

# PHP and MySQL Introduction

Course Code: CSC 3222

Course Title: Web Technologies



**Dept. of Computer Science**  
**Faculty of Science and Technology**

Lecturer No:	06	Week No:	06	Semester:	Fall 2024-2025
Lecturer:	MD.AL-AMIN ( <a href="mailto:alamin@aiub.edu">alamin@aiub.edu</a> )				

# Lecture Outline



1. MySQL Database
2. MySQL installation
3. PHP & MySQL connection
4. MySQLi vs PDO
5. Create DB and Tables
6. Insert data to Tables

# MySQL

## What is MySQL



- MySQL is a database management system that allows you to manage relational databases (**RDBMS**).
- It is **open source** software backed by Oracle.
- MySQL is **easy** to master in comparison with other database software like Oracle Database, or Microsoft SQL Server.
- MySQL can run on **various platforms** UNIX, Linux, Windows, etc.
- It is most noted for its **quick** processing, proven **reliability**, ease and **flexibility** of use.

# Why MySQL



- MySQL is an essential part of almost every **open source** PHP application.
- PHP combined with MySQL are **cross-platform**.
- **Popular** examples for PHP & MySQL-based scripts are WordPress, Joomla, Magento and Drupal.
- MySQL is the standard database system for web sites with **enormous** volumes of both data and end-users.

# MySQLi vs PDO



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 and was deprecated in 2012.

# MySQLi vs PDO



Both MySQLi and PDO have their advantages:

- PDO will work on different database systems, whereas MySQLi will only work with MySQL databases.
- Switching project to use another database with PDO is easy.
- With MySQLi, need to rewrite the entire code - queries included.
- Both are object-oriented, but MySQLi also offers a procedural API.
- Both support Prepared Statements. Prepared Statements protect from SQL injection, and are very important for web application security.
- A great benefit of PDO is that it has **exception** class to handle any problems that may occur in our database queries.

# MySQLi Object-oriented vs MySQLi Procedural



The MySQLi extension features a dual interface.

- procedural
- object-oriented programming

Users migrating from the old MySQL extension may prefer the procedural interface.

- The procedural interface is similar to that of the old MySQL extension.
- the function names differ only by prefix.
- There are **no significant** performance differences between the two interfaces.

# MySQLi Object-oriented

## CREATE DATABASE Example



```
<?php
$servername = "localhost";
$username = "root";
$password = "";
// Create connection
$conn = new mysqli($servername, $username, $password); //MySQLi connection object
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
// Create database
$sql = "CREATE DATABASE myDB"; //query string
if ($conn->query($sql) === TRUE) { //query execute
    echo "Database created successfully";
} else {
    echo "Error creating database: " . $conn->error;
}
$conn->close(); // close MySQLi connection object
?>
```



# MySQLi Procedural

## CREATE DATABASE Example



```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$conn = mysqli_connect($servername, $username, $password); //MySQLi Procedural
connection object
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
$sql = "CREATE DATABASE myDB"; //query string
if (mysqli_query($conn, $sql)) { //query execute
    echo "Database created successfully";
} else {
    echo "Error creating database: " . mysqli_error($conn);
}
mysqli_close($conn); // close MySQLi Procedural connection object
?>
```

# PDO

## CREATE DATABASE Example



```
<?php
$servername = "localhost";
$username = "root";
$password = "";
try { // set the PDO error mode to exception
    $conn = new PDO("mysql:host=$servername", $username, $password); //PDO connection object
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $sql = "CREATE DATABASE myDB"; //query string
    $conn->exec($sql); //query execute
    echo "Database created successfully<br>";
}
catch(PDOException $e)
{
    echo $sql . "<br>" . $e->getMessage();
}
$conn = null; //close PDO connection object
?>
```

# Create Tables and Columns



Once we have created database we can create the connection object with the database name.

The CREATE TABLE statement is used to create a table in MySQL.

create a **table** named "User", with five **columns**: "id", "firstname", "lastname" and "email".

MySQLi object oriented extension has been used in this example.



# Create Tables and Columns

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";//including database name as a connection variable
$conn = new mysqli($servername, $username, $password, , $dbname);
//below is query string
$qry = "CREATE TABLE Users (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50)
)";
$res = $conn->query($qry);//execute query
if($res)
{
    echo "table created successfully";
}
else
{
    echo "error occurred";
}
$conn->close();
?>
```

# Insert Data



Here are some **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 table:  
`INSERT INTO table_name (column1, column2, column3,...) VALUES (value1, value2, value3,...)`

If a column is **AUTO\_INCREMENT** (like the "id" column) , it will automatically add the value. No need to mention in the query string.



# Insert Data

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$sql = "INSERT INTO Users (firstname, lastname, email)
VALUES ('alice', 'redd', 'alice@gmail.com')";
$res = $conn->query($qry);//execute query
if($res)
{
    echo "new record inserted";
}
else
{
    echo "error occurred";
}

$conn->close();
?>
```



# Insert data as php variable

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) { die("Connection failed: " . $conn->connect_error);}
$fame = " alice ";
$lname = " redd ";
$email = " alice@gmail.com ";

$sql = "INSERT INTO Users (firstname, lastname, email)
VALUES (". $fame .", ". $lname .", ". $email .")";
$res = $conn->query($qry);//execute query
if($res)
{
    echo "new record inserted";
}
else
{
    echo "error occurred";
}
$conn->close();
?>
```

# Insert Multiple Data



- Multiple SQL statements must be executed with the `mysqli_multi_query()` function.
- each SQL statement must be separated by a semicolon.





## Insert Data with Multiple Query

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
//appending SQL string
$sql = "INSERT INTO MyGuests (firstname, lastname, email) VALUES ('John', 'Doe',
'john@example.com')";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email) VALUES ('Mary', 'Moe',
'mary@example.com')";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email) VALUES ('Julie', 'Dooley',
'julie@example.com')";
$res = $conn->query($qry);//execute query
if($res){ echo "new record inserted";      }
else { echo "error occurred"; }
$conn->close();
?>
```



## References

- MySQL - [www.mysql.com](http://www.mysql.com)
- W3Schools Online Web Tutorials- [www.w3schools.com](http://www.w3schools.com)
- PHP Manual - [www.php.net](http://www.php.net)



## Books

- Sams Teach Yourself Ajax JavaScript and PHP All in One; Phil Ballard and Michael Moncur;
- Sams Publishing; 2010
- JavaScript Phrasebook; Christian Wenz; Sams Publishing; 2007
- PHP and MySQL Web Development, 4/E; Luke Welling and Laura Thomson; AddisonWesley Professional; 2009
- JavaScript for Programmers Paul J. Deitel and Harvey M. Deitel; Prentice Hall; 2009