UNIT- 4

Basic Commands with PHP Examples, Connection to server, creating database, selecting a database, listing database, listing table names, creating a table, inserting data, altering tables, queries, deleting data and tables, PHP myadmin and database bugs.

by Pratishtha Gupta

PHP

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open-source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP is executed on the server, and the result is returned to the browser as HTML
- PHP files have extension ".php"

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 and send and receive cookies
- PHP can add, delete, and modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data.

1. Dynamic page content
2. open, read, write, delete & close files.
3. collect form data, receive cookies.
4. add, delete, modify data in database.
5. control user access
6. encrypt data.

Basic PHP Syntax

- A PHP script can be placed anywhere in the document.
- A PHP script starts with <?php and ends with ?>:
- The default file extension for PHP files is ".php".
- A PHP file normally contains HTML tags, and some PHP scripting code.
- Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

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

A simple `.php` file with both HTML code and PHP code:

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

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

- **PHP Case Sensitivity:** In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are not case-sensitive.
- **Variables** are "containers" for storing information.
- In PHP, a variable starts with the $ sign, followed by the name of the variable:

```php
$x = 5;
$y = "John";
```

PHP is a Loosely Typed Language

- PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

- With PHP, there are two basic ways to get output: echo and print.
- The echo and print statement can be used with or without parentheses: echo or echo() and print or print ().

```
echo "Hello";
//same as:
echo("Hello");
```

Display Text: The following example shows how to output text with the echo command (notice that the text can contain HTML markup):

```
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
```

Display Variables: The following example shows how to output text and variables with the echo statement:
When using double quotes, variables can be inserted to the string as in the example above.
When using single quotes, variables have to be inserted using the . operator, like this:

```
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";

echo "<h2>$txt1</h2>";
echo "<p>Study PHP at $txt2</p>";
```

PHP MySQL Database

- With PHP, you can connect to and manipulate databases.
- MySQL is the most popular database system used with PHP.

What is MySQL?

- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation
- MySQL is named after co-founder Monty Widenius's daughter: My
- The data in a MySQL database are stored in tables.
- A table is a collection of related data, and it consists of columns and rows.
- Databases are useful for storing information categorically.

- PHP 5 and later can work with a MySQL database using:
- MySQLi extension (the "i" stands for improved)
- PDO (PHP Data Objects)
- Earlier versions of PHP used the MySQL extension. However, this extension was deprecated in 2012.
- Both MySQLi and PDO have their advantages:
- PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.
- So, if you have to switch your project to use another 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 entire code - queries included.
- Both are object-oriented, but MySQLi also offers a procedural API.

Connection to Server:

- Before we can access data in the MySQL database, we need to be able to connect to the server:
- Example using MySQLi Procedural

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

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

// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

Close the Connection

The connection will be closed automatically when the script ends. To close the connection before, use the following:

MySQLi Procedural:

```php
mysqli_close($conn);
```

1. Connection: $conn=mysqli_connect($host, $username, $password, $database);

2. Closing: mysqli_close($conn);

Creating a Database

- You will need special CREATE privileges to create or to delete a MySQL database
- The CREATE DATABASE statement is used to create a database in MySQL.
- The following examples create a database named "myDB":

mysqli_connect_error($conn)

mysqli_error($conn)

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

// Create connection
$conn = mysqli_connect($servername, $username, $password);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

// Create database
$sql = "CREATE DATABASE myDB";
if (mysqli_query($conn, $sql)) {
  echo "Database created successfully";
} else {
  echo "Error creating database: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Select a Database

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

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

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Select database
if (!mysqli_select_db($conn, $dbname)) {
    die("Selection of database failed: " . mysqli_error($conn));
}

echo "Connected and database selected successfully";

// Your code to interact with the database goes here

mysqli_close($conn);
?>
```

Listing a Database

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

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

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Query to list databases
$sql = "SHOW DATABASES";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // Output data of each row
    while($row = mysqli_fetch_assoc($result)) {
        echo "Database: " . $row["Database"] . "<br>";
    }
} else {
    echo "No databases found";
}

mysqli_close($conn);
?>
```

Listing Table Names

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

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

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Query to list tables
$sql = "SHOW TABLES";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // Output data of each row
    while($row = mysqli_fetch_array($result)) {
        echo "Table: " . $row[0] . "<br>";
    }
} else {
    echo "No tables found in database";
}

mysqli_close($conn);
?>
```

Creating a Table

- A database table has its own unique name and consists of columns and rows.
- The CREATE TABLE statement is used to create a table in MySQL.
- We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date":

```
CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)
```

- The data type specifies what type of data the column can hold.
- After the data type, you can specify other optional attributes for each column:

- NOT NULL - Each row must contain a value for that column, null values are not allowed
- DEFAULT value - Set a default value that is added when no other value is passed
- UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
- AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
- PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT

Each table should have a primary key column (in this case: the "id" column). Its value must be unique for each record in the table.

Creating a Table

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

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)";

if (mysqli_query($conn, $sql)) {
  echo "Table MyGuests created successfully";
} else {
  echo "Error creating table: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Inserting Data in Table

Syntax rules to follow:

- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted
- Numeric values must not be quoted
- The word NULL must not be quoted

The INSERT INTO statement is used to add new records to a MySQL table:

INSERT INTO table_name
(column1, column2, column3,...)
VALUES (value1, value2, value3,...)

Inserting Data

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

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if (mysqli_query($conn, $sql)) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Altering Table:

Depending on what alteration you want to perform, your ALTER TABLE statement will vary. Below are the SQL Queries.

- **Add a Column:** ALTER TABLE table_name ADD column_name column_definition;
- **Drop a Column:** ALTER TABLE table_name DROP COLUMN column_name;
- **Modify a Column:** ALTER TABLE table_name MODIFY COLUMN column_name new_column_definition;
- **Rename a Column:** ALTER TABLE table_name CHANGE old_column_name new_column_name column_definition;
- **Rename a Table:** ALTER TABLE old_table_name RENAME TO new_table_name;

Altering Table:

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

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

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// SQL query to alter the table
$sql = "ALTER TABLE my_table ADD new_column VARCHAR(255)";

// Execute the query
if (mysqli_query($conn, $sql)) {
    echo "Table altered successfully";
} else {
    echo "Error altering table: " . mysqli_error($conn);
}

// Close the connection
mysqli_close($conn);
?>
```

- **Changing a Column Type** : $sql = "ALTER TABLE my_table MODIFY COLUMN old_column_name NEW_DATATYPE";
- **Renaming a Column:** $sql = "ALTER TABLE my_table CHANGE old_column_name new_column_name VARCHAR(255)";
- **Dropping a Column:** $sql = "ALTER TABLE my_table DROP COLUMN column_name";
- **Renaming a Table:** $sql = "RENAME TABLE old_table_name TO new_table_name";

Queries:

Below is an example that performs selection, insertion, updation, and deletion of records.

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

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

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// SELECT query
$sql = "SELECT id, firstname, lastname FROM my_table";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    while($row = mysqli_fetch_assoc($result)) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"].
    }
} else {
    echo "0 results";
}
mysqli_free_result($result);
```

```php
// INSERT query
$sql = "INSERT INTO my_table (firstname, lastname, email) VALUES ('John', 'Doe', 'john@exa
if (mysqli_query($conn, $sql)) {
    echo "New record created successfully<br>";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

// UPDATE query
$sql = "UPDATE my_table SET lastname='Smith' WHERE id=1";
if (mysqli_query($conn, $sql)) {
    echo "Record updated successfully<br>";
} else {
    echo "Error updating record: " . mysqli_error($conn);
}

// DELETE query
$sql = "DELETE FROM my_table WHERE id=1";
if (mysqli_query($conn, $sql)) {
    echo "Record deleted successfully<br>";
} else {
    echo "Error deleting record: " . mysqli_error($conn);
}

// Close connection
mysqli_close($conn);
?>
```

WHERE Clause

- Select and Filter Data From a MySQL Database
- The WHERE clause is used to filter records.
- The WHERE clause is used to extract only those records that fulfill a specified condition.
- SELECT column_name(s) FROM table_name WHERE column_name operator value

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

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM MyGuests WHERE lastname='Doe'";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
  // output data of each row
  while($row = mysqli_fetch_assoc($result)) {
    echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
} else {
  echo "0 results";
}

mysqli_close($conn);
?>
```

Select and Order Data From a MySQL Database

- The ORDER BY clause is used to sort the result-set in ascending or descending order.
- The ORDER BY clause sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.
- SELECT column_name(s) FROM table_name ORDER BY column_name(s) ASC|DESC

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

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM MyGuests ORDER BY lastname";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
  // output data of each row
  while($row = mysqli_fetch_assoc($result)) {
    echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
  }
} else {
  echo "0 results";
}

mysqli_close($conn);
?>
```

Delete Data

DELETE FROM table_name WHERE some_column = some_value

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

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if (mysqli_query($conn, $sql)) {
  echo "Record deleted successfully";
} else {
  echo "Error deleting record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Delete Table

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

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

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// SQL query to delete the table
$sql = "DROP TABLE my_table";

// Execute the query
if (mysqli_query($conn, $sql)) {
    echo "Table deleted successfully";
} else {
    echo "Error deleting table: " . mysqli_error($conn);
}

// Close the connection
mysqli_close($conn);
?>
```

phpMyAdmin

- **phpMyAdmin** is a popular web-based tool for managing MySQL and MariaDB databases. It provides an easy-to-use interface to manage and interact with databases, perform queries.
- **Using phpMyAdmin:** Once logged in, you can use phpMyAdmin to manage your databases.
- **Create a Database:**
  - Click on "Databases" in the top menu.
  - Enter a name for the database and click "Create".
- **Create a Table:**
  - Select a database.
  - Enter a name for the table and the number of columns.
  - Define the columns and click "Save".
- **Run SQL Queries:**
  - Select a database.
  - Click on the "SQL" tab.
  - Enter your SQL query and click "Go".
- **Import/Export Data:**
  - Select a database or table.
  - Click on the "Import" or "Export" tab.
  - Follow the prompts to upload or download data.

Database Bugs

- Database bugs can arise from various issues, including design flaws, coding errors, configuration problems, or even unexpected data inputs. Here are some common types of database bugs, their potential causes, and how to troubleshoot or resolve them:

Common Database Bugs

1. **Data Integrity Issues**
   - Cause: Missing constraints or improper transaction handling.
   - Resolution: Add necessary constraints and use proper transaction management.
2. **Performance Problems**
   - Cause: Missing indexes, poorly written queries, or database locks.
   - Resolution: Use EXPLAIN for query analysis, optimize queries, and monitor locks.
3. **SQL Injection Vulnerabilities**
   - Cause: Lack of parameterized queries.
   - Resolution: Always use prepared statements to prevent injections.

4. **Data Loss or Corruption**
    - Cause: Hardware failure, software bugs, or lack of backups.
    - Resolution: Implement regular backups and monitor database logs.
5. **Concurrency Issues**
    - Cause: Improper transaction isolation or long-running transactions.
    - Resolution: Use appropriate isolation levels and minimize transaction durations.
6. **Configuration Errors**
    - Cause: Inappropriate default settings or misconfigured permissions.
    - Resolution: Review and optimize configuration settings and audit permissions.
7. **Database Connection Issues**
    - Cause: Network issues or too many connections.
    - Resolution: Check connection strings and implement connection pooling.
8. **Version Compatibility Issues**
    - Cause: Using deprecated features in new versions.
    - Resolution: Regularly update code for compatibility with the latest database versions.

9. **Error Handling Issues**
    - Cause: Lack of error handling in SQL queries.
    - Resolution: Implement thorough error handling and logging mechanisms.

- **Debugging Tips**
    a. **Logging:** Enable logging for errors and slow queries.
    b. **Testing:** Use unit and integration tests to catch bugs early.
    c. **Monitoring Tools:** Utilize database monitoring tools for performance tracking.
    d. **Documentation:** Maintain clear documentation for database schema and queries.