** to an instance (object) of the class 'Person'.

Once an object is created, the individual (visible) properties and methods of that object can be accessed by using an arrow (->) operator as given below.

```
$c->name = "Sunil";
```

Object assignments

- When assigned an already created instance of a class to a new variable, the new variable also points to the same instance. Example :

```
$p1 = new Person();
```

```
$p1->name = "Sunil";
```

```
$p2 = $p1; // $p1 and $p2 points to the same object
```

```
$p2->name = "Kamal";
```

```
echo $p1->name; // This will print the text "Kamal"  
as $p1 and $p2 points  
to the same object
```

Class Methods

- Class properties are used to hold data inside objects. Functions can be created inside a class to manage its property values. Such functions defined inside classes are called its methods.

Syntax for method definitions:

```
visibility function function_name(parameters)
{
    // method implementation here
}
```

Class Methods

```
class Person{
    public $name;
    public $sex = "m"; // default
value
    public $dob;
    private $bank_account_no = ;

Public function set_name($name) {
    $this->name = $name;
}

Public function print_name() {
    echo $this->name;
}
}
```

\$this is a The pseudo-variable. It is used to refer to the calling object to which the method belongs.

Constructors and Destructors

- In some situations when dealing with classes, you might want to have a way to automatically initialize object variables and/or to perform certain pre-defined actions when the object is created. For such situations, a constructor can be used.
- A constructor is nothing more than a specially named method that is automatically called when an object is instantiated. In PHP5, to implement a constructor, all you need to do is implement a method named “**__construct**”.

Constructors and Destructors

- PHP5 now includes a special method (destructor) that is called when an object is destroyed.
- An object destructor is called when all references to an object are removed, or it is manually destroyed in your code.
- To create a destructor, add a method to your class, and call it **“__destruct”**.

Class Object

{

function __construct() {}

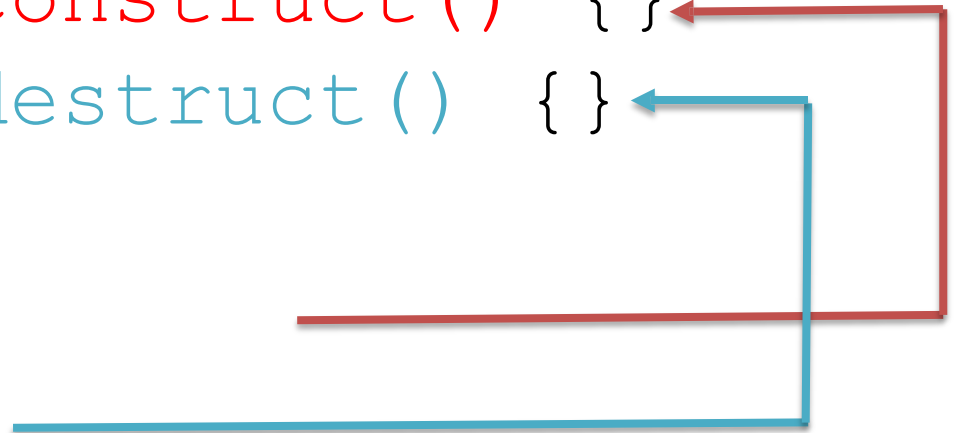
function __destruct() {}

}

\$obj=

newObject();

unset(\$obj);



Example: __construct

```
<?php
class Person{
    public $name = null;
    public $sex = "m";
    public $dob;
    private $bank_account_no;

    function __construct($name,$sex,$dob,$acc){
        $this->name = $name; $this->sex = $sex;
        $this->dob = new DateTime($dob);
        //$dob should be give as "2015-01-15"
        $this->bank_account_no = $acc;
    }

    public function print_age($toDate){
        //$toDate should be give as "2015-01-15"
        $interval = $this->dob->diff(new DateTime($toDate));
        echo "Years - ". $interval->y . " Months - ".$interval->m ." Days
        -
        ".$interval->d ;
    }
}
```



```
$p1 = new Person("Saman","m","1960-11-23","123456");
$p1->print_age("2015-02-06");
?>
```


Static Keyword

- Sometimes when using object-oriented programming, you might need to assign properties/methods with a class rather than with its instances. This can be done by using static properties/methods. Static properties/methods exist only with the class but not with its instances.
- Static Properties/methods of a class can be accessed by using the operator “::”.

self::

- In order to access static variables within the same class, you can use the ***self*** keyword followed by the double-colon (“::”)
- Using **self::** is similar to \$this->, but it is used for static members only.

Example: *self*

```
<?php
class Person{
    public $name = null;
    public $sex = "m";
    private static $ObjectCount =
    0; function _construct($name,$sex){
        $this->name = $name;
        $this->sex = $sex;
        self::$ObjectCount++;
    }
    public function print_object_count(){
        echo "Number of objects instantiated -
". self::$ObjectCount;
    }
}
```

```
$p1 = new Person("Saman","m");$p2 =
new Person("Kamala","f");
Person::print_object_count();
?>
```

Class Constants

- It is possible to define class values that are constant for the class. To define a class constant, use the “**const**” keyword before the constant name.
- Constants differ from normal variables in that you don't use the \$ symbol in their declaration.

```
<?php
class Person{
    const office = "UCSC";
    public $name = null;
    public $sex = "m";

    function __construct($name,$sex){
        $this->name = $name;
        $this->sex = $sex;
    }

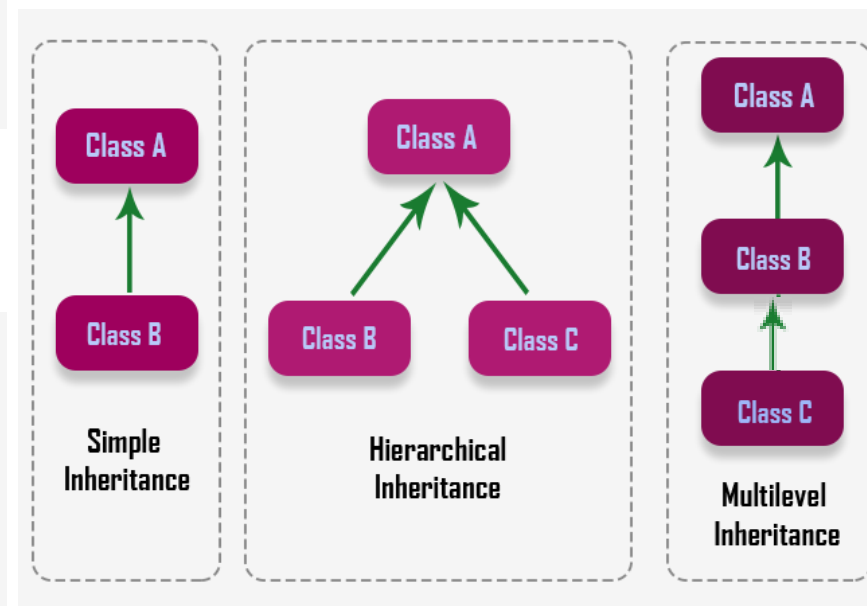
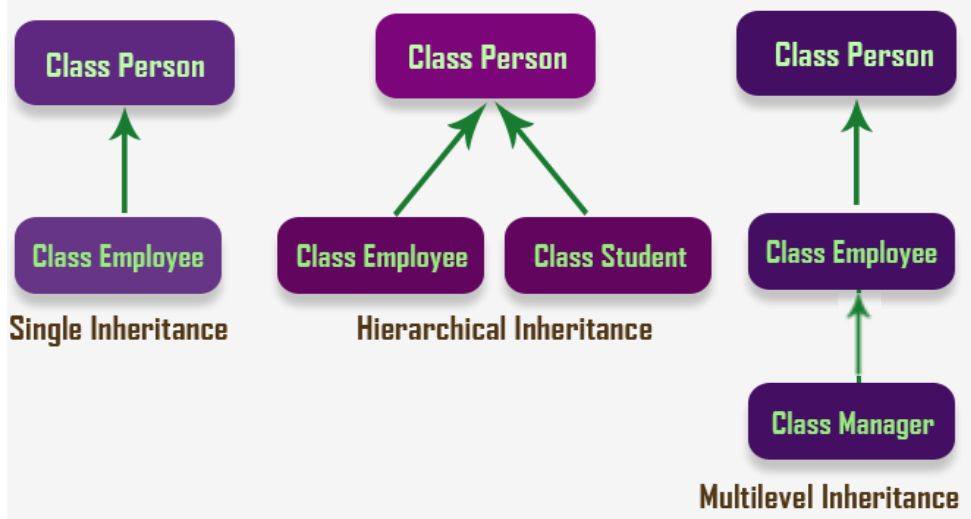
    public function print_office(){
        echo "Office name -".
self::office;
    }
}

Person::print_office();
?>
```

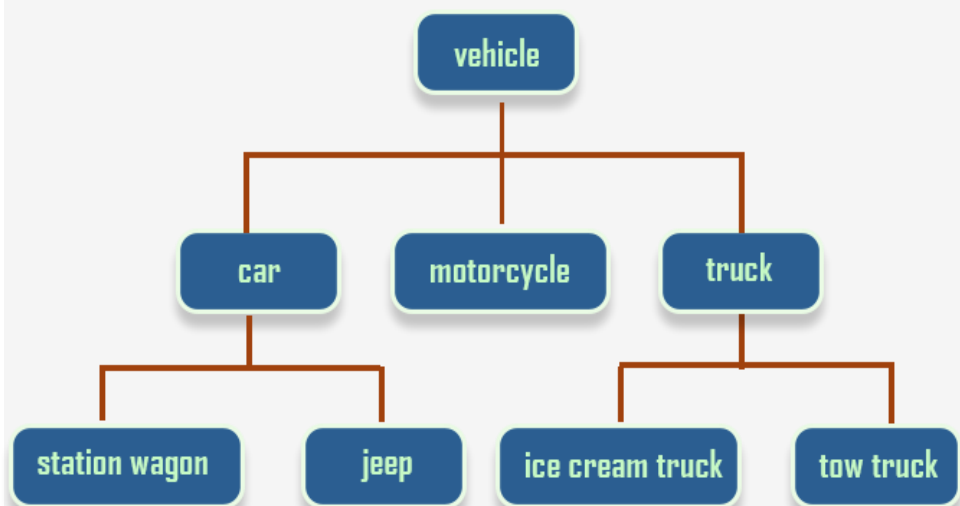
Example:
const

Inheritance

- Allows you to define a base set of properties and methods that belong to a base class and to extend that class by
 - adding additional properties and methods and/or
 - changing the behavior of existing methods.
- The subclass inherits all of the public and protected properties and methods from the parent class. Unless a subclass overrides a method, the subclass retains its original functionality defined in the parent class.
- Inheritance facilitate the implementation of additional functionality in similar objects without the need of re- implementing all of the shared functionality.
- When defining a subclass the parent class must be defined before defining the child class.



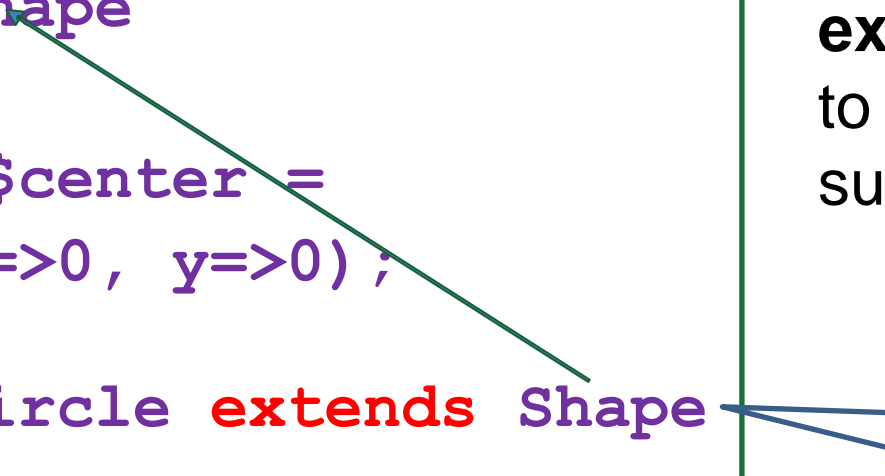
Building classes by inheritance



The extends key word

```
<?php
    class Shape
    {
        public $center =
        array(x=>0, y=>0);
    }
    class Circle extends Shape
    {
        public $radius;
    }
    $c = new Circle();
    print_r($c->center);
?>
```

The keyword **extends** is used to build a subclass



Parent Class

The final Keyword

- There are cases where, you want to restrict a subclass from redefining a member that exists in a parent class.
- You can prevent properties and methods from being redefined(overriding) in a subclass by using the **final** keyword.

Using parent:: References

- In some situations you may want to refer to a property or a method of the parent class, in a subclass.
- To achieve this, you can use the **parent** keyword in conjunction with the **::** (double colon) similar to static members.

```
<?php
class Shape {
    var $x;
    function getName()
    {
        $this->x = "I'm a shape";
        return;
    }
}

class Circle extends Shape {
    // we have var $x; from the parent already here.
    function getParentName()
    {
        parent:: getName() ;
        echo $this->x;
    }
}

$b = new      Circle();
$b-> getParentName(); // prints: " I'm a shape "
?>
```

Abstract Classes

- When a class is defined as abstract, other classes can extend it, but it cannot be instantiated. This feature enables you to define classes as templates.
- A class that contains at least one abstract method is treated as an abstract class.
- Abstract methods only defines the signature of the method, but not its implementation.
- When inheriting from an abstract class, all methods declared as abstract in the parent class must be defined by the child.

```
<?php
abstract class Shape{
    public $origin = array(x=>0, y=>0);
}

class Circle extends Shape{
    // Circle implementation
}

$c = new Circle();
echo $c->origin;
$s = new Shape(); echo $s-
>origin;
?>
```

Interfaces

- Another new object-oriented feature in PHP5 is the ability to create and use interfaces. Interfaces, in a nutshell, are a way to specify what methods a class must explicitly implement. This is useful when dealing with many interconnected objects that rely on the specific methods of one another.
- In PHP5, an interface is defined using the **interface** keyword, and implemented using the **implements** keyword.
- All methods declared in an interface must be public.
- Interfaces can be extended like classes using the **extends** operator.

Interfaces

```
interface TwoDimensionalOperations
{
    public calculateArea() ;
}
class Circle implements
TwoDimensionalOperations
{
    public calculateArea() ;
    {
        // Implementation of calculateArea,
        specific to this Circle class
    }
}
```

Abstract Classes Vs Interfaces

- A child class can extend only one abstract class, whereas a class can implement multiple interfaces.
- An interface does not provide any functionality (method implementations) whereas an abstract class may provide some functionality.

Magic Methods

- Magic methods is a set of methods designed to be executed automatically in response to particular PHP events.
- All names of magic methods starting with two underscores.
- PHP reserves all function names starting with “__” as magical, thus it is recommended not to start any user defined function with “__”.

i.e:

- __call
- __get and __set
- __toString

__call()

- Allows you to provide actions or return values when undefined methods are called on an object.
- Can be used to simulate method overloading, or even to provide smooth error handling when an undefined method is called on an object.

```
public function __call($m, $a){  
    echo "The method " . $m . " was called.<BR> The  
    arguments were as follows:<BR>;  
    print_r($a);  
}
```

__get and __set

- **__get** allows properties which actually not accessible in a class to be read.
- **__set** allows properties which actually not accessible in a class to be written.
- **__get** takes one argument - the name of the property.
- **__set** takes two arguments - the name of the property and the new value.

__toString

- **__toString** returns a custom string value that is automatically used when the object is converted to a string.
- Only called when used directly with **echo** or **print**. If not implemented in a class the object id will be returned by default.

Activity: Classes

- Complete the PHP class given below according to the comments.

```
1  <?php
2  ▼ class Fruit {
3      public $name;
4      public $color; //default color is "green";
5
6  ▼  function __construct($name) {
7      $this->name = $name;
8  }
9
10     public function setColor($c) { //update the color variable
11     }
12
13  ▼  function __destruct() {
14      echo "The fruit is {$this->name}.\n";
15  }
16
17     public function __toString(){ //return "color-name";
18     }
19 }
20
21 $apple1 = new Fruit("Apple");
22 $apple2 = new Fruit("Apple");
23 $apple1->__destruct();
24 $orange1 = new Fruit("Orange");
25 echo $orange1, "\n";
26 $orange1->setColor("yellow");
27 echo $apple2, "\n", $orange1, "\n";
28 ?>
```

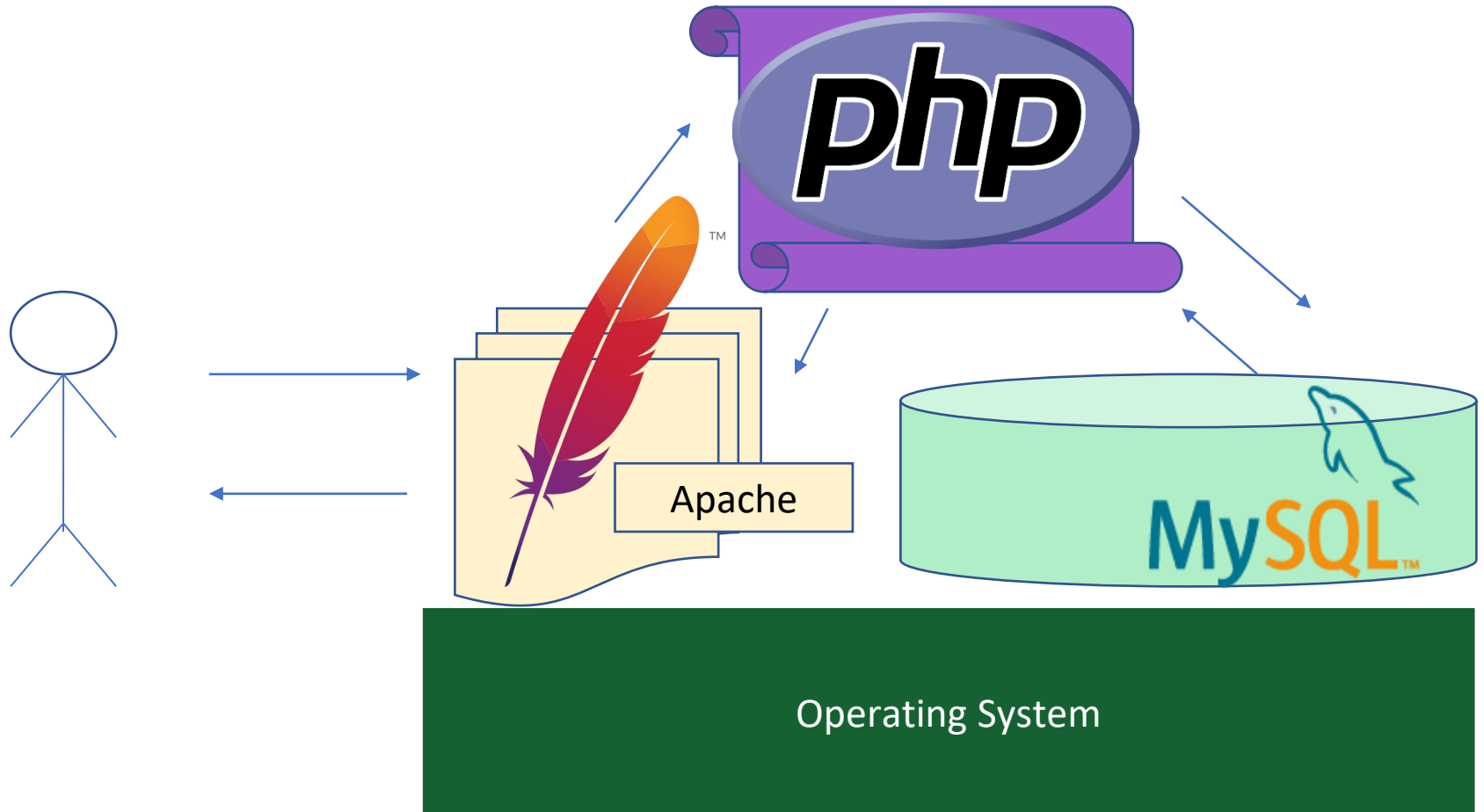
Activity: Classes

Answer:

```
1  <?php
2  class Fruit {
3      public $name;
4      public $color = "green";
5
6      function __construct($name) {
7          $this->name = $name;
8      }
9
10     public function setColor($c) {
11         $this->color = $c;
12     }
13
14     function __destruct() {
15         echo "The fruit is {$this->name}.\n";
16     }
17     public function __toString()
18     {
19         return ($this->color)."-".($this->name)."\n";
20     }
21 }
22
23 $apple1 = new Fruit("Apple");
24 $apple2 = new Fruit("Apple");
25 $apple1->__destruct();
26 $orangel = new Fruit("Orange");
27 echo $orangel;
28 $orangel->setColor("yellow");
29 echo $apple2,$orangel;
30 ?>
```

Developing a web application with PHP

Technology Stack



How does the Apache/PHP/MySQL web application work?

- The client browser send a request to the Apache web server.
- The server checks what is the php resource requested in the request.
- Together with the request data server invokes the PHP interpreter and script
- The interpreter execute the script and communicates with the MySQL data base according to the instructions in the script.
- The databased operation completes, and the scripts output is captured by the interpreter to pass back to the apache server.
- Apache server send the output back to the client browser.

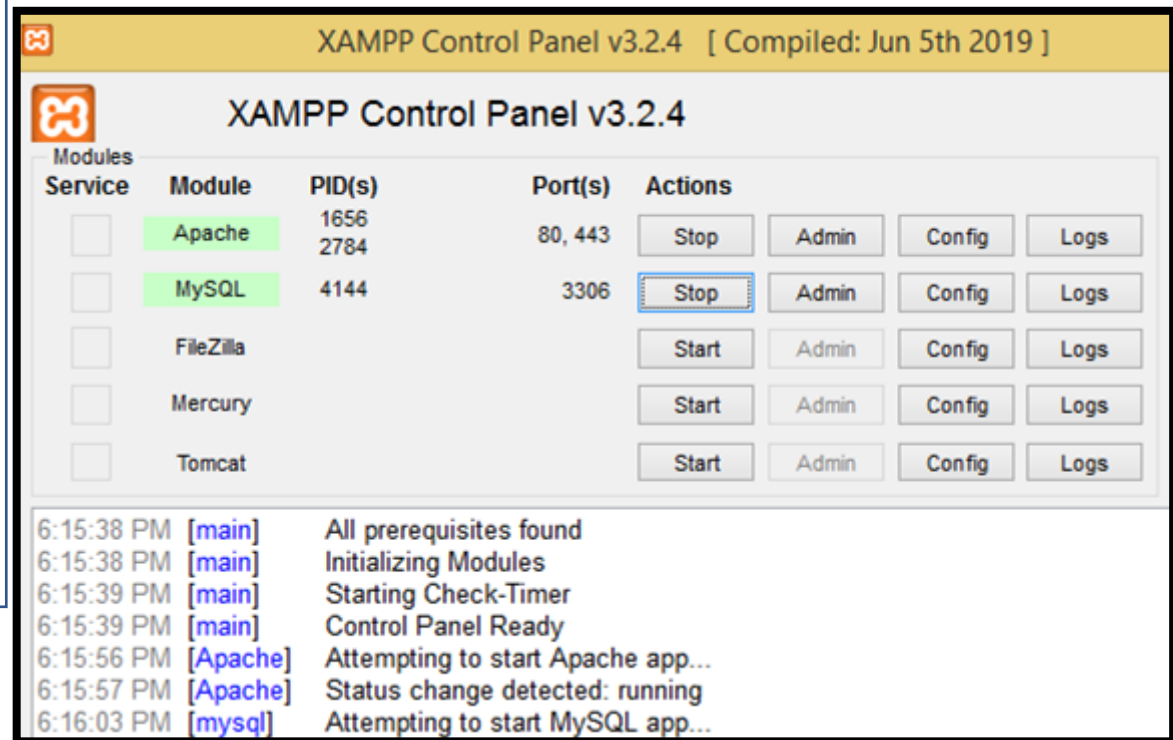
Introduction to MySQL

- A free and open source relational database management system (RDBMS)
- MySQL is used by many database-driven web applications, including Drupal, Joomla, phpBB, and WordPress together with PHP.
- MySQL is also used by many popular websites, including Facebook, Flickr, MediaWiki, Twitter and YouTube.

Installation of MySQL

- We have more than one approach to install MySQL:
 1. Download and install the MySQL server.
 2. Use XAMPP bundled MySQL installation.

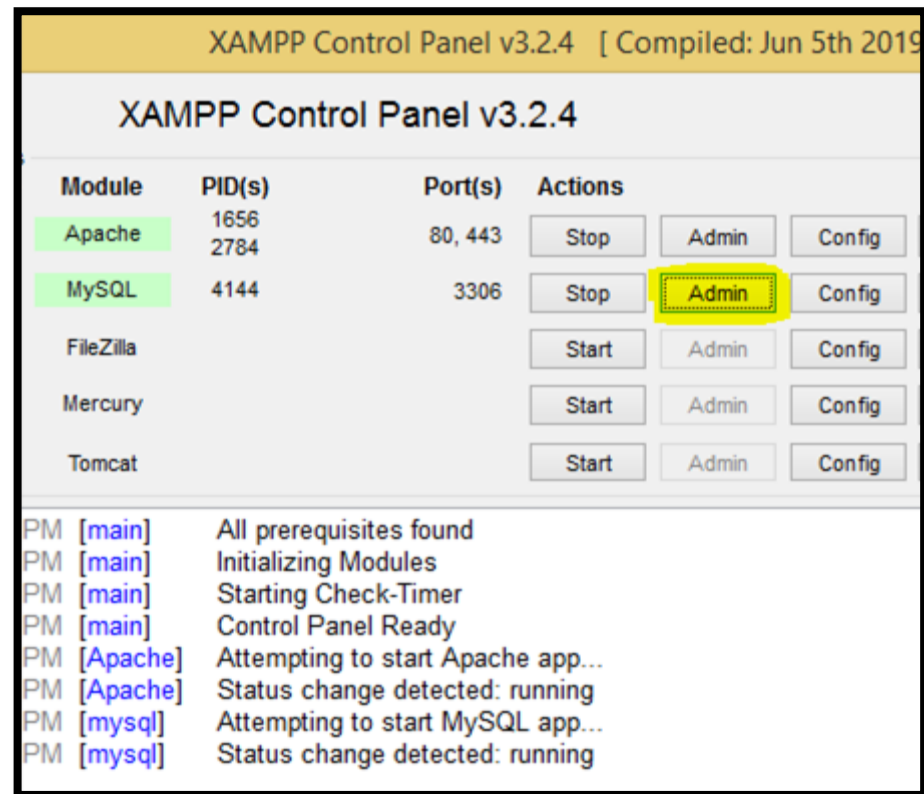
You can start the MySQL server by the control panel as shown here.



Installation of MySQL

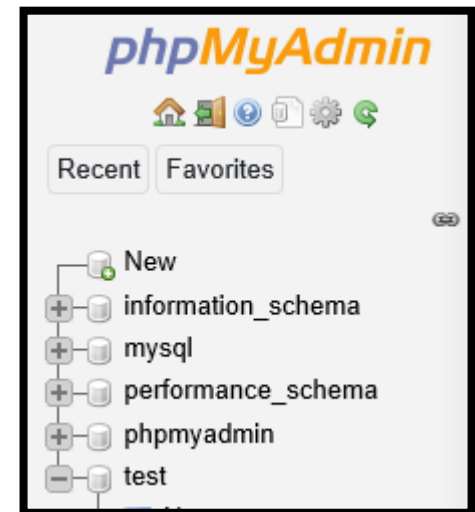
- To access the database click the **Admin** button.

- This will launch the **phpMyAdmin** console to manage the database

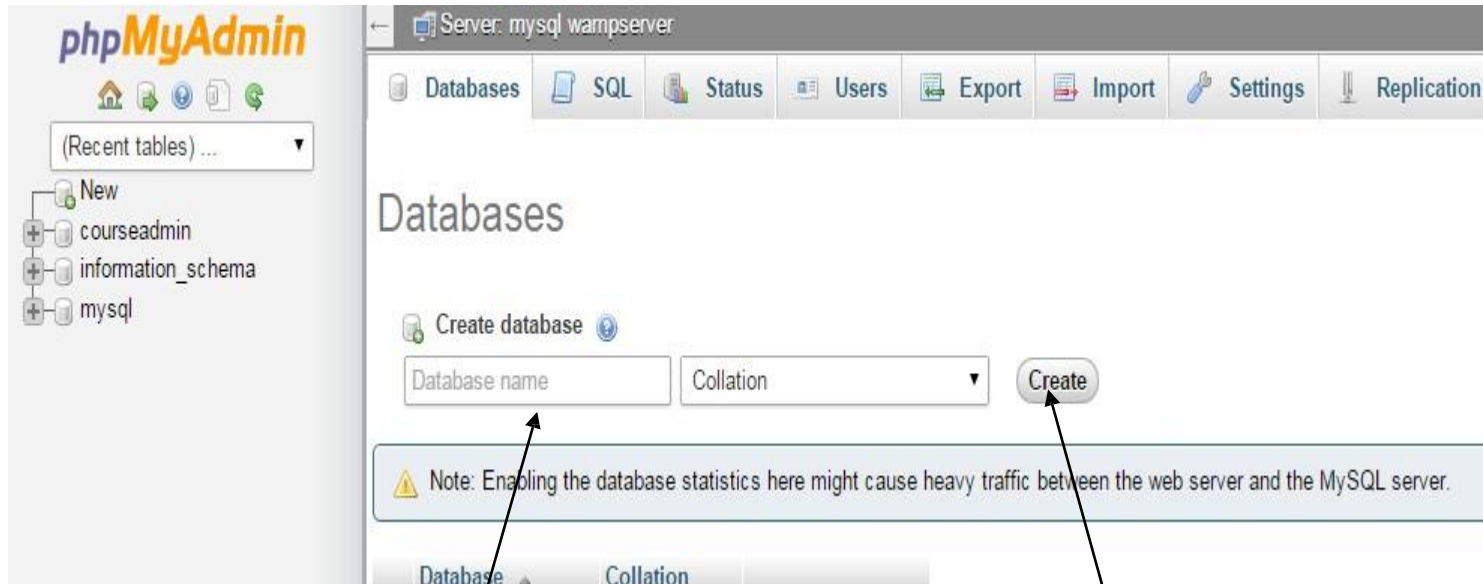


Creating a Database

- There are 2 main ways of creating a database :
 - With the command line
 - By using a tool such as MySQL workbench
- Since you are using XAMPP package, we will use the phpMyAdmin to make tables.



Creating a database by using phpMyAdmin tool



- 1) Click on databases and give a suitable name . Click 'C2) Click 'Create'

Creating Databases-(with SQL command)

The screenshot shows a web-based SQL interface. At the top, there is a navigation bar with tabs: Databases, SQL, Status, Users, Export, Import, and More. The 'SQL' tab is selected. Below the navigation bar, there is a text input field with the placeholder text 'Run SQL query/queries on server "127.0.0.1":'. Below this is a large text area containing the SQL command: `1 CREATE DATABASE IF NOT EXISTS myDB` and `2`. Below the text area is a 'Clear' button. Below the 'Clear' button is a text input field with the placeholder text 'Bookmark this SQL query:'. At the bottom of the interface, there is a row of options: '[Delimiter :]', a checked checkbox 'Show this query here again', an unchecked checkbox 'Retain query box', and a 'Go' button. Three numbered instructions are overlaid on the screenshot: 1) Click on SQL tab (pointing to the SQL tab), 2) Type the correct command (pointing to the SQL command text area), and 3) Click on the Go button (pointing to the Go button).

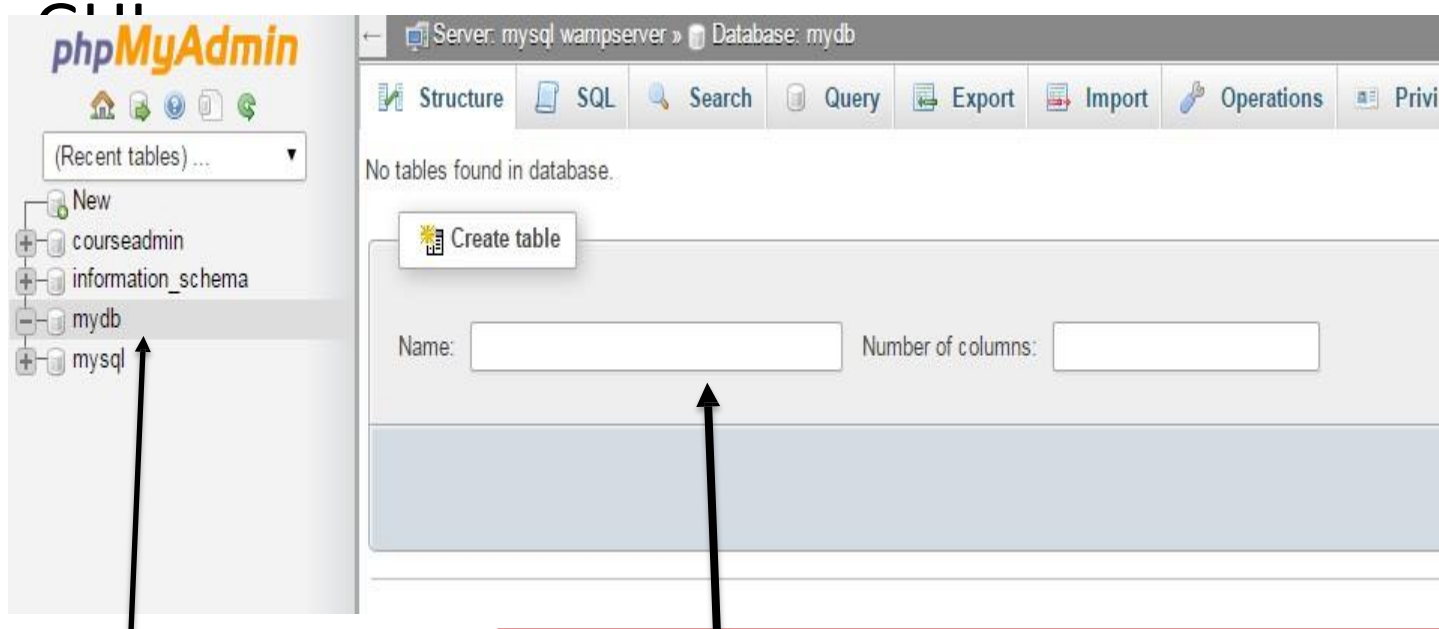
1) Click on SQL tab

2) Type the correct command

3) Click on the Go button

Creating Table (GUI)

- You can create tables in a selected DB by executing the relevant command or by using the

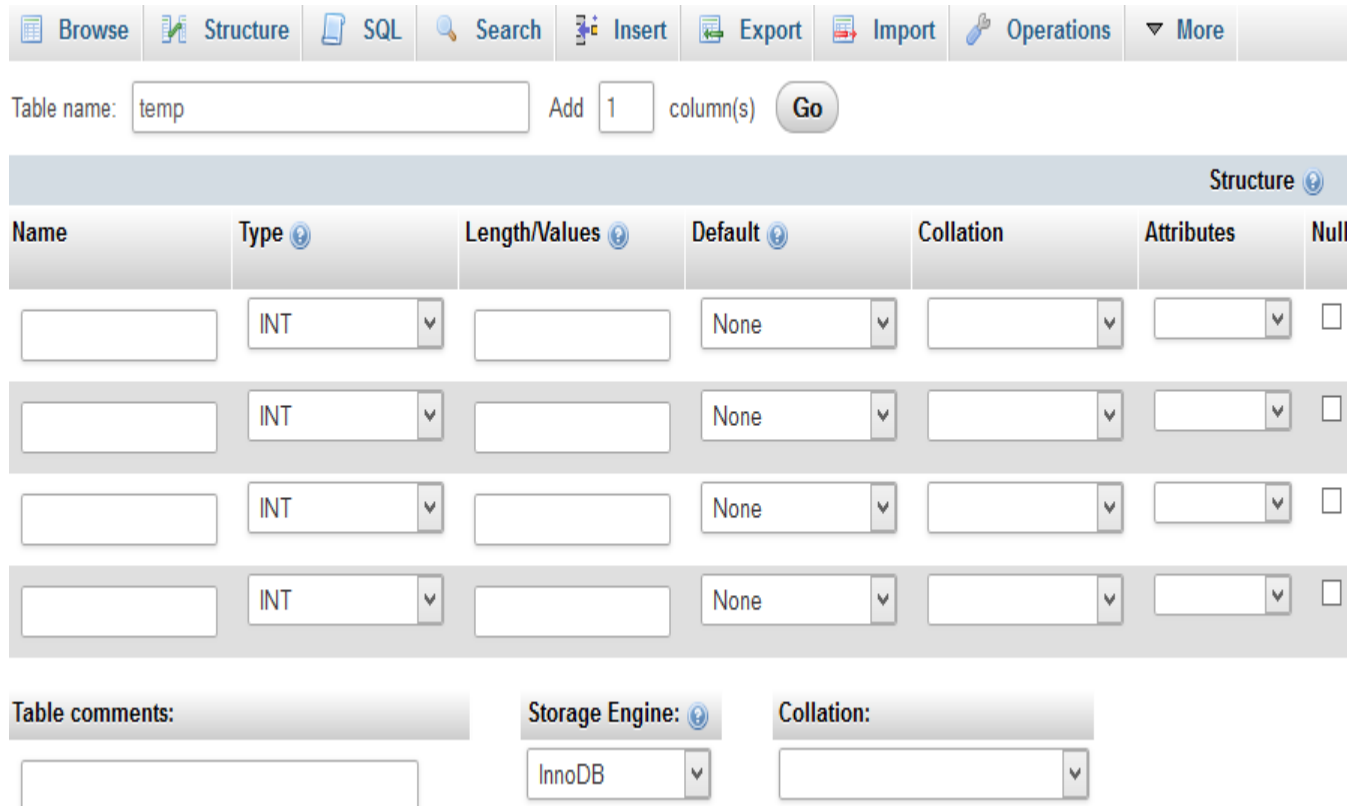


1. Select the database

2. Type a suitable table name and number of columns for the table

Creating Table (GUI)

- Columns (Fields) of the table can be created by filling the subsequent form as required.



The screenshot shows the MySQL Table Creation GUI. At the top is a toolbar with icons for Browse, Structure, SQL, Search, Insert, Export, Import, Operations, and a More dropdown. Below the toolbar, the 'Table name' field contains 'temp', followed by 'Add 1 column(s)' and a 'Go' button. The main section is titled 'Structure' and contains a table with the following columns: Name, Type, Length/Values, Default, Collation, Attributes, and Null. There are four rows of input fields for these columns. Each row has a text box for 'Name', a dropdown for 'Type' (all set to 'INT'), a text box for 'Length/Values', a dropdown for 'Default' (all set to 'None'), a dropdown for 'Collation', a dropdown for 'Attributes', and a checkbox for 'Null'. At the bottom, there are three sections: 'Table comments:' with a text box, 'Storage Engine:' with a dropdown set to 'InnoDB', and 'Collation:' with a dropdown.

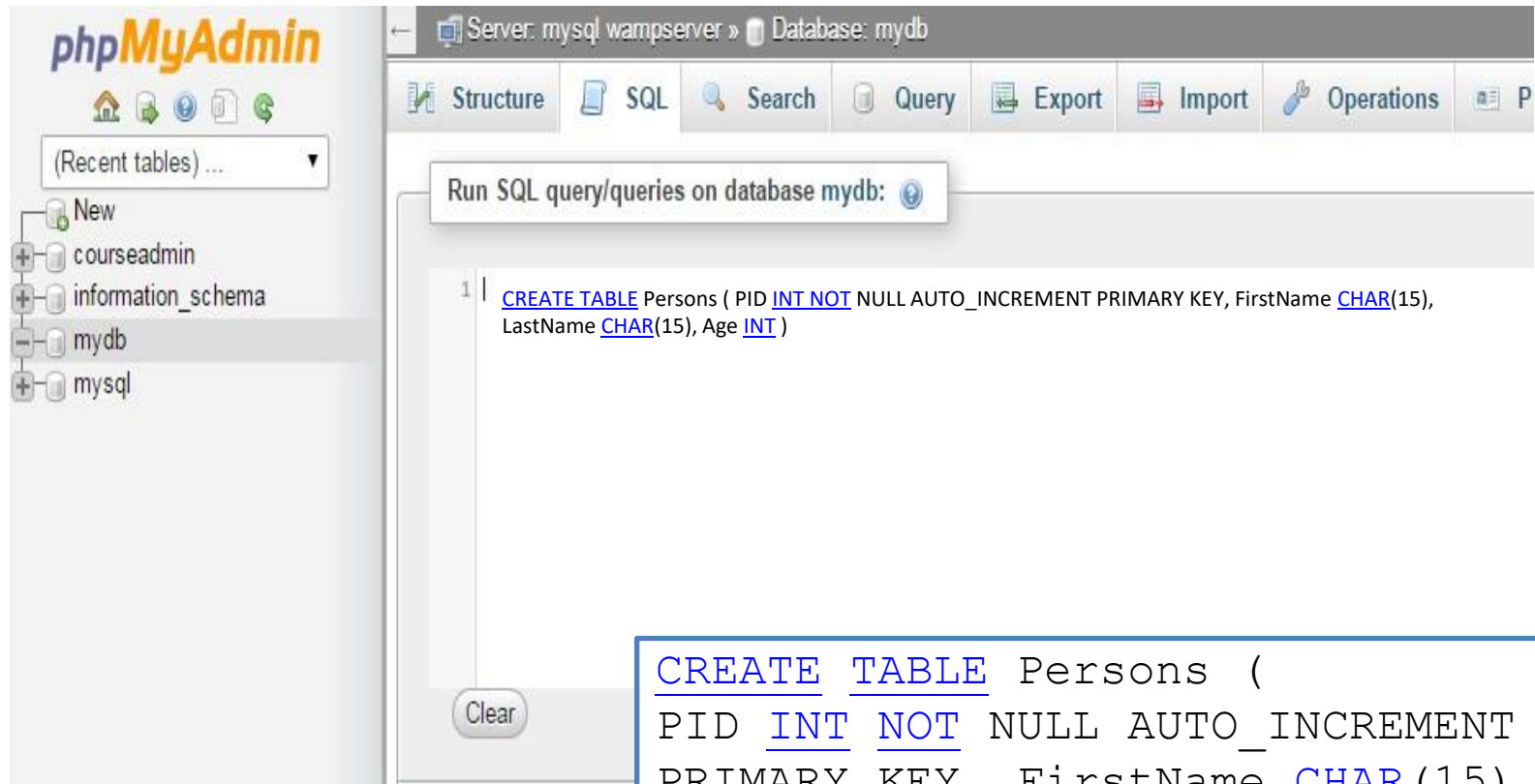
Name	Type	Length/Values	Default	Collation	Attributes	Null
<input type="text"/>	INT	<input type="text"/>	None	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
<input type="text"/>	INT	<input type="text"/>	None	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
<input type="text"/>	INT	<input type="text"/>	None	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>
<input type="text"/>	INT	<input type="text"/>	None	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>

Table comments:

Storage Engine: InnoDB

Collation:

Creating Tables (SQL commands)



```
CREATE TABLE Persons (  
PID INT NOT NULL AUTO_INCREMENT  
PRIMARY KEY, FirstName CHAR(15),  
LastName CHAR(15),  
Age INT )
```

Managing data stored in MySQL DBs Through PHP

Basic Steps in Processing data stored in MySQL through PHP programs

1. Connect to a host server with MySQL installed.
2. Select a database
3. Create a SQL statement
4. Execute the SQL statement.
 - Many SQL statements return the result of a SQL statement as a **record set**
5. Extract data from record set using PHP commands
6. Use the data as required
7. Close the connection

Open a Connection to MySQL

- Opening a connection to a MySQL DB

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password,
$dbname);

// Check connection
if ($conn->connect_error) {
    die("Database connection failed: " . $conn-
>connect_error);
}
echo "Success. Connected to database";
?>
```

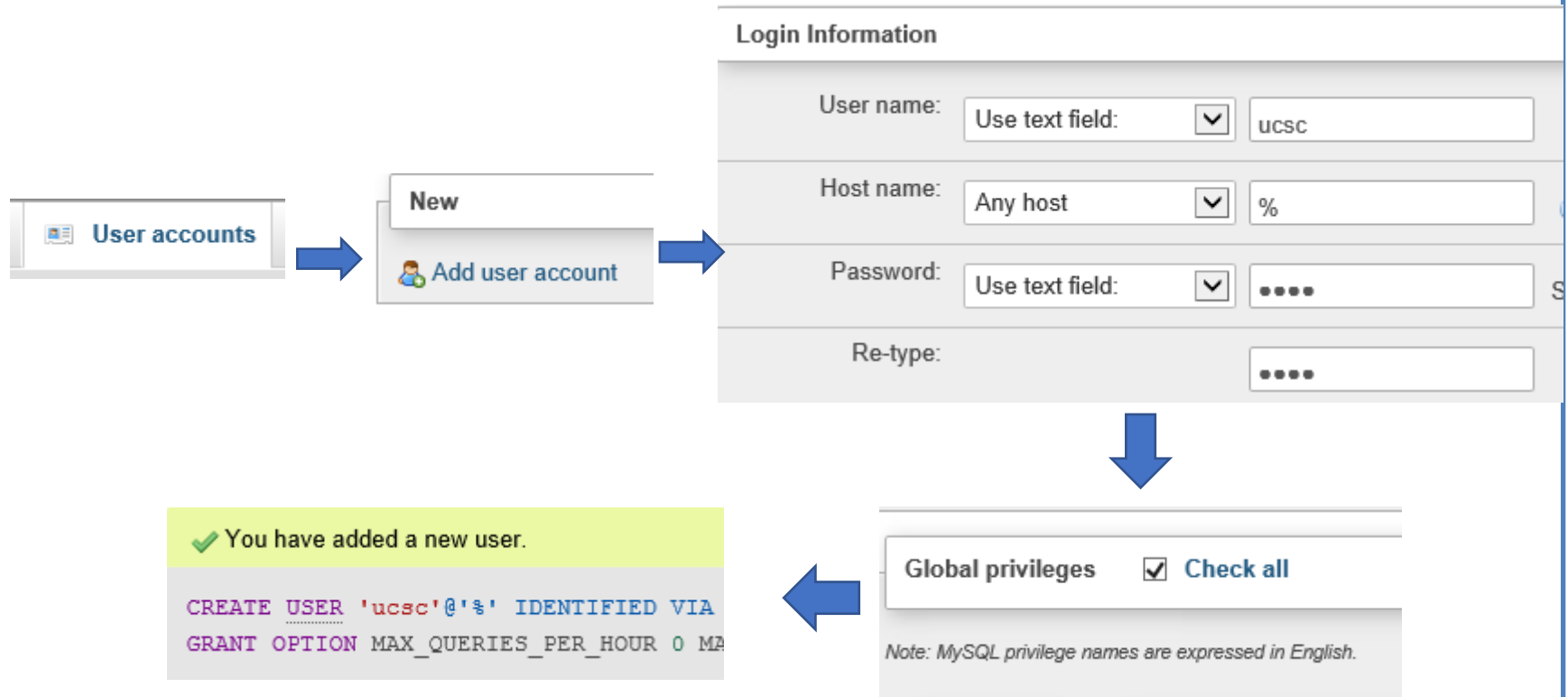
Usually "localhost"

By default – 'root'

By default – ''

Open a Connection to MySQL

- Create a new user from the console to connect to the database myDB



Close the Connection

- It's always a best practice to close a connection once you are done with working with the database.
- Can close the connection using this syntax.

```
// if the connection object is $conn  
$conn->close();
```

mysqli_query()

- This is one of the most important and most used function in php when dealing with MySQL.
- mysqli_query() function is used to command PHP to execute a SQL statement.
- It sends a query or command to a MySQL DBMS through the connection object.

Inserting Data Into a Database Table

- You can use **INSERT INTO** statement to add new records to a database table.
- There are 2 different ways of writing insert queries
 - **INSERT INTO** table_name **VALUES** (value1, value2, value3,...)
 - **INSERT INTO** table_name (column1, column2, column3,...) **VALUES** (value1, value2, value3,...)
- The first form can be used if data is inserted to all columns of the new record.
- The second form can be used if data is inserted only to a selected set of columns in the new record.

Executing a SQL query through PHP

- The following PHP code segment inserts two record to the table 'Persons'

```
<?php
$con=mysqli_connect("localhost", "root", " ",
"myDB");
if ($con->connect_error)
    die("Database connection failed: " .
        $conn->connect_error);

mysqli_query($con,"INSERT INTO Persons (FirstName,
        LastName, Age)VALUES
        ('Nimal', 'Perera',35)");

mysqli_query($con,"INSERT INTO Persons (FirstName,
        LastName, Age)VALUES
        ('Amal', 'Silva',33)");

mysqli_close($con);
?>
```

Structure of the table

Persons{

PID INT NOT NULL

AUTO_INCREMENT

PRIMARY KEY,

FirstName CHAR(15),

LastName CHAR(15),

Age INT)

Inserting data to a MySQL DB through a HTML form.

- This HTML page requests the web server to execute a PHP script named “insert.php” at the server side.

```
<html>
<body>
<form action="insert.php" method="post">
    Firstname: <input type="text"
name="firstname"><br>
    Lastname: <input type="text"
name="lastname"><br>
    Age: <input type="text"
name="age"><br>
    <input type="submit">
</form>
</body>
</html>
```

Insert data into a database table

Content of the PHP script insert.php

```
<?php
$con=mysqli_connect("localhost","root","", "myDB");
if ($con->connect_error) die("Database connection failed:
\".$con->connect_error);
$firstname = $_POST['firstname'];
$lastname = $_POST['lastname'];
$age = $_POST['age'];
$sql = "INSERT INTO persons (FirstName, LastName, Age)"
        . "VALUES ( '$firstname', '$lastname', $age )";
if(mysqli_query($con,$sql)){
    echo "Data inserted to the Table successfully";
}else {
    echo "Error in inserting data". $con->error;
}
mysqli_close($con);
?>
```

Selecting and Displaying Data

```
<?php
$con=mysqli_connect("localhost","root","","myDB");
if ($con->connect_error) die("Database connection failed: " .
    $con->connect_error);
$sql = "select * from persons";
$result = mysqli_query($con,$sql);
if(!$result){
    die("Error in executing the SQL" . $con->error);
}
while ($row = mysqli_fetch_array($result)){
    echo $row['FirstName'] . " " . $row['LastName'] . "<br>";
}
mysqli_close($con);
?>
```

selects all data stored in the “persons” table and display only the content of the ‘FirstName’ and ‘LastName’ columns.

Select Data satisfying a where clause

- We can use the Where clause to filter data.

```
<?php
$con=mysqli_connect("localhost","root","123456","bit
");
if ($con->connect_error) die("Database connection
failed: " .
        $con->connect_error);
$sql = "select * from persons where
FirstName='Nimal'";
$result = mysqli_query($con,$sql);
if(!$result){
    die("Error in executing the SQL" . $con->error);
}
while ($row = mysqli_fetch_array($result)){
    echo $row['FirstName'] . " " . $row['LastName'].
    "<br>";
}
mysqli_close($con);
?>
```

Earlier example selected all the Records from the table , but here we are using a where clause to filter data so that it will only return records where the First name field is 'Nimal'

MySQL Update

- Whenever you need to update a record which exist in a table, you can use update query.

```
UPDATE table_name  
SET column1=value,  
column2=value2,...  
WHERE some_column=some_value
```

Here the 'Where' clause decide which records to be updated. If you remove the WHERE clause, all records will be updated

Changing Data in the DB

```
<?php
$con=mysqli_connect("localhost","root","","myDB"
);
if ($con->connect_error) die("Database
connection failed: " .
    $conn->connect_error);
if(mysqli_query($con,"UPDATE Persons SET Age= 50
WHERE FirstName='Nimal'")){
    echo "Record updated successful";
} else {
    echo "Error in executing the SQL" . $con-
>error;
}
mysqli_close($con);
?>
```

This will search for records which have the Firstname as 'Nimal' and change the Age attribute of those records to '50'

Delete Data In a Database Table

- The delete query is used when you need to remove a record from a table.

```
DELETE FROM table_name  
WHERE some_column = some_value
```

```
<?php  
$con=mysqli_connect("localhost","root","123456","bit");  if  
($con->connect_error) die("Database connection failed: " .  
    $conn->connect_error);  
if(mysqli_query($con, "DELETE from Persons WHERE  
FirstName='Nimal'")) {  
    echo "Record delete successful";  
} else {  
    echo "Error in executing the SQL" . $con->error;  
}  
  
mysqli_close($con);  
?>
```

Frameworks & MVC

What is a framework ?

- A software framework is a re-usable design that can be used to build a software system (or subsystem).



A framework for a house

A framework for a house is a structure that engineers use to build a house . Likewise ,a software framework is a structure with core functionalities and data structures that enable developers to build their applications with ease.

Allows developers to develop applications faster

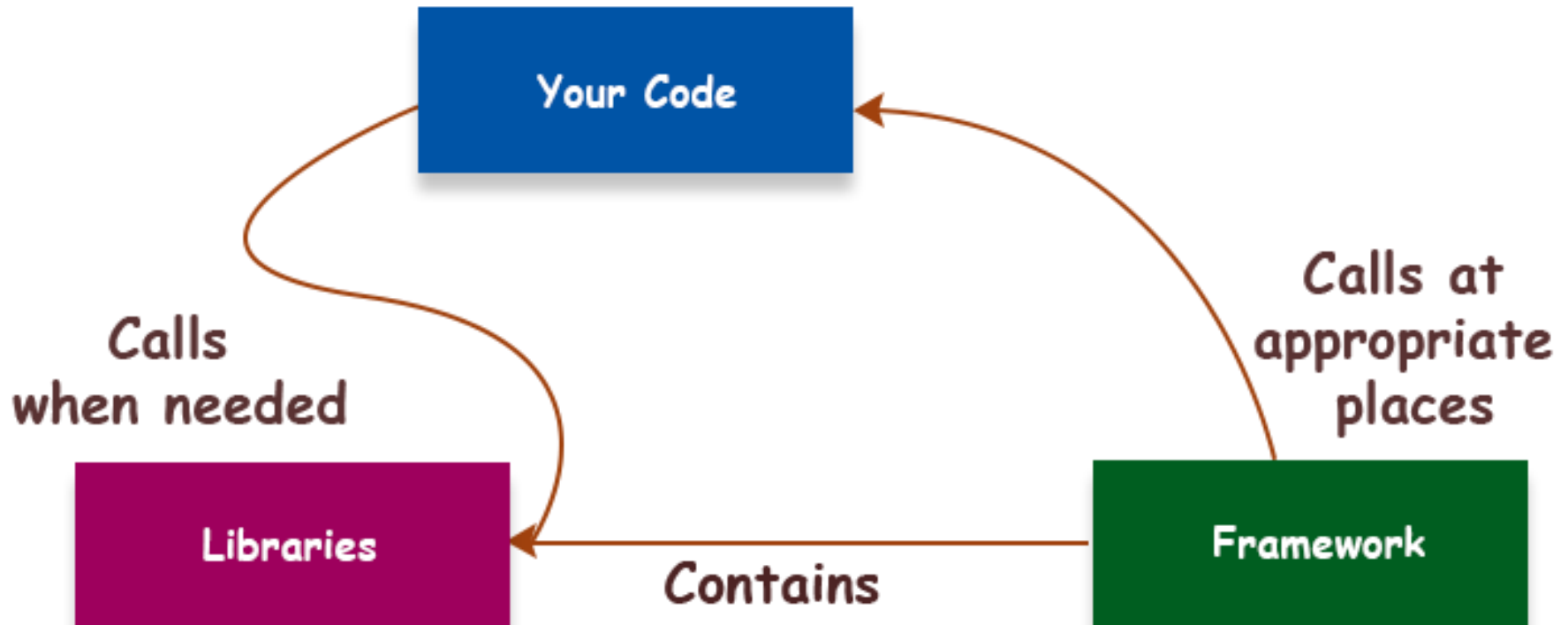
What is a framework ?

- Frameworks support for the development of large-scale software systems. They provide **higher productivity** and **shorter** development time through **design** and **code** reuse.
- Software frameworks include support programs, compilers, code libraries, tool sets, and application programming interfaces (APIs) that bring together all the different components to enable development of a project or solution.

Library vs. Framework

- A **library** performs specific, well-defined operations whereas a **framework** is a skeleton (abstract design) where the application defines what exactly to be done by filling out the skeleton.
- The main objective of a library is the code reuse.
- Typically, in a framework there is a defined control flow with predefined spots that you should fill out with our code. Your inserted code will be called by the framework appropriately.

Library vs. Framework



Why Frameworks ?

- Raw PHP, works very well with small applications. HTML files can be easily extended with dynamic content from the database, form processing, etc.
- But, when applications grow, lots of code repetition occurs across multiple pages.
- Many common tasks will be there for any given web application that may need to redevelop when programming from basic features.
- Its hard for a new developer to work on a code someone else have written.
 - It takes a long time to get familiar with the code.

Model-View-Controller design pattern

- Most common and popular Web application development frameworks are based on the Model-View-Controller(MVC) design pattern.
- A design pattern is a software design best practice derived from experience.
- When the framework provides the building blocks of the proved design pattern the developers can focus on the specific requirements of the project under development.
- Recall in OOP a class is a blue print to generate multiple objects.
- In parallel a design pattern is a design guideline for a given program design problem (i.e.: what is the suitable high-level design for a web application?)

Model-View-Controller design pattern

- When Model-View-Controller pattern is implemented the application is structured to logically separable functions
- For MVC such parts include data-model, presentation aspect and the control flow.
- Typically, application frameworks provide basic building blocks needed by most applications such as
 - Database connections
 - Business logic
 - Form handling

Features of a good framework

- Supports a design pattern.
- Provide libraries, plugins to make application development easier and faster.
- Supports layer of abstraction for database interactions
 - Ability to work with a database without writing queries by SQL language
- A strong community
 - If something goes wrong, a place to get support.

PHP Frameworks

There are many PHP framework. A number of them are listed below

- CakePHP
- Symfony
- CodeIgniter
- Zend Framework
- Yii Framework

PHP application testing and tools

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support.expected_conditions import presence_of_element_located

#This example requires Selenium WebDriver 3.13 or newer
with webdriver.Firefox() as driver:
    wait = WebDriverWait(driver, 10)
    driver.get("https://google.com/ncr")
    driver.find_element(By.NAME, "q").send_keys("cheese" + Keys.RETURN)
    first_result = wait.until(presence_of_element_located((By.CSS_SELECTOR, "h3>div")))
    print(first_result.get_attribute("textContent"))
```

Web Application Testing

- In all areas of software development, including web applications, testing is a crucial step.
- You must ensure the application works correctly before handing it over to your customer.
- That process involves testing. There are generally four levels of testing that you can perform with dynamic web applications. From lowest to highest levels.

Web Application Testing

- **Unit testing:** Perform tests on individual sections of code to ensure that there are no syntax or logic errors. In a web application you can test individual functions/button events and ensure there are no errors or warnings produced for acceptable input.
- **Integration testing:** Perform tests on passing data between different sections of code to ensure that there is no data mismatch. Proper integration testing requires an overall picture of system components(different pages) and the data flow between them. Therefore a picture of the flow should be kept when performing the testing.

Web Application Testing

- **System testing:** Perform tests on the entire website to ensure that all the functionality works correctly and no requirement is missing. System testing requires that you (or a non-developer) walk through the entire application from start to finish. The system testing often refers back to the original feature requirements that were defined by the customer, ensuring that each of the requirements is met by the applications.
- **Acceptance testing:** Customer tests the website to ensure that it works as customer specified and that customer understands the user interface. A checklist for all the requirement features is useful if a user guide for the application is available it can be used as well.

Web Application Testing Automation

- **Automation of Testing:** Rather than manually testing entire application at each update it is convenient to make the testing process mechanical. This is a widely used practice in present software industry.
- To support programmatic/automated testing of web applications in PHP there are frameworks developed. (Frameworks may differ for each programming language)

PHPUnit



Web Application Testing Automation

- **PHPUnit:** PHPUnit is the best-known testing framework for writing Unit Tests for PHP apps. Unit tests take small portions of code called units and test them one by one. Note that programmer has to write code for testing the application code.
- It can be used via the command line, and it provides us with a handy `TestCase` class that we can extend according to our needs. PHPUnit also allows developers to use pre-written assertion methods to assert that the app behaves a certain way.

Web Application Testing Automation

- **Selenium:** Selenium is a sophisticated testing framework that automates browsers. This means it is possible to write User Acceptance Tests that examine the entire app as a whole.
- Selenium is a robust tool that has its own WebDriver API that can drive a browser natively as though a real user would use it either locally or on a remote machine. Selenium is an excellent tool for testing more mature web applications.
- Selenium is not specific to PHP therefore learning it can help in other environments as well.

XSS

data spoofing

SQL injection

Web Application Security

Validation

Sanitizing

Web Application Security

- Security is a very important area of Web application development. The security aspect comparatively more important than the non-web applications. There are reasons for that:
- Web applications are exposed to internet, therefore anyone with a computer and internet connection can reach the application.
- Web applications use a web client that renders the application content for the user in contrast to the direct interface with a desktop application therefore we do not have control over the client.
- Web content reaches the user through a public network in contrast to the locally installed or restricted application.

Web Application Security

Results of application security breaches:

- Sensitive data of both company and customers is exposed
- Loss of trust by the customers
- Direct financial loss due to fraudulent transactions etc.
- High cost of recovering the data, software and even hardware
- Interruption of business

Exploring PHP Vulnerabilities

- Cross-site scripting
- Data spoofing
- Invalid data
- Unauthorized file access

Exploring PHP Vulnerabilities

- **Cross-site scripting** : Cross-site scripting (known as XSS) is quite possibly the most dangerous type of attack made on dynamic web applications. The main idea of an XSS attack is to embed malicious JavaScript code in data that the attacker submits to the web application as part of the normal data input process. When the web application tries to display the data in a client browser, the JavaScript is pushed to the client browser that's viewing the website and runs.

Exploring PHP Vulnerabilities

Cross-site scripting attacks can be in two types:

- **Persistent attack:** The attacker places the rogue script as data that the browser displays when the web page loads. User only has to access the page to be exposed to the malicious code. Attacker can keep a comment for a blog post that runs as victim sees the comment.
- **Reflected attack:** The attacker places the rogue script as a link in the submitted data. Victims must actively click the link to launch the XSS attack. The attacker might send a link through email which need to be clicked by the victim.

Exploring PHP Vulnerabilities

Data spoofing : Data spoofing means externally inserting fraudulent data into a PHP program code. PHP has a setting called **register_globals** in the php.ini configuration file for the PHP server.

- When this setting is enabled, PHP automatically converts any data passed via the GET or POST methods into a PHP variable.
- Attacker can use this feature to create a global variable inside your PHP program by just sending an input parameter with the required name.
- If such a variable is used by the program to take a decision attacker can override the value of that variable making the PHP code vulnerable.

Exploring PHP Vulnerabilities

Invalid data : Invalid data inputs to a web application can be due to two reasons.

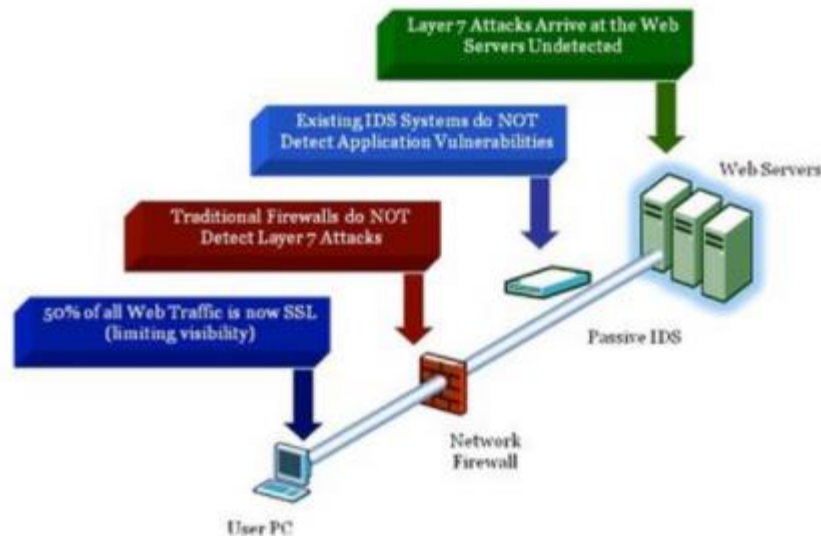
- **Human error of the user:** Often invalid data is just the result of a site visitor not paying close enough attention to the form fields and entering the wrong data into the wrong field, such as typing a zip code into a city name data field.
- **Intentional input by an attacker:** This can vary from as entering an invalid email address into a contact form on purpose to remain anonymous to inserting a data that may reveal a system vulnerability/malfunction i.e. SQL injection.
- The application developer has to anticipate invalid data and try to prevent it before it becomes a problem in the application.

Exploring PHP Vulnerabilities

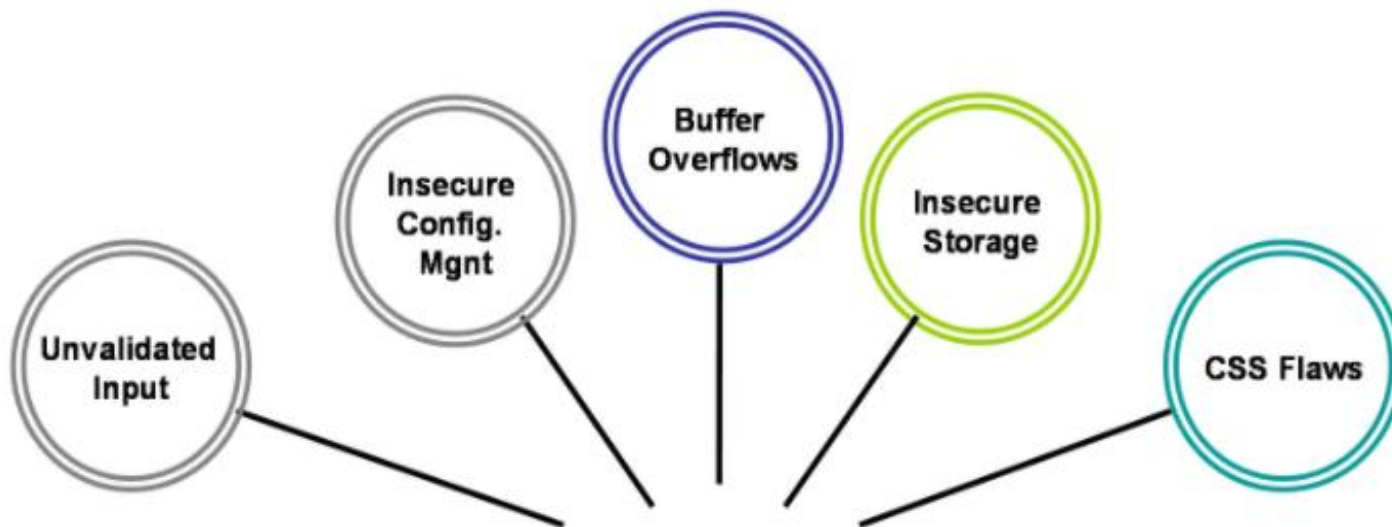
Unauthorized file access : The PHP code in your web applications may contain lots of privileged Information or directions to locate such information i.e.: database user account information.

- Therefore being able to properly protect your PHP files from unauthorized viewing is a must.
- If an attacker tries to access a .php file in the server normally the result will be the processed output of the code but not the source code itself.
- However if an attacker manages to break into the **DocumentRoot** (i.e. htdocs) folder using some attack, your PHP code will be visible giving out more information to attack the system further.

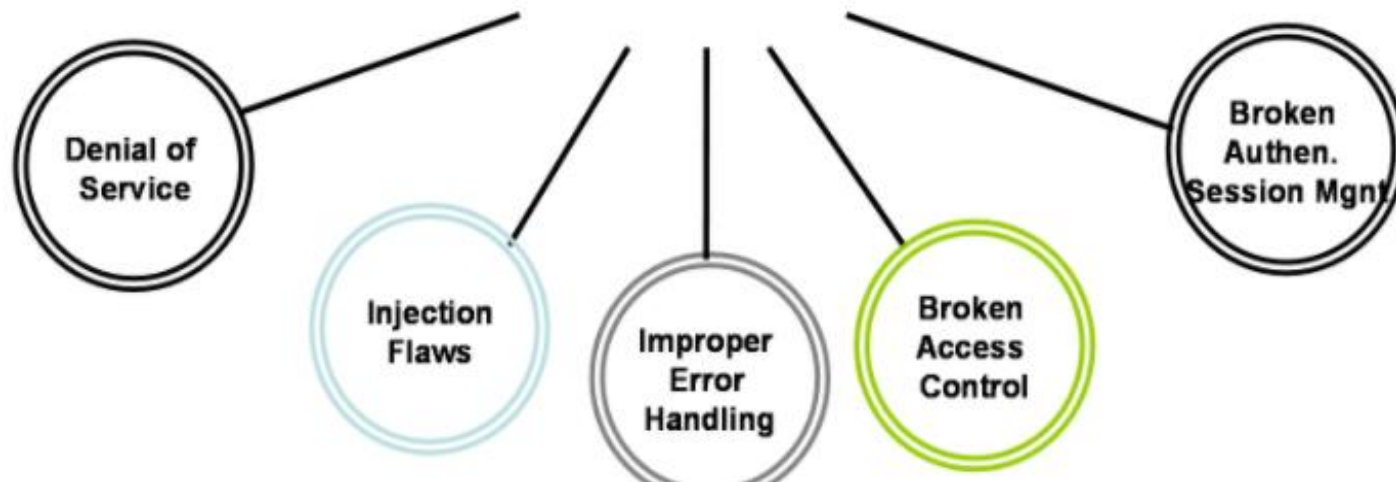
Handling Vulnerability

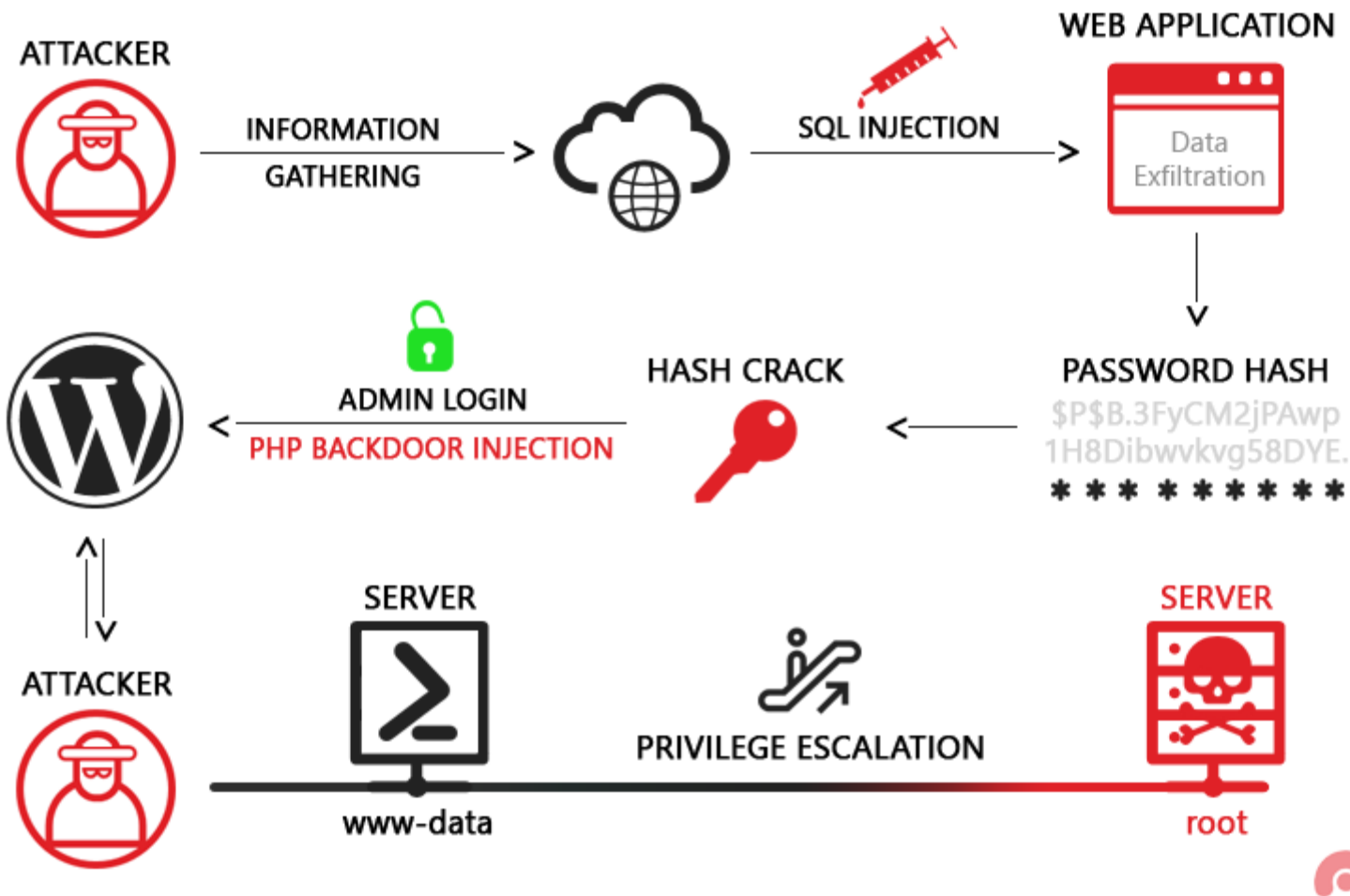


What are vulnerabilities in Web applications?



OWASP Top Ten Most Critical Web Application Security Vulnerabilities





PHP Vulnerability Solutions

Data Sanitizing : Sanitizing data input to PHP code means converting/removing any embedded scripts or HTML content.

- This sanitizing step stops any type of XSS attacks we explained before.
- Two functions in PHP can help the sanitizing:
 1. `htmlspecialchars()`
 2. `filter_var()`
- **`htmlspecialchars()`** : This function detects HTML tags embedded in a data string and converts the greater-than and less-than symbols in the tags to the HTML entity codes `>` and `<`. This doesn't remove the tags from the input but converts them to normal text.
- **`filter_var()`** : The `filter_var()` function provides a host of customized filters for finding and sanitizing different types of data that could potentially cause harm in your PHP application.

PHP Vulnerability Solutions

Data Sanitizing

- htmlspecialchars() function signature is as follows:
htmlspecialchars(string [, flags [,encoding [,double]]])
- By default, the function encodes the following characters: Ampersand (&), Double quote ("), Single quote ('), Less than (<), Greater than (>)
- You can pick and choose which of these items the htmlspecialchars() function converts and which ones it allows through by specifying one or more flags.

PHP Vulnerability Solutions

Some htmlspecialchars() function *flags*:

Flag	Description
ENT_COMPAT	Converts only double quotes.
ENT_QUOTES	Converts both single and double quotes.
ENT_NOQUOTES	Doesn't convert either single or double quotes.
ENT_IGNORE	Doesn't convert anything.
ENT_SUBSTITUTE	Replaces invalid code with Unicode replacement characters instead of returning an empty string.
ENT_XHTML	Handles the code as XHTML.
ENT_HTML5	Handles the code as HTML5.

PHP Vulnerability Solutions

Data Sanitizing

- `filter_var()` function signature is as follows: *`filter_var(string [, filter] [, flags])`*
- The **filter** and **flags** parameters are optional, but in most cases you'll at least specify the filter to use. The filter defines what class of characters the `filter_var()` function should look for, and the flags parameter fine-tunes subsets of characters within the filter class.

PHP Vulnerability Solutions

Some `filter_var()` function *filter* options for sanitizing

Option	Description
<code>FILTER_SANITIZE_EMAIL</code>	Removes invalid characters from an email address.
<code>FILTER_SANITIZE_ENCODED</code>	Encodes a string to make a valid URL.
<code>FILTER_SANITIZE_MAGIC_QUOTES</code>	Escapes embedded quotes.
<code>FILTER_SANITIZE_NUMBER_FLOAT</code>	Removes all characters except digits and float symbols.
<code>FILTER_SANITIZE_NUMBER_INT</code>	Removes all characters except digits and integer symbols.
<code>FILTER_SANITIZE_SPECIAL_CHARS</code>	Removes quotes, as well as greater-than, less-than, and ampersand characters.
<code>FILTER_SANITIZE_STRING</code>	Removes all HTML5 tags.
<code>FILTER_SANITIZE_URL</code>	Removes all invalid URL characters.

PHP Vulnerability Solutions

- **Validating data** : Detecting all types of invalid data can be impossible, but PHP provides a few ways for you to at least detect some types of invalid data to help make things at least a little bit easier.
- **Validating data types** : One primary goal for catching invalid data is to at least determine that the input data is the correct data type. PHP provides a series of functions to do that.

PHP Vulnerability Solutions

List of validating functions available in PHP

Function	Description
is_bool()	Returns TRUE if the value is a Boolean data type.
is_float()	Returns TRUE if the value is in valid float format.
is_int()	Returns TRUE if the value is an integer value.
is_null()	Returns TRUE if the value is NULL.
is_numeric()	Returns TRUE if the value is in a valid numeric format.
is_string()	Returns TRUE if the value is a string as opposed to a number.

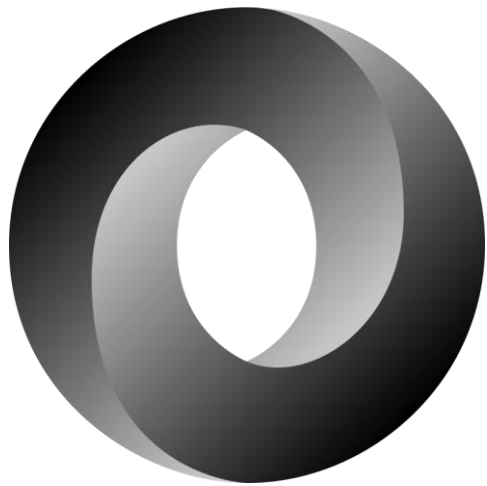
PHP Vulnerability Solutions

- **Validating data format** : Testing for valid data types is fine when you're working with numeric values, but it doesn't help all that much for text values such as names, home addresses, and email addresses.
- The `is_string()` function can tell you that the value is a valid string value, but not the format of the data contained within the string.
- PHP `filter_var()` function can be used for this purpose too (other than sanitizing).

PHP Vulnerability Solutions

filter_var() function *options* for data validating

Option	Description
FILTER_VALIDATE_BOOLEAN	Returns TRUE if the value is a valid Boolean value.
FILTER_VALIDATE_EMAIL	Returns TRUE if the value is in a valid email address format.
FILTER_VALIDATE_FLOAT	Returns TRUE if the value is in a valid float format.
FILTER_VALIDATE_INT	Returns TRUE if the value is in a valid integer format.
FILTER_VALIDATE_IP	Returns TRUE if the value is in a valid IP address format.
FILTER_VALIDATE_MAC	Returns TRUE if the value is in a valid MAC address format.
FILTER_VALIDATE_REGEXP	Returns TRUE if the value matches the specified regular expression.
FILTER_VALIDATE_URL	Returns TRUE if the value is in a valid URL format.



<xml />

REST

SOAP

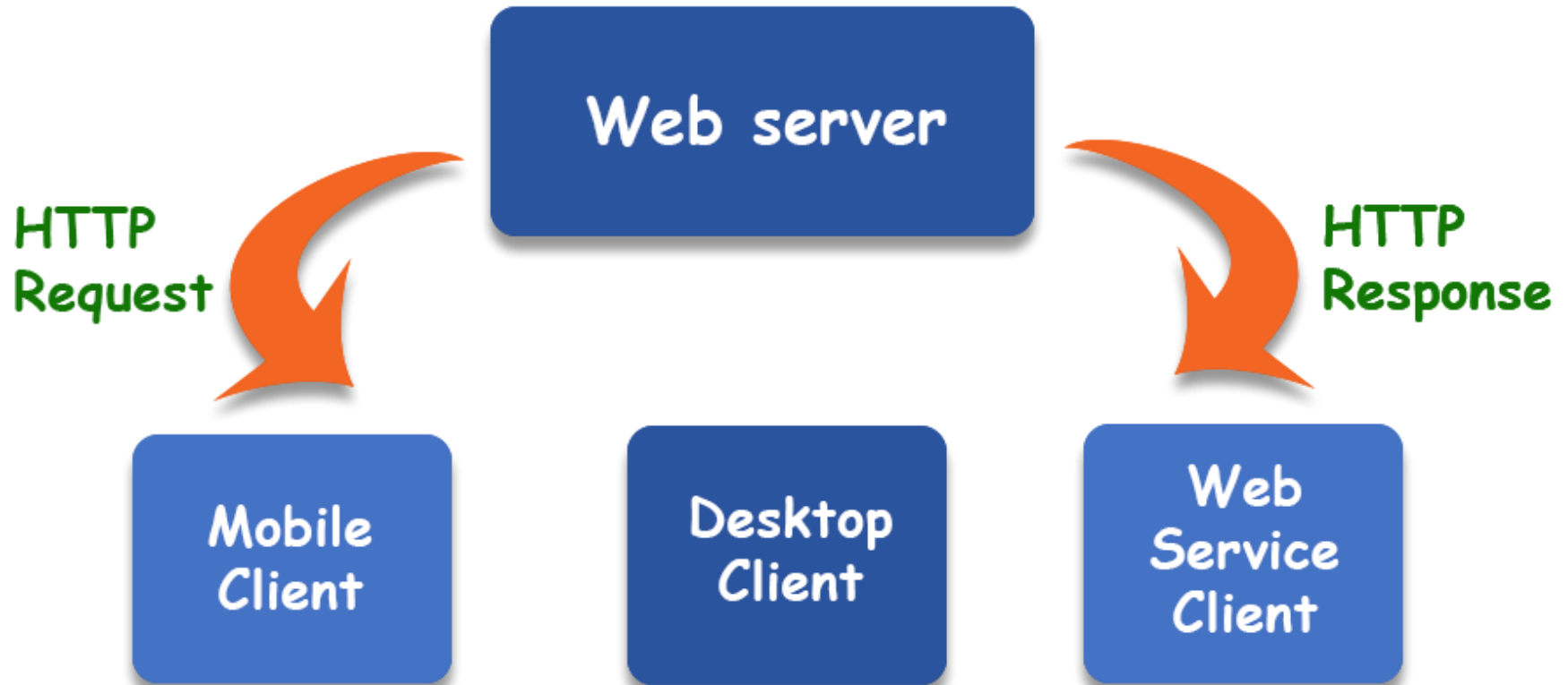
Web Services

Web Services

“A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards.”

— World Wide Web Consortium, Web Services Glossary

Web Services



PHP Web services

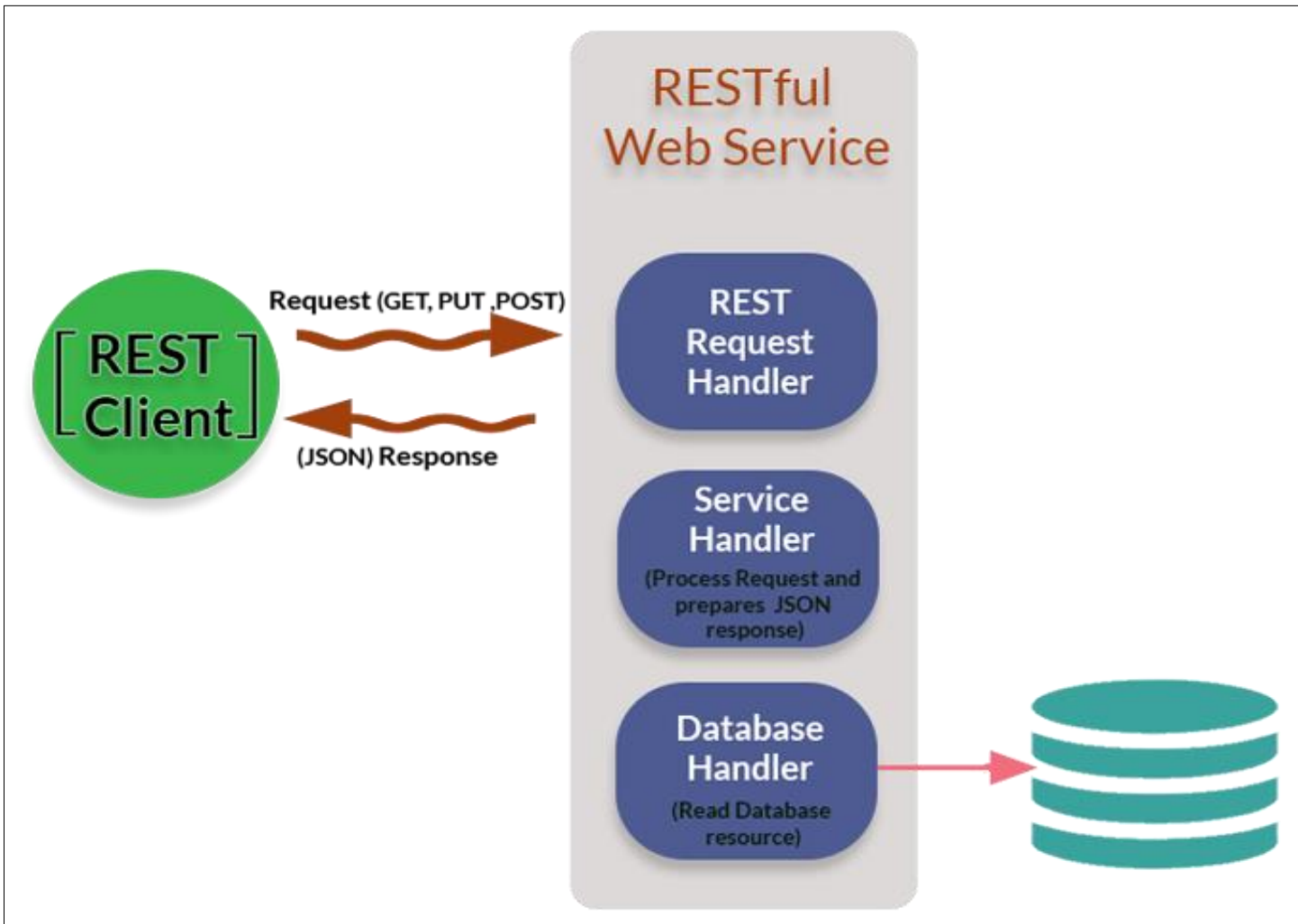
- Web service is a remote service that allows clients to use HTTP protocol to utilize APIs hosted remotely over a network.
- There are different standards to implement web services such as SOAP and REST.
- We will use PHP to implement a REST based web service.
- The web service is technologically same as a web application used by a human but different from it as the usage is to another program rather than a human.
- Web service can be consumed by any client including another web service.

PHP Web services

- **REST** or **Representational State Transfer** is one of the popular architectural style used to develop web services.
- The objective is to build a RESTful web service in PHP to provide resource data based on the request with the network call by the external clients. The steps to create web service:
 1. Create request URI with patterns that follow REST principles.
 2. Make the RESTful service to be capable of responding to the requests in JSON/ XML or HTML formats.
 3. Demonstrate the use of HTTP Status code based on different scenarios.
 4. Demonstrate the use of Request Headers.
 5. Test the RESTful web service using a REST client.

PHP Web Services

- Basic architecture of a RESTful web service:



PHP Web services

- We will create a table and implement HTTP based web services to perform CRUD operations on the data in that table.
- The communication with the browser happens through HTTP protocol similar to viewing a web page in world wide web.
- The message format used to communicate is JSON (Java Script Object Notation).
- Both server and client has to agree what types of messages are supported and their meaning.

PHP Web services

- We create a new databased and a table to keep the data for our web service.
- Using phpMyAdmin we execute the script to make the necessary changes to the database.

```
1
2  -- Database: `rest_web`
3  create database IF NOT EXISTS `rest_web`
4
5
6  -- Table structure for table `user`
7  CREATE TABLE IF NOT EXISTS `user` (
8    `ID` int(11) NOT NULL AUTO_INCREMENT,
9    `name` text NOT NULL,
10   `email` varchar(100) NOT NULL,
11   `password` varchar(100) NOT NULL,
12   `status` text NOT NULL,
13   PRIMARY KEY (`ID`)
14 ) AUTO_INCREMENT=1 ;
15
```


Activity: MySQL

- We create the **database** “rest_web” and **table** “user” as given in the following script.
- Manually insert an entry to the table with your information.
- Notice that ID column is automatically updated.

```
1
2  -- Database: `rest_web`
3  create database IF NOT EXISTS `rest_web`
4
5
6  -- Table structure for table `user`
7  CREATE TABLE IF NOT EXISTS `user` (
8    `ID` int(11) NOT NULL AUTO_INCREMENT,
9    `name` text NOT NULL,
10   `email` varchar(100) NOT NULL,
11   `password` varchar(100) NOT NULL,
12   `status` text NOT NULL,
13   PRIMARY KEY (`ID`)
14 ) AUTO_INCREMENT=1 ;
15
```

PHP Web services

- To connect to the database we have to repeat some code which can be in a reusable php script.
- This file is `connect.php`. We use the technique if including this file in other places.
- Note : you may create new credentials(*username/password*) or use existing ones.

```
1  <?php
2  $servername = "localhost";
3  $username = "username";
4  $password = "password";
5  $dbname = "rest_web";
6
7  // Create connection
8  $conn = new mysqli($servername, $username, $password,
9  $dbname);
10
11 // Check connection
12 if ($conn->connect_error) {
13     die("Database connection failed: " . $conn->connect_error);
14     echo "Error connecting to database";
15 }
16 ?>
```

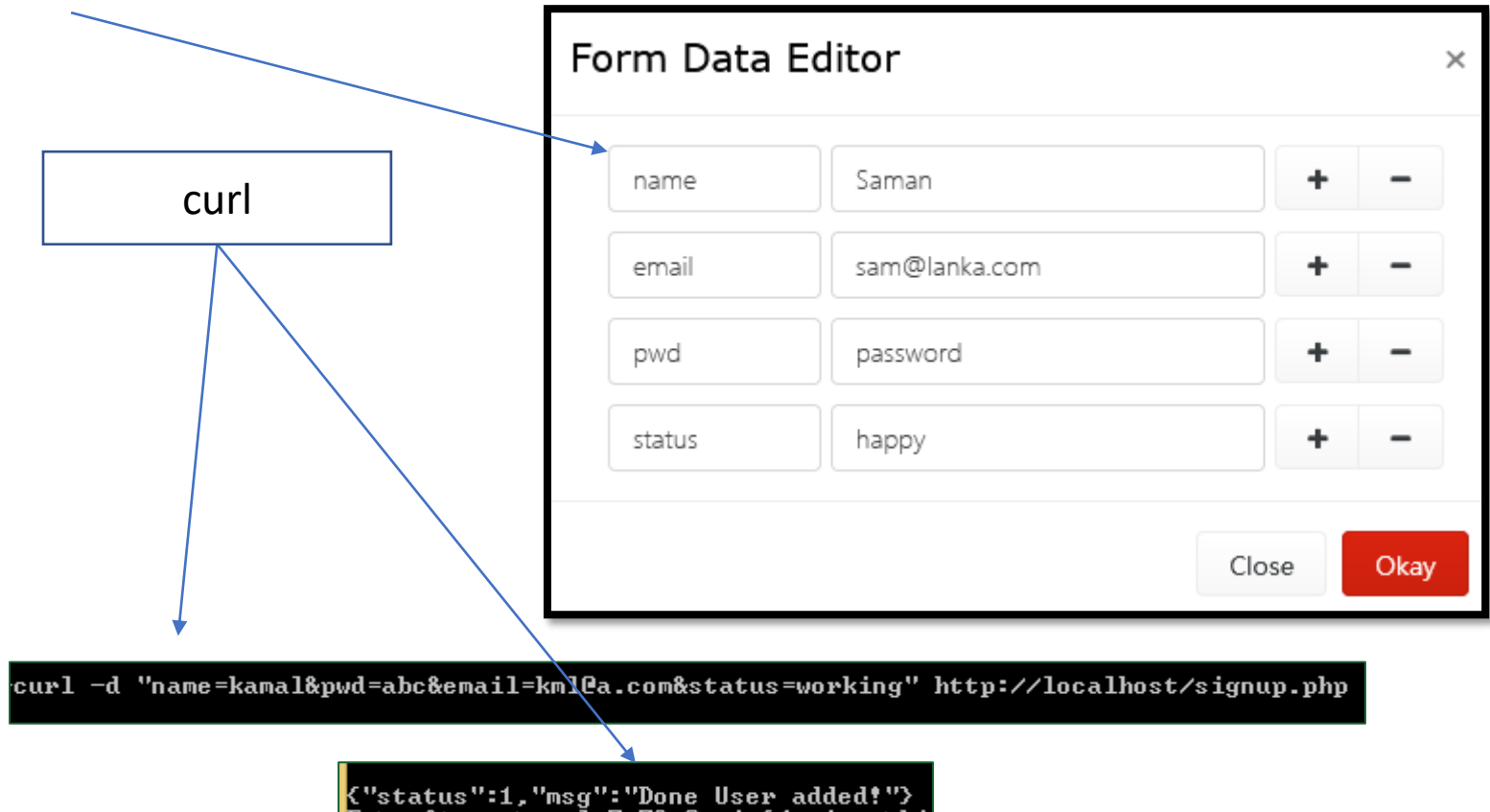
PHP Web services

- Our REST API in **signup.php** is for inserting a user **includes** the **connect.php**.
- It accesses the request parameters and inserts to database.
- Returns a JSON response to the caller
- We can see the response in client

```
1 <?php
2 // include connect.php
3 include_once('connect.php');
4 if($_SERVER['REQUEST_METHOD'] == "POST"){
5     // Get data
6     $name = isset($_POST["name"]) ? ($_POST["name"]) : "";
7     $email = isset($_POST["email"]) ? ($_POST["email"]) : "";
8     $password = isset($_POST['pwd']) ? ($_POST['pwd']) : "";
9     $status = isset($_POST['status']) ? ($_POST['status']) : "";
10
11     // Insert data into data base
12     $sql = "INSERT INTO `rest_web`.`user` (`name`,`email`,`password`,`status`)VALUES('$name','$email','$password','$status')";
13     $qur = mysqli_query($conn,$sql);
14     if($qur){
15         $json = array("status" => 1, "msg" => "Done User added!");
16     }else{
17         $json = array("status" => 0, "msg" => "Error adding user!");
18     }
19 }else{
20     $json = array("status" => 0, "msg" => "Request method not accepted");
21 }
22 // if the connection object exists
23 if (!$conn->connect_error){
24     $conn->close();
25 }
26 /* Output header */
27 header('Content-type: application/json');
28 echo json_encode($json);
29
30
31 ?>
```

PHP Web services

- To call the web service use either curl program or a browser based client



PHP Web services

Mozilla
add-on
based
REST
client



PHP Web services

- This REST API uses HTTP GET method to retrieve data from the database.
- The connection and response methods are same as before. (Not shown in the code)

```
$uid = isset($_GET['uid']) ? htmlspecialchars($_GET['uid']) : "";  
if(!empty($uid)){  
    $qur = mysqli_query($conn,"select name, email, status from `user` where ID='$uid'");  
    $result = array();  
    if(!$qur){  
        $json = array("status" => 0, "msg" => "User ID not define ".$uid);  
    }else{  
        while($row = mysqli_fetch_array($qur)){  
            $result[] = array("name" => $row['name'], "email" => $row['email'], 'status' => $row['status']);  
        }  
        $json = array("status" => 1, "info" => $result);  
    }  
}else{  
    $json = array("status" => 0, "msg" => "User ID not define ".$uid);  
}
```

Activity: PHP Web services


- Using the repeating code for database access and JSON output generation complete the below PHP fragment. (We can name this `view.php`)

```
$uid = isset($_GET['uid']) ? htmlspecialchars($_GET['uid']) : "";
if(!empty($uid)){
    $qur = mysqli_query($conn,"select name, email, status from `user` where ID='$uid'");
    $result = array();
    if(!$qur){
        $json = array("status" => 0, "msg" => "User ID not define ".$uid);
    }else{
        while($row = mysqli_fetch_array($qur)){
            $result[] = array("name" => $row['name'], "email" => $row['email'], 'status' => $row['status']);
        }
        $json = array("status" => 1, "info" => $result);
    }
}else{
    $json = array("status" => 0, "msg" => "User ID not define ".$uid);
}
```

PHP Web services

- Following shows the GET API call request and response in the browser based client.
- GET API maps to the **read** operation on data.

[-] Request

Method	GET	▼	URL		http://localhost/view.php?uid=2
--------	-----	---	-----	---	---------------------------------

[-] Response

Headers Response Preview

```
1 {"status":1,"info":[{"name":"Saman","email":"saman@lanka.com","status":"happy"}]}
```


PHP Web services

- Following shows the PUT API call processing in PHP.
- This API does an **update** on our table.

```
5 ▼ if($_SERVER['REQUEST_METHOD'] == "PUT"){
6     $params = Array();
7     parse_str(file_get_contents('php://input'), $params);
8     $GLOBALS["_PUT"] = $params;
9     // Add these request vars into _REQUEST, mimicing default behavior,
10    //PUT/DELETE will override existing COOKIE/GET vars
11    $_REQUEST = $params + $_REQUEST;
12
13    $uid = isset($_PUT['uid']) ? htmlspecialchars($_PUT['uid']) : "";
14    $status = isset($_PUT['status']) ? htmlspecialchars($_PUT['status']) : "";
15    // Add your validations
16 ▼    if(!empty($uid)){
17        $qur = mysqli_query($conn,
18            "UPDATE `rest_web`.`user` SET `status`='{$status}' WHERE `user`.`ID`='{$uid}'");
19 ▼        if($qur){
20            $json = array("status" => 1, "msg" => "Status updated!!.");
21 ▼        }else{
22            $json = array("status" => 0, "msg" => "Error updating status");
23        }
24 ▼    }else{
25        $json = array("status" => 0, "msg" => "User ID not define 1");
26    }
27 ▼ }else{
28     $json = array("status" => 0, "msg" => "User ID not define 2");
29 }
30
```

PHP Web services

- Notice that we have used some preprocessing code to use HTTP PUT as PHP does not support direct accessing the it similar to GET and POST.

```
5 ▼ if($_SERVER['REQUEST_METHOD'] == "PUT"){
6     $params = Array();
7     parse_str(file_get_contents('php://input'), $params);
8     $GLOBALS["_PUT"] = $params;
9     // Add these request vars into _REQUEST, mimicing default behavior,
10    //PUT/DELETE will override existing COOKIE/GET vars
11    $_REQUEST = $params + $_REQUEST;
12
13    $uid = isset($_PUT['uid']) ? htmlspecialchars($_PUT['uid']) : "";
14    $status = isset($_PUT['status']) ? htmlspecialchars($_PUT['status']) : "";
15    // Add your validations
16 ▼    if(!empty($uid)){
17        $qur = mysqli_query($conn,
18            "UPDATE `rest_web`.`user` SET `status`='{$status}' WHERE `user`.`ID`='{$uid}'");
19 ▼        if($qur){
20            $json = array("status" => 1, "msg" => "Status updated!!.");
21 ▼        }else{
22            $json = array("status" => 0, "msg" => "Error updating status");
23        }
24 ▼    }else{
25        $json = array("status" => 0, "msg" => "User ID not define 1");
26    }
27 ▼ }else{
28     $json = array("status" => 0, "msg" => "User ID not define 2");
29 }
30
```

Activity: PHP Web services


- Complete the below code fragment with the same approach described before and name it as **edit.php**. Access the edit API from client and test the result in the “**rest_web**” table.

```
5 ▼ if($_SERVER['REQUEST_METHOD'] == "PUT"){
6     $params = Array();
7     parse_str(file_get_contents('php://input'), $params);
8     $GLOBALS["_PUT"] = $params;
9     // Add these request vars into _REQUEST, mimicing default behavior,
10    //PUT/DELETE will override existing COOKIE/GET vars
11    $_REQUEST = $params + $_REQUEST;
12
13    $uid = isset($_PUT['uid']) ? htmlspecialchars($_PUT['uid']) : "";
14    $status = isset($_PUT['status']) ? htmlspecialchars($_PUT['status']) : "";
15    // Add your validations
16 ▼    if(!empty($uid)){
17        $qur = mysqli_query($conn,
18            "UPDATE `rest_web`.`user` SET `status`='$status' WHERE `user`.`ID` ='$uid'");
19 ▼        if($qur){
20            $json = array("status" => 1, "msg" => "Status updated!!.");
21 ▼        }else{
22            $json = array("status" => 0, "msg" => "Error updating status");
23        }
24 ▼    }else{
25        $json = array("status" => 0, "msg" => "User ID not define 1");
26    }
27 ▼ }else{
28     $json = array("status" => 0, "msg" => "User ID not define 2");
29 }
30
```

PHP Web services

- Following shows the PUT API call and response in the client program.

[-] Request

Method URL 

Body

```
uid=2&status=updated
```

[-] Response

Headers Preview

```
1 {"status":1,"msg":"Status updated!!."}
```

Additional Information on PHP Web Service

- Curl client program : <https://curl.se/windows/>
- Mozilla REST client :
<https://addons.mozilla.org/en-US/firefox/addon/restclient/>
- JSON processing in PHP :
https://www.w3schools.com/js/js_json_php.asp
- PHP Global variables :
https://www.w3schools.com/php/php_superglobals.asp