CM – 402 UNIT – 5

Web servers and Server side scripting using PHP

Develop Dynamic web site using server side PHP Programming and database connectivity is using PHP

Web Servers: Introduction, HTTP Request Types, System Architecture, Client-Side versus Server-Side Scripting, Accessing Web Servers-IIS, Apache, Requesting HTML, PHP documents.

PHP: Fundamentals of PHP, Data types, String functions, Arrays, form handling, Databases, Cookies, Sessions, Passing data from one web page to other web page.

Web Server

The Architecture of a Web server

A web server is a software application and hardware device that stores, processes, and serves web content to users over the internet. It plays a critical role in the client-server model of the World Wide Web, where clients (typically web browsers) request web pages and resources, and servers respond to these requests by delivering the requested content.

Web servers operate on the Hypertext Transfer Protocol (HTTP), which is the foundation of data communication on the World Wide Web. When you enter a website's URL into your browser, it sends an HTTP request to the web server hosting that website, which then sends back the web page you requested, allowing you to view it in your browser.

The Architecture of a Web server(Contd.)

Web server architecture refers to the structure and design of web servers, outlining how they handle incoming requests and deliver web content.

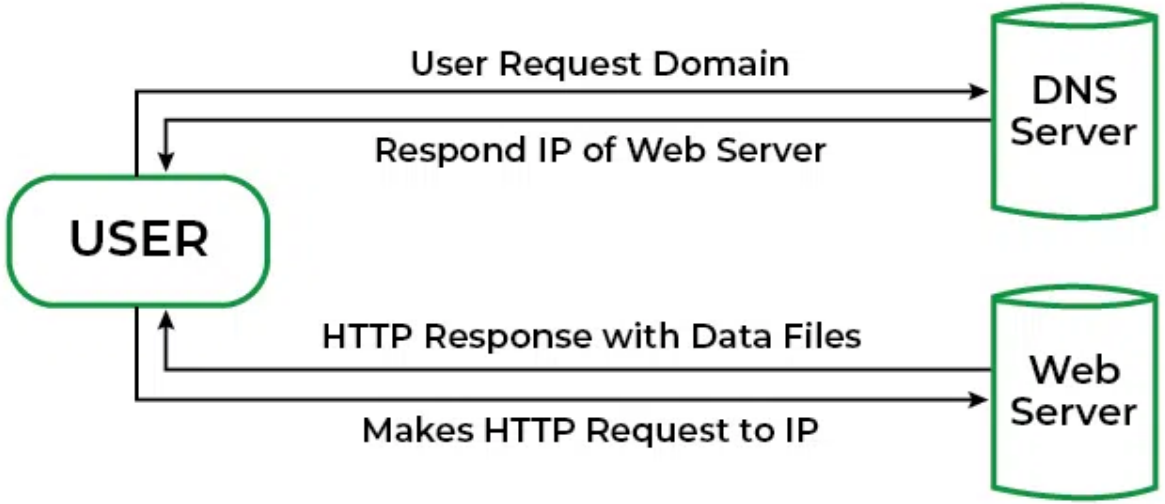
There are two main approaches to web server architecture:

- Single Tier Architecture
- Multiple Tier Architecture

The Architecture of a Web server(Contd.)

Single-Tier (Single Server) Architecture:

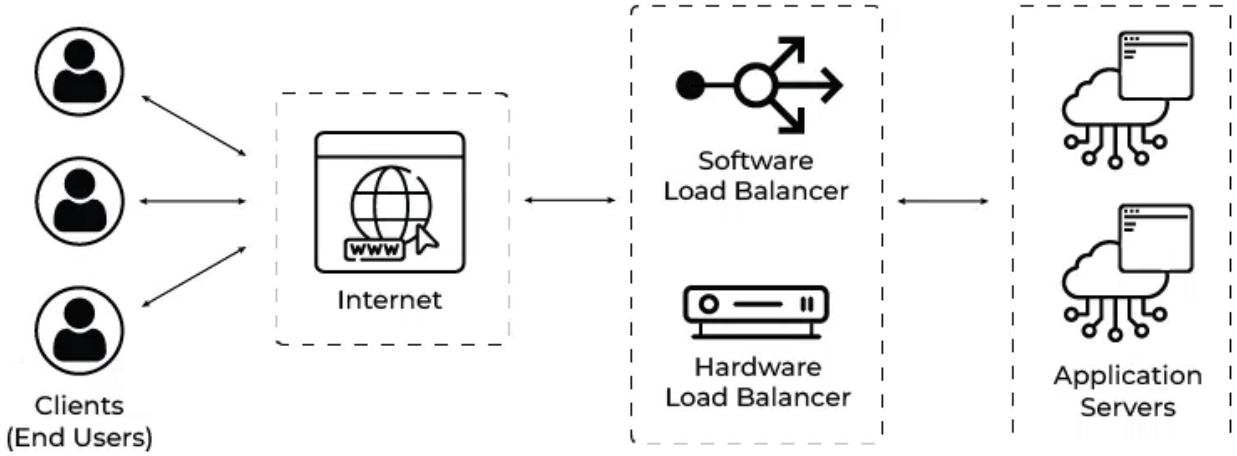
In a single-tier architecture, a single server is responsible for both processing requests and serving web content. This is suitable for small websites or applications with low traffic. However, it has limitations in terms of scalability and fault tolerance. If the server goes down, the entire service becomes unavailable.



The Architecture of a Web server(Contd.)

Multi-Tier (Load-Balanced) Architecture:

In a multi-tier architecture, multiple servers are used to distribute the workload and ensure high availability. This approach often involves load balancers that evenly distribute incoming requests across a cluster of web servers. Each server can serve web content independently, and if one server fails, the load balancer redirects traffic to healthy servers, ensuring uninterrupted service.



The Working of a Web server

Working of Web Servers

A web server works in the following ways:

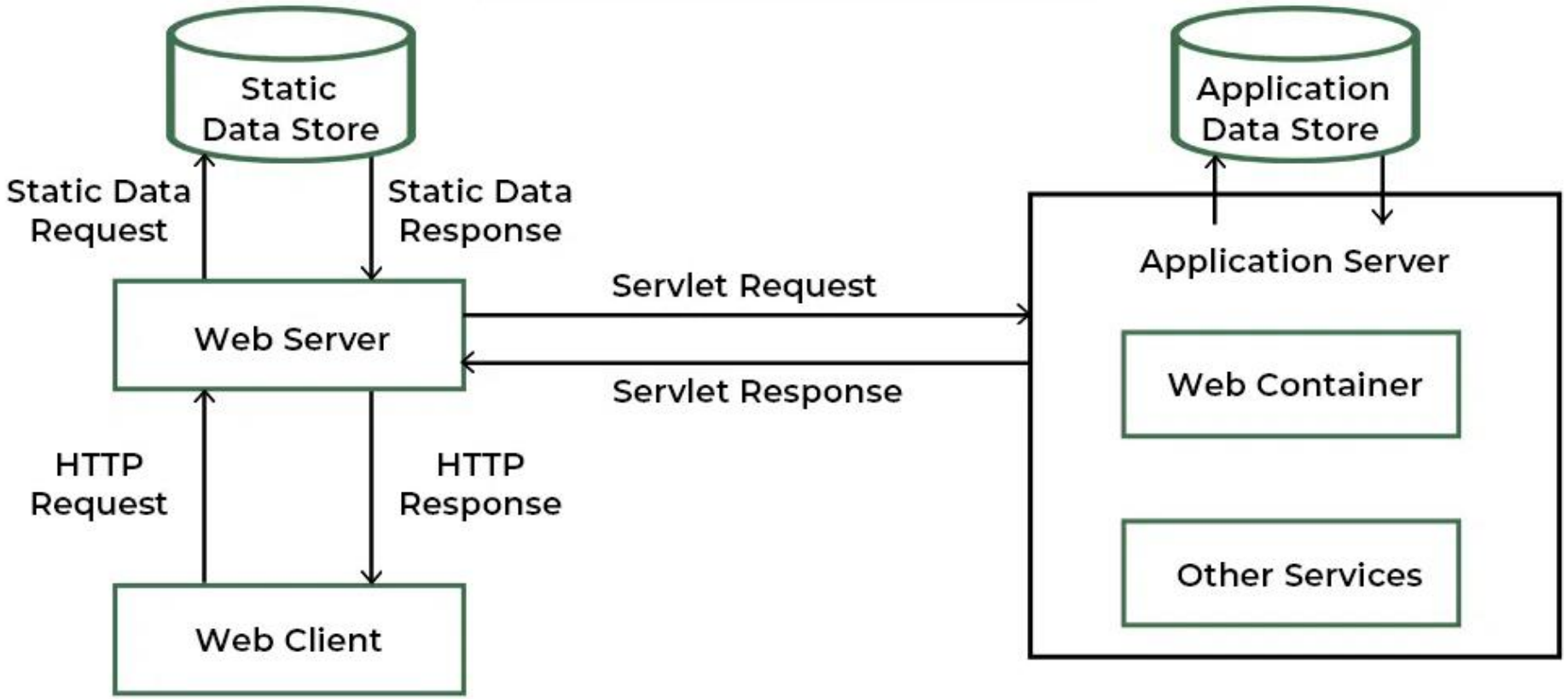
Obtain the IP address from domain name: IP address is obtained in two ways either by searching it in the cache or requesting DNS Servers

Requests full URL from Browsers: After fetching IP address a full URL is demanded from web server

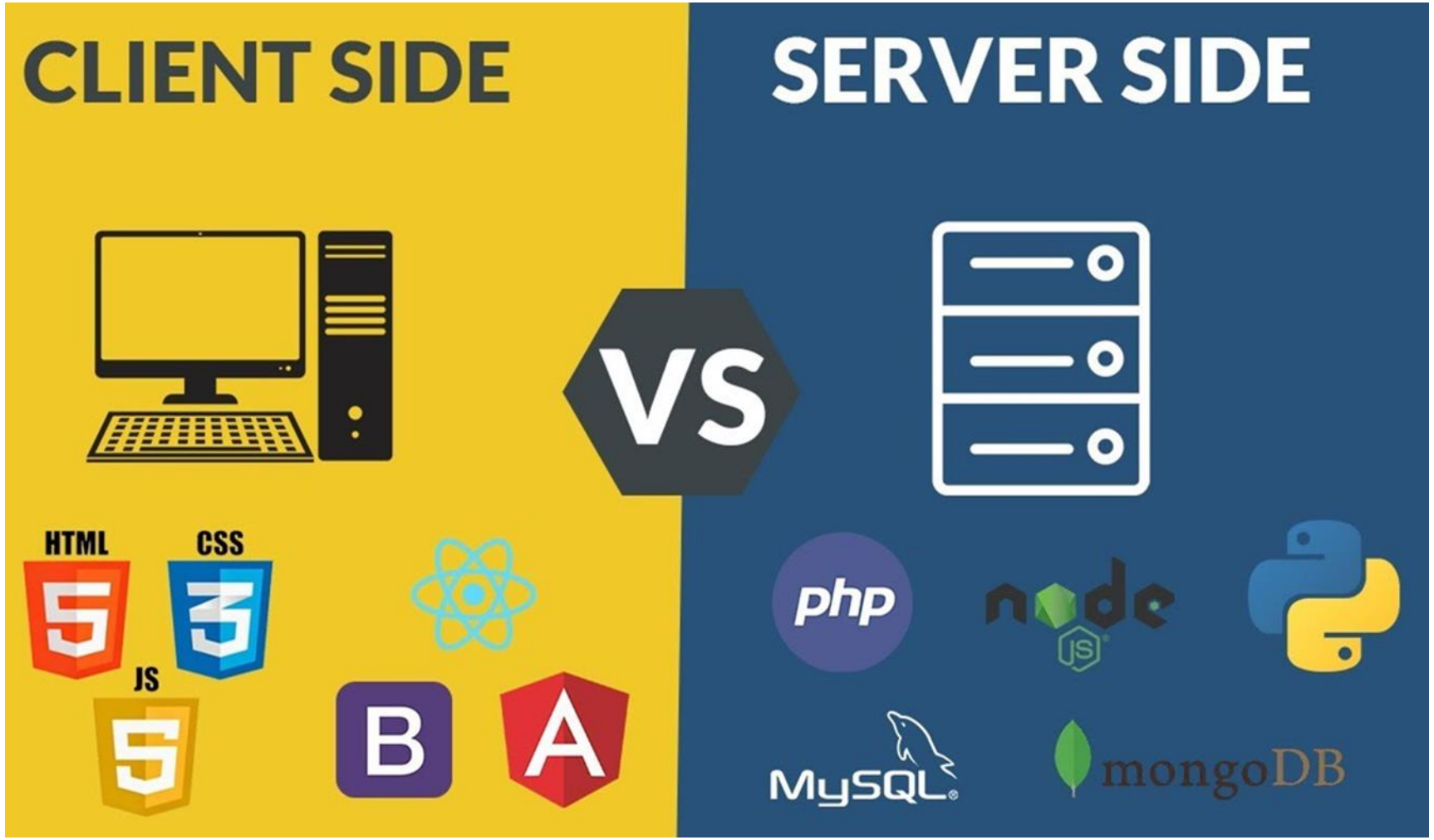
Web Server Responds to the request: In accordance with the request a response is sent by the server in case of successful request otherwise appropriate error message is sent

The Web Page is displayed on the browser: After getting the response from the server, the web browser displays the result

The Working of a Web server(Contd.)



Client Side vs Server Side



Client Side vs Server Side(Contd.)

Client Side Scripting:

Web browsers execute client-side scripting. It is used when browsers have all code. Source code is used to transfer from webserver to user's computer over the internet and run directly on browsers. It is also used for validations and functionality for user events.

It allows for more interactivity. It usually performs several actions without going to the user. It cannot be basically used to connect to databases on a web server. These scripts cannot access the file system that resides in the web browser. Pages are altered on basis of the user's choice. It can also be used to create "cookies" that store data on the user's computer.

Client Side vs Server Side(Contd.)

Server Side Scripting:

Web servers are used to execute server-side scripting. They are basically used to create dynamic pages. It can also access the file system residing at the webserver. A server-side environment that runs on a scripting language is a web server.

Scripts can be written in any of a number of server-side scripting languages available. It is used to retrieve and generate content for dynamic pages. It is used to require to download plugins. In this load times are generally faster than client-side scripting. When you need to store and retrieve information a database will be used to contain data. It can use huge resources of the server. It reduces client-side computation overhead. The server sends pages to the request of the user/client.

Client Side vs Server Side(Contd.)

Client-side scripting	**Server-side scripting**
Source code is visible to the user.	Source code is not visible to the user because its output of server-sideside is an HTML page.
Its main function is to provide the requested output to the end user.	Its primary function is to manipulate and provide access to the respective database as per the request.
It usually depends on the browser and its version.	In this any server-side technology can be used and it does not depend on the client.
There are many advantages linked with this like faster response times, a more interactive application.	The primary advantage is its ability to highly customize, response requirements, access rights based on user.

Client Side vs Server Side(Contd.)

Client-side scripting	**Server-side scripting**
It runs on the user's computer.	It runs on the webserver.
It does not provide security for data.	It provides more security for data.
It is a technique used in web development in which scripts run on the client's browser.	It is a technique that uses scripts on the webserver to produce a response that is customized for each client's request.
HTML, CSS, and javascript are used.	PHP, Python, Java, Ruby are used.
No need of interaction with the server.	It is all about interacting with the servers.
It reduces load on processing unit of the server.	It surge the processing load on the server.

Web Server

Features of Web Servers

Web servers offer a range of features, including:

Content Hosting: They store and serve web content, including HTML pages, images, videos, and other multimedia files.

Security: Web servers implement various security mechanisms to protect against unauthorized access and cyberattacks.

Load Balancing: Some web servers can distribute incoming traffic across multiple server instances to ensure optimal performance and availability.

Logging and Monitoring: They provide tools to track and analyze server performance, user access, and error logs.

Caching: Web servers can cache frequently accessed content to reduce server load and improve response times.

Web Server

Benefits of Web Servers

Using web servers offers several advantages, including:

Scalability: Web servers can handle a large number of simultaneous connections, making them suitable for high-traffic websites.

Reliability: They are designed for continuous operation and can recover from failures gracefully.

Security: Web servers include security features to protect against common web threats like DDoS attacks and SQL injection.

Customization: Web server configurations can be tailored to specific application requirements.

List of Web Servers

The list of popular web servers is,

1. Apache
2. IIS
3. Nginx
4. LiteSpeed

List of Web Servers(Contd.)

Apache Web Server

The of the most popular web server in the world developed by the Apache Software Foundation. Apache is open-source software that supports almost all operating systems including Linux, Unix, Windows, FreeBSD, Mac OS X, and more. About 60% of machines run on Apache Web Server.

Customization of the Apache web server is easy as it contains a modular structure. It is also open-source which means that you can add your own modules to the server when required and make modifications that suit your requirements.

It is more stable than any other web server and is easier to solve administrative issues. It can be installed on multiple platforms successfully.

Recent Apache releases provide you the feasibility of handling more requests when you compare them to its earlier versions.

List of Web Servers(Contd.)

IIS Web Server

IIS is a Microsoft product. This server has all the features just like Apache. But it is not an open-source and moreover adding personal modules is not easy and modification becomes a little difficult job.

Microsoft developed this product and they maintain it, thus it works with all the windows operating system platforms. Also, they provide good customer support if it had any issues.

List of Web Servers(Contd.)

Nginx Web Server

Another free open source web server is Nginx, which includes IMAP/POP3 proxy server. Nginx is known for its high performance, stability, simple configuration, and low resource usage.

This web server doesn’t use threads to handle requests but rather a much more scalable event-driven architecture that uses small and predictable amounts of memory under load. It is getting popular in recent times and it is hosting about 7.5% of all domains worldwide. Most web hosting companies are using this in recent times.

List of Web Servers(Contd.)

LightSpeed Web Server

LiteSpeed (LSWS) is a high-performance Apache drop-in replacement. LSWS is the 4th most popular web server on the internet and it is a commercial web server.

Upgrading your webserver to LiteSpeed will improve performance and lower operating costs.

This is compatible with most common apache features, including mod rewrite, .htaccess, and mod_security. LSWS can load apache configuration files directly and works as a drop-in replacement apache with most of the hosting control panels. It replaces apache in less than 15 minutes with zero downtime.

Unlike other front-end proxy solutions, LSWS replaces all Apache functions, simplifying use and making the transition from Apache smooth and easy. Most of the hosting companies were using LSWS in recent times.

Compare the properties of IIS and Apache

Parameters of Comparison	**IIS**	**Apache**
Developer	Microsoft	Apache Software Foundation
Compatibility	Compatible with Windows OS only.	Compatible with almost every OS.
Operations	It is easy to learn and use.	Its ease of operations depends on the OS.
Security	Prone to security risks.	No security risks.
User Support	It offers corporate Support.	It offers community support.

HTTP Request Types

Different types of HTTP requests,

GET	Used to retrieve data from the specified resource.
POST	Used to submit data to be processed to a specified resource.
PUT	Used to update a resource or create a new resource if it does not exist.
HEAD	Used to update a resource or create a new resource if it does not exist.
DELETE	Used to request that a resource be removed.
PATCH	Used to apply partial modifications to a resource.
OPTIONS	Used to describe the communication options for the target resource.
CONNECT	Used to establish a tunnel to the server identified by a given URI.
TRACE	Used to perform a message loop-back test along the path to the target resource.

Setting up Environment

To continue further from here, we need to work on servers and databases. Let’s explore how to configure our local system to process our needs.

Softwares to be installed,

1. PHP
2. Apache Web Server
3. MySQL

Setting up Environment(Contd.)

PHP

1. Visit https://windows.php.net/download/ and download Thread safe, latest version of PHP.
2. Now extract the contents into a safe folder
3. Then add the folder path into Environment variables to get accessed

Note: Run following command in command prompt to verify PHP is successfully installed. “php --version”

Setting up Environment(Contd.)

Apache Web Server

XAMPP server is sufficient to manage both PHP and MySQL. So let's install XAMPP.

1. Visit https://www.apachefriends.org/download.html and download latest version of XAMPP
2. After successful download, run the downloaded file.
3. Follow the commands and click OK to install successfully.

Setting up Environment(Contd.)

MySQL


php

Getting started with PHP

The term PHP is an acronym for PHP: Hypertext Preprocessor. PHP is a server-side scripting language designed specifically for web development. It is open-source which means it is free to download and use. It is very simple to learn and use. The files have the extension ".php".

Rasmus Lerdorf inspired the first version of PHP and participated in the later versions. It is an interpreted language and it does not require a compiler.

Getting started with PHP(Contd.)

Now we will say Hello to the world using PHP.

```
<html>
  <body>
    <?php echo “Hello, World!” ?>
  </body>
</html>
```

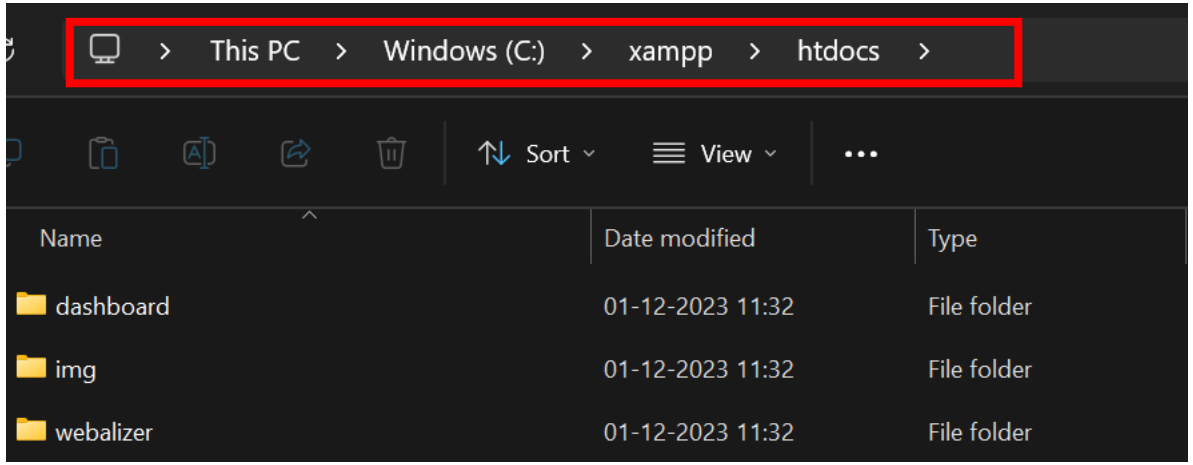
Note: From here onwards, the files should have .php extension

Getting started with PHP(Contd.)

How to run PHP?

It’s quite simple and easy if environment is properly configured. To run PHP programs, we had to copy the into “htdocs” folder of XAMPP.

Then open XAMPP Control Panel, Start the Apache Web Server. It will allow us to access the files of the “htdocs” folder from any local web browser with “localhost” as URL followed by the file path.



Data Types

Data Types define the type of data a variable can store. PHP allows eight different types of data types. All of them are discussed below. There are pre-defined, user-defined, and special data types. They are,

Data Type	Description
bool	a value that's either true or false.
int	a whole number value.
float	a numeric value with decimal.
string	a series of characters.
array	an ordered map of key/value pairs.
object	an instance of a pre-defined class.
callable	a reference to a PHP function.
iterable	represents any array or object implementing the Traversable interface.
resource	a reference to an external resource.
NULL	represents a variable with no value.

Data Types(Contd.)

The **predefined** data types are:

- Boolean
- Integer
- Double
- String

The **user-defined (compound)** data types are:

- Array
- Objects

The **special** data types are:

- NULL
- Resource

Data Types(Contd.)

Integer: Integers hold only whole numbers including positive and negative numbers, i.e., numbers without fractional part or decimal point. They can be decimal (base 10), octal (base 8), or hexadecimal (base 16). The default base is decimal (base 10). The octal integers can be declared with leading 0 and the hexadecimal can be declared with leading 0x. The range of integers must lie between -2^31 to 2^31.

Example:

```
<?php
        $deci1 = 50;
        $deci2 = 654;
        $sum = $deci1 + $deci2;
        echo $sum;
?>
```

Output: 704

Data Types(Contd.)

Double: Can hold numbers containing fractional or decimal parts including positive and negative numbers or a number in exponential form. By default, the variables add a minimum number of decimal places. The Double data type is the same as a float as floating-point numbers or real numbers.

Example:

```
<?php
        $val1 = 50.85;
        $val2 = 654.26;
        $sum = $val1 + $val2;
        echo $sum;
?>
```

Output: 705.11

Data Types(Contd.)

String: Hold letters or any alphabets, even numbers are included. These are written within double quotes during declaration. The strings can also be written within single quotes, but they will be treated differently while printing variables. To clarify this look at the example below.

Example:

```
<?php
        $name = "Krishna";
        echo "The name of the Geek is $name \n";
?>
```

Output: The name of the Geek is Krishna

Data Types(Contd.)

Boolean: Boolean data types are used in conditional testing. Hold only two values, either TRUE(1) or FALSE(0). Successful events will return true and unsuccessful events return false. NULL type values are also treated as false in Boolean. Apart from NULL, 0 is also considered false in boolean. If a string is empty then it is also considered false in boolean data type.

Example:

```
<?php
        if(TRUE)
           echo "This condition is TRUE";
        if(FALSE)
           echo "This condition is not TRUE";
?>
```

Output: This condition is TRUE

Data Types(Contd.)

Array: Array is a compound data type that can store multiple values of the same data type. Below is an example of an array of integers. It combines a series of data that are related together.

```
<?php
        $intArray = array( 10, 20 , 30);
        echo "First Element: $intArray[0]\n";
        echo "Second Element: $intArray[1]\n";
        echo "Third Element: $intArray[2]\n\n";
?>
```

Output

First Element: 10

Second Element: 20

Third Element: 30

Data Types(Contd.)

Objects: Objects are defined as instances of user-defined classes that can hold both values and functions and information for data processing specific to the class. This is an advanced topic and will be discussed in detail in further articles. When the objects are created, they inherit all the properties and behaviors from the class, having different values for all the properties. Objects are explicitly declared and created from the new keyword.

Data Types(Contd.)

```
<?php
class gfg {
  var $message;
  function gfg($message) {
    $this->message = $message;
  }
  function msg() {
    return "This is an example of " . $this->message . "!";
  }
}
$newObj = new gfg("Object Data Type");       // instantiating a object
echo $newObj -> msg();
?>
```

Output: This is an example of Object Data Type!

Data Types(Contd.)

NULL: These are special types of variables that can hold only one value i.e., NULL. We follow the convention of writing it in capital form, but it's case-sensitive. If a variable is created without a value or no value, it is automatically assigned a value of NULL. It is written in capital letters.

Example:

```
<?php
$nm = NULL;
echo $nm;    // this will return no output
?>
```

Output: NULL

Variables and Constants

Variables:

Variable in any programming language is a name given to a memory location that holds a value. You can say that variables are containers for any type of values. There are some rules to write variable names in PHP.

Rules for variable names:

Variable names in PHP start with a dollar ($) sign followed by the variable name.

Variable name can contain alphanumeric characters and underscore (_).

Variable names must start with a letter or an underscore (_). (For eg: $abc, $x1, $_g, $abc_1 etc.)

Variable names cannot start with a number.

Variable names are case-sensitive.

Variables and Constants(Contd.)

Scope of variables:
Variables can be declared anywhere in the program. Scope of a variable is a part of the program where the variable is accessible. PHP has three different variable scopes:
Local
Global
static
Local scope: A variable declared within a function has a local scope and can be accessed within a function only. A function is a small program performing a particular task which is called when required.
Global scope: A variable declared outside a function has a global scope and can be accessed outside the function only. Actually global variables can be accessed anywhere using the global keyword.
Static scope: A variable declared with static keyword is said to have static scope within the function. Normally when variables are executed, they lose their values or memory. But when a variable is declared as static, it doesn’t lose its value. It remains static within multiple function calls.

Variables and Constants(Contd.)

Variable declaration:

$variable_name=value;

Example of variable declaration is given below:

$x=5;

$x is a variable and 5 is a value assigned to $x variable using assignment operator (=). The assignment operator assigns the right hand side value to the left hand side variable in an expression. The variable name can be just alphabets or they can be some descriptive names like $school_name, $names, $games etc.

In PHP we can print a value of variable using an echo statement as follows:

```
<?php
        $x=10;
        echo $x;
?>
```

Variables and Constants(Contd.)

Constants:

Constants are the variables whose values are not changed throughout the script. A valid constant variable do not have a $ sign before its name. It starts with a letter or an underscore (_). Constants have global scope in the whole script.

Constants are useful in situations where same value is used in many places. For example: if we want to calculate an area and perimeter of a circle, we require the value of PI in both the cases. So we can have the value of PI defined as a constant and can use it effectively.

If we want to create an array of names having length 10, we can define the length 10 as a constant which will be used anywhere required. But if for some reason we decided to increase the length to 20, we can just change the value 10 to 20 in the constant definition which will be replicated everywhere.

Variables and Constants(Contd.)

Constants are declared using inbuilt **define()** function.
It takes 3 parameters,
Name of the constant
Value of the constant
Whether the constant should be case-insensitive. Default value is false.
The third parameter of define() function is optional. The false value of third parameter of define() function denotes that the constant name is case-sensitive and true value indicates that the constant name is case-insensitive.
Let us see how it is used in a program.

```
<?php
    define("MESSAGE","Welcome to PHP!");
    echo MESSAGE;
?>
```

String Manipulation Functions

String Manipulation functions allow you to manipulate or compare strings. These functions begin with the str prefix. Expand the category to view a list of the available functions.

Function Name	Description
strcat	Concatenates two strings.
strchr	Returns the pointer to the first occurrence of a character in a string.
strcmp	Compares two strings to determine the alphabetic order.
strcpy	Copies one string to another.
strdup	Duplicates a string.
stricmp	Performs a case-insensitive comparison of two strings.

String Manipulation Functions(Contd.)

Function Name	Description
strlen	Returns the length of a string.
strncat	Concatenates n characters from one string to another.
strncmp	Compares the first n characters of two strings.
strncpy	Copies the first n characters of one string to another.
strnicmp	Performs a case-insensitive comparison of n strings.
strrchr	Finds the last occurrence of a character in a string.
strspn	Returns the length of the leading characters in a string that are contained in a specified string.
strstr	Returns the first occurrence of one string in another.
strtok	Returns a token from a string delimited by specified characters.

Arrays

An array in PHP is actually an ordered map. A map is a type that associates values to keys. This type is optimized for several different uses; it can be treated as an array, list (vector), hash table (an implementation of a map), dictionary, collection, stack, queue, and probably more. As array values can be other arrays, trees and multidimensional arrays are also possible.

An array can be created using the array() language construct. It takes any number of comma-separated key => value pairs as arguments.

Arrays(Contd.)

Syntax:

```
array(
    key1  => value,
    key2 => value2,
    key3 => value3,
    ...
)
```

Arrays(Contd.)

There are basically three types of arrays in PHP:

Indexed or Numeric Arrays:

An array with a numeric index where values are stored linearly.

Associative Arrays:

An array with a string index where instead of linear storage, each value can be assigned a specific key.

Multidimensional Arrays:

An array which contains single or multiple array within it and can be accessed via multiple indices.

Arrays(Contd.)

Indexed or Numeric Arrays

These type of arrays can be used to store any type of elements, but an index is always a number. By default, the index starts at zero. These arrays can be created in two different ways as shown in the following example:

```
$array_one = array(“zero”, “one”, “two”);
```

Or

```
$array_two[0] = “zero”;
$array_two[1] = “one”;
```

Arrays(Contd.)

Associative Arrays

These types of arrays are similar to the indexed arrays but instead of linear storage, every value can be assigned with a user-defined key of string type.

Example:

```
$name_one = array("Zack"=>"Zara",
                  "Anthony"=>"Any",
                  "Ram"=>"Rani",
                  "Salim"=>"Sara", );
```

Arrays(Contd.)

Multidimensional Arrays

Multi-dimensional arrays are such arrays that store another array at each index instead of a single element.

In other words, we can define multi-dimensional arrays as an array of arrays.

As the name suggests, every element in this array can be an array and they can also hold other sub-arrays within.

Arrays or sub-arrays in multidimensional arrays can be accessed using multiple dimensions.

Arrays(Contd.)

```
//Defining Multi dimensional arrays
$favorites = array(
        array("name" => "D", "mob" => "1523", "email" => "d@g.com"),
        array("name" => "S", "mob" => "69721", "email" => "s@g.com"),
        array("name" => "J", "mob" => "47536", "email" => "j@g.com")
);



//Accessing Multi dimensional arrays
echo "Dave email-id is: " . $favorites[0]["email"];
```

Arrays with Examples

Find minimum element of an array.

```
<?php
        $a = array(5,8,1,4,15);
        $n = count($a);
        $min_element = $a[0];
        for($i = 1; $i < $n; $i++)
                if($a[$i] < $min_element)
                        $min_element = $a[$i];
        echo $min_element;
?>
```

Arrays with Examples(Contd.)

Find maximum element of an array.

```
<?php
        $a = array(5,8,1,4,15);
        $n = count($a);
        $max_element = $a[0];
        for($i = 1; $i < $n; $i++)
                if($a[$i] > $max_element)
                        $max_element = $a[$i];
        echo $max_element;
?>
```

Arrays with Examples(Contd.)

Search for an element in an array.

```
$a = array(5,8,1,4,15);
$n = count($a);
$element = 1;
$flag = false;
for($i = 1; $i < $n; $i++)
        if($a[$i] == $element){
                echo "Element found!!!";
                $flag = true;
}
if($flag == false)
        echo "Element not found!!!";
```

Form handling using $_GET,$_POST

Form handling in web development refers to the process of capturing user input submitted through HTML forms, processing that input on the server side, and responding accordingly. Forms are a fundamental part of web applications, allowing users to submit data for various purposes, such as user registration, login, search queries, and more.

$_GET and $_POST are two superglobal arrays in PHP that are used to collect form data submitted by the user. These arrays contain key-value pairs where keys are the names of form elements, and values are the data entered by the user.

Form handling using $_GET,$_POST(Contd.)

The GET Method

GET is used to request data from a specified resource. Note that the query string (name/value pairs) is sent in the URL of a GET request:

/test/demo_form.php?name1=value1&name2=value2

GET requests can be cached

GET requests remain in the browser history

GET requests can be bookmarked

GET requests should never be used when dealing with sensitive data

GET requests have length restrictions

GET requests are only used to request data (not modify)

Form handling using $_GET,$_POST(Contd.)

The POST Method

POST is used to send data to a server to create/update a resource. The data sent to the server with POST is stored in the request body of the HTTP request:

```
POST /test/demo_form.php HTTP/1.1
Host: w3schools.com
name1=value1&name2=value2
```

POST requests are never cached

POST requests do not remain in the browser history

POST requests cannot be bookmarked

POST requests have no restrictions on data length

Form handling using $_GET,$_POST(Contd.)

	GET	**POST**
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history

Form handling using $_GET,$_POST(Contd.)

	GET	**POST**
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

Form Handling using $_GET, $_POST(Contd.)

Let us explore an example to understand better,

sample.html

```
<html>
<body>
   <form method="post" action="sample.php"> //try it with GET also
     <input name="uname" />
     <input type="submit" />
   </form>
 </body>
</html>
```

Form Handling using $_GET, $_POST(Contd.)

Let us explore an example to understand better,

sample.php

```
<html>
<body>
        <?php
                echo “Hello ”, $_POST[“uname”];
        ?>
</body>
</html>
```

MySQL Functions in PHP

Function	Description
affected_rows()	Returns the number of affected rows in the previous MySQL operation
autocommit()	Turns on or off auto-committing database modifications
begin_transaction()	Starts a transaction
change_user()	Changes the user of the specified database connection
character_set_name()	Returns the default character set for the database connection
close()	Closes a previously opened database connection
commit()	Commits the current transaction
connect()	Opens a new connection to the MySQL server
warning_count()	Returns the number of warnings from the last query in the connection

MySQL Functions in PHP

Function	Description
connect_errno()	Returns the error code from the last connection error
connect_error()	Returns the error description from the last connection error
data_seek()	Adjusts the result pointer to an arbitrary row in the result-set
debug()	Performs debugging operations
dump_debug_info()	Dumps debugging info into the log
errno()	Returns the last error code for the most recent function call
error()	Returns the last error description for the most recent function call
error_list()	Returns a list of errors for the most recent function call
thread_safe()	Returns whether the client library is compiled as thread-safe
use_result()	Initiates the retrieval of a result-set from the last query executed

MySQL Functions in PHP

Function	Description
fetch_all()	Fetches all result rows as an associative array, a numeric array, or both
fetch_array()	Fetches a result row as an associative, a numeric array, or both
fetch_assoc()	Fetches a result row as an associative array
fetch_field()	Returns the next field in the result-set, as an object
fetch_field_direct()	Returns meta-data for a single field in the result-set, as an object
fetch_fields()	Returns an array of objects that represent the fields in a result-set
fetch_lengths()	Returns the lengths of the columns of the current row in the result-set
fetch_object()	Returns the current row of a result-set, as an object

MySQL Functions in PHP

Function	Description
fetch_row()	Fetches one row from a result-set and returns it as an enumerated array
field_count()	Returns the number of columns for the most recent query
field_seek()	Sets the field cursor to the given field offset
get_charset()	Returns a character set object
get_client_info()	Returns the MySQL client library version
get_client_stats()	Returns statistics about client per-process
get_client_version()	Returns the MySQL client library version as an integer
get_connection_stats()	Returns statistics about the client connection
get_host_info()	Returns the MySQL server hostname and the connection type

MySQL Functions in PHP

Function	Description
get_proto_info()	Returns the MySQL protocol version
get_server_info()	Returns the MySQL server version
get_server_version()	Returns the MySQL server version as an integer
info()	Returns information about the last executed query
init()	Initializes MySQLi and returns a resource for use with real_connect()
insert_id()	Returns the auto-generated id from the last query
kill()	Asks the server to kill a MySQL thread
more_results()	Checks if there are more results from a multi query
multi_query()	Performs one or more queries on the database
next_result()	Prepares the next result-set from multi_query()
options()	Sets extra connect options and affect behavior for a connection

MySQL Functions in PHP

Function	Description
ping()	Pings a server connection, or tries to reconnect if the connection has gone down
poll()	Polls connections
prepare()	Prepares an SQL statement for execution
query()	Performs a query against a database
real_connect()	Opens a new connection to the MySQL server
real_escape_string()	Escapes special characters in a string for use in an SQL statement
real_query()	Executes a single SQL query
reap_async_query()	Returns result from an async SQL query
refresh()	Refreshes/flushes tables or caches, or resets the replication server information
rollback()	Rolls back the current transaction for the database

MySQL Functions in PHP

Functions	Description
select_db()	Select the default database for database queries
set_charset()	Sets the default client character set
set_local_infile_default()	Unsets user defined handler for load local infile command
set_local_infile_handler()	Set callback function for LOAD DATA LOCAL INFILE command
sqlstate()	Returns the SQLSTATE error code for the error
ssl_set()	Used to establish secure connections using SSL
stat()	Returns the current system status
stmt_init()	Initializes a statement and returns an object for use with stmt_prepare()
store_result()	Transfers a result-set from the last query
thread_id()	Returns the thread ID for the current connection

Steps to Connect MySQL with PHP

PHP 5 and later can work with a MySQL database using:

MySQLi extension (the ‘i’ is abbreviation for improved)

PDO (PHP Data Objects)

Which one should we use **MySQLi** or **PDO**?

-> PDO will work with 12 different database systems, whereas MySQLi will only work with MySQL databases.

-> So, if you have to shift your project to use alternative database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the complete code — queries included.

-> Both are object-oriented, but MySQLi also offers a procedural API.

Steps to Connect MySQL with PHP(Contd.)

Connection to MySQL using MySQLi

PHP provides mysql_connect() function to open a database connection. This function takes a single parameter, which is a connection returned by the mysql_connect() function.

You can disconnect from the MySQL database anytime using another PHP function mysql_close().

There is also a procedural approach of MySQLi to establish a connection to MySQL database from a PHP script.

Steps to Connect MySQL with PHP(Contd.)

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

        // Connection
        $conn = new mysqli($servername, $username, $password);

        // For checking if connection is successful or not
        if ($conn->connect_error) {
                die("Connection failed: ". $conn->connect_error);
        }
        echo "Connected successfully";
?>
```

Steps to Connect MySQL with PHP(Contd.)

MySQL Procudeural

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

	// Connection
	$conn = mysqli_connect($servername, $username, $password);

	// Check if connection is Successful or not
	if (!$conn) {
		die("Connection failed: ". mysqli_connect_error());
	}
	echo "Connected successfully";
?>
```

Steps to Connect MySQL with PHP(Contd.)

Using PDO

```
<?php
        $servername = "localhost";
        $username = "username";
        $password = "password";
        try {
                $conn = new PDO("mysql:host=$servername;dbname=myDB",
                                        $username, $password);

                $conn->setAttribute(PDO::ATTR_ERRMODE,
                                                PDO::ERRMODE_EXCEPTION);
                echo "Connected successfully";
        } catch(PDOException $e) {
                echo "Connection failed: ". $e->getMessage();
        }
?>
```

CRUD Operations On Database

CRUD is the acronym for CREATE, READ, UPDATE and DELETE. These terms describe the four essential operations for creating and managing persistent data elements, mainly in relational and NoSQL databases.

CRUD Operations On Database(Contd.)

CREATE
The operation of CREATE means creating a database or creating table or at least inserting data into the table of a database. It helps to make new data.

Example:

```
<?php
$host = ‘localhost:3306’;
$user = ‘root’;
$pass = ‘password’;
$dbname = ‘vasavi’;
```

CRUD Operations On Database(Contd.)

```
$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
$sql = 'INSERT INTO emp4(name,salary) VALUES ("sonoo", 9000)';
if(mysqli_query($conn, $sql)){
 echo "Record inserted successfully";
}else{
echo "Could not insert record: ". mysqli_error($conn);
}
mysqli_close($conn);
?>
```

CRUD Operations On Database(Contd.)

RETRIEVE

The operation RETRIEVE means reading or selecting or getting the data from a table. It is used to go through existing data.

Example:

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$databasename = "geeksforgeeks";
```

CRUD Operations On Database(Contd.)

```
$conn = new mysqli($servername, $username, $password, $databasename);
if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
}
$query = "SELECT * FROM `Student Details`;";
$result = $conn->query($query);
if ($result->num_rows > 0){
  while($row = $result->fetch_assoc()) {
    echo "Roll No:".$row["Roll_No"]."-Name:".$row["Name"]."|City:". $row["City"]."|Age:".$row["Age"]. "<br>";
  }
} else {
        echo "0 results";
}
$conn->close();
?>
```

CRUD Operations On Database(Contd.)

UPDATE
As name suggests this operation helps to UPDATE the existing data of a database.
Example:

```
<?php
$link = mysqli_connect("localhost", "root", “password", "Mydb");
if($link === false){
        die("ERROR: Could not connect.".mysqli_connect_error());
}
$sql = "UPDATE data SET Age='28' WHERE id=201";
if(mysqli_query($link, $sql)){
        echo "Record was updated successfully.";
} else {
        echo "ERROR: Could not able to execute $sql.". mysqli_error($link);
}
mysqli_close($link);
?>
```

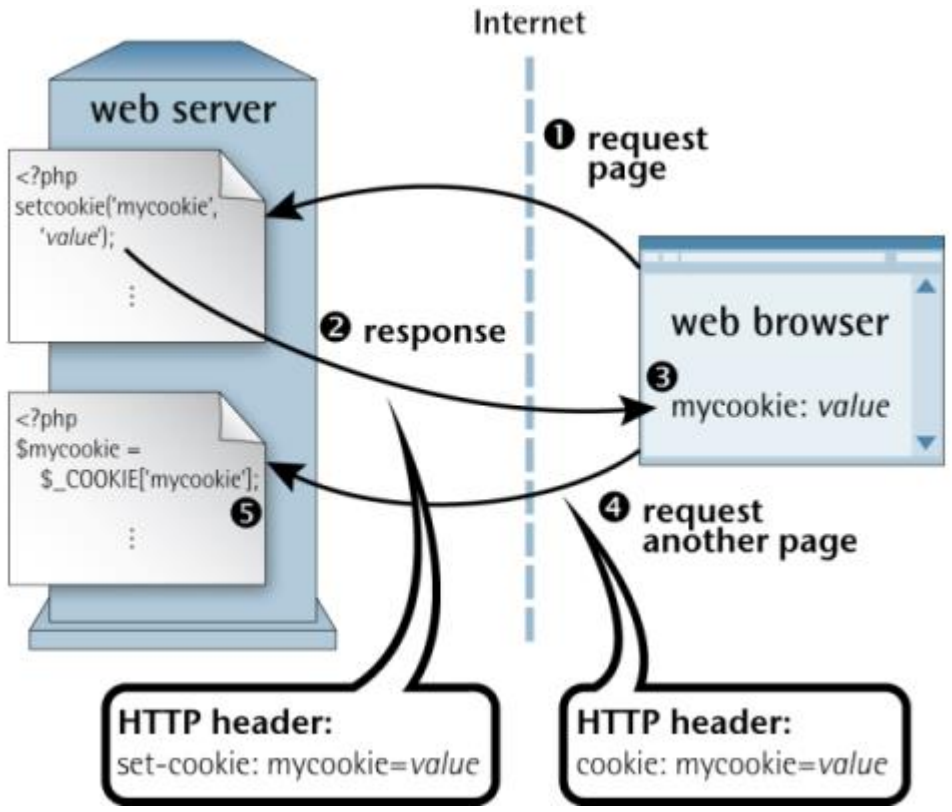
CRUD Operations On Database(Contd.)

DELETE

```
<?php
$link = mysqli_connect("localhost", "root", “password", "Mydb");
if($link === false){
        die("ERROR: Could not connect.". mysqli_connect_error());
}
$sql = "DELETE FROM Data WHERE ID=201";
if(mysqli_query($link, $sql)){
        echo "Record was deleted successfully.";
}
else{
        echo "ERROR: Could not able to execute $sql.". mysqli_error($link);
}
mysqli_close($link);
?>
```

Cookie

A cookie is a tiny file placed on the user's machine by the server. The cookie will be sent each time the same machine requests a page via a browser. Cookie values can be created and retrieved using cookies in PHP. Cookies are text files that are saved on the client computer for the purpose of monitoring. PHP accepts HTTP cookies invisibly.



Purpose of Cookies

To be more concise, cookies are intended to be used for:

Session management: For example, cookies let websites recognize users and recall their individual login information and preferences, such as sports news versus politics.

Personalization: Customized advertising is the main way cookies are used to personalize your sessions. You may view certain items or parts of a site, and cookies use this data to help build targeted ads that you might enjoy. They're also used for language preferences as well.

Purpose of Cookies(Contd.)

Tracking: Shopping sites use cookies to track items users previously viewed, allowing the sites to suggest other goods they might like and keep items in shopping carts while they continue shopping on another part of the website. They will also track and monitor performance analytics, like how many times you visited a page or how much time you spent on a page.

Working with Cookies in PHP

PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the isset() function to find out if the cookie is set:

Working with Cookies in PHP(Contd.)

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html><body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
  echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
</body></html>
```

Working with Cookies in PHP(Contd.)

Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the setcookie() function:

Example

```
<?php
$cookie_name = "user";$cookie_value = "Alex Porter";setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html><body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
  echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
</body></html>
```

Working with Cookies in PHP(Contd.)

Delete a Cookie

To delete a cookie, use the setcookie() function with an expiration date in the past:

Example

```
<?php
setcookie("user", "", time() - 3600); // set the expiration date to one hour ago
?>
<html><body>
<?php
echo "Cookie 'user' is deleted.";
?>
</body></html>
```

Session

A session is a group of user interactions with your website that take place within a given time frame. For example a single session can contain multiple page views, events, social interactions, and ecommerce transactions.

Session in PHP is a way of temporarily storing and making data accessible across all the website pages. It will create a temporary file that stores various session variables and their values. This will be destroyed when you close the website. This file is then available to all the pages of the website to access information about the user.

Session(Contd.)

Purpose Of Session

User Identification: Sessions can be used to identify and track individual users. When a user logs in, you can store their unique identifier (like a user ID) in a session variable. This allows you to recognize the user on subsequent requests and provide a personalized experience.

Data Persistence: Session variables persist across multiple pages during a user's visit to the website. This allows you to store information that needs to be accessed and modified across different parts of your application.

Shopping Carts: Sessions are commonly used in e-commerce websites to store the contents of a user's shopping cart. The cart information is preserved as the user navigates through different pages of the website.

Session(Contd.)

Form Data Persistence: If a user submits a form and there are errors that need to be corrected, session variables can be used to retain the form data so that the user doesn't have to re-enter everything.

Security Tokens: Sessions can be used to store security-related information, such as tokens or keys, to authenticate users or protect against cross-site request forgery (CSRF) attacks.

Customization: You can customize the user experience based on the information stored in session variables. For example, you might adjust the theme, language, or content based on user preferences stored in the session.

Timeouts and Expiry: Sessions can have timeouts or expiry times, meaning that stored information is automatically deleted after a certain period of inactivity. This helps manage server resources and improve security.

Session(Contd.)

Starting a Session

The following things occur when a session is started:

- It creates a random 32 digit hexadecimal value as an identifier for that particular session. The identifier value will look something like 4af5ac6val45rf2d5vre58sd648ce5f7.
- It sends a cookie named PHPSESSID to the user's system. As the name gives out, the PHPSESSID cookie will store the unique session id of the session.
- A temporary file gets created on the server and is stored in the specified directory. It names the file on the hexadecimal id value prefixed with sess_. Thus, the above id example will be held in a file called sess_4af5ac6val45rf2d5vre58sd648ce5f7.

PHP will access the PHPSESSID cookie and get the unique id string to get session variables' values. It will then look into its directory for the file named with that string.

When you close the browser or the website, it terminates the session after a certain period of a predetermined time.

Session(Contd.)

Example:

```
<?php
  session_start();
  if( isset( $_SESSION['counter'] ) ) {
    $_SESSION['counter'] += 1;
  }else {
    $_SESSION['counter'] = 1;
  }
  $my_Msg = "This page is visited ".  $_SESSION['counter'];
  $my_Msg .= " time during this session.";
?>
```

Session(Contd.)

```
<html>
  <head>
    <title>Starting a PHP session</title>
  </head>
  <body>
    <?php  echo ( $my_Msg ); ?>
  </body>
</html>
```

Session(Contd.)

Destroying a Session

Although the web server will terminate the session by default upon closing the browser, you can also destroy it manually. Two functions can help you achieve this.

session_destroy(): Calling this function will eliminate all the session variables

unset(): Calling this function will kill only the specified session variable

Example:

```
<?php
   unset($_SESSION[‘counter’]);
?>
```

Session vs Cookie

Cookie	Session
Cookies are client-side files on a local computer that hold user information.	Sessions are server-side files that contain user data.
Cookies end on the lifetime set by the user.	When the user quits the browser or logs out of the programmed, the session is over.
It can only store a certain amount of info.	It can hold an indefinite quantity of data.
The browser’s cookies have a maximum capacity of 4 KB.	We can keep as much data as we like within a session, however there is a maximum memory restriction of 128 MB that a script may consume at one time.

Session vs Cookie(Contd.)

Cookie	Session
Because cookies are kept on the local computer, we don’t need to run a function to start them.	To begin the session, we must use the session start() method.
Cookies are not secured.	Session are more secured compare than cookies.
Cookies stored data in text file.	Session save data in encrypted form.
Cookies stored on a limited data.	Session stored a unlimited data.

Session vs Cookie(Contd.)

Cookie	Session
In PHP, to get the data from Cookies , $_COOKIES the global variable is used	In PHP , to get the data from Session, $_SESSION the global variable is used
We can set an expiration date to delete the cookie's data. It will automatically delete the data at that specific time.	In PHP, to destroy or remove the data stored within a session, we can use the session_destroy() function, and to unset a specific variable, we can use the unset() function.