CACS 205: Script Language

PREPARED BY KRISHNA PD. ACHARYA

(MECHI MULTIPLE CAMPUS)

Unit-2 Server Side Scripting with Database connectivity-8 hrs.

- Introduction to server side scripting language
- PHP introduction
- Basic PHP syntax
- Comments in PHP
- Variable, PHP operators
- Control structure (if else, switch, all loop)
- Array, for each loop, function
- Form handling, PHP $_GET[],$_POST[],$_REQUEST[],date() function
- PHP include file, File handling, File uploading, PHP Session, sending emails, PHP cookie

Server side script

- Server-side scripting is a technique used in web development which involves employing scripts on a web server which produce a response customized for each user's (client's) request to the website. The alternative is for the web server itself to deliver a static web page.
- Server-side scripting is distinguished from client-side scripting where embedded scripts, such as JavaScript, are run client-side in a web browser, but both techniques are often used together.
- Server-side scripting is often used to provide a customized interface for the user. These scripts may assemble client characteristics for use in customizing the response based on those characteristics, the user's requirements, access rights, etc.
- Server-side scripting also enables the website owner to hide the source code that generates the interface, whereas with client-side scripting, the user has access to all the code received by the client. A down-side to the use of server-side scripting is that the client needs to make further requests over the network to the server in order to show new information to the user via the web browser.

Server side script

- These requests can slow down the experience for the user, place more load on the server, and prevent use of the application when the user is disconnected from the server.
- When the server serves data in a commonly used manner, for example according to the HTTP or FTP protocols, users may have their choice of a number of client programs (most modern web browsers can request and receive data using both of those protocols).
- In the case of more specialized applications, programmers may write their own server, client, and communications protocol, that can only be used with one another.
- Programs that run on a user's local computer without ever sending or receiving data over a network are not considered clients, and so the operations of such programs would not be considered client-side operations.
- Example of server side script PHP, Python, ASP.Net, C#,JSP

Introduction to Dynamic Web Content

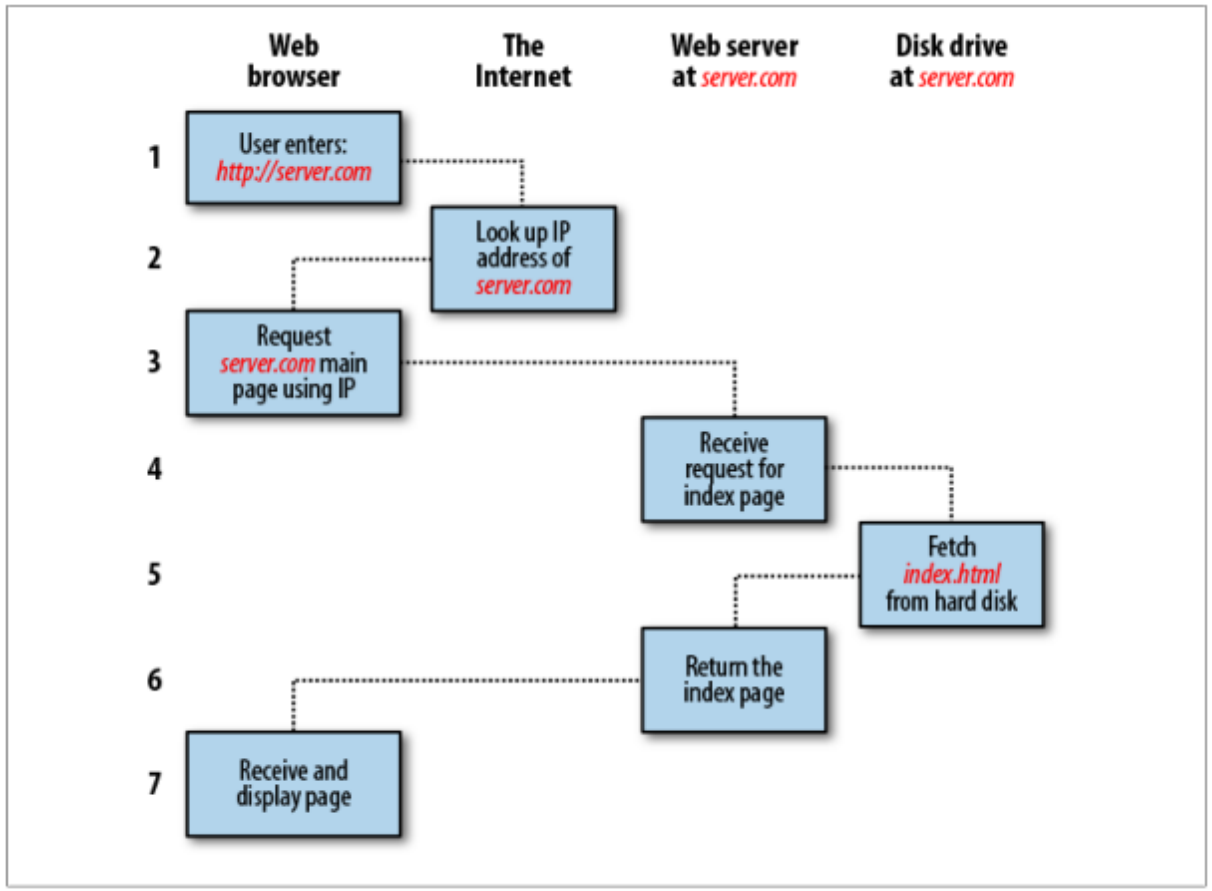


Figure 1-1. The basic client/server request/response sequence

1. You enter http://server.com into your browser's address bar.
2. Your browser looks up the IP address for server.com.
3. Your browser issues a request for the home page at server.com.
4. The request crosses the Internet and arrives at the server.com web server.
5. The web server, having received the request, looks for the web page on its hard disk.
6. The server retrieves the web page and returns it to the browser.
7. Your browser displays the web page.

The Server tier

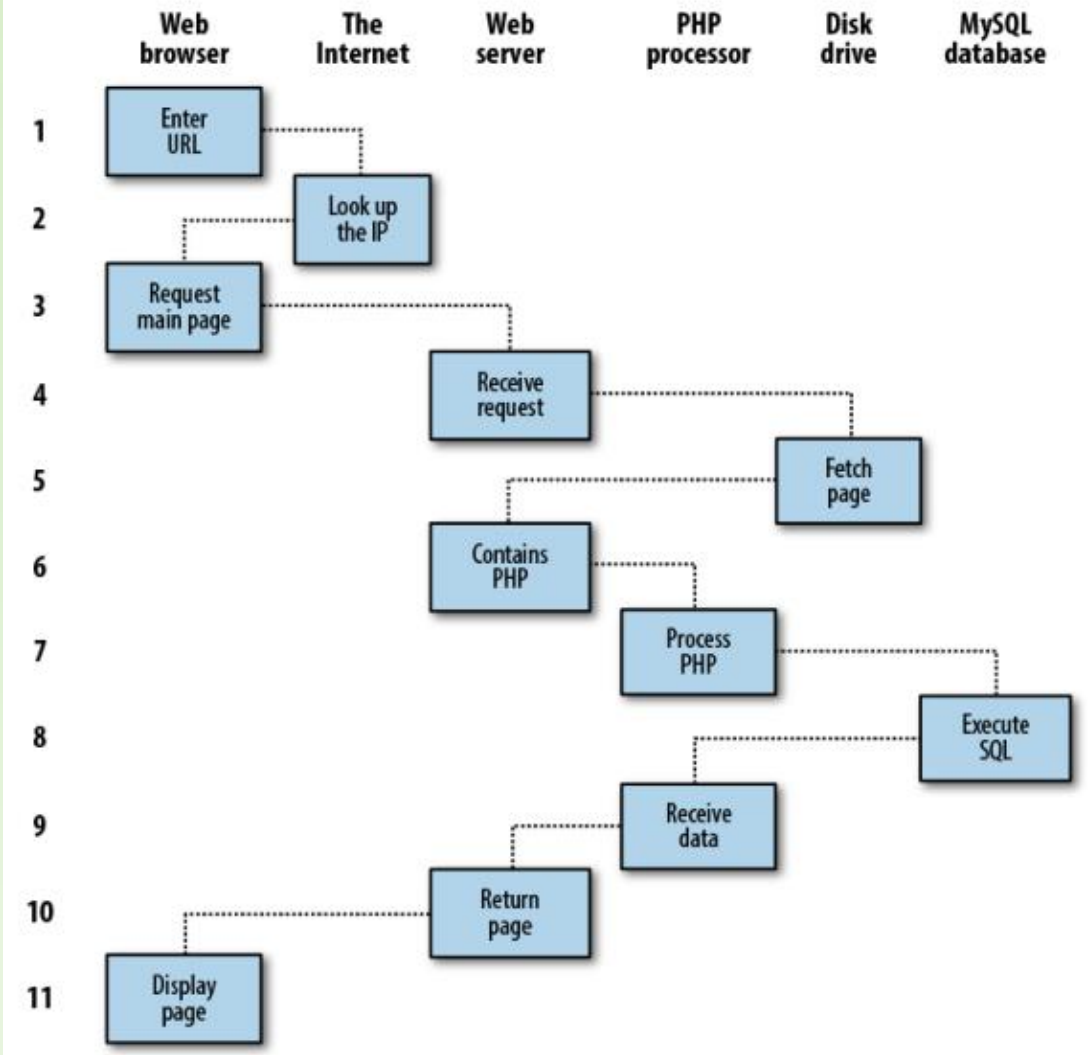


Figure 1-2. A dynamic client/server request/response sequence

1. You enter http://server.com into your browser's address bar.
2. Your browser looks up the IP address for server.com.
3. Your browser issues a request to that address for the web server's home page.
4. The request crosses the Internet and arrives at the server.com web server.
5. The web server, having received the request, fetches the home page from its hard disk.
6. With the home page now in memory, the web server notices that it is a file incorporating PHP scripting and passes the page to the PHP interpreter.
7. The PHP interpreter executes the PHP code.
8. Some of the PHP contains MySQL statements, which the PHP interpreter now passes to the MySQL database engine.
9. The MySQL database returns the results of the statements back to the PHP interpreter.
10. The PHP interpreter returns the results of the executed PHP code, along with the results from the MySQL database, to the web server.
11. The web server returns the page to the requesting client, which displays it.

PHP introduction

Creating a dynamic content

- We can create web content by using Server-side Scripting languages like ASP.Net,C#,PHP etc. Here we are going to discuss PHP Server side scripting.

PHP

- The PHP Hypertext Preprocessor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web based software applications.
- PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.
- PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.
- PHP 7 is the latest stable release. This tutorial uses PHP 7.2.10
- It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)!
- It is deep enough to run the largest social network (Facebook)
- It is also easy enough to be a beginner's first server side language!

PHP introduction

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side.

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data
- With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

PHP introduction

Advantages of PHP over Other Languages

If you're familiar with other server-side languages like ASP.NET or Java, you might be wondering what makes PHP so special. There are several advantages why one should choose PHP.

Easy to learn: PHP is easy to learn and use. For beginner programmers who just started out in web development, PHP is often considered as the preferable choice of language to learn.

Open source: PHP is an open-source project. It is developed and maintained by a worldwide community of developers who make its source code freely available to download and use.

Portability: PHP runs on various platforms such as Microsoft Windows, Linux, Mac OS, etc. and it is compatible with almost all servers used today such Apache, IIS, etc.

Fast Performance: Scripts written in PHP usually execute or runs faster than those written in other scripting languages like ASP, Ruby, Python, Java, etc.

Vast Community: Since PHP is supported by the worldwide community, finding help or documentation related to PHP online is extremely easy.

You know, some huge websites like Facebook, Yahoo, Flickr, and Wikipedia are built using PHP. Most of the major content management systems (CMS), such as WordPress, Drupal, Joomla and Magento are also built in PHP.

PHP introduction

Difference between client-side scripting vs. Server side scripting

Client Side Scripting	Server Side Scripting
The client-side environment used to run scripts is usually a browser.	The **server-side environment** that runs a scripting language is a web server.
The source code is transferred from the web server to the user's computer over the internet and run directly in the browser.	A user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages. This HTML is then sent to the client browser.
Advantages to client-side scripting including faster response times, a more interactive application, and less overhead on the web server.	The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores.
The Disadvantages of client-side scripting are that scripting languages require more time and effort, while the client's browser must support that scripting language.	The disadvantage of server-side processing is the page postback: it can introduce processing overhead that can decrease performance and force the user to wait for the page to be processed and recreated. Once the page is posted back to the server, the client must wait for the server to process the request and send the page back to the client.

Example
`<script>`
`document.getElementById('hello').innerHTML = 'Hello';`
`</script>`
	Example:
	`<h1 id="hello"><?php echo 'Hello'; ?></h1>`

PHP introduction

Basic Syntax

<?php
// PHP code goes here
?>

```
<!DOCTYPE html>
<html>
<body>
<h1>My first PHP page</h1>
<?php
echo "Hello World!";
?>
</body>
</html>
```

Variable names in PHP are case-sensitive
But Keyword is not.

$a,$A are different
For, for are same.
Creating (Declaring) PHP Variables

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

```
<!DOCTYPE html>
<html>
<body>

<?php
// This is a single-line comment

# This is also a single-line comment

/*
This is a multiple-lines comment block
that spans over multiple
lines
*/

// You can also use comments to leave out parts of a code line
$x = 5 /* + 15 */ + 5;
echo $x;
?>

</body>
</html>
```

Comment example

Rules for PHP variables:

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

Date functions

The PHP Date() Function

- The PHP date() function convert a timestamp to a more readable date and time.

date(*format*, *timestamp*)

The format parameter in the date() function is required which specifies the format of returned date and time. However the timestamp is an optional parameter, if not included then current date and time will be used.

```php
<?php
$today = date("d/m/Y");
echo $today;
?>
```

03/08/2019

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

03/08/2019
03-08-2019
03.08.2019

d - Represent day of the month; two digits with leading zeros (01 or 31)
D - Represent day of the week in text as an abbreviation (Mon to Sun)
m - Represent month in numbers with leading zeros (01 or 12)
M - Represent month in text, abbreviated (Jan to Dec)
y - Represent year in two digits (08 or 14)
Y - Represent year in four digits (2008 or 2014)

Date functions

h - Represent hour in 12-hour format with leading zeros (01 to 12)
H - Represent hour in in 24-hour format with leading zeros (00 to 23)
i - Represent minutes with leading zeros (00 to 59)
s - Represent seconds with leading zeros (00 to 59)
a - Represent lowercase ante meridiem and post meridiem (am or pm)
A - Represent uppercase Ante meridiem and Post meridiem (AM or PM)

```php
<?php
echo date("h:i:s") . "<br>";
echo date("F d, Y h:i:s A") . "<br>";
echo date("h:i a");
?>
```

```
12:08:25
August 03, 2019 12:08:25 PM
12:08 pm
```

```php
<?php
$timestamp = time();
echo $timestamp."<br>";
echo(date("F d, Y h:i:s", $timestamp));
?>
```

```
1564834299
August 03, 2019 02:11:39
```

he mktime() function is used to create the timestamp based on a specific date and time. If no date and time is provided, the timestamp for the current date and time is returned.

```
mktime(hour, minute, second, month, day, year)
```

```php
<?php
// Create the timestamp for a particular date
echo mktime(15, 20, 12, 5, 10, 2014);
?>
```

```
1399735212
```

Control flow statement

```php
<!DOCTYPE html>
<html>
<body>

<?php
$txt = "mechi Multiple Campus";
echo "I love $txt!";
  echo "<br>I Love ".$txt;
?>
</body>
</html>
```

- Integers
- Doubles
- Booleans
- NULL
- Strings
- Arrays
- Objects
- Resources

echo gettype($txt) : to find the data type.

define("SITE_URL", "https://www.mechicampus.edu.np/");

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

D-day in short form like Fri

```php
<html>
<head>
<title>Hello World</title>
</head>
<body>
<?php
echo " Today is " . date("l") . ". ";
?>
</body>
</html>
```

Control flow statement

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

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

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

```php
<?php echo ($age < 18) ? 'Child' : 'Adult'; ?>
```

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <title>PHP switch-case Statement</title>
</head>
<body>
<?php
$today = date("D");
switch($today){
    case "Mon":
        echo "Today is Monday. Clean your house.";
        break;
    case "Tue":
        echo "Today is Tuesday. Buy some food.";
        break;

    default:
        echo "No information available for that day.";
        break;
}
?>
</body>
</html>
```

```php
<?php
$i = 1;
while($i <= 3){
    $i++;
    echo "The number is " . $i . "<br>";
}
?>
```

```php
<?php
for($i=1; $i<=3; $i++){
    echo "The number is " . $i . "<br>";
}
?>
```

```php
<?php
$i = 1;
do{
    $i++;
    echo "The number is " . $i . "<br>";
}
while($i <= 3);
?>
```

Control flow statement

```php
<?php
$colors = array("Red", "Green", "Blue");

// Loop through colors array
foreach($colors as $value){
    echo $value . "<br>";
}
?>
```

```php
<?php
// Defining function
function getSum($num1, $num2){
    $total = $num1 + $num2;
    return $total;
}

// Printing returned value
echo getSum(5, 10); // Outputs: 15
?>
```

```php
<?php
// Defining function
function myfunction(){
    echo "Today is " . date('l');
}
// Calling function
myfunction();
?>
```

```php
<?php
$superhero = array(
    "name" => "krishna",
    "email" => "krishna@mail.com",
    "age" => 18
);
foreach($superhero as $key => $value){
    echo $key . " : " . $value . "<br>";
}
?>
```

```php
<?php
function getSum($num1, $num2){
  $sum = $num1 + $num2;
  echo "Sum of the two numbers $num1 and $num2 is : $sum";
}

getSum(10, 20);
?>
```

```php
<?php declare(strict_types=1); // strict requirement
function addNumbers(float $a, float $b) : float {
    return $a + $b;
}
echo addNumbers(1.2, 5.2);
?>
```

```php
<?php
function getSum(int $num1, int $num2){
  $sum = $num1 + $num2;
  echo "Sum of the two numbers: ".$sum;
}

getSum(10, 15); ?>
```

Control flow statement

```php
<?php
// Define a multidimensional array
$contacts = array(
    array(
        "name" => "Krishna Acharya",
        "email" => "krishnaxt@mail.com",
    ),
    array(
        "name" => "Clark Kent",
        "email" => "clarkkent@mail.com",
    ),
    array(
        "name" => "Harry Potter",
        "email" => "harrypotter@mail.com",
    )
);
// Access nested value
echo "Krishna's Email-id is: " . $contacts[0]["email"];
?>
```

`sort()` and `rsort()` — For sorting indexed arrays

`asort()` and `arsort()` — For sorting associative arrays by value

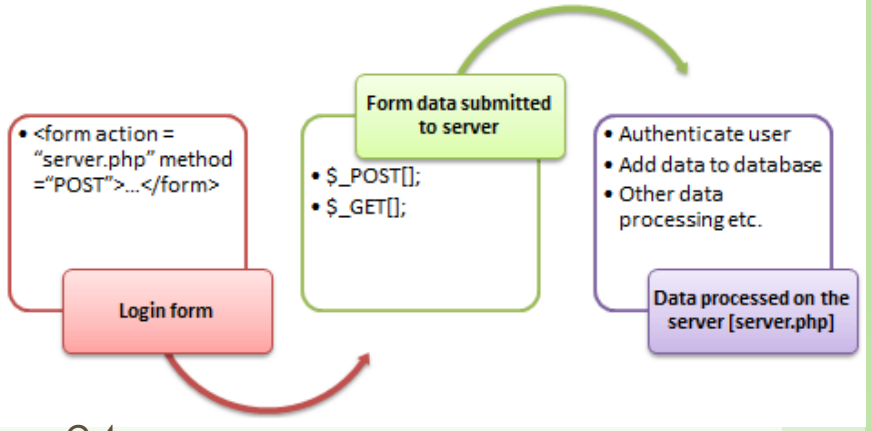
`ksort()` and `krsort()` — For sorting associative arrays by key

```php
<?php
// Define array
$numbers = array(1, 2, 2.5, 4, 7, 10);

// Sorting and printing array
rsort($numbers);
print_r($numbers);
?>
```

Methods of Sending Information to Server

A web browser communicates with the server typically using one of the two HTTP (Hypertext Transfer Protocol) methods; GET,POST,REQUEST. In GET method the data is sent as URL parameters that are usually strings of name and value pairs separated by ampersands (&).



http://www.mechicampus.edu.np/action.php?name=mohan&age=24

- Since the data sent by the GET method are displayed in the URL, it is possible to bookmark the page with specific query string values.
- The GET method is not suitable for passing sensitive information such as the username and password, because these are fully visible in the URL query string as well as potentially stored in the client browser's memory as a visited page.
- Because the GET method assigns data to a server environment variable, the length of the URL is limited. So, there is a limitation for the total data to be sent.
- $REQUEST contains: $COOKIE, $GET, and $POST variables.

Methods of Sending Information to Server

form method="post" action="action.php">
data goes here .....
</form>
Now , In this case action.php is destination page where user will have records .
in PHP : there are 3 methods to retrieve data from receiving page.
1) $_GET
2) $_POST
3) R_REQUEST
$_GET :
$_GET is a super global array which is an inbuilt array. Which collects values from a form sent with method="get" as well it collects values from URL also. (e.g. http://www.mmc.com/index.php?data=123)
You can print $_GET with inbuilt function in php i.e (print_r($_GET)).
IMP : Information sent from this method will be visible in (address bar of any browser) and has limitation on characters .
As per W3C Community limitation can be extended up to 4000 characters
Myth: Search services will not index anything with a "?" in the URI.
Myth: URIs cannot be longer than 256 characters
for details : visit : About $_GET limitation :
You can read more details about $_GET.
$_POST :
$_POST is a super global array which is an inbuilt array. Which collects values from a form sent with method="post".
You can print $_GET with inbuilt function in php i.e (print_r($_POST)).
IMP : Information received from POST method is always invisible in (address bar of any browser) . It has more limit on sending characters to action page.
This limitation can be set in php.ini file
Note: We can change by setting the POST_MAX_SIZE in the php.ini file . by default we will find this value with 8MB in most of the php.ini file.
$_REQUEST :
$_REQUEST is a super global array which is an inbuilt array. Which collects values from a form sent with either method="post" or method="get" .
You can print $_REQUEST with inbuilt function in php i.e (print_r($_REQUEST)).
IMP : One of the best way of getting data on action page , here we did not require to check that which method we have used to pass data from sending page. Even $_REQUEST get records from the COOKIE also.

GET vs POST

	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
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

Methods of Sending Information to Server

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Example of PHP GET method</title>
</head>
<body>
<?php
if(isset($_GET["name"])){
    echo "<p>Hi, " . $_GET["name"] . "</p>";
}
?>
<form method="get" action="<?php echo $_SERVER["PHP_SELF"];?>">
    <label for="inputName">Name:</label>
    <input type="text" name="name" id="inputName">
    <input type="submit" value="Submit">
</form>
```

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Example of PHP POST method</title>
</head>
<body>
<?php
if(isset($_POST["name"])){
    echo "<p>Hi, " . $_POST["name"] . "</p>";
}
?>
<form method="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
    <label for="inputName">Name:</label>
    <input type="text" name="name" id="inputName">
    <input type="submit" value="Submit">
</form>
</body>
```

$_REQUEST["name"]

PHP Include and Require Files

- The include() and require() statement allow us to include the code contained in a PHP file within another PHP file.
- Including a file produces the same result as copying the script from the file specified and pasted into the location where it is called.
- You can save a lot of time and work through including files —Just store a block of code in a separate file and include it wherever you want using the include() and require() statements instead of typing the entire block of code multiple times.
- A typical example is including the header, footer and menu file within all the pages of a website.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Tutorial Republic</title>
</head>
<body>
<?php include "header.php"; ?>
<?php include "menu.php"; ?>
    <h1>Welcome to Our Website!</h1>
    <p>Here you will find lots of useful information.</p>
<?php include "footer.php"; ?>
</body>
</html>
```

PHP Include and Require Files

Difference Between include and require Statements

- The only difference is — the include() statement will only generate a PHP warning but allow script execution to continue if the file to be included can't be found, whereas the require() statement will generate a fatal error and stops the script execution.
- **Tip**: It is recommended to use the require() statement if you're including the library files or files containing the functions and configuration variables that are essential for running your application, such as database configuration file.

- The **include_once** and **require_once** statements will only include the file once even if asked to include it a second time i.e. if the specified file has already been included in a previous statement, the file is not included again.

PHP Basics of File Handling

- File handling is needed for any application. For some tasks to be done file needs to be processed. File handling in PHP is similar as file handling is done by using any programming language like C. PHP has many functions to work with normal files. Those functions are:

1) fopen() – PHP fopen() function is used to open a file. First parameter of fopen() contains name of the file which is to be opened and second parameter tells about mode in which file needs to be opened, e.g.,

```
<?php
$file = fopen("demo.txt",'w');
?>
```

Read:

```
<?php
$filename = "demo.txt";
$file = fopen( $filename, 'r' );
$size = filesize( $filename );
$filedata = fread( $file, $size );
?>
```

Files can be opened in any of the following modes :

- **"w"** – Opens a file for write only. If file not exist then new file is created and if file already exists then contents of file is erased.
- **"r"** – File is opened for read only.
- **"a"** – File is opened for write only. File pointer points to end of file. Existing data in file is preserved.
- **"w+"** – Opens file for read and write. If file not exist then new file is created and if file already exists then contents of file is erased.
- **"r+"** – File is opened for read/write.
- **"a+"** – File is opened for write/read. File pointer points to end of file. Existing data in file is preserved. If file is not there then new file is created.
- **"x"** – New file is created for write only.

PHP Basics of File Handling

2) **fread() –** After file is opened using fopen() the contents of data are read using fread(). It takes two arguments. One is file pointer and another is file size in bytes, e.g.

3) **fwrite()** – New file can be created or text can be appended to an existing file using fwrite() function. Arguments for fwrite() function are file pointer and text that is to written to file. It can contain optional third argument where length of text to written is specified.

```
<?php
$file = fopen("demo.txt", 'w');
$text = "Hello world\n";
fwrite($file, $text);
?>
```

4) **fclose() –** file is closed using fclose() function. Its argument is file which needs to be closed.

```
<?php
$file = fopen("demo.txt", 'r');
//some code to be executed
fclose($file);
?>
```

PHP Basics of File Handling

```php
<?php
$file = "file.txt";

// Check the existence of file
if(file_exists($file)){
    // Attempt to rename the file
    if(rename($file, "newfile.txt")){
        echo "File renamed successfully.";
    } else{
        echo "ERROR: File cannot be renamed.";
    }
} else{
    echo "ERROR: File does not exist.";
}
?>
```

```php
<?php
$file = "note.txt";

// Check the existence of file
if(file_exists($file)){
    // Attempt to delete the file
    if(unlink($file)){
        echo "File removed successfully.";
    } else{
        echo "ERROR: File cannot be removed.";
    }
} else{
    echo "ERROR: File does not exist.";
}
?>
```

PHP Basics of File Handling

```php
<?php
3 references | 0 implementations
class Person {
    1 reference
    public $name;
    1 reference
    public $age;
    1 reference
    public $email;
    3 references | 0 overrides
    public function __construct($name, $age, $email) {
        $this->name = $name;
        $this->age = $age;
        $this->email = $email;
    }
}

// Create an array of Person objects
$persons = array(
    new Person('Ram Rai', 20, 'ram@example.com'),
    new Person('Sita Adhikari', 25, 'sit@example.com'),
    new Person('Mohan Bista', 20, 'mohan@example.com')
);
```

```php
// Serialize the array of objects
$serializedData = serialize($persons);

// Write the serialized data to a file
$file = 'persons.txt';
file_put_contents($file, $serializedData);
echo "Records written to file successfully.";

// Read the serialized data from the file
$serializedData = file_get_contents($file);
// Unserialize the data into an array of objects
$persons = unserialize($serializedData);

// Access and display the records
echo("<br>");
foreach ($persons as $person) {
    echo "Name: " . $person->name . "<br>";
    echo "Age: " . $person->age . "<br>";
    echo "Email: " . $person->email . "<br>";
    echo "<br>";
}
?>
```

Exception handling

- Exception handling is a programming technique used to **gracefully handle** and manage **runtime errors or exceptional situations** that may occur during the execution of a program. It involves identifying, capturing, and handling exceptions to prevent them from causing the program to terminate **abruptly or exhibit unexpected behavior**.
- Exceptions are **objects that represent errors or exceptional conditions**, such as **division by zero, file not found, or database connection failure**. When an exceptional situation occurs, an exception is thrown, which **interrupts the normal flow of the program** and **transfers control to an exception handler.**
- **Throwing**: In response to an exceptional situation, an exception is thrown using the throw statement. This indicates that an error or unexpected condition has occurred.
- **Catching**: The **try-catch block** is used to catch and handle the thrown exception. The code within the try block is monitored for exceptions, and if one occurs, it is caught by the corresponding catch block.
- **Handling**: The catch block contains code that handles the exception, such as logging the error, displaying a meaningful error message, or taking alternative actions to recover from the exceptional situation.

Exception handling

- Exception handling differs from error handling in the following ways:
- Error handling typically refers to the process of dealing with **system-level errors or fatal errors** that may occur **outside the control of the program**, such as **memory allocation errors or stack overflow**. Error handling is usually done at the **system level** or by using error codes and error handling functions.
- Exception handling, on the other hand, is specifically designed to **handle runtime errors or exceptional situations** that occur within the program's execution. It focuses on catching and handling exceptions, which are specific types of objects representing exceptional conditions.
- Exception handling allows for **more fine-grained control and recovery from specific** exceptional situations, while error handling is generally used for handling system-level errors that may require **termination of the program**.
- Overall, exception handling provides a mechanism to handle and **recover from exceptional situations within the program's** execution flow, ensuring better program stability and reliability.

Examples

```php
<?
try {
    // Code that may throw an exception
    $numerator = 10;
    $denominator = 0;
    $result = $numerator / $denominator;
    echo "Result: " . $result;
} catch (DivisionByZeroError $e) {
    // Handle specific exception: DivisionByZeroError
    echo "Division by zero error: " . $e->getMessage();
} catch (Exception $e) {
    // Handle generic exception
    echo "Exception occurred: " . $e->getMessage();
} finally {
    // Code to be executed regardless of exception
    echo "Finally block executed.";
}
?>
```

```php
<?
1 reference
function divide($numerator, $denominator) {
    if ($denominator === 0) {
        throw new Exception("Division by zero is not allowed.");
    }
    return $numerator / $denominator;
}

try {
    $result = divide(10, 0);
    echo "Result: " . $result;
} catch (Exception $e) {
    echo "Exception occurred: " . $e->getMessage();
}
?>
```

Session and State

- Web is Stateless, that means the web server does not know who you are and what you do, because the HTTP address doesn't maintain state.
- If user inserts some information, and move to the next page, that data will be lost and user would not able to retrieve the information so we need to store the information about user.
- Session provides that facility to store information on server memory. Session variables hold information about one single user, and are available to all pages in one application. By default, session variables last until the user closes the browser or destroy the session variable in php code.
- A PHP session stores data on the server rather than user's computer. In a session based environment, every user is identified through a unique number called **session identifier** or SID. This unique session ID is used to link each user with their own information on the server like emails, posts, etc.

Session and State

Advantages

- It helps to maintain user states and data to all over the application.
- It can easily be implemented and we can store any kind of object.
- Stores every client data separately.
- Session is secure and transparent from user.

Disadvantages

- Performance overhead in case of large volume of user, because of session data stored in server memory.
- Overhead involved in serializing and De-Serializing session Data. because In case of StateServer and SQLServer session mode we need to serialize the object before store.

Session and State

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

```php
<?php
session_start();
?>
<html>
<body>
<a href="sessionout.php">Clear Session</a> <br>
<?php
echo "User is: ".$_SESSION["user"].
     "<br>Password: ".$_SESSION["pass"]."<br>";
?>
</body>
</html>
```

```php
<?php
session_start();
//session_destroy();
//Is Used To Destroy All Sessions

if(isset($_SESSION['user'])) //specific
{
  unset($_SESSION['user']);
  unset($_SESSION['pass']);
}
  echo "Clear Session successfully!";
 ?>
```

Cookie

- A cookie is a small text file that lets us store a small amount of data (nearly 4KB) on the user's computer. They are typically used to keeping track of information such as username that the site can retrieve to personalize the page when user visit the website next time.
- Each time the browser requests a page to the server, all the data in the cookie is automatically sent to the server within the request.

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

```
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 sec
setcookie("CookieName", "CookieValue", time()+1*60*60, "/mypath/", "mydomain.com", 1);
```

setcookie("user", "Administrator", time()+30*24*60*60);

Cookie

Advantages of using cookies.

- Cookies are simple to use and implement.
- Occupies less memory, do not require any server resources and are stored on the user's computer so no extra burden on server.
- We can configure cookies to expire when the browser session ends (session cookies) or they can exist for a specified length of time on the client's computer (persistent cookies).
- Cookies persist a much longer period of time than Session state.

Disadvantages of using cookies

- As mentioned previously, cookies are not secure as they are stored in clear text they may pose a possible security risk as anyone can open and tamper with cookies. You can manually encrypt and decrypt cookies, but it requires extra coding and can affect application performance because of the time that is required for encryption and decryption
- Several limitations exist on the size of the cookie text(4kb in general), number of cookies(20 per site in general), etc.
- User has the option of disabling cookies on his computer from browser's setting .
- Cookies will not work if the security level is set to high in the browser.
- Users can delete a cookies.
- Users browser can refuse cookies, so your code has to anticipate that possibility.
- Complex type of data not allowed (e.g. dataset etc). It allows only plain text (i.e. cookie allows only string content)

Cookie

```php
<html>
<body>
<?php
   setcookie("user", "krishna");
   echo "<br/>Cookie Value: " . $_COOKIE["user"];
   echo "<br><a href='viewcookie.php'>Visit next page</a>";
?>
</body>
</html>
```

```php
<html>
<body>
<?php
if(!isset($_COOKIE["user"])) {
    echo "Sorry, cookie is not found!<br>";
    echo "<font color='red' size='20'>Dos Not have previleg to access this page</font>";
} else {
    echo "<br/>Cookie Value: " . $_COOKIE["user"];
    echo "<br><a href='clearCookie.php'>Clear Cookie</a>";
}
?>
</body>
</html>
```

```php
<?php
if(isset($_COOKIE["user"])) {
setcookie("user", "", time() - 3600);
// set the expiration date to one hour ago
echo "cookie Has been cleared";
}
?>
```

Cookie Vs Session

Cookies vs Sessions

- Both cookies and sessions are used for storing persistent data. But there are few differences.
- Sessions are stored on server side. Cookies are on the client side.
- Sessions are closed when the user closes his browser. For cookies, you can set time that when it will be expired.
- Sessions are safer than cookies. Because, since stored on client's computer, there are ways to modify or manipulate cookies.

Application:

Session management

- Logins, shopping carts, game scores, or anything else the server should remember.

Personalization

- User preferences, themes, and other settings.

Tracking

- Recording and analyzing user behavior.

Configuration setting

C:\xampp\sendmail\sendmail.ini

```
smtp_port=587
auth_username=krishnaxt@gmail.com
auth_password=123543543457dgdg@
```

C:\xampp\php\php.inf

```
sendmail_path ="\"C:\xampp\sendmail\sendmail.exe\" -t"
```

PHP built-in mail() function for creating and sending email messages to one or more recipients dynamically from your PHP application either in a plain-text form or formatted HTML. The basic syntax of this function can be given with:

mail(to, subject, message, headers, parameters)

To: The recipient's email address.
Subject: Subject of the email to be sent. This parameter i.e. the subject line cannot contain any newline character (\n).
Message: Defines the message to be sent. Each line should be separated with a line feed-LF (\n). Lines should not exceed 70 characters.
***Optional*:** The following parameters are optional
headers: This is typically used to add extra headers such as "From", "Cc", "Bcc". The additional headers should be separated with a carriage return plus a line feed-CRLF (\r\n).
Parameters: Used to pass additional parameters.

Mail function in php

```php
<?php
$to = 'krishnaxt@gmail.com';
$subject = 'To submit your document!';
$from = 'sharmame@gmail.com';

// To send HTML mail, the Content-type header must be set
$headers  = 'MIME-Version: 1.0' . "\r\n";
$headers .= 'Content-type: text/html; charset=iso-8859-1' . "\r\n";

// Create email headers
$headers .= 'From: '.$from."\r\n".
    'Reply-To: '.$from."\r\n" .
    'X-Mailer: PHP/' . phpversion();

// Compose a simple HTML email message
$message = '<html><body>';
$message .= '<h1 style="color:#f40;">Hi Shrma!</h1>';
$message .= '<p style="color:#080;font-size:18px;">How are you?</p>';
$message .= '</body></html>';
```

```php
// Sending email
if(mail($to, $subject, $message, $headers)){
    echo 'Your mail has been sent successfully.';
} else{
    echo 'Unable to send email. Please try again.';
}
?>
```

Upload setting

- Configure The "php.ini" File
- **file_uploads = On**
- Some rules to follow for the HTML form:
- 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
- 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

Upload file in php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>File Upload Form</title>
</head>
<body>
    <form action="uploadmanager.php" method="post" enctype="multipart/form-data"
        <h2>Upload File</h2>
        <label for="fileSelect">Filename:</label>
        <input type="file" name="photo" id="fileSelect">
        <input type="submit" name="submit" value="Upload">
        <p><strong>Note:</strong>
        Only .jpg, .jpeg, .gif, .png formats allowed to a max size of 5 MB.</p>
    </form>
</body>
</html>
```

Upload file in php

```php
<?php
// Check if the form was submitted
if($_SERVER["REQUEST_METHOD"] == "POST"){
    // Check if file was uploaded without errors
    if(isset($_FILES["photo"]) && $_FILES["photo"]["error"] == 0){
        $allowed = array("jpg" => "image/jpg", "jpeg" => "image/jpeg", "gif" => "image/gif", "png" => "image/png");
        $filename = $_FILES["photo"]["name"];
        $filetype = $_FILES["photo"]["type"];
        $filesize = $_FILES["photo"]["size"];

        // Verify file extension
        $ext = pathinfo($filename, PATHINFO_EXTENSION);
        if(!array_key_exists($ext, $allowed)) die("Error: Please select a valid file format.");

        // Verify file size - 5MB maximum
        $maxsize = 5 * 1024 * 1024;
        if($filesize > $maxsize) die("Error: File size is larger than the allowed limit.");
        // Verify MYME type of the file
        if(in_array($filetype, $allowed)){
            //Check whether file exists before uploading it
            if(file_exists("C:/xampp/upload/".$filename)){
                echo $filename." is already exists.";
            } else{
                move_uploaded_file($_FILES["photo"]["tmp_name"], "C:/xampp/upload/" . $filename);
                echo "Your file was uploaded successfully.";
            }
        } else{
            echo "Error: There was a problem uploading your file. Please try again.";
        }
    } else{
        echo "Error: " . $_FILES["photo"]["error"];
        }
    }
?>
```

Form validation in php

Validating and Sanitizing Data with Filters

- Sanitizing and validating user input is one of the most common tasks in a web application. To make this task easier PHP provides native filter extension that you can use to sanitize or validate data such as e-mail addresses, URL.s, IP addresses, etc.

- **filter_var(variable, filter, options)**

1.Sanitize a String

```php
<?php
// Sample user comment
$comment = "<h1>Hey there! How are you doing today?</h1>";

// Sanitize and print comment string
$sanitizedComment = filter_var($comment, FILTER_SANITIZE_STRING);
echo $sanitizedComment;
?>
```

Hey there! How are you doing today?

2. Validate Integer Values

```php
<?php
// Sample integer value
$int = 20;

// Validate sample integer value
if(filter_var($int, FILTER_VALIDATE_INT)){
    echo "The <b>$int</b> is a valid integer";
} else{
    echo "The <b>$int</b> is not a valid integer";
}
?>
```

Form validation in php

```php
<?php
// Sample integer value
$int = 0;

// Validate sample integer value
if(filter_var($int, FILTER_VALIDATE_INT) === 0 || filter_var($int, FILTER_VALIDATE_INT)){
    echo "The <b>$int</b> is a valid integer";
} else{
    echo "The <b>$int</b> is not a valid integer";
}
?>
```

```php
<?php
// Sample IP address
$ip = "172.16.254.1";

// Validate sample IP address
if(filter_var($ip, FILTER_VALIDATE_IP)){
    echo "The <b>$ip</b> is a valid IP address";
} else {
    echo "The <b>$ip</b> is not a valid IP address";
}
?>
```

```php
<?php
// Sample email address
$email = "someone@@example.com";

// Remove all illegal characters from email
$email = filter_var($email, FILTER_SANITIZE_EMAIL);

// Validate e-mail address
if(filter_var($email, FILTER_VALIDATE_EMAIL)){
    echo "The <b>$email</b> is a valid email address";
} else{
    echo "The <b>$email</b> is not a valid email address";
}
?>
```

Form validation in php

```php
<?php
// Sample website url
$url = "http://www.example.com?topic=filters";

// Validate website url for query string
if(filter_var($url, FILTER_VALIDATE_URL, FILTER_FLAG_QUERY_REQUIRED)){
    echo "The <b>$url</b> contains query string";
} else{
    echo "The <b>$url</b> does not contain query string";
}
?>
```

```php
<?php
// Sample integer value
$int = 75;

// Validate sample integer value
if(filter_var($int, FILTER_VALIDATE_INT, array("options" => array("min_range"
=> 0,"max_range" => 100)))){
    echo "The <b>$int</b> is within the range of 0 to 100";
} else{
    echo "The <b>$int</b> is not within the range of 0 to 100";
}
?>
```

Form validation in php

What is Regular Expression

- Regular Expressions, commonly known as "regex" or "RegExp", are a specially formatted text strings used to find patterns in text. Regular expressions are one of the most powerful tools available today for effective and efficient text processing and manipulations. For example, it can be used to verify whether the format of data i.e. name, email, phone number, etc. entered by the user was correct or not, find or replace matching string within text content, and so on.

Function	What it Does
preg_match()	Perform a regular expression match.
preg_match_all()	Perform a global regular expression match.
preg_replace()	Perform a regular expression search and replace.
preg_grep()	Returns the elements of the input array that matched the pattern.
preg_split()	Splits up a string into substrings using a regular expression.
preg_quote()	Quote regular expression characters found within a string.

Form validation in php

```php
<?php
$pattern = "/ca[kf]e/";
$text = "He was eating cake in the cafe.";
if(preg_match($pattern, $text)){
    echo "Match found!";
} else{
    echo "Match not found.";
}
?>
```

```php
<?php
$pattern = "/\s/";
$replacement = "-";
$text = "Earth revolves around\nthe\tSun";
// Replace spaces, newlines and tabs
echo preg_replace($pattern, $replacement, $text);
echo "<br>";
// Replace only spaces
echo str_replace(" ", "-", $text);
?>
```

```php
<?php
$pattern = "/ca[kf]e/";
$text = "He was eating cake in the cafe.";
$matches = preg_match_all($pattern, $text, $array);
echo $matches . " matches were found.";
?>
```

```php
<?php
$pattern = "/[\s,]+/";
$text = "My favourite colors are red, green and blue";
$parts = preg_split($pattern, $text);

// Loop through parts array and display substrings
foreach($parts as $part){
    echo $part . "<br>";
}
?>
```

```php
<?php
$pattern = "/^J/";
$names = array("Jhon Carter", "Clark Kent", "John Rambo");
$matches = preg_grep($pattern, $names);

// Loop through matches array and display matched names
foreach($matches as $match){
    echo $match . "<br>";
}
?>
```

Form validation in php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Contact Form</title>
    <style type="text/css">
        .error{ color: red; }
        .success{ color: green; }
    </style>
</head>
<body>
    <h2>Contact Us</h2>
    <p>Please fill in this form and send us.</p>
    <form action="contact.php" method="post">
        <p>
            <label for="inputName">Name:<sup>*</sup></label>
            <input type="text" name="name" id="inputName" value="">
            <span class="error"></span>
        </p>
        <p>
            <label for="inputEmail">Email:<sup>*</sup></label>
            <input type="text" name="email" id="inputEmail" value="">
            <span class="error"></span>
        </p>
        <p>
            <label for="inputSubject">Subject:</label>
            <input type="text" name="subject" id="inputSubject" value="">
        </p>
        <p>
            <label for="inputComment">Message:<sup>*</sup></label>
            <textarea name="message" id="inputComment" rows="5" cols="30"></textarea>
            <span class="error"></span>
        </p>
        <input type="submit" value="Send">
        <input type="reset" value="Reset">
    </form>
</body>
</html>
```