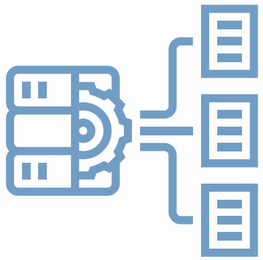
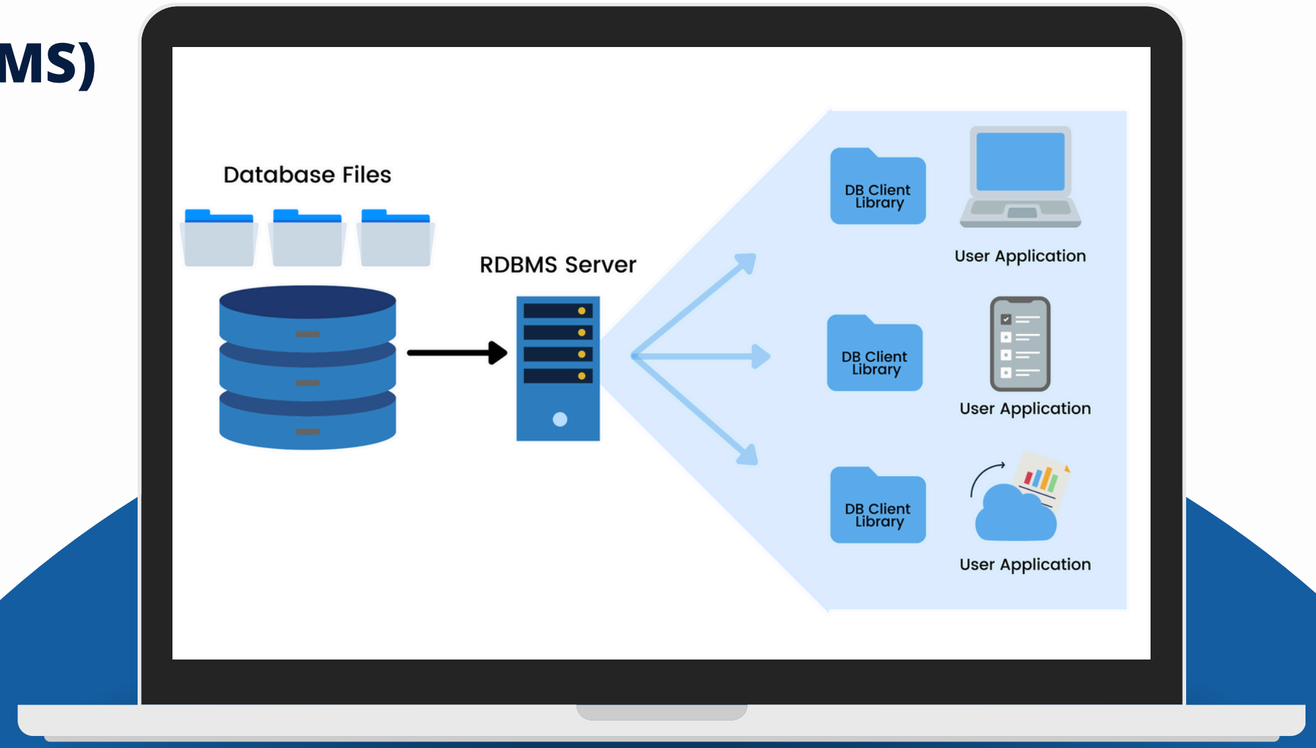


Database Management System (DBMS)



Shaeed Al Hasan - Siam
siamshaeed@gmail.com
01787972797

Overview

- PHP 00-00
- MYSql 00-00

What is Database ?

A database is an organized collection of data that allows for easy access and management. It structures data into tables, rows, and columns, and uses indexing to facilitate the quick retrieval of relevant information.

CustomerID	FirstName	LastName	Email	Phone
1	John	Doe	john.doe@email.com	123-456-7890
2	Jane	Smith	jane.smith@email.com	987-654-3210

Overview of Database Management Systems (DBMS)

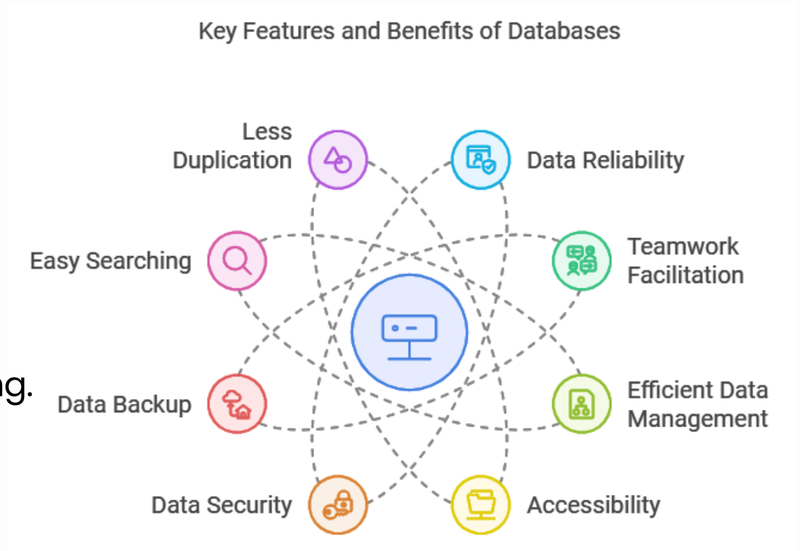
A Database Management System (DBMS) is software that simplifies creating, managing, and using databases. It links users to the database, making it easy to store, retrieve, and process data.

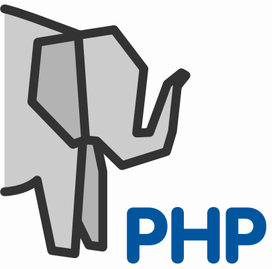
Core functionalities of a DBMS include:

- **Data Storage:** The DBMS keeps data stored securely and organized, making it easy to access and manage.
- **Data Manipulation:** Users can easily insert, update, delete, and retrieve data, allowing for dynamic interaction with the information.
- **Data Security:** A strong DBMS keeps stored data safe from unauthorized access
- **User Access Control:** System sets different access levels for users to maintain data security and privacy.
- **Data Backup and Recovery:** The DBMS offers tools for regularly backing up data and restoring it if it's lost or damaged.

Importance of Databases in Modern Applications

- **Organized Storage:** Keeps data organized for easy access.
- **Reliable Information:** Makes sure data is correct and up-to-date.
- **Growth-Friendly:** Easily handles more data and users as needed.
- **Data Security:** Keeps sensitive information safe with secure access.
- **Multiple Users:** Allows several people to access data at the same time for better teamwork.
- **Data Backup:** Regularly saves copies of data for quick recovery if something goes wrong.
- **Easy Searching:** Lets users find specific information quickly and easily.
- **Less Duplication:** Reduces repeated data storage, keeping everything organized.
- **Easy Sharing:** Makes it simple to share data between different applications and users.





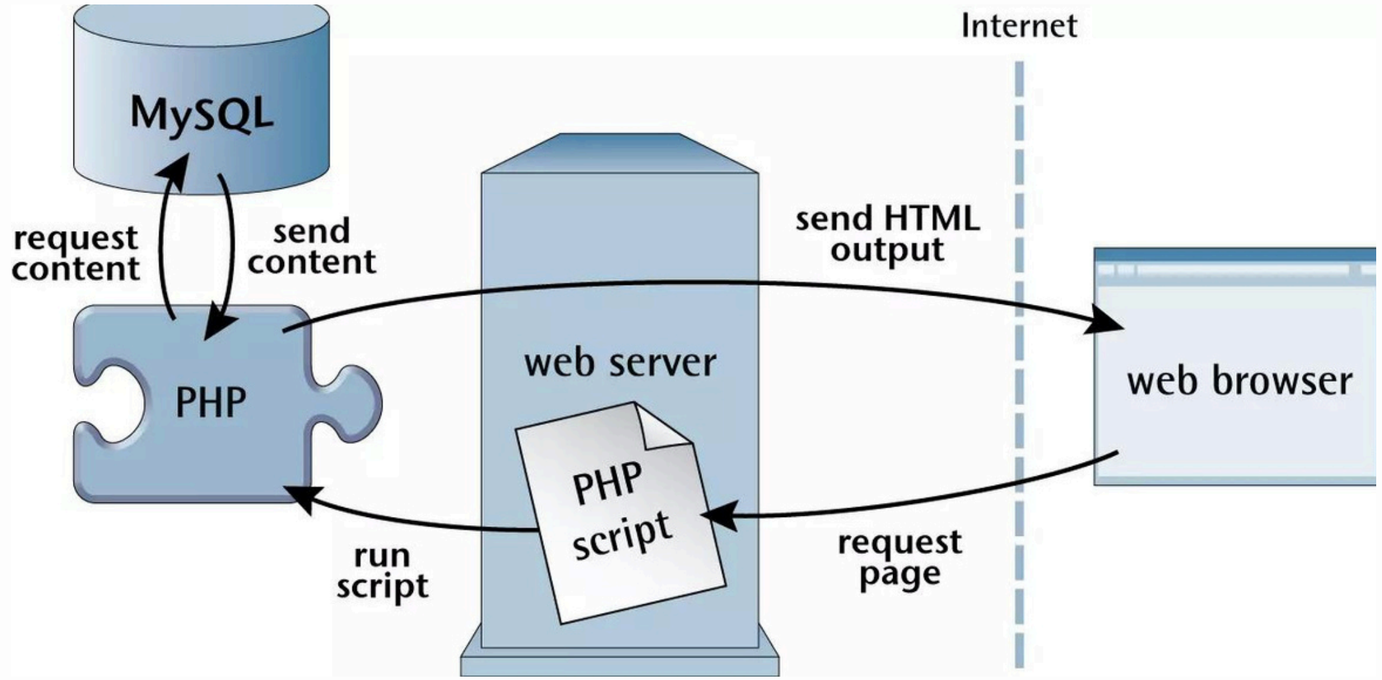
What is PHP

PHP stands for Hypertext Preprocessor. It is an open-source, server-side scripting language, specifically designed for web development. PHP is particularly effective for creating dynamic and interactive web pages, making it a popular choice for building modern websites and applications.

Why use PHP ?

- PHP can generate dynamic page content
- It allows the creation, opening, reading, writing, deletion, and closing of files on the server.
- PHP can collect and process data from forms submitted by users.
- It can send and receive cookies.
- It offers tools for managing user authentication and controlling access to various parts of a website.
- PHP is highly compatible with various platforms, including Windows, Linux, Unix, and macOS.

Understand how a web application runs with PHP and MySQL.



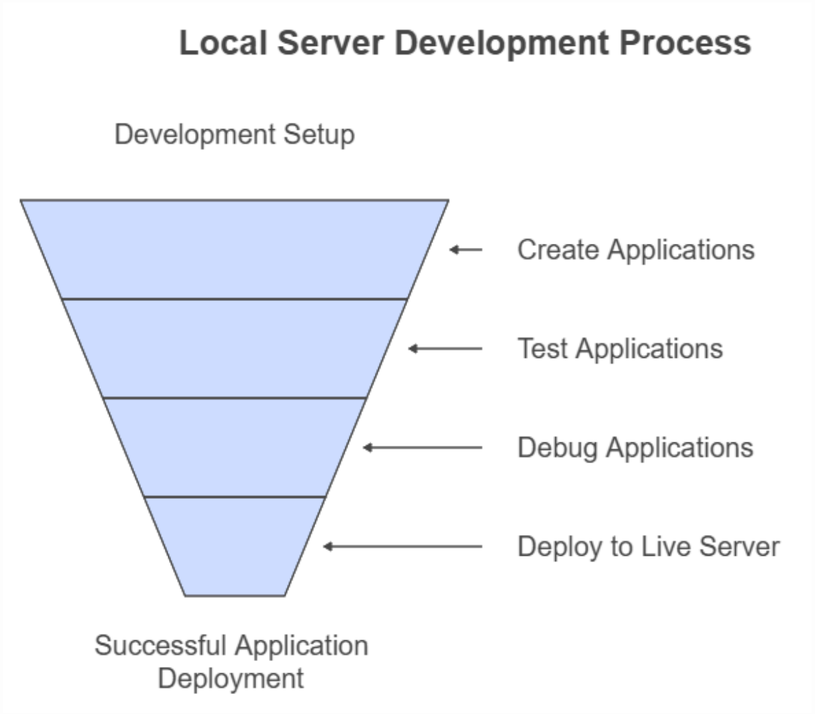
When a user accesses a web page, their browser sends a request to the web server. The server processes this request by executing a PHP script. If the script requires data, it communicates with the MySQL database to retrieve or update information as necessary. Once the data is processed, PHP generates an HTML response and sends it back to the browser. The browser then renders the content, allowing the user to view and interact with the page.

This smooth process updates content in real-time, giving users a responsive and engaging experience.

Why Use a Local Server?

A local server for PHP development is an environment installed on your computer that lets you develop, test, and debug PHP applications locally before moving them to a live server.

This environment replicates the conditions of a web server, including essential components such as a web server (e.g., Apache or Nginx), a database server (e.g., MySQL or MariaDB), and the PHP programming language.



Tools for installing and configuring PHP and MySQL.

Local Server

Laragon : https://laragon.org/download/

XAMPP : https://www.apachefriends.org/download.html

Code Editor

Notepad++ : https://laragon.org/download/

VS Code : https://code.visualstudio.com/download

Browser

Google Chrome : https://www.google.com/chrome/

Basic Syntax

PHP Tags : A PHP script begins with **<?php** and ends with **?>**. This allows the PHP parser to identify which parts of the document are PHP code and which are HTML.
Here's a simple example:

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

Embedding PHP in HTML : PHP can be embedded directly within HTML. For instance:

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

File Extension : PHP files typically use the **.php** extension, indicating that they contain PHP code which will be executed on the server.

Statement Termination : Each PHP statement must end with a semicolon (;). This is similar to languages like C and C++,

PHP Variables : A variable in PHP is the name of the **memory location that holds data**. In PHP, a variable is declared using the **$** sign followed by the variable name. The main way to store information in PHP program is by using a variable.

```
$name = "John"; // String variable
$age = 30;      // Integer variable
```

Rules for PHP variables : A variable starts with the $ sign, followed by the name of the variable names are **case-sensitive** ($age and $AGE are two different variables) One must keep in mind that variable names in PHP names must start with a letter or underscore and no numbers.

Example : $name, $fastName, $fast_name

Data Type

Variables can store data of different types, and different data types can do different things.

Data Type	What It Is	Example
String	A series of characters or text, written inside quotes.	"Hello, World!"
Integer	Whole numbers, positive or negative (no decimals).	42, -15
Float	Numbers with a decimal point.	3.14, -0.99
Boolean	Represents either true (yes) or false (no).	true, false
Array	A collection of values, like a list. It can be numbered or named.	array(1, 2, 3) ["name" =>"John"]
Object	A special type of data created from a class, with properties and methods.	$car = new Car();
NULL	A variable with no value. It means "nothing."	$var = NULL;

PHP Comments

A comment in PHP code is a line that is not executed as a part of the program. Its only purpose is to be read by someone who is looking at the code.

Syntax for single-line comments:

```
<?php
// This is a single-line comment
# This is also a single-line comment
?>
```

Syntax for multiple-line comments:

```
<?php
/*
This is a multiple-lines comment block
that spans over multiple
lines
*/
?>
```

PHP echo and print Statements

echo and print are more or less the same. They are both used to output data to the screen. The differences are small: echo has no return value while print has a return value of 1. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

PHP Operators

An operator is a symbol or set of symbols used to perform operations on variables or values.

Arithmetic Operators : These are used to perform mathematical calculations.

Operator	Description	Example	Result
`+`	Addition	`5 + 3`	`8`
`-`	Subtraction	`5 - 3`	`2`
`*`	Multiplication	`5 * 3`	`15`
`/`	Division	`6 / 3`	`2`
`%`	Modulus (remainder)	`5 % 3`	`2`

Assignment Operators : These assign values to variables.

Operator	Description	Example	Meaning
`=`	Assign value	`$x = 5`	`$x` is `5`
`+=`	Add and assign	`$x += 3`	`$x = $x + 3`
`-=`	Subtract and assign	`$x -= 2`	`$x = $x - 2`
`*=`	Multiply and assign	`$x *= 2`	`$x = $x * 2`
`/=`	Divide and assign	`$x /= 2`	`$x = $x / 2`
`%=`	Modulus and assign	`$x %= 2`	`$x = $x % 2`

Comparison Operators : These compare two values and return true or false.

Operator	Description	Example	Result
`==`	Equal to	`5 == 3`	`false`
`===`	Identical (same type)	`5 === "5"`	`false`
`!=`	Not equal to	`5 != 3`	`true`
`<>`	Not equal to	`5 <> 3`	`true`
`!==`	Not identical	`5 !== "5"`	`true`
`>`	Greater than	`5 > 3`	`true`
`<`	Less than	`5 < 3`	`false`
`>=`	Greater than or equal to	`5 >= 3`	`true`
`<=`	Less than or equal to	`5 <= 3`	`false`

Logical Operators : These are used to combine conditional statements.

Operator	Description	Example	Result
`&&`	And	`true && false`	`false`
||	OR	true && false	true
`!`	Not	`!true`	`false`

Increment/Decrement Operators : These are used to increase or decrease the value of a variable by 1.

Operator	Name	Description	Example	Output
`++$x`	Pre-increment	Increments `$x` by 1 **before** returning the value.	`$x = 5; echo ++$x;`	6 (increments, then prints)
`$x++`	Post-increment	Increments `$x` by 1 **after** returning the current value.	`$x = 5; echo $x++;`	5 (prints, then increments)
`--$x`	Pre-decrement	Decrements `$x` by 1 **before** returning the value.	`$x = 5; echo --$x;`	4 (decrements, then prints)
`$x--`	Post-decrement	Decrements `$x` by 1 **after** returning the current value.	`$x = 5; echo $x--;`	5 (prints, then decrements)

Example:

```php
// Arithmetic operator example
$x = 10;
$y = 5;
$result = $x + $y;  // 15

// Comparison operator example
if ($x > $y) {
    echo "x is greater than y";
}

// Logical operator example
if ($x > 5 && $y < 10) {
    echo "Both conditions are true!";
}
```

PHP Conditional Statements

In PHP, conditional statements allow you to perform different actions based on different conditions. Here are the basic types of conditional statements in PHP:

1. if Statement : Executes code if the specified condition is true.

Syntax :

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

Example 1 : Determine if a given number is positive

```
<?php
$number = 5;

if ($number > 0) {
    echo "The number is positive.";
}
?>
```

Example 2 : Check if a person is eligible to vote based on their age.

```
<?php
$age = 18;

if ($age >= 18) {
    echo "You are eligible to vote.";
}
?>
```

2. If-Else Statement : The if...else statement executes a block of code if a condition is true, and another block of code if that condition is false.

Syntax :

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

Example 1 : You need to check if a person is eligible to vote. A person must be at least 18 years old to vote. If they are eligible, display a message indicating they can vote; otherwise, let them know they cannot.

```
<?php
$age = 16;

if ($age >= 18) {
    echo "You are eligible to vote!";
} else {
    echo "You are not eligible to vote.";
}
?>
```

Example 2 : You are building a simple login system. If a user enters the correct username and password, show a welcome message; otherwise, inform them that the login failed.

```
<?php
    $username_input = "john_doe";
    $password_input = "password123";

   $stored_username = "john_doe";
   $stored_password = "password123";

if ($username_input == $stored_username && $password_input == $stored_password) {
    echo "Welcome, $username_input!";
} else {
    echo "Login failed. Please try again.";
}
?>
```

2. if...elseif...else Statement : The if...elseif...else statement in PHP allows you to evaluate multiple conditions in a structured way. It provides a way to execute different blocks of code based on various conditions

```
if (condition1) {
    // Code to execute if condition1 is true
} elseif (condition2) {
    // Code to execute if condition2 is true
} else {
    // Code to execute if none of the conditions are true
}
```

Example 1 : Write a PHP script to assign a letter grade based on a student's score.

```
<?php
$score = 85;

if ($score >= 90) {
    echo "Grade: A";
} elseif ($score >= 80) {
    echo "Grade: B";
} elseif ($score >= 70) {
    echo "Grade: C";
} elseif ($score >= 60) {
    echo "Grade: D";
} else {
    echo "Grade: F";
}
?>
```

Example 2 : Create a PHP script that checks the temperature and indicates whether it is hot, warm, or cold. If the temperature is above 30 degrees Celsius, display "It's hot." If it's between 15 and 30 degrees, display "It's warm." Otherwise, display "It's cold.

```
<?php
$temperature = 25;

if ($temperature > 30) {
    echo "It's hot.";
} elseif ($temperature >= 15 && $temperature <= 30) {
    echo "It's warm.";
} else {
    echo "It's cold.";
}
?>
```

2. PHP switch Statement : The switch statement in PHP is used to execute one block of code among many based on the value of a variable or expression. It is often used as an alternative to multiple if...elseif...else statements when you need to compare the same variable against different values.

```
switch (expression) {
    case value1:
        // Code to execute if expression matches value1
        break;
    case value2:
        // Code to execute if expression matches value2
        break;
    // Add more cases as needed
    default:
        // Code to execute if no cases match
}
```

Example 1 : This example checks the color of a fruit based on its name.

```
<?php
$fruit = "apple";

switch ($fruit) {
    case "apple":
        echo "Apples are red.";
        break;
    case "banana":
        echo "Bananas are yellow.";
        break;
    case "orange":
        echo "Oranges are orange.";
        break;
    default:
        echo "Unknown fruit.";
        break;
}
?>
```

Example 2 : Create a simple calculator that performs operations based on the input operator.

```
<?php
$num1 = 10;
$num2 = 5;
$operator = "+";

switch ($operator) {
    case "+":
        echo $num1 + $num2;
        break;
    case "-":
        echo $num1 - $num2;
        break;
    case "*":
        echo $num1 * $num2;
        break;
    case "/":
        if ($num2 != 0) {
            echo $num1 / $num2;
        } else {
            echo "Cannot divide by zero.";
        }
        break;
    default:
        echo "Invalid operator";
        break;
}
?>
```

Loop

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

1. while Loop : The while loop continues to run as long as the condition is true.

Syntax :

```
while (condition is true) {
  // Code to be executed while the condition is true
}
```

Example 1 : Here's a simple example that uses a while loop to print numbers from 1 to 5:

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

This will output : 1 2 3 4 5

In this example:

- The $counter variable is initialized to 1.
- While loop continues executing the code inside its block as long as the condition $counter <= 5 is true.
- Inside the loop, echo $counter . "
"; prints the current value of $counter followed by a line break (
).
- After each iteration of the loop, $counter is incremented using the $counter++ statement.

2. do...while Loop : The do...while loop is similar to the while loop, but it always executes the block of code at least once, even if the condition is false.

Syntax :

```
do {
  //code to be executed;
} while (condition is true);
```

Example 1 : Here's a simple example that uses a do...while loop to print numbers from 1 to 5:

```
<?php
$counter = 1;
do {
    echo $counter . "<br>";
    $counter++;
} while ($counter <= 5);
?>
```

This will output : 1 2 3 4 5

In this example:

- The do block contains the code to be executed. It echoes the current value of the $counter variable.
- The loop continues (while ($counter <= 5)) as long as the condition is true. The loop increments the $counter variable in each iteration.
- Since the loop condition is checked at the end of the loop, the block of code inside the do will be executed at least once, even if the initial condition is false.

3. for Loop : The for loop is used when you know in advance how many times the block of code should run.

Syntax :

```
for (initialization; condition; increment/decremen) {
  // code to be executed for each iteration;
}
```

Example 1 : Here's a simple example that uses a for loop to print numbers from 1 to 5:

```
<?php
for ($counter = 1; $counter <= 5; $counter++) {
    echo $counter . "<br>";
}
?>
```

This will output : 1 2 3 4 5

In this example:

- The for loop consists of three parts:
- Initialization: $counter = 1; initializes the loop control variable $counter to 1.
- Condition: $counter <= 5; specifies the condition for the loop to continue executing. As long as this condition is true, the loop will continue.
- Increment/Decrement: $counter++ increments the value of $counter by 1 after each iteration of the loop.
- Inside the loop, echo $counter . "
"; prints the current value of $counter followed by a line break (
).

The for loop is particularly useful when you know in advance how many times you want to execute a block of code.

Example 1 : Here's a simple example that uses a for loop to print even numbers up to a specified limit:

```php
<?php
$endNumber = 10;
echo "Even numbers up to $endNumber: ";
for ($i = 2; $i <= $endNumber; $i++) {
    if ($i % 2 == 0) {
        echo "$i ";
    }
}
?>
```

This will output : 2 4 6 8 10

Note: Even numbers are those numbers Mod(%) 2 and the remainder will be 0. For example, if the entered number is 8 and when '8 % 2' is calculated the result will be 0 so it is an even number.

3. PHP foreach Loop : The foreach loop in PHP is used to iterate over arrays and objects. It provides a simple way to loop through each element in an array or each property in an object. The basic syntax of a foreach loop is as follows:

Syntax :

```
foreach ($array as $value) {
  // Code to be executed for each $value
}
```

Example 1 : Here's an example that uses foreach to iterate over an array:

```
<?php
$colors = array("Red", "Green", "Blue");
foreach ($colors as $color) {
    echo $color . "<br>";
}
?>
```

This will output: Red Green Blue

In this example:

- The foreach loop iterates over each element in the $colors array.
- The current element's value is assigned to the variable $color in each iteration.
- Inside the loop, echo $color . "
"; prints each color followed by a line break (
).

A **superglobal variable** in PHP is a built-in global array accessible from any part of the code, regardless of scope. These variables store data like user input ($_GET, $_POST), session info ($_SESSION), server details ($_SERVER), and more, simplifying access to essential information across the application.

Supergloba	What It Does	When to Use It	Example
$_GET	Holds data from the URL (like search terms)	Get data from the URL	$_GET['name'] for ?name=John
$_POST	Holds data from forms (like login info)	Get data from a form submission	$_POST['email'] for a form field
$_REQUEST	Holds data from GET, POST, or COOKIE	Get data from any type of request	$_REQUEST['name']
$_SESSION	Stores info for a user's session	Keep track of users across pages	$_SESSION['user_id']
$_COOKIE	Stores data on the user's browser	Save small data across visits	$_COOKIE['theme'] for site theme
$_FILES	Holds data about uploaded files	Handle file uploads	$_FILES['upload']['name']
$_SERVER	Holds server info and headers	Get server or script details	$_SERVER['SERVER_NAME']
$_ENV	Holds environment variables	Store settings or environment info	$_ENV['APP_ENV']
$GLOBALS	Holds all global variables	Access any global variable	$GLOBALS['var'] for var

Integrating HTML Forms with PHP :

```
<!DOCTYPE html>
<head>
  <title>Simple Form</title>
</head>
<body>
  <h1>Contact Form</h1>

  <!-- Form to collect user information -->
  <form method="post" action="">
    <label for="name">Name:</label><br>
    <input type="text" id="name" name="name" required><br><br>

    <label for="email">Email:</label><br>
    <input type="email" id="email" name="email" required><br><br>

    <label for="phone">Phone:</label><br>
    <input type="tel" id="phone" name="phone" required><br><br>

    <label for="designation">Designation:</label><br>
    <input type="text" id="designation" name="designation" required><br><br>

    <input type="submit" value="Submit">
  </form>

</body>
</html>
```

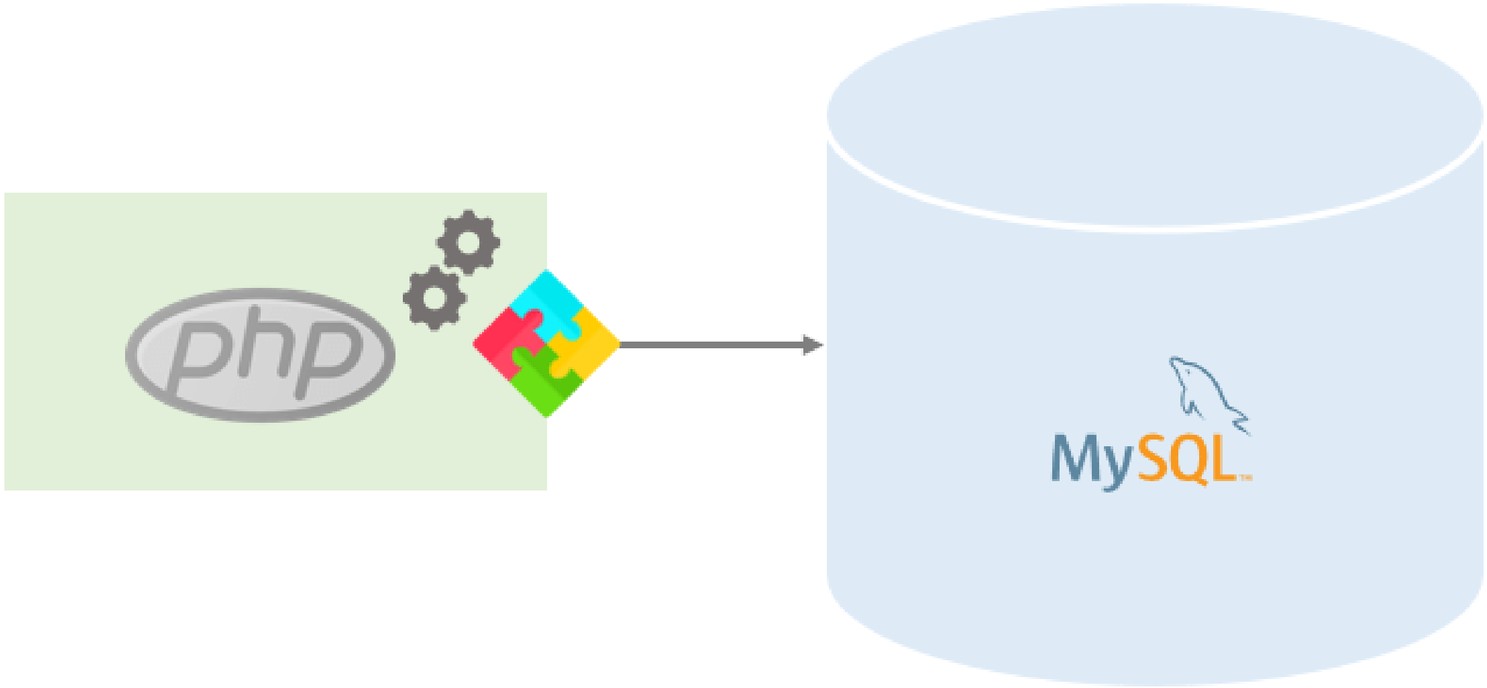
```
<?php
// Check if the form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // Get the form data
  $name = htmlspecialchars($_POST['name']);
  $email = htmlspecialchars($_POST['email']);
  $phone = htmlspecialchars($_POST['phone']);
  $designation = htmlspecialchars($_POST['designation']);

  // Display the submitted data
  echo "<h2>Submitted Information:</h2>";
  echo "Name: $name<br>";
  echo "Email: $email<br>";
  echo "Phone: $phone<br>";
  echo "Designation: $designation<br>";
}
?>
```

Explanation

1. **HTML Form:**
   - The form uses the POST method to submit data. The action attribute is empty, meaning it will submit to the same page.
   - Each input field has a corresponding label and a required attribute to ensure the user fills them out.
2. **PHP Section:**
   - The PHP code checks if the form is submitted using $_SERVER["REQUEST_METHOD"].
   - It retrieves the form data using $_POST and sanitizes it with htmlspecialchars() to prevent XSS (cross-site scripting) attacks.
   - The submitted data is then displayed below the form.

How to Use

1. Save the code in a file with a .php extension, like form.php.
2. Run the file on a local server (like XAMPP, WAMP, or Laragon) or a web server that supports PHP.
3. Fill out the form and submit it to see the submitted information displayed on the page.


php
MySQL

What is MySql ?

MySQL is a popular open-source relational database management system (RDBMS) that is used to store, manage, and retrieve data in structured formats. It's based on SQL (Structured Query Language), a standard language for database management, which makes it widely compatible and easy to use. MySQL is known for its speed, reliability, and flexibility, making it one of the most widely used databases for web applications, software development, and large data-driven systems.

Key Features of MySQL

- **Relational Database:** MySQL organizes data into tables with rows and columns, making it easier to retrieve and work with data that's related across multiple tables.
- **SQL Compatibility:** MySQL uses standard SQL, allowing developers to perform operations like selecting, inserting, updating, and deleting data using SQL queries.
- **Data Security:** MySQL offers strong data protection features like authentication and user access control, ensuring secure data handling.
- **Scalability:** MySQL is suitable for small and large applications, from single-user databases to large applications with millions of records.
- **Open-Source:** MySQL is free to use, but there are also enterprise versions available with additional features and support.
- **Cross-Platform Compatibility:** MySQL runs on multiple operating systems, including Windows, macOS, and Linux, making it versatile for many different setups.

Let's get started with Mysql actual coding ...

Working with Mysql Database

Setting Up a New Database (Create a new database):

```
CREATE DATABAS databasename;
```

Listing All Existing Databases (View all available databases):

```
SHOW DATABASES;
```

Select a Database to Work With :

```
USE databasename;
```

Remove a database permanently:

```
DROP DATABASE databasename
```

Working with Mysql Tables

Tables are the key element of MySQL databases. they let you store all the information together in organized rows. Each row consists of columns that feature a specified data type. You have plenty of options for customization using the commands below.

To create a new table in a database, you can use the CREATE TABLE statement. Here's a basic example:

```
CREATE TABLE table_name (
     column1 datatype,
     column2 datatype
);
```

```
CREATE TABLE student(
   id INT AUTO_INCREMENT,
   name VARCHAR(100),
   roll VARCHAR(50),
   dept VARCHAR(100),
   PRIMARY KEY(id)
);
```

Use the next commands to get more information about the tables stored in your database:

```
SHOW TABLES; – List All Tables in the Database.
DESCRIBE table_name; – View Table Structure.
```

To rename a table in MySQL, you can use the RENAME TABLE statement. Here's the basic syntax:

```
RENAME TABLE old_table_name TO new_table_name;
RENAME TABLE employees TO staff;
```

To permanently delete a table, use the DROP TABLE command:

```
DROP TABLE tablename;
```

Working With Table Columns

Data Types:

In MySQL, when creating tables, you need to specify the data types for each column. The choice of data type depends on the kind of data you want to store. Here are some commonly used data types in MySQL:

Data Type	Description	Examples
INTEGER or INT	Used for whole numbers (positive or negative).	`TINYINT`, `SMALLINT`, `MEDIUMINT`, `INT`, `BIGINT`
FLOAT and DOUBLE	Used for floating-point numbers (numbers with a decimal point).	`FLOAT`, `DOUBLE`
DECIMAL or NUMERIC	Used for fixed-point numbers (exact decimal representation).	`DECIMAL(p, d)`, `NUMERIC(p, d)`
CHAR and VARCHAR	Used for character strings.	`CHAR(n)`, `VARCHAR(n)`
TEXT	Used for large blocks of text.	`TEXT`, `LONGTEXT`
DATE and TIME	Used for date and time values.	`DATE`, `TIME`, `DATETIME`, `TIMESTAMP`
BOOLEAN or BOOL	Used for storing true/false values.	`BOOLEAN`, `BOOL`, `TINYINT(1)`

Working With Table Columns

MySQL provides powerful commands to manage and customize table columns, including adding, modifying, and deleting columns.

Use the ALTER TABLE statement to add a new column to an existing table:

```
ALTER TABLE table_name ADD column_name datatype;
```

Example:

```
ALTER TABLE student ADD email varchar(255);
```

Use the ALTER TABLE statement to modify the data type or attributes of an existing column:

```
ALTER TABLE table_name MODIFY COLUMN column_name new_data_type;
```

Example:

```
ALTER TABLE student MODIFY COLUMN roll MEDIUMINT(3)
```

Use the ALTER TABLE statement to permanently remove a column from a table:

```
ALTER TABLE table_name DROP COLUMN column_name;
```

Example:

```
ALTER TABLE employees DROP COLUMN phone;
```

Working With Table Row

MySQL allows you to manage table rows efficiently by inserting new data, updating existing records, and more. Below are the key commands for working with table rows.

To insert a new row in a table, use the following syntax:

```
INSERT INTO table_name (column_name, column_name, ...) VALUES (value1, value2, ...);
```

Example 1: (Insert single row)

```
INSERT INTO student(name, roll, dept, email) VALUES ('Ashik', '1', 'CSE', 'ashik@gmail.com');
```

Example 2: (Insert Multiple Rows)

```
INSERT INTO student (name, roll, dept, email)
VALUES
('Sakib', '2', 'EEE', 'sakib@gmail.com'),
('Shuvox', '3', 'CSE', 'shuvo@gmail.com'),
('Tirtho', '4', 'CSE', 'tirtho@gmail.com');
```

To Update Row/Data in a table, use the following syntax:

```
UPDATE table_name SET column = value, …, WHERE unique_id = ‘value’;
```

Example :

```
UPDATE student SET name ="Shuvo", dept="EEE" WHERE id ='2';
```

Working With Mysql SELECT statement

The SELECT statement is used to retrieve data from tables in MySQL. Below is the syntax and examples for its usage.

```
SELECT select_column_list FROM table_name;
```

Example:

```
SELECT name, roll, dept FROM student;
```

Alternatively, you can use the asterisk (*) which is the shorthand for all columns. For example:

```
SELECT * FROM table_name;
```

Example:

```
SELECT * FROM student;
```

You can bring up a certain data range using the next command:

```
SELECT * FROM table_name WHERE column_unique BETWEEN value AND value;
```

Example:

```
SELECT * FROM student WHERE roll BETWEEN 2 AND 3;
```

Working With Mysql Sort Entries in a Column

Sorting data in MySQL allows you to organize the rows of a table in a specific order, much like sorting in Excel. You can sort data alphabetically, numerically, or by ascending/descending order.

To sort the rows of a table based on a column:

```
SELECT * FROM table ORDER BY column ASC/DESC;
```

Example:

```
SELECT * FROM student ORDER BY name ASC/DESC;
```

MySQL ORDER BY clause

When you use the SELECT statement to query data from a table, the result set is not sorted. It means that the rows in the result set can be in any order.

To sort the result set, you add the ORDER BY clause to the SELECT statement. The following illustrates the syntax of the ORDER BY clause:

```
SELECT select_list FROM table_name ORDER BY column1 [ASC|DESC], column2 [ASC|DESC];
```

Example:

```
SELECT contactLastname,contactFirstname FROM customers ORDER BY contactLastname DESC, contactFirstname ASC;
```

Mysql Where clause

The WHERE clause in MySQL allows you to filter rows returned by a query based on a specified condition. This is useful for retrieving only the data that meets specific criteria.

```
SELECT column1, column2, ...
FROM table_name
WHERE search_condition;
```

Example: The following query uses the WHERE clause to find all employees whose job titles are Sales Rep:

```
SELECT lastname, firstname, jobtitle FROM employees WHERE jobtitle = 'Sales Rep';
```

The following example uses the WHERE clause to find employees whose job titles are Sales Rep and office codes are 1:

```
SELECT lastname, firstname, jobtitle,officeCode FROM employees WHERE jobtitle = 'Sales Rep' AND officeCode = 1;
```

The following query returns employees with office code less than or equal 4 (<=4):

```
SELECT lastname, firstname, officeCode FROM employees WHERE officecode ⩽ 4;
```

MySQL AND Operator

The AND operator is a logical operator that combines two or more Boolean expressions and returns true only if both expressions evaluate to true. The AND operator returns false if one of the two expressions evaluates to false.

The following statement uses the AND operator to find customers who are located in California (CA), USA:

```
SELECT select_list FROM table_name WHERE search_condition AND search_condition;
```

Example:

```
SELECT customername, country, state FROM customers WHERE country = 'USA' AND state = 'CA';
```

By using the AND operator, you can combine more than two Boolean expressions. For example, the following query returns the customers who locate in California, USA, and have the credit limit greater than 100K.

```
SELECT select_list FROM table_name WHERE search_condition AND search_condition AND search_condition;
```

Example:

```
SELECT customername, country, state, creditlimit FROM customers WHERE country = 'USA' AND state = 'CA'
AND creditlimit > 100000;
```

MySQL OR Operator

The MySQL OR operator combines two Boolean expressions and returns true when either condition is true.

```
SELECT select_list FROM table_name WHERE search_condition OR search_condition;
```

For example, to get the customers who locate in the USA or France, you use the OR operator in the WHERE clause as follows

```
SELECT customername, country FROM customers WHERE country = 'USA' OR country = 'France';
```

THANK YOU!

Shaeed Al Hasan - Siam

Software Engineer | Educator

01787972797

siamshaeed.com

J Block, Baridhara, Dhaka