Unit 4

What is PHP

PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side. PHP is well suited for web development. Therefore, it is used to develop web applications (an application that executes on the server and generates the dynamic page.).

PHP was created by **Rasmus Lerdorf in 1994** but appeared in the market in 1995. **PHP 7.4.0** is the latest version of PHP, which was released on **28 November**. Some important points need to be noticed about PHP are as followed:

- PHP stands for Hypertext Preprocessor.
- PHP is an interpreted language, i.e., there is no need for compilation.
- PHP is faster than other scripting languages, for example, ASP and JSP.
- PHP is a server-side scripting language, which is used to manage the dynamic content of the website.
- PHP can be embedded into HTML.
- PHP is an object-oriented language.
- PHP is an open-source scripting language.
- PHP is simple and easy to learn language.

Why use PHP

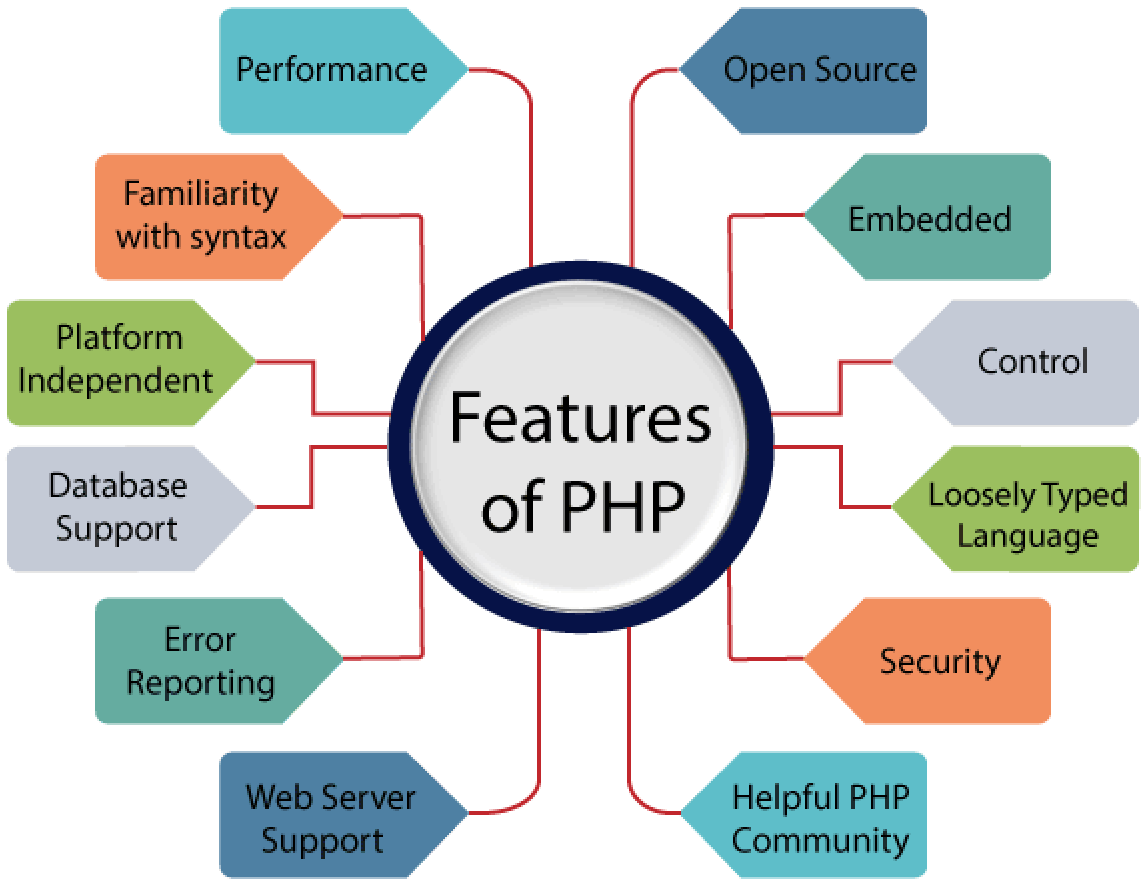
PHP is a server-side scripting language, which is used to design the dynamic web applications with MySQL database.

- It handles dynamic content, database as well as session tracking for the website.
- You can create sessions in PHP.

- It can access cookies variable and also set cookies.
- It helps to encrypt the data and apply validation.
- PHP supports several protocols such as HTTP, POP3, SNMP, LDAP, IMAP, and many more.
- Using PHP language, you can control the user to access some pages of your website.
- As PHP is easy to install and set up, this is the main reason why PHP is the best language to learn.
- PHP can handle the forms, such as - collect the data from users using forms, save it into the database, and return useful information to the user. **For example** - Registration form.

PHP Features

PHP is very popular language because of its simplicity and open source. There are some important features of PHP given below:



Performance:

PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP. PHP uses its own memory, so the server workload and loading time is automatically reduced, which results in faster processing speed and better performance.

Open Source:

PHP source code and software are freely available on the web. You can develop all the versions of PHP according to your requirement without paying any cost. All its components are free to download and use.

Familiarity with syntax:

PHP has easily understandable syntax. Programmers are comfortable coding with it.

How to run PHP code in XAMPP

Generally, a PHP file contains HTML tags and some PHP scripting code. It is very easy to create a simple PHP example. To do so, create a file and write HTML tags + PHP code and save this file with .php extension.

Note: PHP statements ends with semicolon (;).

All PHP code goes between the php tag. It starts with <?php and ends with ?>. The **Syntax of PHP tag is given below:**

```
<?php
//your code here
?>
```

Let's see a simple PHP example where we are writing some text using PHP echo command.

File: first.php

```
<!DOCTYPE>
<html>
<body>
<?php
echo "<h2>Hello First PHP</h2>";
```

```
?>
</body>
</html>
```

Output:

```
Hello First PHP
```

PHP Echo

PHP echo is a language construct, not a function. Therefore, you don't need to use parenthesis with it. But if you want to use more than one parameter, it is required to use parenthesis.

The syntax of PHP echo is given below:

```
void echo ( string $arg1 [, string $... ] )
```

PHP echo statement can be used to print the string, multi-line strings, escaping characters, variable, array, etc. Some important points that you must know about the echo statement are:

- echo is a statement, which is used to display the output.
- echo can be used with or without parentheses: echo(), and echo.
- echo does not return any value.
- We can pass multiple strings separated by a comma (,) in echo.
- echo is faster than the print statement.

PHP echo: printing string

File: echo1.php

```
<?php
echo "Hello by PHP echo";
?>
```

Output:

```
Hello by PHP echo
```

PHP echo: printing multi line string

File: echo2.php

```
<?php
echo "Hello by PHP echo
this is multi line
text printed by
PHP echo statement  ";  ?>
```

Output:

```
Hello by PHP echo this is multi line text printed by PHP echo statement
```

PHP echo: printing escaping characters

File: echo3.php

```php
<?php
echo "Hello escape \"sequence\" characters";
?>
```

Output:

```
Hello escape "sequence" characters
```

PHP echo: printing variable value

File: echo4.php

```php
<?php
$msg="Hello JavaTpoint PHP";
echo "Message is: $msg";
?>
```

Output:

```
Message is: Hello JavaTpoint PHP
```

PHP Print

Like PHP echo, PHP print is a language construct, so you don't need to use parenthesis with the argument list. Print statement can be used with or without parentheses: print and print(). Unlike echo, it always returns 1.

The syntax of PHP print is given below:

```
int print(string $arg)
```

PHP print statement can be used to print the string, multi-line strings, escaping characters, variable, array, etc. Some important points that you must know about the echo statement are:

- print is a statement, used as an alternative to echo at many times to display the output.
- print can be used with or without parentheses.
- print always returns an integer value, which is 1.
- Using print, we cannot pass multiple arguments.
- print is slower than the echo statement.

PHP print: printing string

File: print1.php

```php
<?php
print "Hello by PHP print ";
print ("Hello by PHP print()");
?>
```

Output:

```
Hello by PHP print Hello by PHP print()
```

PHP print: printing multi line string

File: print2.php

```php
<?php
print "Hello by PHP print
this is multi line
text printed by
PHP print statement ";
?>
```

Output:

Hello by PHP print this is multi line text printed by PHP print statement

PHP print: printing escaping characters

File: print3.php

```
<?php
print "Hello escape \"sequence\" characters by PHP print";
?>
```

Output:

Hello escape "sequence" characters by PHP print

PHP print: printing variable value

File: print4.php

```
<?php
$msg="Hello print() in PHP";
print "Message is: $msg";
?>
```

Output:

Message is: Hello print() in PHP

PHP data types are used to hold different types of data or values. PHP supports 8 primitive data types that can be categorized further in 3 types:

1. Scalar Types (predefined)
2. Compound Types (user-defined)
3. Special Types

PHP Data Types: Scalar Types

It holds only single value. There are 4 scalar data types in PHP.

1. boolean
2. integer

3. float
4. string

PHP Data Types: Compound Types

It can hold multiple values. There are 2 compound data types in PHP.

1. array
2. object

PHP Data Types: Special Types

There are 2 special data types in PHP.

1. resource
2. NULL

PHP Boolean

Booleans are the simplest data type works like switch. It holds only two values: **TRUE (1)** or **FALSE (0)**. It is often used with conditional statements. If the condition is correct, it returns TRUE otherwise FALSE.

Example:

```
<?php
   if (TRUE)
      echo "This condition is TRUE.";
   if (FALSE)
      echo "This condition is FALSE.";
?>
```

Output:

```
This condition is TRUE.
```

PHP Array Functions

PHP provides various array functions to access and manipulate the elements of array. The important PHP array functions are given below.

1) PHP array() function

PHP array() function creates and returns an array. It allows you to create indexed, associative and multidimensional arrays.

Syntax

```
array array ([ mixed $... ] )
```

Example

```
<?php
$season=array("summer","winter","spring","autumn");
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
?>
```

Output:

```
Season are: summer, winter, spring and autumn
```

2) PHP array_change_key_case() function

PHP array_change_key_case() function changes the case of all key of an array.

Note: It changes case of key only.

Syntax

```
array array_change_key_case ( array $array [, int $case = CASE_LOWER ] )
```

Example

```
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
```

```php
print_r(array_change_key_case($salary,CASE_UPPER));
?>
```

Output:

```
Array ( [SONOO] => 550000 [VIMAL] => 250000 [RATAN] => 200000 )
```

Example

```php
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_change_key_case($salary,CASE_LOWER));
?>
```

Output:

```
Array ( [sonoo] => 550000 [vimal] => 250000 [ratan] => 200000 )
```

3) PHP array_chunk() function

PHP array_chunk() function splits array into chunks. By using array_chunk() method, you can divide array into many parts.

Syntax

```php
array array_chunk ( array $array , int $size [, bool $preserve_keys = false ] )
```

Example

```php
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_chunk($salary,2));
?>
```

Output:

```
Array (
[0] => Array ( [0] => 550000 [1] => 250000 )
[1] => Array ( [0] => 200000 )
)
```

4) PHP count() function

PHP count() function counts all elements in an array.

Syntax

```
int count ( mixed $array_or_countable [, int $mode = COUNT_NORMAL ] )
```

Example

```
<?php
$season=array("summer","winter","spring","autumn");
echo count($season);

?>
```

Output:

```
4
```

5) PHP sort() function

PHP sort() function sorts all the elements in an array.

Syntax

```
bool sort ( array &$array [, int $sort_flags = SORT_REGULAR ] )
```

Example

```
<?php
```

```
$season=array("summer","winter","spring","autumn");
sort($season);
foreach( $season as $s )
{
  echo "$s<br />";
}
?>
```

Output:

```
autumn
spring
summer
winter
```

6) PHP array_reverse() function

PHP array_reverse() function returns an array containing elements in reversed order.

Syntax

```
array array_reverse ( array $array [, bool $preserve_keys = false ] )
```

Example

```
<?php
$season=array("summer","winter","spring","autumn");
$reverseseason=array_reverse($season);
foreach( $reverseseason as $s )
{
  echo "$s<br />";
}
```

```
?>
```

Output:

```
autumn
spring
winter
summer
```

7) PHP array_search() function

PHP array_search() function searches the specified value in an array. It returns key if search is successful.

Syntax

```
mixed array_search ( mixed $needle , array $haystack [, bool $strict = false ] )
```

Example

```
<?php
$season=array("summer","winter","spring","autumn");
$key=array_search("spring",$season);
echo $key;
?>
```

Output:

```
2
```

8) PHP array_intersect() function

PHP array_intersect() function returns the intersection of two array. In other words, it returns the matching elements of two array.

Syntax

```
array array_intersect ( array $array1 , array $array2 [, array $... ] )
```

Example

```
<?php
$name1=array("sonoo","john","vivek","smith");
$name2=array("umesh","sonoo","kartik","smith");
$name3=array_intersect($name1,$name2);
foreach( $name3 as $n )
{
  echo "$n<br />";
}
?>
```

Output:

```
sonoo
smith
```

PHP String Functions

PHP provides various string functions to access and manipulate strings.

A list of PHP string functions are given below.

addcslashes()	It is used to return a string with backslashes.
addslashes()	It is used to return a string with backslashes.

bin2hex()	It is used to converts a string of ASCII characters to hexadecimal values.
chop()	It removes whitespace or other characters from the right end of a string
chr()	It is used to return a character from a specified ASCII value.
chunk_split()	It is used to split a string into a series of smaller parts.
convert_cyr_string()	It is used to convert a string from one Cyrillic character-set to another.
convert_uudecode()	It is used to decode a uuencoded string.
convert_uuencode()	It is used to encode a string using the uuencode algorithm.
count_chars()	It is used to return information about characters used in a string.
crc32()	It is used to calculate a 32-bit CRC for a string.
crypt()	It is used to create hashing string One-way.
echo()	It is used for output one or more strings.
explode()	It is used to break a string into an array.
fprint()	It is used to write a formatted string to a stream.

get_html_translation_table()	Returns translation table which is used by htmlspecialchars() and htmlentities().
hebrev()	It is used to convert Hebrew text to visual text.
hebrevc()	It is used to convert Hebrew text to visual text and new lines (\n) into .
hex2bin()	It is used to convert string of hexadecimal values to ASCII characters.
htmlentities()	It is used to convert character to HTML entities.
html_entity_decode()	It is used to convert HTML entities to characters.
htmlspecialchars()	Converts the special characters to html entities.
htmlspecialchars_decode()	Converts the html entities back to special characters.
Implode()	It is used to return a string from the elements of an array.
Join()	It is the Alias of implode() function.
Levenshtein()	It is used to return the Levenshtein distance between two strings.

Lcfirst()	It is used to convert the first character of a string to lowercase.
localeconv()	Get numeric formatting information
ltrim()	It is used to remove whitespace from the left side of a string.
md5()	It is used to calculate the MD5 hash of a string.
md5_files()	It is used to calculate MD5 hash of a file.
metaphone()	It is used to calculate the metaphone key of a string.
money_format()	It is used to return a string formatted as a currency string.
nl2br()	It is used to insert HTML line breaks in front of each newline in a string.
nl_langinfo()	Query language and locale information
number_format()	It is used to format a number with grouped thousands.
ord()	It is used to return ASCII value of the first character of a string.
parse_str()	It is used to parse a query string into variables.
print()	It is used for output one or more strings.

printf()	It is used to show output as a formatted string.
quoted_printable_decode()	Converts quoted-printable string to an 8-bit string
quoted_printable_encode()	Converts the 8-bit string back to quoted-printable string
quotemeta()	Quote meta characters
rtrim()	It is used to remove whitespace from the right side of a string.
setlocale()	It is used to set locale information.
sha1()	It is used to return the SHA-1 hash of a string.
sha1_file()	It is used to return the SHA-1 hash of a file.
similar_text()	It is used to compare the similarity between two strings.
Soundex()	It is is used to calculate the soundex key of a string.
sprintf()	Return a formatted string
sscanf()	It is used to parse input from a string according to a format.
strcasecmp()	It is used to compare two strings.

strchr()	It is used to find the first occurrence of a string inside another string.
strcmp()	Binary safe string comparison (case-sensitive)
strcoll()	Locale based binary comparison(case-sensitive)
strcspn()	It is used to reverses a string.
stripcslashes()	It is used to unquote a string quoted with addcslashes().
stripos()	It is used to return the position of the first occurrence of a string inside another string.
stristr()	Case-insensitive strstr
strlen()	It is used to return the length of a string.
strncasecmp()	Binary safe case-insensitive string comparison
strnatcasecmp()	It is used for case-insensitive comparison of two strings using a "natural order" algorithm
strnatcmp()	It is used for case-sensitive comparison of two strings using a "natural order" algorithm
strncmp()	It is used to compare of the first n characters.
strpbrk()	It is used to search a string for any of a set of characters.

strripos()	It finds the position of the last occurrence of a case-insensitive substring in a string.
strrpos()	It finds the length of the last occurrence of a substring in a string.
strpos()	It is used to return the position of the first occurrence of a string inside another string.
strrchr()	It is used to find the last occurrence of a string inside another string.
strrev()	It is used to reverse a string.
strspn()	Find the initial length of the initial segment of the string
strstr()	Find the occurrence of a string.
strtok()	Splits the string into smaller strings
strtolower()	Convert the string in lowercase
strtoupper()	Convert the strings in uppercase
strtr()	Translate certain characters in a string or replace the substring
str_getcsv()	It is used to parse a CSV string into an array.

str_ireplace()	It is used to replace some characters in a string (case-insensitive).
str_pad()	It is used to pad a string to a new length.
str_repeat()	It is used to repeat a string a specified number of times.
str_replace()	It replaces all occurrences of the search string with the replacement string.
str_rot13()	It is used to perform the ROT13 encoding on a string.
str_shuffle()	It is used to randomly shuffle all characters in a string.
str_split()	It is used to split a string into an array.
strcoll()	It is locale based string comparison.
strip_tags()	It is used to strip HTML and PHP tags from a string.
str_word_count()	It is used to count the number of words in a string.
substr()	Return the part of a string
substr_compare()	Compares two strings from an offset up to the length of characters. (Binary safe comparison)
substr_count()	Count the number of times occurrence of a substring

substr_replace()	Replace some part of a string with another substring
trim()	Remove whitespace or other characters from the beginning and end of the string.
ucfirst()	Make the first character of the string to uppercase
ucwords()	Make the first character of each word in a string to uppercase
vfprintf()	Write a formatted string to a stream
vprintf()	Display the output as a formatted string according to format
vsprintf()	It returns a formatted string
wordwrap()	Wraps a string to a given number of characters

PHP Form Handling

We can create and use forms in PHP. To get form data, we need to use PHP superglobals $_GET and $_POST.

The form request may be get or post. To retrieve data from get request, we need to use $_GET, for post request $_POST.

PHP Get Form

Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. You can send limited amount of data through get request.

Let's see a simple example to receive data from get request in PHP.

File: form1.html

```
<form action="welcome.php" method="get">
Name: <input type="text" name="name"/>
<input type="submit" value="visit"/>
</form>
```

File: welcome.php

```
<?php
$name=$_GET["name"];//receiving name field value in $name variable
echo "Welcome, $name";
?>
```

PHP Post Form

Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc.

The data passed through post request is not visible on the URL browser so it is secured. You can send large amount of data through post request.

Let's see a simple example to receive data from post request in PHP.

File: form1.html

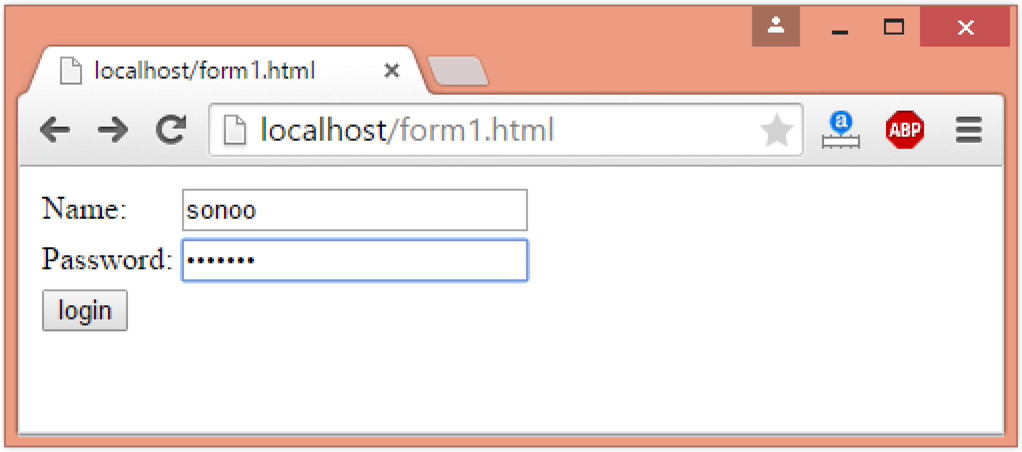
```
<form action="login.php" method="post">
<table>
<tr><td>Name:</td><td> <input type="text" name="name"/></td></tr>
<tr><td>Password:</td><td> <input type="password" name="password"/></td></tr>
<tr><td colspan="2"><input type="submit" value="login"/>  </td></tr>
```

```
</table>
</form>
```

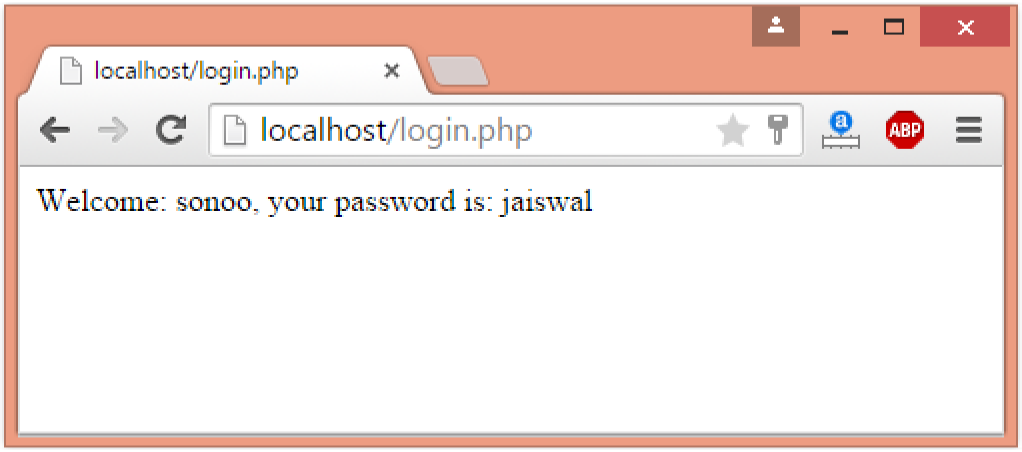
File: login.php

```
<?php
$name=$_POST["name"];//receiving name field value in $name variable
$password=$_POST["password"];//receiving password field value in $password variable
echo "Welcome: $name, your password is: $password";
?>
```

Output:



Name: sonoo
Password: •••••••
login

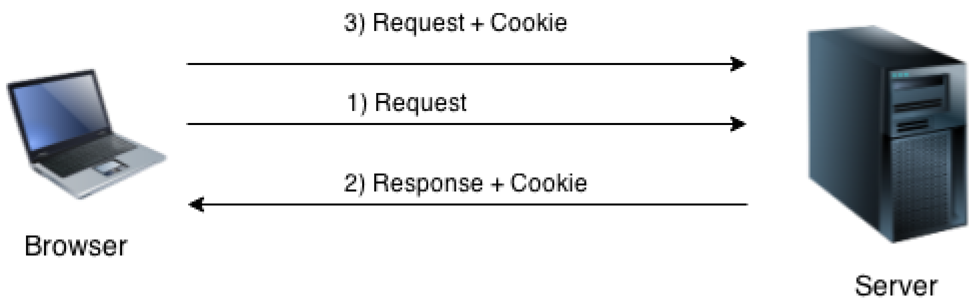


Welcome: sonoo, your password is: jaiswal

PHP Cookie

PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.

Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.



In short, cookie can be created, sent and received at server end.

Note: PHP Cookie must be used before <html> tag.

PHP setcookie() function

PHP setcookie() function is used to set cookie with HTTP response. Once cookie is set, you can access it by $_COOKIE superglobal variable.

Syntax

```
bool setcookie ( string $name [, string $value [, int $expire = 0 [, string $path
[, string $domain [, bool $secure = false [, bool $httponly = false ]]]]]] )
```

Example

```php
setcookie("CookieName", "CookieValue");/* defining name and value only*/
setcookie("CookieName", "CookieValue", time()+1*60*60);//using expiry in 1 hour(1*60*60 seconds or 3600 seconds)
setcookie("CookieName", "CookieValue", time()+1*60*60, "/mypath/", "mydomain.com", 1);
```

PHP $_COOKIE

PHP $_COOKIE superglobal variable is used to get cookie.

Example

```php
$value=$_COOKIE["CookieName"];//returns cookie value
```

PHP Cookie Example

File: cookie1.php

```php
<?php
setcookie("user", "Sonoo");
?>
<html>
<body>
<?php
if(!isset($_COOKIE["user"])) {
    echo "Sorry, cookie is not found!";
} else {
    echo "<br/>Cookie Value: " . $_COOKIE["user"];
}
?>
</body>
</html>
```

Output:

```
Sorry, cookie is not found!
```

Firstly cookie is not set. But, if you *refresh* the page, you will see cookie is set now.

Output:

```
Cookie Value: Sonoo
```

PHP Delete Cookie

If you set the expiration date in past, cookie will be deleted.

File: cookie1.php

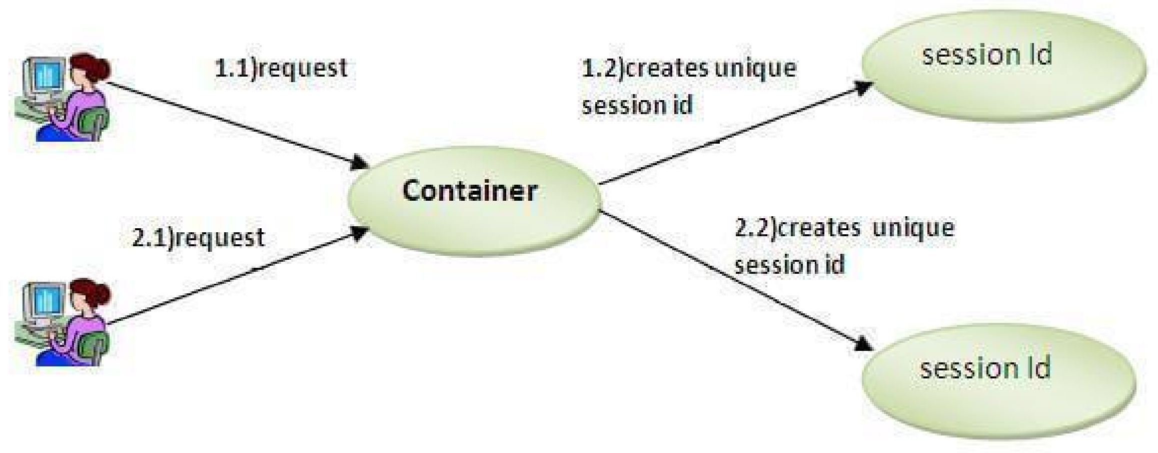
```php
<?php
setcookie ("CookieName", "", time() - 3600);// set the expiration date to one hour ago
?>
```

PHP Session

PHP session is used to store and pass information from one page to another temporarily (until user close the website).

PHP session technique is widely used in shopping websites where we need to store and pass cart information e.g. username, product code, product name, product price etc from one page to another.

PHP session creates unique user id for each browser to recognize the user and avoid conflict between multiple browsers.



PHP session_start() function

PHP session_start() function is used to start the session. It starts a new or resumes existing session. It returns existing session if session is created already. If session is not available, it creates and returns new session.

Syntax

```
1. bool session_start ( void )
```

Example

```
1. session_start();
```

PHP $_SESSION

PHP $_SESSION is an associative array that contains all session variables. It is used to set and get session variable values.

Example: Store information

```
1. $_SESSION["user"] = "Sachin";
```

Example: Get information

```php
echo $_SESSION["user"];
```

PHP Session Example

File: session1.php

```php
<?php
session_start();
?>
<html>
<body>
<?php
$_SESSION["user"] = "Sachin";
echo "Session information are set successfully.<br/>";
?>
<a href="session2.php">Visit next page</a>
</body>
</html>
```

File: session2.php

```php
<?php
session_start();
?>
<html>
<body>
<?php
echo "User is: ".$_SESSION["user"];
?>
</body>
</html>
```

PHP Session Counter Example

File: sessioncounter.php

```php
<?php
  session_start();

  if (!isset($_SESSION['counter'])) {
    $_SESSION['counter'] = 1;
  } else {
    $_SESSION['counter']++;
  }
  echo ("Page Views: ".$_SESSION['counter']);
?>
```

PHP Destroying Session

PHP session_destroy() function is used to destroy all session variables completely.

File: session3.php

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

The PHP Date() Function

The PHP `date()` function formats a timestamp to a more readable date and time.

Syntax

```
date(format,timestamp)
```

Parameter	Description
format	Required. Specifies the format of the timestamp
timestamp	Optional. Specifies a timestamp. Default is the current date and time

A timestamp is a sequence of characters, denoting the date and/or time at which a certain event occurred.

Get a Date

The required *format* parameter of the date() function specifies how to format the date (or time).

Here are some characters that are commonly used for dates:

- d - Represents the day of the month (01 to 31)
- m - Represents a month (01 to 12)
- Y - Represents a year (in four digits)
- l (lowercase 'L') - Represents the day of the week

Other characters, like"/", ".", or "-" can also be inserted between the characters to add additional formatting.

The example below formats today's date in three different ways:

Example

```php
<?php
echo "Today is " . date("Y/m/d") . "<br>";
echo "Today is " . date("Y.m.d") . "<br>";
echo "Today is " . date("Y-m-d") . "<br>";
echo "Today is " . date("l");
?>
```

Get a Time

Here are some characters that are commonly used for times:

- H - 24-hour format of an hour (00 to 23)
- h - 12-hour format of an hour with leading zeros (01 to 12)
- i - Minutes with leading zeros (00 to 59)
- s - Seconds with leading zeros (00 to 59)
- a - Lowercase Ante meridiem and Post meridiem (am or pm)

The example below outputs the current time in the specified format:

Example

```php
<?php
echo "The time is " . date("h:i:sa");
?>
```

Note that the PHP date() function will return the current date/time of the server!

Get Your Time Zone

If the time you got back from the code is not correct, it's probably because your server is in another country or set up for a different timezone.

So, if you need the time to be correct according to a specific location, you can set the timezone you want to use.

The example below sets the timezone to "America/New_York", then outputs the current time in the specified format:

Example

```php
<?php
date_default_timezone_set("America/New_York");
echo "The time is " . date("h:i:sa");
?>
```

PHP Global Variables - Superglobals

Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

The PHP superglobal variables are:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

PHP - Sort Functions For Arrays

In this chapter, we will go through the following PHP array sort functions:

- `sort()` - sort arrays in ascending order
- `rsort()` - sort arrays in descending order
- `asort()` - sort associative arrays in ascending order, according to the value
- `ksort()` - sort associative arrays in ascending order, according to the key
- `arsort()` - sort associative arrays in descending order, according to the value
- `krsort()` - sort associative arrays in descending order, according to the key

PHP File Upload

With PHP, it is easy to upload files to the server.

However, with ease comes danger, so always be careful when allowing file uploads!

Configure The "php.ini" File

First, ensure that PHP is configured to allow file uploads.

In your "php.ini" file, search for the `file_uploads` directive, and set it to On:`file_uploads = On`

Create The HTML Form

Next, create an HTML form that allow users to choose the image file they want to upload:

```
<!DOCTYPE html>

<html>

<body>



<form action="upload.php" method="post" enctype="multipart/form-data">

  Select image to upload:

  <input type="file" name="fileToUpload" id="fileToUpload">

  <input type="submit" value="Upload Image" name="submit">

</form>

</body>

</html>
```

Some rules to follow for the HTML form above:

- Make sure that the form uses method="post"
- The form also needs the following attribute: enctype="multipart/form-data". It specifies which content-type to use when submitting the form

Without the requirements above, the file upload will not work.

Other things to notice:

- The type="file" attribute of the <input> tag shows the input field as a file-select control, with a "Browse" button next to the input control

The form above sends data to a file called "upload.php", which we will create next.

Create The Upload File PHP Script

The "upload.php" file contains the code for uploading a file:

```
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
  $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
  if($check !== false) {
    echo "File is an image - " . $check["mime"] . ".";
  $uploadOk = 1;
  } else {
    echo "File is not an image.";
  $uploadOk = 0;
  }
}
?>
```

PHP script explained:

- $target_dir = "uploads/" - specifies the directory where the file is going to be placed
- $target_file specifies the path of the file to be uploaded
- $uploadOk=1 is not used yet (will be used later)
- $imageFileType holds the file extension of the file (in lower case)

- Next, check if the image file is an actual image or a fake image

- Note: You will need to create a new directory called "uploads" in the directory where "upload.php" file resides. The uploaded files will be saved there.

Check if File Already Exists

Now we can add some restrictions.

First, we will check if the file already exists in the "uploads" folder. If it does, an error message is displayed, and $uploadOk is set to 0:

```
// Check if file already exists
if (file_exists($target_file)) {
  echo "Sorry, file already exists.";
  $uploadOk = 0;
}
```

Limit File Size

The file input field in our HTML form above is named "fileToUpload".

Now, we want to check the size of the file. If the file is larger than 500KB, an error message is displayed, and $uploadOk is set to 0:

```
// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
  echo "Sorry, your file is too large.";
  $uploadOk = 0;
}
```

Limit File Type

The code below only allows users to upload JPG, JPEG, PNG, and GIF files. All other file types gives an error message before setting $uploadOk to 0:

```
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"

&& $imageFileType != "gif" ) {

  echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";

  $uploadOk = 0;

}
```

Complete Upload File PHP Script

The complete "upload.php" file now looks like this:

```
<?php

$target_dir = "uploads/";

$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);

$uploadOk = 1;

$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));


// Check if image file is a actual image or fake image

if(isset($_POST["submit"])) {

  $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);

  if($check !== false) {

    echo "File is an image - " . $check["mime"] . ".";

            $uploadOk = 1;

  } else {

    echo "File is not an image.";

            $uploadOk = 0;

  }

}


// Check if file already exists

if (file_exists($target_file)) {

  echo "Sorry, file already exists.";

  $uploadOk = 0;

}


// Check file size

if ($_FILES["fileToUpload"]["size"] > 500000) {
  echo "Sorry, your file is too large.";
  $uploadOk = 0;
}

// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
  echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
  $uploadOk = 0;
}
// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
  echo "Sorry, your file was not uploaded.";
// if everything is ok, try to upload file
} else {
  if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
    echo "The file ". htmlspecialchars( basename( $_FILES["fileToUpload"]["name"])). " has been
uploaded.";
  } else {
    echo "Sorry, there was an error uploading your file.";
  }}
?>
```

PHP Form Validation

This and the next chapters show how to use PHP to validate form data.

PHP Form Validation

Think SECURITY when processing PHP forms!

These pages will show how to process PHP forms with security in mind. Proper validation of form data is important to protect your form from hackers and spammers!

The HTML form we will be working at in these chapters, contains various input fields: required and optional text fields, radio buttons, and a submit button:

The validation rules for the form above are as follows:

Field	**Validation Rules**
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)

Gender	Required. Must select one

First we will look at the plain HTML code for the form:

Text Fields

The name, email, and website fields are text input elements, and the comment field is a textarea. The HTML code looks like this:

```
Name: <input type="text" name="name">

E-mail: <input type="text" name="email">

Website: <input type="text" name="website">

Comment: <textarea name="comment" rows="5" cols="40"></textarea>
```

Radio Buttons

The gender fields are radio buttons and the HTML code looks like this:

```
Gender:

<input type="radio" name="gender" value="female">Female

<input type="radio" name="gender" value="male">Male

<input type="radio" name="gender" value="other">Other
```

The Form Element

The HTML code of the form looks like this:

```
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

When the form is submitted, the form data is sent with method="post".

What is the $_SERVER["PHP_SELF"] variable?

The $_SERVER["PHP_SELF"] is a super global variable that returns the filename of the currently executing script.

So, the $_SERVER["PHP_SELF"] sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.

What is the htmlspecialchars() function?

The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with &lt; and &gt;. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

Big Note on PHP Form Security

The $_SERVER["PHP_SELF"] variable can be used by hackers!

If PHP_SELF is used in your page then a user can enter a slash (/) and then some Cross Site Scripting (XSS) commands to execute.

Cross-site scripting (XSS) is a type of computer security vulnerability typically found in Web applications. XSS enables attackers to inject client-side script into Web pages viewed by other users.

Assume we have the following form in a page named "test_form.php":

```
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
```

Now, if a user enters the normal URL in the address bar like "http://www.example.com/test_form.php", the above code will be translated to:

```
<form method="post" action="test_form.php">
```

So far, so good.

However, consider that a user enters the following URL in the address bar:

```
http://www.example.com/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E
```

In this case, the above code will be translated to:

```
<form method="post" action="test_form.php/"><script>alert('hacked')</script>
```

This code adds a script tag and an alert command. And when the page loads, the JavaScript code will be executed (the user will see an alert box). This is just a simple and harmless example how the PHP_SELF variable can be exploited.

Be aware of that any JavaScript code can be added inside the <script> tag! A hacker can redirect the user to a file on another server, and that file can hold malicious code that can alter the global variables or submit the form to another address to save the user data, for example.

How To Avoid $_SERVER["PHP_SELF"] Exploits?

$_SERVER["PHP_SELF"] exploits can be avoided by using the htmlspecialchars() function.

The form code should look like this:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

The htmlspecialchars() function converts special characters to HTML entities. Now if the user tries to exploit the PHP_SELF variable, it will result in the following output:

```
<form method="post" action="test_form.php/&quot;&gt;&lt;script&gt;alert('hacked')&lt;/script&gt;">
```

The exploit attempt fails, and no harm is done!

Validate Form Data With PHP

The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.

When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:

<script>location.href('http://www.hacked.com')</script>

- this would not be executed, because it would be saved as HTML escaped code, like this:

&lt;script&gt;location.href('http://www.hacked.com')&lt;/script&gt;

The code is now safe to be displayed on a page or inside an e-mail.

We will also do two more things when the user submits the form:

1. Strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP trim() function)
2. Remove backslashes (\) from the user input data (with the PHP stripslashes() function)

The next step is to create a function that will do all the checking for us (which is much more convenient than writing the same code over and over again).

We will name the function test_input().

Now, we can check each $_POST variable with the test_input() function, and the script looks like this:

Example

```
<?php

// define variables and set to empty values

$name = $email = $gender = $comment = $website = "";
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

Notice that at the start of the script, we check whether the form has been submitted using $_SERVER["REQUEST_METHOD"]. If the REQUEST_METHOD is POST, then the form has been submitted - and it should be validated. If it has not been submitted, skip the validation and display a blank form.

However, in the example above, all input fields are optional. The script works fine even if the user does not enter any data.

The next step is to make input fields required and create error messages if needed.

PHP MySQL Connect

Since PHP 5.5, **mysql_connect()** extension is *deprecated*. Now it is recommended to use one of the 2 alternatives.

- **mysqli_connect()**
- **PDO::__construct()**

PHP mysqli_connect()

PHP **mysqli_connect() function** is used to connect with MySQL database. It returns *resource* if connection is established or *null*.

Syntax

1. resource mysqli_connect (server, username, password)

PHP mysqli_close()

PHP **mysqli_close() function** is used to disconnect with MySQL database. It returns *true* if connection is closed or *false*.

Syntax

1. bool mysqli_close(resource $resource_link)

PHP MySQL Connect Example

Example

```php
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$conn = mysqli_connect($host, $user, $pass);
if(! $conn )
{
  die('Could not connect: ' . mysqli_error());
}
echo 'Connected successfully';
mysqli_close($conn);

?>
```

Output:
```
Connected successfully
```

PHP MySQL Create Database

Since PHP 4.3, **mysql_create_db()** function is *deprecated*. Now it is recommended to use one of the 2 alternatives.

- **mysqli_query()**
- **PDO::__query()**

PHP MySQLi Create Database Example

Example

```php
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$conn = mysqli_connect($host, $user, $pass);
if(! $conn )
{
  die('Could not connect: ' . mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$sql = 'CREATE Database mydb';
if(mysqli_query( $conn,$sql)){
  echo "Database mydb created successfully.";
}else{
echo "Sorry, database creation failed ".mysqli_error($conn);
}
mysqli_close($conn);

?>
```

Output:

```
Connected successfully
Database mydb created successfully.
```

PHP MySQL Create Table

PHP mysql_query() function is used to create table. Since PHP 5.5, **mysql_query()** function is *deprecated*. Now it is recommended to use one of the 2 alternatives.

- **mysqli_query()**
- **PDO::__query()**

PHP MySQLi Create Table Example

Example

```
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$dbname = 'test';

$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$sql = "create table emp5(id INT AUTO_INCREMENT,name VARCHAR(20) NOT NULL,
emp_salary INT NOT NULL,primary key (id))";
if(mysqli_query($conn, $sql)){
 echo "Table emp5 created successfully";
}else{
echo "Could not create table: ". mysqli_error($conn);
}

mysqli_close($conn);
```

```
?>
```

Output:

```
Connected successfully
Table emp5 created successfully
```

deprecated. Now it is recommended to use one of the 2 alternatives.

- **mysqli_query()**
- **PDO::__query()**

PHP MySQLi Insert Record Example

Example

```php
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$dbname = 'test';

$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$sql = 'INSERT INTO emp4(name,salary) VALUES ("sonoo", 9000)';
if(mysqli_query($conn, $sql)){
 echo "Record inserted successfully";
}else{
```

```
echo "Could not insert record: ". mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Output:

```
Connected successfully
Record inserted successfully
```

Laravel Tutorial

Laravel is an open-source PHP framework. It also offers the rich set of functionalities that incorporates the basic features of PHP frameworks such as CodeIgniter, Yii, and other programming languages like Ruby on Rails.

Our Laravel tutorial includes all the topics such as introduction, history of Laravel, installation, Laravel routes, Laravel controllers, Laravel views, etc.

What is Laravel?

Laravel is a PHP framework that uses the MVC architecture.

where,

- **Framework:** It is the collection of methods, classes, or files that the programmer uses, and they can also extend its functionality by using their code.
- **Architecture:** It is the specific design pattern that the framework follows. Laravel is following the MVC architecture.

Let's first understand the MVC architecture.

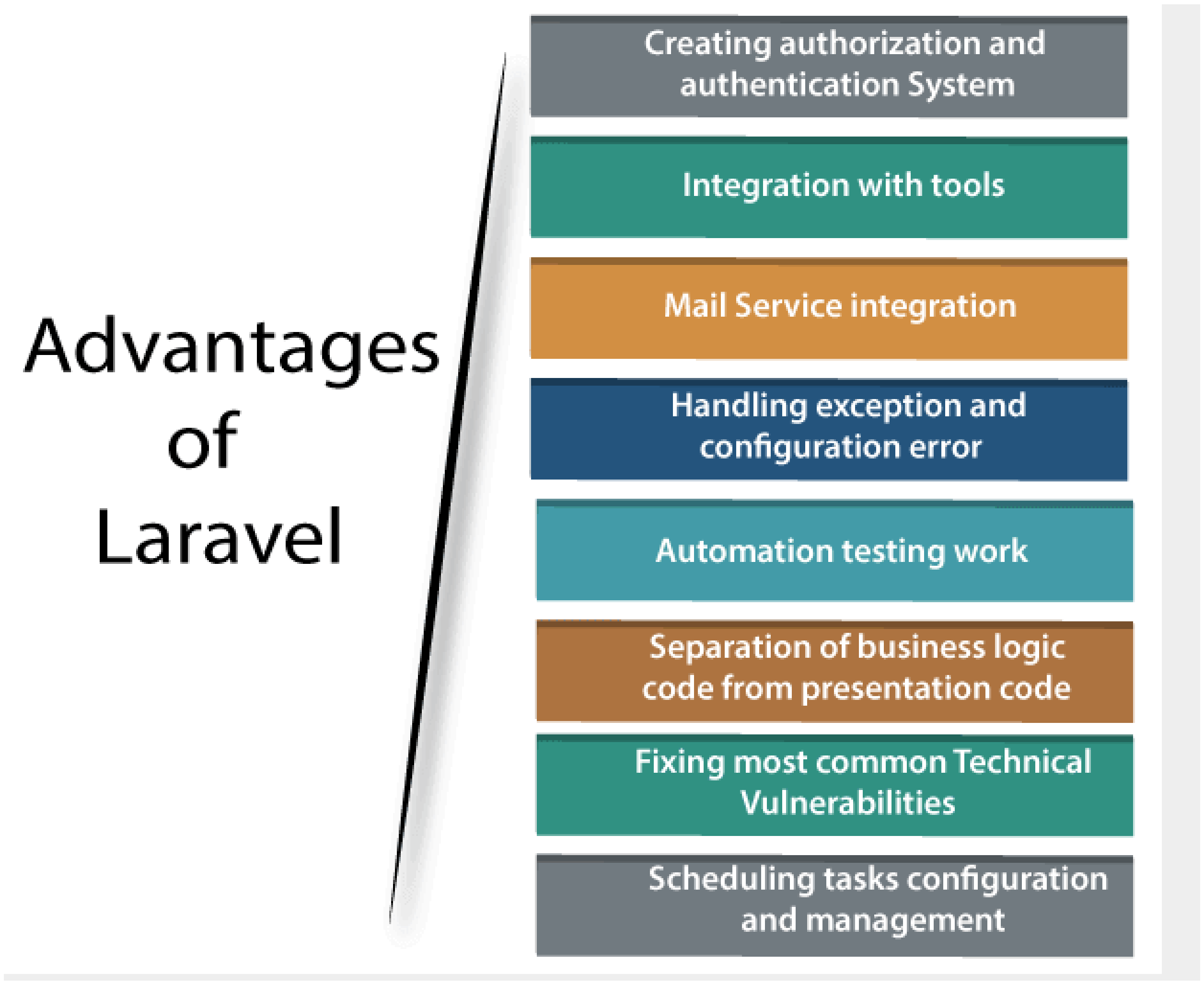
MVC is divided into three letters shown below:

- **M:** 'M' stands for **Model**. A model is a class that deals with a database. For example, if we have users in an application then we will have users model that deals with a database to query the table of users if we have users model, then we will also have a users table. We conclude from the example that the model is going to have a table for that specific model.
- **V:** 'V' stands for **View**. A view is a class that deals with an HTML. Everything that we can see on the application in the browser is the view or the representation.
- **C:** 'C' stands for **Controller**. A controller is the middle-man that deals with both model and view. A controller is the class that retrieves the data from the model and sends the data to the view class.

Laravel is an open-source PHP framework. It also offers the rich set of functionalities that incorporates the basic features of PHP frameworks such as CodeIgniter, Yii, and other programming languages like Ruby on Rails.

Advantages of Laravel

Following are some advantages of Laravel:


Advantages of Laravel
Creating authorization and authentication System
Integration with tools
Mail Service integration
Handling exception and configuration error
Automation testing work
Separation of business logic code from presentation code
Fixing most common Technical Vulnerabilities
Scheduling tasks configuration and management

- **Creating authorization and authentication systems**

  Every owner of the web application makes sure that unauthorized users do not access secured or paid resources. It provides a simple way of implementing authentication. It also provides a simple way of organizing the authorization logic and control access to resources.

- **Integration with tools**

  Laravel is integrated with many tools that build a faster app. It is not only necessary to build the app but also to create a faster app. Integration with the caching back end is one of the major steps to improve the performance of a web app. Laravel is integrated with some popular cache back ends such as **Redis**, and **Memcached**.

- **Mail service integration**
  Laravel is integrated with the Mail Service. This service is used to send notifications to the user's emails. It provides a clean and simple API that allows you to send the email quickly through a local or cloud-based service of your choice.
- **Handling exception and configuration error**
  Handling exception and configuration errors are the major factors on the app's usability. The manners in which the software app handles the errors have a huge impact on the user's satisfaction and the app's usability. The organization does not want to lose their customers, so for them, Laravel is the best choice. In Laravel, error and exception handling is configured in the new Laravel project.
- **Automation testing work**
  Testing a product is very important to make sure that the software runs without any errors, bugs, and crashes. We know that automation testing is less time-consuming than manual testing, so automation testing is preferred over the manual testing. Laravel is developed with testing in mind.
- **Separation of business logic code from presentation code**
  The separation between business logic code and presentation code allows the HTML layout designers to change the look without interacting with the developers. A bug can be resolved by the developers faster if the separation is provided between the business logic code and presentation code. We know that Laravel follows the **MVC architecture**, so separation is already done.
- **Fixing most common technical vulnerabilities**
  The **security vulnerability** is the most important example in web application development. An American organization, i.e., OWASP Foundation, defines the most important security vulnerabilities such as SQL injection, cross-site request forgery, cross-site scripting, etc. Developers need to consider these vulnerabilities

and fix them before delivery. Laravel is a secure framework as it protects the web application against all the security vulnerabilities.

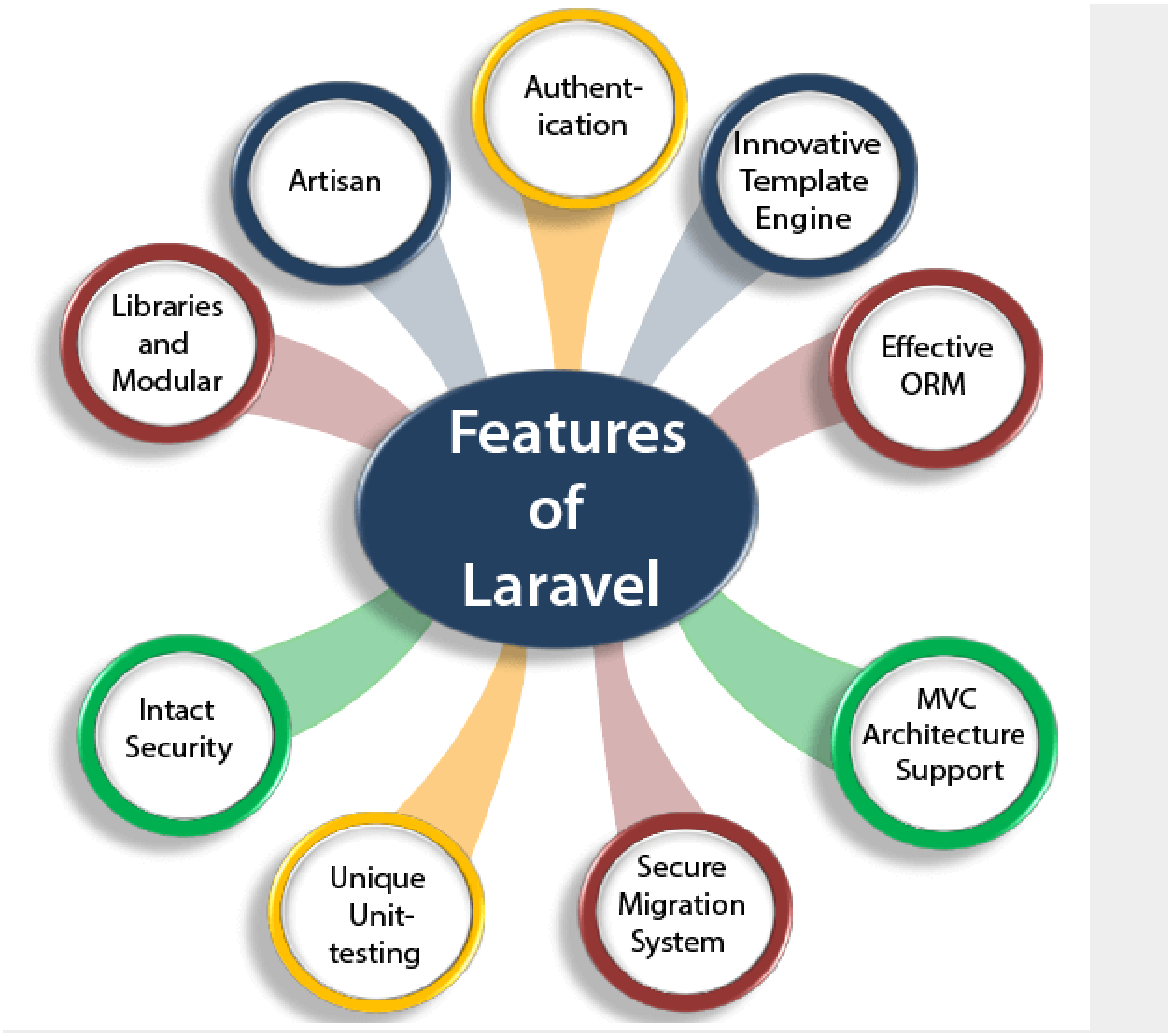
- **Scheduling tasks configuration and management**
  The web app requires some task scheduling mechanism to perform the tasks in time for example, when to send out the emails to the subscribers or when to clean up the database tables at the end of the day. To schedule the tasks, developers need first to create the **Cron entry** for each task, but **Laravel command scheduler** defines a command schedule which requires a single entry on the server.

Features of Laravel

We know that PHP is the oldest programming language used by the programmers, and more than 20 million websites are developed by using PHP. PHP is a very suitable programming language as it satisfies the business requirements whether the business is big or small. Laravel is one of the most popular frameworks having a high rich set of functionalities.

Laravel is provided with a well-defined toolbox that allows the developers to write less code leading to less possibility of errors.

Following are the features of Laravel:



Authentication

Authentication is the most important factor in a web application, and developers need to spend a lot of time writing the authentication code. Laravel makes a simpler authentication when Laravel is updated to Laravel 5. Laravel contains an inbuilt authentication system, you only need to configure models, views, and controllers to make the application work.

Innovative Template Engine

Laravel provides an innovative template engine which allows the developers to create the dynamic website. The available widgets in Laravel can be used to create solid structures for an application.

Effective ORM

Laravel contains an inbuilt ORM with easy PHP Active Record implementation. An effective ORM allows the developers to query the database tables by using the simple PHP syntax without writing any SQL code. It provides easy integration between the developers and database tables by giving each of the tables with their corresponding models.

MVC Architecture Support

Laravel supports MVC architecture. It provides faster development process as in MVC; one programmer can work on the view while other is working on the controller to create the business logic for the web application. It provides multiple views for a model, and code duplication is also avoided as it separates the business logic from the presentation logic.

Secure Migration System

Laravel framework can expand the database without allowing the developers to put much effort every time to make changes, and the migration process of Laravel is very secure and full-proof. In the whole process, **php code** is used rather than **SQL code**.

Unique Unit-testing

Laravel provides a unique unit-testing. Laravel framework can run several test cases to check whether the changes harm the web app or not. In Laravel, developers can also write the test cases in their own code.

Intact Security

Application security is one of the most important factors in web application development. While developing an application, a programmer needs to take effective ways to secure the application. Laravel has an inbuilt web application security, i.e., it itself takes care of the security of an application. It uses "Bcrypt Hashing Algorithm" to generate the salted password means that the password is saved as an encrypted password in a database, not in the form of a plain text.

Libraries and Modular

Laravel is very popular as some Object-oriented libraries, and pre-installed libraries are added in this framework, these pre-installed libraries are not added in other **php frameworks**. One of the most popular libraries is an **authentication library** that contains some useful features such as password reset, monitoring active users, Bcrypt hashing, and CSRF protection. This framework is divided into several modules that follow the php principles allowing the developers to build responsive and modular apps.

references

https://www.includehelp.com/php/cookies-aptitude-questions-and-answers.aspx